

Article

Enhanced Bio and Cultural Tourist Navigation and Guiding Application System for Android OS Smartphones, Supporting Augmented Tour Operating Experience

George Tsamis ^{1,2,*}, Giannis Vassiliou ¹, Athanasios Malamos ¹, Alexandros Garefalakis ³, Maria Rousaki ⁴, Aris Papakostas ², Haralampos Tzagkarakis ¹, Charles D. White ⁵, Evangelos Tzirakis ⁶ and Nikos Papadakis ¹

¹ Department of Electrical and Computer Engineering, Hellenic Mediterranean University (HMU), 71410 Heraklion, Greece; vassil@hmu.gr (G.V.); amalamos@hmu.gr (A.M.); hartzag@hmu.gr (H.T.); npapadak@hmu.gr (N.P.)

² Department of Computer Science, Metropolitan College, 41005 Larisa, Greece; apapakostas@mitropolitiko.edu.gr

³ Department of Business Administration and Tourism, Hellenic Mediterranean University (HMU), 71410 Heraklion, Greece; agarefalakis@hmu.gr

⁴ Department of Accounting and Finance, Hellenic Mediterranean University (HMU), 71410 Heraklion, Greece; mrousaki@windowslive.com

⁵ Department of Technical Presales, Westpole, 1800 Vilvoorde, Belgium; c.white@westpole.be

⁶ Department of Mechanical Engineering, Hellenic Mediterranean University (HMU), 71410 Heraklion, Greece; vtzirakis@hmu.gr

* Correspondence: gtsamis@hmu.gr; Tel.: +30-2810-379196

Abstract

In this publication we investigate in depth the design, architecture and implementation of a bio and cultural guiding system with augmented capabilities. Our platform will be able to provide a flexible and user-friendly application for smart mobile devices with Android OS, which will be able to provide to the user the ability to discover nearby places of interest such as museums, archeological sites, religious sites, natural beauty sites, etc. In addition, our application is able to offer users an enhanced, comprehensive and augmented tour experience, based on visual and audio smartphone services, using asynchronous and real-time user-server communication mechanisms, thus eliminating the need for a human tour guide operator in cases of a remote area, guiding service cost or unavailable time slot. The proposed platform integrates visual overlays, audio narration, Geolocation services, and cloud-based data management to enable users to explore cultural, historical, and natural points of interest without the need for a human tour guide. A modular, layered system architecture is developed, combining Firebase Realtime Database, RESTful web services, OpenStreetMap-based navigation, and Android-native components to ensure scalability, flexibility, and real-time responsiveness. UML modeling, including class and sequence diagrams, is employed to formally describe system structure and behavior, while a mathematical data model validates the consistency of the underlying database schema. The implementation demonstrates how AR guiding systems can enhance spatial understanding, accessibility, and user engagement while supporting sustainable tourism practices and efficient destination management. The results indicate that the proposed solution is technically feasible, user-centered, and adaptable to diverse bio-cultural contexts, contributing to the advancement of intelligent, inclusive, and sustainable digital tourism platforms.

Keywords: design; architecture; implementation; smart; augmented; bio and cultural;

Academic Editor: Jun Kong

Received: 10 October 2025

Revised: 13 July 2026

Accepted: 23 July 2026

Published: 6 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

guiding system; visual and audio smartphone services

1. Introduction

Tourism in the twenty-first century represents a multidimensional phenomenon that integrates culture, environment, and technology. As global mobility increases, visitors increasingly seek authentic, educational, and sustainable travel experiences that foster a deeper connection with local heritage and natural environments. At the same time, cultural institutions and destination managers face the challenge of protecting fragile ecosystems and heritage assets while ensuring accessibility and engagement for diverse audiences. Within this context, digital transformation has emerged as a powerful enabler of innovation in the tourism sector, with mobile technologies and augmented reality (AR) applications leading this evolution [1–3].

The widespread adoption of smartphones equipped with high-resolution cameras, sensors, and global positioning systems (GPS) has made it possible to develop intelligent and interactive tour guiding solutions. These systems are capable of providing context-aware, multimodal, and personalized information directly to users in real time. Traditional guiding methods such as printed materials, signage, or human guides often face limitations related to language, availability, and scalability. Conversely, mobile and AR-based platforms overcome these barriers by merging virtual information layers with real-world environments, thereby enhancing spatial understanding, engagement, and inclusivity [4–6].

Recent developments in artificial intelligence (AI), cloud computing, and geospatial data services have further expanded the potential of mobile guiding applications. Such systems can now integrate biological, cultural, and environmental datasets to deliver adaptive recommendations and immersive storytelling experiences that promote both learning and sustainability. In this context, “bio-cultural” tourism integrates natural and cultural elements, emphasizing respect for biodiversity and local traditions as complementary dimensions of sustainable development [7].

Recent advancements in digital and immersive technologies are reshaping bio-cultural tourism by enabling richer, more engaging, and personalized visitor experiences. The global digital cultural tourism market, encompassing technologies such as augmented reality (AR), virtual reality (VR), mixed reality (MR), and artificial intelligence (AI), was valued at approximately USD 5.2 billion in 2023 and is projected to grow to around USD 18.3 billion by 2032 at a CAGR of about 15.4%, reflecting rapid adoption and investment in these technologies across heritage and cultural sites worldwide [8,9]. Immersive digital solutions enhance accessibility and interaction by overlaying historical narratives, contextual multimedia, and interpretive content onto physical environments, significantly extending traditional tourism formats with interactive storytelling, 360° visualizations, and mobile-based experiences. This growth is driven by declining hardware costs, improved mobile device capabilities, and increasing traveler demand for interactive and meaningful cultural encounters that merge the digital with the physical.

The research gap addressed in this manuscript was identified through a systematic review of the existing literature and an analysis of current technological solutions used in cultural and bio-cultural tourism. While prior studies extensively explore the use of augmented reality, mobile applications, and immersive technologies for enhancing visitor engagement, most existing solutions focus on isolated functionalities such as visualization, museum-based interpretation, or generic navigation, often lacking integrated, end-to-end guiding frameworks tailored to bio-cultural contexts. Furthermore, limited attention has been given to lightweight, scalable mobile platforms that combine aug-

mented reality, real-time navigation, cloud-based data management, and multimodal interaction while remaining accessible across a wide range of device capabilities. In response to this gap, the objective of this research is to design, implement, and evaluate an enhanced bio-cultural tourist navigation and guiding system that integrates AR-based guidance, location-aware services, and cloud-supported content delivery within a unified Cloud Android platform.

In details, the present work introduces an enhanced bio and cultural tourist navigation and guiding application system for Android OS smartphones. The proposed platform combines augmented reality visualization, audio narration, and real-time data exchange through Firebase cloud infrastructure to deliver a comprehensive, self-guided tour experience. Using technologies such as OpenStreetMap for Geolocation and multimedia overlays, the system allows users to explore museums, archeological sites, religious landmarks, and natural attractions without the need for a human guide.

Beyond its technological innovation, the proposed system contributes to sustainable tourism practices by reducing dependence on resource-intensive tour operations and promoting equitable access to cultural and environmental knowledge. It can also support local authorities, educators, and policymakers in designing smart, data-driven strategies for heritage preservation and tourism management. Ultimately, this research aims to bridge the gap between technological capability and cultural experience, demonstrating how digital tools can foster responsible, immersive, and inclusive exploration of our shared bio-cultural heritage.

The remainder of this paper is organized as follows: Section 2 reviews related work on AR-based guiding systems and mobile cultural applications. Section 3 examines the economic and social role of tour guide operators. Section 4 presents the technological background of augmented reality systems. Section 5 outlines the Android OS environment, while Section 6 describes the architecture and design of the proposed platform. Finally, Section 7 details the implementation of the designed platform, Section 8 concludes with key findings showing the implementation results and evaluating the system, and Section 9 concludes this publication by evaluating the final results.

2. Related Work

The development of smart bio-cultural guiding systems intersects several established research domains, including augmented reality (AR) guiding technologies, artificial intelligence driven personalization, and mobile applications for tourism and cultural heritage.

2.1. Augmented Reality and Guiding Systems

Augmented reality has evolved from experimental prototypes in the 1960s to robust mobile platforms, enabled by advances in computer vision, sensor fusion, and mobile hardware [10,11]. Early studies focused on AR for industrial training and navigation, demonstrating that context-aware overlays could reduce cognitive load and improve task efficiency [12,13]. More recently, AR guiding systems have been applied to museums, archeological sites, and urban navigation, enabling immersive visitor experiences [14,15].

In cultural heritage, AR has proven particularly effective in overlaying digital reconstructions on ruined monuments, presenting multilingual interpretive content, and guiding visitor flows in crowded sites [16]. For example, museum AR applications allow users to view 3D models of artifacts and interact with reconstructions of historical events [17]. However, these systems often face challenges such as device dependency, lack of personalization, and limited adaptability to diverse user contexts. Many rely on marker-based tracking or predefined scenarios, which constrain scalability and flexibility in real-world deployments [18].

2.2. Mobile Tourism and Cultural Heritage Applications

The proliferation of smartphones has significantly expanded the scope of guiding systems. Mobile applications now integrate Geolocation, QR code scanning and Bluetooth beacon technologies to support tourists in museums, heritage sites, and natural landscapes [19,20]. Platforms such as Google Arts & Culture and city-specific tourism apps have demonstrated the potential for mobile-first guiding, offering maps, curated tours, and multimedia content [21].

Nevertheless, most mobile tourism applications emphasize content delivery rather than intelligent adaptation. They present standardized routes and information, often replicating the role of static guidebooks rather than leveraging mobile devices' interactive and sensor-rich capabilities [22]. Security and data protection also remain underexplored, particularly when handling sensitive information such as biometric authentication or location tracking [23].

Another limitation of existing mobile cultural applications is the lack of integration across modalities. Visual AR overlays, audio narration, and Geolocation-based prompts are often implemented separately, without a unified architecture capable of delivering seamless, multimodal experiences [24]. This fragmentation undermines the potential for inclusive and accessible guiding, especially for users with disabilities or diverse learning preferences [25].

2.3. Current State of Research on Technologies for Bio and Cultural Tourism

A growing body of research explores the integration of digital technologies, particularly augmented reality (AR) and virtual reality (VR), in cultural tourism to enhance interpretation, engagement, and sustainability at heritage sites. Narrative and systematic reviews show that mobile AR applications are widely developed and evaluated for cultural heritage settings, with positive impacts on learning outcomes, visitor engagement, and user experience, while highlighting design and evaluation approaches for these technologies [26]. Meta-analytic studies further confirm that immersive technologies significantly affect tourists' experiences by helping visitors recognize and interact with heritage elements, supporting cognitive and affective dimensions of the tourism experience [27]. Empirical research comparing AR and VR applications has shown that both technologies can generate immersive cultural experiences with distinct emotional and experiential effects [28]. Literature also identifies how AR and VR can support sustainable tourism strategies by enhancing preservation efforts and stakeholder engagement in small and cultural tourism contexts [29]. Additionally, studies emphasize challenges and adoption dynamics of these technologies in urban heritage tourism, suggesting that effective communication and management frameworks are necessary to balance authenticity and technological innovation [28]. Collectively, research highlights that digital reality technologies are becoming crucial in cultural tourism research and practice, fostering richer, technology-mediated cultural engagement while also informing sustainable and user-centered design frameworks.

2.4. Positioning the Present Work

In summary, related work demonstrates the potential of AR for immersive guidance and mobile platforms for accessible cultural engagement. Yet, gaps persist in integrating these strands into a unified, adaptive system. Prior approaches are often limited by:

- **Lack of real-time personalization:** few systems dynamically adapt based on user behavior and contextual signals;
- **Fragmented architectures:** most solutions focus on a single modality (e.g., AR overlays) without combining multimodal services;

- **Scalability concerns:** reliance on heavy computation or predefined markers restricts usability in diverse, real-world settings;
- **Privacy and ethical considerations:** few studies adequately address the implications of behavioral data collection in cultural contexts.

The present research advances the field by designing and implementing a modular Android-based platform that integrates Firebase cloud services and multimodal user interaction (visual, audio, AR). By evaluating system performance and demonstrating real-world feasibility, this work contributes a step toward next-generation guiding systems that are not only technically robust but also socially and culturally sustainable.

3. The Tour Guide Operator Field and Its Impact on the Economy

3.1. Overview of the Tour Guide Operator Industry

Tour guide operators form an essential segment of the tourism industry [30–32]. They plan, organize, and deliver experiences for travelers, ranging from historical city walks to wildlife safaris and cultural immersion programs. Tourism is one of the largest and fastest-growing industries in the world [33–35], and within this expansive sector, tour guide operators play a vital role. Tour guide operators are the facilitators of experience [36], culture [37,38], and learning [39,40]. They serve as the interface between the traveler and the destination, helping bridge language barriers, cultural gaps, and logistical challenges. The profession of tour guiding is as old as travel itself, evolving alongside technological advancement, changing traveler expectations, and global mobility trends.

Their role extends beyond merely showing places as they interpret culture, manage logistics, ensure safety, and enhance the visitor experience. The field of tour guiding is highly varied [41,42]. It includes freelance guides, small local operators, and large multinational companies. These operators work with tourists directly or through partnerships with travel agencies, hotels, and transport providers. Tour guide operators enhance the overall travel experience by providing curated, informed, and often entertaining tours that bring history, art, culture, and nature to life. However, their contribution extends beyond the tourist experience. They are major contributors to the economic ecosystems of destinations [43,44]. Through direct employment, support of ancillary industries, and enhancement of local economies, tour guide operators serve as economic multipliers.

The economic impact of tour guide operators is significant but often underappreciated [44–46]. They generate direct income through their services, create employment opportunities, stimulate demand for local goods and services, and help distribute tourism revenue more evenly, especially in rural or developing regions. In this paper, we will try to explore the tour guide operator field in detail, describing its structure, functions, and the wide-ranging economic impacts it has on local, regional, and national economies.

3.2. The Structure of the Tour Guide Operator Industry

Typically, the industry has a large number of small and micro-enterprises, especially in developing regions. In advanced economies, some large tour operators consolidate services to offer scaled packages, but local guides still play a vital role in delivering the personal, authentic aspect of the experience.

At the local level, individual guides typically specialize in specific cities, heritage sites, or natural areas, offering in-depth, place-based knowledge and personalized experiences. Inbound tour operators function at a larger scale, organizing and coordinating guided experiences for international visitors, often acting as intermediaries between travelers, local guides, accommodation providers, and transport services. Outbound tour

operators, by contrast, design and sell packaged tours for domestic travelers visiting foreign destinations, managing logistics across borders and ensuring consistency of service. In addition, specialized tour operators focus on niche markets such as adventure tourism, eco-tourism, culinary tours, or cultural heritage routes, tailoring experiences to specific interests and traveler profiles. Together, these different actors form a layered industry structure in which small local enterprises coexist with larger organizations, enabling both authenticity and scalability within the global tourism ecosystem.

3.3. Roles and Responsibilities of Tour Guide Operators

Tour guide operators play a vital role in creating enjoyable, meaningful, and responsible travel experiences [47,48]. They design well-balanced itineraries that align traveler interests with safety, accessibility, and local engagement, while supporting community businesses. Beyond logistics, guides serve as interpreters and storytellers who bring destinations to life through history, culture, and narrative, fostering authentic connections and cultural appreciation. They also manage complex logistics and safety measures, demonstrating adaptability and preparedness in changing conditions. As cultural mediators and customer service professionals, tour guides bridge language and cultural gaps, handle group dynamics, and ensure positive interactions between visitors and locals. Their professionalism and hospitality make them key ambassadors who shape travelers' perceptions and encourage sustainable tourism.

3.4. Direct Economic Impacts

The most obvious economic impact of tour guide operators is direct income generation [49]. Tourists pay fees for guided tours and these revenues flow to local guides, companies, and support staff. For many destinations, especially in developing countries, guiding services provide one of the most accessible ways for local people to participate in the tourism economy.

Tour guide operators generate direct economic benefits primarily through the income they earn from providing services to tourists. These earnings support not only the guides themselves but also a wide range of related roles, including office staff, coordinators, drivers, and translators. The industry creates formal and informal employment opportunities, particularly in regions where other job sectors may be limited. In developing countries, guiding often serves as a key entry point into the formal economy, offering livelihoods to individuals with knowledge of local culture and language skills.

3.5. Indirect and Induced Economic Impacts

Beyond their direct revenues, tour guide operators generate substantial indirect and induced economic impacts [50,51] through a multiplier effect that extends tourism benefits across the wider economy. Indirect impacts arise as operators purchase goods and services from local supply chains, including transportation, fuel, accommodation, meals, entrance tickets, and equipment, thereby stimulating activity in multiple supporting sectors. Induced impacts occur when income earned by guides and associated staff is spent within local communities on everyday goods and services, further reinforcing economic circulation and employment. This multiplier effect is particularly significant in rural, agricultural, or economically marginalized regions, where tourism-related income can represent a critical source of financial stability. Additionally, tour guiding activities often contribute to seasonal economic continuity by sustaining demand during periods when other industries experience downturns. Collectively, these indirect and induced effects amplify the overall economic contribution of tour guide operators, supporting infrastructure development, local business resilience, and long-term regional economic diversification.

3.6. Cultural, Social and Economic Benefits

Tour guide operators generate important cultural, social and economic benefits that extend beyond monetary gains, contributing to community resilience and sustainable development. By interpreting and promoting local heritage, traditions, and historical narratives, guides play a key role in cultural preservation and raise awareness about the value of protecting tangible and intangible cultural assets. The demand for authentic tourist experiences often leads to the revitalization of traditional crafts, performances, and storytelling practices, creating new income opportunities while reinforcing cultural identity. Additionally, tour guiding fosters community engagement by enabling local residents to participate directly in tourism activities through cultural programs, local cuisine, homestays, and artisanal markets. These interactions help ensure that tourism revenues are more equitably distributed within communities, strengthening social cohesion and empowering local populations. As a result, tour guide operators function not only as economic actors but also as cultural mediators who support inclusive growth and the long-term sustainability of local cultural economies.

3.7. Environmental and Regional or National Economic Impacts

Well-managed tourism can help fund and justify conservation areas. This creates jobs while maintaining the ecosystem services, like clean water, biodiversity, and climate regulation that underpin long-term economic health [52]. Poorly managed tour guiding, however, can have the opposite effect, causing overcrowding, habitat destruction, and cultural commodification. Hence, professional, trained, and responsible operators are critical for maximizing positive impacts.

Tour guide operators play a vital role in regional development by helping distribute tourism activity beyond major cities and popular hubs. By designing and leading tours in rural, remote, or underserved areas, guides encourage travelers to explore less visited destinations. This helps spread tourism revenue more evenly, reducing pressure on overcrowded urban centers while bringing new sources of income to regions that might otherwise have limited economic opportunities. The resulting investment in local businesses, infrastructure, and community services can reduce regional inequalities and improve quality of life for residents.

At the national level, the tourism sector, including guiding services, can make a substantial contribution to Gross Domestic Product (GDP). In some countries, particularly small island or developing nations, tourism can account for a significant share of GDP, sometimes exceeding 50%. Tour guide operators are central to this contribution because they enhance the visitor experience, encourage longer stays, and boost overall spending. Their services make destinations more appealing and accessible to international tourists, increasing foreign exchange earnings that strengthen national economies and help balance trade debt.

Tour guiding also serves as an important employment strategy, especially in labor-rich but capital-poor countries. The industry offers accessible, low-barrier job opportunities for people with local knowledge and language skills, supporting formal employment where other options may be difficult to be achieved. Governments often recognize this potential by including tourism development in national employment and poverty reduction plans. Beyond direct employment for guides themselves, the sector supports many related jobs in hospitality, transport, retail, and cultural services, making it a cornerstone of inclusive economic growth strategies.

Finally, tour guides act as ambassadors for their countries, shaping how visitors perceive the nation's culture, people, and values. Positive experiences delivered by skilled guides can improve a country's reputation abroad, boosting communication and encouraging diplomatic goodwill. Satisfied visitors are more likely to recommend des-

tinations to others or return in the future, creating a cycle of sustained demand. In addition, quality tourism experiences attract investors interested in developing hotels, transport networks, and other infrastructure, further amplifying the sector's long-term economic impact at a national scale.

3.8. Challenges Facing Tour Guide Operators

Despite their economic and cultural importance, tour guide operators face a range of structural and external challenges that threaten the stability and sustainability of the sector. Seasonality remains a major issue, as demand fluctuates significantly throughout the year, leading to income insecurity and workforce instability during low seasons. Informality is another persistent challenge, particularly in developing regions, where unlicensed or unregulated guiding undermines service quality, fair competition, and tax revenues. Training gaps further limit sector performance, as insufficient professional development can negatively affect safety standards, interpretive quality, and environmental responsibility. Over-tourism poses a growing threat in popular destinations, where excessive visitor numbers strain infrastructure, damage cultural and natural resources, and reduce resident acceptance of tourism activities. Finally, external and unpredictable events such as pandemics, political instability, and natural disasters can abruptly disrupt travel demand, exposing the sector's vulnerability and highlighting the need for stronger policy support, regulation, and resilience planning.

3.9. The Future of the Tour Guide Operator Field

The future of the tour guide operator field is expected to be shaped by increasing demand for personalized, authentic, and meaningful travel experiences [53]. As tourists become more experienced and discerning, they seek deeper engagement with local cultures, traditions, and environments rather than generic sightseeing. This trend is pushing tour operators to specialize in niche markets like culinary tourism, adventure travel, heritage trails, and eco-tourism. Guides will need to develop richer storytelling skills, in-depth local knowledge, and the ability to customize experiences to individual interests, making their role even more central to delivering high-quality tourism products.

Technological innovation will also transform the field in multiple ways. Online booking platforms, review sites, and mobile apps have already revolutionized how travelers choose and book tours, increasing competition but also expanding market reach for local operators. Virtual and augmented reality tools may enhance pre-trip planning or even enable remote tours. At the same time, operators will increasingly use digital marketing and social media to connect with customers and build their brand. Tour guides themselves may use technology on tours to provide richer content, translations, or interactive experiences. Embracing these tools will be essential for operators who want to stay relevant and competitive in a rapidly changing industry.

4. Augmented Reality Guiding Systems

4.1. Introducing Augmented Reality Guiding Systems

A guiding system refers to any system designed to help users navigate, learn about, or interact with physical spaces or objects [54,55]. Classic guiding systems include maps, audio tour guides, signage, and even human guides. In modern contexts, guiding systems can be interactive apps, GPS navigation tools, or museum audio devices. Guiding systems aim to improve user orientation, provide relevant information at the right time and place, reduce confusion, and enhance the overall experience of moving through or learning about space, events, history, heritage, culture, etc.

Augmented Reality (AR) is a technology that superimposes computer-generated content onto a user's perception of the real world [56,57]. Unlike Virtual Reality (VR), which immerses the user in a completely synthetic environment, AR blends digital elements with the real environment in real time. AR relies on sensors, cameras, displays, and software to detect and interpret the user's environment and position, enabling accurate placement of digital objects. This real-time integration creates an "augmented" experience in which users can interact with both real and virtual elements seamlessly.

By visually embedding guidance such as directional arrows, labels, or indicators within the physical environment, these systems significantly improve routing and reduce user confusion, particularly in complex or unfamiliar spaces. Information is delivered precisely when and where it is needed, enabling users to access contextual knowledge without diverting attention from their surroundings. Many AR guiding systems also support hands-free operation through audio cues or head-mounted displays, allowing users to receive instructions while maintaining physical engagement with their environment. Furthermore, AR enables a high degree of personalization, adapting content to user preferences, language, expertise level, or location. Collectively, these features increase user engagement by creating immersive, intuitive, and interactive experiences that surpass the effectiveness of traditional static media such as maps, signs, or brochures.

4.2. Core Technologies of AR Guiding Systems

AR guiding systems rely on a complex interplay of hardware, software, and algorithmic components [58,59]. The effectiveness of these systems depends not only on the quality of AR content but also on how well the underlying technologies work together to provide a seamless and context-aware user experience such as cameras, inertial sensors, GPS receivers, microphones, and displays capture environmental data and user motion, forming the physical interface between the digital and real worlds. Software platforms and development frameworks process this sensor data, manage system logic, and support the creation and deployment of AR content across devices. Central to system performance are tracking and registration techniques, which determine the user's position and orientation and ensure that virtual elements are precisely aligned with physical objects or locations. Rendering engines generate and display these digital overlays with the appropriate scale, lighting, and perspective, maintaining visual coherence and realism. Finally, user interface and interaction mechanisms enable intuitive communication between the user and the system through touch, voice, gestures, or multimodal input, ensuring that guidance is delivered effectively without overwhelming or distracting the final user. Together, these core technologies form an integrated pipeline that allows AR guiding systems to operate reliably in dynamic real-world environments.

4.3. Types of AR Guiding Systems

AR guiding systems come in many forms, tailored to different use cases, environments, and user needs [60]. They vary in terms of the senses they engage (visual, auditory, haptic), the devices they run on, and the ways they deliver guidance.

Augmented Reality (AR) guiding systems use multiple sensory channels (visual, auditory, and haptic) to deliver context-aware information that enhances navigation, comprehension, and situational awareness [61]. Visual overlays are a core feature, projecting cues like arrows, labels, or instructions directly into the user's view to reduce cognitive load and improve decision making, though they must remain clear, aligned, and uncluttered to maintain trust. Audio AR complements visuals through spatial cues and spoken guidance, supporting hands-free navigation and accessibility, especially in safety-critical or visually demanding contexts. Haptic feedback adds a tactile dimension with vibrations or pulses that discreetly convey direction or confirmation, enriching in-

teraction without overwhelming visual or auditory senses. Together, these multimodal guidance systems integrate sensory inputs to create intuitive, inclusive, and adaptive user experiences.

4.4. Fields of Applications for AR Guiding Systems

Augmented Reality guiding systems are applied across a wide range of domains where spatial understanding, real-time information, and contextual assistance are critical [62]. In tourism and cultural heritage, AR enhances visitor experiences by overlaying historical reconstructions, interpretive narratives, and navigation cues onto physical sites, allowing users to engage more deeply with destinations. Museums and exhibitions employ AR to enrich static displays with interactive digital content, animations, and multilingual explanations, increasing accessibility and learning outcomes. In industrial maintenance and training, AR guiding systems support workers by projecting step-by-step instructions directly onto equipment, reducing errors and improving efficiency. Navigation in indoor and outdoor environments, such as airports, hospitals, and urban spaces, benefits from AR-based navigation that provides intuitive directional guidance without reliance on abstract maps. In education and professional training, AR facilitates experiential learning by visualizing complex concepts, while in healthcare it supports clinical procedures, rehabilitation, and patient guidance. Retail and customer experience applications use AR to guide consumers through stores, visualize products, and deliver personalized recommendations, demonstrating the versatility of AR guiding systems across both public and commercial contexts.

4.5. Design Considerations for AR Guiding Systems

Designing effective AR guiding systems is far more than just overlaying graphics on the real world [63–65]. It requires careful attention to user needs, technological limits, safety, and usability. A well-designed AR guiding system minimizes user frustration, enhances task performance, and delivers value without causing confusion or distraction.

Augmented reality guiding systems rely on strong user experience (UX), interface, and interaction design to seamlessly integrate digital content with the physical world, ensuring clarity, safety, and usability. Effective AR design prioritizes intuitive visuals, accessible controls, and context-aware guidance that adapt to diverse environments, lighting conditions, and user abilities. Safety and situational awareness are paramount to inform without distracting or obscuring real-world hazards. Accessibility and inclusivity ensure AR benefits users of all abilities through multimodal input and output options, while cultural and social sensitivity makes AR experiences relevant and respectful across contexts.

4.6. Benefits and Technical Challenges

4.6.1. Benefits of AR Guiding Systems

Augmented reality guiding systems offer a wide range of transformative benefits that fundamentally change how people access, interpret, and use information in the real world [66–68]. By enhancing spatial understanding, AR helps users perceive relationships between locations, objects, and routes more intuitively than traditional maps or textual instructions. The provision of contextual, just-in-time information reduces cognitive load, as guidance appears precisely when and where it is needed, improving decision making and task performance. AR guiding systems also increase user engagement and motivation through immersive and interactive experiences that encourage exploration and sustained attention. Accessibility is improved through multimodal interaction, combining visual, audio, and sometimes haptic feedback to accommodate users with diverse abilities and preferences. Hands-free operation further enhances usability in

scenarios where manual interaction is impractical or unsafe. Finally, AR systems enable personalization by adapting content, language, and guidance based on user context and preferences, leading to more efficient, inclusive, and user-centered experiences across a variety of application domains.

4.6.2. Challenges of AR Guiding Systems

Despite their advantages, augmented reality guiding systems face several technical, usability, and societal challenges that affect their reliability and large-scale adoption. From a technical perspective, achieving accurate and stable tracking remains difficult in dynamic real-world environments due to variations in lighting conditions, sensor noise, and limited positioning accuracy, particularly in outdoor or crowded settings. AR applications also place high demands on device hardware, leading to increased battery consumption and performance constraints on mobile platforms. Usability challenges arise when visual overlays become cluttered, misaligned, or distracting, potentially increasing cognitive load or compromising user safety. Cost and infrastructure requirements, including specialized hardware, content creation, and continuous system maintenance, further limit widespread deployment. Effective AR guidance also relies on dynamic, location-specific content that must be continuously managed, updated, and localized without compromising accuracy or security [69,70]. Additionally, privacy and security concerns emerge from the collection of sensitive data such as location, visual feeds, and user behavior, raising ethical and regulatory issues. Social acceptance and accessibility gaps also persist, as not all users are comfortable with AR interfaces or have access to compatible devices. Environmental constraints, such as poor connectivity or adverse weather conditions, further complicate reliable operation, highlighting the need for robust, inclusive, and privacy-aware design strategies.

5. The Android Operating System

5.1. *Introducing Android OS*

The Android operating system is the world's most widely used mobile platform [71–73]. Developed by Google, Android powers billions of devices globally, from smartphones and tablets to TVs, cars, and wearables. Its open-source nature, broad device support, and customizable user experience have made it a dominant force in consumer electronics. Android's development began with Android Inc. in 2003, later acquired by Google in 2005. In 2008, the first commercial Android device launched, marking the start of an extraordinary growth trajectory that transformed the mobile industry.

Android Inc. was founded in October 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White. Their initial goal was to build an advanced operating system for digital cameras, but they quickly pivoted to mobile phones. In 2005, Google acquired Android Inc., recognizing the potential to disrupt the mobile space. Under Google, Android was positioned as an open-source platform to counter the dominance of proprietary systems like Symbian, BlackBerry OS, iPhone OS and Windows Mobile. Google formed the Open Handset Alliance (OHA) in 2007, a consortium of hardware manufacturers, software companies, and carriers committed to advancing open standards for mobile devices. The early OS featured basic smartphone capabilities, integration with Google services, and support for third-party apps via the Android Market (now Google Play Store). As of 2025, Android powers over 70% of smartphones worldwide.

5.2. *Android's Key Features and Capabilities*

Android's features and capabilities are central to its global success, offering a highly customizable user experience that distinguishes it from competing platforms [74]. The

user interface is designed to be flexible, supporting multi-touch gestures, dynamic widgets, and home screen personalization through launchers and themes. Multitasking and connectivity are additional pillars of Android's design. Android's architecture supports true multitasking through background services, foreground notifications, and split-screen modes, allowing users to perform multiple tasks seamlessly. Connectivity options are equally comprehensive, with support for Wi-Fi, Bluetooth, NFC, and 5G enabling diverse use cases ranging from mobile payments to high-speed data transfers. Google service integration further enhances Android's appeal, offering deep connections with widely used applications such as Gmail, Maps, and the Google Play Store.

5.3. Main Advantages and Challenges

One of Android's most significant advantages lies in its open-source nature, which allows device manufacturers, developers, and even users to freely modify and distribute the operating system [75]. This openness has fostered a vast, diverse hardware ecosystem, ranging from budget smartphones to premium devices and specialized platforms like smart TVs and automotive systems. Consequently, Android has achieved a dominant global market share, especially in emerging markets where cost-effective hardware is crucial.

Android also excels in terms of flexibility and user customization. Unlike more closed platforms, Android permits users to change launchers, modify UI elements, install third-party apps from outside the Play Store, and personalize virtually every aspect of their device. From a technological perspective, Android supports a wide array of hardware features and connectivity standards, including 5G, NFC, and biometric sensors. Its multitasking capabilities and scalable UI makes it suitable for various device types, including foldables, tablets, and wearables.

Despite its numerous strengths, Android faces persistent challenges with the foremost being fragmentation. Due to the diverse range of manufacturers and devices using Android, software updates are often delayed or inconsistent. Many devices continue running outdated versions of the OS, limiting access to the latest features and leaving users vulnerable to security risks.

6. System Architecture and Design

6.1. Personal Tour Guide Application and Web Platform Architecture

The success of any digital platform, whether a website or a smartphone application, relies heavily on its underlying system architecture and design. System architecture provides the structural framework that defines how components interact, while design focuses on the usability, efficiency, and adaptability of the system. Breaking the system into smaller, independent components makes it easier to develop, test, and maintain [76,77]. Moreover, Mobile applications follow different architectural patterns based on platform (iOS, Android, or cross-platform) [78,79]. On the other hand, design considerations play a critical role in determining the effectiveness and longevity of websites and smartphone applications, as they directly influence usability, performance, and user satisfaction [80].

Based on the points above, our platform consists of three parts as shown in Figure 1. Three basic packages, of which the first one, called External Services will include all necessary source code and APIs to communicate with remote services. The second one, called Libraries, will include all important libraries that are crucial for our application. The third main part of the hierarchy is the Android application itself.

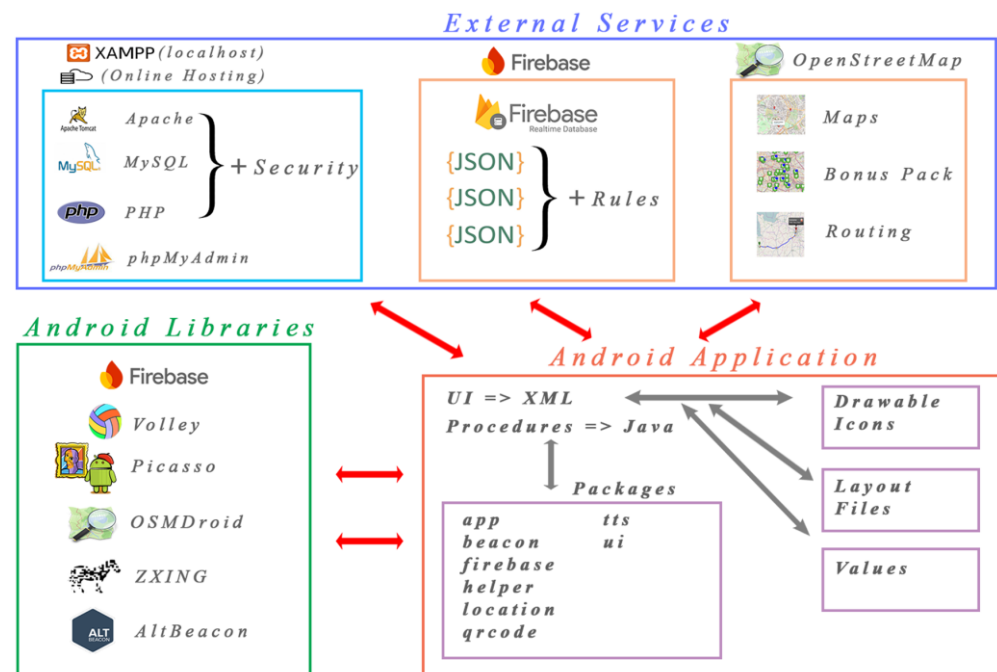


Figure 1. Personal Tour Guide Platform's Architecture.

A component-based UML architecture of the Personal Tour Guide Application and Web Platform, illustrating the separation between the Android application layer, middleware libraries, and external services, as well as their interactions are shown in Appendix A in Appendix A.1, Figure A1. The proposed Personal Tour Guide platform is modeled as a layered, component-based architecture, designed to clearly separate concerns related to presentation, application logic, data management, and external service integration. This architectural approach improves scalability, maintainability, and extensibility, while also facilitating formal modeling using UML diagrams. The External Services layer represents all remote systems that provide data persistence, geospatial services, and interoperability. From a modeling perspective, this layer functions as the infrastructure and data provider tier of the system. The Android Application layer represents the core functional model of the system and is responsible for user interaction, domain logic, and orchestration of services.

The proposed enhanced bio and cultural tourist navigation and guiding platform has been designed as a modular, cloud-connected, multi-layer software architecture that integrates mobile computing, geospatial services, cloud databases, multimedia content delivery, and augmented tour guiding functionalities. The primary objective of the architecture is to provide visitors with a self-guided digital tourism experience capable of delivering contextual information, navigation assistance, multimedia content, and interactive services in real time without requiring the physical presence of a professional tour guide.

From a software engineering perspective, the system follows a layered architecture model that separates presentation concerns, business logic, data management, and infrastructure services. Such separation of concerns improves maintainability, scalability, extensibility, and fault isolation while simplifying future integration of additional services such as artificial intelligence recommendation engines, machine learning analytics, multilingual support mechanisms, and advanced augmented reality functionalities.

The proposed architecture consists of four primary layers:

1. Presentation Layer;
2. Application Logic Layer;

3. Data Access Layer;
4. Infrastructure and External Services Layer.

This layered organization, visualized in Appendix A Appendices A.4 and A.5 (Figures A4 and A5), allows each subsystem to perform a clearly defined responsibility while communicating with adjacent layers through well-defined interfaces and service abstractions. Moreover, it illustrates the overall architecture of the proposed platform and the interactions between mobile components, middleware services, cloud databases, and external providers.

The **Presentation Layer** represents the user-facing component of the system and is implemented as a native Android application. This layer is responsible for all interactions between the tourist and the platform, providing intuitive access to navigation services, cultural information, multimedia content, and personalized guidance.

The presentation layer includes:

- Login and Registration interfaces;
- Main navigation dashboard;
- Category browsing interfaces;
- Guide information screens;
- Multimedia presentation views;
- Interactive map interfaces;
- QR code interaction screens;
- Biometric authentication interfaces;
- Augmented guiding visualizations;
- Audio narration controls.

The design philosophy of the presentation layer emphasizes simplicity, accessibility, and ease of use. Since the target audience includes visitors with different technological backgrounds, the interface has been designed to minimize user effort while maximizing information accessibility. The graphical user interface employs Android-native components and follows Material Design principles to ensure consistency across different smartphone devices and Android OS versions.

In addition, the presentation layer supports multimodal interaction by combining visual, audio, and location-based information delivery. This approach enhances accessibility for users with different preferences and contributes to a richer and more immersive tourism experience.

The **Application Logic Layer** constitutes the core intelligence of the proposed platform. This layer orchestrates all functional operations of the system and coordinates communication between the user interface, cloud services, and external data providers.

Its primary responsibilities include:

- User authentication management;
- Session management;
- Guide selection and retrieval;
- Category filtering;
- Route generation;
- GPS position processing;
- QR code processing;
- Bluetooth beacon detection;
- Audio narration management;
- Multimedia synchronization;
- Geolocation monitoring;
- Data validation;
- User preference management.

This layer operates as the central controller of the application, ensuring that all services work together seamlessly. Whenever a user requests information about a cultural site, monument, museum, natural landmark, or archeological location, the application logic layer coordinates data retrieval, navigation calculations, multimedia presentation, and user interaction processes.

The modular design of this layer allows independent development and maintenance of individual subsystems. For example, modifications to the QR code subsystem do not affect the routing subsystem, while updates to the Text-to-Speech functionality remain isolated from database communication modules.

Such modularity significantly improves software maintainability and facilitates future system expansion.

The **Data Access Layer** acts as an intermediary between the application logic and the underlying data repositories. Its purpose is to abstract database operations and external service communications from the rest of the application.

This layer manages:

- Firebase communication;
- REST API communication;
- User profile retrieval;
- Guide information retrieval;
- Content synchronization;
- Data caching;
- Remote data updates;
- Query optimization.

By isolating data operations within a dedicated layer, the system achieves greater flexibility and independence from specific storage technologies. Future migration from Firebase Realtime Database to Cloud Firestore, MongoDB, PostgreSQL, or another cloud-based solution could be accomplished with minimal impact on higher software layers.

The Data Access Layer also provides mechanisms for handling communication failures, network interruptions, and data synchronization issues, thereby improving system reliability under real-world operating conditions.

The **Infrastructure Layer** contains all external systems and third-party services required by the platform. These services provide persistent storage, mapping capabilities, routing functionalities, authentication support, and communication infrastructure.

The main infrastructure components are:

- Firebase Realtime Database;
- Web Hosting Environment;
- OpenStreetMap Services;
- Bluetooth Beacon Infrastructure;
- QR Code Infrastructure.

The *Firebase Realtime Database* serves as the primary repository for tourism content. It stores:

- Cultural guides;
- Historical descriptions;
- Multimedia references;
- Geographical coordinates;
- Ratings;
- QR identifiers;
- Beacon identifiers;
- Language-specific content.

Firebase was selected due to its real-time synchronization capabilities, native Android integration, automatic scaling mechanisms, and support for cloud-based distributed architectures.

As a *Web Hosting Environment*, a traditional hosting environment based on Apache, PHP, and MySQL is employed for user account management and administrative operations.

The MySQL database stores:

- User accounts;
- Authentication credentials;
- User preferences;
- Administrative data;
- Access permissions.

This separation between MySQL and Firebase follows a functional segregation strategy. Transactional and authentication-related information is maintained within the relational database, while tourism content is stored in the cloud-based Firebase environment optimized for real-time mobile access.

OpenStreetMap provides geographic visualization and navigation capabilities. Through the *osmdroid* and *osmbonuspack* libraries, the system offers:

- Interactive maps;
- Route calculation;
- Point-of-interest visualization;
- Real-time navigation assistance;
- Geospatial data processing.

The use of *OpenStreetMap* ensures independence from commercial mapping providers while maintaining flexibility and scalability.

Bluetooth Low Energy (BLE) beacon technology enables proximity-based interactions within museums, archeological sites, and indoor cultural environments. Each beacon broadcasts a unique UUID that is matched against entries stored within Firebase. Upon detection, the application automatically retrieves and presents contextual information related to the corresponding location or exhibit.

QR Code Infrastructure provides an alternative mechanism for content activation and location identification. Users can scan QR codes positioned near monuments, exhibits, or landmarks to instantly access associated multimedia content and guidance information.

6.1.1. External Services

In details, External Services hold the key to functionality and interoperability. First, we choose to use a local host environment using XAMPP [81] (for testing and development purposes), implementing source code and functionalities with full compatibility with online hosting services provided currently by shared or dedicated hosting companies these days. The local or online hosting service (containing Apache web server [82], MySQL database [83], PHP [84] installation and phpMyAdmin [85]) will provide the necessary tools to host the main App's Users tables to store all information that deals with the users' accounts and their preferences. It will also be able to host the necessary RESTful API [86] implemented in the PHP language for Android applications to be able to communicate safely with the MySQL database. Moreover, this section also contains the necessary functionality for approved and selected users only (e.g., representatives from each community) to provide the ability to create new content for our application or to update the existing information, communicating accordingly with the backend Firebase Realtime Database.

Secondly, Firebase Realtime Database [87] is chosen as our main operational database. Firebase is the most preferred and most commonly used database from Google LLC, Android OS and Android applications generally. In this case, Firebase will be able to store all available data that users can exploit using our application and security issues are solved using the corresponding rules.

In details, Firebase Realtime Database offers several advantages over other databases like MongoDB or MySQL, for example, in the context of mobile and real-time applications, particularly for Android-based systems. Its tightly integrated real-time data synchronization allows changes to be instantly propagated to all connected clients without requiring additional infrastructure or complex configuration, making it especially suitable for applications that demand live updates and seamless user experiences. Firebase also provides built-in user authentication, fine-grained security rules, and automatic scaling, significantly reducing backend development and maintenance effort compared to a self-managed MongoDB deployment. Also, as a fully managed cloud service, Firebase eliminates the need for server provisioning, database administration, and manual replication setup, enabling faster development and easier deployment. Additionally, its native integration with Android SDKs, offline data persistence, and event-driven data listeners simplify client-side implementation and improve reliability in environments with intermittent connectivity. While MongoDB offers greater flexibility and control in large-scale or complex backend systems, Firebase Realtime Database is often more efficient and developer-friendly for real-time, mobile-first, and location-aware applications [88].

Thirdly, since many users may not be indigenous, we have to provide a digital map service. This is why we choose to use Open Street Map through its Android's "sibling", the osmdroid extension. Also, because of the fact that users may also request a path from their current position to some destination, we chose to use the osmbonuspack, which is a library to interact with OpenStreetMap data inside an Android application and moreover to perform real-time on demand routing.

6.1.2. Android Libraries

Android libraries [89,90] are collections of pre-written code, resources, and tools that developers can use to speed up the app development process. Instead of building features from scratch, developers can integrate libraries to handle common tasks such as networking, image loading, data storage, or UI design. For example, libraries like Retrofit simplify API calls, Glide or Picasso make image loading efficient, and Room provides a robust database management solution. These libraries not only save time but also ensure reliability, since they are often maintained by large developer communities and thoroughly tested in real-world projects.

A key benefit of using Android libraries is the ability to keep projects modular and maintainable. By leveraging these ready-made solutions, developers can focus more on building unique features instead of reinventing the wheel. Libraries also help with ensuring compatibility across different Android versions and devices, which can be a challenge in the fragmented Android ecosystem. Additionally, many libraries are open-source, giving developers the flexibility to customize them according to their project needs. This collaborative ecosystem makes Android development more efficient, innovative, and accessible for developers of all levels.

The necessary libraries to achieve our goals and the desired results are described in the previous figure in section Android Libraries. In order for our application to be able to perform its functionalities correctly, we need the corresponding external libraries. These libraries are the Firebase extension which will provide the ability to connect to the Firebase Realtime Database, the Volley library in order to perform HTTP requests to online

servers, the Picasso library to load image from online hosting providers, the Osmroid library to load maps and perform routing, the ZXING library is the most famous extension for usage in autonomous QR code readers and last the AltBeacon library which is able to provide available algorithms for Android's Bluetooth Detectors.

6.1.3. Android Application

The goal of the Android application is to offer users a complete digital tour guiding experience as long as the user is on-site, using mostly voice as the preferred means of communication. The preferred language for graphic design in Android is XML and for computational logic we prefer Java instead of Kotlin.

XML is used for graphical design in Android because it provides a simple, structured, and readable way to define the user interface separately from the application logic. By describing layouts, buttons, text fields, images, and other UI components in XML files, developers can easily design and modify the appearance of an app without changing the Java or Kotlin code that controls its behavior. This separation improves code organization, makes maintenance easier, and allows designers and developers to work independently. Additionally, XML supports reusable layouts, different screen sizes and orientations, and resource management [91,92], helping developers to create consistent and responsive user interfaces across a wide range of Android devices.

On the other hand, Java is used for programming in Android because it is a reliable, object-oriented programming language that is well suited for developing robust and scalable mobile applications. It provides features such as code reusability, strong memory management, exception handling, and a large collection of libraries that simplify app development. Java integrates seamlessly with the Android SDK, allowing developers to implement application logic, manage user interactions, access device hardware, and communicate with databases and web services. Although Kotlin is now the preferred language for Android development, Java remains widely used because of its stability, extensive documentation, and compatibility with existing Android applications and libraries.

The hierarchy makes use of packages to divide and conquer. In our implementation, we are trying to keep things simple and stupid in order to achieve an agile approach. This means that we are using self-sustained classes and parameterized variables to build components. The application contains eight main packages operating as the backbone of the system and using the corresponding services; users will be able to be guided to points of interest using only their mobile phone without the need for the extra cost of the physical presence of a guide.

6.1.4. Multi-Layered Security Architecture

The proposed Personal Tour Guide Platform, visualized in Appendix A in Appendix A.6 (Figure A6), adopts a multi-layer security architecture based on the principle of Defense-in-Depth, where multiple complementary security mechanisms operate together to protect user information, tourism content, cloud resources, and communication channels. The architecture is designed to address the security requirements of a cloud-connected mobile application that utilizes Geolocation services, multimedia content delivery, QR codes, Bluetooth beacons, and online databases. By implementing security controls at different layers of the system, the platform minimizes the impact of potential attacks and provides comprehensive protection against unauthorized access, data manipulation, and information disclosure.

At the *User Security Layer*, the architecture employs secure authentication mechanisms to ensure that only legitimate users can access protected services. Biometric authentication technologies, such as fingerprint recognition and facial identification, can be

integrated through the Android security framework to enhance user convenience while improving account protection. Secure session management techniques are also applied to maintain authenticated user sessions and prevent unauthorized access resulting from session hijacking or credential theft.

The *Application Security Layer* is responsible for enforcing authentication and authorization policies throughout the platform. Role-Based Access Control (RBAC) mechanisms are utilized to distinguish between ordinary tourists, content managers, and system administrators, ensuring that users can only access functions appropriate to their privileges. Input validation and sanitization procedures are implemented for all user-generated data, including login credentials, search requests, QR code contents, and beacon identifiers. These controls reduce the risk of common attacks such as SQL injection, malicious data manipulation, and unauthorized content access. Additional security checks are performed when processing QR codes and Bluetooth beacon interactions to ensure that only trusted and verified content is presented to users.

Communication Security is achieved through the use of encrypted network protocols that protect data exchanged between the Android application, cloud services, and external providers. All communications are transmitted through HTTPS connections secured by TLS 1.3, preventing eavesdropping, data interception, and man-in-the-middle attacks. Certificate validation and certificate pinning mechanisms can be employed to verify the authenticity of remote servers and further strengthen communication security, particularly when users connect through public wireless networks commonly found in tourism environments.

The *Cloud Security Layer* protects backend services and online repositories used by the platform. Firebase security rules and access control policies restrict access to tourism content based on user roles and authentication status. Administrative operations are protected through privilege management mechanisms that limit content modification capabilities to authorized personnel only. Audit logging and monitoring services record important security events, including authentication attempts, content updates, and administrative actions, enabling system administrators to detect suspicious activities and perform forensic investigations when necessary.

At the *Data Security Layer*, sensitive information is protected both during storage and processing [93]. User credentials are secured using modern password hashing algorithms such as Argon2 or bcrypt combined with unique cryptographic salts, providing strong resistance against brute-force and dictionary attacks. Sensitive user information, authentication tokens, and application data can be encrypted using AES-256 encryption mechanisms to protect against unauthorized disclosure. On Android devices, secure storage facilities such as the Android Keystore system are recommended for protecting locally stored credentials and cryptographic keys. Regular backup procedures, integrity verification mechanisms, and secure recovery processes further enhance the resilience and reliability of the platform.

Overall, the proposed security architecture provides a comprehensive framework for protecting the Personal Tour Guide Platform against a wide range of cyber security threats. By integrating user authentication, application security controls, encrypted communications, cloud-based access management, and secure data storage techniques, the architecture ensures the confidentiality, integrity, availability, and privacy of both user information and tourism content while supporting the reliable operation of the platform in real-world deployment environments.

6.2. Personal Tour Guide Application's Design

Application design and mockups play a crucial role in the development process because they serve as the blueprint for the final product. Before a single line of code is

written, designers and developers collaborate to visualize how the application will look and function. The design phase focuses on defining the layout, color schemes, typography, icons, and overall user experience (UX). It ensures that the application not only looks appealing but also provides intuitive navigation, making it easy for users to accomplish tasks. A well-structured design reduces confusion and enhances usability, which directly impacts user satisfaction and retention.

Mockups are detailed static representations of an app's interface. Unlike rough sketches or wireframes, mockups include realistic design elements such as colors, fonts, and imagery, giving stakeholders a clear preview of the end product. They help developers, clients, and team members align on the visual direction and functionality before moving into coding. Mockups can also be used to test design concepts, gather feedback, and make adjustments early, saving time and resources in the long run. Tools like Figma, Sketch, and Adobe XD made creating mockups faster and more collaborative, allowing multiple team members to work simultaneously on a design.

Beyond visualization, mockups play a vital role in communication. They act as a bridge between designers and developers, ensuring that ideas are translated into functional features accurately. They also provide a foundation for prototyping, where interactivity is added to simulate user flows and actions. This step is especially useful for identifying usability issues early. In short, application design and mockups are not just about aesthetics. They are essential for aligning vision, reducing risks, and ensuring that the final product meets user needs effectively.

Login and registration windows are essential components in many Android applications, as they provide users secured access and personalized experiences. The login window typically includes input fields for a username, email, or phone number along with a password field. Additional features such as "Forgot Password," "Remember Me," and support for biometric authentication (like fingerprint or face recognition) are often included to enhance usability and security. A well-designed login blueprint ensures that users can access the app quickly and safely while minimizing friction. Clear error messages, input validation, and user-friendly layouts are also crucial to prevent frustration and improve the overall experience.

The registration window, on the other hand, focuses on creating new user accounts by collecting necessary information such as name, email, password, and sometimes phone numbers or profile details. Modern registration blueprints often include password strength indicators, confirmation fields, and optional integrations with social media or Google accounts for quicker sign-ups. The design should remain simple and uncluttered, guiding users step-by-step through the process without overwhelming them. Security measures, such as email verification or CAPTCHA, are commonly integrated to prevent spam and unauthorized access. Together, login and registration window blueprints form the foundation of user authentication, balancing security with convenience to create a smooth onboarding experience.

The proposed design for our application's Login and Registration windows is shown in Figure 2. Login and Registration windows are equipped with EditText fields which are able to retrieve given inputs for inner processing and an activation button that triggers communication with remote MySQL.

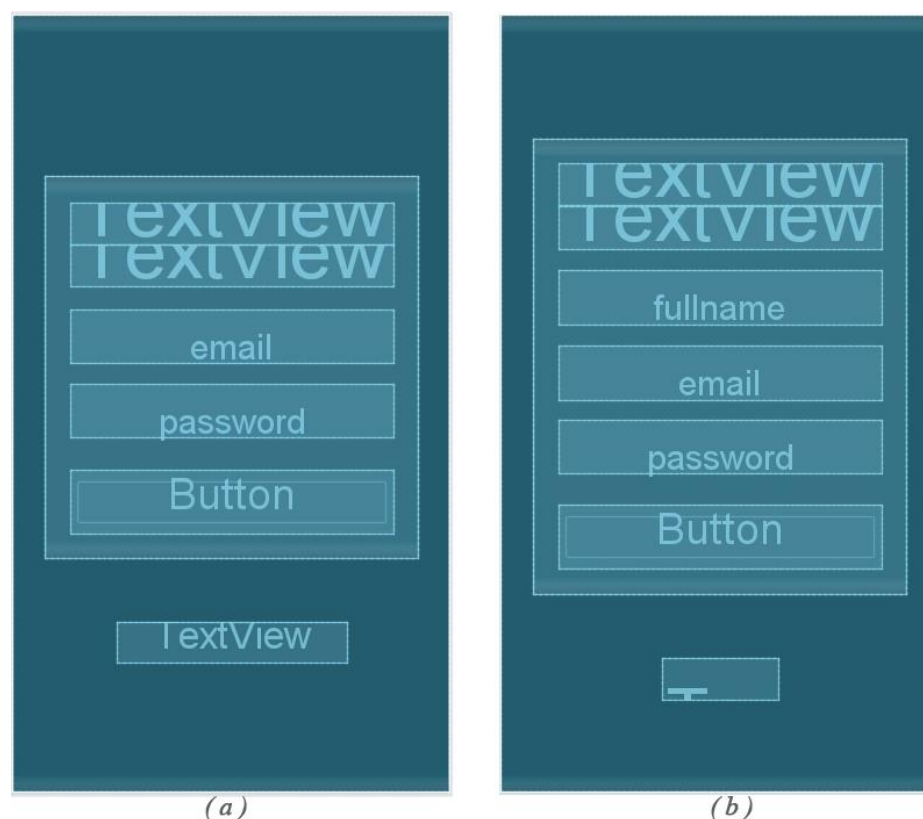


Figure 2. (a) Login window and (b) Registration window wireframes.

As some readers may correctly notice, in the Login and Registration windows, we do not focus on the interface on security concerns since these points are addressed mainly on the server side. The online hosting platform is able to distinguish users and keep them in different layers with different permissions. Plugins which are able to detect and prevent DDos and Brute Force attacks are installed on the server side. Simple techniques such as files accessing permission can be edited using the .htaccess files. Last, all communications with the Android devices are secured using safe and encrypted SSL when transmitting and receiving data and all sensitive information stored in database uses MD5 encryption. Moreover, as we show later, a Biometric Authentication service is also supported by our application, ensuring the correct and appropriate operation of our application only by authenticated users.

On the other hand, the design for the application's Main and Guide windows is presented in Figure 3. In the Main window, a GridLayout is used to divide space into equal parts. Each cell contains a CardView with an ImageView and a TextView beneath. Each CardView will be responsible for one category from the available guides in our system. On the other hand, the Guide window provides all available information for each guide to users. The design starts with an ImageView (the main image for the selected guide) and then provides information like the title, intro, ratings and guidance itself on the left side. Next to the title and the intro sections, on the right side, we can host the two necessary options, which are the favorite and location icons. Each of these windows will also provide a basic toolbar on top of the design (included from an external file), holding main features and operations.

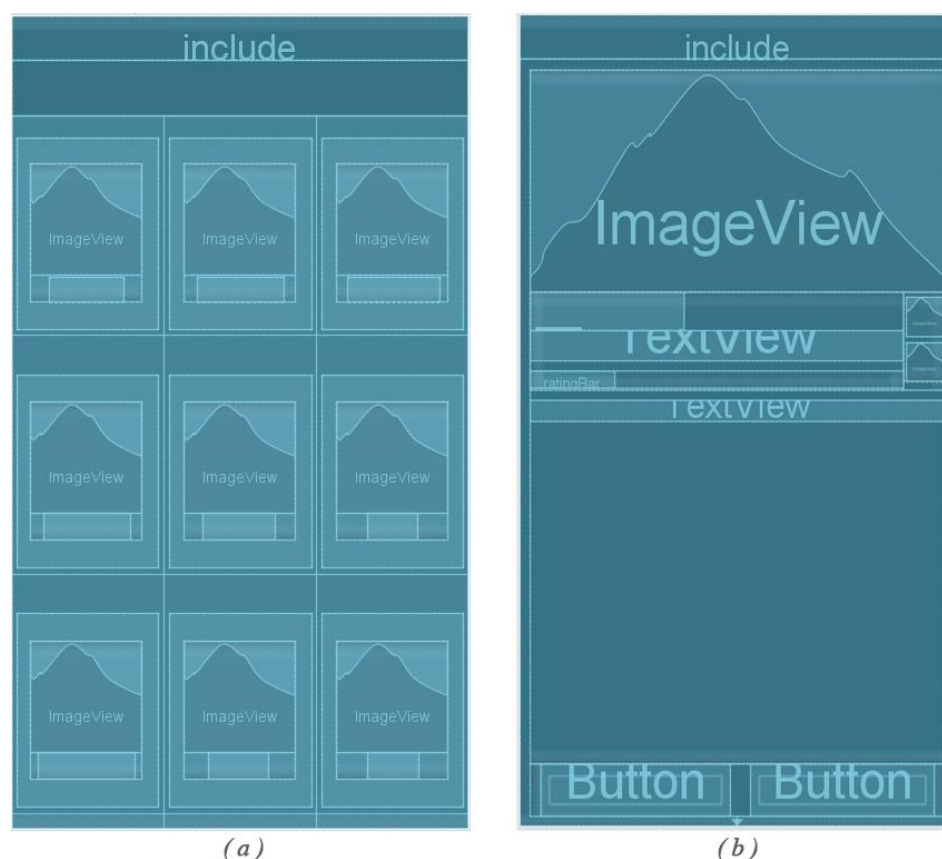


Figure 3. (a) Main Navigation window and (b) Guide window wireframes.

6.3. Firebase UML Diagram

Firebase databases, whether it is the Realtime Database or Cloud Firestore, are NoSQL systems that follow a document-oriented model. Instead of tables, data is organized into collections of documents, with each document structured like a JSON file. These documents can also include nested collections or other documents.

In theory, every document within a collection can differ entirely from the others. Because of this, using a UML class diagram would not be appropriate, as classes are based on shared attributes and behaviors.

In practice, typically, the documents within a collection consist of objects that are quite similar or at least related. Some objects may have a few extra or fewer properties, but overall, they represent the same type of entities.

6.3.1. Describing the Database Schema

In this situation, UML class diagrams can be effectively applied, since UML is class-oriented rather than table-oriented. In this case, we must pay attention to the following:

- A class represents a specific type of entity. In reverse engineering, we would typically begin with the known collections;
- Minor structural variations between documents of the same type can be shown in the UML diagram as optional properties (e.g., multiplicity [0..1], like lastLogin in the users example);
- Larger differences between documents that still share common traits are usually modeled with specialized subclasses;
- The main challenge lies in modeling embedded documents and sub-collections, which often correspond to additional or composite classes.

In principle, UML class diagrams are not meant solely to represent the database schema (data only). They are intended to design the object model of our application (data and behavior) independently of the database. This object model can then guide the database design according to the types of objects required to meet users' needs.

In our case, as shown in Figure 4, Firebase Realtime Database will include six types of classes, with each one containing the entries for its field of interest. These six types are Nature, Aquariums, Religion, Museums, Cities and Ruins. Each one of these classes will contain a list of instances of the Guide class. The Guide class will contain all necessary information about a tour guide such as the unique auto generated identifier of each entry "id", its title "title", an introduction text "intro", a URL path to an image hosted online "image", the text content of the guidance itself "guidance", a unique self-generated key to be used in case of a QR code "key", the language of the presentation "language", the latitude and the longitude of the place in maps "latitude and longitude", the users' average rating value "rating", a URL path to the webpage of the selected entry "link" and last but not least the UUID of the Bluetooth beacon device that may or may not exist in the area "uuid". We must notice that each Bluetooth beacon device is assigned a unique UUID upon activation and will keep broadcasting this information continuously until told otherwise. Once our application detects in the surrounding space a UUID that matches one of the UUIDs supported by our database, it instantly informs users about its existence and its relevant information. All values for the entries above will be String type to simplify communication from the Android device to the Firebase Realtime Database.

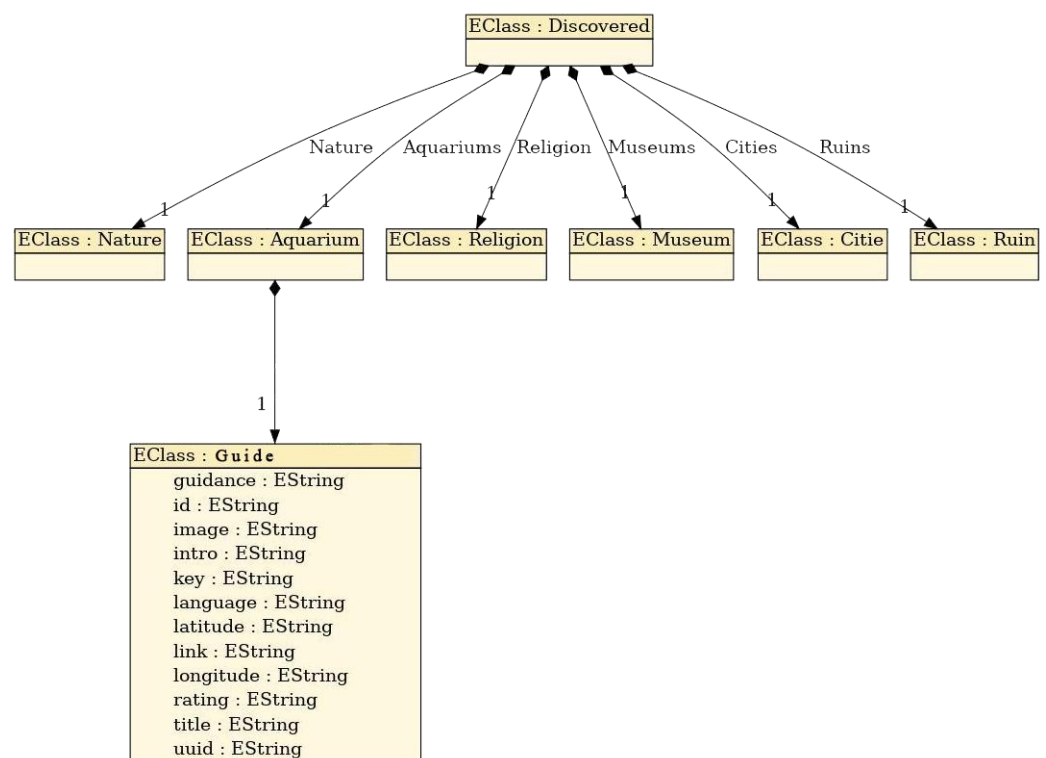


Figure 4. Firebase JSON diagram.

To validate the above JSON UML Diagram, described in Figure 4, using mathematical modeling we must next prove that the structure, constraints, and relationships represented in UML (and encoded in JSON) are formally correct and consistent using math-based models (sets, relations, logic, graphs, automata, or category theory). In our case and for the current version of the Firebase Realtime Database, the formal data model can be described in a hierarchical structure as follows.

Formal Data Model (Sets):

$$\begin{aligned}
 C &= \{\text{Aquariums, Cities, Museums, Nature, Religion, Ruins}\} \\
 \text{For each category } c \in C \text{ let } E_c &= \{e_{c,1}, e_{c,2}, \dots\} \text{ be the set of entities.} \\
 |E| &= \sum_{c \in C} |E_c|
 \end{aligned} \tag{1}$$

Entities Tuple Model (Constraints):

$$\begin{aligned}
 \forall e \in E, e &= (\text{id, title, intro, guidance, image, key, language, lat, lon, rating,} \\
 &\quad \text{link, uuid}) \\
 \forall e_i \neq e_j, \text{id}_i &\neq \text{id}_j \\
 -90 \leq \text{latitude} &\leq 90 \\
 -180 \leq \text{longitude} &\leq 180 \\
 35 \leq \text{latitude} \leq 41 \text{ and } 19 \leq \text{longitude} \leq 28, &\text{ all points currently inside} \\
 &\text{Greece} \\
 \forall e \in E, 0 \leq \text{rating}(e) &\leq 5 \\
 \text{uuid} \in \text{UUID} \cup \{\emptyset\} \\
 \text{language} \in \{\text{English, Greek, Chinese,}\dots\}
 \end{aligned} \tag{2}$$

Type constraints define what kind of value each field of the JSON file is allowed, using mathematical sets (for example, strings, real numbers, or fixed categories). In our database file, as Table 1 shows, most fields like title, intro, and guidance belong to the string type, while language belongs to a finite categorical set (English, Greek, Chinese). Fields such as latitude, longitude, and rating are semantically numeric (they satisfy real-number bounds like valid coordinates and ratings between 0 and 5) but are stored as strings, which is mathematically valid for representation but weak for computation. The uuid field behaves as a partial identifier because some values are empty. Overall, the data obeys its logical and numerical rules. Stricter typing (using actual numbers instead of numeric strings) could make it safer for mathematical operations like distance calculation, sorting, and optimization but this operation is successfully performed on the Android client's side.

Table 1. Current database fields with expected and observed-representing types.

Field	Expected Type	Observed (Representing)
id	string	string (numeric)
title	string	string
intro	string	string
guidance	string	string (long text)
image	string	string (filename)
key	string	string
language	string	English/Greek/Chinese
latitude	string	string (represented real)
longitude	string	string (represented real)
rating	string	string (represented real)
uuid	string	string (empty or UUID)
link	string	string (URL)

6.3.2. Superimposition with Mathematical Modeling

The principle of superimposition (superposition) is a fundamental concept used in physics, engineering, mathematics and informatics to analyze systems where multiple effects act simultaneously. Superimposition states that in a linear system, the total response caused by multiple inputs is equal to the sum of the responses caused by each input individually. In simple terms, if several influences act on a system, you can analyze each one separately and then add the results. In our case, we will connect superimposition directly to our system and show how it applies in a real engineering sense within our AR tourist guiding platform.

Our application combines multiple independent subsystems, such as GPS/Geolocation (OpenStreetMap), AR visual overlays, Audio narration (TTS), Firebase data retrieval, Bluetooth beacons and QR code inputs. A System-Level mathematical model can model our application as a linear combination of independent services as follows.

Total User Experience and System Response ($R(t)$):

$$R(t) = R_{GPS}(t) + R_{AR}(t) + R_{Audio}(t) + R_{DB}(t) + R_{Beacon}(t) + R_{QR}(t) \quad (3)$$

In our architecture, the Android app operates as the orchestrator, the Firebase as the data source, the OSM as the spatial layer manager and the AR as the visual and audio service performer. When a user visits a monument the GPS detects location, Firebase returns content, AR overlays visuals, Audio narrates, Beacon confirms proximity and the final experience contains the sum of all effects.

In the proposed bio-cultural tourist guiding platform, sensor fusion integrates information from GPS services, Bluetooth beacons, QR codes, and smartphone camera inputs to improve user localization and environmental understanding.

Location Sensor Fusion (P):

$$P = P_{GPS} + P_{Beacon} + P_{QR} \quad (4)$$

GPS provides approximate outdoor positioning, Bluetooth beacons refine proximity detection within specific areas, and QR codes enable precise identification of monuments or exhibits. For example, GPS says "User is near the museum", the Beacon says "User is inside Room A" and the QR code says "User is viewing Artifact #12". By mathematically combining these independent inputs, the application can determine the user's exact position and contextual state more effectively, enabling accurate augmented reality overlays, personalized audio narration, and adaptive navigation guidance.

Attributes stored in the Firebase Realtime Database collectively contribute to the formation of a complete tour guide entity within the application. Each guide record consists of multiple independent information components such as the title, introduction, guidance text, image, geographic coordinates, rating, and language. The final user experience is produced through the combination of these individual data elements, which can be mathematically represented as the summation G of attributes' contribution to the overall guide structure, where each g_i is (title, intro, image, guidance, rating, etc.).

Database Contribution Model (G):

$$G = \sum_{i=1}^n g_i \quad (5)$$

Sequence Behavior through Dynamic Superposition describes how the overall state of the application evolves over time as multiple user actions and system events occur sequentially and interactively. In the proposed tourist guiding platform, operations such as user login, GPS updates, QR code scanning, Firebase data retrieval, and augmented reality rendering each contribute incrementally to the system's behavior. The final application state at any given moment can therefore be viewed as the cumulative superpo-

sition of all previous events and interactions, called “S”. This dynamic process enables the system to adapt continuously to user context and environmental changes, ensuring responsive navigation, real-time content delivery, and an interactive guiding experience. In our case, $Events(t)$ cases can be considered the Login event, Location update, QR scan and Data fetch events.

Sequence Behavior and Dynamic Superposition (S):

$$S_{(t)} = S_0 + \sum_{t=1}^n Events_{(t)} \quad (6)$$

6.4. UML Class Diagram

A UML (Unified Modeling Language) [94–96] class diagram [97] is one of the most widely used diagrams in software engineering for representing the static structure of a system. It provides a blueprint of the system by visually describing its classes, attributes, operations (methods), and the relationships between objects. Class diagrams help developers, designers, and stakeholders understand the architecture of a system at a conceptual, logical, or implementation level.

The UML class diagram serves as a foundational modeling tool for representing the static structure of the proposed system by clearly defining its classes, attributes, operations, and interrelationships. By visualizing system architecture prior to implementation, the class diagram helps identify core components, clarify responsibilities, and reduce design ambiguity. It supports the identification of reusable classes and shared functionalities, minimizing redundancy and promoting modularity across the application. Additionally, the diagram facilitates effective communication between technical and non-technical stakeholders by providing a clear, standardized representation of system structure and dependencies. As a form of technical documentation, the UML class diagram also plays a crucial role in system maintenance and future development, enabling developers to understand existing designs, implement extensions efficiently, and ensure scalability as system requirements evolve.

In summary, a UML class diagram is a powerful tool for modeling the structure of object-oriented systems. By defining classes, attributes, operations, and relationships, it creates a clear representation of how different parts of a system interact. This clarity supports efficient development, reduces complexity, and ensures scalability of both websites and mobile applications.

As class diagram shows in Figure 5, we try to use the technique of divide and conquer, separating classes into packages. Analyzing the class diagram figure from left to right, the package qrcode will hold the classes responsible for reading and decoding images with QR codes printed on them. The firebase package contains classes responsible for communicating with the online Firebase Realtime Database and synchronizing when it is necessary. The location package contains classes responsible for communication with the online repository of the open street map service and finding routing paths. The tts package holds classes responsible for the embedded text to speech service provided by the Android OS. The beacon package provides algorithms able to support Bluetooth Beacons protocols. The app package contains classes responsible for the correct application functionality and setup. The helper package contains classes crucial for apps operation and provides the necessary data templates. Last, the ui package contains all necessary classes that provide basic user interface functionalities.

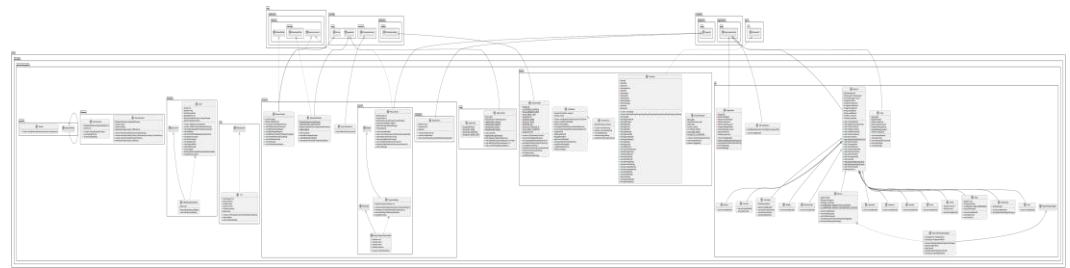


Figure 5. Personal Tour Guide Application Class diagram.

Because of the size and the complexity of the class diagram, we have separated it into four different pieces. These figures are shown individually in Appendix B (Appendices B.1–B.4, Figures A7–A10).

6.5. UML Sequence Diagram

A UML (Unified Modeling Language) sequence diagram [98] is a type of interaction diagram that models the dynamic behavior of a system by showing how objects communicate with each other over time. Unlike class diagrams, which focus on static structure, sequence diagrams emphasize the flow of messages between system components, making them valuable for visualizing system processes and use-case scenarios.

Sequence diagrams use several elements to illustrate object interactions:

- **Objects (Participants):** Represented at the top of the diagram as boxes, each object is typically named according to its role (e.g., User, System, Database);
- **Lifelines:** Dashed vertical lines extending downward from each object, indicating the existence of that object over time;
- **Activation Bars:** Narrow rectangles placed on a lifeline to show when an object is active or performing a process;
- **Messages:** Horizontal arrows exchanged between objects that represent communication. Messages can be synchronous (waiting for a response), asynchronous (not waiting), or return messages;
- **Loops and Conditions:** Represented using frames, they indicate repeated interactions (loops) or alternative flows (conditions).

UML sequence diagrams are powerful tools for modeling time-ordered object interactions within a system. They not only help designers and developers understand the step-by-step flow of processes but also serve as valuable documentation throughout the software development lifecycle. By focusing on communication between components, sequence diagrams bridge the gap between requirements and technical implementation.

The following two figures (Figures 6 and 7) show the sequence diagrams for the Main Activity and the Tour Guide activity. For instance, in a Main Activity sequence:

- The onCreate event calls the BasicUI class;
- BasicUI class initializes all necessary tools of the app;
- If user is not Logged In, then activity returns;
- Users can call FavoriteTours or Firebase RecordsLoader on-demand.

This flow, represented visually, helps ensure that all required interactions are considered and correctly implemented.

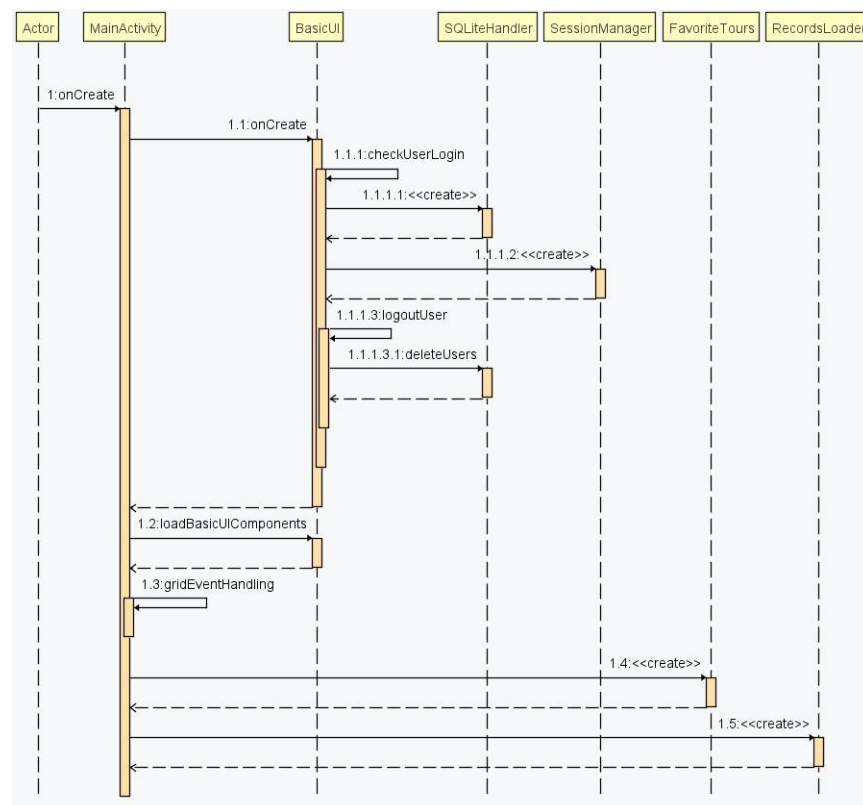


Figure 6. Main window Sequence diagram.

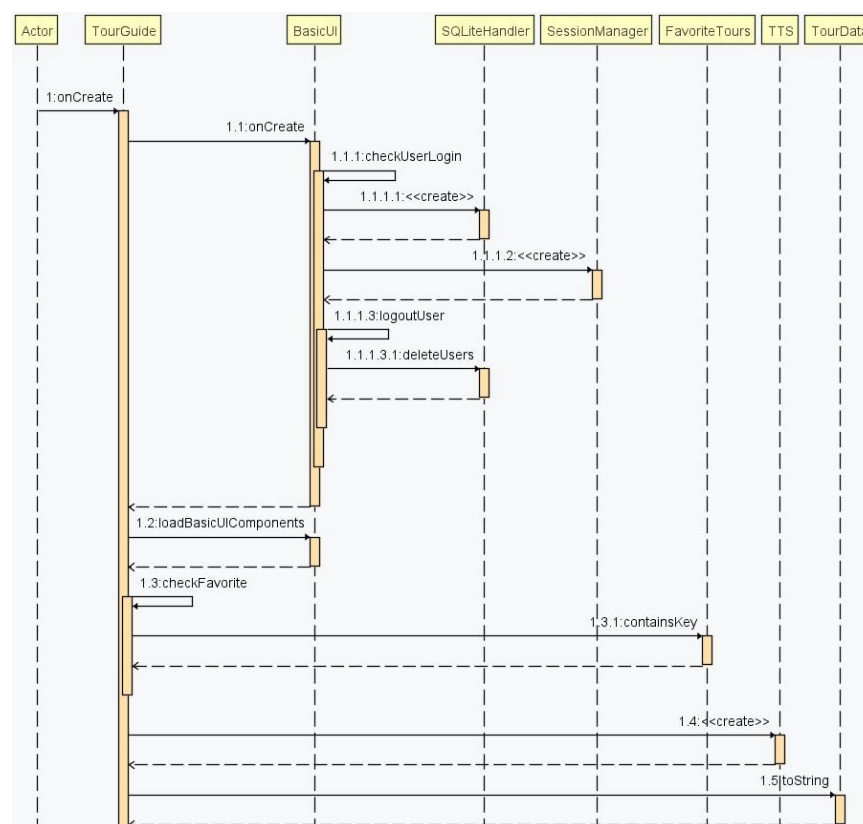


Figure 7. Tour Guide window Sequence diagram.

6.6. UML Use Case Diagram

The presented use case diagram illustrates the functional structure and interaction flow of the AR Bio-Cultural Tourist Guiding System developed for Android

smartphones. The primary actor, identified as the Tourist/User, interacts with several core system functionalities including user registration and login, map and augmented reality navigation, QR code scanning, beacon detection, audio guidance, cultural information retrieval, favorites management, and language selection operations. These user-centered use cases are connected to the central process of retrieving data from the Firebase Realtime Database, which serves as the main repository for tourism-related information and multimedia content. The diagram demonstrates how the application integrates location-based services, augmented reality features, and contextual information delivery to enhance the tourist experience.

The use case diagram also illustrates the interaction between the system and external actors such as the Administrator and External Services. The Administrator actor is responsible for managing users, multimedia content, points of interest, and system analytics, ensuring that the platform remains updated and operational. Meanwhile, the External Services actor represents third-party integrations including GPS/location and TTS services, OpenStreetMap (OSM) services, and REST web services that support navigation, route visualization, and real-time communication. Include associations shown in the diagram indicate dependencies between use cases, such as retrieving user location before navigation and accessing Firebase data for content presentation. Overall, the diagram provides a high-level overview of the system's operational behavior, modular architecture, and interaction between software components and external services. The Use Case diagram is shown in Appendix A in Appendix A.2 (Figure A2).

6.7. UML Activity Diagram

The activity diagram illustrates the workflow and operational sequence of the proposed AR Bio-Cultural Tourist Guiding System from the moment a tourist launches the application until the session ends. The process begins with the user opening the Android application and performing registration or login authentication. The system then validates the user credentials through MySQL and Firebase services before granting access to the main application features. Once authenticated, the user can select different functionalities such as map navigation, augmented reality viewing, QR code scanning, beacon detection, and cultural information retrieval. Depending on the selected feature, the system requests GPS location data and retrieves relevant tourism information from Firebase or external web services, and displays multimedia content including maps, AR overlays, and audio narration.

The diagram also demonstrates the interaction between the Android application and external services throughout the execution flow. External services such as GPS providers, Firebase databases, REST web services, and audio streaming services support the application by supplying location information, tourism content, and multimedia resources in real time. User activities such as adding favorites, rating locations, sharing content, and interacting with AR elements trigger data update operations that are stored in Firebase for future personalization and analytics. Finally, the workflow concludes when the tourist logs out, causing the system to terminate the active session and optionally clear temporary session data. Overall, the activity diagram provides a clear representation of the system's dynamic behavior, control flow, and interaction between users, software components, and external services. The Activity diagram is shown in Appendix A in Appendix A.3 (Figure A3).

7. Platform Implementation

7.1. MySQL & RESTful API, Android Login and Registration with PHP, MySQL, and SQLite

7.1.1. System Architecture

Implementing login and registration in Android often involves combining local storage with a remote backend. A common stack for this is PHP + MySQL for the server side, and SQLite for local persistence on the Android device. This setup allows users to authenticate securely while also enabling offline access.

System Architecture is based on the following:

- **Android App (Client):**
 - a. Provides the user interface for login and registration.
 - b. Sends requests (via HTTP/JSON) to a PHP backend.
 - c. Uses **SQLite** to store user sessions or cached data locally for offline mode.
- **PHP Backend (Server):**
 - a. Receives HTTP requests from the Android app.
 - b. Processes login and registration logic.
 - c. Interacts with **MySQL** to store and verify user data.
- **MySQL Database (Remote):**
 - a. Stores registered user credentials (username, email, password hash).
 - b. Supports authentication and data persistence.

The architecture for implementing login and registration in Android using PHP, MySQL, and SQLite is built on the interaction of three distinct but interconnected layers: the Android client, the PHP backend, and the Database Systems (MySQL and SQLite). Each component fulfills a specific role, and the system's strength lies in the way they collaborate to provide both online functionality and offline persistence. By separating concerns across these layers, the architecture achieves flexibility, scalability, and reliability.

7.1.2. System Workflow

To interact with the MySQL database, we need to build a REST API first. REST Api job is to get the request from client, interact with database and finally give the response back to client. So we will create a simple PHP and MySQL API first, as shown in Figure 8. Our API carries out the following functions:

- ⇒ Accepts requests in GET/POST methods;
- ⇒ Interacts with the database by inserting/fetching data;
- ⇒ Finally, gives a response back in JSON format.

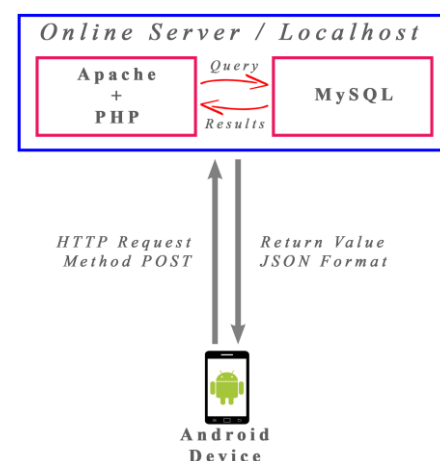


Figure 8. Android device to MySQL Server Communication.**Registration Flow**

- User enters details (username, email and password) in the Android app.
- App sends data via **HTTP POST** to a PHP API endpoint (e.g., register.php).
- PHP script validates inputs, checks for existing users, and inserts new records into MySQL.
- PHP returns a **JSON response** indicating success or failure.
- Android app displays a confirmation and optionally stores the user in **SQLite** for local access.

Login Flow

- User inputs email and password in the app.
- App sends credentials via **HTTP POST** to a PHP API endpoint (e.g., login.php).
- PHP script queries MySQL to verify the user and password hash.
- If valid, PHP returns user details as JSON.
- Android app saves the user session/token in **SQLite** for persistent login.

7.1.3. MySQL and SQLite Database Structure

The database structure in a login and registration system built with Android, PHP, MySQL, and SQLite represents the backbone of user authentication and data management. At its core, the design revolves around two complementary databases. MySQL on the server side and SQLite on the client side. Each serves a distinct purpose, and together they establish a hybrid model that ensures both centralized control and localized accessibility.

The MySQL database is the authoritative storage system for user accounts. Its relational design provides structure and integrity, with the users table typically including fields such as an auto-incrementing primary key (id), a unique email, a password hash, and timestamps for record creation. This schema (shown in Figure 9) not only guarantees uniqueness and prevents duplicate accounts but also supports scalability as the application grows. By hashing passwords before storing them, MySQL ensures that sensitive information is never preserved in plain text, aligning the structure with modern security best practices. Furthermore, relational constraints help maintain consistency across records, ensuring that every registered user is represented accurately within the central repository.

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(100) UNIQUE,
  password VARCHAR(255),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Figure 9. MySQL table (users).

On the client side, SQLite plays a different but equally important role. Unlike MySQL, it is lightweight, file-based, and embedded directly within the Android device. The local_users table is designed to store essential session details such as the user's name, email, or session token, rather than serving as a complete mirror of the remote database. This local storage supports offline persistence, allowing users to remain logged in or ac-

cess cached information even without an active internet connection. The design (shown in Figure 10) is intentionally minimalist, reducing the risk of data duplication or inconsistency while still providing functionality that improves user experience.

```
CREATE TABLE local_users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT,
    email TEXT,
    session_token TEXT
);
```

Figure 10. SQLite table (local_users).

The coexistence of these two databases highlights a thoughtful architectural decision. MySQL ensures that all accounts and credentials remain centralized and universally accessible, making it the single source of truth. SQLite complements this by delivering convenience and responsiveness, especially in environments with unreliable connectivity. The structural distinction between the two of them, one relational and global, the other lightweight and local creates a system that balances robustness with usability. In essence, the database structure is not just a technical implementation detail but a deliberate design strategy that reinforces the system's goals of security, accessibility, and efficiency.

7.1.4. The PHP API

The PHP APIs provided for registration and login are central to the interaction between the Android client and the MySQL database. These scripts function as the gateway through which data flows, transforming user input into secure database transactions and returning structured responses that the Android app can interpret. Their simplicity is intentional, making them easy to understand and deploy, but beneath this simplicity lies careful logic that ensures both security and reliability in handling sensitive operations. The hierarchy of the server-side files is described in Figure 11. The overall system was hosted under XAMPP software bundle version 3.2.4 for Windows 10 on localhost, including PHP Version 7.4.7 and database server 10.4.13-MariaDB. Nevertheless, the system is also fully tested in a real/commercial online Apache web server, using a shared Web Hosting Environment.



Figure 11. Web Platform Hierarchy.

7.1.5. The Android Side

The Android side of the login and registration system represents the user-facing layer where functionality meets experience. Its role is not only to collect input and dis-

play results but also to mediate communication with the backend and manage local persistence. The architecture ensures that the Android client remains lightweight yet capable of handling essential logic, particularly in parsing JSON responses and interacting with SQLite for session management. By leveraging libraries such as Volley, the Android app simplifies network communication, allowing developers to send HTTP requests and handle asynchronous responses without extensive boilerplate code. This abstraction is significant because it reduces complexity while improving reliability, ensuring that user data is transmitted securely and responses are processed efficiently. The overall Android side workflow contains the following:

- Use **Volley** to send requests to the PHP backend;
- Parse the JSON response;
- Store login state in **SQLite** for offline persistence.

Android application setup and configuration can be handled quickly and easily since we have developed a single file holding all the necessary information for the server's IP and all required secondary URLs for our application to be able to perform its functioning as shown in Figure 12.

```
public class AppConfig {
    public static String SERVER_IP = "http://192.168.1.10/";
    // Server user login url
    public static String URL_LOGIN = SERVER_IP+"personaltourguide/login.php";
    // Server user register url
    public static String URL_REGISTER = SERVER_IP+"personaltourguide/register.php";
    public static String URL_IMAGE_HOST = SERVER_IP+"personaltourguide/images/";
}
```

Figure 12. Content of the AppConfig.class file.

In order for the Android App to be able to handle HTTP requests using the Volley external library we must inform the build.gradle file of our intentions, as shown in Figure 13, and we must update the AndroidManifest.xml file for our intention to use Internet by adding the tag uses-permission with value “android.permission.INTERNET”.

```
build.gradle
dependencies {
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'
    compile 'com.mcxiaoke.volley:library-aar:1.0.0'
}
```

Figure 13. Gradle dependencies for HTTP Requests.

An essential feature of the Android side is its ability to interpret and act upon the JSON data received from the PHP backend. Once a login request is validated and user details are returned, the app must store this information locally for persistent sessions. This is where SQLite becomes critical since it allows the app to cache credentials, session tokens, or user profiles, enabling offline access and improving load times by reducing repeated server calls. The local database is intentionally lightweight and tailored to the specific needs of the authentication system, ensuring that it supports responsiveness without overwhelming device resources.

Beyond data handling, the Android side also embodies usability and security considerations. For instance, error messages from failed login attempts are displayed to

guide users, while session tokens retrieved from the backend are stored securely to prevent unauthorized access. By managing the state between online validation through PHP/MySQL and offline persistence through SQLite, the Android client (in this case our application) ensures continuity of experience. This design reflects the hybrid nature of modern mobile applications, which must balance connectivity with resilience.

7.2. Implementing SQLite in Android Apps

Working with SQLite [99–101] in Android involves creating a database, defining tables, and performing CRUD operations (Create, Read, Update, and Delete). Android provides the SQLiteOpenHelper class to manage database creation and version control. This helper class simplifies the process of managing database lifecycles, including creating tables and handling upgrades when the schema changes.

To use SQLite in an Android project, a developer typically creates a class that extends SQLiteOpenHelper and overrides two key methods: onCreate() and onUpgrade(). The onCreate() method defines the SQL statements used to set up the initial schema, such as creating tables. The onUpgrade() method handles changes between database versions, ensuring data migration or schema alterations are done smoothly without losing existing data.

Once the helper class is set up, a developer can open the database using getReadableDatabase() or getWritableDatabase(). These methods return a SQLiteDatabase object, which can be used to execute SQL statements or call convenience methods like insert(), query(), update(), and delete().

7.3. Navigation Service on Android

GPS navigation tools on the Android platform play a central role in location-aware mobile applications by enabling real-time positioning, routing, and contextual guidance. Android provides access to satellite-based Global Positioning System (GPS) data through the Location Services framework, which integrates GPS, Wi-Fi, and cellular network signals to deliver accurate and energy-efficient positioning. This hybrid approach allows applications to dynamically adapt to outdoor and semi-urban environments while balancing precision and battery consumption. Android GPS APIs support continuous location updates, geofencing, and proximity detection, which are essential for navigation, tour guidance, and context-aware content delivery. When combined with digital map services such as OpenStreetMap or Google Maps, GPS navigation tools enable turn-by-turn routing, distance calculation, and spatial visualization of points of interest. In tourism and cultural guiding applications, these capabilities allow users to navigate unfamiliar environments safely and intuitively, while enabling systems to trigger location-based multimedia content and personalized guidance, thereby enhancing both usability and immersion.

7.3.1. Integrating OpenStreetMap with Android

OpenStreetMap [102,103] data is widely used in Android applications, especially those focused on mapping, navigation, tracking, and location services. Android developers can integrate OSM in various ways, using libraries and APIs that render OSM tiles or convert OSM data into usable formats. Some popular tools for integrating OSM with Android include OSMDroid, Mapsforge, Mapbox, and GraphHopper. OSMDroid is an open-source Android library that allows developers to use OSM maps as an alternative to Google Maps. It supports features such as zoom controls, overlays, offline maps, and GPS tracking. Developers can use OSMDroid to create custom map applications without depending on Google Play Services.

7.3.2. Implementing Osmdroid in Android with Java and Osmbonuspack Library

One of the basic features of an application which is able to help tourists guide and find places of interest should be a service that could be able to make use of interactive digital maps and support geo-positioning and live routing. Google Maps could be an option, but its pricing policy makes it quite expensive. On the other hand, osmdroid is free and makes use of the OpenStreetMap project. The main advantage of OpenStreetMap is its open data policy. Anyone can use it, modify it, and share it, provided they follow the Open Database License (ODbL). This makes it very popular for developers, researchers, and organizations that need customizable map data without the restrictions or licensing costs that often come with commercial map providers.

On the other hand, osmdroid is an open-source Android library that provides a MapView component and related tools for displaying and interacting with maps based on OpenStreetMap (OSM) data. It serves as an alternative to Google Maps for Android, allowing developers to integrate customizable maps into their apps without needing an API key or being tied to Google's licensing restrictions.

In this section, we will walk through the deployment of the osmdroid library (an open-source Android map view) and osmbonuspack (an Android library with useful overlays, geocoding, routing, KML, clustering, etc.) to a Java Android project. It covers setup, permissions, layout, core MapView usage, common overlays (markers, polylines), geocoding/routing, offline tiles, and useful tips/troubleshooting.

7.3.3. Android Project Setup for OSM

In the Project setup stage, we must begin by adding the required dependencies for osmdroid (which provides the MapView and tile handling) and osmbonuspack (which adds extra features like overlays, clustering, routing, and geocoding) to our app's build.gradle file. Since both libraries are hosted on Maven Central, we usually do not need extra repositories, but we should always use the latest stable versions.

Next, we update the AndroidManifest.xml with essential permissions such as INTERNET, ACCESS_NETWORK_STATE, and location permissions (ACCESS_FINE_LOCATION and ACCESS_COARSE_LOCATION), plus storage access if we plan to use offline tiles. On devices running Android 6.0 (Marshmallow) or higher, we also need to request users' personal permissions (like location and storage) at runtime. Finally, we must configure the osmdroid user agent string in our code to identify our app to tile servers, which is a key requirement to avoid being blocked when fetching map tiles. This requires the following actions:

- Create a new Android project (Java) in Android Studio. Minimum SDK: API 21+ is recommended but osmdroid supports older versions.
- Add dependencies (Gradle, app/build.gradle), as shown in Figure 14:

```
implementation 'org.osmdroid:osmdroid-android:6.1.14'
implementation 'com.github.MKergall:osmbonuspack:6.9.0'
```

Figure 14. Dependencies for osmdroid and osmbonuspack.

- Add repository if needed (most versions are available via Maven Central).
- Add AndroidManifest permissions and features, as shown in Figure 15:

```

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

```

Figure 15. Android Manifest permissions for osmdroid and osmbonuspack.

7.3.4. Implementing Layout and Basic MapView

When setting up the layout and basic MapView, we first define the map in our activity's XML layout file. The MapView widget provided by osmdroid works like a regular Android view and can be placed inside any layout container, such as RelativeLayout or ConstraintLayout. By giving it an id, we can later reference it in our Java code. This setup makes the map occupy the entire screen or a portion of it, depending on our design. Unlike Google Maps, osmdroid does not require an API key to start showing tiles, but we must ensure that the view has proper width, height, and positioning in the layout to display correctly. Adding match_parent for both width and height is a common practice when building a full-screen map activity. So, in res/layout/activity_main.xml we have to add a MapView element, as shown in Figure 16.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <org.osmdroid.views.MapView
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>

```

Figure 16. MapView design implementation.

In activity MainActivity.java, as shown in Figure 17, we initialize osmdroid and the MapView. In the Java activity class, we then configure it for use. The first critical step is to set the user agent value via Configuration.getInstance().setUserAgentValue(getPackageName()), which ensures that tile servers recognize your app and do not block your requests. After inflating the layout with setContentView, we retrieve the MapView using findViewById, enable multi-touch controls for pinch zooming, and set a starting location and zoom level through a MapController. The map can be centered on any GeoPoint (latitude, longitude), and from there, we can interact with it like a standard map using zooming, panning, and preparing to add overlays such as markers and paths. Additionally, calling map.onResume() and map.onPause() inside the activity's lifecycle methods ensures proper resource handling when the app is paused or resumed.

```

public class MainActivity extends AppCompatActivity {
    5 usages
    private MapView map;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Important: set user agent for tile requests
        Configuration.getInstance().setUserAgentValue(getPackageName());
        setContentView(R.layout.activity_main);
        map = findViewById(R.id.map);
        map.setMultiTouchControls(true);

        MapController mapController = (MapController) map.getController();
        mapController.setZoom(15.0);
        GeoPoint startPoint = new GeoPoint(aLatitude: 37.98, aLongitude: 23.72); // e.g., Athens
        mapController.setCenter(startPoint);
    }

    @Override
    public void onResume() {
        super.onResume();
        map.onResume();
    }

    @Override
    public void onPause() {
        super.onPause();
        map.onPause();
    }
}

```

Figure 17. MapView configuration example.

7.4. QR Code Reader

7.4.1. Implementing QR Code Readers in Android with ZXing Library

QR codes (Quick Response codes) [104,105] have become a widely used method of storing information such as URLs, contact details, and product information. In Android applications, QR code scanning is often needed in e-commerce apps, payment gateways, ticket verification systems, and more. One of the most popular open-source libraries for handling QR codes is ZXing (Zebra Crossing). This library supports both generating and scanning barcodes/QR codes, making it a convenient choice for developers.

We will focus on implementing a QR code reader in Android using Java and ZXing library. We will cover project setup, permissions, integration, and code implementation.

To setup the project we have to:

- **Create an Android project** with a minimum SDK version that supports our target devices (API level 21 or above is recommended)
- **Add ZXing dependency.** ZXing is available via Gradle. Add the following dependency inside the app/build.gradle file:
 - a. implementation 'com.journeyapps:zxing-android-embedded:4.3.0'
 - b. The library (zxing-android-embedded) also provides camera integration for scanning.
- **Sync the project** to download and include the libraries.

QR code scanning requires access to the device's camera. We have to add the following permission inside AndroidManifest.xml:

```
<uses-permission android:name="android.permission.CAMERA"/>
```

Also, we have to add the camera feature declaration to inform the Play Store that our app uses the camera:

```
<uses-feature android:name="android.hardware.camera.any"/>
```

7.4.2. Creating the Scanner Activity

The zxing-android-embedded library provides a simple way to open a scanning interface using the following barLauncher. As soon as the QR code reader detects a QR code figure it translates its content and the result is forwarded to an object of Record-Loader class, which is able to communicate with our Firebase Realtime Database and load the corresponding TourGuide reference. In each QR code image we are able to store

the unique “key” value of a TourGuide. Using this technique, we are able to distinguish the records hosted in our online database. Figure 18 shows how to implement it in the Android activity.

```
public class QRCodeReader extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Reader reader = new Reader(barLauncher);
    }

    1 usage
    ActivityResultLauncher<ScanOptions> barLauncher = registerForActivityResult(new ScanContract(), result->{
        if(result.getContents() != null){ // Found QRCode from camera
            Log.d( tag: "QRCodeReader",result.getContents() );

            RecordLoader recordLoaderByKey = new RecordLoader( ctx: this);
            recordLoaderByKey.readKey(result.getContents());
        }
        finish();
    });
}
```

Figure 18. Activation of QR code reader.

The following code, in Figure 19, automatically opens the camera, scans the QR code, and returns the decoded result in the barLauncher variable. The result can be displayed in a Toast, stored in a database, or used to perform an action such as opening a webpage.

```
public class Reader{

    1 usage
    public Reader(ActivityResultLauncher<ScanOptions> barLauncher) {
        ScanOptions options = new ScanOptions();
        options.setPrompt("Volume up to flash on");
        options.setBeepEnabled(true);
        options.setOrientationLocked(true);
        options.setCaptureActivity(CaptureActivity.class);
        barLauncher.launch(options);
    }
}
```

Figure 19. Implementation of QR code reader.

To test and deploy the algorithms above, we have to:

- Run the app on a physical device (emulators generally lack camera support);
- Point the camera at a QR code;
- Verify that the scanned result matches the QR code’s content.

Implementing a QR code reader in Android using Java and the ZXing library is straightforward and efficient. With just a few lines of code, developers can integrate QR scanning capabilities into their applications. ZXing offers flexibility by providing both intent-based scanning and custom in-app scanning views. This feature is useful in real-world scenarios such as payments, authentication, product identification, and ticketing systems. By following the steps above, developers can quickly create reliable and user-friendly QR code readers in their Android applications.

7.5. Biometric Authentication Implementation in Android

In this section we will briefly show how to implement biometric authentication [106,107] (fingerprint, face, or other device-supported biometrics) in an Android app using Java. It covers setup, code examples using the AndroidX BiometricPrompt API, UX and security considerations. The examples target Android API levels where Bio-

metricPrompt is available, although AndroidX supports back to older API levels through compatibility libraries.

First, we need a device or emulator with biometric support (in emulator configure virtual sensors in AVD Manager). Then we must check the minimum SDK required since BiometricPrompt via AndroidX works on many older API levels, but some biometric features require newer Android versions. And next we need to add the AndroidX Biometric dependency in gradle as shown in Figure 20.

```
dependencies {
    implementation "androidx.biometric:biometric:1.2.0" // check latest version
}
```

Figure 20. Biometrics dependency.

Before developing the source code, we must explore some important concepts:

- **BiometricPrompt:** the recommended high-level API to show biometric UI and handle results;
- **BiometricManager:** checks whether biometric authentication is available and enrolled;
- **BiometricPrompt.AuthenticationCallback:** receives authentication results (success, failure, errors).

Checking availability

Before showing the biometric prompt, we have to confirm hardware and enroll biometrics as shown in Figure 21. We can use BIOMETRIC_STRONG for modern biometrics (Authentication allows fingerprint sensor only), BIOMETRIC_WEAK (Authentication allows all biometrics i.e., fingerprint, face, and iris) or DEVICE_CREDENTIAL, which allows fallback to device PIN/pattern.

```
BiometricManager biometricManager = BiometricManager.from(context);

int canAuthenticate = biometricManager.canAuthenticate(BiometricManager.Authenticators.BIOMETRIC_STRONG |
    BiometricManager.Authenticators.DEVICE_CREDENTIAL);

if (canAuthenticate == BiometricManager.BIOMETRIC_SUCCESS) {
    // good to go
} else {
    // fallback: ask for PIN/password or show message
}
```

Figure 21. Declaring the Types of Biometric Authentication.

Building the Biometric Prompt

To build a biometric prompt we have to create an Executor, as shown Figure 22, a BiometricPrompt.AuthenticationCallback and show the prompt. If we prefer device PIN/fallback but not a negative button, include setAllowedAuthenticators(BiometricManager.Authenticators.BIOMETRIC_STRONG|BiometricManager.Authenticators.DEVICE_CREDENTIAL) in the builder. The AndroidX Biometrics Library provides a method called setNegativeButtonText() that provides an option for using an account/app credential if the user does not want to use biometrics.

```

Executor executor = ContextCompat.getMainExecutor(this);

BiometricPrompt.AuthenticationCallback callback = new BiometricPrompt.AuthenticationCallback() {
    @Override
    public void onAuthenticationSucceeded(@NonNull BiometricPrompt.AuthenticationResult result) {
        super.onAuthenticationSucceeded(result);
        // Successful auth - proceed to protected action
    }

    @Override
    public void onAuthenticationFailed() {
        super.onAuthenticationFailed();
        // biometric matched but not recognized - keep UX friendly
    }

    @Override
    public void onAuthenticationError(int errorCode, @NonNull CharSequence errString) {
        super.onAuthenticationError(errorCode, errString);
        // handle permanent errors (e.g., lockout)
    }
};

BiometricPrompt biometricPrompt = new BiometricPrompt(this, executor, callback);

BiometricPrompt.PromptInfo promptInfo = new BiometricPrompt.PromptInfo.Builder()
    .setTitle("Sign in")
    .setSubtitle("Use your fingerprint to sign in")
    .setNegativeButtonText("Use account password") // required if not using DEVICE_CREDENTIAL
    .build();

// To show the prompt:
biometricPrompt.authenticate(promptInfo);

```

Figure 22. Setup of Biometric prompt in Android.

Handle results

In `onAuthenticationSucceeded` we have to unlock the protected content or continue the operation. We have to keep UI feedback simple (toast, navigate, unlock key material). Moreover, some security best practices techniques include the followings:

- Always provide a fallback (PIN/password) for users without biometrics enrolled;
- Do not store raw biometric data. Android never exposes raw biometric templates if we rely on platform APIs and Keystore;
- Keep prompt text clear and explain why biometric is requested;
- Handle lockouts and error codes gracefully (e.g., too many failed attempts triggers `BiometricConstants.ERROR_LOCKOUT`);
- Avoid long-running operations in the authentication callback, do minimal work and then move to the background if necessary;
- When using `setUserAuthenticationValidityDurationSeconds`, understand that short windows require re-authentication for each operation.

7.6. Text-to-Speech Implementation in Android

Text-to-Speech [108,109] adds a valuable auditory layer to Android apps, and its use cases are vast. Accessibility is one of the primary motivations for TTS. Screen readers for blind or visually impaired users rely on TTS to read UI elements, notifications, and other content. Android's native accessibility service, TalkBack, uses TTS for this purpose. Developers can enhance their apps by integrating spoken feedback for important interactions or status updates, making the app more inclusive.

Implementing Text-to-Speech in Android is straightforward thanks to the `TextToSpeech` class. Developers begin by initializing a `TextToSpeech` object, typically in an activity or service. This initialization requires a context and a listener to handle the success or failure of the TTS engine startup. Once initialized, developers can call the `speak()` method to read out any string of text.

Figure 23 shows a basic outline of how to implement TTS.

```

TextToSpeech tts = new TextToSpeech(context, new TextToSpeech.OnInitListener() {
    @Override
    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {
            tts.setLanguage(Locale.US);
            tts.speak("Hello, welcome to the app!", TextToSpeech.QUEUE_FLUSH, null, null);
        }
    }
});

```

Figure 23. TTS engine setup example.

This snippet sets up the TTS engine, specifies the language (e.g., US English), and speaks a simple greeting. The `QUEUE_FLUSH` parameter tells the engine to clear any existing queue before speaking the new text. Developers can also adjust parameters like pitch (`setPitch(float)`) and speech rate (`setSpeechRate(float)`) to better match the app's tone or target audience.

TTS can be used in various components of an Android app. For instance, an educational app can pronounce vocabulary words aloud when clicked, or a news app can read articles to the user. Navigation apps often use TTS to deliver turn-by-turn directions. It is also common in chatbots or voice assistants to read back responses to user queries. Moreover, Android supports output to either the device's speaker or a connected Bluetooth audio device. Last, TTS can be used offline, as long as the necessary language files are installed on the device. This makes it suitable for use cases where network access is limited or unavailable.

7.7. Bluetooth Beacon

7.7.1. Definition and Technology

Bluetooth Beacons are small wireless devices that broadcast signals using Bluetooth Low Energy (BLE) technology [110,111]. These devices are designed to transmit a constant identifier to nearby Bluetooth-enabled devices, such as smartphones, tablets, or wearables. The identifier, which is typically a Universally Unique Identifier (UUID) [112], can be picked up by an app or operating system, prompting location-specific actions. Unlike GPS, which is designed for outdoor positioning and consumes significant power, Bluetooth beacons are optimized for indoor environments and are known for their energy efficiency. This has enabled a wide range of applications where precise indoor navigation and contextual awareness are essential. Beacons do not send data to mobile devices directly, but rather provide proximity triggers that apps can interpret and act upon, such as displaying a notification or offering a discount when a user walks near a product in a store.

The core technology behind Bluetooth beacons relies on BLE, a protocol developed to support low-energy consumption and rapid transmission of small data packets. Beacons typically emit signals at intervals ranging from milliseconds to several seconds, depending on their configuration. These signals consist of a small payload that includes information such as the UUID, major and minor identifiers, and transmit power. The UUID identifies the beacon group, while the major and minor values allow for more precise identification within a location. Mobile devices within range listen for these signals and use the signal strength (RSSI—Received Signal Strength Indicator) to estimate proximity. This process does not require active pairing, which makes the interaction seamless and non-intrusive. Beacon hardware is often simple and durable, capable of running for months or even years on small batteries due to their low energy requirements.

The proximity UUID is used to differentiate our beacons from those in other networks that we do not manage. When we receive our beacons, they will all have proximity UUID, which will appear in the following format:

f7428da6-4fa3-7e98-8724-bc9b71e0193e

In Bluetooth beaconing, the major and minor values are used to further classify beacons, typically within a larger group identified by a unique UUID. The UUID identifies a specific network or organization, while the major and minor values help distinguish between individual beacons within that network.

7.7.2. Implementing Bluetooth Beacons Detectors in Android with Java and AltBeacon Library

This technology is widely used in retail, museums, airports, smart homes, and asset tracking systems. Bluetooth beacons use Bluetooth Low Energy proximity sensing to transmit a Universally Unique Identifier picked up by a compatible app or operating system. Beacons broadcast in regular intervals (e.g., something between 100 ms and 1 s), sending out specific data like their ID.

The ID consists of three parts:

- UUID (specific ID per device);
- Major (e.g., maximum detection distance);
- Minor (e.g., minimum detection distance).

In Android development, beacon detection requires scanning for BLE signals and interpreting them according to beacon protocols such as iBeacon, AltBeacon, UriBeacon or Eddystone. While we could implement scanning using the native BluetoothLeScanner, this involves handling raw advertisement packets, which can be complex. Instead, libraries like AltBeacon (Android Beacon Library) simplify beacon detection, ranging, and monitoring.

The AltBeacon library, created by Radius Networks, supports multiple beacon formats, provides consistent APIs, and makes it easier to build proximity-based applications in Java or Kotlin.

Firstly, we have to add a dependency in app's gradle. To use AltBeacon, you need to add its dependency in the build.gradle file as shown in Figure 24.

```
dependencies {
    implementation 'org.altbeacon:android-beacon-library:2.19.6'
}
```

Figure 24. AltBeacon dependency implementation.

Then we have to allow permissions in the application's Manifest, as shown in Figure 25. Beacons rely on Bluetooth and location permissions. Add the following to AndroidManifest.xml. Moreover, in each Android device, in order for the application to be able to work accordingly we must provide through the device's OS the permissions for both Location and Nearby Devices detection. Location is only necessary if we want our device to be detected.

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
```

Figure 25. Bluetooth Beacon Manifest implementation.

- **ACCESS_FINE_LOCATION** is required because beacon scanning is tied to location permissions on Android;
- **ACCESS_BACKGROUND_LOCATION** is necessary if you want scanning to work even when the app is in the background (Android 10+).

Then we have to extend the Application class and set up the beacon manager. In this step, we must create a custom Application class that initializes and configures the AltBeacon library as soon as the app starts. By extending the Application class, you can set up a BeaconManager instance globally so it runs in the background and is not tied to any single activity, as shown in Figure 26. Inside the onCreate() method, we register the beacon parser (to define the beacon protocol format, such as iBeacon or AltBeacon), adjust scan intervals for performance, and define one or more regions that specify which beacons to monitor. We also implement the BootstrapNotifier interface to receive callbacks when the device enters, exits, or changes state relative to a beacon region. This setup ensures that beacon detection begins automatically when the app is launched, without requiring the user to open a specific activity. It also enables background detection, which is critical for real-world scenarios like retail notifications or proximity-based triggers.

```
public class MyApplication extends Application implements BootstrapNotifier {
    private BeaconManager beaconManager;
    private RegionBootstrap regionBootstrap;

    @Override
    public void onCreate() {
        super.onCreate();

        beaconManager = BeaconManager.getInstanceForApplication(this);

        // iBeacon Layout
        beaconManager.getBeaconParsers().add(new BeaconParser().
            setBeaconLayout(BeaconParser.ALTBEACON_LAYOUT));

        // Set scanning interval
        beaconManager.setForegroundScanPeriod(1100L);
        beaconManager.setForegroundBetweenScanPeriod(0L);

        // Define a region to monitor (UUID can be changed to your beacon's UUID)
        Region region = new Region("myRegionId", null, null, null);
        regionBootstrap = new RegionBootstrap(this, region);
    }

    @Override
    public void didEnterRegion(Region region) {
        // Called when a beacon is detected
        System.out.println("Beacon detected!");
    }

    @Override
    public void didExitRegion(Region region) {
        // Called when beacon is lost
        System.out.println("Beacon exited!");
    }

    @Override
    public void didDetermineStateForRegion(int state, Region region) {
        // Called when state changes between inside/outside
    }
}
```

Figure 26. Bluetooth Beacon Manager creation.

- BeaconParser defines the protocol layout (e.g., iBeacon, AltBeacon);
- Region specifies which beacons to look for (you can filter by UUID, major, and minor, or leave them null for all beacons);
- RegionBootstrap allows scanning to start when the app launches, even if the UI is not running.

7.8. Firebase

7.8.1. Firebase and Its Role in Android Development

Firebase was a company that developed backend software. It was founded in San Francisco in 2011 [113–115]. In 2014, Firebase was bought by Google. Now Firebase is a comprehensive platform maintained by Google that provides a suite of cloud-based services to help developers build, improve, and grow Android applications. Designed with mobile-first development in mind, Firebase offers backend services such as real-time databases, user authentication, cloud storage, analytics, and messaging. All accessible through simple SDKs and APIs.

For Android developers, Firebase eliminates much of the traditional complexity associated with backend development. Instead of setting up custom servers or writing backend logic from scratch, developers can focus on building powerful front-end experiences while Firebase handles data storage, synchronization, and server-side logic. Its tight integration with the Android SDK and Google Cloud infrastructure makes it highly scalable, secure, and efficient.

Firebase also supports real-time updates, offline capabilities, and cross-platform synchronization (Android, iOS, web), making it ideal for collaborative or dynamic applications such as chat apps, multiplayer games, and live tracking services. Whether you are building a simple utility app or a feature-rich platform, Firebase can act as a full backend-as-a-service (BaaS) solution.

The main key features of Firebase Realtime Database include:

- **Real-time synchronization:** Any update is propagated instantly to all connected devices;
- **Offline support:** Data is cached locally and synchronized once the device reconnects;
- **Cross-platform support:** Works on Android, iOS, and web applications;
- **Security:** Access is controlled using Firebase Authentication and Security Rules.

7.8.2. Implementing Android Application with Firebase Realtime Database

To use Firebase Realtime Database in an Android project, several setup steps are required:

- **Create a Firebase Project (on Google's Firebase Servers)**
 - a. Go to the Firebase console;
 - b. Create a new project and enable Realtime Database;
 - c. Configure rules (by default, it may allow public access, but security rules should be set before production).
- **Add Firebase SDK to Android App**
 - a. Download google-services.json from the Firebase console;
 - b. Place it in the app/ directory of the Android project;
 - c. Update the root build.gradle file.

The first thing we need to do is go to <https://firebase.google.com/> and make an account to gain access to their console (we need to already have an active Gmail Account). After we gain access to the console, we can start by creating our first project. Give the package name of your project (e.g., db.personaltourguide.firebaseio.com) in which we are going to integrate Firebase. Here, the google-services.json file will be downloaded when we press add app button.

Next, in the Android Studio IDE, we need to create a new project. While filling the project details, use the same package name that we gave in the Firebase console. Next, paste the "google-services.json" file to our project's app folder. This step is very im-

portant as our project will not build without this file. Now open the “build.gradle” located in the project’s home directory and add the Google Play Store dependency as shown in Figure 27.

```
classpath 'com.google.gms:google-services:4.3.15'
```

Figure 27. Gradle dependency for Google Services.

Then, open “app/build.gradle” and add the Firebase Database dependency. At the very bottom of the file, add plugin with id: “com.google.gms.google-services” and apply the following implementations in the dependencies section as shown in Figure 28.

```
implementation 'com.google.firebase:firebase-database:20.3.0'
implementation 'com.google.firebase:firebase-auth:22.3.1'
```

Figure 28. Implementing Firebase dependencies.

7.8.3. Performing Read Operations in Firebase Realtime Database

An example of the JSON tree structure that we use in our application is shown in Figure 29. Under the main node we host the nodes Aquariums, Cities, Museums, Nature, Religion and Ruins. Each of these options contains nodes entitled with the names of the places of interest. Each place (e.g., Museum Benaki) contains pairs of data concerning a specific Tour Guide. In order to distinguish each Tour Guide from the rest of the collective each entry is equipped with three different identification tags. The “id” node is used only internally in our database. The “key” node is used externally in case we want to identify a place or an object using a QR Code. Last, the “uuid” node is used in case we want to identify a place or an object using Bluetooth Beacon technology.

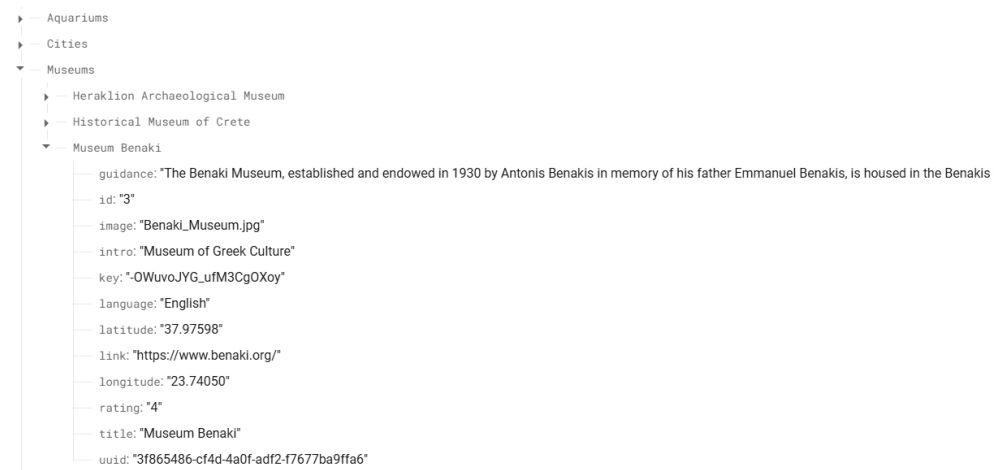


Figure 29. Firebase JSON example.

Using this approach in our application we have to create a new class named “TourData” that contains the corresponding references for all nodes hosted online in Firebase. This means that our class must provide internal variables matching the node names of the information hosted in Firebase and the corresponding Java class constructors. In this case, we must provide at least two constructors. One must be able to create a local instance using all available fields and the empty constructor. A non-complete template of the TourData class is shown in Figure 30.

```

public class TourData implements Serializable{
    4 usages
    private String id;
    4 usages
    private String title;
    4 usages
    private String intro;
    4 usages
    private String guidance;
    4 usages
    private String link;
    4 usages
    private String image;
    5 usages
    private String rating;    // integer value
    4 usages
    private String latitude;    // latitude, double value
    4 usages
    private String longitude;    // longitude, double value
    4 usages
    private String language;    // tour guide language
    4 usages
    private String key;    // unique key for qrcode identification
    4 usages
    private String uuid;    // universally unique identifier, only for bluetooth beacons when they are available

    no usages
    public TourData() {
    }

    no usages
    public TourData(String id, String title, String intro, String guidance, String link, String image, String rating,

```

Figure 30. Content of the TourData class.

Read operations in Firebase form the foundation of its real-time functionality. Data can be retrieved once or continuously through listeners. The `get()` method allows developers to read data at a specific path one time, similar to a typical database query. However, the more powerful feature is the `ValueEventListener`, which attaches a real-time listener to a node and automatically receives updates whenever the data changes. This event-driven model allows Android apps to stay synchronized without polling, which is essential for features like live feeds, dashboards, or chat. While reads are straightforward, they require careful consideration of data size. Since Firebase downloads entire JSON nodes by default, reading deeply nested or large datasets can consume significant bandwidth and increase costs. Developers often mitigate this by structuring their data in flatter formats, limiting queries with methods such as `orderByChild()`, `startAt()`, or `limitToFirst()`, and ensuring only the necessary portion of the data tree is fetched. Offline persistence further enhances read operations by allowing cached data to be accessed even when the device is disconnected, with changes synchronized once connectivity resumes.

To read data, we need to attach the `ValueEventListener()` to the database reference. This event will be triggered whenever there is a change in data in real time. In `onDataChange()` we can perform the desired operations on the new data. Below, in Figure 31, is the event listener that is triggered whenever there is a change in user profile data that we created earlier.

```

mDatabase.child(userId).addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {

        User user = dataSnapshot.getValue(User.class);

        Log.d(TAG, "User name: " + user.getName() + ", email " + user.getEmail());
    }

    @Override
    public void onCancelled(DatabaseError error) {
        // Failed to read value
        Log.w(TAG, "Failed to read value.", error.toException());
    }
});

```

Figure 31. Firebase data change listener.

8. Results and Evaluation

The final results of the Android application represent the culmination of the entire development process, where all features, design elements, and functionalities are integrated into a complete product. At this stage, the application is tested to ensure it meets the original requirements, works smoothly across different devices, and provides an intuitive user experience. The final version showcases a polished interface, responsive navigation, and stable performance, reflecting the goals set during the planning and design phases. This stage is where we confirm that key features such as login, registration, data handling, and any unique functionality operate as intended without major issues.

The proposed bio-cultural tourist guiding platform was evaluated based on five perspectives: effectiveness, functionality, usability, performance, and compatibility. The objective was to determine whether the integrated system could reliably support real-time navigation, multimedia content delivery, QR code interaction, Bluetooth beacon detection, and audio guidance in realistic tourism scenarios.

8.1. Final User Interface Implementation Results

The final result for the main window is shown in Figure 32. As previously designed and presented, the mockup for this window was enriched with a toolbar providing different types of functionality at any time to the final user. Moreover, using application revealed the need for a centralized hidden navigation bar, which again must be present at any time while using the application. On the menu button click event by users, this navigation bar will become visible. These facts drove us to the final implementation shown below, in which we present the final result with and without the navigation bar. The initial visual design ratios were kept and the use of the Card widget was really helpful since it gave us the ability to produce a pleasant, user-friendly, adaptive and attractive result. Each presented option can activate the corresponding activity.

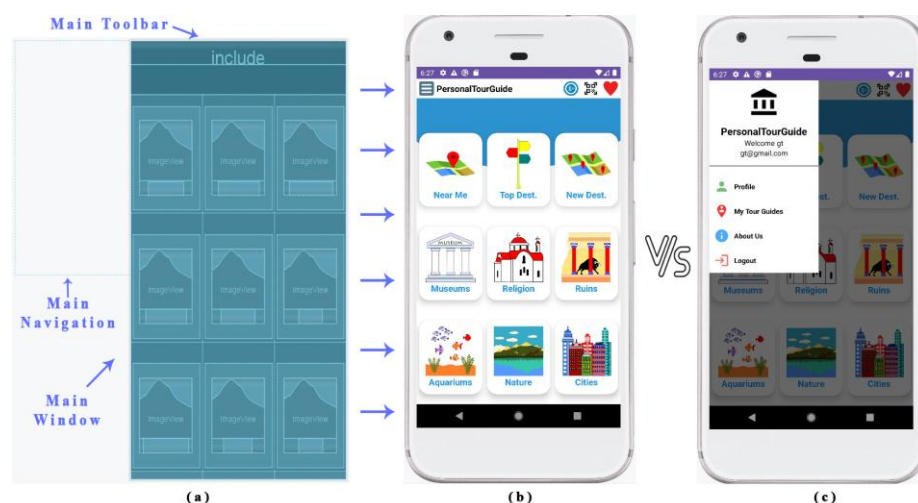


Figure 32. (a) Main window Blueprint. (b) and (c) show Main window design, with and without Navigation Menu respectively.

The main toolbar (shown in Figure 32) becomes important and necessary since we want users to always and instantly be able to (from left to right) activate the menu, navigate to the main window by pressing PersonalTourGuide, activate the Bluetooth Beacon detector, activate the QR code reader and visit the favorite tour guides list.

In Figure 33, we can see the blueprint and the final result of our design for the Tour Guide activity. The design reveals for each selected guide its image, the title, a small script which serves as the introduction, the rating value and the guidance text itself.

Moreover, user can select if the current guide must be appended in his favorites list or not (pressing the heart icon on the left-centered icon) and navigate to this destination (pressing the location icon). At the bottom of the window, user can choose either to activate the Text to Speech service or to stop it. If the Text to Speech service is activated, then another thread will wake up highlighting the section of the text that is currently being read.

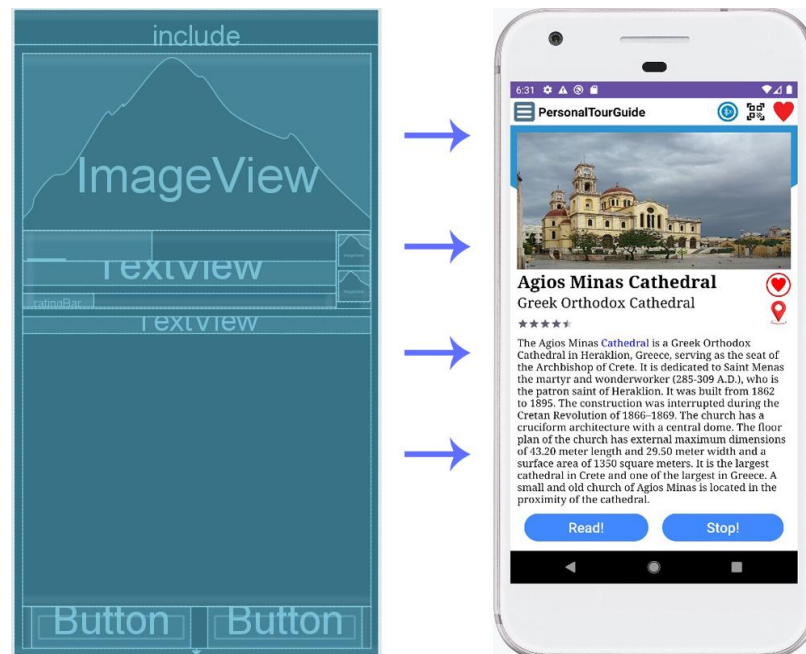


Figure 33. Tour Guide window design.

Using the main toolbar users can choose to activate the QR code reader, as shown in Figure 34 in option (a), find a possible route from their current position to the selected guide destination, as shown in option (b). Moreover, using the Near Me option in the main application window, as shown in option (c), user can investigate possible destinations around him and choose one of them as his next guidance.

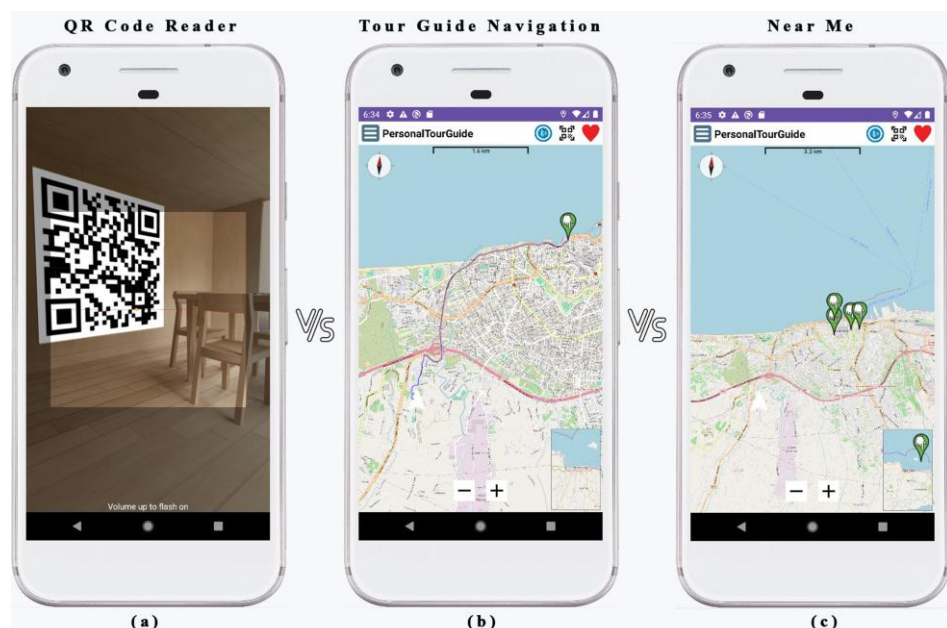


Figure 34. Main Services design: (a) QR Code Reader. (b) Maps for user routing and (c) finding nearby possible points of interest as destinations.

8.2. Functional Evaluation

The implementation successfully integrated all planned subsystems into a single Android application. The platform supports user authentication, guide retrieval from Firebase, GPS-based navigation through OpenStreetMap, QR code identification, Bluetooth beacon detection, Text-to-Speech narration, and multimedia presentation. Testing confirmed that these components operated correctly both independently and as a unified guiding system. Users were able to retrieve contextual information, navigate to points of interest, activate audio narration, and access location-specific content through QR codes and Bluetooth beacons without requiring additional hardware. The results therefore demonstrate the technical feasibility of combining multiple location-aware and multimedia technologies within a single mobile tourism application.

8.3. Usability Evaluation

To evaluate the usability of the application in real conditions, we conducted a survey at the Hellenic Mediterranean University of Greece. About one hundred students took part in the survey, and about eighty agreed to provide answers. The survey results, shown in Appendix C (Appendices C.1–C.4, Figures A11–A14), indicate very high overall user satisfaction with the mobile app's usability and performance. Most respondents rated the installation and initial setup process as Good to Excellent, with nearly all confirming that installation was clear and successful. Core usability aspects, such as ease of learning, navigation, feature discovery, and interface clarity received consistently strong ratings, with the majority selecting Very Easy or the highest score on the 5-point scale. Labels, instructions, and terminology were widely perceived as clear and consistent, suggesting that the app's design supports intuitive use even for first-time users.

8.4. Performance Evaluation

In terms of performance and reliability, feedback was similarly positive. Most users rated the app's speed and responsiveness as Excellent and described interactions as instant, with minimal reports of lag, bugs, or technical issues. Crashes and unexpected behavior were reported as rare or nonexistent, and users generally trusted the accuracy of the app's results. While a small number of respondents did indicate occasional difficulties or technical issues, these were not widespread enough to significantly impact overall satisfaction. Taken together, the results suggest that the app is stable, user-friendly, and well-optimized, though future evaluations could benefit from deeper qualitative feedback to uncover subtle issues that may not be reflected in largely positive quantitative responses.

Evaluating CPU and Memory consumption, as Figure 35 shows, we can see that in the case of a standby situation on Main Activity, case (a), the consumption is negligible. We can also notice the same results in case (b) and (d), which are the cases where the application is actively executing a Tour Guide procedure or in (d) where the application is searching for a Bluetooth Beacon. CPU consumption reaches 10–12% during image processing while searching for a QR Code in case (c) and memory increases by 10–15 MB. CPU consumption rises heavily and rapidly in case (e), while activating osmdroid maps as application reaches 32–35% and memory consumes almost 150 MB. Case (e) is considered to be the most heavy-duty case but is not used that often and not for long periods of time.

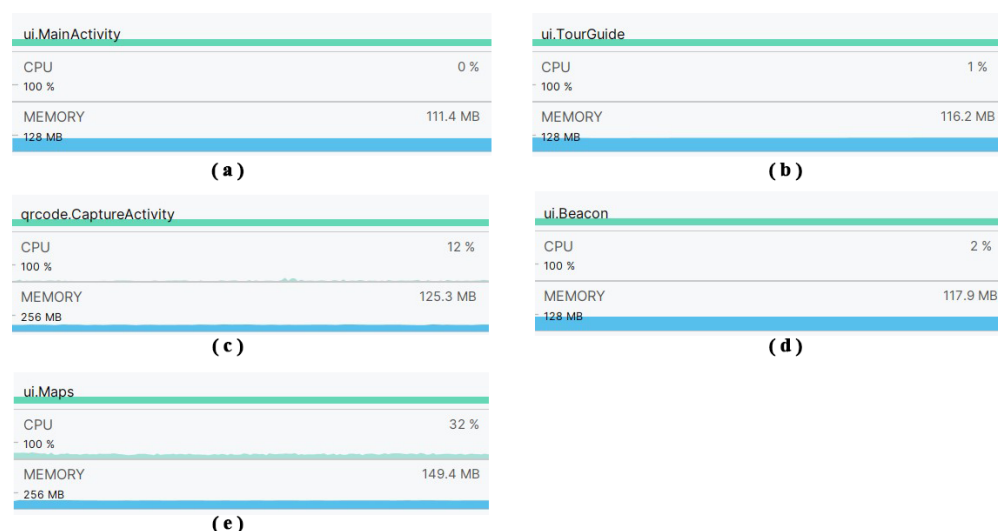


Figure 35. Main Services Memory and CPU consumption: (a) Main window, (b) during Tour Guide activity, (c) while searching through camera for QR code, (d) during search for Bluetooth Beacon, (e) during online Map navigation.

On the network side, the heaviest function is the procedure of loading maps. As Figure 36 shows, map loading involves the procedure of loading tiles in “png” files using http GET requests. Downloaded file sizes differ from 20 KB to 300 KB and response time usually varies from 300 ms to 1000 ms depending on network traffic and bandwidth hotspots.

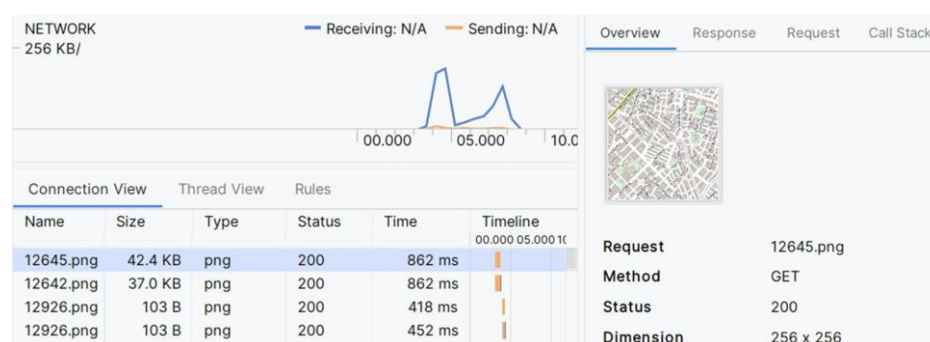


Figure 36. Mobile Network Traffic Monitoring.

Overall performance measurements focused on CPU utilization, memory consumption, and network activity during the execution of the platform’s core services. The results showed that normal application operation, including guide presentation and Bluetooth beacon detection, imposed only a small computational burden on the device. QR code scanning generated slightly higher processing requirements because of continuous camera-based image analysis, increasing CPU usage to approximately 10–12% and memory consumption by approximately 10–15 MB. The most demanding operation was map-based navigation using OpenStreetMap services. Despite these higher requirements, navigation functions remained responsive and stable during testing.

On the other hand, Network monitoring demonstrated that map loading represented the largest source of network traffic. These results indicate that cloud-based operation is practical under normal mobile network environments and that the application can support real-time content delivery without excessive bandwidth requirements.

8.5. Device Compatibility Evaluation

An important objective of the study was to verify that the platform could operate across a broad range of Android devices. Experimental testing was conducted on high-end smartphones, mid-range devices, low-end devices, and Android emulators. The results demonstrated that the application maintained stable operation and reliable synchronization across all tested configurations. Core services, including navigation, multimedia presentation, audio guidance, and cloud-based content retrieval, functioned consistently regardless of hardware specifications.

The proposed platform was successfully tested across a wide range of Android devices, including both high-end smartphones with advanced hardware capabilities (like Xiaomi Redmi Note 13 Pro 5G with Snapdragon 7 s Gen 2, 8 GB RAM, Android 15), mid-range devices (like Xiaomi Redmi Note 7 with Snapdragon 660, 4 GB RAM, Android 10), low-end devices with limited processing power and memory resources (like Samsung S3 Neo with Snapdragon 400, 1.5 GB RAM, Android 4.4) and also in different Android Studio Emulators like Pixel. The minimum compatible Android SDK version tested was Level 19 Android 4.4. In reality, we have used Level 29 (Android 10) as the minimum SDK version and Level 34 (Android 14) as the target SDK. Experimental testing demonstrated that the application maintained stable performance, responsive user interaction, and reliable data synchronization on high-end devices, while also operating effectively on low-end devices without critical performance degradation. These results indicate that the platform's lightweight architecture, efficient resource management, and modular design enable broad device compatibility, making it suitable for deployment in diverse user environments and ensuring accessibility for a wide spectrum of users regardless of device specifications.

8.6. Overall Findings

Finally, the implementation results demonstrate that the proposed bio-cultural tourist navigation and guiding system is technically robust and operationally effective within real-world mobile environments. The modular Android-based architecture, combined with Firebase cloud services and OpenStreetMap navigation, enabled reliable real-time data retrieval, location-aware guidance, and multimedia content delivery with low latency. Analytical evaluation of system behavior shows that the integration of augmented reality overlays and audio guidance significantly improves spatial comprehension and user orientation compared to conventional map-based or text-based guiding approaches. The system's ability to dynamically associate digital content with geographic coordinates, QR codes, and Bluetooth beacons confirms the feasibility of deploying scalable AR guiding solutions without reliance on specialized hardware. Furthermore, UML-based modeling and the formal data validation process ensured structural consistency, reducing runtime errors and simplifying future system extensions.

From a broader perspective, the results indicate that the proposed platform effectively supports sustainable and inclusive tourism objectives. By eliminating the continuous need for physical tour guides while preserving high-quality interpretive content, the system reduces operational costs and environmental pressure associated with mass tourism activities. The multimodal interaction design (visual, audio, and location-based cues) enhances accessibility for users with diverse preferences and abilities, contributing to a more inclusive visitor experience. Analytical discussion of system outcomes suggests that such digital guiding solutions can assist policymakers and cultural authorities in managing visitor flows, protecting sensitive sites, and promoting underrepresented locations. Overall, the results validate the platform's potential as a practical, adaptable, and socially responsible tool for next-generation bio-cultural tourism, while also highlighting

a clear pathway for future enhancements such as personalization algorithms and AI-driven recommendation mechanisms.

User feedback confirmed the usability of the application, while performance measurements showed that resource consumption remained within acceptable limits for modern mobile devices. Although map rendering remains the most resource-intensive operation, the overall results indicate that the platform is suitable for real-world deployment as a scalable and cost-effective digital tourist guiding solution [116].

9. Discussion

This study contributes to the growing body of research on digital and augmented reality-based cultural tourism by proposing and implementing an integrated bio-cultural tourist navigation and guiding platform that addresses several limitations observed in previous systems. Earlier designs in the literature often focus on isolated functionalities, such as static mobile guides, museum-centric augmented reality experiences, or generic navigation tools, with limited integration between spatial guidance, interpretive content, and real-time data management. In contrast, the proposed system adopts a holistic, modular architecture that combines augmented reality guidance, location-aware navigation, cloud-based data storage, and multimodal interaction within a single lightweight Android application. This unified design enables seamless transitions between navigation, interpretation, and user interaction, while maintaining compatibility with both high-end and low-end mobile devices. Unlike many prior solutions that rely on specialized hardware or fixed installations, the proposed platform emphasizes accessibility, scalability, and ease of deployment, making it suitable for broader adoption in real-world tourism environments.

Another key distinction lies in the system's adaptability to diverse bio-cultural contexts. While many existing applications are developed for specific sites or narrowly defined use cases, the modular design and flexible data model of the proposed platform allow it to be applied across a wide range of scenarios, including urban cultural heritage sites, rural eco-tourism destinations, archeological parks, museums, and mixed natural-cultural landscapes. By decoupling content management from core application logic and leveraging cloud-based services, the system can be easily customized with new routes, points of interest, languages, and multimedia resources without requiring significant redevelopment. This adaptability supports use cases such as self-guided city tours, thematic cultural routes, biodiversity interpretation trails, and educational field trips, demonstrating the platform's potential as a reusable digital infrastructure rather than a single-purpose application. Moreover, the inclusion of multimodal interaction (visual AR overlays, audio guidance, and map-based navigation) enhances accessibility for users with different preferences and abilities, extending the system's applicability across diverse visitor groups.

Despite these advantages, several challenges and limitations remain and should be acknowledged. Environmental factors such as variable lighting conditions, GPS inaccuracies, limited network connectivity, and crowded settings can affect the precision and stability of augmented reality overlays and location-based services. Device-level constraints, particularly on low-end smartphones, may also limit rendering performance or battery life during prolonged use, even though the system was designed to minimize resource consumption. From a content perspective, the effectiveness of the platform depends heavily on the quality, accuracy, and cultural sensitivity of the interpretive material provided, highlighting the need for collaboration with cultural experts, local communities, and heritage authorities. Privacy and data protection concerns related to location tracking and cloud-based data storage further emphasize the importance of transparent data governance and compliance with relevant regulations.

Future development of the platform can address these challenges and extend its capabilities in several directions. The integration of artificial intelligence techniques could enable adaptive content personalization, intelligent recommendation of routes, and predictive visitor flow management. Advances in computer vision and sensor fusion may improve tracking accuracy and robustness under challenging environmental conditions. Support for offline or hybrid operation modes could enhance usability in areas with limited connectivity, while expanded analytics tools could provide destination managers with insights into visitor behavior and system performance. Finally, longitudinal user studies and comparative evaluations against existing guiding solutions would strengthen empirical validation and support evidence-based refinement of the platform. Overall, the proposed system represents a meaningful step toward scalable, inclusive, and sustainable digital solutions for bio-cultural tourism, while also opening clear pathways for future research and technological advancement.

10. Conclusions

This publication designs, implements, and evaluates a bio-cultural guiding system that combines Android mobile technologies and cloud-based infrastructure to enrich users' experiences in cultural, natural, and educational contexts. The research addressed both technical and conceptual challenges of delivering a platform capable of operating efficiently on mobile devices, adapting dynamically to user preferences, and supporting sustainable cultural engagement.

The system architecture was successfully implemented using Firebase Realtime Database, Bluetooth beacons, QR code scanning, and Text-to-Speech services, all integrated within a modular Android application. Evaluation demonstrated that resource usage remained well within acceptable limits under typical operating conditions, with only map-loading functions showing higher CPU and Ram Memory demands. Network monitoring confirmed predictable traffic patterns, validating the scalability of the platform in real-world settings.

Beyond technical contributions, this publication demonstrated value across multiple domains. In tourism and cultural heritage, it offers a scalable alternative to human guides. In education, the augmented reality and Text-to-Speech services make knowledge more accessible and interactive. In sustainability and policy, the platform can support smarter management of natural and cultural resources.

The findings underscore three main implications:

- **Technical Feasibility:** Mobile-centric guiding systems can integrate personalization without imposing prohibitive computational or network costs.
- **User Experience:** Blending multimodal inputs (visual, audio, AR) with adaptive outputs creates inclusive, engaging, and context-aware journeys.
- **Social Value:** Such systems can act as bridges between technology, culture, and ecology, encouraging respectful and sustainable interaction with both people and places.

Despite its promise, several limitations must be acknowledged. The reliance on Firebase introduces cost and vendor lock-in concerns, while its limited querying capabilities may constrain large-scale deployments. Moreover, the high computational demands of map rendering remain a bottleneck for performance in less powerful devices.

In conclusion, this research provides a comprehensive framework for a next-generation cultural guiding system. One that is technically robust, contextually adaptive, and socially relevant. The results highlight that the future of guiding systems lies not merely in delivering information, but in shaping meaningful, personalized experiences that honor both human diversity and environmental responsibility.

Author Contributions: Conceptualization, G.T. and N.P.; methodology, G.T., G.V. and N.P.; software, G.T.; validation, G.T., G.V., A.G., M.R., A.P., H.T., C.D.W., A.M., E.T. and N.P.; formal analysis, G.T., G.V. and N.P.; investigation, G.T.; resources, G.T., G.V., M.R. and N.P.; data curation, G.T., A.G., M.R., A.P., H.T., C.D.W., A.M., E.T. and N.P.; writing—original draft preparation, G.T.; writing—review and editing, G.T., G.V. and N.P.; visualization, G.T. and M.R.; supervision, N.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: This article’s content is based on the Ph.D. thesis of George Tsamis, The design, architecture, implementation and artificial intelligence support of a smart bio and cultural guiding system, Hellenic Mediterranean University (HMU), Heraklion, Greece, 10/2025

Conflicts of Interest: The authors declare no conflicts of interest. Author Charles D. White was employed as Technical Presales Engineer and IT Infrastructure Project Manager by the company Westpole Belgium. Westpole has been informed of the author’s collaboration, and no company or client information was used or shared during the writing of the article. The author’s contribution was limited to validation and data curation, and no financial or other agreements were made which can be construed as a conflict as interest. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

OS	Operating System
ART	Android Runtime
HAL	Hardware Abstraction Layer
OEM	Original Equipment Manufacturer
AOSP	Android Open-Source Project
AR	Augmented Reality
IMU	Inertial Measurement Unit
GPS	Global Positioning System
SLAM	Simultaneous Localization and Mapping
AI	Artificial Intelligence
IP	Internet Protocol Address
UUID	Universally Unique Identifier
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CPU	Central Processing Unit
GDP	Gross Domestic Product
VR	Virtual Reality
UX	User Experience
OHA	Open Handset Alliance
MVC	Model View Controller
UCD	User-Centered Design
HTTPS	Hypertext Transfer Protocol Secure
UML	Unified Modeling Language
JSON	JavaScript Object Notation
RESTful	Representational State Transfer
API	Application Programming Interface
CRUD	Create, Read, Update, and Delete

GIS	Geographic Information Systems
OSM	OpenStreetMap
ODbL	Open Database License
POI	Point of Interest
OSRM	Open-Source Routing Machine
QR	Quick Response
URL	Uniform Resource Locator
TTS	Text-to-Speech
BLE	Bluetooth Low Energy
RSSI	Received Signal Strength Indicator
BaaS	Backend-as-a-Service
CAGR	Compound Annual Growth Rate

Appendix A

Appendix A.1

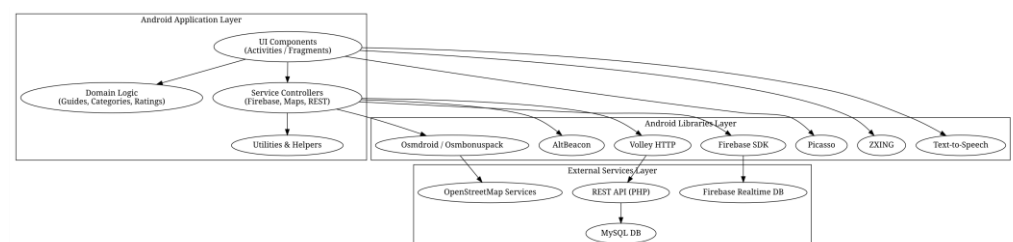


Figure A1. Component-based UML architecture of the Personal Tour Guide Application and Web Platform.

Appendix A.2

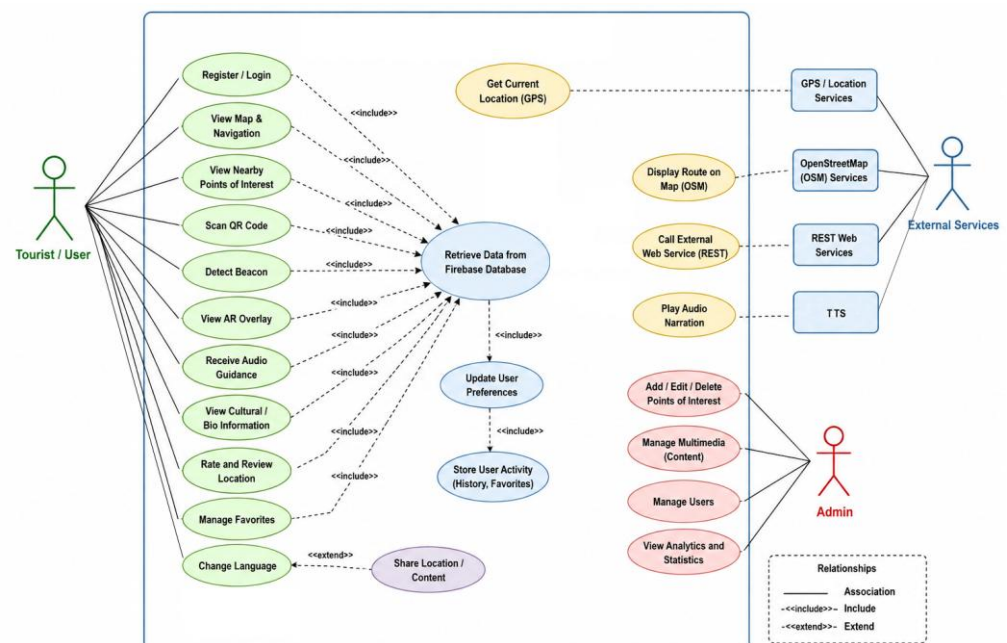


Figure A2. Use Case UML diagram of the Personal Tour Guide Application and Web Platform.

Appendix A.3

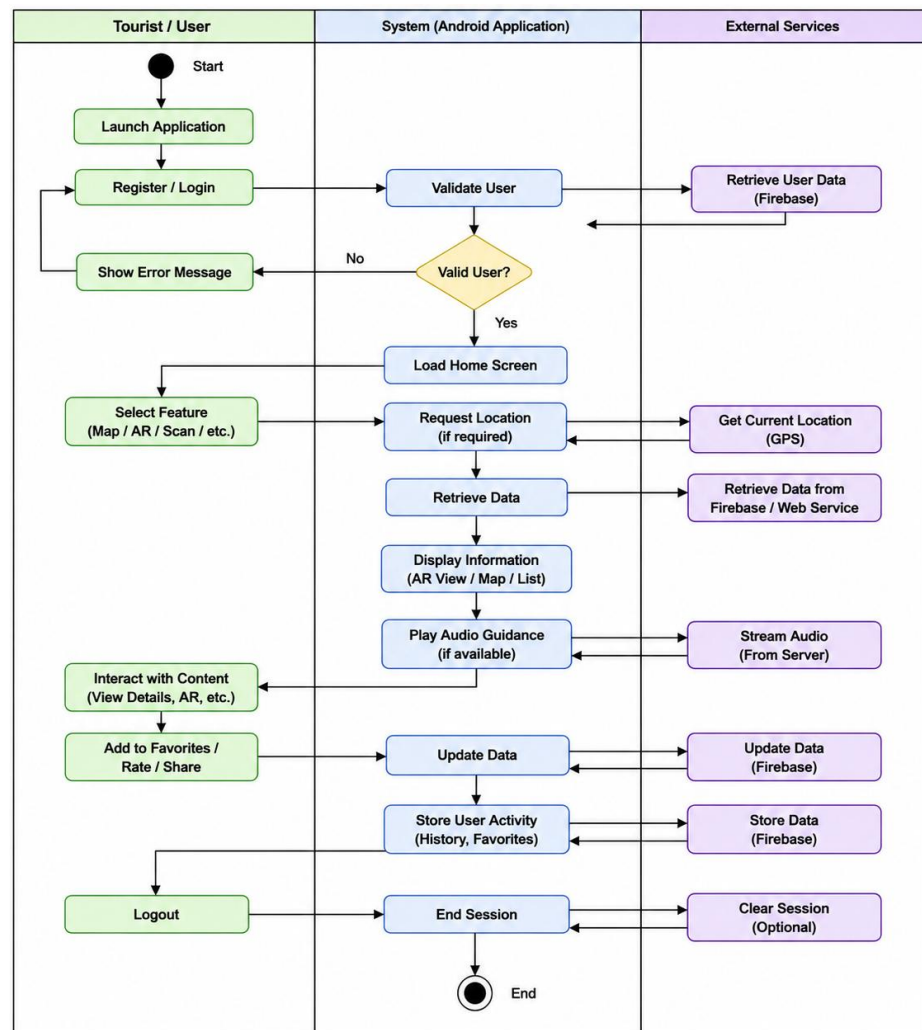


Figure A3. Activity UML diagram of the Personal Tour Guide Application and Web Platform.

Appendix A.4

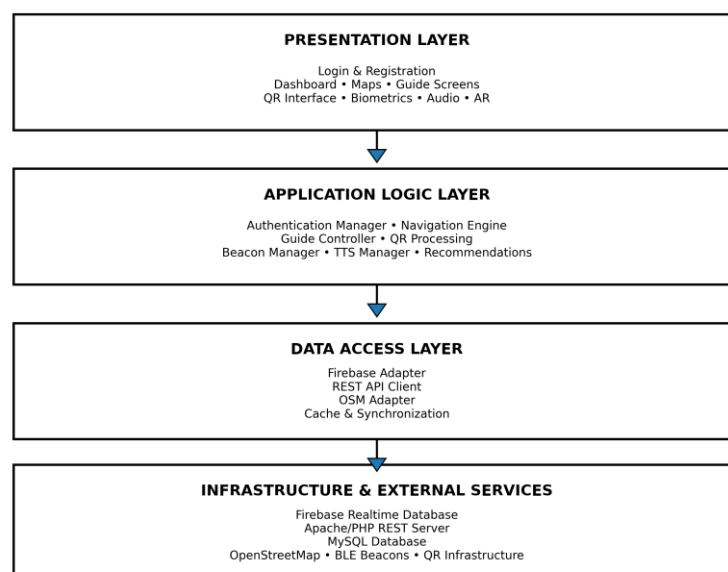


Figure A4. Layered System Architecture of the Personal Tour Guide Application.

Appendix A.5

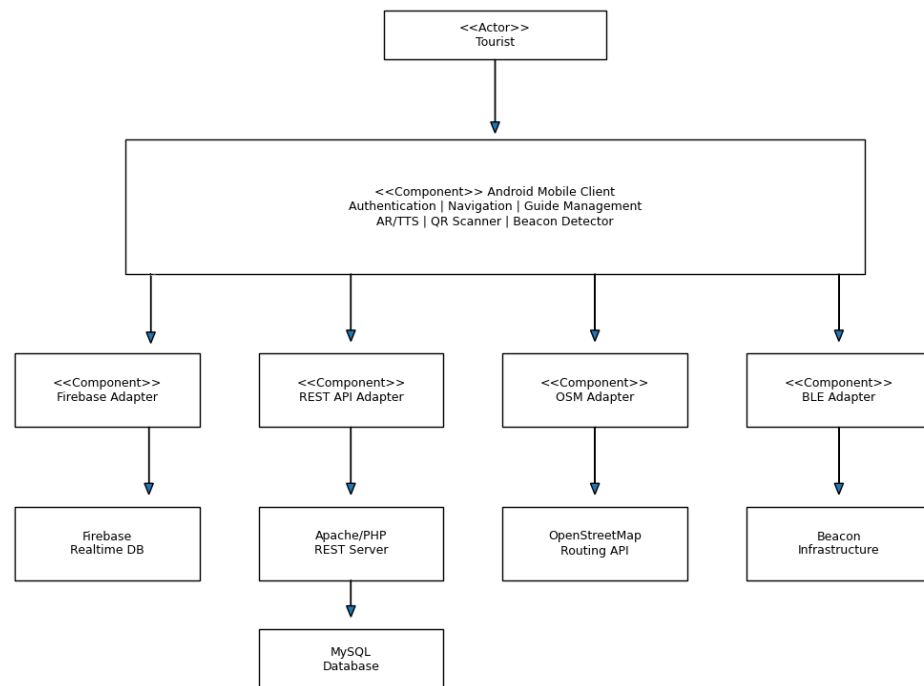


Figure A5. UML Component diagram of the Personal Tour Guide Application.

Appendix A.6

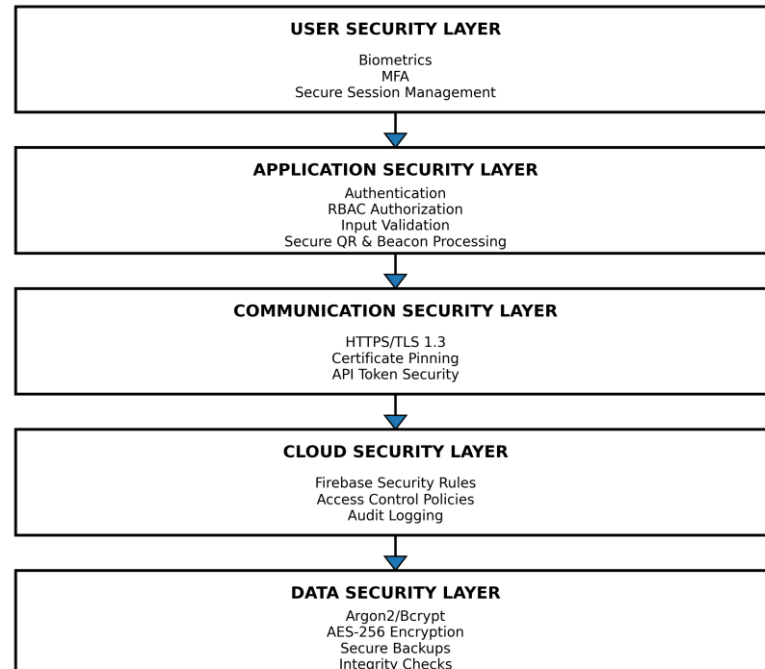


Figure A6. Security Architecture overview.

Appendix B

Appendix B.1

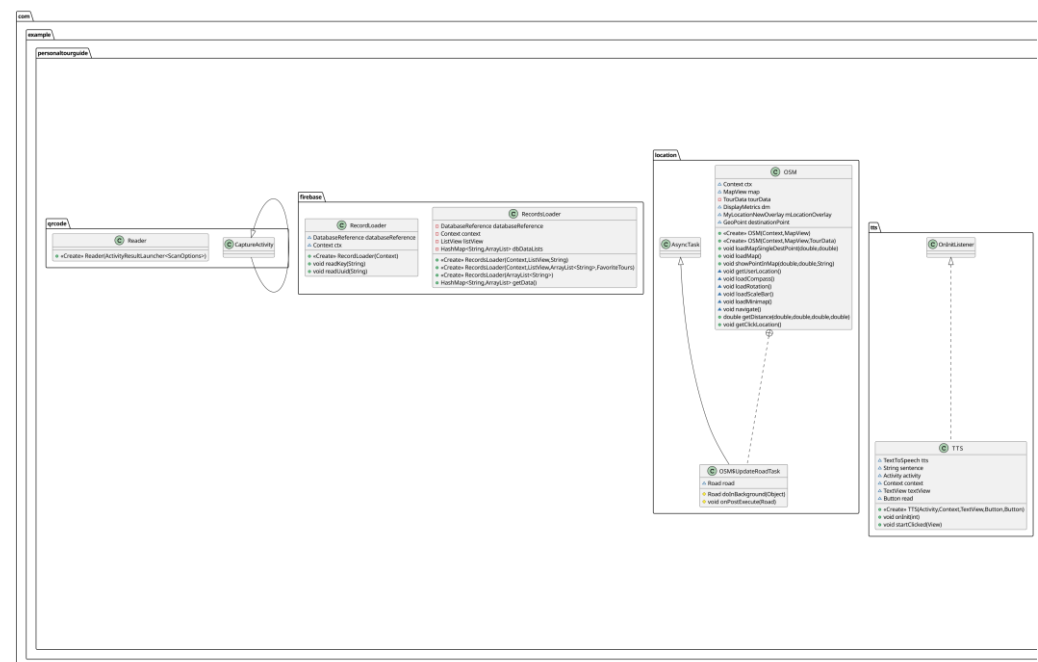


Figure A7. Class diagram, first section.

Appendix B.2

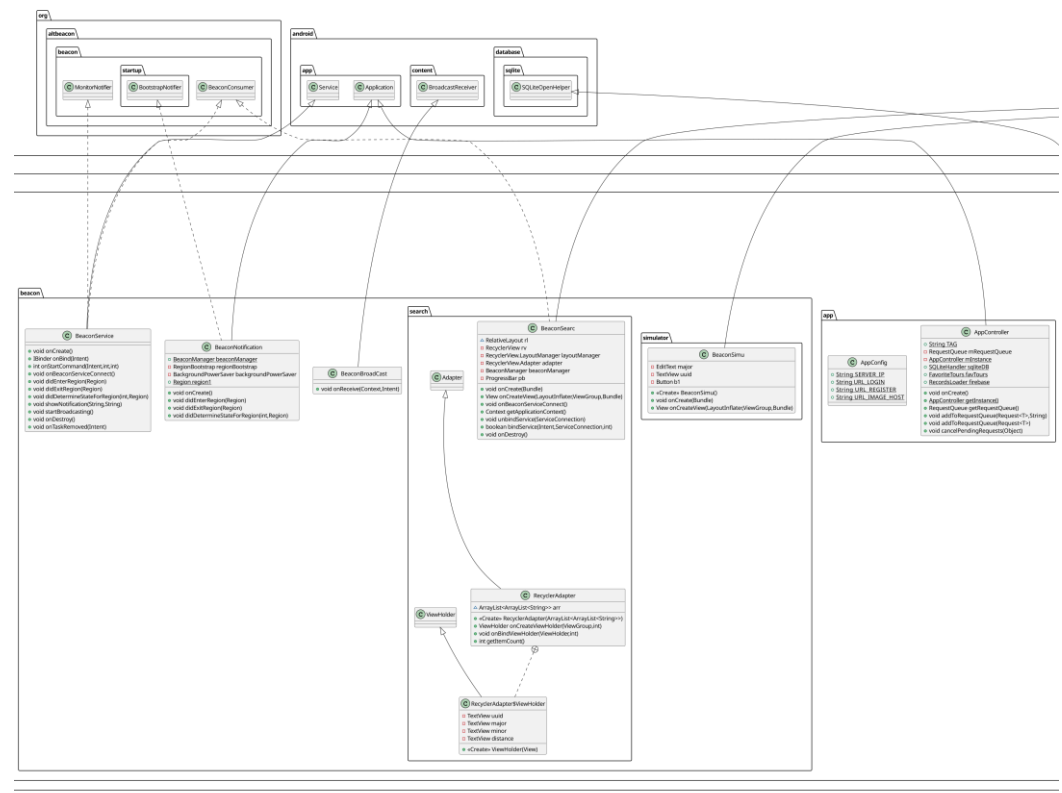


Figure A8. Class diagram, second section.

Appendix B.3

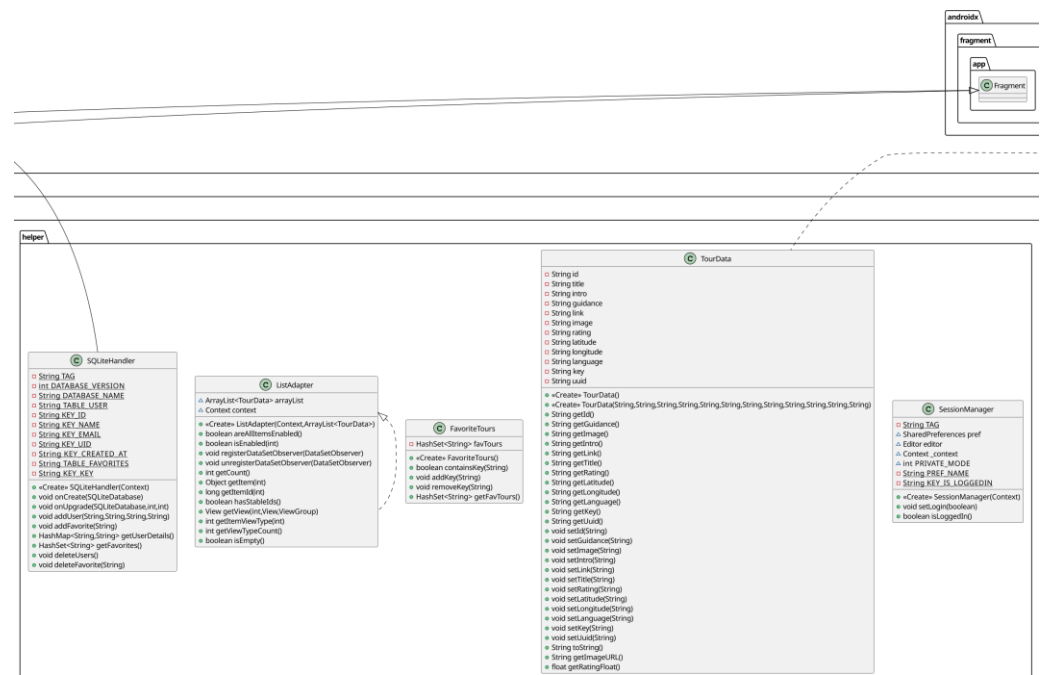


Figure A9. Class diagram, third section.

Appendix B.4

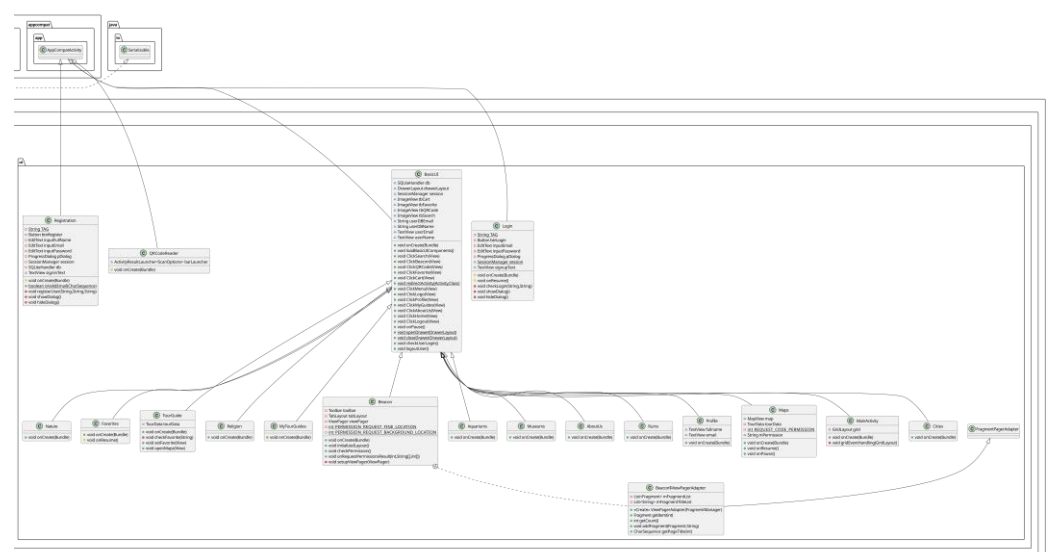


Figure A10. Class diagram, fourth section.

Appendix C

Appendix C.1

Basic Operations

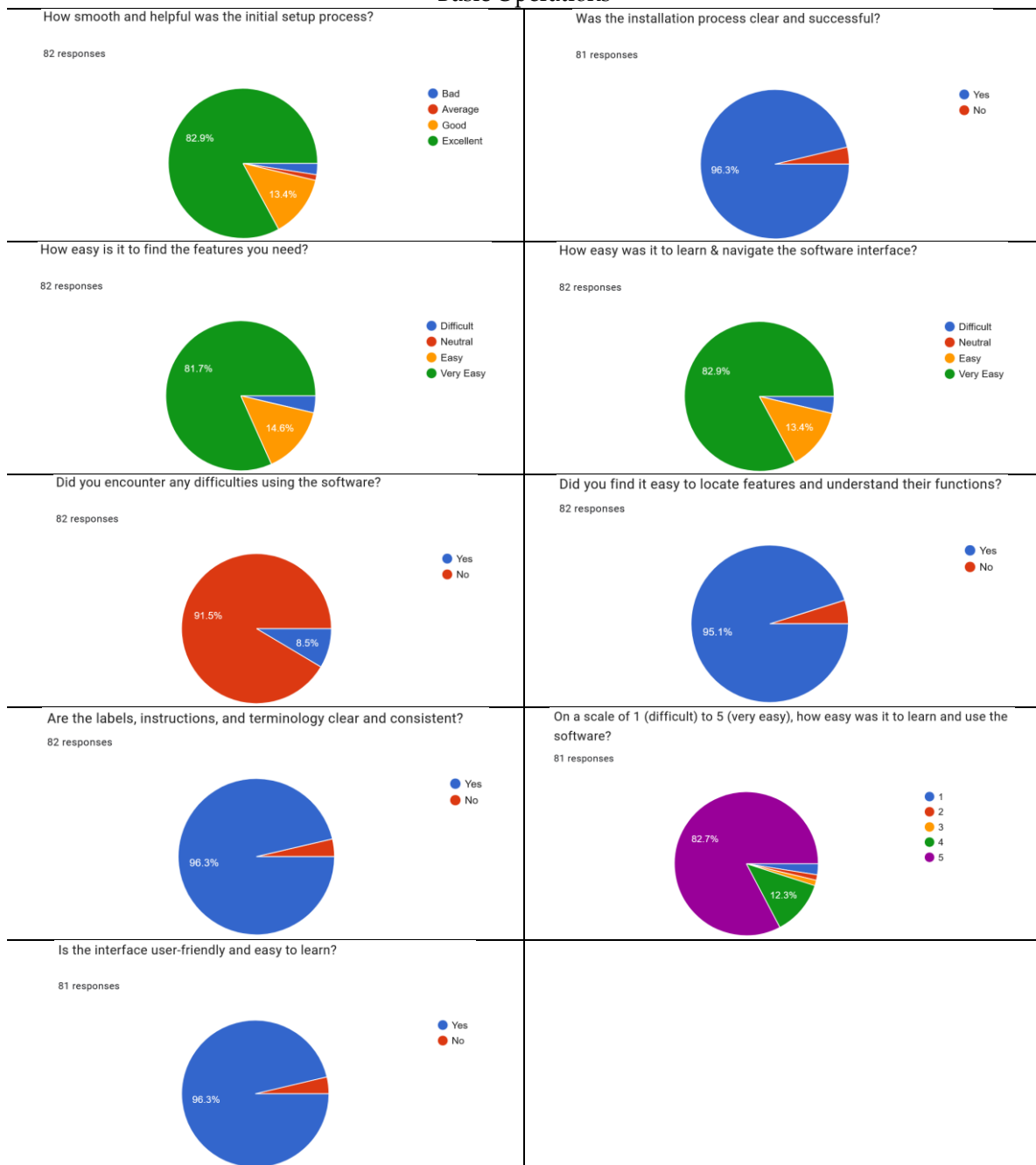


Figure A11. Evaluation of basic operations and the overall performance of the application.

Appendix C.2

Functionality and Features



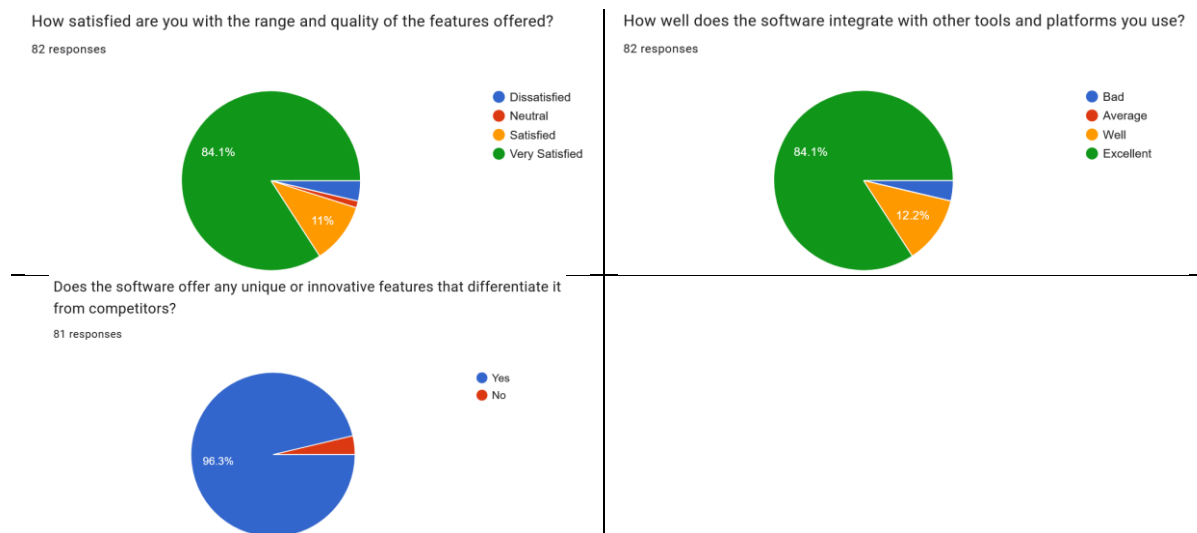


Figure A12. Evaluation of main functionality and features of the application.

Appendix C.3

Software—Users Satisfaction

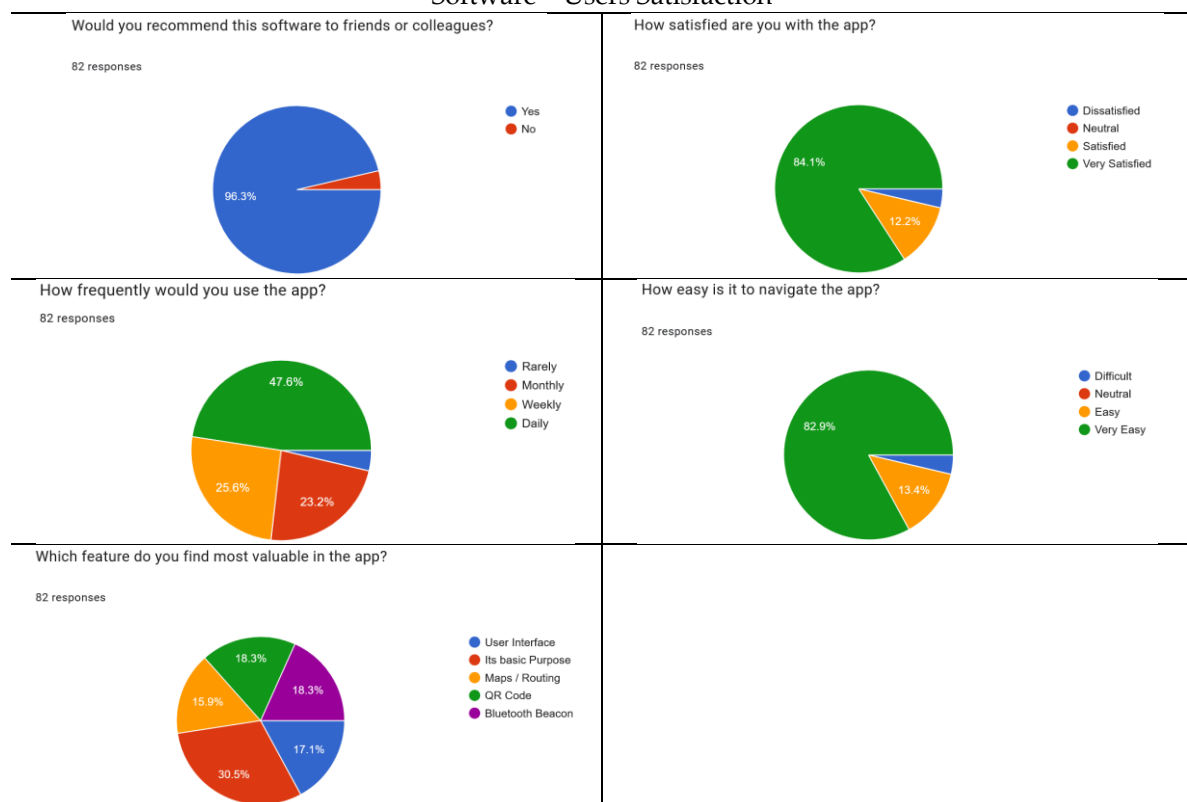


Figure A13. Evaluation of software compatibility and users' satisfaction level regarding the application.

Appendix C.4

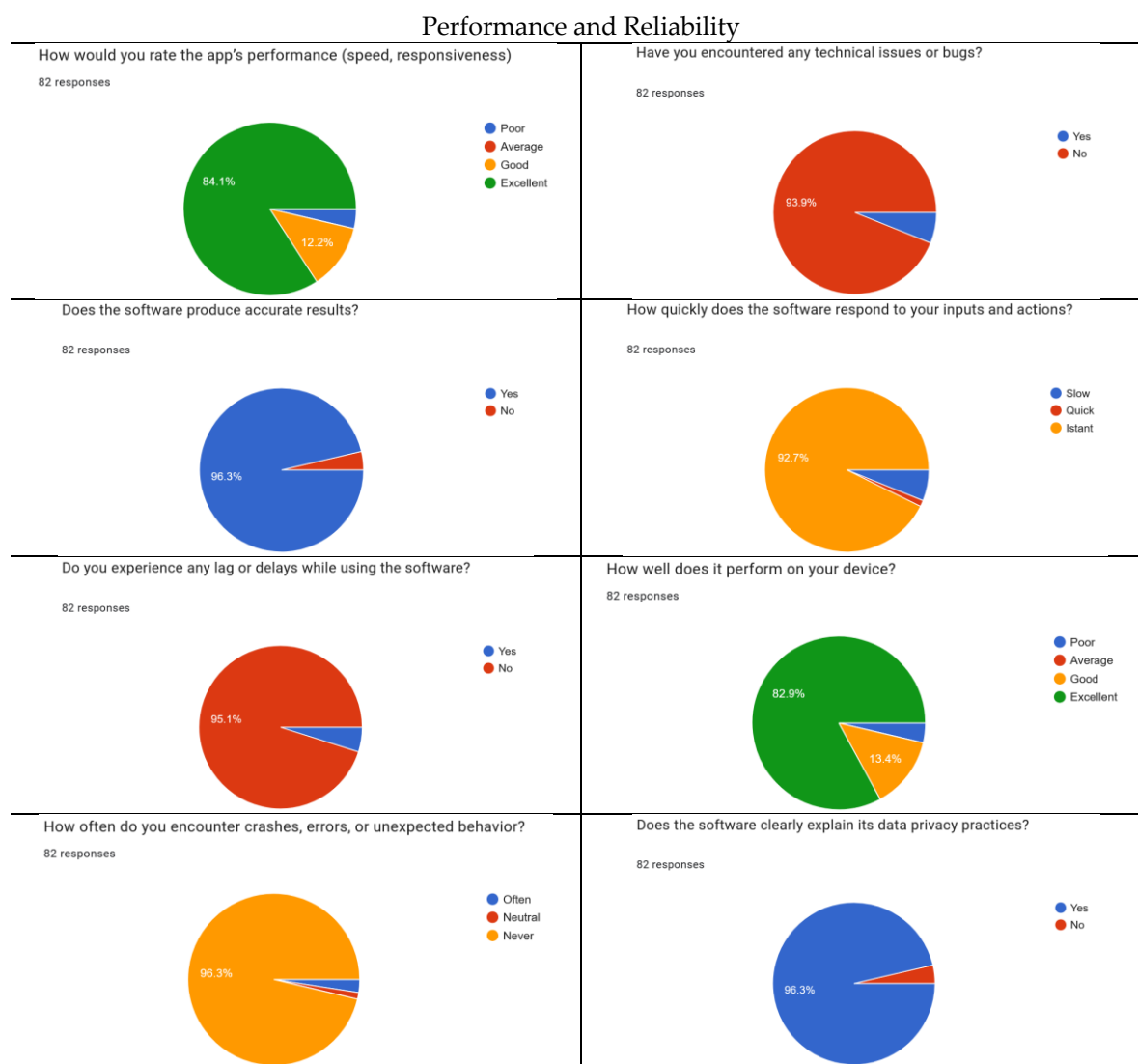


Figure A14. Evaluation of overall performance and reliability of the application.

References

- Boboc, R.G.; Băutu, E.; Gîrbacia, F.; Popovici, N.; Popovici, D.-M. Augmented Reality in Cultural Heritage: An Overview of the Last Decade of Applications. *Appl. Sci.* **2022**, *12*, 9859. <https://doi.org/10.3390/app12199859>.
- Zhang, K.; Wang, J.; Zhang, J.; Wang, Y.; Zeng, Y. Exploring the Impact of Location-Based Augmented Reality on Tourists' Spatial Behavior, Experience, and Intention Through a Field Experiment. *Tour. Manag.* **2024**, *102*, 104886. <https://doi.org/10.1016/j.tourman.2024.104886>.
- Ramtohl, A.; Khedo, K.K. Augmented reality systems in the cultural heritage domains: A systematic review. *Digit. Appl. Archaeol. Cult. Herit.* **2024**, *32*, e00317.
- Song, B.L.; Kaur, D.; Subramaniam, M.; Tee, P.K.; Wong, L.C.; Zin, N.A.M. The adoption of mobile augmented reality in tourism industry: Effects on customer engagement, intention to use and usage behaviour. *J. Tour. Serv.* **2024**, *15*, 235–252.
- Nugroho, I.S.; Priyanto, D.; Purnama, Y. Exploring the Role of Artificial Intelligence (AI) in Designing and Optimizing Personalization Strategies to Enhance Customer Experience in the Tourism Industry. *Technol. Soc. Perspect. (TACIT)* **2024**, *2*, 292–298.
- Setar, L.; MacFarland, M. Top 10 Fastest-Growing Industries. Special Report IbisWorld.com. 2012. Available online: <https://www.phillymag.com/wp-content/uploads/sites/3/2012/04/Fastest-Growing-Industries-2.pdf> (accessed on 22 July 2026)
- de Vasconcellos Pegas, F.; Grignon, J.; Morrison, C. Interdependencies among traditional resource use practices, sustainable tourism, and biodiversity conservation: A global assessment. *Hum. Dimens. Wildl.* **2015**, *20*, 454–469.

8. Zheng, D.; Huang, C.; Oraltay, B. Digital cultural tourism: Progress and a proposed framework for future research. *Asia Pac. J. Tour. Res.* **2023**, *28*, 234–253.
9. Ammirato, S.; Felicetti, A.M.; Linzalone, R.; Carlucci, D. Digital business models in cultural tourism. *Int. J. Entrep. Behav. Res.* **2022**, *28*, 1940–1961.
10. Sutherland, I.E. A head-mounted three dimensional display. In Proceedings of the December 9–11, 1968, Fall Joint Computer Conference, Part I, San Francisco, CA, USA, 9–11 December 1968; pp. 757–764. <https://doi.org/10.1145/1476589.1476686>.
11. Milgram, P.; Kishino, F. A taxonomy of mixed reality visual displays. *IEICE Trans. Inf. Syst.* **1994**, *77*, 1321–1329.
12. Azuma, R.T. A survey of augmented reality. *Presence Teleoperators Virtual Environ.* **1997**, *6*, 355–385. <https://doi.org/10.1162/pres.1997.6.4.355>.
13. Henderson, S.J.; Feiner, S.K. Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE Trans. Vis. Comput. Graph.* **2011**, *17*, 1355–1368. <https://doi.org/10.1109/TVCG.2010.245>.
14. Kounavis, C.D.; Kasimati, A.E.; Zamani, E.D. Enhancing the tourism experience through mobile augmented reality: Challenges and prospects. *Int. J. Eng. Bus. Manag.* **2012**, *4*, 10. <https://doi.org/10.5772/51644>.
15. Tom Dieck, M.C.; Jung, T. Value of augmented reality at cultural heritage sites: A stakeholder approach. *J. Destin. Mark. Manag.* **2017**, *6*, 110–117. <https://doi.org/10.1016/j.jdmm.2017.03.002>.
16. Chung, N.; Han, H.; Joun, Y. Tourists' intention to visit a destination: The role of augmented reality (AR) application for a heritage site. *Comput. Hum. Behav.* **2018**, *87*, 273–283. <https://doi.org/10.1016/j.chb.2018.06.017>.
17. Sylaiou, S.; Mania, K.; Karoulis, A.; White, M. Exploring the relationship between presence and enjoyment in a virtual museum. *Int. J. Human-Comput. Stud.* **2010**, *68*, 243–253. <https://doi.org/10.1016/j.ijhcs.2009.11.002>.
18. Billinghurst, M.; Clark, A.; Lee, G. A survey of augmented reality. *Found. Trends Human-Comput. Interact.* **2015**, *8*, 73–272. <https://doi.org/10.1561/11000000049>.
19. Jung, T.; Tom Dieck, M.C.; Moorhouse, N.; Dieck, D. Tourists' experience using mobile augmented reality: A case study of the Roman Baths. *Tour. Manag.* **2015**, *51*, 430–440. <https://doi.org/10.1016/j.tourman.2015.05.003>.
20. Linaza, M.T.; Marimon, D.; Carrasco, P.; Álvarez, R.; Montesa, J. Evaluation of mobile augmented reality applications for tourism destinations. In *Information and Communication Technologies in Tourism 2014*; Springer: Cham, Switzerland, 2014; pp. 431–445. https://doi.org/10.1007/978-3-319-03973-2_32.
21. Mortara, M.; Catalano, C.E.; Bellotti, F.; Fiucci, G.; Houry-Panchetti, M.; Petridis, P. Learning cultural heritage by serious games. *J. Cult. Herit.* **2014**, *15*, 318–325. <https://doi.org/10.1016/j.culher.2013.04.004>.
22. Wang, D.; Park, S.; Fesenmaier, D.R. The role of smartphones in mediating the touristic experience. *J. Travel Res.* **2012**, *51*, 371–387. <https://doi.org/10.1177/0047287511410320>.
23. Tussyadiah, I. The influence of innovative technologies on travel decisions. In *Information and Communication Technologies in Tourism 2015*; Springer: Cham, Switzerland, 2015; pp. 335–347. https://doi.org/10.1007/978-3-319-14343-9_24.
24. Damala, A.; Cubaud, P.; Bationo, A.; Houlier, P.; Marchal, I. Bridging the gap between the digital and the physical: Design and evaluation of a mobile augmented reality guide for the museum visit. *Personal. Ubiquitous Comput.* **2019**, *23*, 167–181.
25. Liarokapis, F.; Petridis, P.; Andrews, D.; de Freitas, S. Multimodal serious games technologies for cultural heritage. *Mediterr. Archaeol. Archaeom.* **2017**, *17*, 73–87.
26. Chatsiopolou, A.; Michailidis, P.D. Augmented Reality in Cultural Heritage: A Narrative Review of Design, Development and Evaluation Approaches. *Heritage* **2025**, *8*, 421. <https://doi.org/10.3390/heritage8100421>.
27. Omran, W.; Ramos, R.F.; Casais, B. Virtual reality and augmented reality applications and their effect on tourist engagement: A hybrid review. *J. Hosp. Tour. Technol.* **2024**, *15*, 497–518.
28. Wardiani, W.; Rusmana, A.; Damayani, N.A.; Khadija, U.L.S. The Role of Augmented Reality (AR) and Virtual Reality (VR) in Urban Heritage Tourism: A Study on Adoption and Communication Challenges. *J. Comput. Sci.* **2025**, *21*, 1908–1920.
29. Pinto, I.; Huertas, A. A comparative study of VR and AR heritage applications on visitor emotional experiences: A case study from a peripheral Spanish destination. *Virtual Real.* **2025**, *29*, 36.
30. Cetin, G.; Yarkan, S. The professional relationship between tour guides and tour operators. *Scand. J. Hosp. Tour.* **2017**, *17*, 345–357.
31. Black, R.; Ham, S. Improving the quality of tour guiding: Towards a model for tour guide certification. *J. Ecotourism* **2005**, *4*, 178–195.
32. Ap, J.; Wong, K.K. Case study on tour guiding: Professionalism, issues and problems. *Tour. Manag.* **2001**, *22*, 551–563.
33. Sofronov, B. The development of the travel and tourism industry in the world. *Ann. Spiru Haret. Univ. Econ. Ser.* **2018**, *18*, 123–137.

34. Camilleri, M.A. The tourism industry: An overview. In *Travel Marketing, Tourism Economics and the Airline Product: An Introduction to Theory and Practice*; Springer: Cham, Switzerland, 2017; pp. 3–27.
35. Leiper, N. The framework of tourism: Towards a definition of tourism, tourist, and the tourist industry. *Ann. Tour. Res.* **1979**, *6*, 390–407.
36. Kapa, M.G.; de Crom, E.P.; Hermann, U.P. An exploration of tourist guides' competencies to create memorable tourist experiences. *Geoj. Tour. Geosites* **2022**, *44*, 1350–1358.
37. McDonnell, I. *The Role of the Tour Guide in Transferring Cultural Understanding*; School of Leisure, Sport and Tourism, University of Technology Sydney: Sydney, Australia, 2001.
38. Feldman, J.; Skinner, J. Tour guides as cultural mediators. *Ethnol. Eur.* **2018**, *48*, 5–13.
39. Boyle, A.; Arnott, A. What tour guide stories can tell us about learning, education and training. In *CAUTHE 2004: Creating Tourism Knowledge*; Common Ground Publishing: Brisbane, Australia, 2004; Volume 75.
40. Chitima, S.S. Tour guides as facilitators of learning under field trips. *J. Teach. Travel Tour.* **2025**, *25*, 41–59.
41. Tsaur, S.H.; Teng, H.Y. Exploring tour guiding styles: The perspective of tour leader roles. *Tour. Manag.* **2017**, *59*, 438–448.
42. Mujio, M.; Rosandi, V.B.; Waskitaningsih, N.; Zahirah, K.R.; Chaniago, I.Y. Economic advancement of rural communities through tourism guide services as a paradigm for sustainable business transformation in the tropics. In *BIO Web of Conferences*; EDP Sciences: Les Ulis, France, 2025; Volume 186, p. 02001.
43. Hu, W. *Tour Guides and Sustainable Development: The Case of Hainan, China*; University of Waterloo: Waterloo, ON, Canada, 2007.
44. Düz, B. Professional Ethics in Tour Guiding: Issues, Challenges, and Policies. In *Modern Social Challenges Impacting Tourism: Ethics, Societal Disruption, and Intimacy*; IGI Global Scientific Publishing: Hershey, PA, USA, 2025; pp. 55–80.
45. Nyahunzvi, D.K.; Njerekai, C. Tour guiding in Zimbabwe: Key issues and challenges. *Tour. Manag. Perspect.* **2013**, *6*, 3–7.
46. Chowdhary, N.; Prakash, M. Challenges of tourist guiding—an assessment of situation in India. In *Proceedings of the National Conference on Tourism in India—Challenges Ahead*, Kozhikode, India, 15–17 May 2008.
47. Prakash, M.; Chowdhary, N. Tour guides: Roles, challenges and desired competences a review of literature. *Int. J. Hosp. Tour. Syst.* **2010**, *3*.
48. Hughes, K.; Ballantyne, R. Interpretation in ecotourism settings: Investigating tour guides' perceptions of their role, responsibilities and training needs. *J. Tour. Stud.* **2001**, *12*, 2–9.
49. Abd El Kafy, J. Challenges facing tour guide profession and their impacts on the Egyptian guides performance. *J. Assoc. Arab Univ. Tour. Hosp.* **2020**, *19*, 113–130.
50. Zejda, D.; Pásková, M. Destination Management Support System and Intelligent Destination Guide for Natural Destinations. In *Proceedings of the 14th Conference Public Recreation and Landscape Protection—With Environment Hand in Hand?*; Mendel University in Brno, Brno, Czech Republic, 2023; pp. 130–134.
51. Sengoz, A.; Dogru, T.; Mody, M.; Isik, C. Guiding the path to sustainable tourism development: Investigating the role of tour guides within a social exchange theory paradigm. *Tour. Manag.* **2025**, *110*, 105162.
52. Yamada, N. Why tour guiding is important for ecotourism: Enhancing guiding quality with the ecotourism promotion policy in Japan. *Asia Pac. J. Tour. Res.* **2011**, *16*, 139–152.
53. Örnek, N.A.; Karamustafa, K. Scenarios for the future of tour guiding: A Delphi-based analysis for 2040. *Futures* **2025**, *165*, 103497.
54. Abe, Y.; Sato, S.; Kato, K.; Hyakumachi, T.; Yanagibashi, Y.; Ito, M.; Abumi, K. A novel 3D guidance system using augmented reality for percutaneous vertebroplasty. *J. Neurosurg. Spine* **2013**, *19*, 492–501.
55. Katz, B.F.; Dramas, F.; Parsehian, G.; Gutierrez, O.; Kammoun, S.; Brilhault, A.; Brunet, L.; Gallay, M.; Oriola, B.; Auvray, M.; et al. NAVIG: Guidance system for the visually impaired using virtual augmented reality. *Technol. Disabil.* **2012**, *24*, 163–178.
56. Vertucci, R.; D'Onofrio, S.; Ricciardi, S.; De Nino, M. History of augmented reality. In *Springer Handbook of Augmented Reality*; Springer International Publishing: Cham, Switzerland, 2023; pp. 35–50.
57. Wei, X.; Weng, D.; Liu, Y.; Wang, Y. A tour guiding system of historical relics based on augmented reality. In *Proceedings of the 2016 IEEE Virtual Reality (VR), Greenville, SC, USA, 19–23 March 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 307–308.
58. Laviola, E. Optimization of augmented reality interfaces for technical documentation. Ph.D. Thesis, Politecnico di Bari, Bari, Italy, October 2023.
59. Saha, N. Augmented Reality (AR) and Virtual Reality (VR)-Based Data Visualization Frameworks for the Manufacturing Industry. Master's Thesis, University of South Carolina, Columbia, SC, USA, 2024.

60. Jasche, F.; Hoffmann, S.; Ludwig, T.; Wulf, V. Comparison of different types of augmented reality visualizations for instructions. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, Yokohama, Japan, 8–13 May 2021; pp. 1–13.
61. Gerstweiler, G. Guiding people in complex indoor environments using augmented reality. In *Proceedings of the 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, Tuebingen/Reutlingen, Germany, 18–22 March 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 801–802.
62. Aukstakalnis, S. *Practical Augmented Reality: A Guide to the Technologies, Applications, and Human Factors for AR and VR*; Addison-Wesley Professional: Boston, MA, USA, 2016.
63. Zubizarreta, J.; Aguinaga, I.; Amundarain, A. A framework for augmented reality guidance in industry. *Int. J. Adv. Manuf. Technol.* **2019**, *102*, 4095–4108.
64. Steele, P.; Burleigh, C.; Kroposki, M.; Magabo, M.; Bailey, L. Ethical considerations in designing virtual and augmented reality products—Virtual and augmented reality design with students in mind: Designers’ perceptions. *J. Educ. Technol. Syst.* **2020**, *49*, 219–238.
65. Blokša, J. Design Guidelines for User Interface for Augmented Reality. Master’s Thesis, Masaryk University, Brno, Czech Republic, 2017.
66. Diegmann, P.; Schmidt-Kraepelin, M.; Eynden, S.; Basten, D. Benefits of augmented reality in educational environments—a systematic literature review. In *Proceedings of the 12th International Conference on Wirtschaftsinformatik*, Osnabrück, Germany, 4–6 March 2015.
67. Reinhart, G.; Patron, C. Integrating augmented reality in the assembly domain—fundamentals, benefits and applications. *CIRP Ann.* **2003**, *52*, 5–8.
68. Lee, C.I. Benefit analysis of gamified augmented reality navigation system. *Appl. Sci.* **2022**, *12*, 2969.
69. Rose, E.; Breen, D.; Ahlers, K.H.; Crampton, C.; Tuceryan, M.; Whitaker, R.; Greer, D. Annotating real-world objects using augmented reality. In *Computer Graphics*; Academic Press: Cambridge, MA, USA, 1995; pp. 357–370.
70. Nuernberger, B.; Ofek, E.; Benko, H.; Wilson, A.D. Snapto reality: Aligning augmented reality to the real world. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, San Jose, CA, USA, 7–12 May 2016; pp. 1233–1244.
71. Haris, M.; Jadoon, B.; Yousaf, M.; Khan, F.H. Evolution of android operating system: A review. *Asia Pac. J. Contemp. Educ. Commun. Technol.* **2018**, *4*, 178–188.
72. Nimodia, C.; Deshmukh, H.R. Android operating system. *Softw. Eng.* **2012**, *3*, 10.
73. Chinetha, K.D.; Joann, J.D.; Shalini, A. An evolution of android operating system and its version. *Int. J. Eng. Appl. Sci.* **2015**, *2*, 257997.
74. Verma, A.; Arora, S.; Verma, P. Android OS, its security and features. *Int. J. Recent Res. Asp.* **2017**, *4*, 241–251.
75. Lazareska, L.; Jakimoski, K. Analysis of the advantages and disadvantages of Android and iOS systems and converting applications from Android to iOS platform and vice versa. *Am. J. Softw. Eng. Appl.* **2017**, *6*, 116–120.
76. Radice, R.A.; Roth, N.K.; O’Hara, A.C.; Ciarfella, W.A. A programming process architecture. *IBM Syst. J.* **1985**, *24*, 79–90.
77. Baumeister, H.; Hacklinger, F.; Hennicker, R.; Knapp, A.; Wirsing, M. A component model for architectural programming. *Electron. Notes Theor. Comput. Sci.* **2006**, *160*, 75–96.
78. Wang, C.; Duan, W.; Ma, J.; Wang, C. The research of Android System architecture and application programming. In *Proceedings of 2011 International Conference on Computer Science and Network Technology*, Harbin, China, 24–26 December 2011; IEEE: Piscataway, NJ, USA, 2011; Volume 2, pp. 785–790.
79. Gutierrez, A.; Dreslinski, R.G.; Wenisch, T.F.; Mudge, T.; Saidi, A.; Emmons, C.; Paver, N. Full-system analysis and characterization of interactive smartphone applications. In *Proceedings of the 2011 IEEE International Symposium on Workload Characterization (IISWC)*, Austin, TX, USA, 6–8 November 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 81–90.
80. Li, C.; Neugroschl, J.; Zhu, C.W.; Aloysi, A.; Schimming, C.A.; Cai, D.; Grossman, H.; Martin, J.; Sewell, M.; Loizos, M.; et al. Design considerations for mobile health applications targeting older adults. *J. Alzheimer’s Dis.* **2021**, *79*, 1–8.
81. Dvorski, D.D. Installing, configuring, and developing with Xampp. *Skills Canada* **2007**, 492. Available online: <https://tesidottorato.depositolegale.it/bitstream/20.500.14242/65233/1/36%20ciclo-LAVIOLA%20Enricoandrea.pdf> (accessed on 22 July 2026)
82. Laurie, B.; Laurie, P. *Apache: The Definitive Guide*; O’Reilly Media, Inc.: Sebastopol, CA, USA, 2003.
83. Grippa, V.M.; Kuzmichev, S. *Learning MySQL*; O’Reilly Media, Inc.: Sebastopol, CA, USA, 2021.
84. Lerdorf, R.; Tatroe, K. *Programming Php*; O’Reilly Media, Inc.: Sebastopol, CA, USA, 2002.

85. Wibowo, M.C.; Putra, T.W.A. Utilizing phpMyAdmin for System Design in Enterprise Administration. *J. Technol. Inform. Eng.* **2024**, *3*, 217–234.
86. Ehsan, A.; Abuhaliqa, M.A.M.; Catal, C.; Mishra, D. RESTful API testing methodologies: Rationale, challenges, and solution directions. *Appl. Sci.* **2022**, *12*, 4369.
87. Moroney, L. The firebase realtime database. In *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform*; Apress: Berkeley, CA, USA, 2017; pp. 51–71.
88. Tsamis, G.; Kavvadias, S.; Papagrigoriou, A.; Grammatikakis, M.D.; Papadimitriou, K. Efficient bandwidth regulation at memory controller for mixed criticality applications. In *Proceedings of the 2016 11th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Tallinn, Estonia, 27–29 June 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 1–8.
89. Li, L.; Bissyandé, T.F.; Klein, J.; Le Traon, Y. An investigation into the use of common libraries in android apps. In *Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Osaka, Japan, 14–18 March 2016*; IEEE: Piscataway, NJ, USA, 2016; Volume 1, pp. 403–414.
90. Zhan, X.; Liu, T.; Fan, L.; Li, L.; Chen, S.; Luo, X.; Liu, Y. Research on third-party libraries in android apps: A taxonomy and systematic literature review. *IEEE Trans. Softw. Eng.* **2021**, *48*, 4181–4213.
91. Tsamis, G.; Vassiliou, G.; Chatzinikolaou, S.; Kondylakis, H.; Papadakis, N. A Multi-Constrained Knapsack Approach for Educational Resource Allocation: Genetic Algorithm with Category-Specific Optimization. *Electronics* **2025**, *14*, 3898.
92. Vassiliou, G.; Tsamis, G.; Chatzinikolaou, S.; Nipurakis, T.; Papadakis, N. Enhancing Discoverability: A Metadata Framework for Empirical Research in Theses. *Algorithms* **2025**, *18*, 490.
93. Tsamis, G.; Grammatikakis, M.D.; Papagrigoriou, A.; Petrakis, P.; Piperaki, V.; Mouzakitis, A.; Coppola, M. Soft real-time smartphone ECG processing. In *Proceedings of the 2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES), Toulouse, France, 14–16 June 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 1–4.
94. Rumpe, B. *Modeling with UML*; Springer: Cham, Switzerland, 2016; Volume 98.
95. Dobing, B.; Parsons, J. How UML is used. *Commun. ACM* **2006**, *49*, 109–113.
96. Eriksson, H.E.; Penker, M. *Business Modeling with UML*; John Wiley & Sons, Inc.: New York, NY, USA, 2000; Volume 12, p. 444.
97. Berardi, D.; Calvanese, D.; De Giacomo, G. Reasoning on UML class diagrams. *Artif. Intell.* **2005**, *168*, 70–118.
98. Li, X.; Liu, Z.; Jifeng, H. A formal semantics of UML sequence diagram. In *Proceedings of the 2004 Australian Software Engineering Conference. Proceedings, Melbourne, Australia, 13–16 April 2004*; IEEE: Piscataway, NJ, USA, 2004; pp. 168–177.
99. Aditya, S.K.; Karn, V.K. *Android SQLite Essentials*; Packt Publishing: Birmingham, UK, 2014; pp. 88–90.
100. Vogel, L. Android sqlite database and contentprovider-tutorial. *Java Eclipse Android Web Program. Tutor.* **2010**, *8*, 5.
101. Ferreira, C.; Lopes, M.; Correia, L.; Wanzeller, C.; Sá, F.; Martins, P.; Abbasi, M. Database Performance on Android Devices, A Comparative Analysis. In *Marketing and Smart Technologies: Proceedings of ICMarTech 2022*; Springer: Singapore, 2023; Volume 2, pp. 261–273.
102. Mooney, P.; Minghini, M. A review of OpenStreetMap data. In *Mapping and the Citizen Sensor*; Ubiquity Press: London, UK, 2017; pp. 37–59.
103. Haklay, M.; Weber, P. Openstreetmap: User-generated street maps. *IEEE Pervasive Comput.* **2008**, *7*, 12–18.
104. Soon, T.J. QR code. *Synth. J.* **2008**, *2008*, 59–78.
105. Tiwari, S. An introduction to QR code technology. In *Proceedings of the 2016 International Conference on Information Technology (ICIT), Bhubaneswar, India, 22–24 December 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 39–44.
106. Bhagavatula, R.; Ur, B.; Iacovino, K.; Kywe, S.M.; Cranor, L.F.; Savvides, M. Biometric authentication on iphone and android: Usability, perceptions, and influences on adoption. In *Proceedings of the USEC '15: Workshop on Usable Security, San Diego, CA, USA, 8 February 2015*.
107. Zhang, X.; Cheng, D.; Jia, P.; Dai, Y.; Xu, X. An efficient android-based multimodal biometric authentication system with face and voice. *IEEE Access* **2020**, *8*, 102757–102772.
108. Ahlawat, S.; Dahiya, R. A Novel Approach of Text to Speech Conversion Under Android Environment. *Int. J. Comput. Sci. Manag. Stud.* **2013**, *13*, 253.
109. Sonawane, A.R.; Wankhede, A.; Rasane, K.; Baraskaev, V.; Borde, G. Android Application for Visually Impaired People using Text-To-Speech. *Int. J. Adv. Res. Comput. Commun. Eng.* **2016**, *5*, 12.
110. Song, W.; Lee, H.; Lee, S.H.; Choi, M.H.; Hong, M. Implementation of android application for indoor positioning system with estimote BLE beacons. *J. Internet Technol.* **2018**, *19*, 871–878.
111. Kriz, P.; Maly, F.; Kozel, T. Improving indoor localization using bluetooth low energy beacons. *Mob. Inf. Syst.* **2016**, *2016*, 2083094.

112. Zuo, C.; Wen, H.; Lin, Z.; Zhang, Y. Automatic fingerprinting of vulnerable ble iot devices with static uuids from mobile apps. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 1469–1483.
113. Chougale, P.; Yadav, V.; Gaikwad, A.; Vidyapeeth, B. Firebase-overview and usage. *Int. Res. J. Mod. Eng. Technol. Sci.* **2021**, *3*, 1178–1183.
114. Khawas, C.; Shah, P. Application of firebase in android app development-a study. *Int. J. Comput. Appl.* **2018**, *179*, 49–53.
115. Chatterjee, N.; Chakraborty, S.; Decosta, A.; Nath, A. Real-time communication application based on android using Google firebase. *Int. J. Adv. Res. Comput. Sci. Manag. Stud.* **2018**, *6*, 74–79.
116. Tsamis, G.; Evangelos, G.; Papakostas, A.; Vassiliou, G.; Grafanakis, M.; Garefalakis, A.; Vassalos, M.; Mylona, A.; Papadakis, N. Cost-Effective Design, Content Management System Implementation and Artificial Intelligence Support of Greek Government AADE, myDATA Web Service for Generic Government Infrastructure, a Complete Analysis. *Algorithms* **2025**, *18*, 339.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.