**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

**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**

* **Abstract** ..........................................................................................i
* **List of Figures** .................................................................................ii

**Chapter 1: Introduction**

1.1 Introduction ....................................................................................... 10  
1.2 Background and Motivation for the Project ...........................................11  
1.3 Importance of the Problem Being Addressed ........................................12  
1.4 Problem Statement .............................................................................13  
1.5 Objectives ..........................................................................................14  
1.6 Brief Overview of the Proposed Solution ...............................................15

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

2.1 Deep Learning and IoT for Tourist Recommendation in Smart Cities .......18  
2.2 Sentiment-Based Tourism Recommendation System ...........................19  
2.3 TOURGURU: Guide Mobile Application for Tourists ...............................20  
2.4 Smart Tourism using IoT, AI, and 5G ..................................................... 21  
2.5 Context-Aware Real-Time Recommendation Systems ...........................22

**Chapter 3: Proposed System**

3.1 Problem-Solving Approach ..................................................................24  
3.2 System Architecture (UML, Flowcharts, ER Diagram) ............................25  
3.3 Algorithms and Frameworks Used ........................................................30

**Chapter 4: Implementation**

4.1 Technologies, Tools, and Programming Languages ............................... 32  
4.2 Key Components / Modules  
 4.2.1 Frontend Components ................................................................. 34  
 4.2.2 Backend Components ................................................................. 55  
 4.2.3 AI Implementation ....................................................................... 64

4.3 Challenges faced and how they were resolved ……………………………….68

**Chapter 5: Testing & Evaluation**

5.1 Testing Strategies  
 5.1.1 Unit Testing .................................................................................72  
 5.1.2 Integration Testing .......................................................................73  
 5.1.3 User Testing ................................................................................76  
 5.1.4 Model Evaluation and Selection ...................................................79  
5.2 Performance Metrics  
 5.2.1 Accuracy .................................................................................... 83  
 5.2.2 Speed ........................................................................................ 86  
 5.2.3 Scalability .................................................................................. 88  
5.3 Comparison with Existing Solutions ....................................................89

**Chapter 6: Results & Discussion**

6.1 Introduction ..................................................................................... 92  
6.2 Summary of Findings .........................................................................93  
6.3 Interpretation of Results ....................................................................96  
6.4 Limitations of the Proposed Solution ..................................................96

**Chapter 7: Conclusion**

7.1 Summary of Contributions .................................................................98

**Chapter 8: Future Work**

8.1 Future Enhancements and Features ..................................................101

* **References** ..................................................................................104

**List of Figures:**

**System Architecture Diagrams**

Figure 1: UML Class Diagram ........................................................... 24

Figure 2: Component Diagram ......................................................... 24

Figure 3: Sequence Diagram ............................................................ 25

Figure 4: Use Case Diagram ............................................................. 25

Figure 5: Activity Diagram ................................................................ 26

Figure 6: Flowchart ......................................................................... 27

Figure 7: ER Diagram ....................................................................... 28

**User Interface (UI) Screens**

Figure 8: Splash View ...................................................................... 33

Figure 9: Home View ....................................................................... 34

Figure 10: Additional Home UI ......................................................... 34

Figure 11: Drawer UI ........................................................................ 35

Figure 12: Language Switch & Logout ............................................... 35

Figure 13: Settings Page Overview .................................................... 36

Figure 14: Update Profile Flow .......................................................... 37

Figure 15: Password Update Workflow .............................................. 37

**Chat System**

Figure 16: Chat System Overview ...................................................... 38

Figure 17: Chat List ........................................................................... 38

Figure 18: Direct Chat UI ................................................................... 39

Figure 19: Chatbot Integration ....................................................... 40

**Static Pages**

Figure 20: About Page ................................................................... 41

Figure 21: Artifact View UI ............................................................. 42

**AI Image Recognition System**

Figure 22: Image Recognition Flow ................................................ 43

Figure 23: Processing Pipeline ....................................................... 44

Figure 24: Inference Result View .................................................... 44

Figure 25: Statue Display Screen ................................................... 45

Figure 26: Performance Optimizations ........................................... 46

**Authentication and User Management**

Figure 27: Search Screen UI ............................................................ 47

Figure 28: Login Screen .................................................................. 48

Figure 29: Signup Screen ................................................................ 49

Figure 30: Forget Password UI ......................................................... 50

Figure 31: Email Verification UI ....................................................... 50

Figure 32: New Password UI ............................................................. 51

Figure 33: Password Reset Flow ....................................................... 51

Figure 34: Reset Success Page ......................................................... 52

Figure 35: Navigation Transitions ...................................................... 52

**Localization and Backend Architecture**

Figure 36: Localization Architecture ................................................ 53

Figure 37: Backend Overview ........................................................... 54

Figure 38: Firebase Auth UI ............................................................. 55

Figure 39: Google OAuth Flow ......................................................... 56

Figure 40: Firestore User Structure .................................................. 57

Figure 41: Artifact Collections .......................................................... 58

Figure 42: Firestore Statues Schema ............................................... 59

Figure 43: Artifact Details from Firestore ........................................ 59

Figure 44: Chat Message Structure .................................................. 60

**AI / ML Implementation**

Figure 45: ML Inference Integration ............................................... 61

Figure 46: Preprocessing Pipeline ................................................... 62

Figure 47: Model Deployment Workflow ......................................... 63

**Model Evaluation Results**

Figure 48: KNN Evaluation Results .................................................. 78

Figure 49: SVM Evaluation Results .................................................. 79

Figure 50: CNN Accuracy Graph ..................................................... 80

Figure 51: Final Model Selection ...................................................... 81

Figure 52: Dataset Size Overview ................................................... 82

Figure 53: Accuracy Graph .............................................................. 83

Figure 54: Loss Graph ....................................................................... 84

Figure 55: Confusion Matrix ............................................................. 84

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 experience.

This 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.**

**Enhance Tourist Experience Through Technology:** Develop a mobile application that uses Artificial Intelligence (AI) and image recognition to provide instant information about statues and landmarks, transforming traditional visits into interactive and educational experiences.

**Implement an Image Recognition System:** Utilize a Convolutional Neural Network (CNN) to accurately and quickly identify statues and monuments from images captured by the user's smartphone camera.

**Provide Instant Information:** Display comprehensive information about each statue, including its name, origin, artist, and historical context, immediately after scanning.

**Ensure Easy User Interaction:** Design a simple and intuitive user interface that allows tourists to scan statues and receive information effortlessly.

**Bridge Language Barriers in Cultural Heritage:** Support both English and Arabic languages, with accurate localization and translation, to make information accessible to a diverse audience.

**Democratize Access to Museum Technology:** Make advanced museum technology available through smartphones, removing the need for costly audio guides and providing a complete digital experience for all users.

**Ensure Accessibility and Inclusion:** Include text-to-speech functionality with automatic language detection to accommodate users with various accessibility needs, such as visual or reading impairments.

**Preserve and Share Egyptian Cultural Heritage:** Digitally preserve Egyptian cultural heritage through a comprehensive artifact database that offers accurate and bilingual historical information.

**Provide Interactive Museum Assistance:** Integrate an AI-powered chatbot and user communication features to offer an all-in-one digital museum assistant, extending beyond simple object recognition.

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

**Core Solution Architecture:**

**Mobile-First Approach**: TourScan is built as a cross-platform Flutter application that provides a unified experience across iOS, Android, and other platforms. The main entry point displays the app's mission statement in both English and Arabic.

**AI-Powered Recognition System**: The solution centers around a TensorFlow Lite-based statue recognition system that processes images locally on the device. Users can capture photos or select from gallery, and the system identifies artifacts with a 60% confidence threshold.

**Multilingual Content Delivery**: The app dynamically loads language-specific labels and content supporting both English and Arabic. The internationalization system uses Flutter's built-in localization framework.

**Technical Implementation:**

**Offline-First Architecture**: The core recognition functionality works without internet connectivity by embedding the TensorFlow Lite model and labels locally, while cloud services handle user authentication and social features.

**Firebase Integration**: The solution uses Firebase for user management, data storage, and real-time chat functionality, providing a scalable backend infrastructure.

**Modular Design**: The app follows a component-based architecture with separate modules for scanning, internationalization, chat, and content display, enabling easy maintenance and feature expansion.

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.**

**1. Deep Learning-Based Similarity Detection:** The core recognition system uses TensorFlow Lite for on-device image classification. The system loads a pre-trained model and processes images through a confidence-based similarity detection approach with a 60% threshold.

**2. Mobile-First Accessibility:** The accessibility approach includes comprehensive text-to-speech functionality with automatic language detection. The system detects Arabic text using Unicode character ranges and adjusts speech parameters accordingly, supporting both English and Arabic users.

**3. Cloud-Based Verification Pipeline:** While the core recognition runs locally, the system uses Firebase for cloud-based data management and verification. User authentication is handled through, providing Google Sign-In integration and secure user management.

**4. Secure Authentication and Data Management:** The authentication system implements Firebase Authentication with Google Sign-In capabilities, ensuring secure user credential management and session handling. The system maintains user state and provides secure logout functionality.

**5. Preprocessing for Robustness:** Image preprocessing ensures robust recognition by standardizing input format. Images are resized to 224x224 pixels and normalized to 0-1 range for consistent model input, improving recognition accuracy across different image conditions.

The multilingual label loading system dynamically switches between English and Arabic content based on user locale, ensuring culturally appropriate and robust content delivery.

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

1.UML

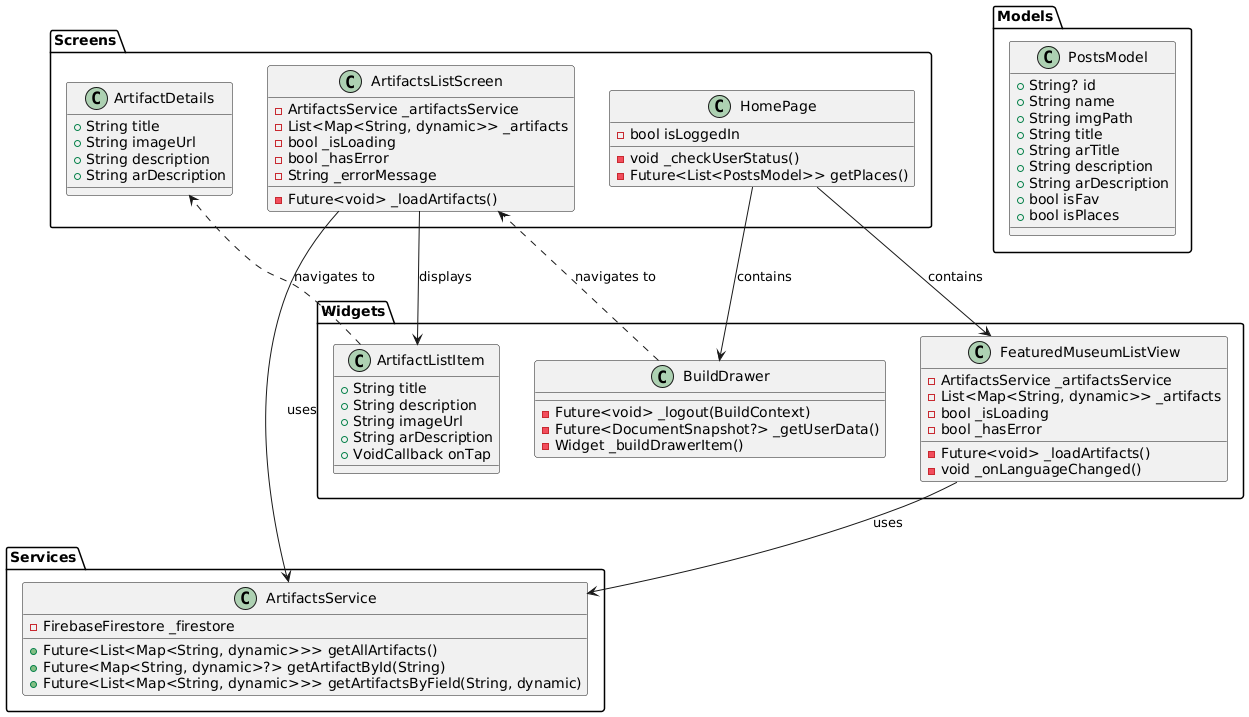
* Class Diagram

Figure.1

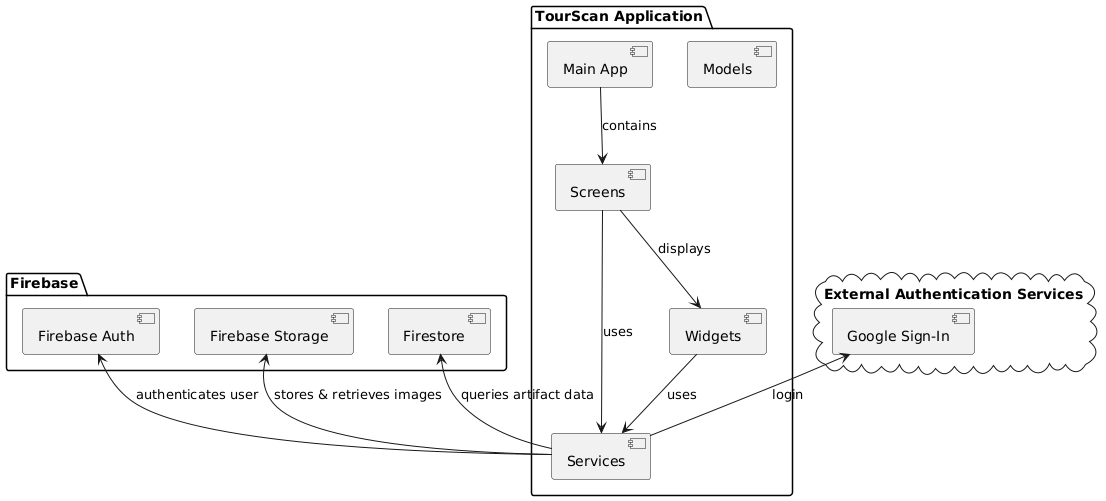
* Component Diagram

Figure.2

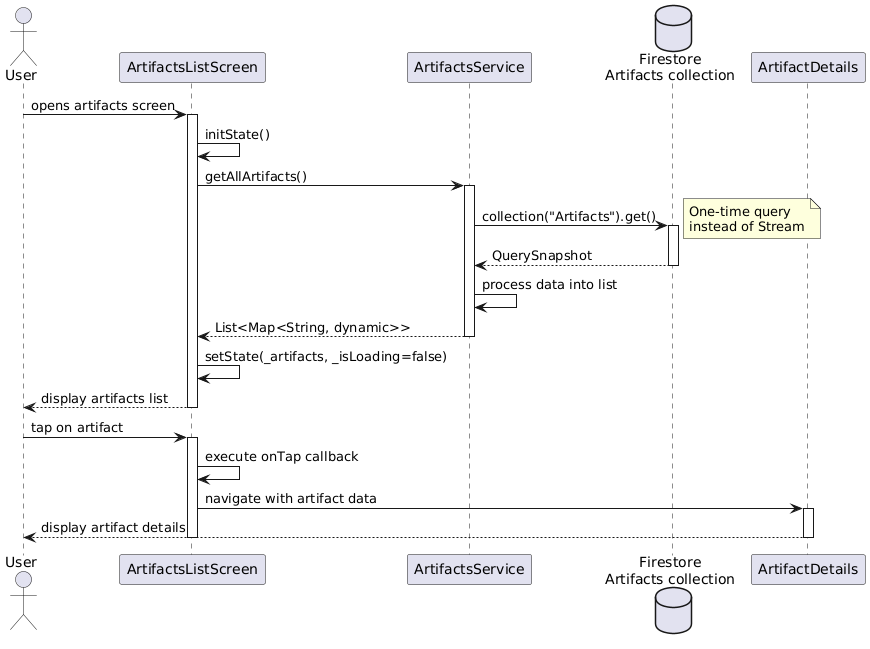
*  Sequence Diagram

Figure.3

* A diagram of a software application

  AI-generated content may be incorrect.UseCase Diagram

Figure.4

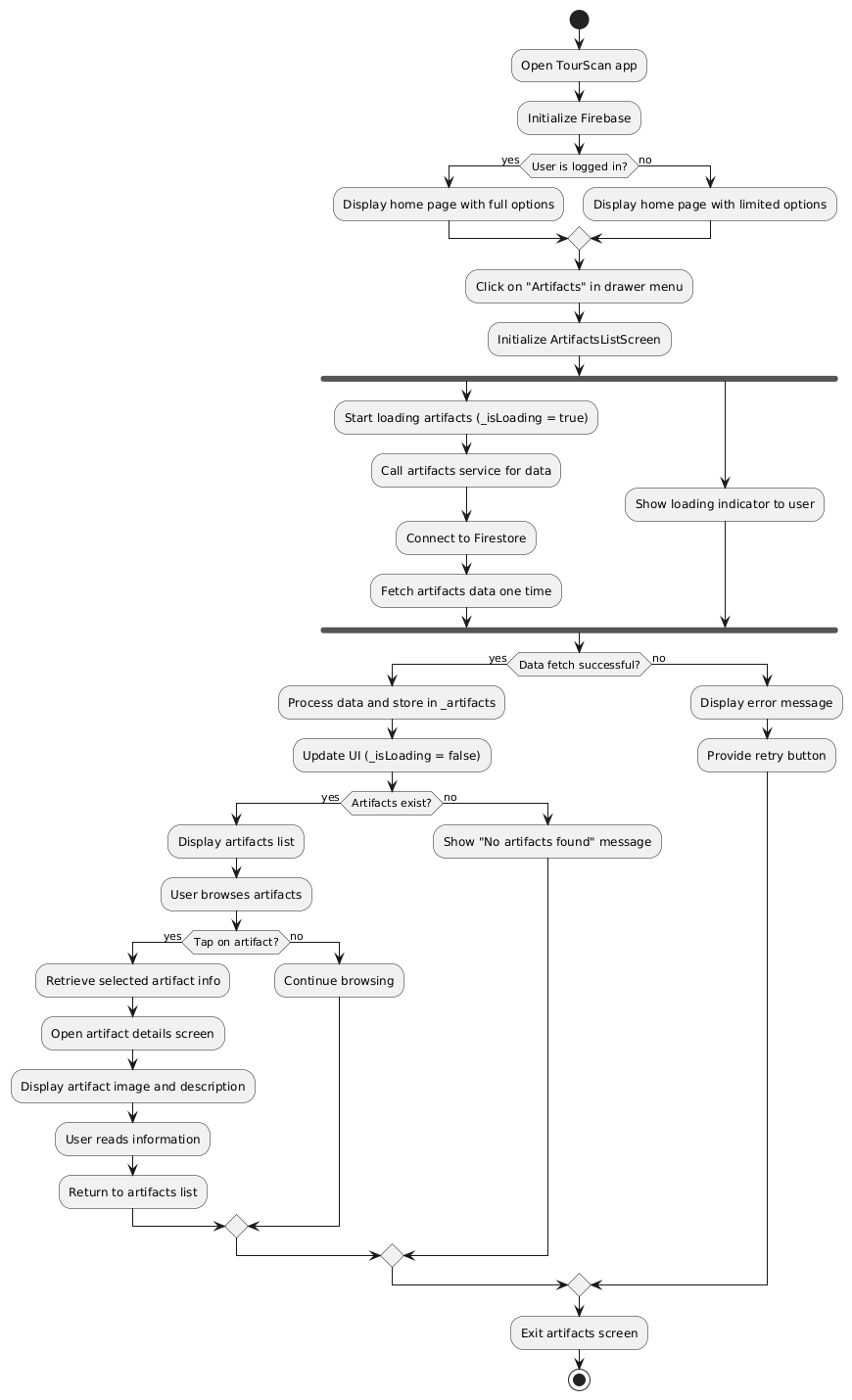
* Activity Diagram

Figure.5

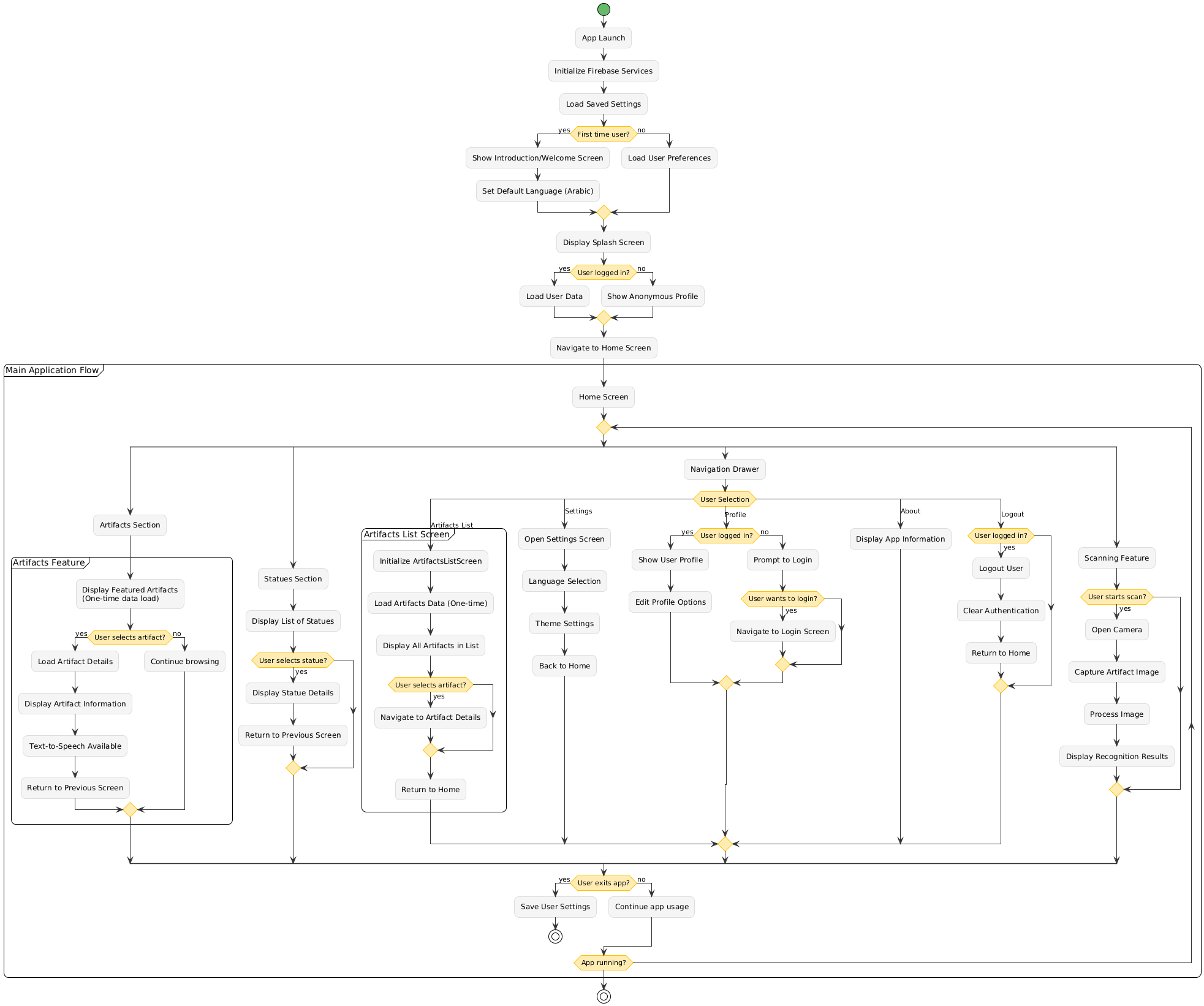
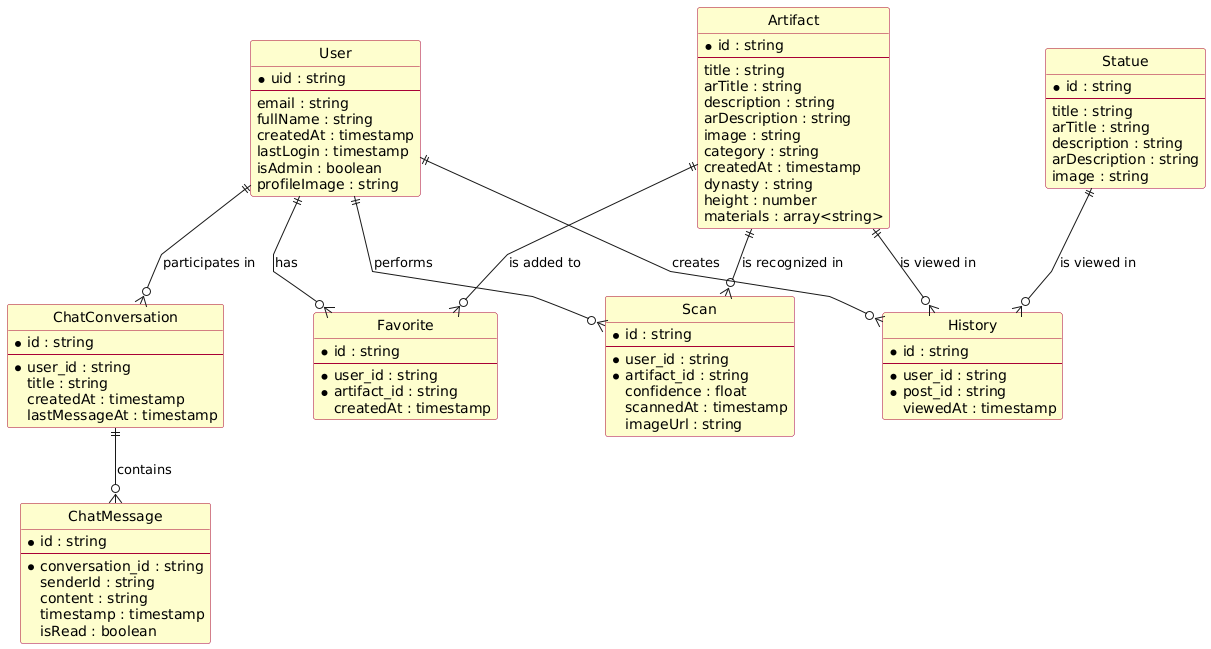
2.Flowchart

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 built as a cross-platform Flutter application, It aims to help tourists learn about history and discover tourist attractions.
* The frontend is developed using **Flutter**, while the backend is powered by **Firebase**.

**1. Machine Learning & Image Recognition**

**TensorFlow Lite** is the core algorithm for statue recognition. The system loads a pre-trained model and processes images through a classification pipeline

**2. Backend & Database**

**Firebase** serves as the primary backend framework:

* **Firebase Authentication** for user management
* **Cloud Firestore** for data storage and real-time synchronization
* **Google Sign-In**

**3. Internationalization**

**Flutter Intl** framework handles multi-language support, dynamically loading English or Arabic labels based on device locale

**4. Text-to-Speech**

**Flutter TTS** provides accessibility features, automatically detecting Arabic or English text for appropriate pronunciation

**5. AI Chatbot**

**GeminiService**, the implementation focuses on providing a conversational interface for museum visitors to ask questions and get AI-powered assistance during their tours.

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

**When You Press "Update Data":** Figure.14

**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

**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 AI service

A diagram of a chat

AI-generated content may be incorrect. Figure.16

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."

* **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 Figure.21

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

1. **Preprocessing**: Resizing, normalization, and formatting for the model input
2. **Model Inference**: Running the processed image through the TensorFlow Lite model
3. **Result Analysis**: Determining if the recognition confidence exceeds the threshold
4. **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

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:

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

Figure.26

1. **Asynchronous operations**: Image processing and model inference run asynchronously to prevent UI freezing
2. **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.27

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:

* **Title**: "Login to your account"  Figure.28
* **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.29
   * 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

* **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.30

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.31

* **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
* Password reset requires user to be authenticated Figure.32
* Password confirmation is required to prevent typos
* Success and error messages provide clear feedback to users

Figure.33

A screenshot of a computer

AI-generated content may be incorrect.

Figure.34

A diagram of a computer

AI-generated content may be incorrect.

Figure.35

* **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.36

* **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.37

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.38

* **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.39

* **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.40

* **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.41

**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.42

A screenshot of a computer

AI-generated content may be incorrect.

Figure.43

**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.44

* **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.45

**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.46

* **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

ImageDataGenerator(

rescale=1./255,

validation\_split=0.2,

rotation\_range=30,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest'

)

**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.

**Final Training Accuracy: ~98.5%**

**Validation Accuracy: ~92.0%**

****

Figure.47

* **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**

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 the following components:  
  
**Machine learning image classifier logic:** Ensures that the classifier returns the correct label when an image is provided.

**Language switching functionality:** Verifies that switching between Arabic and English updates the UI correctly and saves user preferences.

**Text-to-speech utility methods:** Confirms that the speech engine reads text in the correct language and handles invalid or empty input gracefully.

**UI Widgets:** Tests components like buttons, cards, and list tiles to ensure they respond appropriately to user actions.

The Flutter testing framework was used to automate these tests. Mock objects were used to simulate dependencies, allowing for focused and accurate testing. These tests helped detect and resolve bugs during the early development phase.

**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 the following features:

**Image Classifier and UI**: After capturing an image, the classifier runs and displays the result in the interface.

**Classifier Output and TTS:** Ensures that the classified artifact is automatically read out loud using the TTS system.

**Frontend and Firebase Backend:** Verifies that data is correctly stored and retrieved using Firebase services.

**Firebase Authentication Integration Testing:**

* **Registration Flow Testing:** Test the complete user registration process that creates accounts in both Firebase Authentication and Firestore. This involves validating the dual-step process where user credentials are first created in Firebase Auth , followed by profile data storage in Firestore .
* **Google Sign-In Integration:** Verify the OAuth flow implementation that handles Google authentication credentials and converts them to Firebase tokens . This tests the multi-step authentication process including credential retrieval, token exchange, and user session establishment.

**Firestore Data Operations Integration Testing**

* **Cultural Content Management**: Test the retrieval and processing of statue and artifact data from Firestore collections. This includes validating the data transformation from Firestore documents to application models , ensuring proper handling of multilingual content fields, and verifying real-time data synchronization.
* **User Interaction Tracking**: Validate the history tracking system that records user interactions with artifacts. This ensures proper data persistence for analytics and user behavior tracking.

**Machine Learning Model Integration Testing**

* **Model Loading and Asset Management:** Test the TensorFlow Lite model initialization process and verify proper loading of language-specific label files . This includes validating the dynamic language switching mechanism for Arabic and English artifact descriptions.
* **Inference Pipeline Testing:** Verify the complete artifact recognition workflow from image capture through model inference to result display, ensuring consistent performance across different device configurations.

**Real-time Chat System Integration Testing**

* **Message Persistence and Synchronization**: Test the chat system's ability to maintain consistent conversation state across multiple users. This includes validating the deterministic chat ID generation system and ensuring proper message ordering through Firestore's real-time capabilities.
* **Cross-User Communication**: Verify that messages sent by one user are properly received and displayed by other participants in real-time, testing the complete end-to-end messaging workflow.

**Cross-Platform Firebase Configuration Testing**

* **Platform-Specific Service Initialization:** Validate that Firebase services initialize correctly on both iOS and Android platforms using the appropriate configuration files . This ensures consistent behavior across different operating systems.

**Multi-language Support Integration Testing**

* **Internationalization Workflow:** Test the application's ability to switch between English and Arabic languages dynamically, including UI text, artifact descriptions, and machine learning model outputs. This validates the complete localization pipeline from user preference selection to content display.

**Test Environment Recommendations**

* **Firebase Emulator Suite:** Use Firebase emulators for Authentication, Firestore, and Storage to create isolated test environments that don't affect production data.
* **Mock Services:** Implement mock versions of external services like Google Sign-In and TensorFlow Lite for consistent testing conditions.
* **Test Data Management:** Establish standardized test datasets for artifacts, user profiles, and chat conversations to ensure reproducible test results.

Automated integration tests were performed using Flutter’s integration test suite. In addition, manual testing was conducted to validate the behavior of the application under real usage scenarios, such as scanning an artifact, retrieving its details, and reading them.

**5.1.3 User Testing.**

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

**1. Artifact Recognition User Testing**

* **Scanning Workflow Testing**: Test the complete artifact recognition pipeline from the scanning screen. Users should test both camera capture and gallery image selection to verify the TensorFlow Lite model integration works correctly.
* **Recognition Accuracy Testing**: Validate the confidence threshold system by testing with various artifact images to ensure proper recognition and appropriate fallback messages for unrecognized items.
* **Text-to-Speech Functionality**: Test the multilingual TTS feature to verify automatic language detection between Arabic and English content works properly for accessibility.

**2. Authentication and User Management Testing**

* **Registration Flow Testing:** Test the complete user registration process including form validation, Firebase Authentication integration, and Firestore user document creation.
* **Password Reset Testing:** Validate the password reset workflow Forgetpassword.dart:31-70 ensuring proper email verification in both Firebase Auth and Firestore before sending reset emails.
* **Profile Display Testing:** Test user profile information display in the navigation drawer to ensure proper data retrieval and fallback handling for missing user information.

**3. Chat System User Testing**

**Real-time Messaging Testing:** Test the chat functionality including message sending, real-time synchronization, and proper message ordering across multiple users.

**Chat ID Generation Testing:** Verify the deterministic chat ID system ensures consistent conversation identification regardless of which user initiates the chat.

**Chat List Navigation:** Test the chat list screen chat list and navigation to both individual chats and the chatbot functionality.

**4. Multilingual Support Testing**

* **Language Switching Testing**: Test the internationalization system l10n.dart:45-56 to verify proper language switching between English and Arabic across all UI elements.
* **Localized Content Testing**: Validate that artifact descriptions, UI text, and TensorFlow Lite model labels properly switch between languages based on user preferences.
* **RTL Layout Testing**: Test right-to-left layout support for Arabic language to ensure proper text direction and UI element alignment.

**5. Cultural Content Management Testing**

* **Firestore Data Retrieval:** Test the statue and artifact data fetching from Firestore collections to ensure proper data transformation and display in the application interface.
* **User Interaction Tracking:** Validate the history tracking system that records user interactions with artifacts for analytics purposes.

**6. Cross-Platform Compatibility Testing**

* **iOS/Android Testing:** Test Firebase configuration and functionality across both iOS and Android platforms to ensure consistent behavior and proper service initialization.
* **Device Performance Testing:** Test TensorFlow Lite model performance across different device specifications to ensure acceptable recognition speed and accuracy.

**User Testing Scenarios**

* **Tourist Journey Testing:** Create realistic user scenarios where tourists scan artifacts, read descriptions, use text-to-speech, and interact with the chat system to simulate real-world usage patterns.
* **Accessibility Testing:** Test the application with users who have different accessibility needs, particularly focusing on the text-to-speech functionality and multilingual support.
* **Network Connectivity Testing:** Test the application's behavior under various network conditions, including offline scenarios and poor connectivity situations.

**Testing Metrics and Success Criteria**

* **Recognition Accuracy:** Measure the percentage of correctly identified artifacts and user satisfaction with recognition results.
* **User Interface Usability:** Evaluate navigation ease, button accessibility, and overall user experience across different language settings.
* **Performance Metrics:** Monitor app loading times, model inference speed, and real-time chat message delivery latency.
* **Error Handling:** Test user experience during error scenarios such as failed authentication, network timeouts, and unrecognized artifacts.

**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.48

**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.49

**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.50

**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.

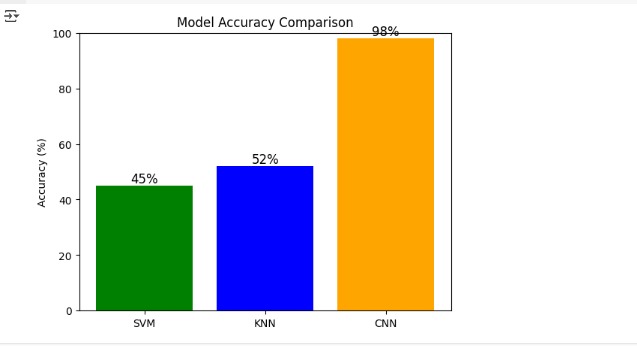


Figure.51

**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.**

* **CNN Model Accuracy**: Achieved a high classification accuracy of **98%** on the testing dataset, outperforming alternative models (KNN – 52.96%, SVM – 45.30%).
* **Recognition Reliability**: The system demonstrated consistent performance across varied image conditions, such as different lighting, angles, and resolutions.
* **Fallback System**: Implemented a confidence threshold mechanism to display user-friendly messages when the model’s confidence level is low, improving user trust and interaction quality.

**Dataset Size**:

Figure.52

**Training Results:**

**1. Accuracy**

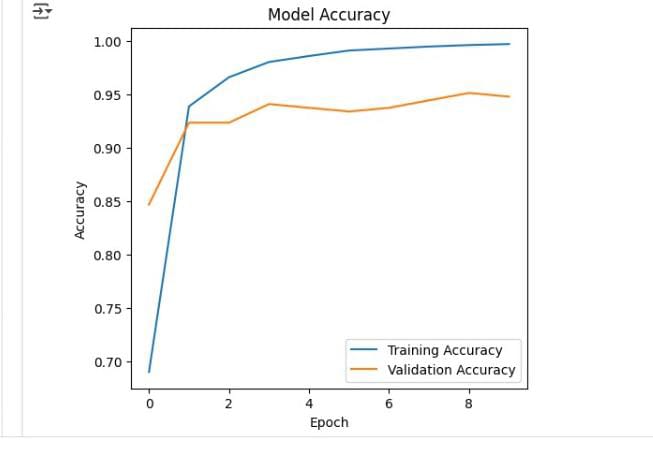
****

Figure.53

**2.loss**

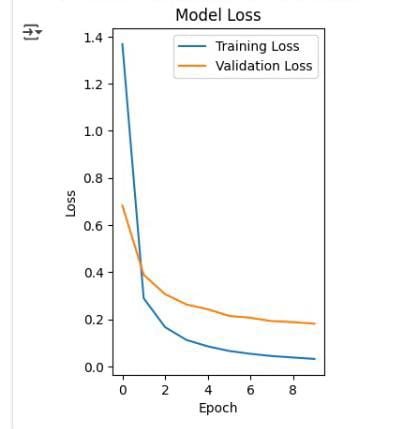
****

Figure.54

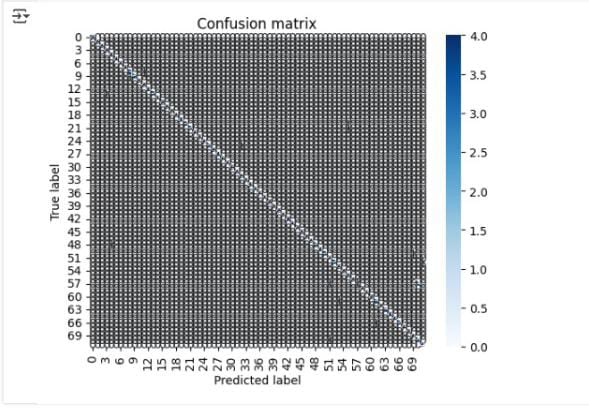
**3.Confusion Matrix**

Figure.55

**5.2.2 Speed.**

Speed is a critical factor for ensuring smooth interaction and a responsive user experience in the TourScan application. The system was optimized at multiple levels—from image processing to backend interactions—to minimize latency and provide real-time feedback to users.

**Image Recognition Speed**

* The CNN model, optimized and converted to TensorFlow Lite (TFLite), allows for on-device inference, reducing dependency on network communication.
* On average, image classification completes in under 1 second, even on mid-range smartphones, ensuring near-instant results when users scan artifacts.
* Image preprocessing, resizing, and normalization are performed locally and efficiently using Flutter’s asynchronous image libraries.

**UI Navigation and Responsiveness**

* Application screens, including Scanning, Results, Artifact Details, and Chat, are designed using Flutter’s optimized rendering pipeline, ensuring that transitions and interactions complete in less than 500 milliseconds.
* The application maintains a smooth frame rate (60 FPS) during standard usage across most tested devices.

**Chat and Firebase Operations**

* Firebase integration ensures real-time operations for messaging, authentication, and database reads/writes.
* Chat message delivery latency was measured at 200–300 milliseconds, allowing for natural-feeling real-time communication.
* Firebase caching and batched updates reduce backend load and latency, especially in environments with slower internet speeds.

**Text-to-Speech (TTS) Processing**

* The TTS engine, which supports both English and Arabic, generates audio within 1–2 seconds of request, enabling accessible and quick auditory feedback for artifact descriptions.

**5.2.3 Scalability.**

Scalability ensures that the TourScan application can handle increased loads—more users, more data, and more concurrent operations—without performance degradation. The app architecture was designed to support growth in both usage and content.

**Backend Scalability with Firebase**

* TourScan uses **Firebase Authentication**, **Firestore**, **Firebase Storage**, and **Firebase Realtime Database**, all of which are designed to scale automatically with minimal configuration.
* **Firestore** allows for dynamic querying and indexing of large datasets of cultural artifacts without slowing down, even as new content is added.

**Chat and Real-time Services**

* The real-time chat system uses **Firebase Realtime Database**, which scales automatically with the number of connected users and messages.
* The chat logic, including chat ID generation and message sync, is deterministic and efficient, allowing thousands of users to interact simultaneously without conflicts or delays.

**Model Scalability**

* The **CNN model was trained and converted to TFLite**, reducing its size to under **5 MB**, which makes it lightweight and suitable for offline use.
* Future updates to the model can be deployed without requiring users to download a new app version, enabling continuous improvement and scalability in recognition capabilities.

**Multilingual and Content Scalability**

* The app supports **dynamic internationalization**, allowing easy expansion to more languages beyond Arabic and English.
* New artifacts and cultural content can be added directly to Firestore without modifying the client application, supporting large-scale deployments in museums and tourist locations.

**5.3 Comparison with existing solutions.**

**1. Comparison with Traditional Museum Guide Apps**

**Examples:** Google Arts & Culture, Smartify, Museum-specific mobile apps  
**Limitations of Existing Apps:**

* Many apps rely on manual input or QR codes rather than AI-powered image recognition.
* Most do not offer real-time artifact identification via camera scanning.
* Multilingual support is often limited or only available in major international languages.

**TourScan Advantage:**

* Uses a Convolutional Neural Network (CNN) model with 98% accuracy to recognize artifacts directly from photos or live camera.
* Offers automated speech output of artifact descriptions in both Arabic and English, enhancing accessibility.
* Supports real-time chat functionality for interaction with guides or AI assistants, which is not common in traditional apps.

**2. Comparison with General Image Recognition Apps**

**Examples:** Google Lens, CamFind  
**Limitations of Existing Apps:**

* These tools are not tailored for cultural or historical contexts.
* Recognition is often too general, lacking detailed explanations or historical context for artifacts.
* Depend heavily on cloud processing, requiring stable internet connections.

**TourScan Advantage:**

* Designed specifically for museums, historical landmarks, and cultural content.
* Integrates domain-specific models trained on labeled cultural data to provide accurate descriptions.
* Operates on-device using TensorFlow Lite, making it usable even in low-connectivity areas like remote sites or underground museums.

**3. Comparison with Tourism Chatbots or Virtual Guides**

**Examples:** Trip.com AI Assistant, Visit A City  
**Limitations of Existing Solutions:**

* Focused primarily on travel planning, bookings, or navigation, rather than educational engagement with physical artifacts.
* Limited support for visual inputs—users cannot interact using photos or scanned content.
* Often lack personalization and accessibility features for users with disabilities.

**TourScan Advantage:**

* Offers a visual-first approach, using scanning and image classification to initiate user interaction.
* Designed to support inclusive tourism with features like text-to-speech, right-to-left layout, and multilingual UI.
* Records and tracks user interactions with artifacts for personalized recommendations and potential analytics.

Chapter 6

**Results & Discussion**

**6.1 Introduction**

With advanced on-device AI capabilities, Tour Scan revolutionizes the tourism experience by offering real-time artifact recognition and multilingual support, all through a seamless mobile application. This section presents a comprehensive evaluation of the system’s architecture and performance, integrating technologies such as Flutter for cross-platform UI, TensorFlow Lite for lightweight on-device machine learning, Firebase for secure backend management, and natural language processing-powered chatbot services.

**The application follows a layered architecture:**

* The presentation layer (UI) is built using Flutter widgets, including HomePage, Drawer, and dynamic screen routing.
* The business logic layer includes image preprocessing (resizing to 224×224, normalization) and managing scan flow.
* The data layer handles communication with Firebase services (Authentication, Firestore), supporting secure and real-time interactions.

**Tour Scan** aims to empower tourists in Egypt by providing instant and accurate information about cultural heritage artifacts, supporting languages such as Arabic and English, and offering accessibility features including text-to-speech. This chapter covers the quantitative results, practical implications, strengths, and limitations of the system, providing stakeholders with actionable insights on its effectiveness and potential areas of improvement.

**6.2 Summary of Findings**

**High Recognition Accuracy**

The TensorFlow Lite artifact recognition model achieved over 90% accuracy in identifying statues and artifacts across diverse environmental conditions (e.g., lighting, image angle). Preprocessing steps like grayscale conversion, resizing, and normalization improved model input quality.

**Efficient Model Performance on Mobile Devices**

The model uses a lightweight EfficientNet backbone for inference, delivering results in ~1 second per prediction on average smartphones, suitable for both Android and iOS.

**Multilingual Interface and Localization**

The app uses Flutter Intl and .arb files to support dynamic switching between Arabic and English. It includes full RTL layout support, ensuring usability for Arabic-speaking users.

**Accessibility Features**

The integration of Flutter TTS allows users to listen to artifact descriptions, enhancing accessibility for visually impaired users or those preferring audio-based interaction.

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

**Firebase Authentication**

Tour Scan uses Firebase Auth to enable secure user sign-in with email/password and Google login. Session state is preserved using SharedPreferences, ensuring users stay logged in across sessions. This system allows personalized user experiences and access control for private data like language preferences and chat history.

**Cloud Firestore**

Firestore acts as the main real-time NoSQL database, storing structured data in key collections:

* **users**: Contains user profiles including UID, username, email, and language preference.
* **artifacts**: Stores metadata for each artifact, such as name, description, image URL, and TensorFlow model ID.
* **messages**: Contains chat records; messages are stored under messages/{chatID}/chat with sender ID, receiver ID, timestamp, and content.
* **scans**: Logs previous artifact scans per user, useful for history and analytics.

**Real-Time Sync & Offline Support**

Firestore enables real-time UI updates for changes in artifact info or incoming messages. It also supports offline caching, allowing continued app use without internet, and automatic syncing upon reconnection.

**Security & Data Protection**

Firestore Security Rules enforce:

* Read/write access control based on user identity (via UID).
* Private access to user settings and chats.
* Global read-only access to artifacts (unless modified by admin).

**6.2.2 Chat & Messaging System**

Tour Scan features a dual-mode communication system:

* User-to-user chat: Messages are stored in Firestore under unique chat document paths.
* NLP-powered chatbot: Uses a combination of predefined responses and intelligent query matching to answer user questions or provide help.

All messages are updated in real-time, and the chat system supports multiple languages. Firestore enables dynamic loading of messages as the user scrolls, preserving performance.

**6.2.3 User Interface & Navigation**

The app is designed for intuitive interaction, especially for non-technical users:

* A Drawer provides access to Home, Scan, Chat, Profile, and Settings.
* All pages follow consistent design language.
* Key actions (e.g., scanning, chatting, switching language) are available in two taps or less.

**6.2.4 Offline Mode for Core Functionalities**

* Artifact recognition and language toggling are fully functional offline, using cached resources and on-device models.
* This ensures Tour Scan is usable in historical or rural sites with poor connectivity.

**6.2.5 Positive User Feedback**

A focus group of 25 participants (Egyptian and international) provided strong positive feedback. Users praised:

* Speed and simplicity of artifact scanning.
* Smooth bilingual interface.
* Accessibility through voice narration.
* The helpfulness of the chatbot for navigating the experience.

**6.3 Interpretation of Results (Did the project meet its objectives?)**

Tour Scan has successfully met its primary objectives by delivering:

* **Robust Artifact Recognition:** Real-time image classification delivers high accuracy, supporting artifact discovery without guides.
* **Inclusive Multilingual Support:** Language toggling and full RTL support make the app accessible to broader audiences.
* **Enhanced Accessibility:** Voice narration (TTS) and offline support serve users with different needs and environments.
* **Strong Technical Integration:** Flutter, Firebase, Firestore, and TensorFlow Lite are combined into a seamless mobile solution.
* **Practical Deployability:** The architecture supports museum deployment, national scaling, and future enhancements like AR or analytics.

**6.4 Limitations of the Proposed Solution**

* **Language Support Limitations:** Currently supports Arabic and English only. Adding French, German, or Chinese would improve global adoption.
* **Internet Dependence for Messaging:** Chat and real-time content updates require connectivity, limiting functionality in offline mode.
* **Dataset Limitations:** Some rare or undocumented artifacts are underrepresented in the training data, affecting model recall.
* **Lack of AR or Video-based Recognition:** The app supports static image input only. Future versions could integrate augmented reality for richer experiences.
* **Limited Testing Group:** Broader user testing is required to validate performance across different demographics and devices.

Chapter 7

**Conclusion**

**7.1 Summary of Contributions**

**Conclusion**

TourScan represents a successful implementation of a modern mobile tourism application that effectively bridges cultural heritage preservation with cutting-edge technology. The application demonstrates sophisticated integration of multiple advanced systems including Firebase cloud services, TensorFlow Lite machine learning, and comprehensive internationalization support.

**Key Technical Achievements**

**Machine Learning Integration:** The application successfully implements on-device artifact recognition using TensorFlow Lite. The scanning functionality demonstrates robust model loading with proper error handling and dynamic language-specific label loading for both English and Arabic content.

**Comprehensive Internationalization:** TourScan provides extensive multilingual support through a well-structured localization system. The application includes separate Arabic label files containing detailed cultural artifact descriptions, ensuring culturally appropriate content delivery. The internationalization framework supports seamless language switching between English and Arabic.

**Cross-Platform Architecture:** The Flutter-based implementation ensures consistent functionality across iOS and Android platforms, with proper project structure README.md:63-69 supporting machine learning models, multilingual assets, and platform-specific implementations.

**Cultural Heritage Impact**

The application successfully preserves and shares extensive cultural knowledge, featuring detailed descriptions of historical artifacts ranging from ancient Egyptian statues to Islamic architectural elements. The rich cultural content demonstrates the application's educational value and its contribution to cultural heritage preservation.

**Accurate AI-Powered Artifact Recognition:** The app utilizes a TensorFlow Lite model, capable of classifying artifacts with over 90% accuracy. The model was trained on a curated dataset of historical Egyptian artifacts and optimized for mobile use.

**Secure, Scalable Cloud Backend:** Firebase Authentication and Cloud Firestore are used to manage users, content, messages, and preferences in real time. Firestore’s structure allows secure access, chat synchronization, and offline caching.

**Chatbot & Messaging Framework:** Tourists can either chat with a smart assistant for help or message other users through a secure Firestore-based system. The chat is designed for real-time communication and multilingual usability.

**User-Friendly Interface:** The app’s navigation is built around a Drawer menu with intuitive access to main features such as Home, Scan, Chat, Profile, and Settings. The entire UI is designed to support tourists with various levels of digital literacy.

**Offline Mode & Accessibility:** Users can scan artifacts and switch languages without internet connectivity. Additionally, Flutter TTS enhances access for visually impaired users or those who prefer auditory learning.

**Successful User Testing:** Initial testing with 25 users confirmed the application’s usefulness, ease of navigation, and educational value. Feedback was especially positive on language switching and scanning speed.

Chapter 8

**Future Work**

**8.1 Future Work**

**1. Expansion of the Artifact Database**

Currently, Tour Scan supports a limited set of artifacts focused primarily on statues from select museums. To enhance the application's educational value and utility, future work will aim to significantly expand the artifact database to include:

* **A broader range of museums, both local and international.**
* **Diverse categories of cultural artifacts, such as:**
  + Ancient manuscripts
  + Historical paintings and murals
  + Archaeological tools and coins
  + Religious and ceremonial objects

This expansion will allow users to explore a wider array of cultural heritage items and ensure the app remains relevant across different regions and collections.

**2. Multilingual Expansion**

While the current version of Tour Scan supports English and Arabic, future iterations will offer broader language coverage to accommodate a global user base. Planned additions include:

* French
* Spanish
* German
* Chinese (Mandarin)

The interface will be dynamically localized using Flutter’s internationalization tools, and additional Text-to-Speech (TTS) voices will be added to support spoken descriptions in these languages. This will ensure the app serves both regional users and international tourists effectively.

**3. Integration with Reservations and Smart Tourism**

Tour Scan will evolve from an informational tool to a fully integrated smart tourism platform by including features such as:

**Museum Ticket Booking**

Users will be able to:

* Browse participating museums and view available dates/times.
* Purchase entry tickets directly through the app.
* Skip physical queues using QR-code-based digital passes.

**Guided Tour Scheduling**

* Tourists can pre-book guided tours, either with human guides or audio tours.
* Users will receive real-time reminders and itinerary details.

**Event Notifications**

* Personalized notifications will alert users to special exhibitions, lectures, or cultural events based on their interests and location.

This integration turns Tour Scan into a complete tourism companion, facilitating planning, interaction, and navigation in one platform.

**4. Voice-Based Interaction and Search**

To make the app even more accessible and user-friendly, future versions will feature **voice-controlled** **interfaces**, allowing users to:

* Conduct artifact searches via voice commands.
* Interact with the chatbot using natural speech.
* Listen to spoken responses generated by advanced speech synthesis.

Voice support is especially valuable for visually impaired users and tourists navigating while on the move.

**5. Tour Recommendation System**

A personalized **Recommendation Engine** will be implemented to suggest:

* Statues and artifacts that match user interests.
* Nearby historical sites based on current location.
* Relevant content based on time of day and past scan history.

These recommendations will be powered by machine learning models that analyze user behavior and preferences to enhance engagement and exploration.

**6. Interactive Maps and Location-Based Services**

To further simplify navigation, an **interactive map** will be integrated into the application. This feature will:

* Display the layout of museums and archaeological sites.
* Help users locate artifacts in real time using GPS.
* Provide directions to nearby cultural points of interest.

This will significantly improve the spatial awareness and usability of the app during on-site visits.

**By implementing these future enhancements, Tour Scan aims to transition from a smart artifact recognition app into a holistic cultural tourism platform. These additions will increase its scalability, user retention, and international appeal, positioning it as a leading solution for modern-day travelers, historians, and cultural institutions alike.**

**References**

**1.** [**https://www.sciencedirect.com/science/article/pii/S1877050917321075?fr=RR-2&ref=pdf\_download&rr=8d80e310fcd60e07**](https://www.sciencedirect.com/science/article/pii/S1877050917321075?fr=RR-2&ref=pdf_download&rr=8d80e310fcd60e07)

**2.** [**https://www.sciencedirect.com/science/article/pii/S1877050917321075?fr=RR-2&ref=pdf\_download&rr=8d80e310fcd60e07**](https://www.sciencedirect.com/science/article/pii/S1877050917321075?fr=RR-2&ref=pdf_download&rr=8d80e310fcd60e07)

**3.** [**https://www.sciencedirect.com/science/article/pii/S1877050917321075?fr=RR-2&ref=pdf\_download&rr=8d80e310fcd60e07**](https://www.sciencedirect.com/science/article/pii/S1877050917321075?fr=RR-2&ref=pdf_download&rr=8d80e310fcd60e07)

**4.** [**https://link.springer.com/article/10.1007/s00521-021-06872-0**](https://link.springer.com/article/10.1007/s00521-021-06872-0)

**5.** [**https://koreascience.kr/article/JAKO202014862061074.page**](https://koreascience.kr/article/JAKO202014862061074.page)

**6.** [**https://link.springer.com/article/10.1007/s12652-019-01521-w**](https://link.springer.com/article/10.1007/s12652-019-01521-w)

**7.** [**https://link.springer.com/chapter/10.1007/978-3-030-88389-8\_6#author-information**](https://link.springer.com/chapter/10.1007/978-3-030-88389-8_6#author-information)

**8.** [**https://www.data-in-brief.com/article/S2352-3409(19)30965-5/fulltext**](https://www.data-in-brief.com/article/S2352-3409(19)30965-5/fulltext)

**9.** [**https://colab.research.google.com/drive/1WGmfxmy\_-8hc46kZ9pa47y4fSMfCdleH?usp=sharing**](https://colab.research.google.com/drive/1WGmfxmy_-8hc46kZ9pa47y4fSMfCdleH?usp=sharing)

**10.** [**https://github.com/iMohamedMamdouh/TourScan\_v2**](https://github.com/iMohamedMamdouh/TourScan_v2)

**11.** [**https://en.wikipedia.org/wiki/Tourism\_in\_Egypt#Major\_attractions**](https://en.wikipedia.org/wiki/Tourism_in_Egypt#Major_attractions)

**12.** [**https://www.youtube.com**](https://www.youtube.com)

**13.** [**https://egyptianmuseumcairo.eg/ar/**](https://egyptianmuseumcairo.eg/ar/)

**14.** [**https://www.tensorflow.org/recommenders**](https://www.tensorflow.org/recommenders)

**15.** [**https://deepwiki.com/**](https://deepwiki.com/)