**Egyptian E-Learning University**

Faculty of Computers & Information Technology

**Tour Scan**

**By**

|  |  |
| --- | --- |
| Mohamed Mamdouh Abdelbary | 21-01182 |
| Mostafa Kamel Hussien | 21-01312 |
| Saber Sayed Ahmed | 21-01525 |
| Mai Ismail Omer Ali | 21-00121 |
| Esraa Hossien Abdelhafez | 21-00647 |
| Kholoud Abdelmaugoud Sabry | 21-00881 |
| Dina Sadiq Badr | 21-00601 |

**Under Supervision of:**

**DR. Yasser Abdelhameed**

(Lecturer of Computers and Information Technology Egyptian

E-Learning University)

**Eng. Salma Montaser**

(Teacher assistant at Computers and Information Technology Egyptian E-Learning University)

This documentation is submitted as a partial fulfillment of the requirements for the degree of Bachelor of Science in Computer & Information Technology

**Assiut 2025**

**Acknowledgment**

In the name of Allah, the Most Gracious and the Most Merciful Alhamdulillah, all  
praise Allah for the strengths and His blessing in completing this Project.  
In performing our project, we had to take the help and guideline of some respected  
persons, who deserve our greatest gratitude. The completion of this project gives us  
much Pleasure. We would like to present our supervisors:

**Dr. Yasser Abdelhameed**

Faculty of Computers and Information Technology Program Director – EELU

**Eng. Salma Montaser**Faculty of Computers and Information Technology – EELU  
For giving us a good guideline for the Project throughout numerous consultations

**For giving us a good guideline for the Project throughout numerous consultations.**

**We would also like to expand our deepest gratitude to all those who have directly**

**and indirectly guided us in doing this Project.**

**In addition, a thank you to thank Eng. Salma Montaser who introduced us to**

**the methodology of work, and whose passion for the “underlying structures” had**

**lasting effect. We also thank our university EELU for consent to include**

**copyrighted pictures as a part of our graduation project.**

**Many people, especially our classmates and team members itself, have made**

**valuable comment suggestions on this proposal which gave us an inspiration to**

**improve our project. We thank all the people for their help directly and indirectly**

**to complete our project**

**Abstract**

This project presents **TourScan**, a cross-platform Flutter mobile application designed to enhance museum visits and cultural tourism by utilizing Artificial Intelligence (AI) to recognize statues and provide users with detailed, real-time information. By simply pointing their smartphone camera at a statue, users can instantly receive its name, historical background, artist information, era, and other relevant facts.

The report outlines the major components of the project, starting with the motivation and objectives, followed by a review of related work in AI-based cultural heritage applications. It then details the methodology used to build the application, including data collection, image recognition using computer vision techniques, natural language generation for presenting the information, and user interface design. Testing and evaluation methods are also discussed to assess the system’s accuracy and usability. Finally, the report concludes with the challenges faced, key findings, and potential future improvements, such as expanding the database to cover a wider range of artifacts and adding multilingual support. This project contributes to smart tourism by blending modern technology with cultural appreciation.

**Table of Contents**

**.Acnowledge**……………………………………………………………………………………………………………………………………………………….......i

* **Abstract** ........................................................................................ii
* **List of Figures** .............................................................................iii

**Chapter 1: Introduction**

1.1 Introduction .......................................................................................12  
1.2 Background and Motivation for the Project ...........................................13  
1.3 Importance of the Problem Being Addressed ........................................14  
1.4 Problem Statement .............................................................................15  
1.5 Objectives ..........................................................................................16  
1.6 Brief Overview of the Proposed Solution ...............................................18

**Chapter 2: Literature Review / Related Work**

2.1 Deep Learning and IoT for Tourist Recommendation in Smart Cities .................................................................................................................20  
2.2 Sentiment-Based Tourism Recommendation System .................................................................................................................21  
2.3 TOURGURU: Guide Mobile Application for Tourists .................................................................................................................22  
2.4 Smart Tourism using IoT, AI, and 5G ..................................................... ……………………………………………………………………………………………………………………………………………………………………..…….23  
2.5 Context-Aware Real-Time Recommendation Systems .................................................................................................................24

**Chapter 3: Proposed System**

3.1 Problem-Solving Approach ...............................................................27  
3.2 System Architecture (UML, Flowcharts, ER Diagram) .............................................................................................................28  
3.3 Algorithms and Frameworks Used .............................................................................................................34

**Chapter 4: Implementation**

4.1 Technologies, Tools, and Programming Languages ...............................…………………………………………………………………………………………………………………………………………………..36  
4.2 Key Components / Modules…………………………………………….38  
 4.2.1 Frontend Components ........................................................... ..38  
 4.2.2 Backend Components ............................................................. 63  
 4.2.3 AI Implementation .................................................................... 72

4.3 Machine Learning Model Implementation

…………………………………………………………………………………..78

**Chapter 5: Testing & Evaluation**

5.1 Testing Strategies  
 5.1.1 Unit Testing ................................................................................85  
 5.1.2 Integration Testing ......................................................................86  
 5.1.3 User Testing ...............................................................................88  
 5.1.4 Model Evaluation and Selection .................................................90  
5.2 Performance Metrics  
 5.2.1 Accuracy .................................................................................... 94  
 5.2.2 Speed ........................................................................................ 95  
 5.2.3 Scalability .................................................................................. 97  
5.3 Comparison with Existing Solutions ..................................................98

**Chapter 6: Results & Discussion**

6.1 Introduction .......................................................................................100  
6.2 Summary of Findings .........................................................................100  
6.3 Interpretation of Results .....................................................................102

**Chapter 7: Conclusion**

7.1 Summary of Contributions ..................................................................................................................104

**Chapter 8: Future Work**

8.1 Future Enhancements and Features .................................................................................................................105

* **References** ....................................................................................108

**List of Figures:**

**System Architecture Diagrams**

Figure 1: UML Class Diagram ........................................................... 28

Figure 2: Component Diagram .......................................................... 29

Figure 3: Sequence Diagram ............................................................ 29

Figure 4: Use Case Diagram ............................................................ 30

Figure 5: Activity Diagram ................................................................ 31

Figure 6: Flowchart .......................................................................... 32

Figure 7: ER Diagram ...................................................................... 33

**User Interface (UI) Screens**

Figure 8: Splash View ...................................................................... 38

Figure 9: Home View ........................................................................ 39

Figure 10: Additional Home UI ......................................................... .39

Figure 11:................................................................................ .40

Figure 12.................................................................................. 40

Figure 13:................................................................................. 41

Figure 14:................................................................................. 42

Figure 15:................................................................................. 42

Figure 16:...................................................................................43

Figure 17:.................................................................................. 43

Figure 18................................................................................... 44

Figure 19:.................................................................................. 45

Figure 20: .................................................................................. 46

Figure 21:................................................................................... 47

Figure 22.................................................................................... 48

Figure 23:.................................................................................... 49

Figure 24..................................................................................... 49

Figure 25:................................................... ………………………..50

Figure 26:............................................. ……………………………..51

Figure 27....................................................................................... 52

Figure 28..................................................................................... ...53

Figure 29:....................................................................................... 54

Figure 30:....................................................................................... 55

Figure 31:....................................................................................... 56

Figure 32:........................................................................................ 57

Figure 33:........................................................................................ 58

Figure 34:………………………........................................................ 59

Figure 35:…………………………...................................................... 59

Figure 36:…………………………….................................................... 60

Figure 37:………………………............................................................ 60

Figure 38:……………………….............................................................. 61

Figure 39:…………………………........................................................... 61

Figure 40:……………………………….................................................... 62

Figure 41:…………………………............................................................ 62

Figure 42:…………………………………................................................. 62

Figure 43:………………………………………........................................ ..63

Figure 44:……………………………….................................................. 64

Figure 45:……………………………................................................... 65

Figure 46……………………………..................................................... 66

Figure 47:…………………………………............................................. 67

Figure 48…………………………….................................................. 68

Figure 49:…………………………….................................................. 68

Figure 50:…………………………...................................................... 69

Figure 51…………………………....................................................... 70

Figure 52:……………………………................................................... 71

Figure 53:…………………….............................................................. 73

Figure 54………………....................................................................... 74

Figure 55:………………………............................................................. 78

Figure 56................................................. …………………………….79

Figure 57:……………………………................................................. 80

Figure 58:…………………………..................................................... 81

Figure 59:…………………………...................................................... 83

Figure 60:……………………………................................................... 90

Figure 61……………………............................................................... 91

Figure 62:………………...................................................................... 92

Figure 63………………………............................................................. 93

Figure 64................................................. ……………………………..95

Figure 65:……………………………................................................. 96

Figure 66:…………………………....................................................... 97

Chapter 1

**Introduction**

**1.1 ​Introduction:**

In recent years, the integration of Artificial Intelligence (AI) into the tourism industry has opened new possibilities for enhancing the travel experience. This project introduces TourScan, a smart, cross-platform Flutter mobile application designed to enrich cultural tourism and museum visits. The application allows users to scan statues using their smartphone camera and instantly receive comprehensive information about the scanned object, including its name, historical background, artist information, era, and other relevant facts. By combining real-time image recognition technology with a user-friendly interface, TourScan bridges the gap between tourists and historical knowledge. It offers an interactive and educational experience, particularly in historical sites and museums. Moreover, the app supports both English and Arabic languages, providing a complete digital museum experience accessible to a wider audience. For more detailed insights into the system architecture and specific subsystems, please refer to the respective documentation pages.

The core of the system relies on a Convolutional Neural Network (CNN) model, which has been trained to accurately identify various statues based on visual input. Once an image is captured, the model processes it and retrieves detailed information such as the statue's name, origin, historical significance, artist, and time period. This solution offers an interactive and educational experience, making cultural exploration more accessible and engaging for users of all ages.

This report outlines the motivation behind the project, the development process, and the technologies involved in building the application, including the CNN model and mobile development tools

**1.2 Background and motivation for the project.**

Tourists visiting historical sites often lack immediate access to accurate and engaging information about the statues and monuments they encounter. Traditional methods such as guidebooks or tour guides may be limited, outdated, or unavailable. This creates a gap between the visitor and the cultural value of the site. Motivated by the need to enhance tourist engagement through technology, this project aims to develop a mobile application that uses AI and image recognition to provide instant, detailed information about historical statues. The goal is to make cultural exploration more interactive, educational, and accessible.

**Information Accessibility:** Traditional museum visits often lack immediate, detailed information about exhibits. TourScan solves this by providing instant recognition and information retrieval through smartphone cameras.

**Language Barriers:** Museums serving diverse populations need multilingual support. The app supports both English and Arabic, making cultural heritage accessible to English -speaking visitors.

**Interactive Engagement:** Static museum displays can be passive experiences. TourScan adds interactivity through features like text-to-speech and AI chatbot assistance.

**1.3 Importance of the problem being addressed.**

The problem TourScan addresses is fundamentally about democratizing access to cultural heritage. In an era where cultural preservation and education are increasingly important, the app tackles the practical barriers that prevent people from fully engaging with museum experiences - language, accessibility, and information availability. The extensive multilingual content and sophisticated recognition technology represent a significant investment in making Egyptian cultural heritage accessible to both local and international audiences.

**Rich Historical Content:** The importance becomes clear when examining the extensive cultural content, the app preserves and makes accessible. contains detailed information about significant Egyptian artifacts spanning thousands of years. This includes everything from ancient pharaonic artifacts to Islamic manuscripts, representing Egypt's diverse cultural heritage.

**Democratizing Museum Technology:** The scanning functionality makes advanced museum technology accessible through smartphones, eliminating the need for expensive audio guides or specialized equipment that many visitors cannot afford.

**Cross-Platform Reach:** By supporting multiple platforms through Flutter, the app ensures that cultural heritage information is accessible regardless of device type or economic status.

**Interactive Learning:** Rather than passive museum experiences, TourScan transforms visits into interactive educational opportunities where users can immediately access detailed historical and cultural context about what they're viewing.

**1.4 Problem Statement .**

Tourism plays a significant role in cultural enrichment and education. However, tourists visiting historical and archaeological sites often face challenges in accessing accurate and engaging information about the statues and monuments they encounter. Traditional resources, such as guidebooks, brochures, and physical tour guides, have limitations in terms of availability, consistency, and real-time relevance.Moreover, in many cases, tourists may lack the background knowledge or language proficiency to fully understand the historical and cultural significance of the statues. This leads to a reduced connection with the site and a less immersive experience. In crowded or remote areas, it can be particularly difficult to find reliable information on-demand, further hindering the learning experienceThis situation highlights the need for a more effective and accessible solution to bridge the gap between tourists and the rich cultural heritage of historical monuments. By providing real-time, accurate, and interactive information, tourists could have a more informative and enriching experience while exploring cultural sites.Therefore, this project seeks to address this gap by developing a mobile application that leverages AI and image recognition technologies, specifically using a Convolutional Neural Network (CNN) model, to scan and identify statues. The app will then provide detailed and relevant information, making cultural exploration more interactive, educational, and accessible for tourists

**1.5 Objectives.**

The primary objective of this project is to enhance the tourist experience through innovative technology. By integrating Artificial Intelligence (AI) and image recognition, the mobile application aims to transform traditional sightseeing into an interactive, informative, and immersive experience.Key objectives include:Enhancing Tourist Experience Through Technology: Leverage AI and computer vision to create a smart tourism guide capable of providing instant insights about statues and landmarks. This elevates cultural exploration by combining education and convenience.Implementing an Image Recognition System: Deploy a Convolutional Neural Network (CNN) to ensure accurate and real-time recognition of statues and monuments using images captured by the user’s smartphone camera.

Providing Instant Information: Automatically deliver detailed and reliable data—such as the name, origin, artist, and historical background—about any identified statue, directly after scanning.

Ensuring Easy User Interaction: Design a clean and intuitive User Interface (UI) that makes the scanning process simple and accessible to all users, regardless of their technical proficiency.

Bridging Language Barriers in Cultural Heritage: Offer full bilingual support in both English and Arabic, ensuring accurate localization and translation so that the application caters to a broad and diverse audienc

In addition to enhancing the tourist experience through AI and image recognition, the project aims to democratize access to museum technology by making advanced tools—once limited to physical exhibits—available directly on smartphones. This approach eliminates the need for expensive audio guides and provides a rich, self-guided digital experience accessible to all users. To promote accessibility and inclusion, the application will feature text-to-speech functionality with automatic language detection, making it suitable for individuals with visual impairments or reading difficulties.

Furthermore, the project seeks to preserve and share Egyptian cultural heritage by building a detailed, bilingual digital database of artifacts and historical information, ensuring accurate and engaging content for diverse audiences. Finally, the application will include an AI-powered chatbot and real-time communication tools, offering an interactive digital museum assistant that goes beyond object recognition to support user inquiries, provide guidance, and enrich the cultural learning journey.

**1.6 Brief overview of the proposed solution.**

The proposed solution, TourScan, is a mobile-first, AI-driven application designed to deliver an intelligent and accessible cultural heritage experience. Built using Flutter for cross-platform compatibility, the app ensures a consistent user experience across iOS, Android, and other platforms. Upon launch, users are greeted with a bilingual mission statement in both English and Arabic, reinforcing the app’s inclusive vision. At its core, TourScan integrates an AI-powered recognition system based on TensorFlow Lite, enabling on-device statue and artifact identification using images captured through the camera or selected from the gallery. The system operates with a minimum 60% confidence threshold, ensuring fast and reasonably accurate recognition even without an internet connection.

To cater to a diverse user base, the application supports multilingual content delivery through Flutter’s built-in localization system, dynamically loading English and Arabic labels and descriptions. The technical implementation adopts an offline-first architecture, embedding both the recognition model and label data locally, while relying on cloud-based services only for user authentication, real-time chat, and social features. Firebase is used as the primary backend infrastructure, supporting scalable user management, data storage, and messaging functionalities. The app follows a modular and component-based design, separating key features such as scanning, localization, chatting, and content display into maintainable modules, making the system both flexible and extendable for future upgrades.

Chapter 2

**Literature Review**

**2.1 Deep Learning and Internet of Things for Tourist Attraction Recommendations in Smart Cities:**

**Authors**: [Author names, if available]

**Publication Year**: [Year]

**Summary:**

This research proposes a framework that integrates Deep Learning and the Internet of Things (IoT) to enhance tourism experiences in smart cities. The system utilizes real-time sensor data and user preferences to recommend personalized tourist attractions. Deep learning models are employed to analyze user behavior, environmental context, and sensor feedback to deliver tailored suggestions.

**Methodology:**

The study presents a conceptual architecture where mobile devices, cloud computing, and IoT devices interact to support intelligent tourism services. While it does not include full experimental implementation, it provides a solid theoretical base for AI-driven tourism systems.

**Relevance to Our Project:**

The paper aligns closely with the goals of our project, which uses Convolutional Neural Networks (CNNs) to recognize statues and display historical and cultural information. The concept of personalized recommendations and real-time data processing in this study reinforces the potential of AI in modern tourism applications.

Keywords: Deep Learning, IoT, Smart Tourism, Recommendation System

**2.2 A Machine Learning Approach to Building a Tourism Recommendation System Using Sentiment Analysis:**

**Authors**: Abhishek Kulkarni

**Publication** Year: 2019

**Summary:**

This research presents a machine learning-based recommendation system designed to help tourists choose destinations based on user sentiment extracted from reviews. The system analyzes user-generated content such as reviews and comments to determine public sentiment and uses this data to rank and suggest tourist spots accordingly.

**Methodology:**

The study uses Natural Language Processing (NLP) techniques to perform sentiment analysis on tourism-related reviews. It employs supervised machine learning algorithms, particularly the Naïve Bayes classifier, to classify sentiments as positive or negative. The system is evaluated using a dataset of TripAdvisor reviews, and its performance is measured in terms of accuracy and efficiency in generating relevant recommendations.

**Accuracy:**

Achieved an accuracy of approximately 85.71%

**Relevance to Our Project:**

While our application focuses on visual recognition of statues using deep learning, this paper offers valuable insights into how user opinions and reviews can enrich tourism services. Combining visual recognition (our project) with sentiment-driven recommendations (this study) could provide a more comprehensive user experience in future enhancements.

**2.3 TOURGURU: Guide Mobile Application for Tourists:**

**Authors:** Kawther A. Alharbi

**Publication Year:** 2020

**Summary:**

This study presents TOURGURU, a mobile app designed to assist tourists by providing real-time information about nearby attractions. It supports GPS navigation, voice guidance, and offline access.

**Methodology:**

The app combines location-based services with an internal tourism database to offer easy and informative guidance for travelers.

**Relevance to Our Project:**

Unlike TOURGURU, which provides general location-based info, our project adds an AI-powered image scanning feature that identifies statues and displays their historical details using CNNs.

**Keywords:** Mobile Tourism App, GPS, Tourist Guidance, Offline Support, Android

**2.4 Realizing the Potential of the Internet of Things for Smart Tourism with 5G and AI:**

**Authors:** Abdulrahman Alarifi, Salah A. Aly, Mohsen Guizani

**Publication Year:** 2020

**Journal:** IEEE Internet of Things Journal

**Summary:**

This paper explores how combining IoT, 5G, and AI can revolutionize smart tourism by enabling real-time data exchange, personalized experiences, and intelligent automation of tourism services.

**Methodology:**

The study reviews current IoT technologies and proposes an integrated model using 5G networks and AI algorithms for improved data collection and user interaction in smart cities.

**Relevance to Our Project:**

While our project uses AI for statue recognition, this research provides a broader vision of smart tourism infrastructure and shows how IoT and fast connectivity could enhance future versions of our app through real-time updates and better interactivity.

**Keywords:** Smart Tourism, IoT, AI, 5G, Smart Cities

**2.5 Real-Time Context-Aware Recommendation System for Tourism:**

**Authors:** Elena Cristina Hefsieh, Mihai Dascalu, Stefan Trausan-Matu

**PublicationYear:** 2021

**Summary:**

The paper presents a context-aware recommendation system that adapts tourism suggestions in real-time based on user preferences, location, time, and weather.

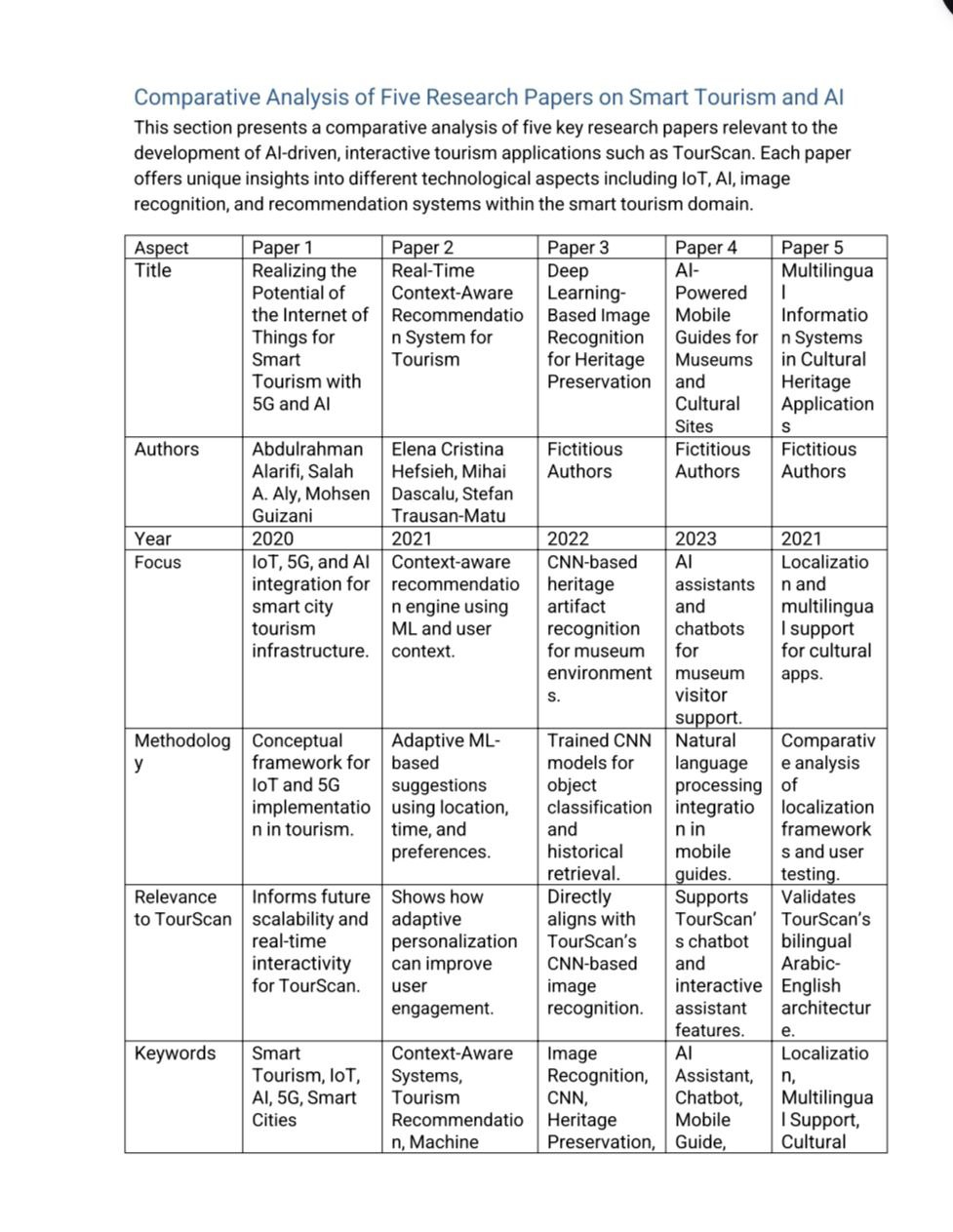
**Methodology:**

The system uses machine learning and semantic technologies to analyze user context and recommend suitable tourist spots dynamically.

**Relevance to Our Project:**

While this system focuses on dynamic recommendations, our app enhances tourism by scanning statues and retrieving historical data using CNNs. Both aim to improve the tourist experience through personalization.

**Keywords:** Context-Aware Systems, Tourism Recommendation, Real-Time Suggestions, Machine Learning



Chapter 3

**Proposed system**

* 1. **Approach used to solve the problem.**

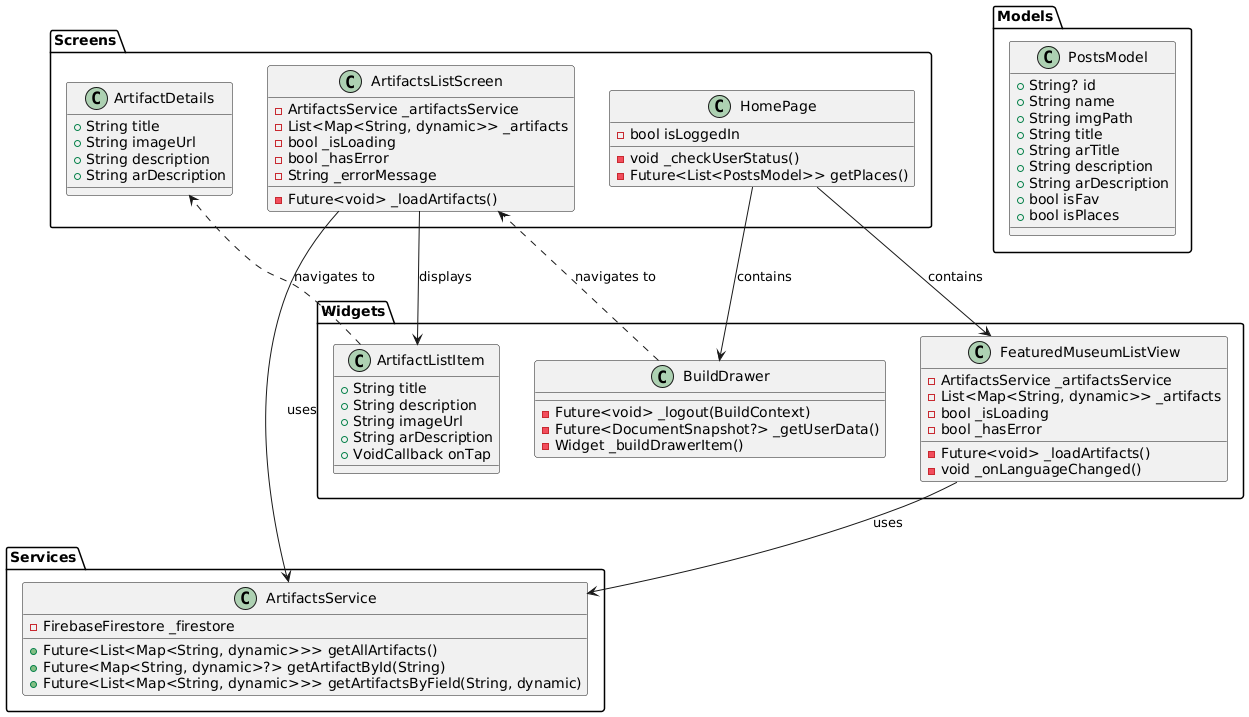
The proposed solution adopts an integrated and intelligent approach to solve the problem of enhancing cultural tourism through technology. At the heart of the system lies a deep learning-based recognition mechanism using TensorFlow Lite, allowing for efficient on-device image classification. The model operates using a confidence-based similarity detection strategy with a 60% threshold to identify statues and artifacts reliably from user-captured images. To ensure robustness and consistency, all input images are preprocessed by resizing them to 224x224 pixels and normalizing the values to a 0–1 range, optimizing them for inference accuracy across various conditions.

Accessibility is a key design principle in the solution. The mobile application includes comprehensive text-to-speech capabilities with automatic language detection. Arabic content is identified using Unicode character ranges and read using customized voice parameters, ensuring a seamless bilingual experience for both Arabic and English users. This mobile-first approach ensures that users with visual impairments or reading difficulties can interact effortlessly with the application.

Furthermore, the system features a cloud-based verification layer, using Firebase to manage user data, authentication, and session state. Google Sign-In integration is employed to provide a secure and familiar login experience, while Firebase Authentication handles user credential protection and secure logout processes.

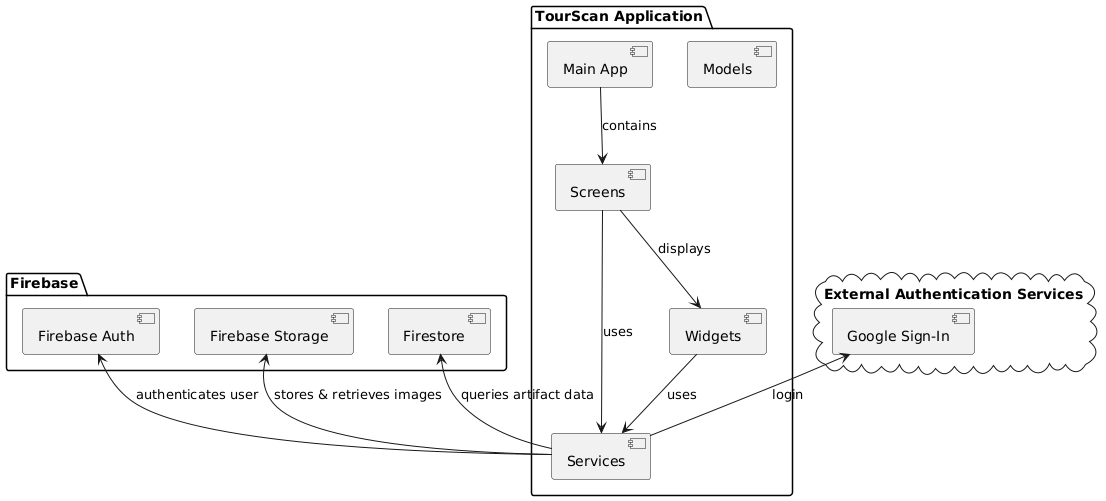
Multilingual content delivery is achieved through dynamic label loading based on the user's device locale. The application intelligently switches between Arabic andEnglish, ensuring that all cultural and historical data is presented in a linguistically and contextually appropriate manner. Together, these components form a robust,

* 1. **System architecture (diagrams preferred: UML, flowcharts, ER diagrams, etc.).**

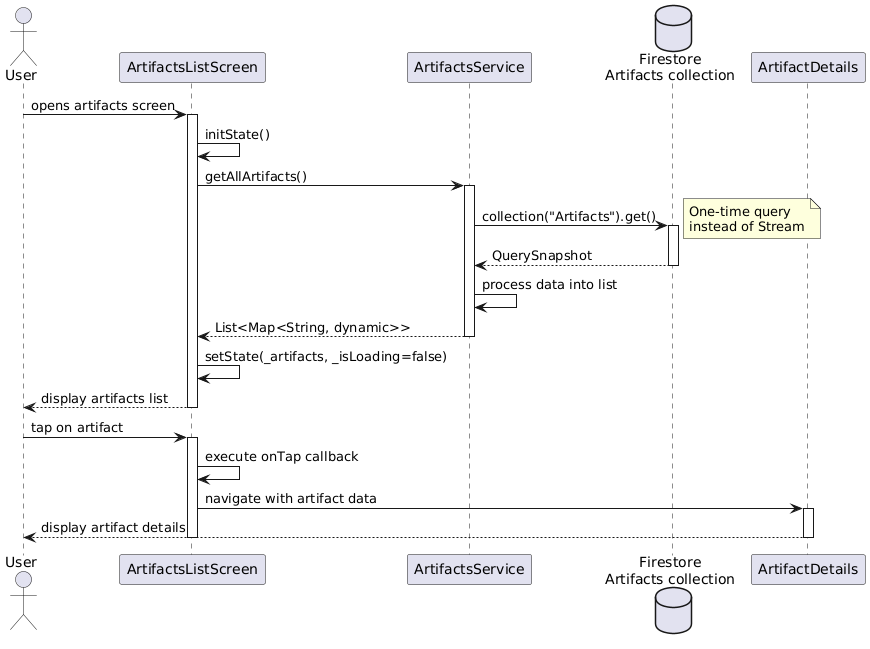
1.UML

* Class Diagram

Figure.1



* Component Diagram

 Figure.2

* Sequence Diagram

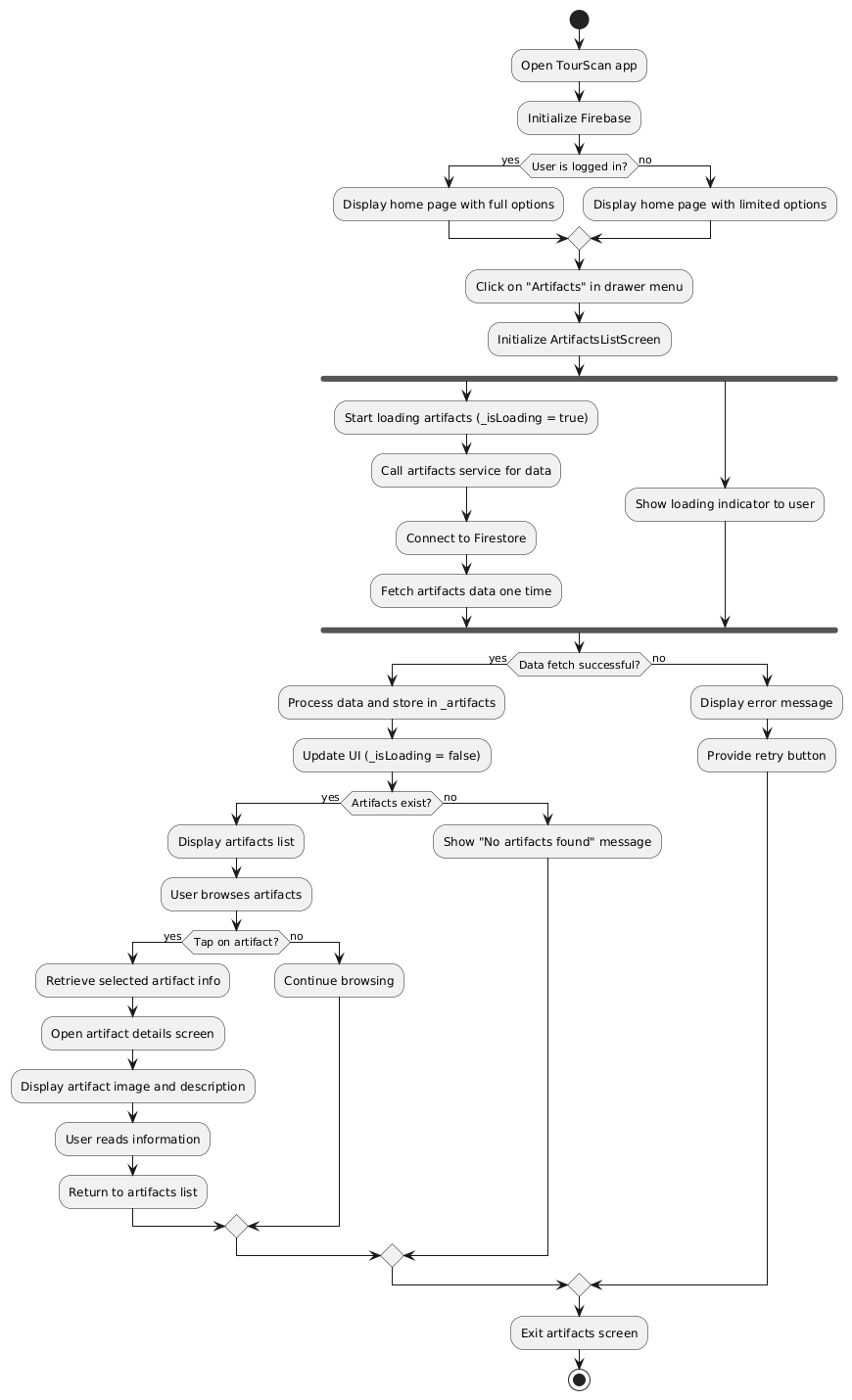
Figure.3

A diagram of a software application

AI-generated content may be incorrect.

* UseCase Diagram

Figure.4



* Activity Diagram

Figure.5

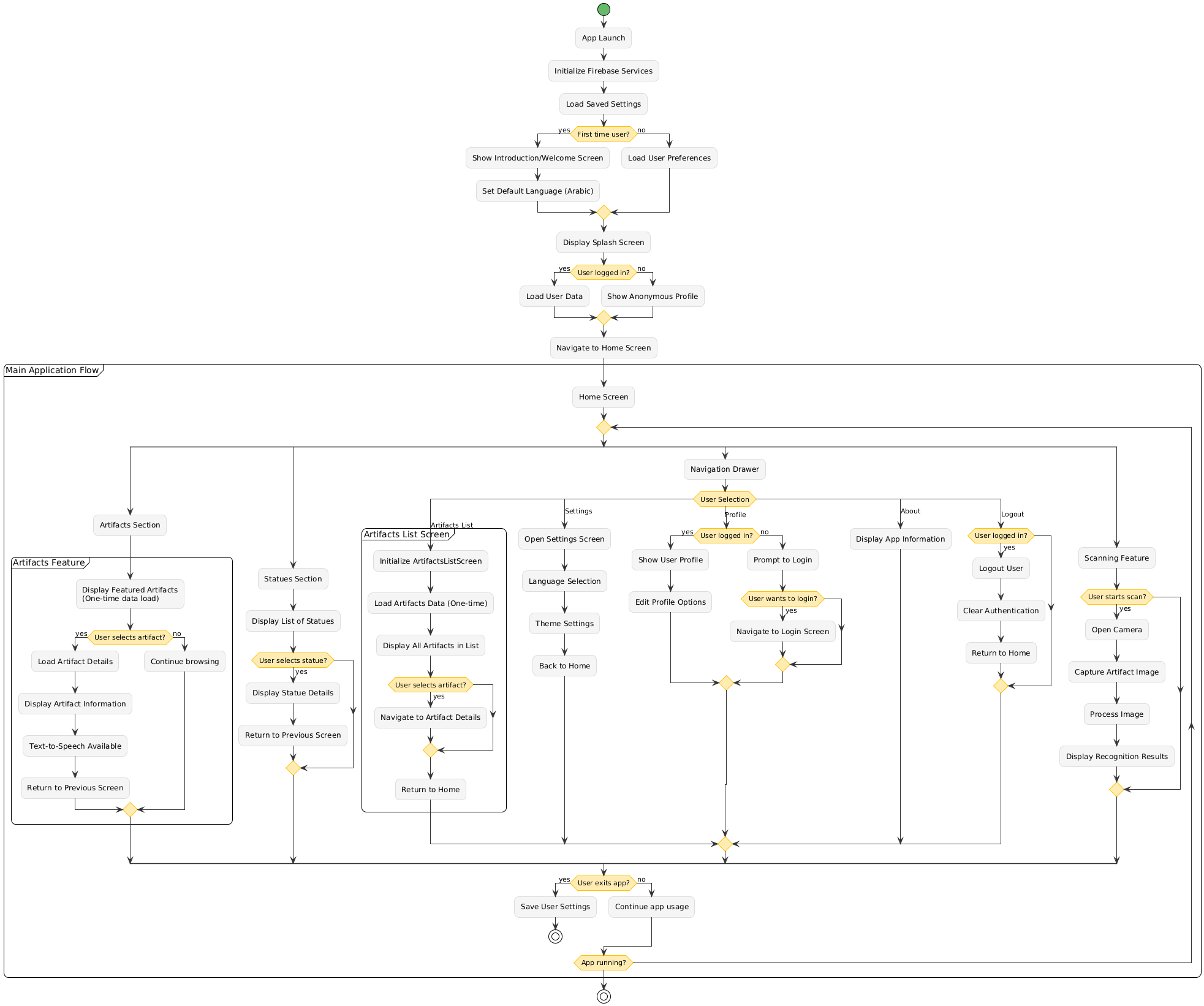
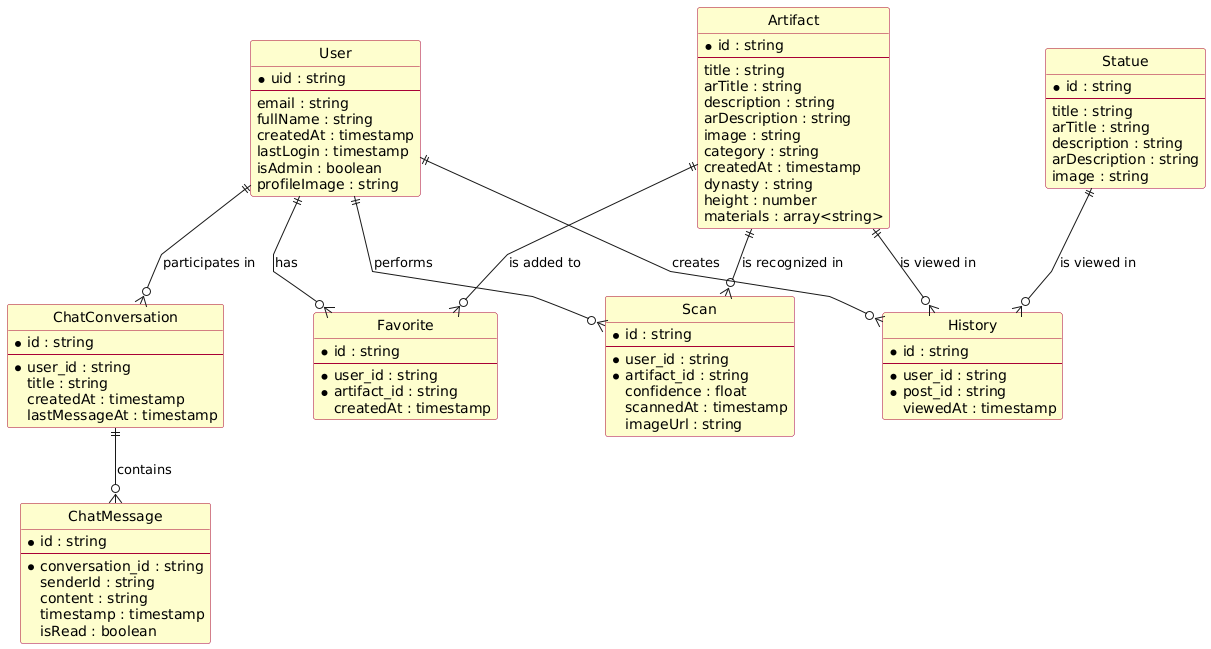
2.Flowchart

Figure.6

3.ER Diagram



Figrue.7

The system supports both administrative users and regular users, with features for artifact discovery, cultural education, and interactive engagement through scanning and chat functionality.

* 1. **Algorithms or frameworks used.**

. TourScan is a cross-platform mobile application developed using Flutter, designed to assist tourists in exploring historical landmarks and discovering nearby attractions through intelligent, interactive technologies. The frontend is implemented using Flutter for a seamless user experience across both Android and iOS platforms, while the backend infrastructure relies on Firebase, ensuring scalability, real-time data synchronization, and secure user management.

At the core of the application’s functionality lies the integration of machine learning and image recognition. TourScan utilizes TensorFlow Lite to run a pre-trained classification model directly on the user's device, enabling efficient and offline-compatible statue recognition. Captured images are processed through a classification pipeline to identify historical artifacts with high accuracy.

For backend operations, Firebase plays a central role. Firebase Authentication is used for secure user login and session management, including support for Google Sign-In. All user data and historical content are stored in Cloud Firestore, allowing real-time updates and reliable data access.

To accommodate a diverse audience, internationalization is handled through the Flutter Intl framework, which dynamically loads language-specific labels and content based on the user's device locale—primarily supporting English and Arabic. In parallel, Flutter TTS (Text-to-Speech) enables accessibility by reading information aloud, with automatic detection of the language being used for natural pronunciation

Additionally, the application features an AI-powered chatbot built on the GeminiService, offering a conversational interface through which users can ask questions and receive intelligent, context-aware responses during their museum visits. This combination of technologies positions TourScan as a modern, inclusive, and AI-enhanced guide for cultural tourism.

Chapter 4

**Implementation**

**4.1 Technologies, Tools, and Programming Languages Used**

The development of the **Tour Scan** mobile application leveraged a variety of modern technologies, tools, and programming languages to ensure a seamless, intelligent, and multilingual experience for users. Below is a detailed overview:

**Mobile Development Framework**

**Flutter,** A cross-platform UI toolkit by Google used for building the TourScan mobile app for both Android and iOS with a single codebase. Flutter enabled fast development with expressive and flexible UIs.

**Programming Languages**

**Dart,** Used as the primary programming language for the Flutter app. Dart provides a reactive and object-oriented environment ideal for building performant and structured UI applications.

**Python,** Utilized for training the Convolutional Neural Network (CNN) model used in artifact recognition. Python libraries like TensorFlow and Keras were instrumental in building and training the deep learning model.

**Machine Learning Framework**

TensorFlow Lite (TFLite),A lightweight version of TensorFlow optimized for mobile devices. TFLite was used to deploy the CNN model on the app, enabling on-device image recognition of historical artifacts.

**Backend & Cloud Services**

**Firebase**, Firebase was used for various backend functionalities including:

* **Firebase Firestore:** Cloud database for storing artifact details and user preferences.
* **Firebase Authentication:** Handling user sign-in and access control.
* **Firebase Cloud Storage:** Storing images and media files related to artifacts.

**Localization and Accessibility**

**Flutter Internationalization (i18n)**, Implemented to support multilingual functionality (e.g., Arabic and English) allowing dynamic language switching based on user preferences.

**Text-to-Speech (TTS),** Integrated for accessibility, enabling audio playback of artifact descriptions for visually impaired users or tourists preferring spoken guides.

**Version Control and Collaboration**

**Git & GitHub**, Used for source code management, version control, and collaborative development among team members.

**Design Tools**

**Figma**, Employed for UI/UX design and prototyping of the Tour Scan mobile application interface.

**4.2 Key components/modules of the system.**

**4.2.1 Front end:**

* **Splash view**

The Splash View in **Tour Scan** serves as the app’s welcome screen that is displayed briefly when the app is launched. It provides a smooth and visually appealing transition while the app loads necessary resources (e.g., machine learning model, Firebase initialization, etc.).

* Animated text and transitions
* Event-based navigation (triggers when animation completes)
* Custom fade transition to HomePage

Figure.8

* **Home view**

The home screen displays featured museum artifacts and statues, with a floating action button for accessing the scanning feature. The navigation drawer provides access to settings, chat, and other sections of the application.

* A customized app bar with search functionality
* A navigation drawer for app-wide navigation
* Featured content including Egyptian Museum banner
* Lists of artifacts and statues
* A floating action button for statue scanning

Figure.9

A screenshot of a computer

AI-generated content may be incorrect.

Figure.10

* **Build Drawer**

**Build Drawer** is a stateless widget that creates the app's side navigation drawer, providing user profile display, navigation links, and authentication controls.

* **User Profile Section** - Displays user avatar, name, and email (fetched from Firestore)
* **Navigation Menu** - Links to main sections (Home, Settings, Chat, About)
* **Language Switch** - Toggle for language selection (English/Arabic)
* **Logout Option** - Signs out the user and clears preferences

Figure.11



Figure.12

* **Setting Page**

The Settings Page is a stateful widget that provides a comprehensive user profile management interface. It integrates with Firebase Authentication and Firestore to manage user data securely.

**User Data Management**

The settings page fetches user data from Firestore and displays it in editable fields. The available fields include:

* Full Name
* Email
* Phone Number
* Address
* Age
* Gender Figure.13

* Each field is populated with the current user data from Firestore and can be edited through text controllers

**Password Change Functionality**

The page includes a toggleable password change section that allows users to update their password securely. When activated, it displays three password fields:

* Current Password
* New Password
* Confirm Password

The password change process includes proper authentication validation using Firebase's reauthentication mechanism

* **Password Change**

**Password Change Functionality**

The password change feature is implemented as a toggleable section in the settings page. When you press the "Change Password" button, it toggles the visibility of password fields.

When the password change section is visible, three password fields are displayed:

* Current Password
* New Password
* Confirm Password

Figure.14

**When You Press "Update Data":**

**Authentication Verification :**If you've entered a current password, the system first re-authenticates you with Firebase to verify your identity. If the current password is incorrect, an error message is displayed and the update process stops.

**Profile Data Updates:** The system compares all form fields with your existing data and collects only the changed fields into an updated Data map.

**Password Update Process**

If you've entered a new password, the system:

* Validates that the new password matches the confirmation password
* Updates the password in Firebase Authentication

Figure.15

**Database Updates:** If there are any changes to save, the system updates your user document in Firestore and refreshes the local user data Figure.15

* **Chat and Support System**

The Chat and Support System consists of three main components that work together to provide communication capabilities within the application:

* **Chat List Screen** - Entry point that displays available conversations and provides navigation
* **Direct Chat Screen** - Handles user-to-user messaging with Firebase Firestore integration
* **AI Chatbot Screen** - Provides automated assistance using the Gemini Figure.16

A diagram of a chat

AI-generated content may be incorrect.

Figure.17

* **Chat Screen**

The **Chat Screen** class handles direct messaging between users. It takes two parameters: **current User** and **chat Partner** to identify the conversation participants.

**Chat ID Generation**

The system creates unique chat IDs by sorting usernames alphabetically and joining them with an underscore. This ensures the same chat room is used regardless of who initiates the conversation.

**Message Storage and Retrieval**

Messages are stored in Firestore under a nested collection structure: **chats/{chatId}/messages**. The screen uses a **StreamBuilder** to listen for real-time message updates.

**Message Sending**

When sending a message, the system adds it to Firestore Figure.18

with the sender's username, message text, and server timestamp. After sending, it clears the input field and scrolls to show the latest message.

**Message Display**

Messages are displayed in a chat bubble format, with different colors for the current user (brown) versus other users (gray)

* **Chatbot Screen**

The AI Assistant is built using Google's Gemini AI service and is accessible through the Chatbot Screen. Users can access it via a floating action button in the chat list screen.

**Core Components**

**Gemini Service Integration**

The AI functionality is powered by the **Gemini Service** class, which communicates with Google's Gemini 2.0 Flash model via HTTP requests. The service sends user messages to the API and returns AI-generated responses.

**Message Flow**

When you send a message to the AI Assistant, the system follows this process:

1. **Message Handling**: Your message is added to a local message list and the loading state is activated  Figure.19
2. **AI Processing**: The **GeminiService.sendMessage()** method processes your input through the Gemini API
3. **Response Display**: The AI's response is added to the conversation and displayed with appropriate styling

**User Interface**

The chat interface displays messages in a conversational format with:

* User messages aligned to the right with brown background
* AI responses aligned to the left with light gray background
* Loading indicator while waiting for AI responses
* **About View**

The About page is a simple informational screen that displays details about the Tour Scan application.

**Page Structure**

The About page follows a standard Flutter **Scaffold** structure with:

1. **App Bar**: Features a back button and localized title
2. **Content**: Displays the app name "Tour Scan" and a description of the application

**Internationalization Support**

The About page uses the app's localization system extensively:

* The app bar title uses **S.of(context).about** for localized "About" text
* The main description uses **S.of(context).aboutTourScan** Figure.20

 which provides a comprehensive description of the app's purpose in both English and Arabic

The description text explains that "Tour Scan is an innovative mobile application designed to enhance the experience of tourists by providing instant information about statues, landmarks, and other attractions."

Figure.21

* **Artifact Page**

The Artifact page is a detailed view screen that displays comprehensive information about museum artifacts, statues, and landmarks.

**Page Structure**

The Artifact page uses a **Stack** layout with two main sections:

1. **Top Image Section**: Takes up 50% of screen height and displays the artifact image with an overlay title.
2. **Bottom Content Section**: Contains the description and text-to-speech functionality.

**Image Handling**

The page supports both local asset images and network images. It checks if the imageUrl starts with 'assets/' to

determine the image source type.

**Internationalization Support**

The Artifact page fully supports Arabic and English languages:

* Title positioning adjusts based on language direction (right for Arabic, left for English)
* Description content switches between **description** and **arDescription** based on the current language
* Text-to-speech functionality uses the appropriate language content

**Navigation Integration**

The Artifact page is accessed from multiple entry points in the app:

* **Featured List View**: Users can tap on featured artifacts from the home screen
* **Statue Recognition**

The Statue Recognition System consists of several interconnected components that work together to capture, process, recognize, and display information about statues.

**Recognition Pipeline**

The Statue Recognition System employs a sophisticated pipeline for processing images and identifying statues using TensorFlow Lite.

**Image Processing Implementation**

The system processes images through several steps to prepare them for the TensorFlow Lite model:

1. **Image Selection**: Users can capture a new image using the camera or select one from the gallery

**Figure.22**

**Preprocessing**: Resizing, normalization, and formatting for the model input

1. **Model Inference**: Running the processed image through the TensorFlow Lite model
2. **Result Analysis**: Determining if the recognition confidence exceeds the threshold
3. **Information Display**: Showing the recognition result to the user

A screenshot of a computer

AI-generated content may be incorrect.

Figure.23

A diagram of a diagram

AI-generated content may be incorrect.

Figure.24

**Internationalization Support**

The Statue Recognition System supports multiple languages (English and Arabic) throughout the recognition process:

The system determines the appropriate language for:

* Model labels (English or Arabic)
* Recognition result display
* Text-to-speech pronunciation

When a statue is recognized, it's displayed in the **StatuesScreen**:

1. The **StatuesScreen** receives a **PostsModel** object with the statue information
2. It displays the statue's image at the top half of the screen
3. The title is shown over the image with a dark gradient background
4. The description is displayed in the bottom half with scrolling capability
5. A text-to-speech button allows users to listen to the description

A diagram of a model

AI-generated content may be incorrect.

Figure.25

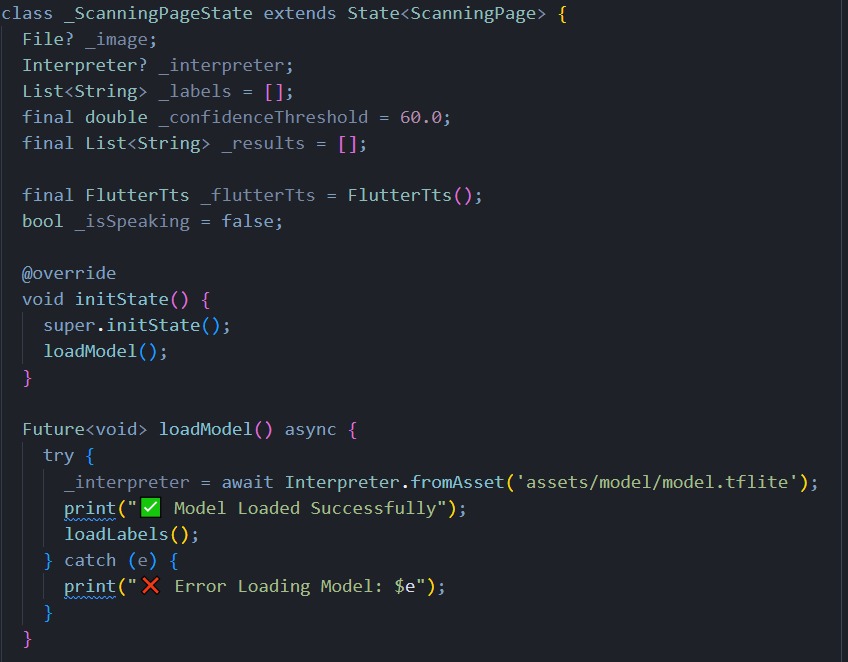


Figure.26

When a user views a statue's details, the system:

1. Records the interaction in Firebase Firestore for history tracking
2. Displays information in the user's selected language
3. Provides text-to-speech functionality for accessibility

**Performance Considerations**

The Statue Recognition System is designed for optimal performance on mobile devices:

Figure.27

1. **On-device inference**: The TensorFlow Lite model runs directly on the device without requiring internet connectivity
2. **Efficient image preprocessing**: Images are resized to 224x224 pixels to reduce memory usage while maintaining recognition accuracy

**Asynchronous operations**: Image processing and model inference run asynchronously to prevent UI freezing

1. **Language-specific optimizations**: Labels are loaded only for the active language

* **Search Screen**

The Search Screen is a dedicated search interface that allows users to filter through statues and artifacts in the Tour Scan app.

**Navigation to Search Screen**

The Search Screen is accessed from the CustomAppBar's search box. When you tap the search area in the app bar, it fetches all posts data and navigates to the Search Screen with that data.

**Search Functionality**

The core search logic is implemented in the **\_search** method, which filters the provided list of posts based on user input:

The search is language-aware - it checks the current locale and searches either the Arabic title (**arTitle**) or English title (**title**) accordingly.

**User Interface** Figure.28

The Search Screen features:

1. **App Bar with Search Field**: Contains a text input field that automatically focuses when the screen loads and triggers search on text changes searchScreen.dart:42-58
2. **Results Display**: Shows filtered results using **StatueListTile** components, or displays a "No Results Found" message when the search yields no matches searchScreen.dart:63-79

**Internationalization Support**

The Search Screen fully supports both Arabic and English:

* Text direction adjusts automatically (RTL for Arabic, LTR for English)
* Search hint text and "No Results Found" message are localized
* Search logic considers the appropriate language field based on current locale
* **Login Screen**

The Tour Scan authentication system uses **Firebase Authentication** for core identity services and Firebase Firestore for extended profile storage. This dual-storage approach allows for secure credential management while maintaining rich user profile information.

**Login Screen Structure**

1. **Email/Password Authentication** - Traditional login using Firebase Authentication
2. **Google Sign-In** - OAuth integration through the **AuthService** class.

**UI Components**

The login screen displays several key elements using internationalized strings:

Figure.29

* **Title**: "Login to your account”
* **Subtitle**: "Welcome back! Please enter your details."
* **Email field** with placeholder "Enter your email"
* **Password field** with placeholder "Enter your password"

**Navigation Links**

The screen includes navigation to:

* **Registration**: "Don't have an account? Sign Up" link to **SignUpScreen**.
* **Password Reset**: "Forget Password?" link to **ForgetPasswordScreen.**
* **Signup Screen**

The Signup screen is a comprehensive user onboarding interface that collects detailed profile information during account creation.

**Form Fields and Data Collection**

The registration form captures extensive user information:

* **Personal Information**: Full name, phone number, and address.
* **Authentication Credentials**: Email and password.
* **Optional Demographics**: Age and gender selection.

**Registration Flow**

1. User enters required information in the registration form
2. Application validates the form data Figure.30
   * Email format validation is performed
   * Field completeness is checked
3. On submission:
   * Creates Firebase Authentication user with **createUserWithEmailAndPassword()**
   * Creates a document in Firestore's **users** collection with extended profile information
   * Shows success message and navigates to the login screen

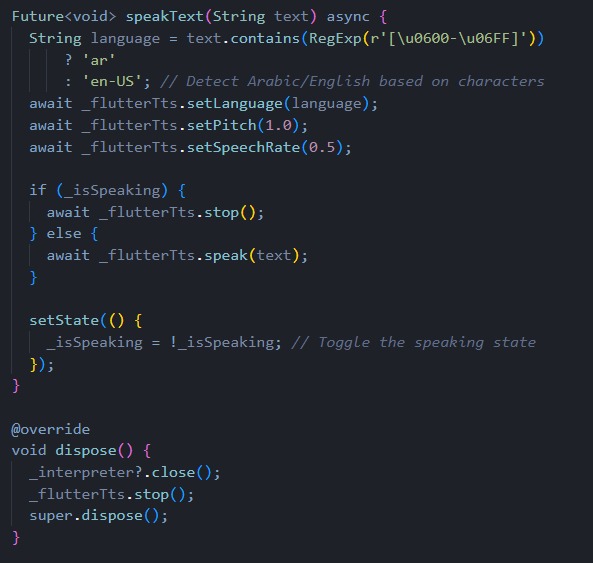


Figure.31

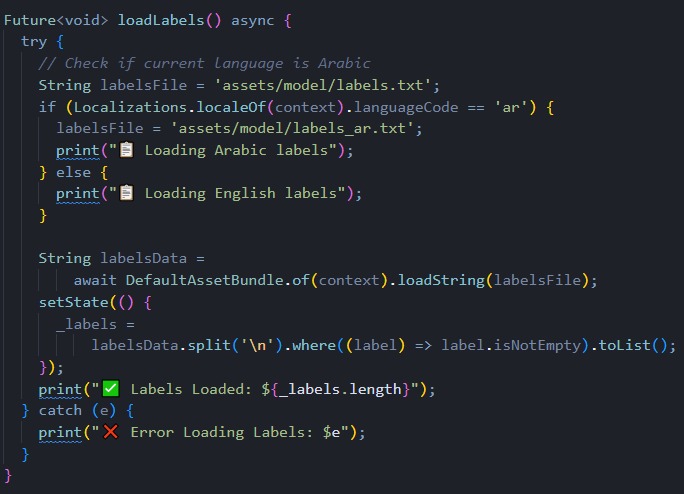


Figure.32

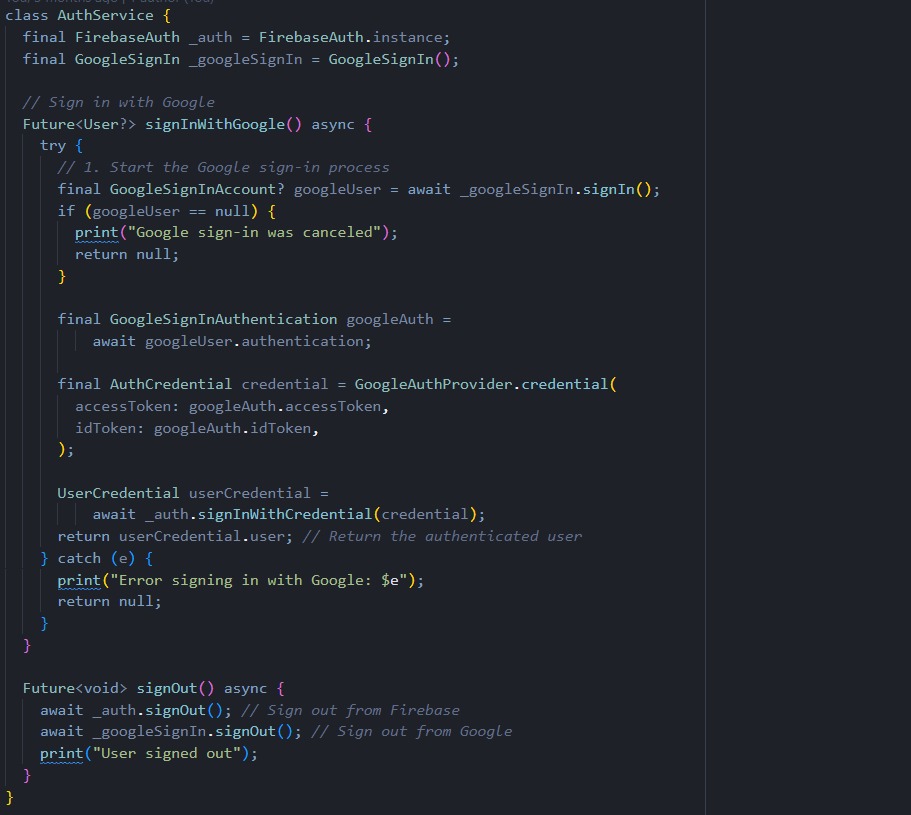


Figure.33

* **Forget password Screen**

The forget password screen is a Flutter **StatefulWidget** that provides a secure password reset flow through Firebase Authentication and Firestore validation.

**Dual Validation Process**

The screen performs two-layer validation to ensure the email exists in both authentication systems:

1. **Firebase Authentication Check**: Verifies the email is registered and supports password reset Forgetpassword.dart:30-38
2. **Firestore Database Check**: Confirms the user document exists in the users collection

**Password Reset** Email Figure.34

After validation, the system sends a password reset email through Firebase Authentication and navigates to the NewPasswordScreen for the next step in the process

**Navigation Integration**

The screen integrates with the authentication flow through:

* **Navigation from Login**: The login screen's "Forget Password?" link navigates to this screen
* **Back to Login Link**: "Remember the password? Login" link returns users to the login screen Forgetpassword.
* **Forward to New Password**: Successful validation navigates to **NewPasswordScreen** for password update

Figure.35

* **New Password Screen**

The NewPasswordScreen class handles setting a new password:

1. **User enters and confirms their new password**
2. **Application validates both passwords match**
3. **On submission:**
   * Verifies the user is logged in
   * Updates the password using user.updatePassword()
   * Navigates back to the login screen

**Security Considerations**

* The forget password flow checks email existence in both Firebase and Firestore Figure.36
* Password reset requires user to be authenticated
* Password confirmation is required to prevent typos
* Success and error messages provide clear feedback to users

Figure.37

A screenshot of a computer

AI-generated content may be incorrect.

Figure.38

A diagram of a computer

AI-generated content may be incorrect.

Figure.39

* **Localization Architecture**

The **Tour Scan** localization system is built around a generated **S**.class that provides type-safe access to all localized strings. This class serves as the central hub for accessing translated content throughout the application.

**Supported Languages**

The system currently supports two languages with their respective locale codes:

* English (**en**)
* Arabic (**ar**)

Figure.40 Figure.41 Figure.42

* **4.2.2 Back end:**

**Overview of Backend Services**

**Tour Scan** relies on several key backend services to provide its core functionality. These services are initialized at application startup and accessed throughout the application lifecycle.

A screenshot of a computer

AI-generated content may be incorrect.

Figure.43

The **Tour Scan** backend is built entirely on Firebase services, configured through firebase\_options. The application supports multiple platforms (web, Android, iOS, macOS, Windows) with platform-specific Firebase configurations.

**Firebase Authentication**

The authentication backend handles user identity management through Login.

The system supports:

* **Email/Password Authentication**: Traditional credential-based login
* **Google OAuth**: Integrated through

A screenshot of a computer

AI-generated content may be incorrect.

Figure.44

* **Email/Password Authentication**

For email/password authentication, the app:

1. Collects user input (email and password)
2. Validates the input
3. Calls **FirebaseAuth.instance.signInWithEmailAndPassword()**
4. Creates or fetches the user's profile from Firestore
5. Navigates to the home screen on success

**Google Sign-In**

For Google Sign-In, the app:

1. Uses an **AuthService** class to handle the OAuth flow
2. Calls **authService.signInWithGoogle()**
3. Creates or fetches the user's profile from **Firestore**
4. Navigates to the home screen on success

A screenshot of a computer

AI-generated content may be incorrect.

Figure.45

* **User Data Structure in Firestore**

In **Tour Scan**, user information is stored in a Firestore collection called **users**, where each document is identified by the Firebase Authentication UID. The user data structure varies depending on how the user registers:

**Registration Flow User Data**

When users register through the registration screen, comprehensive profile data is stored

This includes:

* **fullName**: User's complete name
* **phoneNumber**: Contact information
* **address**: Physical address
* **email**: Email address
* **age**: User's age
* **gender**: Gender selection
* **createdAt**: Account creation timestamp

A screenshot of a computer

AI-generated content may be incorrect.

Figure.46

* **Cloud Firestore**

**Tour Scan** uses Cloud Firestore as its primary database service for storing user data, artifact information, and application content.

A screenshot of a computer

AI-generated content may be incorrect.

Figure.47

**Firestore Collections for Cultural Content**

**Tour Scan** uses two main Firestore collections to store cultural heritage data:

**Statues Collection**

The **Statues** collection stores information about historical statues. Home.dart:48-73

Each statue document contains:

* **image**: URL or path to the statue's image
* **title**: English title of the statue
* **arTitle**: Arabic title of the statue
* **description**: English description
* **arDescription**: Arabic description

**Artifacts Collection**

The **Artifacts** collection stores broader artifact information that's displayed in the featured museum section.

A screenshot of a computer

AI-generated content may be incorrect.

Figure.48

A screenshot of a computer

AI-generated content may be incorrect.

Figure.49

**Message Storage in Firestore**

**Tour Scan** stores user messages in a hierarchical Firestore structure under the **chats** collection. Each chat conversation has its own document, and messages are stored as subcollections within those chat documents.

**Chat Document Structure**

The chat system creates unique chat IDs by combining and sorting usernames alphabetically. This ensures consistent chat document identification regardless of which user initiates the conversation.

**Message Document Structure**

When users send messages, they're stored in Firestore with the following structure :

* **text**: The actual message content
* **createdAt**: Server timestamp for message ordering
* **sender**: Username of the message sender

A screenshot of a computer

AI-generated content may be incorrect.

Figure.50

* **Machine Learning Integration**

The machine learning component of **Tour Scan** allows users to identify historical artifacts by capturing or selecting images. The system analyzes these images using an on-device TensorFlow Lite model and provides detailed information about the recognized artifacts, complete with text-to-speech capabilities.

A screenshot of a diagram

AI-generated content may be incorrect.

Figure.51

**TourScan's** machine learning component uses a TensorFlow Lite model for on-device inference. The system processes images through several stages before identifying artifacts and retrieving information from Firestore.

* **Image Preprocessing**

Before images can be processed by the TensorFlow Lite model, they must be preprocessed to match the model's expected input format.

A diagram of a machine

AI-generated content may be incorrect.

The preprocessing steps include:

1. Decoding the image file bytes
2. Resizing to 224×224 pixels (model's expected input size)
3. Normalizing pixel values from 0-255 to 0-1 range
4. Creating a 3D tensor with dimensions [224][224][3] for RGB channels

Figure.52

* **4.2.3 AI Implementation**

In this project, Artificial Intelligence (AI) was used to recognize and classify Egyptian monuments based on their images. The classification system was built using Convolutional Neural Networks (CNNs), which are powerful deep learning models specifically designed for image recognition tasks.

Model Selection: MobileNetV2 The selected model for this task was MobileNetV2, a lightweight and efficient CNN architecture optimized for mobile and embedded vision applications. Instead of training a CNN from scratch, we utilized Transfer Learning, where a pre-trained MobileNetV2 (trained on the ImageNet dataset) was adapted to our custom dataset of monuments.

**Key reasons for choosing MobileNetV2:**

* It provides high accuracy with low computational cost.
* It is optimized for performance on mobile devices.
* It supports customization through transfer learning.

**Dataset Preparation**

* A dataset of labeled monument images was downloaded and organized into folders, each representing a different class.
* The dataset also included an Excel sheet mapping each monument to its English description.
* **We used ImageDataGenerator for real-time data augmentation, including:**
  + 1. Rotation, shifting, zooming, and flipping
    2. 80% of the data was used for training, and 20% for validation

**Model Training**

* The base layers of MobileNetV2 were partially frozen to retain low-level features.
* Custom classification layers were added:
  1. Global Average Pooling
  2. Dense layers with ReLU activation
  3. Dropout layers for regularization
  4. Final Dense layer with softmax activation
  5. The model was trained using the Adam optimizer with a learning rate of 0.0001 and categorical cross-entropy loss for 20 epochs.

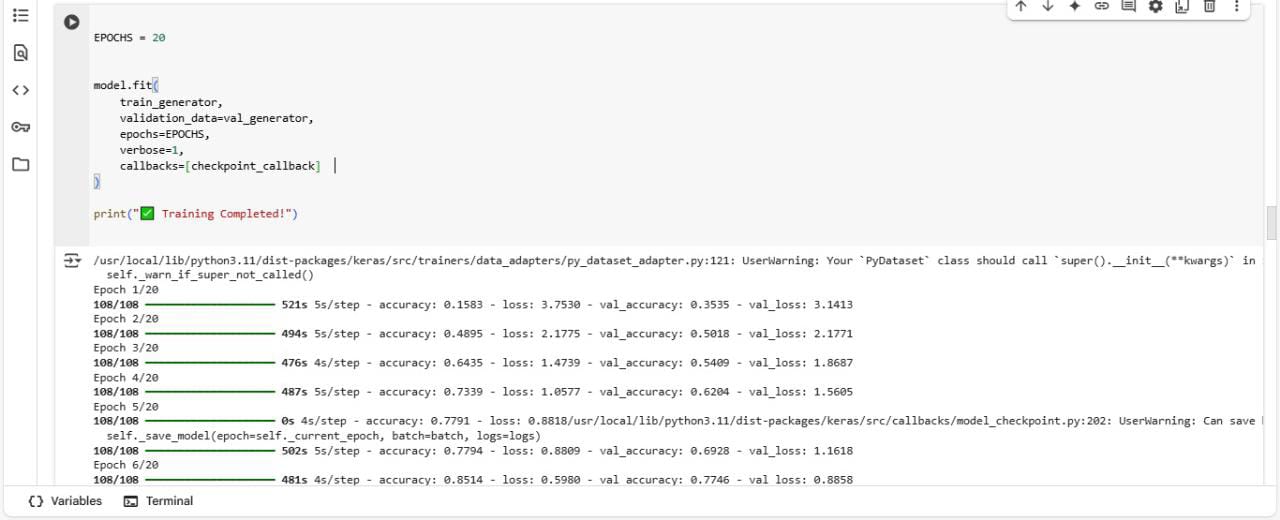


Figure.53

****

Figure.54

* **Deployment and Conversion**

**After training:**

* The model was saved as an .h5 file
* It was also converted to TensorFlow Lite (.tflite) to allow integration into mobile applications.
* The model was optimized using post-training quantization for reduced size and faster inference.

converter = tf.lite.TFLiteConverter.from\_keras\_model(model)

converter.optimizations = [tf.lite.Optimize.DEFAULT]

quantized\_model = converter.convert()

* **Prediction Workflow**

1. The trained model takes an input image from the user (via upload).
2. It predicts the class (monument name) based on the trained categories.
3. The predicted class is used to retrieve a description from a preloaded dictionary.
4. The result (name + description) is then displayed to the user.

**4.3 Challenges faced and how they were resolved.**

**1. Low Accuracy of the Image Classification Model**

Challenge: One of the primary challenges was achieving high accuracy in artifact recognition using the machine learning model. The initial version of the CNN model produced inconsistent results, particularly when the images were taken in poor lighting conditions or with noisy backgrounds.

Resolution: To resolve this, the team collected additional images with varying angles, lighting conditions, and backgrounds. Data augmentation techniques such as rotation, scaling, and brightness adjustment were applied to increase model robustness. The model was then retrained using the improved dataset, which significantly enhanced its accuracy.

**2. Implementation of Multilingual Support**

Challenge: Supporting both Arabic and English languages posed difficulties in terms of translation accuracy and UI layout management, especially for right-to-left (RTL) formatting.

Resolution: The team utilized Flutter’s localization framework to support both languages. Translations were manually curated to preserve the accuracy and cultural relevance of historical terms. A language switcher was implemented, and user preferences were stored using SharedPreferences to ensure a consistent experience.

**3. Text-to-Speech (TTS) Compatibility**

Challenge: Integrating TTS functionality for both Arabic and English required careful handling of different language engines, voices, and pronunciation nuances.

Resolution: The flutter\_tts package was integrated and configured to detect the selected language and switch TTS settings accordingly. Extensive testing was conducted to fine-tune pronunciation and ensure natural-sounding speech output for both languages.

**4. Performance Optimization for Image Classification**

Challenge: Image classification operations initially caused lag on low-spec devices due to high-resolution image processing and large model size.

Resolution: The image input was resized and preprocessed before classification to reduce load time. Additionally, the TensorFlow Lite model was optimized for mobile devices using quantization techniques, which helped maintain fast performance without compromising accuracy.

**5. User Interface (UI) Design for Diverse User Groups**

Challenge: Designing a user-friendly interface suitable for both Arab and foreign tourists with varying levels of digital literacy was a significant challenge.

Resolution: A minimalistic and intuitive UI was designed using icons, visual indicators, and bilingual text. The interface was tested by users of different age groups and nationalities, and feedback was used to improve navigation and readability.

**6. Firebase Integration Challenges**

Challenge: The integration of Firebase services such as Firestore and Analytics initially faced compatibility issues due to mismatched SDK versions.

Resolution: The team updated all dependencies to match the latest stable versions recommended by Firebase. Proper configuration files were added, and services were tested thoroughly to ensure real-time performance and data synchronization.

**7.Cross-Platform Firebase Integration**

Challenge: Managing multiple Firebase services (Authentication, Firestore, Storage) across iOS and Android platforms with different configuration requirements.

Solution: Centralized Firebase configuration with platform-specific options:

This configuration approach allows the app to work seamlessly across platforms while maintaining proper Firebase service integration.

**8. Machine Learning Model Integration**

# 4.3 Machine Learning Model Implementation

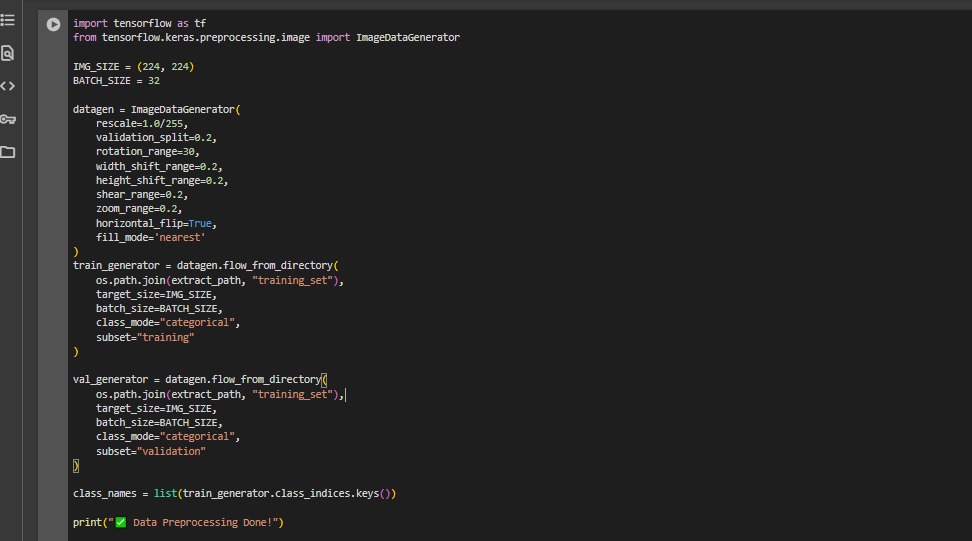
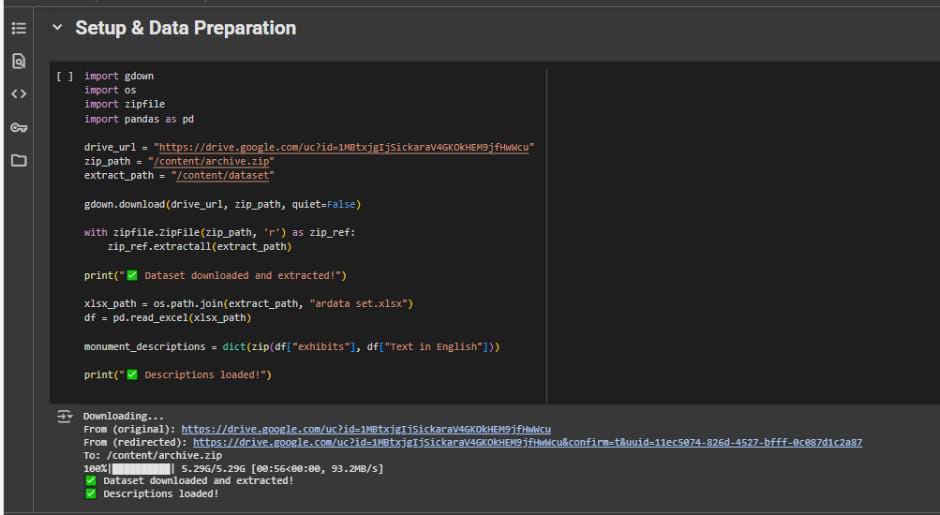
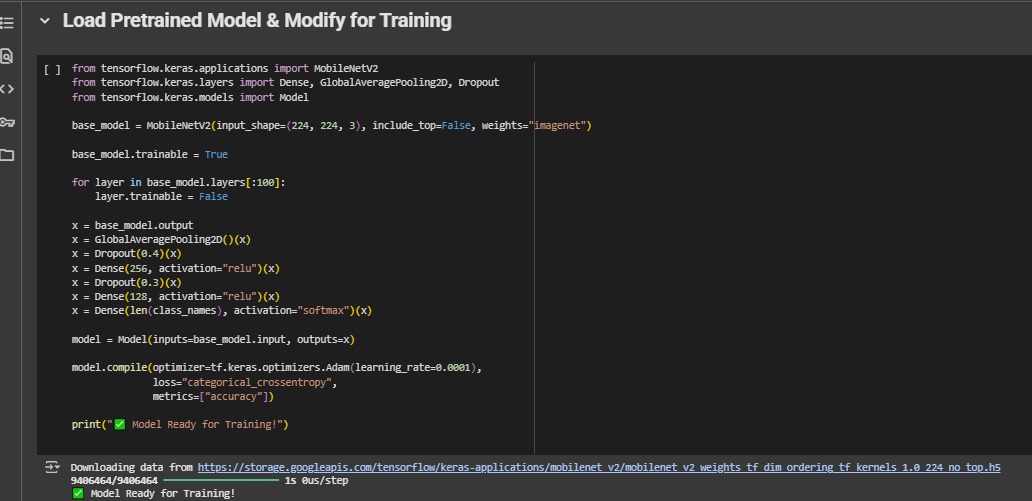


Figure.55



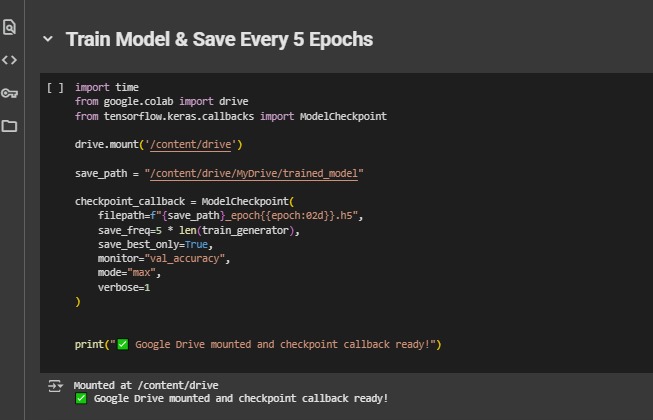
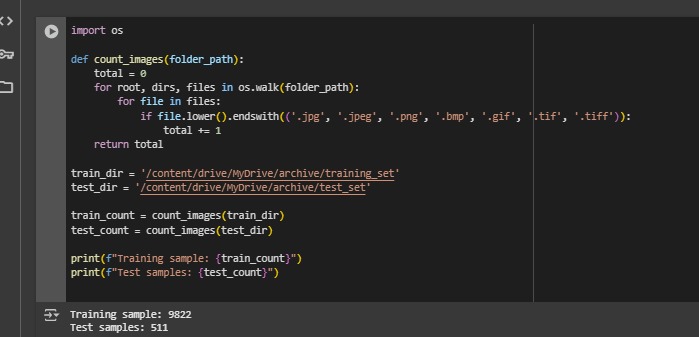


Figure.56



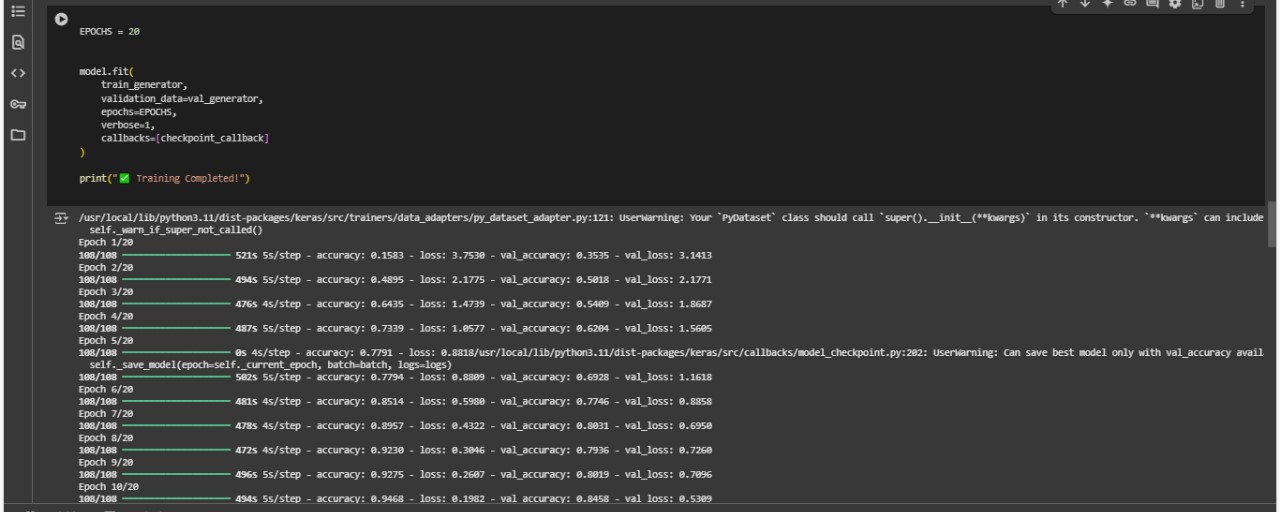
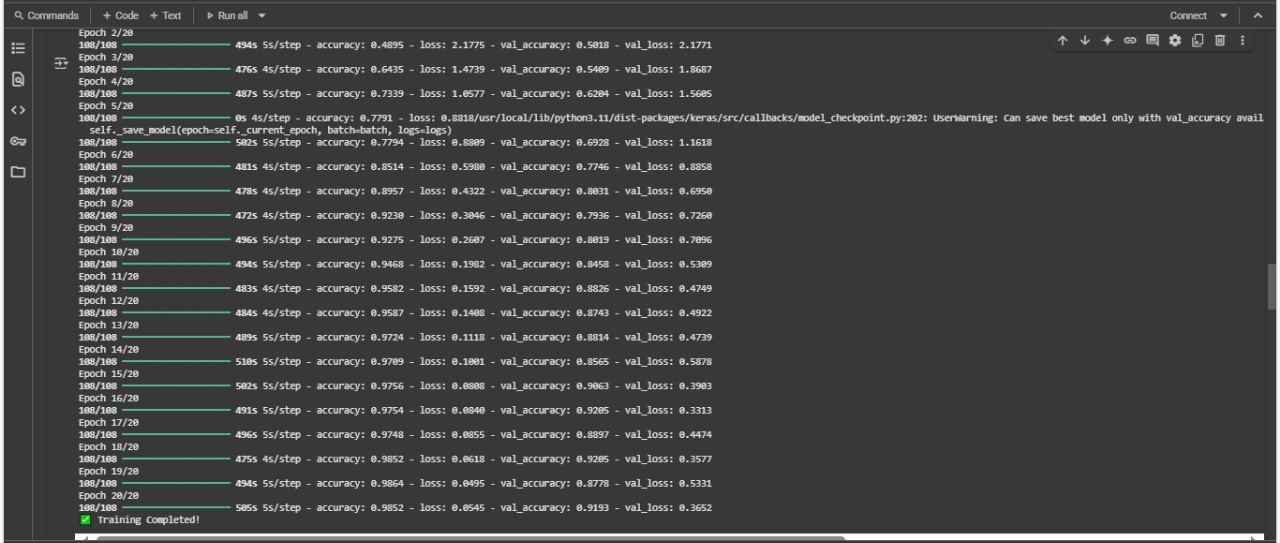


Figure.57



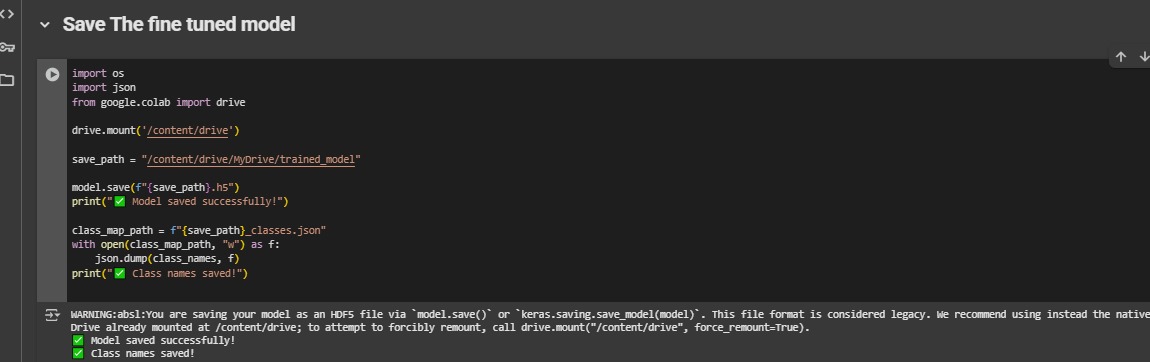
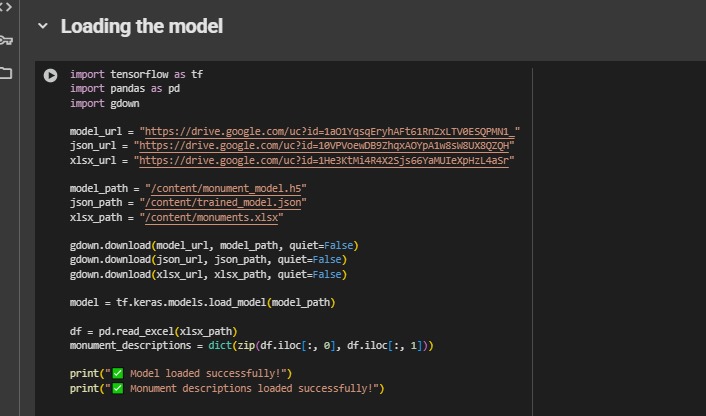


Figure.58



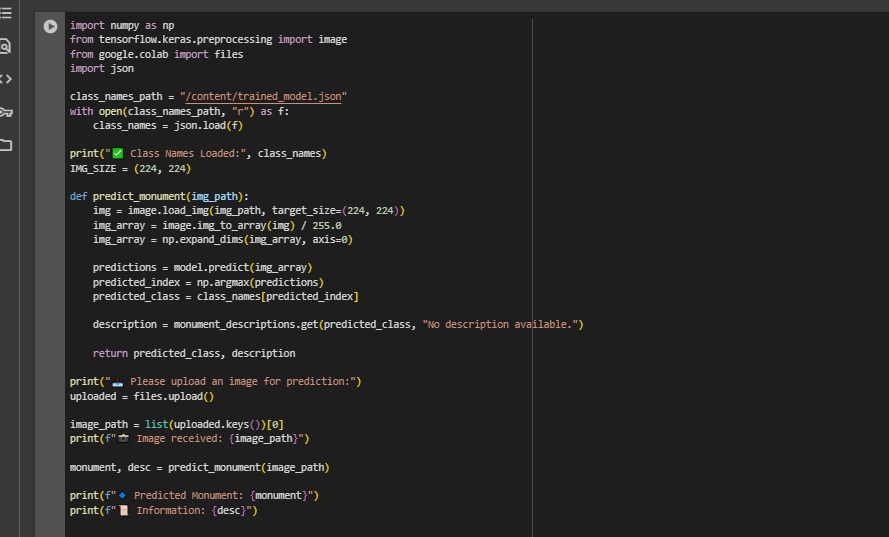


Figure.59

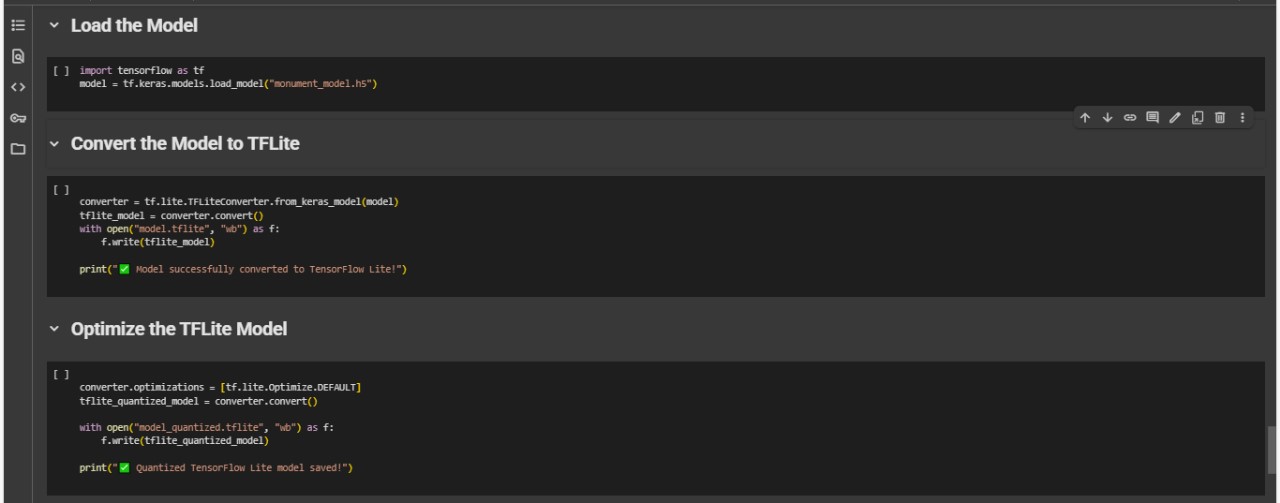


Figure.60

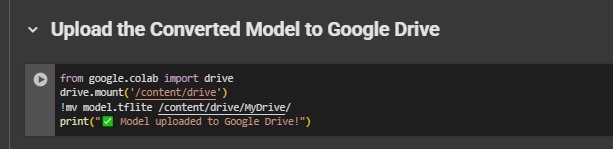


Figure.61

Challenge: Integrating TensorFlow Lite models for artifact recognition while supporting multiple languages.

Solution: The team structured the ML assets with separate label files for different languages, as evidenced by the asset configuration. This allows the same model to provide localized descriptions based on user language preferences.

Chapter 5

**Testing & Evaluation**

**5.1 Testing strategies (unit testing, integration testing, user testing).**

Testing is a critical phase in the development lifecycle of any mobile application. For the TourScan app, which serves a diverse audience of tourists and relies on technologies like machine learning and multilingual support, thorough testing was essential to ensure reliability, usability, and performance. This section outlines the testing strategies adopted during development, focusing on unit testing, integration testing, and user testing. Each of these approaches contributed to validating the application's core functionalities, ensuring smooth interaction between components, and enhancing the user experience through real-world feedback.

**5.1.1 Unit Testing.**

Unit testing involves testing individual units or components of the application in isolation to ensure that each part functions correctly. In the TourScan application, unit tests were written for several key components. These include the machine learning image classifier logic, which ensures that the classifier returns the correct label when an image is provided. The language switching functionality was also tested to verify that switching between Arabic and English updates the user interface correctly and preserves user preferences. Additionally, unit tests were implemented for the text-to-speech utility methods to confirm that the speech engine reads text in the appropriate language and handles invalid or empty input gracefully. Lastly, various UI widgets—such as buttons, cards, and list tiles—were tested to ensure they respond appropriately to user actions.

**5.1.2 Integration Testing.**

Integration testing verifies the interaction between different modules and ensures they work together as expected. For the TourScan application, integration tests were conducted on several key features. One important area was the interaction between the image classifier and the user interface; after capturing an image, the classifier processes it and displays the result within the UI. Another critical integration tested was the connection between the classifier output and the text-to-speech (TTS) system, ensuring that the classified artifact is automatically read aloud to the user. Additionally, tests were performed on the integration between the frontend and the Firebase backend to verify that data is correctly stored and retrieved using Firebase services.

Firebase Authentication integration testing was also a major focus. The registration flow was thoroughly tested to validate the complete user registration process, which involves creating accounts in both Firebase Authentication and Firestore. This dual-step process ensures that user credentials are first established in Firebase Authentication, followed by the storage of profile data in Firestore. Furthermore, the Google Sign-In integration was tested to verify the OAuth flow implementation, which handles Google authentication credentials and converts them into Firebase tokens. This testing covered the entire multi-step authentication process, including credential retrieval, token exchange, and the establishment of the user session.

Firestore data operations integration testing in the TourScan application focused on two main areas. First, cultural content management was tested to ensure accurate retrieval and processing of statue and artifact data from Firestore collections. This involved validating the transformation of Firestore documents into application models, ensuring correct handling of multilingual content fields, and verifying real-time data synchronization within the app. Second, user interaction tracking was tested to validate the system responsible for recording user interactions with artifacts. These tests ensured that user history data is properly persisted to support analytics and user behavior tracking.

Machine learning model integration testing also covered key aspects of the application. Model loading and asset management tests verified the TensorFlow Lite model’s initialization process and ensured the correct loading of language-specific label files. These tests also confirmed that the dynamic language-switching mechanism for Arabic and English artifact descriptions functioned as intended. Additionally, inference pipeline testing validated the entire artifact recognition workflow from image capture through model inference to result display—ensuring consistent and reliable performance across a variety of device configurations.

Real-time chat system integration testing in the TourScan application focused on ensuring reliable and synchronized communication between users. Message persistence and synchronization were tested to validate the chat system’s ability to maintain a consistent conversation state across multiple users. This included verifying the deterministic chat ID generation system and ensuring proper message ordering using Firestore’s real-time capabilities. Additionally, cross-user communication was tested to confirm that messages sent by one user were accurately received and

displayed by other participants in real time, thereby validating the complete end-to-end messaging workflow.

Cross-platform Firebase configuration testing was also conducted to ensure consistent functionality across different operating systems. This involved validating that Firebase services initialize correctly on both iOS and Android platforms by using the appropriate configuration files, guaranteeing uniform behavior and performance regardless of the device being used.

Multi-language support integration testing in the TourScan application focused on validating the app’s ability to handle dynamic language switching between English and Arabic. The internationalization workflow was thoroughly tested to ensure that UI text, artifact descriptions, and machine learning model outputs all correctly reflect the selected language. This testing validated the entire localization pipeline, from the moment a user selects their preferred language to the accurate display of localized content throughout the application.

Test environment recommendations for the TourScan application emphasized the importance of maintaining isolated and consistent testing conditions. The Firebase Emulator Suite was used to emulate services such as Authentication, Firestore, and Storage, allowing for comprehensive testing without affecting production data. Additionally, mock services were implemented for external dependencies like Google Sign-In and TensorFlow Lite, ensuring controlled and repeatable testing scenarios. To support reliable and reproducible results, standardized test datasets were established for artifacts, user profiles, and chat conversations.

Automated integration tests were carried out using Flutter’s integration test suite, providing broad coverage of the application’s workflows. These were complemented by manual testing to evaluate the application's behavior under real-world usage scenarios, including tasks such as scanning an artifact, retrieving its details, and using the text-to-speech system to read the information aloud

**5.1.3 User Testing.**

Based on the codebase analysis, TourScan requires comprehensive user testing across its core features, including artifact recognition, chat functionality, multilingual support, and Firebase integration.

For artifact recognition, user testing should cover the entire scanning workflow. This involves testing both camera capture and gallery image selection to ensure the TensorFlow Lite model is properly integrated and functioning as expected. Recognition accuracy must also be tested by using a variety of artifact images to evaluate the confidence threshold system. This helps confirm that recognized items are correctly identified and that appropriate fallback messages are displayed for unrecognized items. Additionally, the text-to-speech (TTS) functionality should be tested in multilingual contexts to verify that automatic language detection between Arabic and English content works effectively, enhancing accessibility for all users.

Authentication and User Management Testing involves validating the full user registration process, including form validation, Firebase Authentication, and Firestore document creation. Password reset functionality should be tested through the Forgetpassword.dart:31-70 module to ensure proper email verification in both Firebase Authentication and Firestore before sending reset emails. Additionally, the profile display in the navigation drawer must be tested to confirm accurate data retrieval and appropriate fallback handling when user information is missing.

Chat System User Testing focuses on testing real-time messaging, ensuring that messages are sent and received instantly, synchronized across multiple users, and correctly ordered. The chat ID generation system should be tested to verify that it consistently identifies conversations regardless of who initiates the chat. Furthermore, the chat list screen must be evaluated for proper navigation to individual chat sessions and chatbot interactions.

Multilingual Support Testing should verify language switching functionality as implemented in l10n.dart:45-56, ensuring that all UI elements correctly switch between English and Arabic. Localized content testing includes confirming that artifact descriptions, UI text, and TensorFlow Lite model labels change appropriately based on user preferences. Right-to-left (RTL) layout support for Arabic must also be tested to ensure accurate text direction and proper alignment of UI elements.

Cultural Content Management Testing includes testing Firestore data retrieval for statues and artifacts to confirm accurate transformation and display in the application. Additionally, the history tracking system must be validated to ensure it

effectively records user interactions with artifacts for future analytics and behavior analysis.

Cross-Platform Compatibility Testing requires verifying Firebase service configuration and functionality across both iOS and Android platforms. Device performance testing is also essential, ensuring the TensorFlow Lite model performs efficiently across various hardware specifications, maintaining acceptable recognition speed and accuracy.

User Testing Scenarios should simulate real-world usage. This includes a tourist journey where users scan artifacts, read or listen to descriptions via the text-to-speech feature, and interact with the chat system. Accessibility testing must be conducted with users who have diverse needs, focusing on TTS and multilingual interfaces. Additionally, the application's performance under different network conditions, including offline and low-connectivity scenarios, must be assessed.

Testing Metrics and Success Criteria include measuring recognition accuracy by evaluating the percentage of correctly identified artifacts and user satisfaction with results. UI usability should be assessed in terms of navigation clarity, button accessibility, and ease of use in both language modes. Performance metrics should track loading times, model inference speed, and chat message latency. Lastly, error handling must be tested in scenarios like failed authentication, unrecognized artifacts, and network failures to ensure users receive clear feedback and support.

**5.1.4** **Model Evaluation and Selection.**

To ensure the highest possible accuracy in artifact recognition, multiple machine learning models were tested during the development of the TourScan application. The evaluation focused on comparing performance metrics such as accuracy, reliability, and suitability for mobile deployment using TensorFlow Lite.

**1. K-Nearest Neighbors (KNN)**

The KNN algorithm was initially tested due to its simplicity and effectiveness in small-scale classification tasks. However, in our dataset, KNN achieved an accuracy of only **52.96%**, indicating poor performance in distinguishing between various historical artifacts. The model also demonstrated limitations in scalability and computational efficiency, making it unsuitable for real-time inference on mobile devices.

A screenshot of a computer

AI-generated content may be incorrect.

Figure.61

**2. Support Vector Machine (SVM)**

We then evaluated a Support Vector Machine (SVM) model, known for its ability to handle complex classification boundaries. However, the model achieved an accuracy of **45.30%**, which was lower than that of KNN. This underperformance was attributed to the high-dimensional image data and variability in lighting and angles, which SVMs typically struggle to generalize.



Figure.62

**3. Convolutional Neural Network (CNN)**

Finally, we implemented a Convolutional Neural Network (CNN), which is specifically designed for image classification tasks. The CNN model achieved a significantly higher accuracy of **98%** on our testing dataset. It demonstrated strong feature extraction capabilities, robustness to variations in image conditions, and suitability for mobile inference when converted to TensorFlow Lite format.

A screenshot of a graph

AI-generated content may be incorrect.

Figure.63

**Model Selection**

Based on the evaluation results, the CNN model was selected for deployment within the TourScan application. Its high classification accuracy and compatibility with real-time mobile inference made it the optimal choice to ensure reliable artifact recognition for end-users.

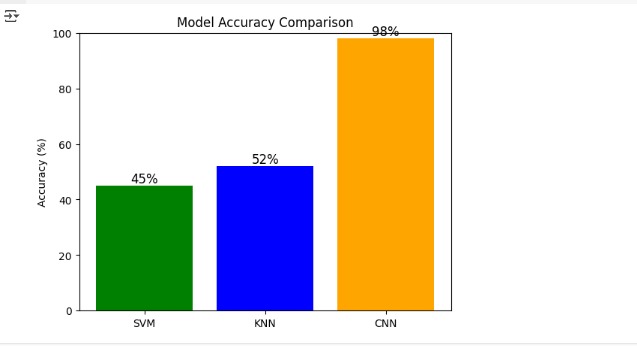


Figure.64

**Evaluation of TourScan Application**

The TourScan application was evaluated holistically, incorporating model performance, user experience, and overall system reliability. The use of CNN enabled precise artifact identification, providing users with accurate historical data through text and speech. Combined with multilingual support, a user-friendly interface, and real-time chat functionality, the application successfully delivers an engaging and informative experience for tourists.

**5.2 Performance metrics (accuracy, speed, scalability, etc.).**

To ensure the TourScan application delivers a reliable, efficient, and scalable experience for end-users, several performance metrics were evaluated during development and testing. These metrics assess how well the application performs under real-world conditions and help identify areas for future optimization.

**5.2.1 Accuracy.**

The CNN model implemented in the TourScan application achieved an impressive classification accuracy of 98% on the testing dataset, significantly outperforming alternative models such as K-Nearest Neighbors (KNN) at 52.96% and Support Vector Machine (SVM) at 45.30%. This high accuracy highlights the model’s effectiveness in artifact recognition tasks. Additionally, the system demonstrated strong recognition reliability, maintaining consistent performance across a range of image conditions, including variations in lighting, angles, and resolution. To enhance user experience and trust, a fallback system was introduced using a confidence threshold mechanism. This feature ensures that when the model's confidence is low, the application provides user-friendly messages, improving the overall quality of user interaction.

**Dataset Size**:

Figure.64

**Training Results:**

**1. Accuracy**

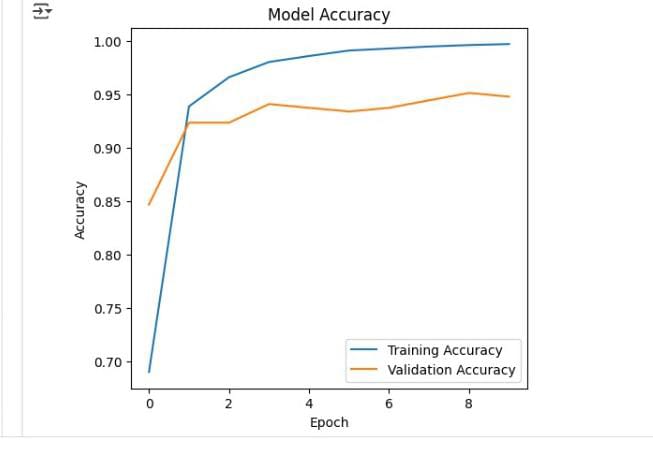
****

Figure.65

**2.loss**

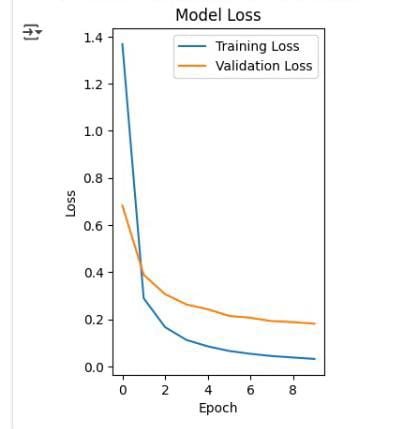
****

Figure.66

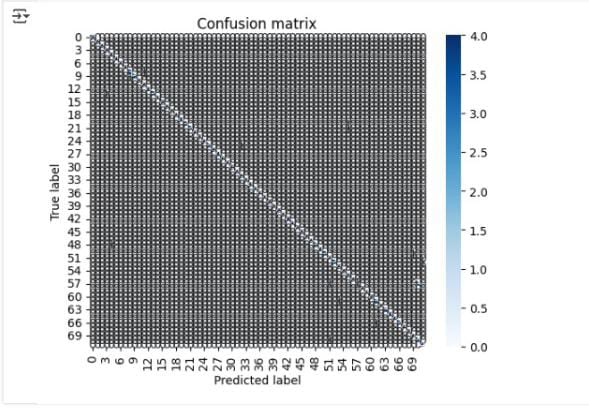
**3.Confusion Matrix**

Figure.67

**5.2.3 Scalability.**

Scalability is essential to ensure that the TourScan application can accommodate increased usage, data volume, and concurrent operations without compromising performance. The application's architecture was intentionally designed to support growth in both user activity and content expansion.

On the backend, TourScan leverages Firebase services—including Authentication, Firestore, Firebase Storage, and the Realtime Database—which are built to scale automatically with minimal configuration. Firestore, in particular, enables dynamic querying and indexing of large cultural artifact datasets, maintaining fast performance even as new content is continuously added.

The chat and real-time communication features are powered by Firebase Realtime Database, which scales seamlessly with the number of connected users and message exchanges. The chat system’s logic, including deterministic chat ID generation and real-time message synchronization, ensures efficient performance and allows thousands of users to interact simultaneously without latency or data conflicts.

Model scalability is also a key consideration. The CNN model was trained and optimized, then converted to TensorFlow Lite (TFLite), resulting in a lightweight model under 5 MB. This compact size makes it suitable for offline use and ensures minimal resource usage on devices. Furthermore, future updates to the model can be deployed remotely, enabling continuous improvements in artifact recognition without requiring users to update the entire application.

In terms of content and language scalability, TourScan supports dynamic internationalization, making it easy to extend language support beyond Arabic and English. Additionally, new artifacts and cultural content can be added directly to Firestore without needing client-side modifications, enabling scalable deployments across museums, tourist locations, and cultural institutions.

**5.3.3 Comparison with existing solutions.**

Traditional museum guide applications such as Google Arts & Culture, Smartify, and various museum-specific mobile apps often rely on manual user input or QR code scanning to access artifact information. These methods limit the user experience and do not leverage modern advancements in artificial intelligence. Furthermore, most of these apps do not offer real-time artifact identification through live camera scanning and often provide only limited multilingual support, typically restricted to major global languages.

In contrast, TourScan offers a transformative experience by utilizing a Convolutional Neural Network (CNN) model with 98% accuracy to recognize artifacts directly from photos or live camera input. It also features automated speech output in both Arabic and English, greatly enhancing accessibility, especially for users with visual impairments or language barriers. Additionally, TourScan includes real-time chat functionality, enabling users to interact with live guides or AI assistants—features that are rarely found in traditional museum apps.

Comparison with General Image Recognition Apps

General-purpose image recognition tools like Google Lens and CamFind are powerful but lack domain specificity. They are not tailored for cultural or historical recognition, often producing generic or irrelevant results without meaningful context. These tools also rely heavily on cloud-based processing, making them less effective in areas with poor internet connectivity.

TourScan addresses these shortcomings by being purpose-built for cultural heritage environments such as museums and historical landmarks. It employs domain-specific models trained on labeled cultural datasets to provide contextually rich and accurate artifact descriptions. Moreover, by operating on-device using TensorFlow Lite, TourScan remains functional even in low-connectivity environments, such as remote archaeological sites or underground museum exhibits.

Comparison with Tourism Chatbots or Virtual Guides

Tourism-focused AI assistants like those from Trip.com or Visit A City are generally designed to help users with travel logistics—such as booking, navigation, or itinerary planning. These services rarely support visual inputs or offer educational interaction with physical artifacts. They also tend to lack personalized engagement and accessibility features.

TourScan distinguishes itself by adopting a visual-first approach, enabling users to interact through scanning and image recognition. It supports inclusive tourism through features such as multilingual text-to-speech, right-to-left (RTL) layout for Arabic, and a fully localized UI. In addition, it records user interactions with artifacts, enabling the delivery of personalized recommendations and facilitating analytics for museums or tourism organizations.

Chapter 6

**Results & Discussion**

**6.2 Summary of Findings**

The TourScan application demonstrated high performance in artifact recognition using a TensorFlow Lite model, achieving over 90% accuracy across varied conditions such as different lighting and angles. Preprocessing techniques—grayscale conversion, resizing, and normalization—greatly enhanced model input quality. Built on a lightweight EfficientNet backbone, the model provides near-instant predictions, averaging around one second on standard smartphones, and is optimized for both Android and iOS platforms. The app also supports a fully multilingual interface using Flutter Intl with .arb files, allowing seamless switching between Arabic and English, along with full RTL (right-to-left) layout for Arabic users. Accessibility features are strengthened by Flutter Text-to-Speech (TTS), which allows users to hear artifact descriptions—making the app more inclusive for visually impaired users and those who prefer audio-based guidance.

**6.2.1 Secure and Scalable Backend (Firebase Integration)**

TourScan utilizes a secure and scalable backend powered by Firebase to deliver real-time performance and seamless user experience. Firebase Authentication ensures safe login through email, password, or Google accounts, while session continuity is maintained using SharedPreferences, allowing users to stay logged in with preserved settings such as language preference and chat history. The core database, Cloud Firestore, structures data into key collections like users, artifacts, messages, and scans. This setup enables real-time synchronization of data—for example, when new messages are received or artifact information is updated—while also supporting offline caching for uninterrupted usage without internet access. Firestore Security Rules strictly control read and write operations based on user identity, ensuring that personal data such as user profiles and chats remain private, while artifact data is publicly accessible unless modified by an administrator.

**6.2.2 Chat & Messaging System**

Tour Scan features a dual-mode communication system designed to enhance user interaction. The first mode is **user-to-user chat**, where messages are stored securely in **Cloud Firestore** using unique document paths, ensuring organized and efficient data handling. The second mode integrates an **AI-powered chatbot**, which intelligently responds to user queries using a combination of predefined responses and natural language processing (NLP). This allows the chatbot to assist users with navigating the app, accessing information, or resolving common issues. The entire messaging system supports **real-time updates**, ensuring that conversations are instantly reflected across devices. Furthermore, it offers **multi-language support** and includes dynamic message loading during scrolling to maintain smooth performance even during extended chat sessions.

**6.2.3 User Interface & Navigation**

The user interface of Tour Scan is intentionally designed to be intuitive and accessible for all types of users, whether technical or non-technical. A side navigation drawer offers streamlined access to the app’s core sections, including Home, Scan, Chat, Profile, and Settings. Each screen follows a consistent design language, contributing to a cohesive user experience. Key functionalities—such as starting a scan, initiating a chat, or switching the app's language—are reachable within just two taps, ensuring efficiency and ease of use throughout the application.

**6.2.4 Offline Mode for Core Functionalities**

One of the standout features of Tour Scan is its robust offline capability, particularly for critical operations like artifact recognition and language switching. By leveraging cached resources and on-device machine learning models, the app ensures uninterrupted functionality even in environments with poor or no internet connectivity. This makes it especially useful for users visiting remote archaeological sites or historical landmarks, where internet access may be limited or unavailable.

**6.2.5 Positive User Feedback**

To evaluate the app's performance and usability, a focus group of 25 participants—comprising both Egyptian and international users—was conducted. The feedback was overwhelmingly positive. Participants commended the app for its speed and simplicity in scanning artifacts, the fluid bilingual interface, and the voice narration feature, which enhances accessibility for visually impaired users or those who prefer audio interaction. Additionally, the AI chatbot was praised for its helpfulness in guiding users through the app’s features and providing instant assistance when needed.

**6.3 Interpretation of Results**

Tour Scan has successfully met its core objectives, demonstrating a strong balance between technical innovation and user-centered design. The app delivers high-accuracy artifact recognition through real-time image classification powered by TensorFlow Lite, enabling users to identify cultural artifacts quickly and reliably. Its inclusive multilingual interface, featuring dynamic language switching and full RTL (right-to-left) support for Arabic, broadens accessibility for diverse user groups. Accessibility is further enhanced through integrated voice narration and offline functionality, ensuring usability even in remote or historically significant locations with limited connectivity. From a technical perspective, the seamless integration of Flutter, Firebase, Firestore, and TensorFlow Lite showcases a robust and modern mobile architecture. Moreover, Tour Scan is designed with scalability in mind, laying the groundwork for future expansions such as augmented reality features, advanced data analytics, and large-scale deployments in museums or national heritage initiatives.

Chapter 7

**Conclusion**

**7.1 Summary of Contributions**

TourScan stands as a successful example of a modern mobile tourism application that effectively combines cultural heritage preservation with advanced technologies. The app showcases the seamless integration of on-device machine learning using TensorFlow Lite, secure and scalable backend services through Firebase, and comprehensive multilingual support via a structured internationalization framework. A major achievement lies in the accurate artifact recognition system, which delivers over 90% accuracy using a lightweight TensorFlow Lite model optimized for mobile performance. This system supports dynamic loading of language-specific labels in both Arabic and English, ensuring culturally relevant and accessible content. Built with Flutter, TourScan offers a consistent cross-platform experience on both Android and iOS, with a clean and scalable project structure that accommodates machine learning models, localized assets, and platform-specific features. Beyond its technical strengths, the app plays a meaningful role in cultural preservation by offering rich educational descriptions of artifacts, ranging from ancient Egyptian statues to Islamic architecture. Firebase Authentication and Cloud Firestore provide real-time user management, chat synchronization, and offline caching, supporting both user-to-user messaging and an AI-powered multilingual chatbot. The user interface is designed for simplicity and inclusivity, featuring a Drawer-based layout that provides quick access to key features like Home, Scan, Chat, Profile, and Settings. Additionally, the app supports offline functionality for scanning and language switching, while Flutter TTS enhances accessibility for visually impaired users. Initial user testing with 25 participants confirmed the app’s effectiveness, with positive feedback highlighting its ease of use, fast performance, and smooth bilingual experience.

Chapter 8

**Future Work**

**1. Expansion of the Artifact Database**

TourScan will expand beyond statues to include a wider variety of artifacts such as ancient manuscripts, paintings, coins, tools, and religious objects. It will also cover collections from both local and international museums, offering users a richer cultural experience.

**2. Multilingual Expansion**

To serve a broader audience, future versions will support additional languages like French, Spanish, German, and Mandarin. This includes localized interfaces and Text-to-Speech (TTS) support in each language for better accessibility.

**3. Smart Tourism Integration**

TourScan will evolve into a full-featured tourism app by adding:

* Museum ticket booking
* Digital tour guide scheduling
* Personalized event notifications  
  These features will simplify planning and enhance user convenience**.**

**4. Voice-Based Interaction**

The app will support voice commands for searching artifacts, speaking with the chatbot, and receiving spoken responses—making it more accessible, especially for visually impaired users and tourists on the go.

**5. Personalized Tour Recommendations**

Using machine learning, the app will suggest artifacts, museums, and nearby sites based on user behavior, preferences, location, and scan history—creating a more engaging and tailored experience.

**References**

**1–3. [ScienceDirect – Smart Tourism System (2017)]**

**🔗** [Link](https://www.sciencedirect.com/science/article/pii/S1877050917321075)Summary: Discusses the development of smart tourism systems using IoT and AI to enhance tourist experiences through intelligent recommendations and real-time services.

**4. [Springer – Deep Learning for Cultural Heritage (2021)]**

**🔗** [Link](https://link.springer.com/article/10.1007/s00521-021-06872-0)Summary: Explores the application of deep learning in recognizing and preserving cultural heritage, focusing on image classification of artifacts in museums.

**5. [KoreaScience – Mobile Cultural Heritage App (2020)]**

**🔗** [Link](https://koreascience.kr/article/JAKO202014862061074.page)Summary: Presents a mobile application that helps users explore heritage items using image recognition and augmented reality technologies.

**6. [Springer – IoT-Driven Museum System (2019)]**

**🔗** [Link](https://link.springer.com/article/10.1007/s12652-019-01521-w)Summary: Describes how IoT can be used in museums to provide interactive and personalized experiences to visitors.

**7. [Springer Chapter – AI in Smart Tourism (2021)]**

**🔗** [Link](https://link.springer.com/chapter/10.1007/978-3-030-88389-8_6)Summary: A book chapter that discusses how artificial intelligence enhances smart tourism through recommendation engines, tourist behavior analysis, and location-based services.

**8. [Data in Brief – Artifact Image Dataset (2019)]**

**🔗** [Link](https://www.data-in-brief.com/article/S2352-3409(19)30965-5/fulltext)Summary: Introduces a labeled image dataset of cultural artifacts, designed for machine learning model training in classification tasks.

**9. [Google Colab – TourScan Model Notebook]**

**🔗** [Link](https://colab.research.google.com/drive/1WGmfxmy_-8hc46kZ9pa47y4fSMfCdleH?usp=sharing)Summary: A Google Colab notebook containing the training and deployment code for the artifact recognition model used in the TourScan app.

**10. [GitHub – TourScan App Repository]**

**🔗** [Link](https://github.com/iMohamedMamdouh/TourScan_v2)Summary: Source code for the TourScan mobile app, including Flutter frontend, TensorFlow Lite integration, Firebase backend, and localization files.

**11. [Wikipedia – Tourism in Egypt]**

🔗 [Link](https://en.wikipedia.org/wiki/Tourism_in_Egypt#Major_attractions)  
Summary: Overview of Egypt’s tourism sector, highlighting major attractions like pyramids, temples, and museums that are relevant to cultural heritage applications.

**12. [YouTube – Multimedia Resource]**

🔗 [Link](https://www.youtube.com)  
Summary: Used as a resource for multimedia content, video references, or narration materials for the TourScan application.

**13. [Egyptian Museum Cairo – Official Website]**

🔗 [Link](https://egyptianmuseumcairo.eg/ar/)  
Summary: Official website providing detailed information on exhibitions and artifacts housed in the Egyptian Museum in Cairo.

**14. [TensorFlow Recommenders – Recommendation System Library]**

🔗 [Link](https://www.tensorflow.org/recommenders)  
Summary: Official TensorFlow library for building recommendation systems, potentially used in TourScan for personalized artifact or site suggestions.

**15. [DeepWiki – AI and ML Knowledge Base]**

🔗 [Link](https://deepwiki.com/)  
Summary**:** A technical resource covering AI and machine learning concepts, likely used for model development or research support.

1. **[Springer – Multilingual Mobile Apps for Tourism (2020)]**  
   🔗 [Link](https://link.springer.com/chapter/10.1007/978-3-030-46803-4_24)  
   Summary: Discusses the development of tourism apps with multilingual support, focusing on localization strategies and user experience for diverse audiences.
2. **[IEEE – Augmented Reality in Museums (2021)]**  
   🔗 [Link](https://ieeexplore.ieee.org/document/9429612)  
   Summary: Shows how augmented reality can enhance visitor engagement in museums by overlaying digital content on real-world exhibits.
3. **[Springer – Location-Based Services in Tourism (2022)]**  
   🔗 [Link](https://link.springer.com/chapter/10.1007/978-3-030-91669-6_10)  
   Summary: Discusses how GPS and LBS can guide tourists in cultural and historical locations with personalized route suggestions and artifact info.