Helwan University
Faculty of Engineering
Department of Electronics, Communications & Computer B.sc
Final Year Project

InMuse: A Smarter Museum Experience

Powered by Indoor Localization and Augmented Reality

BY
Ziad Assem Muhammad Abdulaziz
Ramy Mohamed Sayed Abdel-Khalik
Pavly Safwat Georgy Salib
Shrouk Ashraf Elsaeid Elsaeid

Under Supervision of

Prof. Shahira Habshy
Professor of Electronics, Communication, and Computer Engineering
Faculty of Engineering - Helwan University

July 2022/2023

ACKNOWLEDGEMENT

For our professor Prof. Shahira Habshy: We would like to express our heartfelt gratitude and appreciation for your invaluable guidance and support throughout our project. Your expertise, motivation , and guidance have been instrumental in helping us navigate the complexities of this endeavor.

From the very beginning, you have been a constant source of inspiration, pushing us to explore new ideas and challenging us to think critically. Your unwavering belief in our abilities has fueled our motivation and encouraged us to strive for excellence.

We are truly grateful for the knowledge and skills you have imparted to us, which will undoubtedly have a lasting impact on our academic and professional journeys. Thank you once again, Professor Shahira, for your unwavering support and for being an exceptional mentor.

With utmost gratitude,

ABSTRACT

In this project, we present an innovative Android application that revolutionizes museum visits and tourism through the integration of cutting-edge technologies, including Artificial Intelligence (AI), indoor localization, chatbots, and augmented reality (AR). The primary aim of our project is to create an engaging, interactive, and educational platform that bridges the gap between traditional exhibits and the digital era, offering visitors an enriched experience.

The Android application leverages AI-powered indoor localization to provide real-time, accurate positioning within the museum, ensuring users navigate effortlessly through intricate exhibit layouts. A customized chatbot interface facilitates seamless interaction with the application, enabling users to access personalized information, answer questions, and delve deeper into historical contexts and artifacts.

The integration of augmented reality introduces a dynamic layer to the museum experience, superimposing digital content onto physical exhibits. Users can witness historical scenes, interact with virtual artifacts, and embark on virtual tours, fostering a deeper connection with the cultural and historical significance of the showcased items.

The project encompasses extensive user testing and iterative design, focusing on intuitive and user-friendly features that cater to a diverse audience. Security and privacy measures are also paramount, ensuring data protection and user confidentiality in compliance with international standards.

The research contributes to the growing field of smart tourism and digital cultural heritage by showcasing the potential of AI-driven applications to redefine the way visitors engage with museums and historical sites. The proposed Android application holds promise for curators and tourism officials seeking to enhance the overall visitor experience, foster cultural awareness, and promote the preservation of our shared heritage. Moreover, the study opens avenues for further exploration into the intersection of AI, AR, and tourism, setting the stage for transformative advancements in the realm of digital travel and cultural appreciation.

List of figures

List of Tables

List of abbreviations

AI - Artificial Intelligence
GPS - Global Positioning System
RSSI - Received Signal Strength Indicator
ML - Machine Learning
AR - Augmented reality
GAN - Generative adversarial network

LTE - Long-Term Evolution
NFC - Near Field Communication
GCM - Google Cloud Messaging
BaaS - Backend as a Service
SDK - Software Development Kit
SQL - Structured Query Language
NoSQL - not only SQL

RPC - Remote Procedure Call
IDL - Interface Definition Language
gRPC - Google Remote Procedure Call
HTTP - Hypertext Transfer Protocol
Protobuf - Protocol Buffers
TCP - Transmission Control Protocol
IP - Internet Protocol
UDP - User Datagram Protocol
VR - Virtual Reality
SLAM - Simultaneous Localization and Mapping
API - Application Programming Interface
JVM - Java Virtual Machine
OpenGL - Open Graphics Library
GPU - Graphics Processing Unit
UI - User Interface
APK - Android Package Kit
KNN - K-Nearest Neighbour
ANN - Artificial Neural Networks
MLP - Multilayer Perceptron
LLM - Large Language Models
ID - Identity
URL - Uniform Resource Locator
lat - Latitude
lng - longitude
CSV - Comma-Separated Values
RF - Random Forest
MLM - Masked Language Modeling
PCM - Pulse Code Modulation
WAV - Waveform Audio File Format
CD - Compact Disc

CHAPTER ONE
INTRODUCTION

1.1 Introduction

Museums and historical places offer great income, representatives for a nation's history, and entertainment places. But for now, the museums' income is about 69.1% of the total income for entertainment places although they attract almost 23.5% of total visitors (source Central Agency for Public Mobilization And Statistics book version 2020), that is because museums and historical places provide the first and the second mentioned features without the third one which is entertainment.

Now Mobile application features and new AI technologies can be combined together to give a smarter, more powerful, and unforgettable experience for visiting any museum or even any place.

1.2 Problem Statement

The project is focusing on solving multiple problems and providing multiple solutions that can be implemented **independently** of each other, we chose a **museum application** from our belief that it is the best representative for our solutions:

1) The first problem is **Indoor localization**:

- **GPS** is not the best solution for indoor localization due to GPS signals being blocked or being reflected by walls and cannot enter the room. As a result, satellite signals cannot be received properly, so it is impossible to calculate the location due to **insufficient** signal strength inside the room.
- Tracking people or customers inside indoor regions (e.g. inside Mall or museum) is done in other several ways including reading Wi-Fi RSSI signals but, it is not the most accurate method due to accuracy and multipath issues, **so our suggestion is to use ML models to localize mobile phones inside buildings(eg. museums)**.

2) The second problem is that some historic places are boring or just empty places so we people are not visiting them. Hence we will make use of **AR Figures**; by displaying 3D AR Models inside their appropriate place we will enhance the experience.

3) There is a **sociotechnical problem** that the tour guides will oppose the application because the main function of the application is to replace the tour guides.

4) Another **inherent problem** in the project is that the users will **uninstall** the application after visiting the museums, or in other words they don't find their **community** in the application.

1.3 Available Solution

The Application introduces solutions for the stated problems:

- **Indoor localization** uses state of art **neural networks** and data augmented by GAN to localize mobile phone users. This technology is ready to be applied indoors (in malls, shops, labs, ...etc).
- **Augmented reality** is not widely implemented in Android phones, hence few applications support AR, the Application uses new solutions for AR in Android development and GPS tracking to position the 3D Models in their desired place.
- The sociotechnical problem arises when the application replaces the tour guides, so our application will cooperate with tour guides and not replace them , hence can be adopted and distributed in the market, by providing **broadcasting services** over the Wi-Fi network through the group phones and replacing the special broadcasting device.
- An inherent problem is the application is uninstalled when the user isn't active or not planning to visit any museums he will uninstall the application. to increase engagement with the application we will send push notifications and community posts between the users and provide an AI-Powered chatbot to help the users.
- A technical problem is how to integrate different components written in different programming languages inside a single application, the project provides an effective solution for it with organized dataflow.

1.4 Project Phases

Phase 1: Develop museums browsing application with indoor localization models.

Phase 2: improve the application with a community section

Phase 3: enhance the application with a Question answering bot

Phase 4: power the museum visiting experience with AR features

1.5 System Block diagram

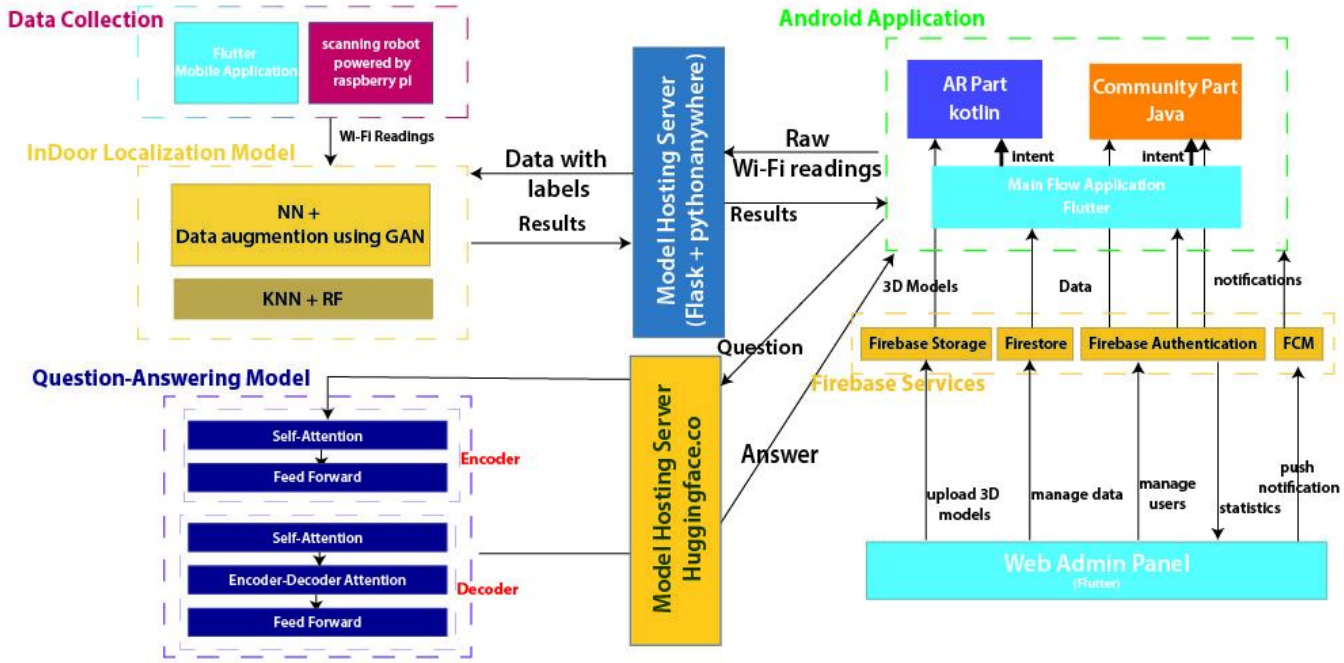


Fig 1.1 System block diagram

CHAPTER TWO
BACKGROUND

2.1 Android

Android is an open-source operating system for mobile devices that is based on the Linux kernel. It was developed by the Open Handset Alliance, led by Google, and other companies.

Android's open-source nature means that the operating system's source code is freely available to developers, who can modify and distribute it as they see fit.

2.1.1 Why Android?

When a device goes from just working to actually making life easier, Android is behind it. It's the reason your GPS avoids traffic, your watch can text and your Assistant can answer questions. It's the operating system inside 2.5 billion active devices. Everything from 5G phones to stunning tablets, Android powers them all.

On Android, you get to decide when and if your data is shared, like your Web & App Activity or Location History. If an app accesses your location while you are not using it, you'll get a notification. And if you ever want to change permissions, all your privacy settings are in one place. It's privacy, with you in the driver's seat.

Everyone has their way of using their devices. That's why we build accessible features and products that work for the various ways people want to experience the world. Screen readers, sound mufflers, and even AR walking guides. Because when it comes to technology, there's no one-size-fits-all.

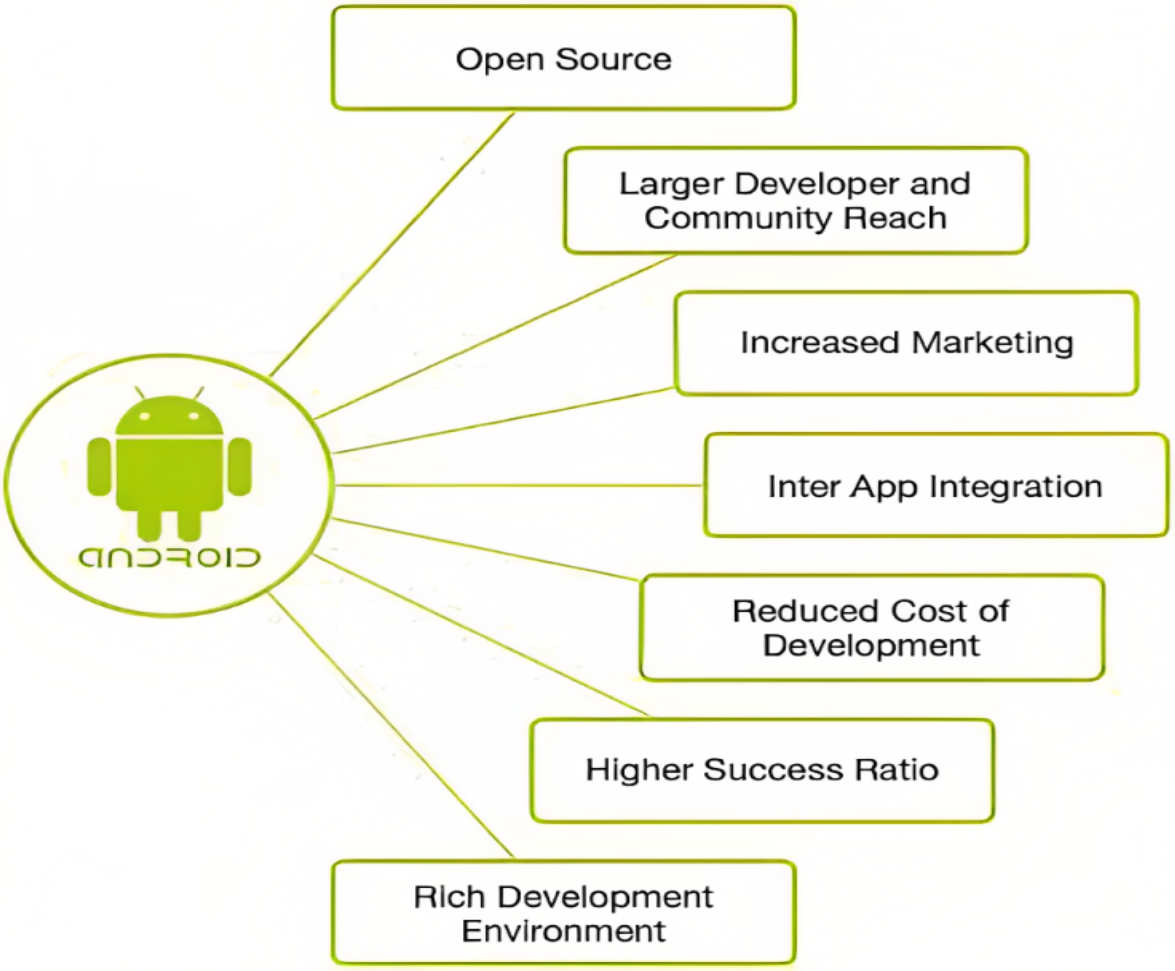


Figure 2.1 Android Benefits

2.1.2 Android Features

1- Connectivity: Bluetooth, Wi-Fi, LTE, NFC and WiMAX.

2- Storages: SQLite used for data storage purposes.

3- Media Support: JPEG, PNG, GIF and BMP.

4- Messaging: SMS and MMS.

5- Multi-tasking: Users can jump from one task to another and at the same time various applications can run simultaneously.

6- Resizable Widgets: Widgets are resizable, so users can expand them to show more content or shrink them to save space.

7- Multi-Language: Supports single direction and bi-directional text.

8- GCM: Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.

2.2 Java

Java is a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies. It is a computing platform for application development. Java is fast, secure, and reliable, therefore. It is widely used for developing Java applications in laptops, data centers, game consoles, scientific supercomputers, cell phones, etc.

2.2.1 Java Platform

Java Platform is a collection of programs that help programmers to develop and run Java programming applications efficiently. It includes an execution engine, a compiler, and a set of libraries in it. It is a set of computer software and specifications. James Gosling developed the Java platform at Sun Microsystems, and the Oracle Corporation later acquired it.

2.2.2 What is java used for?

1- It is used for developing Android Apps.
2- Helps you to create Enterprise Software.
3- Wide range of Mobile java Applications.
4- Scientific Computing Applications.
5- Used for Big Data Analytics.
6- Java Programming of Hardware devices.
7- Used for Server-Side Technologies like Apache, JBoss, GlassFish, etc.

2.3 Firebase Services

2.3.1 Firebase Introduction

Firebase is a Backend-as-a-Service (BaaS) platform that provides developers with a range of tools and services to help them build and scale their applications. It was originally developed by Firebase Inc., which was acquired by Google in 2014.

Firebase offers a range of features and services, including real-time database, cloud storage, authentication, hosting, and analytics. These services are built on top of Google's infrastructure, which means that they are highly scalable, reliable, and secure.

2.3.2 Firebase Key Features

2.3.2.1 Authentication

It supports authentication using passwords, phone numbers, Google, Facebook, Twitter, and more. The Firebase Authentication (SDK) can be used to manually integrate one or more sign-in methods into an app.

2.3.2.2 Realtime database

Data is synced across all clients in real-time and remains available even when an app goes offline.

2.3.2.3 Hosting

Firebase Hosting provides fast hosting for a web app; content is cached into content delivery networks worldwide.

2.3.2.4 Test lab

The application is tested on virtual and physical devices located in Google’s data centers.

2.3.2.5 Notifications

Notifications can be sent with Firebase with no additional coding.

2.3.3 Firestore

Firestore is a NoSQL (aka "not only SQL"), document-oriented database that is designed to store and manage large amounts of structured data. It is a part of the Firebase platform and is often used in mobile and web applications.

Unlike traditional SQL databases, Firestore does not use tables or rows to organize data. Instead, data is stored in documents, which are organized into collections. Each document contains a set of key-value pairs, which can be nested to represent complex data structures.

Firestore is designed to provide a flexible and scalable data storage solution for modern applications. It allows developers to store and retrieve data in real-time, making it ideal for applications that require frequent updates and real-time synchronization across multiple devices.
Firestore also offers a range of features and services, including real-time updates, offline data support, security rules, and integration with other Firebase services. These features help developers to build powerful and scalable applications with ease.

2.3.4 Remote Procedure Call (RPC)

Remote Procedure Call is a software communication protocol that one program can use to request a service from a program located in another computer on a network without having to understand the network's details.
RPC is used to call other processes on remote systems like a local system. A procedure call is also sometimes known as a function call or a subroutine call.

RPC uses the client-server model. The requesting program is a client, and the service-providing program is the server. Like a local procedure call, an RPC is a synchronous operation requiring the requesting program to be suspended until the results of the remote procedure are returned. However, the use of lightweight processes or threads that share the same address space enables multiple RPCs to be performed concurrently.

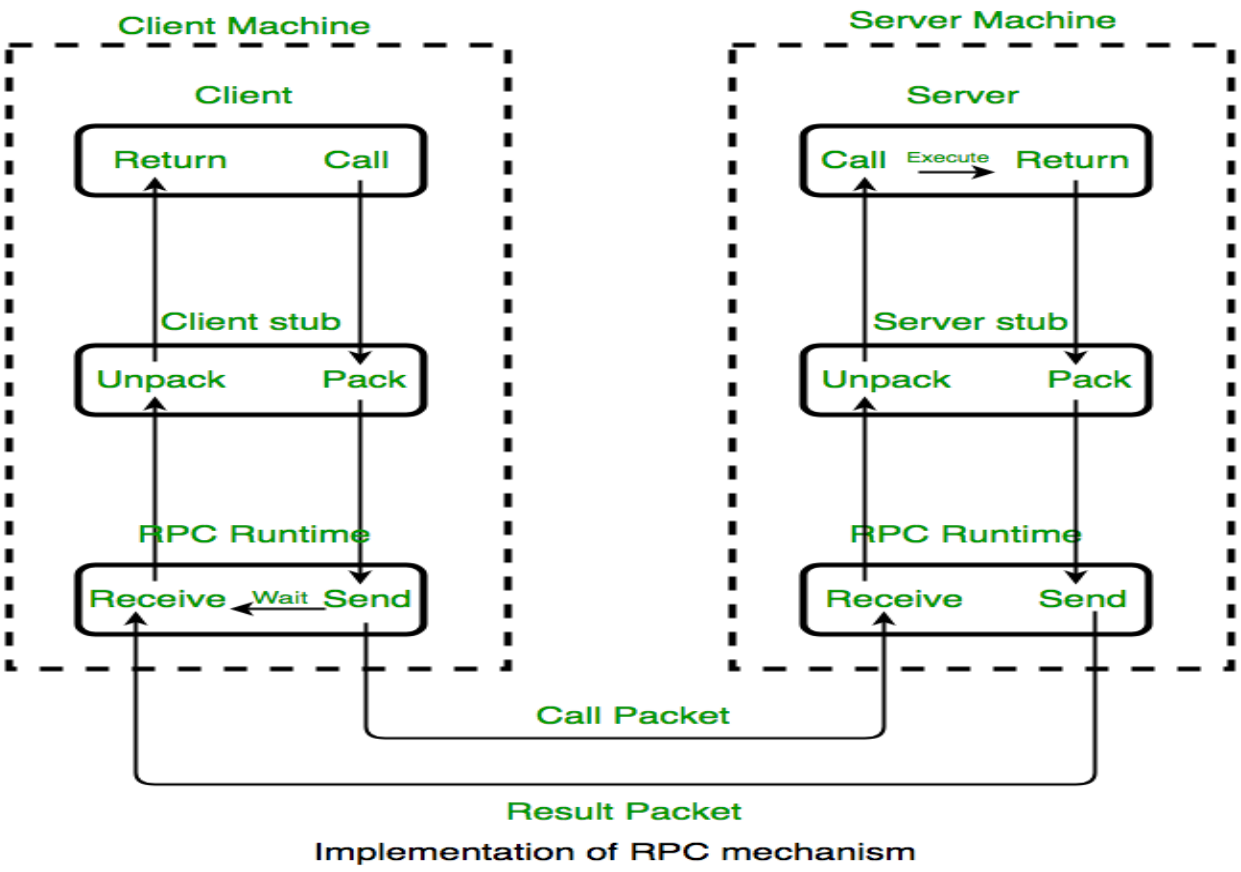


Figure 2.2 RPC Architecture

It uses IDL (Interface Definition Language) as a form of contract on functions to be called and on the data type.

IDL is a programming language-independent specification language that is used to define the interface between software components. IDL is typically used in distributed computing environments, where software components need to communicate with each other over a network.

IDL provides a standardized way to describe the interface of a software component, including the types of data that it can accept and return, the operations that it supports, and any error conditions that it can generate. IDL is used to generate code stubs and skeletons that can be used to implement the software component in a specific programming language.

2.3.5 Google Remote Procedure Call (gRPC)

gRPC is an open-source remote procedure call framework that was developed by Google. It is designed to be fast, efficient, and scalable, making it ideal for building distributed systems and microservices.

gRPC uses a binary protocol and supports multiple programming languages, including C++, Java, Python, and Go. **It is based on the HTTP/2 protocol**, which allows for efficient transport of data over the network.

2.3.5.1 gRPC Architecture:

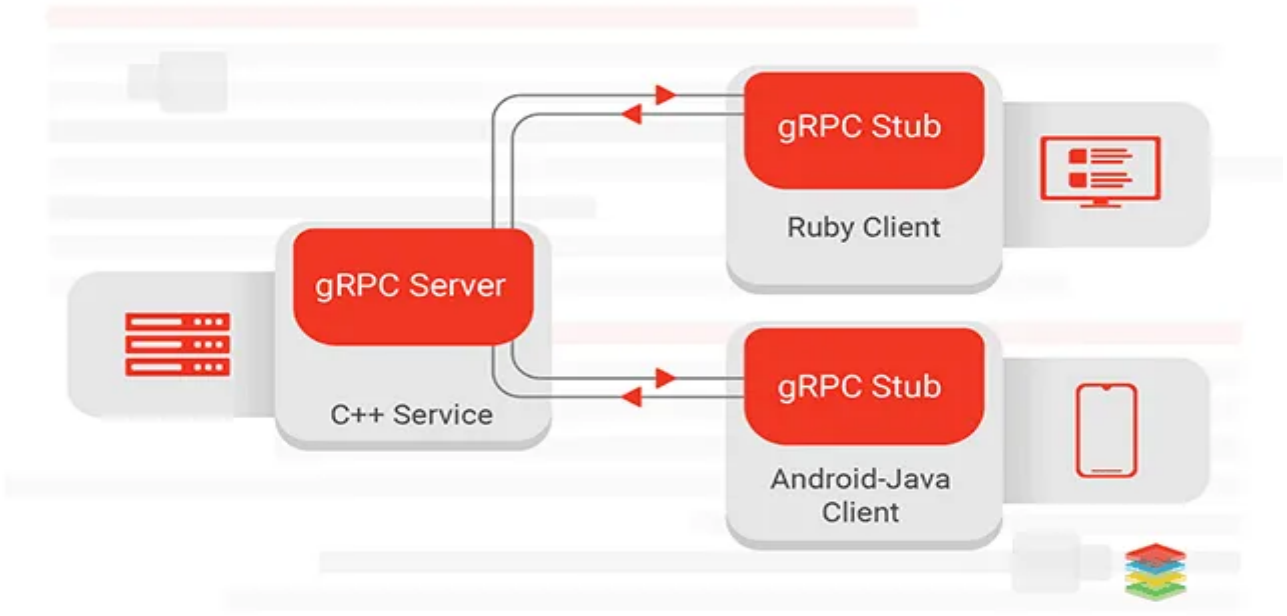


Figure 2.3 gRPC Architecture

- A client application is able to call methods directly on a server-side application present on other machines.
- Service is defined, and methods are specified which can be further remotely called with their parameters and return types.
- On the other hand, the server runs a gRPC server to handle client calls.
- It uses protocol buffers as the Interface Definition Language to enable communication between two different systems used for describing the service interface and the structure of payload messages.

- HTTP/2 — gRPC is basically a protocol built on top of HTTP/2. HTTP/2 is used as a transport.
- Protobuf serialization — Messages that we serialize both for the request and response are encoded with protocol buffers.
- Clients open one long-lived connection to the gRPC server.
- A new HTTP/2 stream for each RPC call.
- Allows Client-Side and Server-Side Streaming.

2.3.5.2 Why use gRPC?

1) Facilitate push notifications:

It can be used as a communication protocol between a mobile app and a backend server to facilitate push notifications. When a mobile app registers to receive push notifications, it can use gRPC to send a registration request to the backend server, along with its device token. The backend server can then use gRPC to send push notifications to the mobile app when new content is available.

2) Support Bidirectional Streaming:

It supports bidirectional streaming and efficient data serialization makes it an ideal choice for implementing push notifications in modern mobile and web applications. It allows for real-time updates and efficient communication between the client and server, making it a popular choice among developers for building scalable and responsive applications.

3) It is supported in a lot of languages.

2.4 Audio Multicast

In networking, multicast is group communication where data transmission is addressed to a group of destination devices simultaneously. Multicast can be one-to-many or many-to-many distribution. Multicast should not be confused with physical layer point-to-multipoint communication.

In multicast, the single sender to multiple receivers is aware of the destination addresses of the receivers, unlike broadcast where the single sender is not aware of his subscribers.

2.4.1 Network Sockets

Network sockets are a fundamental concept in computer networking that enables communication between applications over a network. A socket acts as an endpoint for sending or receiving data across a network. It provides an interface for programs to connect, communicate, and exchange data with other systems. Sockets are essential for building networked applications such as web browsers, email clients, and file transfer protocols.

In a typical client-server model, a socket establishes a connection between a client and a server. The client-side socket initiates the connection by specifying the IP address and port number of the server it wants to communicate with. The server socket listens for incoming connection requests on a specific port. Once the connection is established, both the client and server sockets can send and receive data through the connection. Sockets provide a reliable and efficient means of data transfer, allowing applications to communicate across different network protocols such as TCP/IP or UDP.

Sockets also support various communication paradigms, such as connection-oriented and connectionless protocols. Connection-oriented sockets, like TCP sockets, ensure data delivery by establishing a reliable, ordered, and error-checked connection between the communicating parties. On the other hand, connectionless sockets, such as UDP sockets, offer a lightweight, low-latency communication method that doesn't guarantee the delivery or ordering of packets. The flexibility and versatility of network sockets make them a powerful tool for developers to create networked applications that can communicate and exchange data seamlessly over different networks.

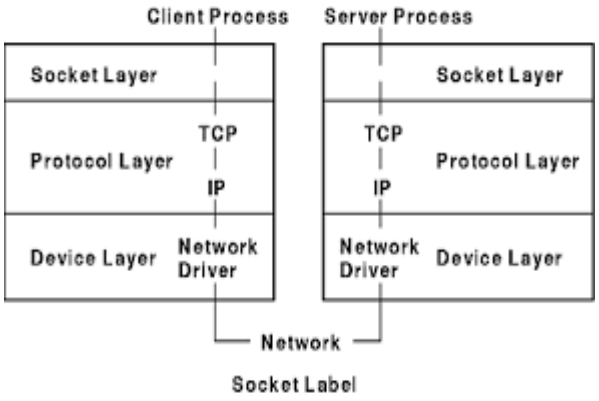


Fig 2.4 Socket communication

2.5 Augmented Reality

Augmented Reality (AR) is a technology that combines the real world with computer-generated elements, enhancing a user's perception and interaction with their environment. AR overlays virtual content, such as images, videos, 3D models, or information, onto the real world, allowing users to experience an enhanced reality.

Unlike virtual reality (VR), which immerses users in a completely simulated environment, AR blends digital content seamlessly into the real world. It enables users to see and interact with both the physical and virtual worlds simultaneously.

AR technology utilizes various techniques to achieve this integration, including

1. Marker-based AR: In this approach, specific markers or patterns in the real world act as triggers for displaying virtual content. When the device's camera recognizes these markers, it superimposes the corresponding digital content onto the marker.

2. Markerless AR: This technique uses computer vision algorithms and sensors, such as GPS, accelerometers, or gyroscopes, to understand the real-world environment without the need for predefined markers. It enables the placement of virtual content in the real world based on the device's position and orientation.

3. Projection-based AR: This method involves projecting virtual images or videos onto real-world surfaces, such as walls or objects, using projectors. The projected content interacts with the physical environment, creating an augmented experience.

4. Simultaneous Localization and Mapping (SLAM): SLAM combines camera and sensor data to create a real-time map of the environment while tracking the device's position and orientation. It allows virtual content to be placed accurately and interact with the surroundings.

AR technology has a wide range of applications across various industries, including gaming, education, healthcare, architecture, marketing, and more. It offers unique opportunities for interactive experiences, visualization, training, and information display, providing users with a new way to engage with the world around them.

In our project implementation, we are using the library of Google Sceneform alongside another API named SceneView.

Google Sceneform primarily utilizes markerless AR technology to create augmented reality experiences. It leverages computer vision algorithms, along with device sensors like cameras, gyroscopes, and accelerometers, to understand and interact with the real-world environment without the need for specific markers.

Sceneform uses techniques such as environmental understanding and plane detection to analyze the surroundings and accurately place virtual objects in the physical space. It relies on the ARCore framework, which provides the underlying technology for markerless AR functionality.
With Sceneform, developers can create AR applications that seamlessly integrate virtual content into the real world, allowing users to interact with 3D models, animations, and other digital elements in a natural and immersive manner.

2.5.1 Kotlin

Kotlin is a modern, statically-typed programming language that runs on the Java Virtual Machine (JVM). It was developed by JetBrains and first released in 2011. Kotlin is designed to be interoperable with existing Java code, allowing developers to easily integrate it into their Java-based projects.

Key features of Kotlin include:

1. Concise Syntax: Kotlin offers a more concise and expressive syntax compared to Java, reducing boilerplate code and improving code readability.

2. Null Safety: Kotlin has built-in null safety features, reducing the risk of null pointer exceptions by distinguishing nullable and non-nullable types at the language level.

3. Type Inference: Kotlin's type inference system allows developers to omit explicit type declarations in many cases, making the code more concise and readable.

4. Functional Programming Support: Kotlin includes support for functional programming concepts such as higher-order functions, lambda expressions, immutability, and extension functions, enabling more concise and expressive code.

5. Coroutines: Kotlin provides native support for coroutines, which simplifies asynchronous programming by allowing developers to write asynchronous code in a more sequential and readable manner.

6. Interoperability: Kotlin seamlessly interoperates with existing Java code, allowing developers to leverage their Java libraries, frameworks, and tools.

7. Official Android Support: Kotlin is fully supported by Google as a first-class language for Android app development. It offers enhanced productivity, improved code safety, and better integration with existing Android APIs.

Kotlin has gained significant popularity among developers due to its modern features, improved developer experience, and strong community support. It is widely used for Android app development, server-side development, and various other application domains.

2.5.2 ARCore

ARCore is a software development platform developed by Google that enables the creation of augmented reality (AR) experiences on Android devices. It provides developers with tools, APIs (Application Programming Interfaces), and libraries to build AR applications that can overlay virtual content onto the real world.

ARCore uses the device's camera and sensors to understand and track the user's environment. It performs tasks such as motion tracking, environmental understanding, and light estimation to create a digital representation of the physical world. This allows virtual objects to be accurately placed and anchored in the real world, maintaining their position and orientation as the user moves the device.

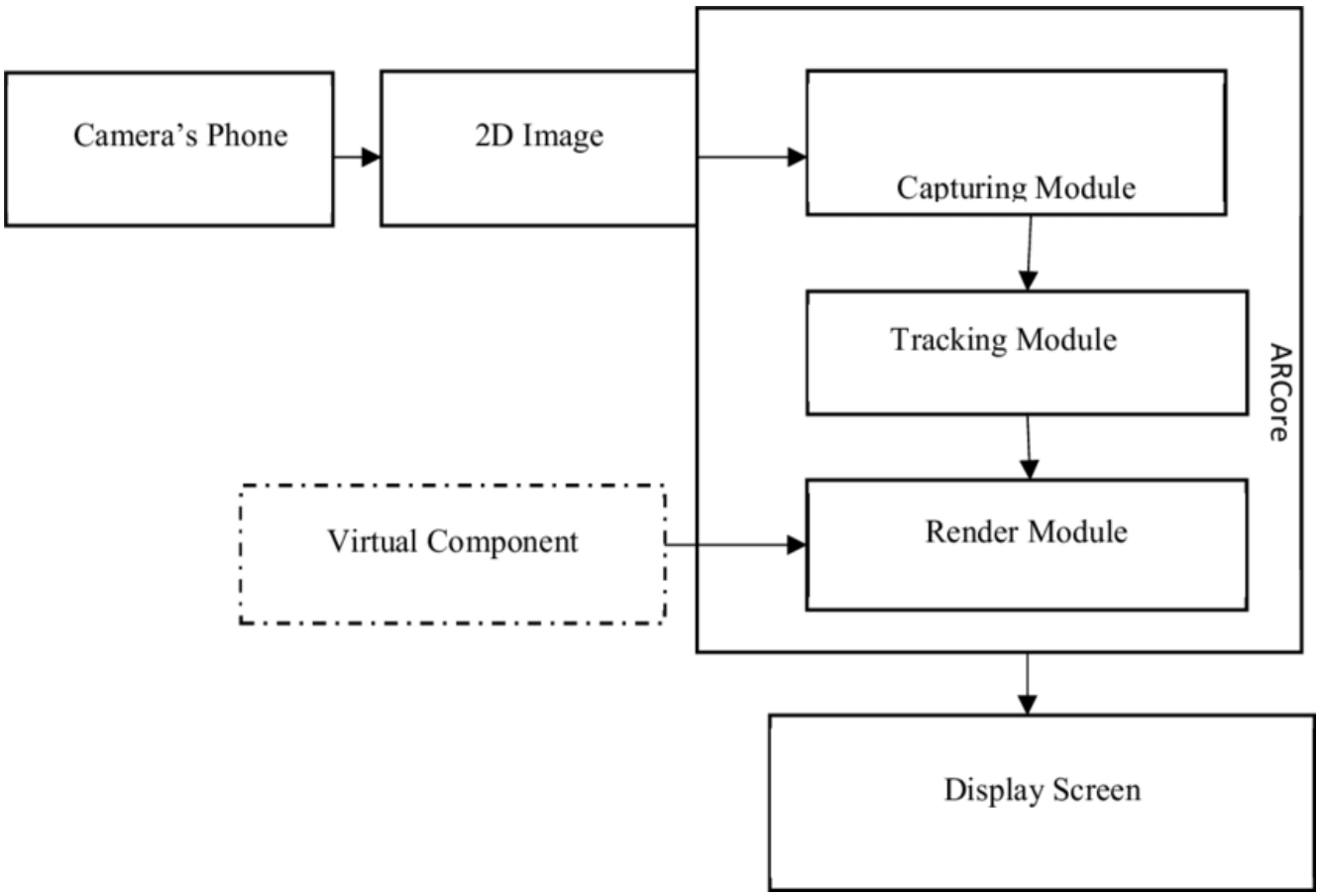


Figure 2.5 ARCore architecture

Key features of ARCore include:

1. Motion Tracking: ARCore enables accurate tracking of the device's position and orientation, allowing virtual objects to be placed and anchored in the real world.

2. Environmental Understanding: ARCore can detect and understand the user's environment, recognizing flat surfaces such as floors, tables, and walls. This information helps in placing virtual content more accurately.

3. Light Estimation: ARCore analyzes the ambient light conditions in the real world, allowing virtual objects to be rendered with appropriate lighting, shadows, and reflections.

4. Anchors: ARCore supports the concept of anchors, which are points in the physical environment where virtual objects can be attached or placed. Anchors help maintain the position and alignment of virtual objects even as the user moves the device.

ARCore is widely used by developers to create a variety of AR experiences, including interactive games, educational applications, visualization tools, e-commerce experiences, and more. It provides a powerful and accessible platform for building augmented reality applications on Android devices.

2.5.3 Unity

Unity is a popular cross-platform game development engine that is widely used for creating interactive and immersive experiences, including augmented reality (AR) applications. Unity provides a robust set of tools and features specifically designed for AR development, making it a popular choice for AR projects.

Unity's AR functionality is primarily powered by two key components:

1. Unity AR Foundation: AR Foundation is a framework provided by Unity that allows developers to create AR applications that can run on multiple AR platforms, including ARCore for Android and ARKit for iOS. It provides a unified API for common AR tasks, such as camera access, plane detection, raycasting, and object tracking. AR Foundation simplifies the development process by allowing developers to write code once and deploy it across different AR platforms.

2. Unity ARKit Plugin and Unity ARCore Plugin: These are additional plugins provided by Unity that offer platform-specific functionality for ARKit and ARCore respectively. They provide access to advanced features and capabilities specific to each platform, allowing developers to take full advantage of the AR capabilities on iOS and Android devices.

With Unity's AR functionality, developers can create AR applications that can track and place virtual objects in the real world, recognize and interact with real-world surfaces and objects, and provide immersive AR experiences through features like animations, audio, and user interactions.

Unity's comprehensive development environment, extensive community support, and the ability to deploy applications to multiple platforms make it a popular choice for building AR experiences. It simplifies the development process and empowers developers to create rich and engaging augmented reality applications.

2.5.4 OpenGL

OpenGL (Open Graphics Library) is an open-source cross-platform API (Application Programming Interface) for rendering 2D and 3D computer graphics. It provides a set of functions and commands that enable developers to interact with the graphics hardware and create visually rich and interactive applications.

Key features and concepts of OpenGL include:

1. Rendering Pipeline: OpenGL follows a programmable rendering pipeline that consists of several stages, including vertex processing, primitive assembly, rasterization, fragment processing, and framebuffer operations. Developers can customize and control each stage to achieve the desired visual output.

2. Rendering Primitives: OpenGL supports rendering various geometric primitives such as points, lines, and triangles. These primitives can be combined and transformed to create complex 3D models and scenes.

3. Shaders: OpenGL allows developers to write custom shaders, which are programs that run on the GPU (Graphics Processing Unit). Shaders are responsible for processing vertices and fragments during the rendering pipeline, enabling advanced lighting, shading, and special effects.

4. Texture Mapping: OpenGL supports texture mapping, allowing developers to apply 2D images or textures to 3D models. This enables realistic rendering of surfaces and enhances visual quality.

5. Transforms and Matrices: OpenGL provides functions for transforming and manipulating objects in 3D space using matrices. Developers can apply translations, rotations, scaling, and other transformations to position and orient objects within the scene.

6. Hardware Acceleration: OpenGL takes advantage of the graphics hardware capabilities to perform efficient rendering, leveraging the power of the GPU to achieve high-performance graphics rendering.

OpenGL is widely used in various domains, including video games, virtual reality, computer-aided design, scientific visualization, and more. It provides a low-level, platform-independent interface for graphics programming, allowing developers to create visually compelling and interactive applications across different platforms and devices.

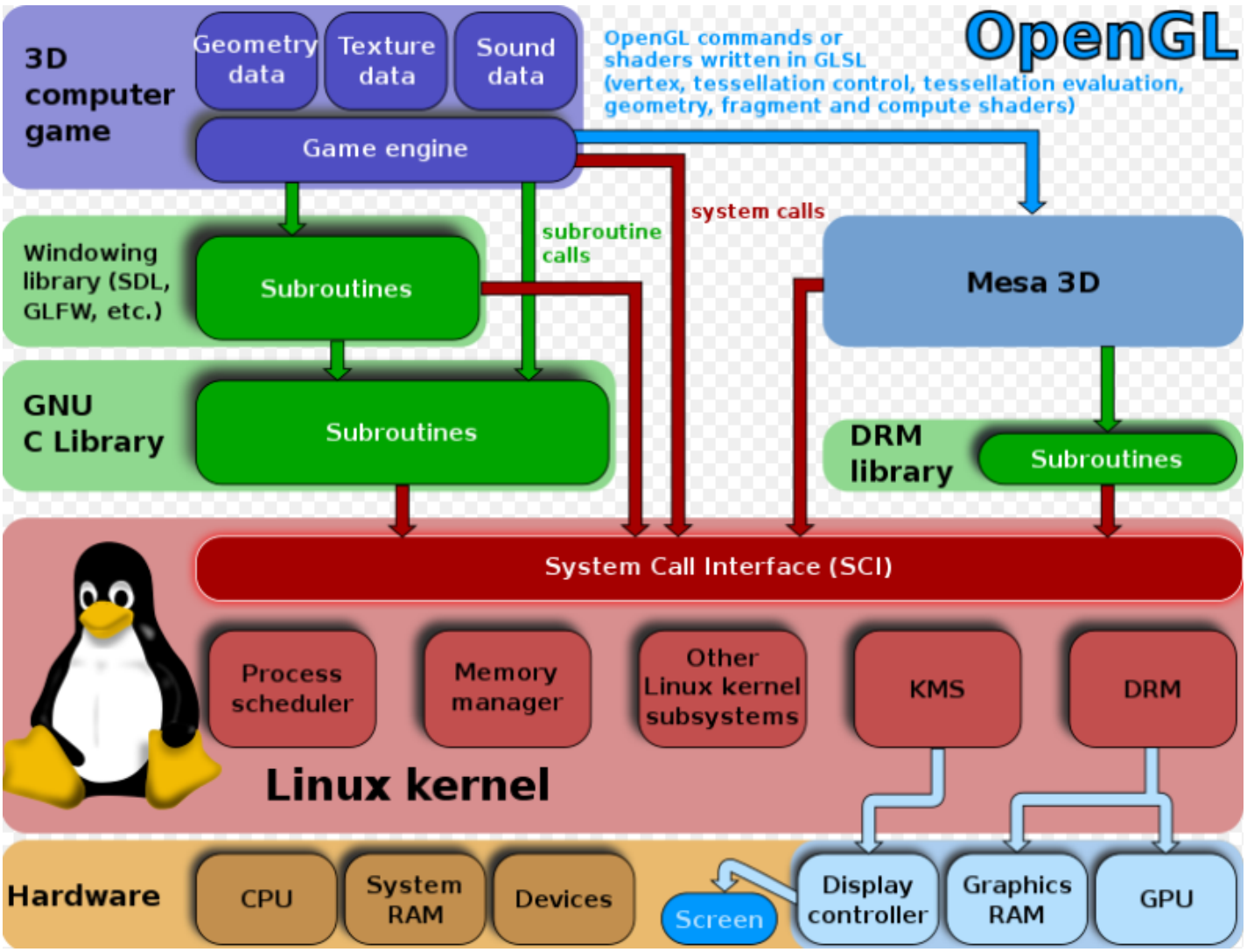


Figure 2.6 OpenGL Architecture

2.5.5 Vuforia

Vuforia is an augmented reality (AR) software development platform that allows developers to create AR applications for a variety of platforms, including mobile devices, tablets, and smart glasses. It provides a comprehensive set of tools, APIs (Application Programming Interfaces), and computer vision capabilities to enable the development of marker-based and markerless AR experiences.

Key features and components of Vuforia include:

1. Image Target Recognition: Vuforia supports marker-based AR by allowing developers to define and recognize specific images or objects as targets. When the device's camera detects these predefined targets, virtual content is overlaid on top of them.

2. Object Recognition: Vuforia also offers the ability to recognize and track 3D objects in the real world. By creating a 3D model of an object and training Vuforia with it, developers can enable AR experiences that interact with the recognized objects.

3. Ground Plane Detection: Vuforia includes functionality for detecting and tracking horizontal surfaces, such as floors or tables. This allows virtual objects to be placed accurately on the detected surfaces, enhancing the realism of the AR experience.

4. Smart Terrain: Vuforia provides a feature called Smart Terrain, which allows developers to create interactive AR experiences by recognizing and augmenting physical objects and surfaces.

5. Extended Tracking: Vuforia supports extended tracking, which means that once a target or surface is detected, the AR experience can persist even if the target or device moves out of the camera's view. This provides a more seamless and continuous AR experience.

6. Unity and Other Platform Integration: Vuforia seamlessly integrates with popular game engines, including Unity, allowing developers to leverage their existing skills and workflows. It also supports native development for iOS, Android, and other platforms.

Vuforia is widely used in various industries for creating AR applications, including gaming, education, retail, marketing, and industrial applications. It provides a powerful and accessible platform for developers to build immersive and interactive AR experiences on a wide range of devices.

2.5.6 Sceneview and Sceneform

Sceneform and SceneView are two related components used for creating augmented reality (AR) experiences in Android applications.

2.5.6.1 Sceneform

Sceneform is a 3D framework provided by Google for Android developers to easily create AR applications. It simplifies the process of building AR experiences by providing a higher-level API and tools for rendering 3D models, handling lighting and materials, and performing hit testing and interactions with virtual objects.

Key features of Sceneform include:

- High-Level API: Sceneform abstracts the complexities of low-level 3D graphics programming, allowing developers to focus on creating AR experiences without needing extensive knowledge of OpenGL or other graphics APIs.

- 3D Rendering: Sceneform provides a rendering engine that supports loading and displaying 3D models, applying materials and textures, handling animations, and rendering lighting effects.

- ARCore Integration: Sceneform seamlessly integrates with ARCore, Google's AR platform, enabling developers to easily place and manipulate virtual objects in the real world using ARCore's tracking and environmental understanding capabilities.

2.5.6.2 SceneView

SceneView is a component of the Sceneform library that provides a high-level view for rendering AR scenes. It extends the Android View class and handles rendering the AR scene, managing the camera, and updating the position and orientation of virtual objects.

Key features of SceneView include:

- AR Scene Rendering: SceneView renders the camera view with an overlay of virtual objects, allowing users to see the real world and the augmented content together.

- Interaction Handling: SceneView handles user interactions, such as gestures and touch events, for manipulating virtual objects within the AR scene.

- Scene Graph Management: SceneView manages the scene graph, which represents the hierarchy of virtual objects, their transformations, and relationships within the AR scene.

Sceneform and SceneView together provide a simplified and streamlined development experience for creating AR applications on Android. They leverage the power of ARCore and abstract many of the complexities of 3D graphics programming, allowing developers to focus on designing and implementing engaging AR experiences.

2.6 Flutter

“Flutter transforms the app development process. Build, test, and deploy beautiful mobile, web, desktop, and embedded apps from a single codebase.” This is what the Flutter team defines their framework, hence in a more detailed version we can define flutter as an open-source UI software development kit created by Google. It is used to develop cross-platform applications from a single codebase for any **web browser, Fuchsia, Android, iOS, Linux, macOS, and Windows.** First described in 2015, Flutter was released in May 2017.

2.6.1 Framework architecture:

Flutter is designed as an extensible, layered system. It exists as a series of independent libraries that each depend on the underlying layer. No layer has privileged access to the layer below, and every part of the framework level is designed to be optional and replaceable.

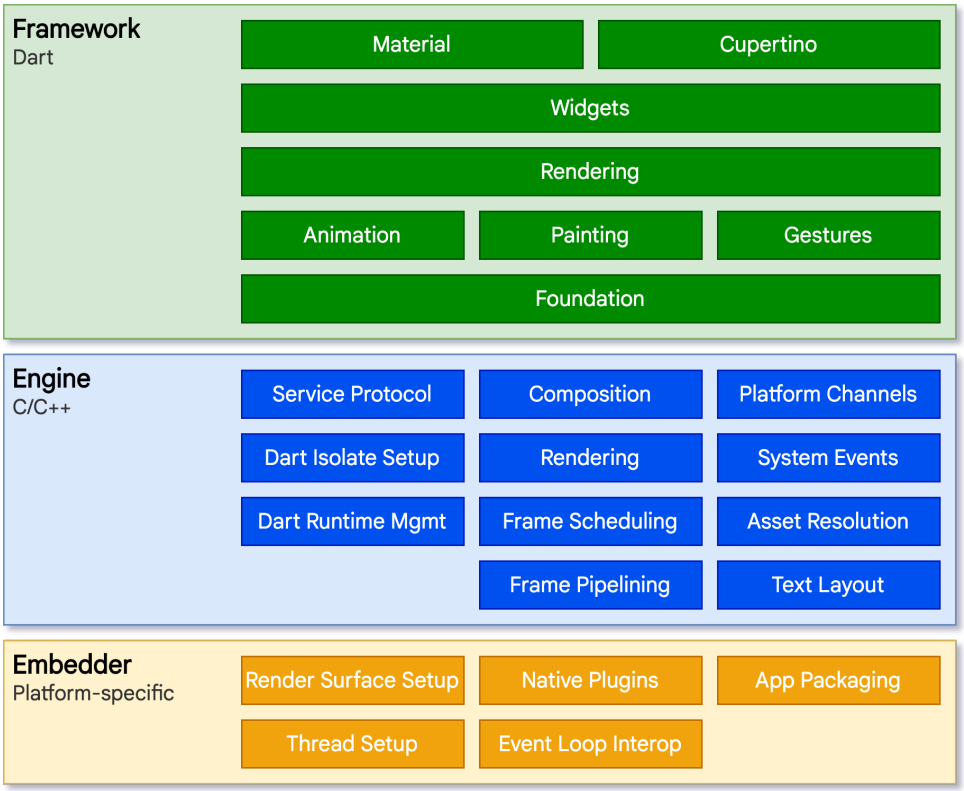


Figure 2.7 Flutter framework architecture from Flutter documentation

To the underlying operating system, Flutter applications are packaged in the same way as any other native application. A platform-specific embedder provides an entrypoint; coordinates with the underlying operating system for access to services like rendering surfaces, accessibility, and input; and manages the message event loop. The embedder is written in a language that is appropriate for the platform: currently **Java and C++ for Android, Objective-C/Objective-C++ for iOS and macOS, and C++ for Windows and Linux**. Using the embedder, Flutter code can be integrated into an existing application as a module, or the code may be the entire content of the application. Flutter includes a number of embedders for common target platforms, but other embedders also exist.

At the core of **Flutter is the Flutter engine, which is mostly written in C++** and supports the primitives necessary to support all Flutter applications. The engine is responsible for rasterizing composited scenes whenever a new frame needs to be painted. It provides the low-level implementation of Flutter's core API, including graphics (through Impeller on iOS and coming to Android, and Skia on other platforms) text layout, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile toolchain.

The engine is exposed to the Flutter framework through dart:ui, which wraps the underlying C++ code in Dart classes. This library exposes the lowest-level primitives, such as classes for driving input, graphics, and text rendering subsystems.

Typically, developers interact with Flutter through the Flutter framework, which provides a modern, reactive framework written in the Dart language. It includes a rich set of platform, layout, and foundational libraries, composed of a series of layers. Working from the bottom to the top, we have:

Basic foundational classes, and building block services such as animation, painting, and gestures that offer commonly used abstractions over the underlying foundation.

The rendering layer provides an abstraction for dealing with layout. With this layer, you can build a tree of renderable objects. You can manipulate these objects dynamically, with the tree automatically updating the layout to reflect your changes.
The widgets layer is a composition abstraction. Each render object in the rendering layer has a corresponding class in the widgets layer. In addition, the widgets layer allows you to define combinations of classes that you can reuse. This is the layer at which the reactive programming model is introduced.
The Material and Cupertino libraries offer comprehensive sets of controls that use the widget layer's composition primitives to implement the Material or iOS design languages.
The Flutter framework is relatively small; many higher-level features that developers might use are implemented as packages, including platform plugins like camera and webview, as well as platform-agnostic features like characters, http, and animations that build upon the core Dart and Flutter libraries. Some of these packages come from the broader ecosystem, covering services like in-app payments, Apple authentication, and animations.

The rest of this overview broadly navigates down the layers, starting with the reactive paradigm of UI development. Then, we describe how widgets are composed together and converted into objects that can be rendered as part of an application. We describe how Flutter interoperates with other code at a platform level, before giving a brief summary of how Flutter's web support differs from other targets.

2.6.2 Flutter web

Flutter is a powerful framework that allows developers to build web applications with ease. With its reactive and component-based architecture, Flutter enables the creation of visually stunning and performant web interfaces. By leveraging a single codebase, developers can build applications that work seamlessly across multiple platforms, including web browsers.

One of the key advantages of using Flutter for web application development is its hot-reload feature. This feature enables developers to see the changes they make to the code in real-time, without the need for time-consuming compilation and deployment processes. This greatly speeds up the development cycle and enhances productivity. Additionally, Flutter offers a rich set of pre-built UI components and widgets, allowing developers to quickly create beautiful and responsive user interfaces. The framework also provides access to powerful animations and transitions, enabling the creation of engaging and interactive web experiences.

Moreover, Flutter's web support provides a consistent and cohesive experience across different platforms. Whether it's a mobile device or a desktop browser, Flutter ensures that the web application looks and behaves consistently, thanks to its pixel-perfect rendering. Furthermore, Flutter allows developers to access native device features and APIs, giving them the flexibility to integrate platform-specific functionalities into their web applications.

In conclusion, Flutter offers a robust and efficient framework for building web applications. Its hot-reload feature, extensive widget library, and platform adaptability make it an excellent choice for developers looking to create high-quality web experiences. By leveraging Flutter's capabilities, developers can streamline the development process, achieve code reuse, and deliver visually appealing and performant web applications to a wide range of users.

2.7 Android Application Integration between Java, Kotlin, and Flutter

An Android application can be developed by an Android framework built by Java, an Android framework built by Kotlin, or Flutter cross-platform framework developed by Dart. As we can see all of these frameworks are built with different programming languages but they produce the same android APK, the question is can these frameworks be integrated together?

Yes, an Android application shares some files between these frameworks which can be considered as a shared environment. These files are Android manifest, gradle, and resources file.

The manifest file provides a way of communication between different platform languages as a class in Java can find a class in Kotlin defined in manifest and send an intent.

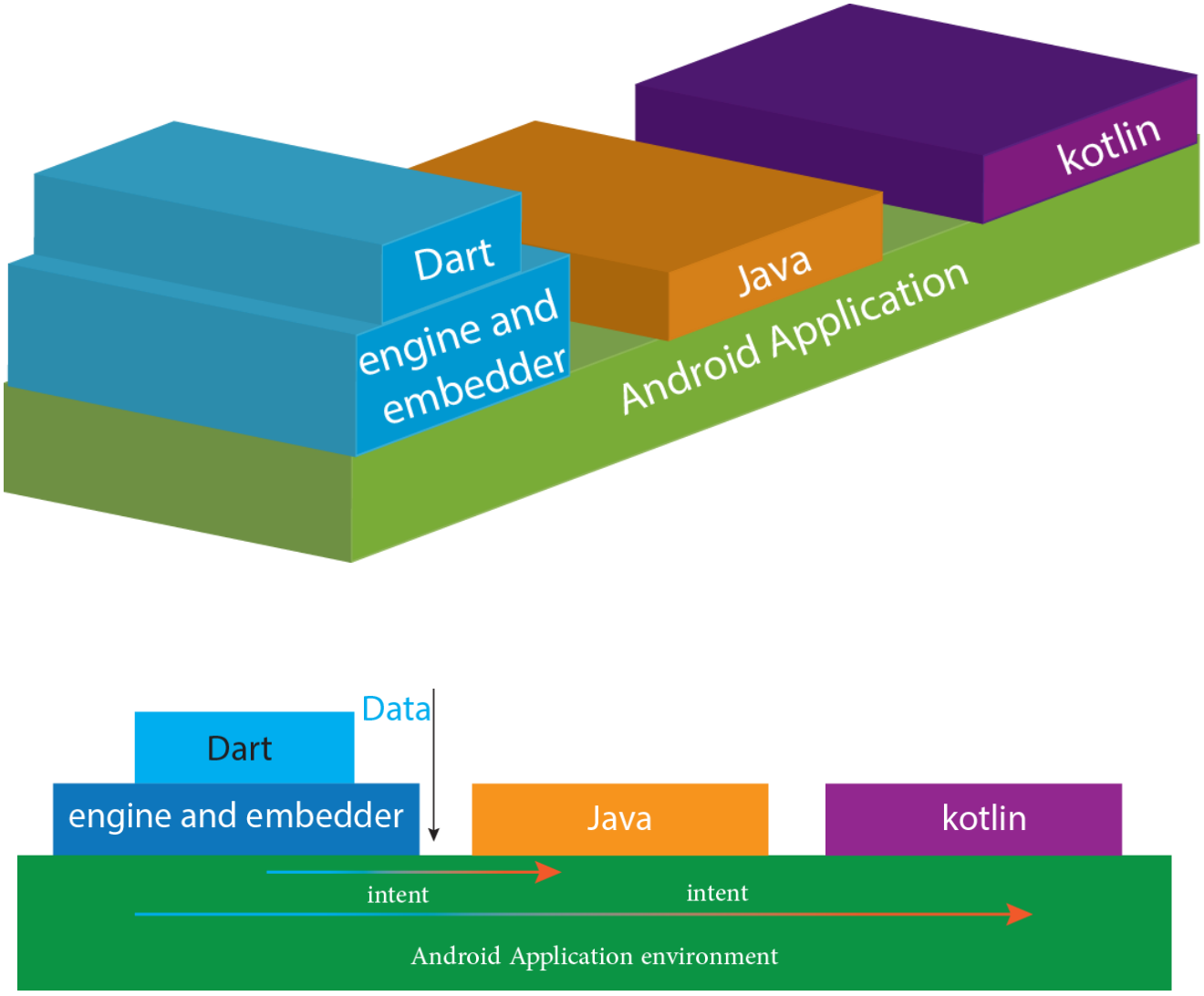


Figure 2.8 Communication between different components of android application

An object written in Dart will use the native method by engine and embedder that will call a native function in java or kotlin and we can navigate to another Activity object with the means of intents.

2.8 Machine learning/ AI components:

2.8.1 Indoor localization problem formulation:

The process of determining the location of a device within an indoor environment is difficult for GPS (Global Positioning System), which is inconsistent and often error prone due to signal attenuation. Several techniques have been developed as solutions to such problems.

2.8.2 Wi-Fi-based Localization:

Wi-Fi networks are widely available in indoor environments, and their signal strength can be used for localization. By measuring the signal strength from different Wi-Fi access points, a device can estimate its position This technique requires a pre-built signal strength map of the area . By measuring the signal strength and triangulating the distances to multiple access points, the device can determine its location.

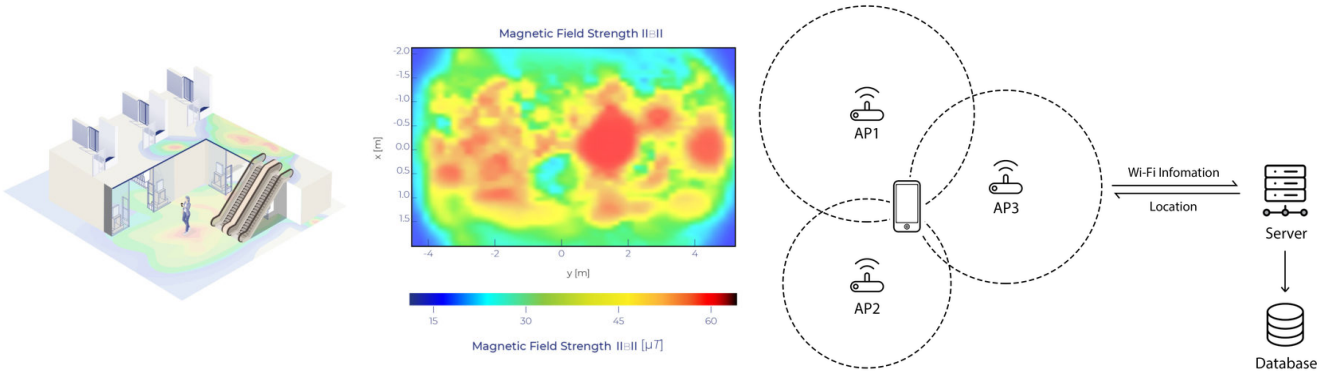


Figure 2.9 localization methods

The most common ways to locate a device are

- **Triangulation**: where the connection to each access point, RSSI (Received Signal Strength Indicator), is input into a function to calculate the distance to that access point, repeat for all access points then get the intersection of these distances

- **Fingerprinting**: where all RSSIs are compared to an already known fingerprint, the closest match of the current RSSI and the saved RSSI values is the result

These techniques have some disadvantages as they are based on algorithms that don't factor in noise, multipath, blocking, and change in access point position; this makes them prone to errors while predicting the location of the device, therefore machine learning techniques can provide better results as they can extract information without the restrictions of normal algorithms.

2.8.3 Machine learning techniques:

- K-Nearest Neighbour (Also known as KNN or k-NN):
  is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

Fig 2.10: K-nearest neighbor

It groups data points based on a distance metric (ex. Euclidean, or Manhattan). These distance metrics help to form decision boundaries, which partition query points into different regions.

Advantages of KNN:

- Easy adaptation
- Fast processing
- Few hyperparameters (K value and Distance metric)

Disadvantages of KNN:

- Not easily scalable (lazy algorithm)
- Curse of dimensionality
- Prone to overfitting
- Random Forest:

  The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees. Feature randomness, also known as feature bagging generates a random subset of features, which ensures low correlation among decision trees. This is a key difference between decision trees and random forests. While decision trees consider all the possible feature splits, random forests only select a subset of those features.

  Random forest algorithms have three main hyperparameters, which need to be set before training. These include node size, the number of trees, and the number of features sampled. From there, the random forest classifier can be used to solve regression or classification problems.

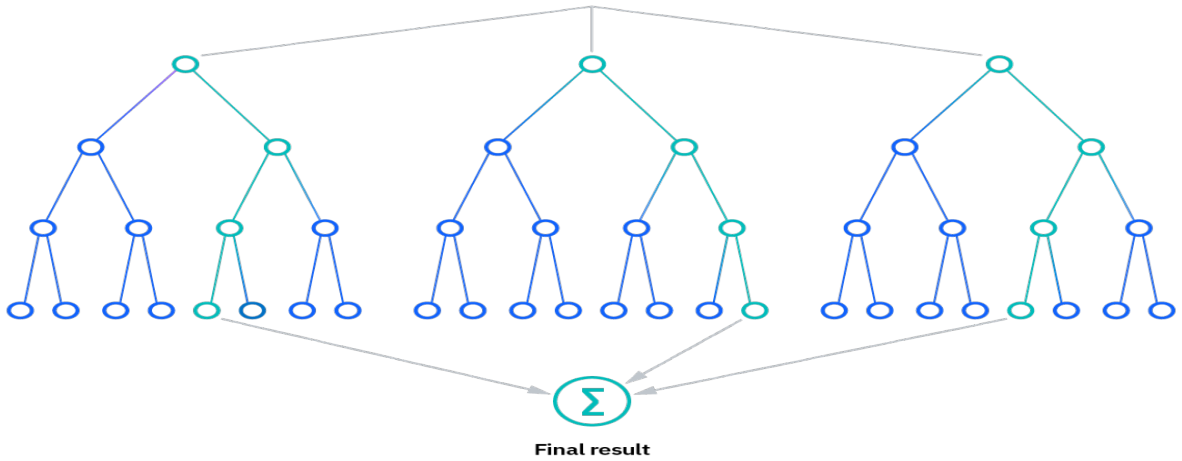


Figure 2.11 Random forest

- Artificial Neural Networks (ANN)/ Multilayer Perceptron (MLP):
  Neural networks, also known as artificial neural networks (ANNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

  Artificial neural networks (ANNs) are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

  A multilayer perceptron (MLP) is a fully connected class of feedforward artificial neural networks (ANN). An MLP consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a chain rule-based supervised learning technique called backpropagation or reverse mode of automatic differentiation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

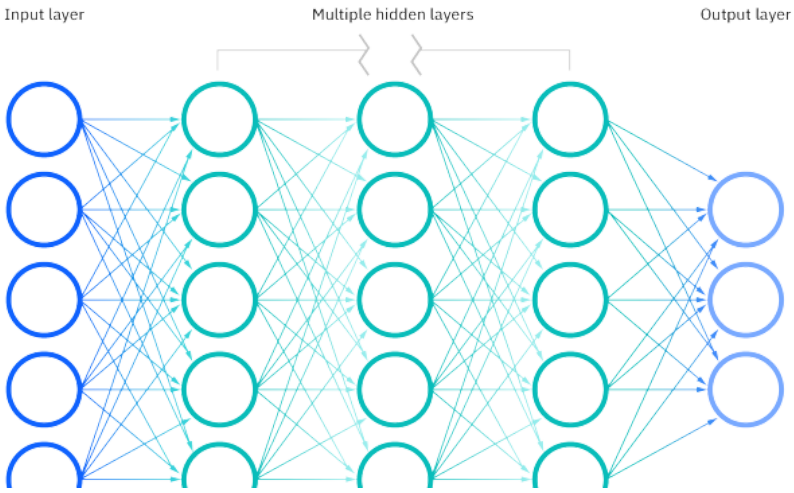


Figure 2.12 Neural network

How do neural networks work?

Think of each individual node as its own linear regression model, composed of input data, weights, a bias (or threshold), and an output. The formula would look something like this:

$\sum w_i x_i + bias = w_1x_1 + w_2x_2 + w_3x_3 + bias$

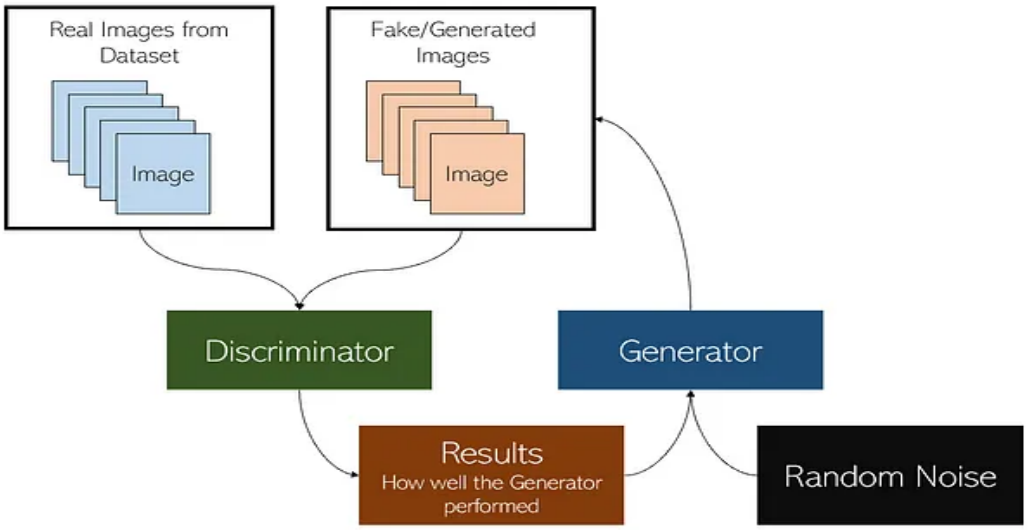
$output = f(x) = 1 \text{ if } \sum w_1x_1 + b >= 0; 0 \text{ if } \sum w_1x_1 + b < 0$

- Generative Adversarial Networks (GAN):
  GANs are generative models that focus on generating data that can't be distinguished from real data, this can be done by having two models that work against each other, generator and discriminator , one generates the data and another discriminates. Each model trains on getting better in their task.

  A generative adversarial network (GAN) has two parts:

  - **The generator** learns to generate plausible data. The generated instances become negative training examples for the discriminator.
  - **The discriminator** learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.

  Both generator and discriminator are Neural Networks.



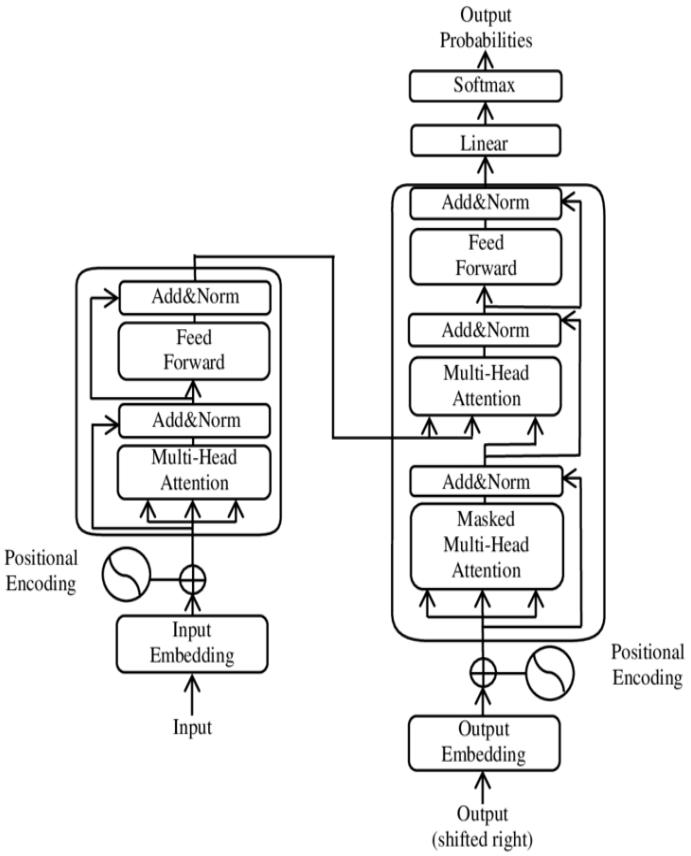
- Large Language Models (LLM):
  Is a computerized language model, embodied by an artificial neural network using an enormous amount of of "parameters" ("neurons" in its layers with up to tens of millions to billions "weights" between them), that are (pre-)trained on many GPUs in relatively short time due to massive parallel processing of vast amounts of unlabeled texts containing up to trillions of tokens (parts of words) provided by Wikipedia Corpus and Common Crawl, using self-supervised learning or semi-supervised learning, resulting in a tokenized vocabulary with a probability distribution. LLMs can be upgraded by using additional GPUs to (pre-)train the model with even more parameters on even vast amounts of unlabeled texts.

  The invention of an algorithm known as transformer, either unidirectional (such as used by GPT models) or bidirectional (such as used by BERT model), allows for such massively parallel processing.

  All transformers have the same primary components:

  - Tokenizers, which convert text into machine-readable symbols known as tokens
  - Embedding layers, which convert the machine-readable symbols into semantically meaningful representations
  - Transformer layers, which carry out the reasoning capabilities of the models

Transformer layers come in two types known as encoders and decoders. While the original transformers paper was comprised of both encoder layers and decoder layers, subsequent work has also explored encoder-only architectures (BERT) and decoder-only architectures (GPT) as well. While all three have their benefits and uses, decoder-only models are the dominant form at very large scales due to being substantially more efficient to train at scale.



These LLMs are very flexible and can do many tasks of which is

- Text generation. The ability to generate text on any topic that the LLM has been trained on is a primary use case.
- Translation. For LLMs trained in multiple languages, the ability to translate from one language to another is a common feature.
- Content summary. Summarizing blocks or multiple pages of text is a useful function of LLMs.
- Rewriting content. Rewriting a section of text is another capability.
- Classification and categorization. An LLM is able to classify and categorize content.
- Sentiment analysis. Most LLMs can be used for sentiment analysis to help users to better understand the intent of a piece of content or a particular response.
- Conversational AI and chatbots. LLMs can enable a conversation with a user in a way that is typically more natural than older generations of AI technologies.

CHAPTER THREE
SYSTEM COMPONENTS

The developed application utilizes the GPS capabilities of the user's device to provide location-based augmented reality (AR) experiences. The application determines the user's precise location by leveraging the device's current GPS coordinates. The system employs geofencing techniques to define specific virtual boundaries around designated areas of interest. Upon entering a predefined geofenced location, the application initiates the retrieval and rendering of AR content stored in Firebase Storage. This content consists of 3D figures that are seamlessly superimposed onto the real-world view seen through the device's camera. The AR figures are accurately positioned and aligned with the physical environment, providing users with an immersive and interactive experience. The integration of GPS-based geofencing and AR content delivery enhances user engagement and exploration of the designated areas, effectively blending the virtual and real worlds.

3.1 Application

3.1.1 Browsing Museums

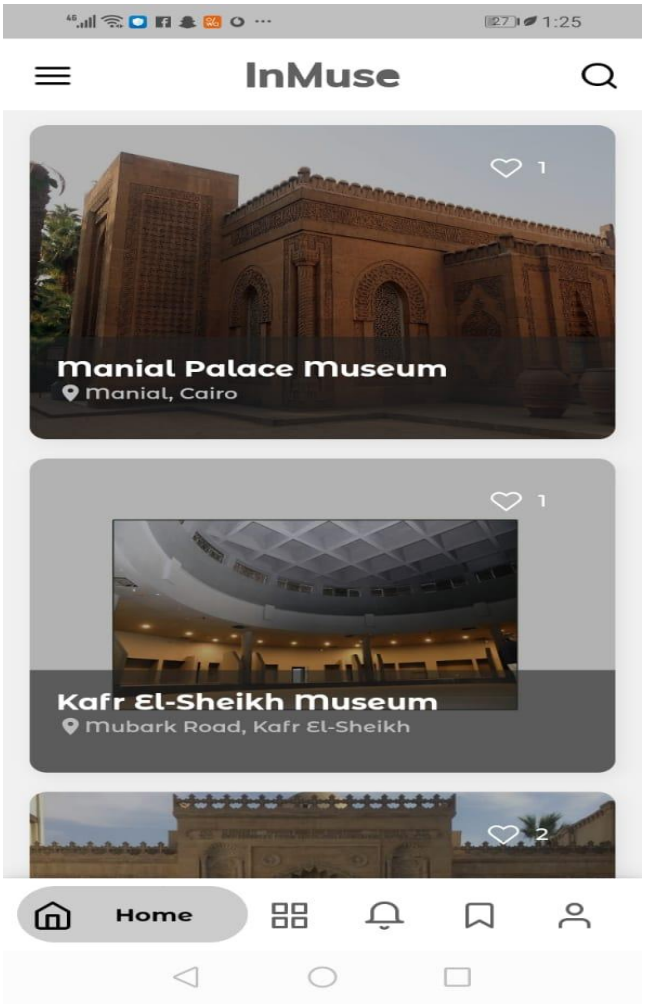


Fig 3.1 Home screen

3.1.2 Nearby Museums

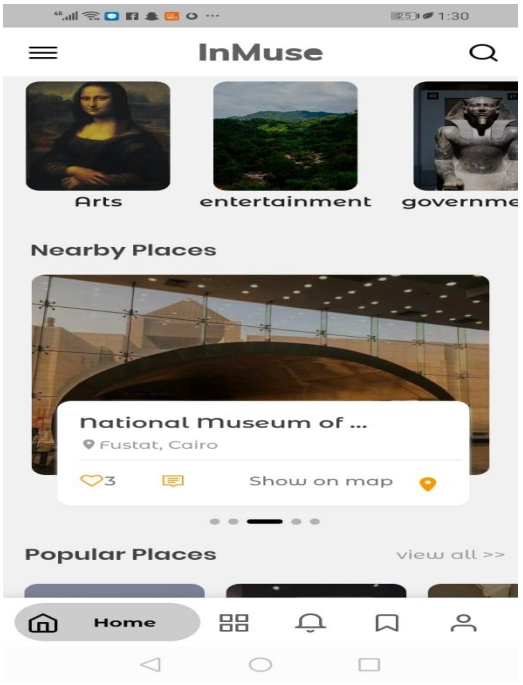
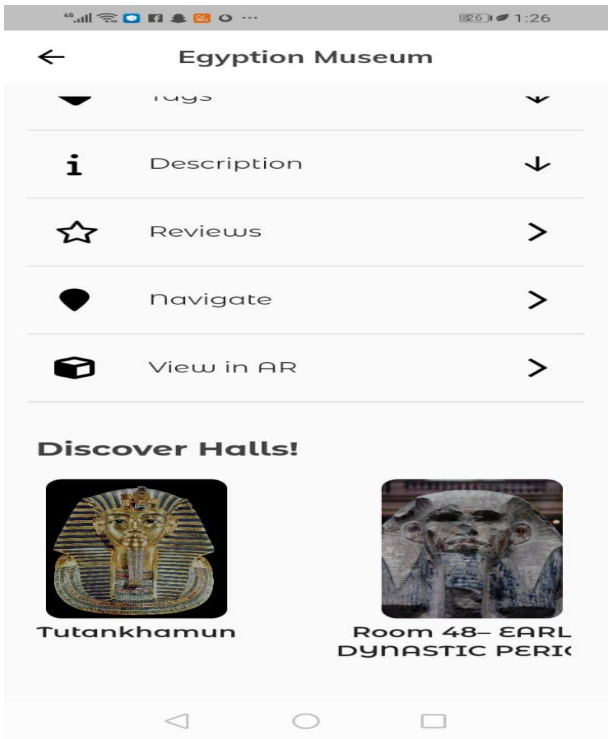


Fig 3.2 Nearby museums

3.1.3 Sections



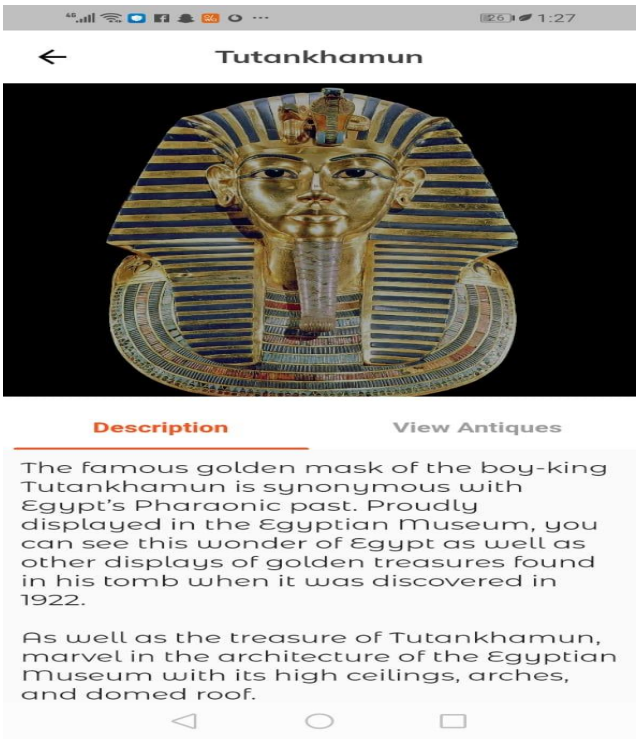
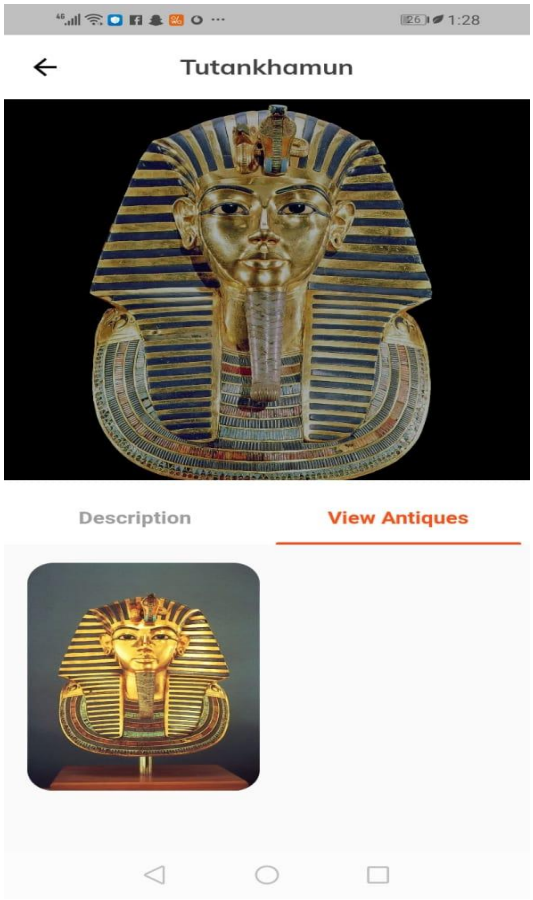


Fig 3.3 and Fig 3.4 Sections screen

3.1.4 Antiques



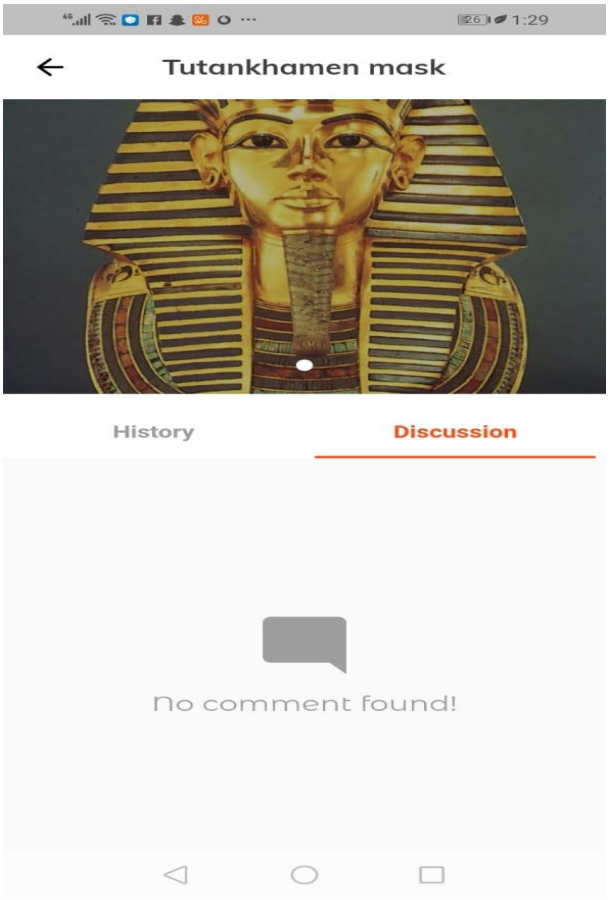


Fig 3.5 and Fig 3.6 antiques screen

3.1.5 Saved Museums

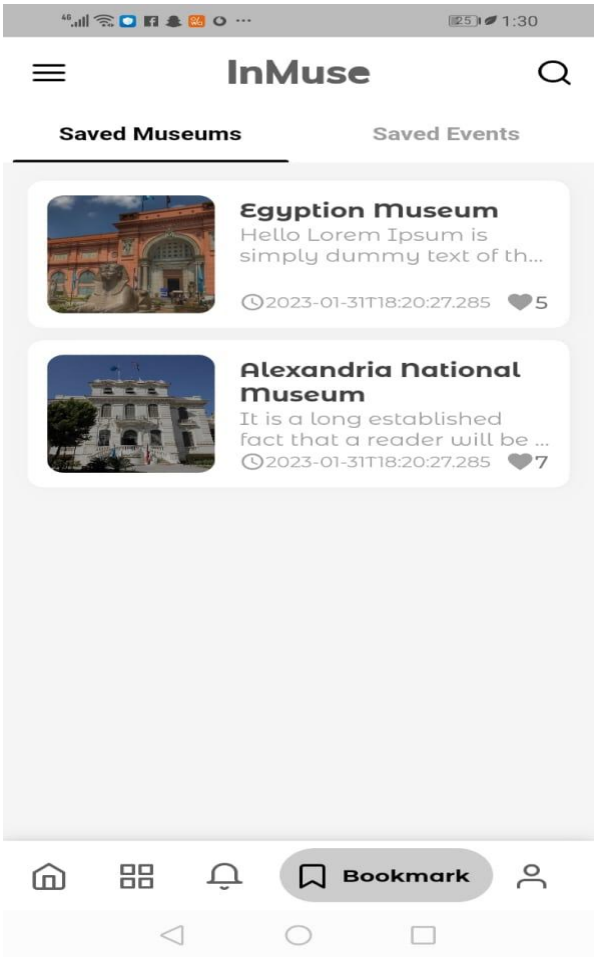


Fig 3.7 saved museums screen

3.1.6 Events

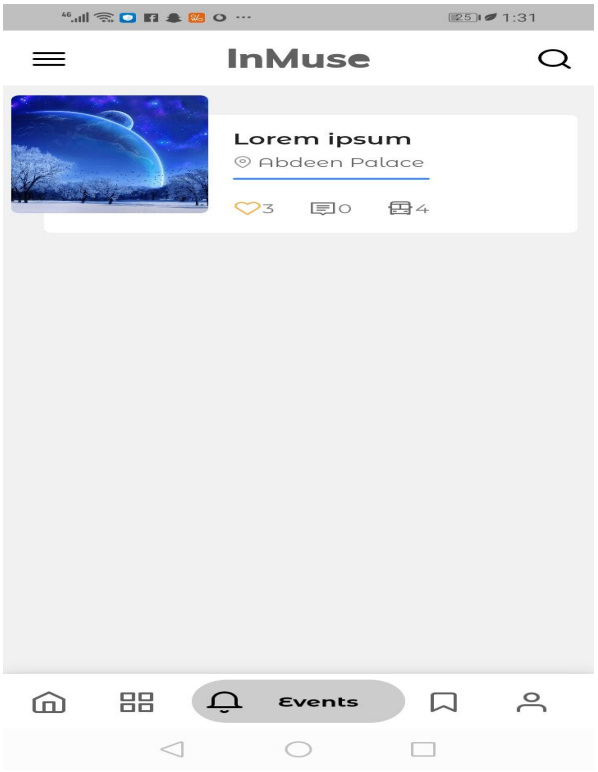


Fig 3.8 event screen

3.2 Admin Panel

For a content-based application with content either created or updated by different users an admin is required to manage this content. It is not preferred for an admin to his operation directly against to database, but it is required for an admin to have a user-friendly application (either mobile, desktop, or web) to manage content and users hence we have an **admin panel.** Admin panel or maybe called Dashboard it is a web-based interface that allows authorized individuals, typically administrators or system operators, to manage and control various aspects of a software application, website, or system. It serves as a centralized hub where administrators can access and manipulate data, settings, and functionalities to oversee and maintain the system.

The admin panel provides a user-friendly interface that simplifies administrative tasks and enables efficient management of the system. It often includes features such as user management, content management, data analytics, configuration settings, security controls, and reporting tools.

The primary purpose of an admin panel is to empower **administrators with the ability to perform administrative tasks,** monitor system performance, make changes, and gather insights. It plays a crucial role in maintaining the functionality, security, and overall performance of an application or system, ensuring smooth operations and optimal user experiences.

Hence the term web application is used. We need to develop it, many frameworks are used to develop web applications but one of these frameworks is Flutter. As previously mentioned flutter can develop applications for any screen including web applications. An admin panel is developed by means of the Flutter framework, the developed framework is a responsive screen, portable, and easy to deploy.

Firebase hosting services are used to host the Flutter web admin panel application. In the following section screenshots are provided to illustrate the operation of the admin panel.

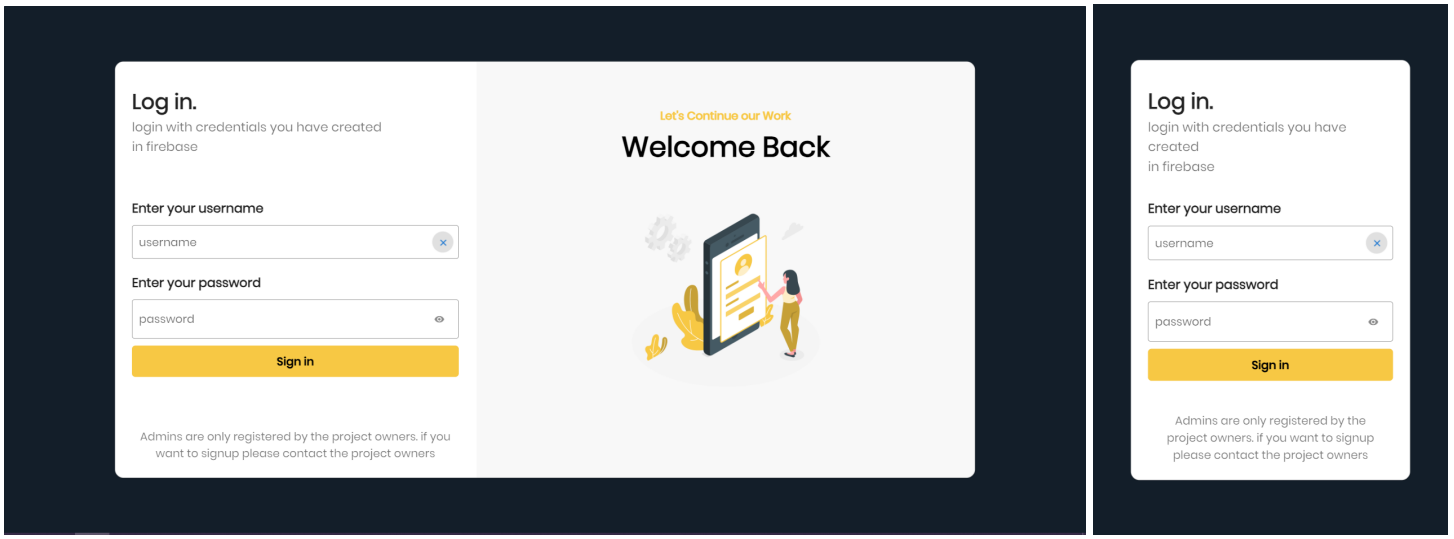


Fig 3.9 A responsive Sign in screen that can be displayed on the web and mobile

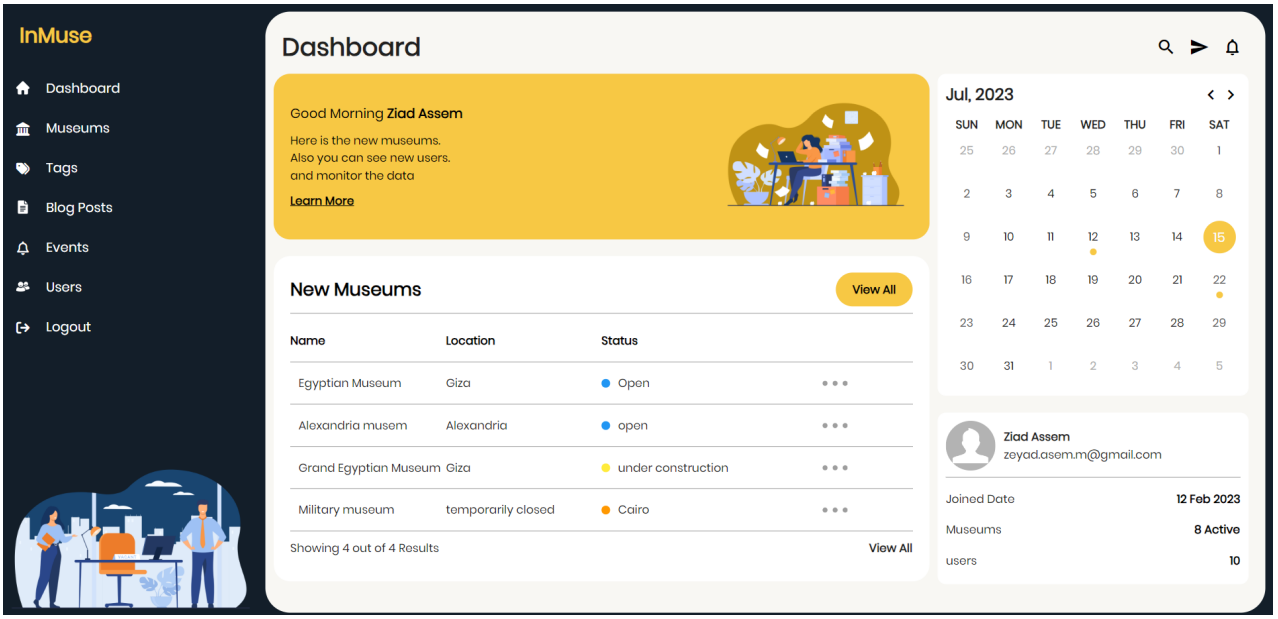


Fig 3.10 Home screen for showing brief information

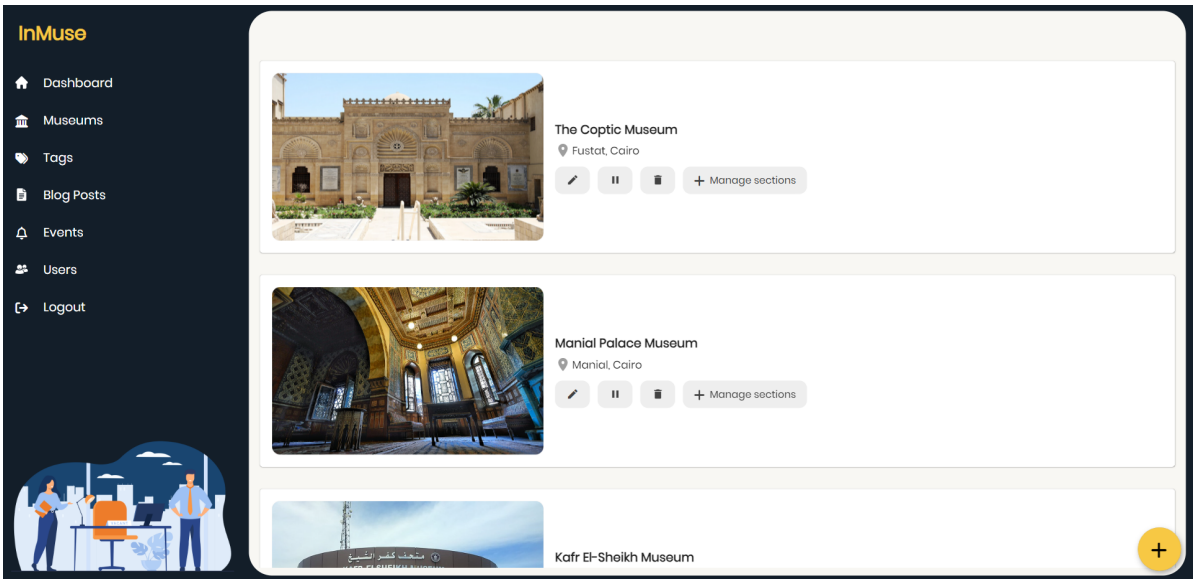


Fig 3.11 Museums screen, an admin can edit, create, delete, halt, and manage museums

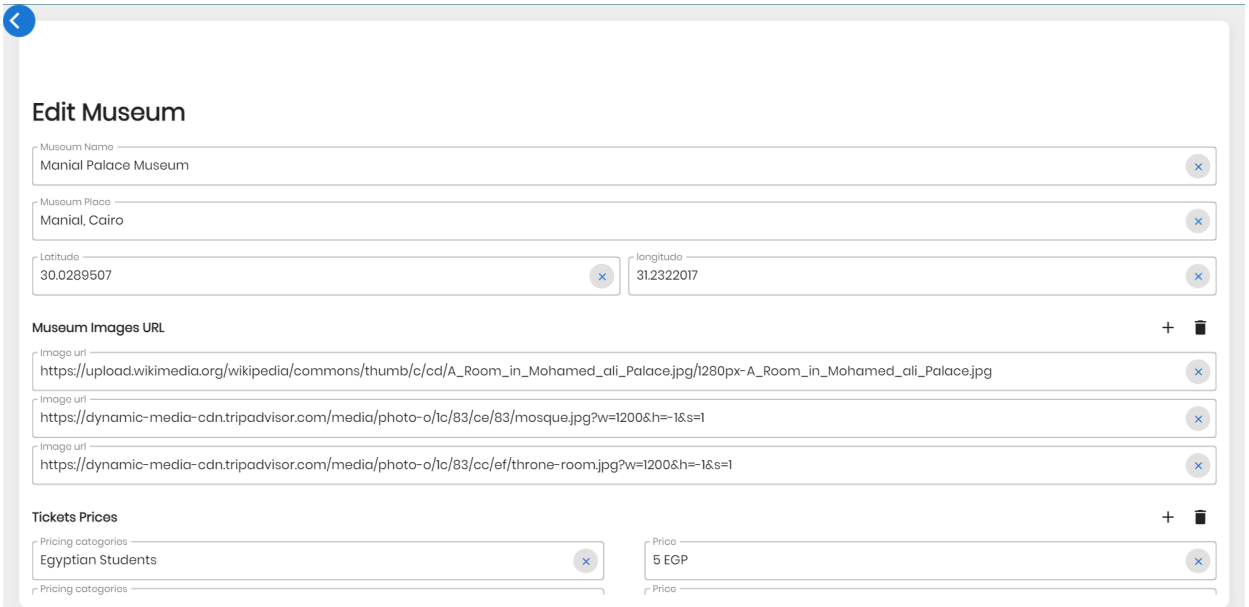


Fig 3.12 Museum create/edit screen; an admin can insert new data of museums

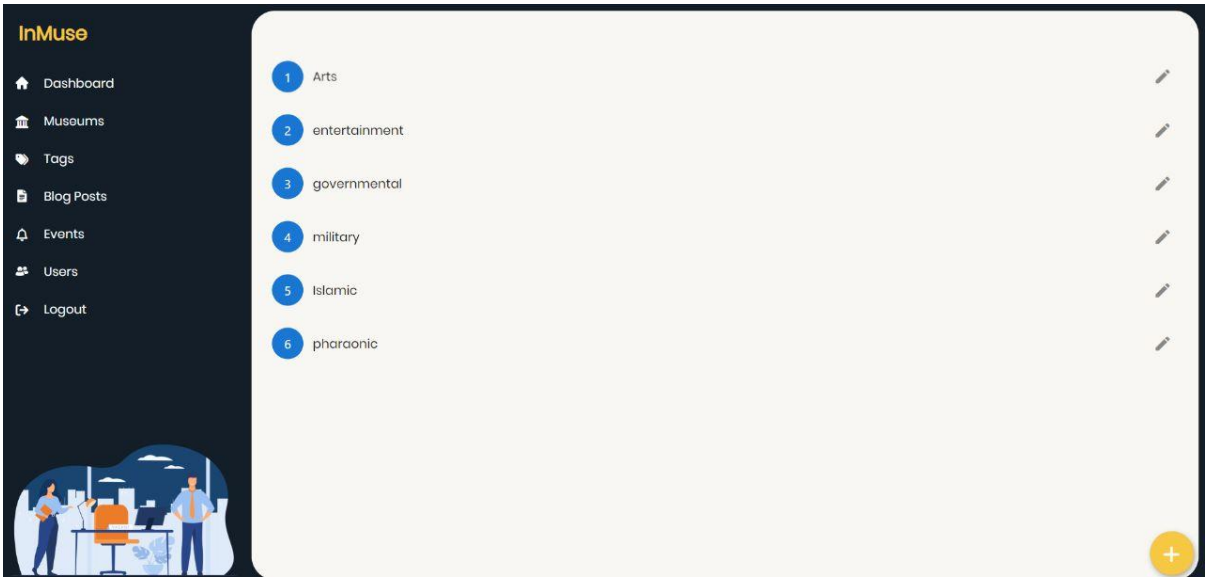


Fig 3.13 Tags Screen; For tasks related to recommendation systems we need to categorize museums by tags, so an admin can manage tags from this page

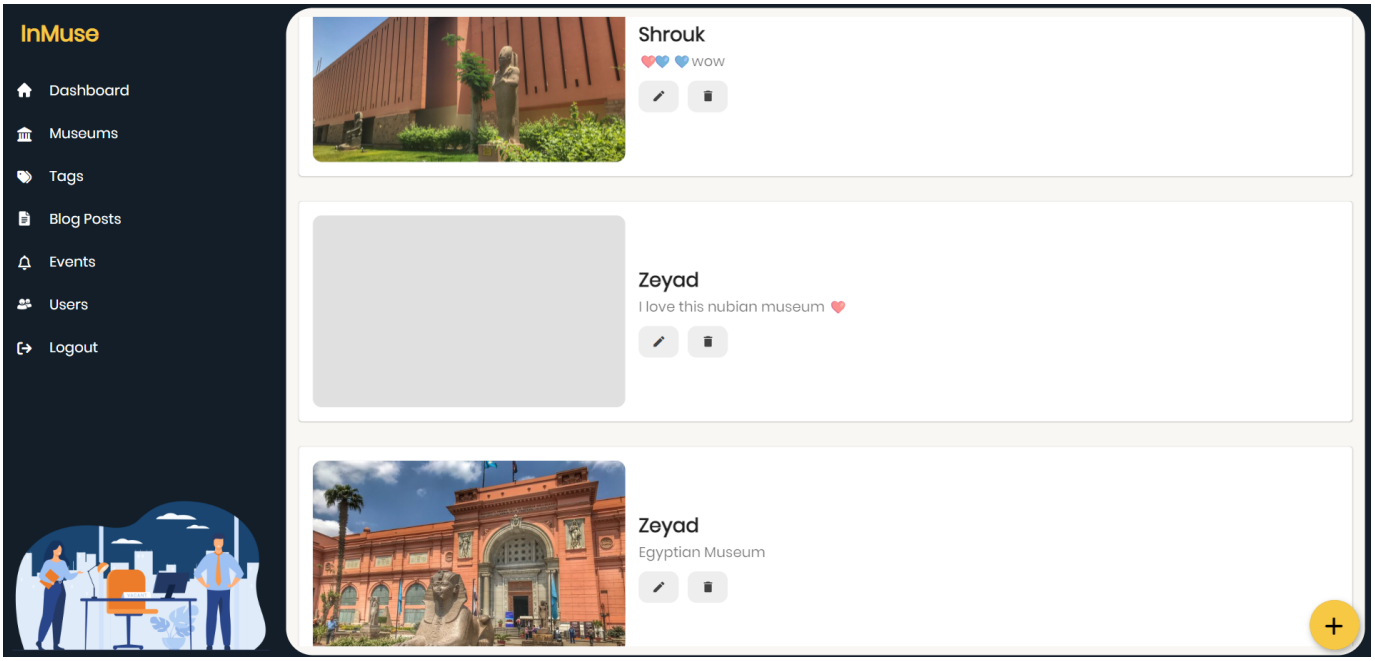


Figure 3.14 Posts create/edit screen; an admin can manage content

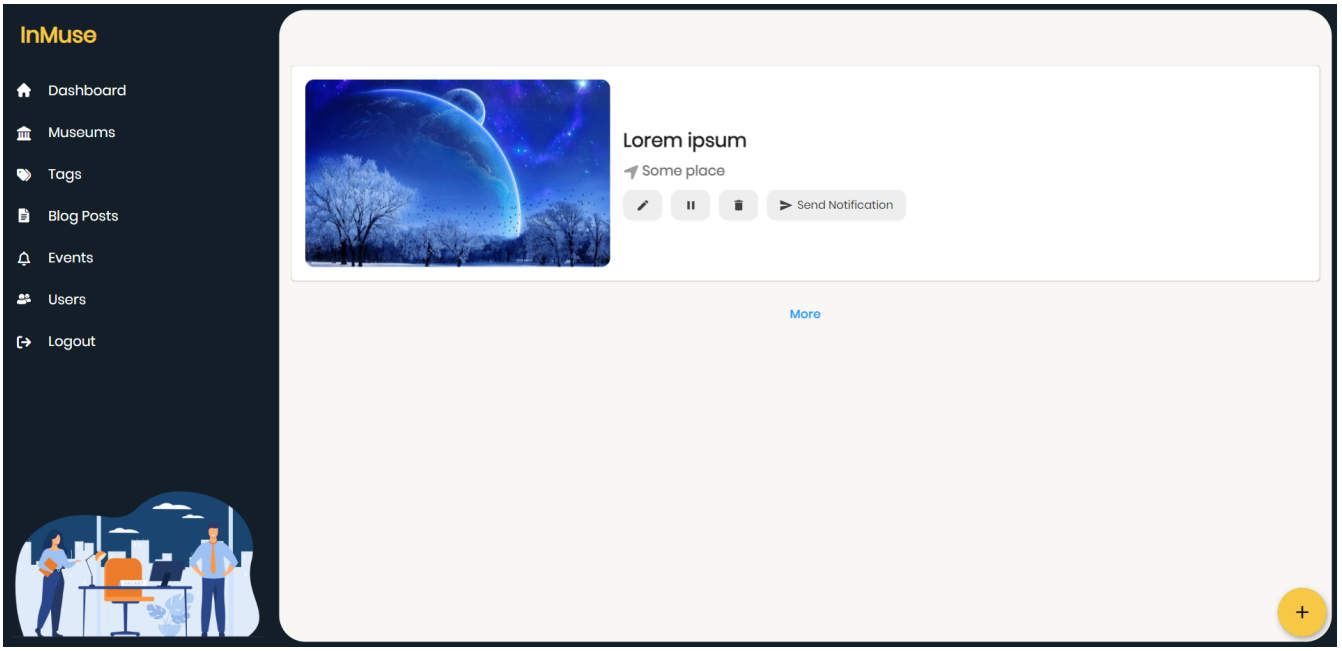


Figure 3.15 Events screen; an admin can publish new events and send notifications

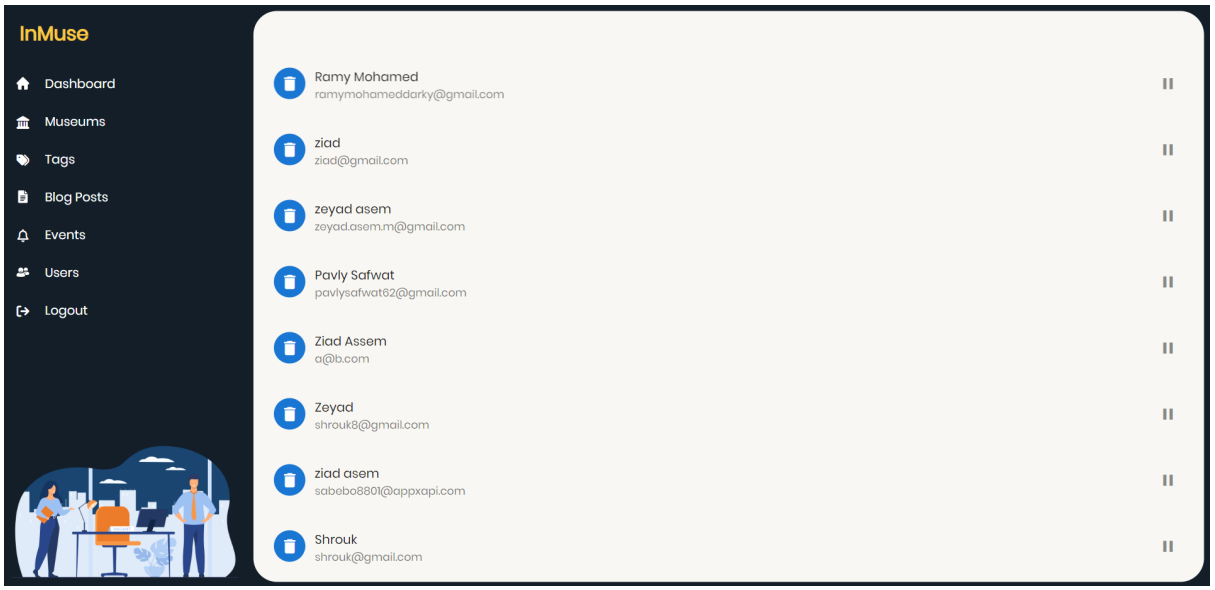


Figure 3.16 User screen; an admin can monitor and delete users

3.3 Firebase

As mentioned in chapter two, Firebase consists of many collections to handle and store all data needed in the application.

The collection contains documents with a random ID generated by Firebase or a unique ID to ensure that each document in the collection is unique and can be easily retrieved or updated using its ID.

3.3.1 Admins Collection

Admins collection contains documents for each admin. Each document has fields that describe each admin as email, id, join, name, password, URL and username.

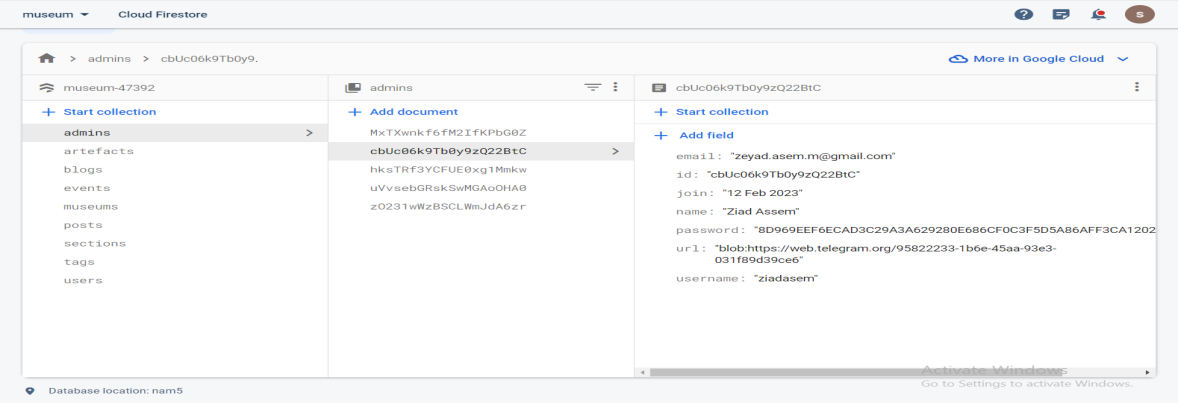


Fig 3.17 Admins collection.

3.3.2 Users Collection

Each document in the users collection contains fields that describe each user as uid, name , image, joining date, halt, bookmarked events, bookmarked places, loved events, and loved places.

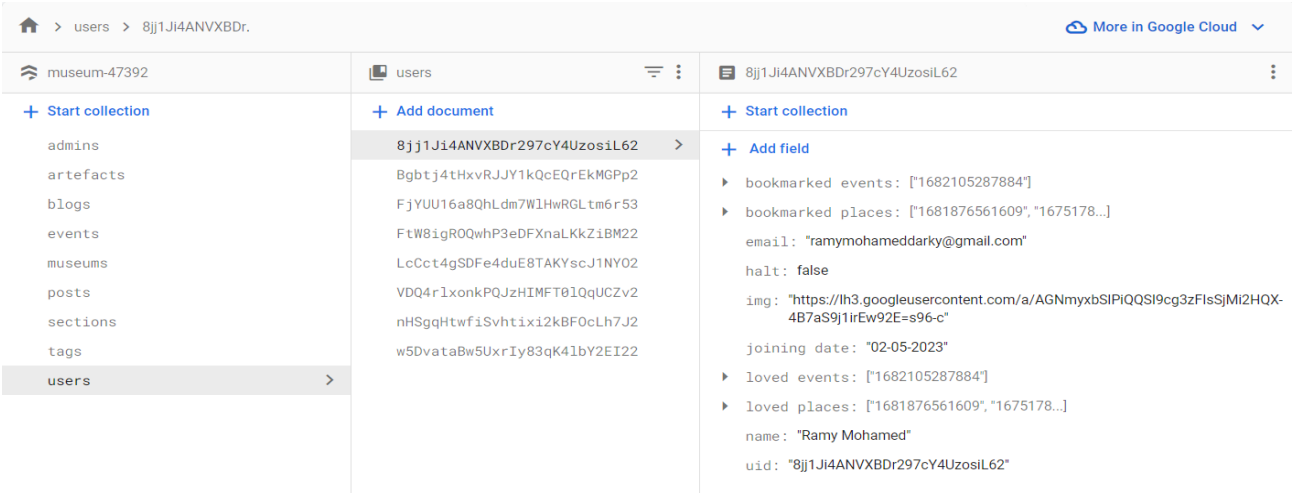


Fig 3.18 Users collection

3.3.3 Artifacts Collection

Artifacts collection contains documents. Each document has fields that describe each artifact in the museum as age, id, description, images, material, museum-id, name, and section-id.

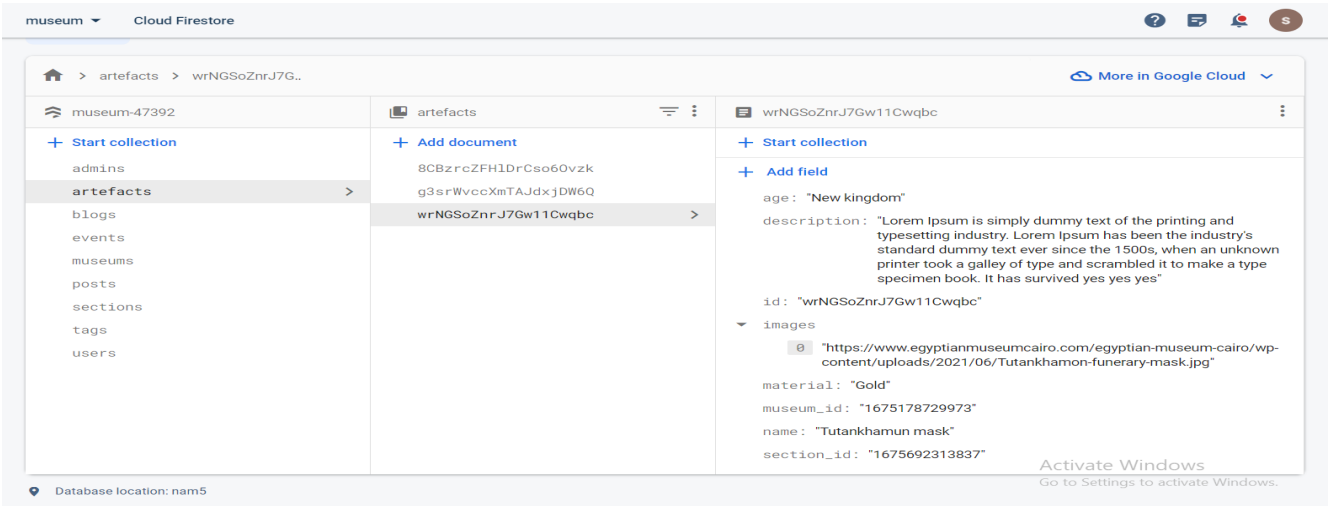


Fig 3.19 Artifacts collection

3.3.4 Blogs Collection

Each document in the Blog collection has fields that describe each blog as author, body, date, images, loves, name, place, and uid (user id).

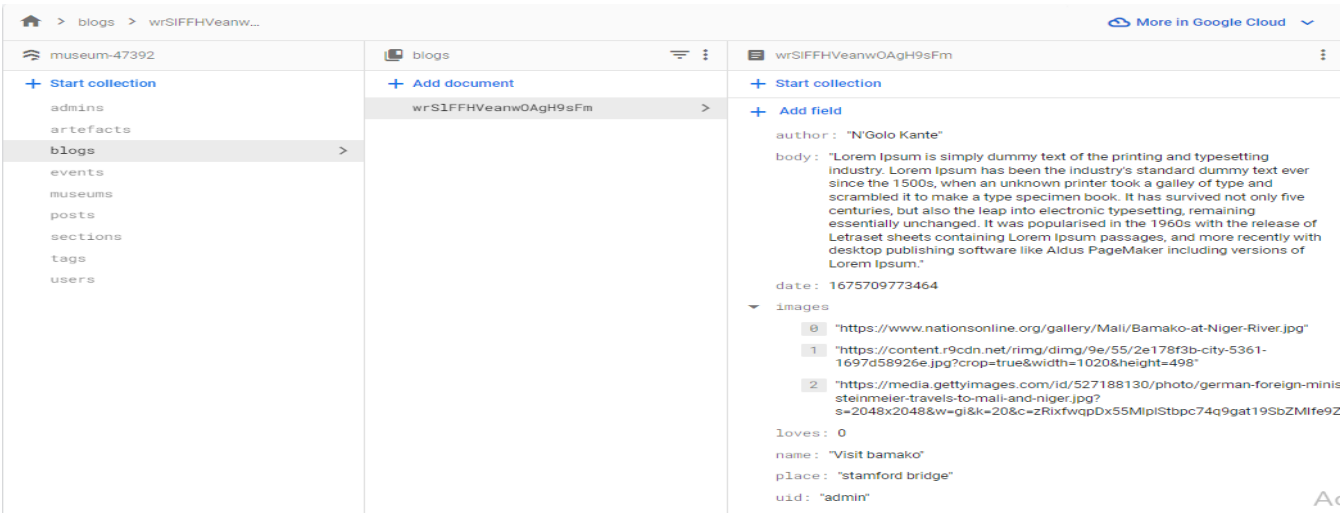


Fig 3.20 Blogs collection

3.3.5 Events Collection

Each document in the events collection contains fields to describe the event as body, comments, date, end date, id, images, museum-id, name, place, and start date. It also contains a collection to describe the comments for the event.

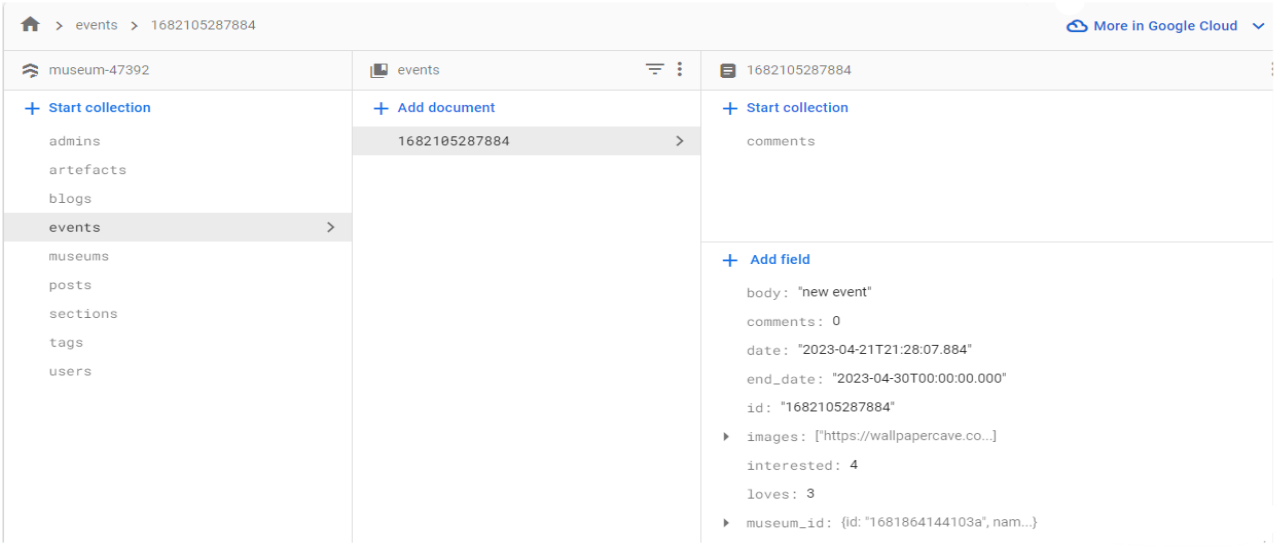


Fig 3.21 Events collection

3.3.6 Museums Collection

Each document in this collection contains fields to describe the museum as a city, date, description, id, images, lat (Latitude), lng (longitude), name, loves, tickets, and working hours. It also contains collections as comments and sections to describe the comments and sections of this museum.

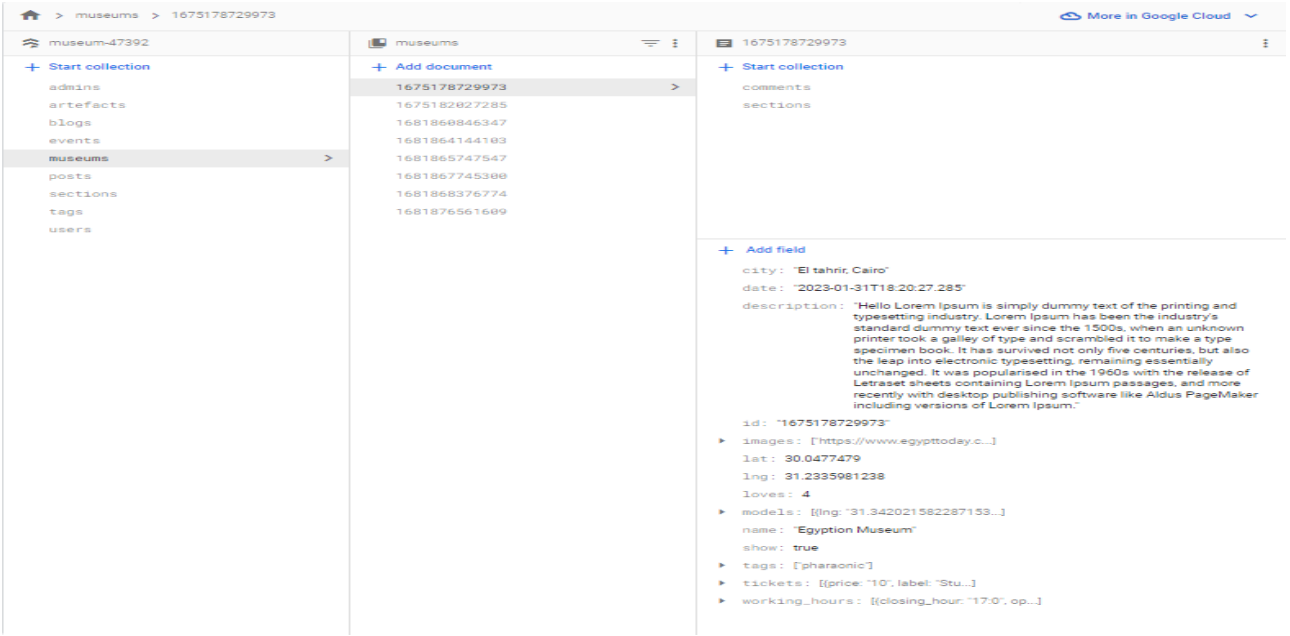


Fig 3.22 Museums collection

3.3.7 Sections Collection

Each document in the sections collection has fields that describe each section as date, description, id, image, museum-id, shown, and name.

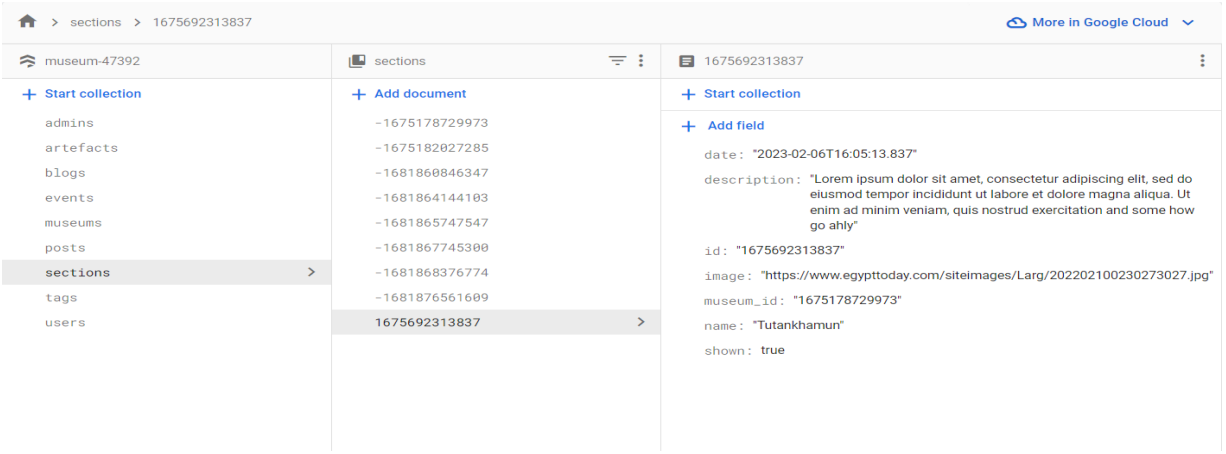


Fig 3.23 Sections collection

3.3.8 Tags Collection

Each document in the tags collection has fields that describe each tag as id, image, and name.

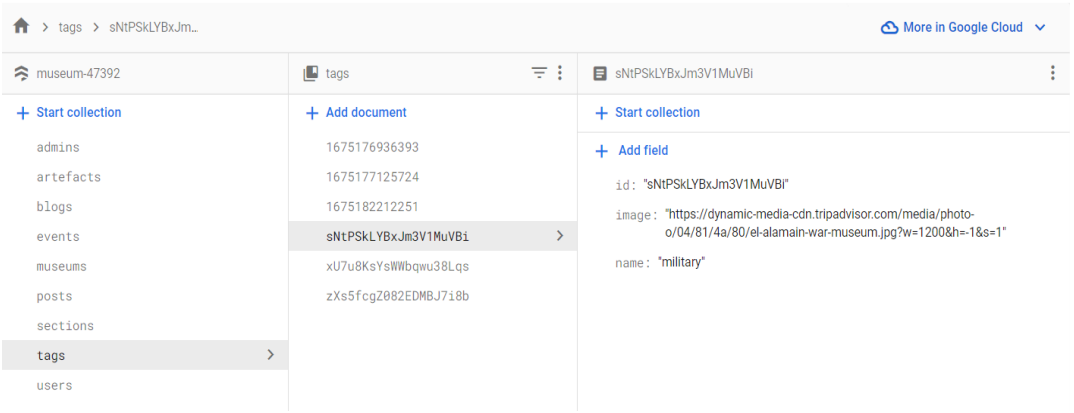


Fig 3.24 Tags collection

3.3.9 Posts Collection

Each document in the posts collection contains fields to describe the post as description, comments, email, likes, images, name, post id, timestamp, uid, user image, and who likes.

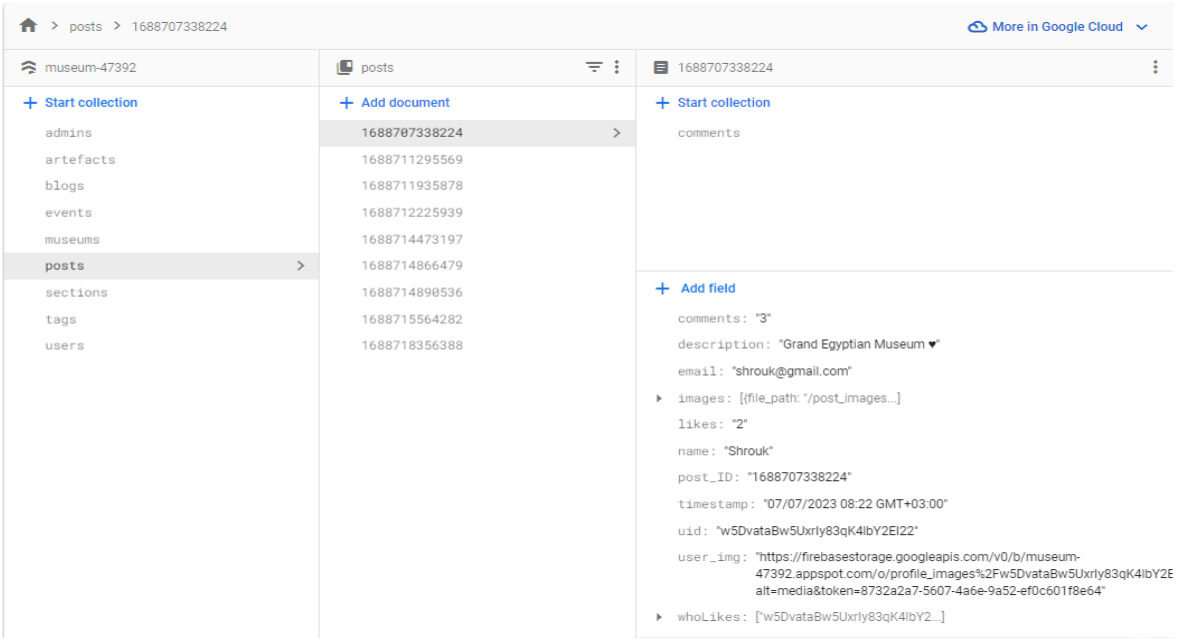


Fig 3.25 Posts collection

3.4 Indoor Localization

GPS is not the best solution for indoor localization due to GPS signals being blocked or being reflected by walls and cannot enter the room. As a result, satellite signals cannot be received properly, so it is impossible to calculate the location due to insufficient signal strength inside the room.

3.4.1 Data collection

AI models are built with the collected data of the required regions' Wi-Fi strengths values. Different methods may achieve this task including collection with mobile phones, laptops, microcomputers, or even any device with a Wi-Fi module. A mobile application is chosen because the assumed mode of operation will be a user with his mobile phone receiving and sending signals so it is desired to train the model on data similar to what a user will receive.

Using Flutter framework **an android application** is created to scan Wi-Fi signals and handle missing records and at last, it can export readings map into a CSV or XLSX files to be used for further tasks and processes.

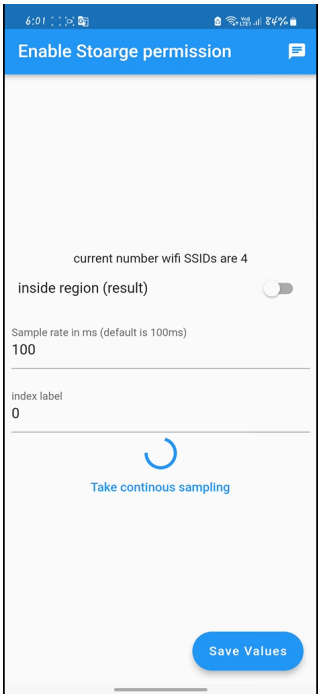


Figure 3.26 Mobile application for scanning Wi-Fi signals

For large areas taking about 10,000 distinctive samples requires moving many times around this large area. This may be hard to do, so a Raspberry pi attached to the robot is developed and used to do this task.

This robot will read Wi-Fi signals through raspbian OS which causes a delay in scanning operations in comparison with mobile phone scanning on the other hand, raspberry pi gives better variations in readings than mobile phones so we choose to use both of them in collecting data.

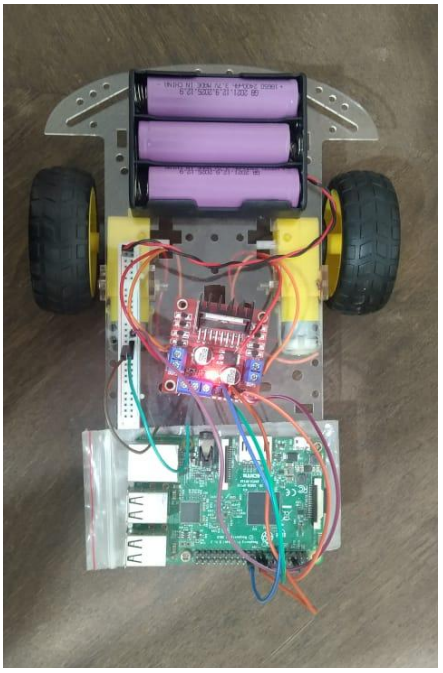


Fig 3.27 raspberry pi robot

	result	lat-position	lng-position	index	c8:0c:c8:13:74:cc	b0:ac:d2:42:96:0c	58:ba:d4:b4:33:5c
0	0	30.041852	31.374153	0	-56	-62	-69
1	1	30.041868	31.374139	0	-64	-69	-73
2	0	30.041860	31.374124	0	-59	-61	-95
3	1	30.041865	31.374126	2	-62	-65	-95
4	0	30.041852	31.374153	0	-57	-63	-70
...	...	...	...	...	...	...	...
1474	1	30.041877	31.374146	0	-65	-63	-95
1475	1	30.041878	31.374151	2	-62	-68	-95
1476	1	30.041894	31.374146	0	-61	-66	-95
1477	0	30.041856	31.374123	0	-58	-69	-69
1478	1	30.041884	31.374136	0	-62	-67	-69

Fig 3.28 Readings sample

3.4.2 Initial Indoor Localization Model Using KNN & RF

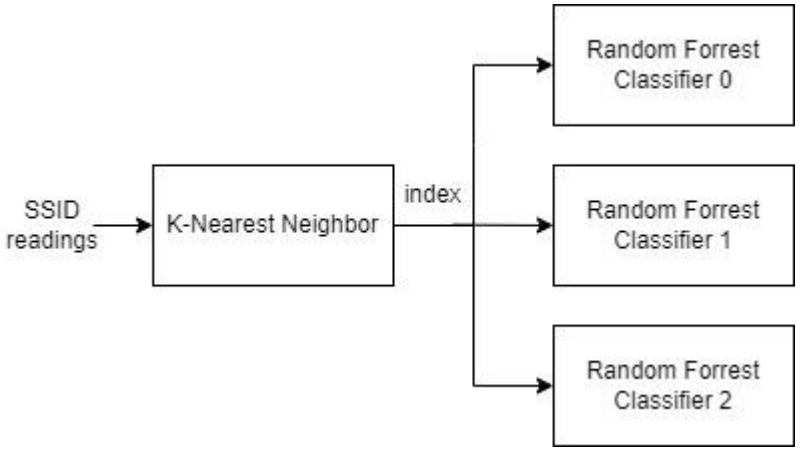


Figure 3.29 Model Arch

3.4.3 KNN as a classifier for indoor localization

KNN works by measuring the distances between the target point (whose location needs to be determined) and the labeled training data points. In indoor localization, these training data points often represent known locations within the building. By comparing the distances between the target point and the labeled data points, KNN can identify the k-nearest neighbors, where k is a user-defined parameter. The classification of the target point is then determined by a majority vote among the labels of its k-nearest neighbors. This approach is suitable for indoor localization because the physical distances between points within a building can provide valuable information about their relative positions.

KNN is particularly effective in scenarios where the indoor environment exhibits certain patterns or clusters. For example, if certain areas of a building tend to have similar features or be used for similar purposes, the KNN algorithm can leverage this spatial correlation to make accurate predictions. In such cases, the labeled data points in close proximity to the target point are likely to share similar characteristics, allowing KNN to classify the target point correctly. However, it's worth noting that the effectiveness of KNN for indoor localization depends on the quality and representativeness of the training data, as well as the appropriate selection of the parameter k. Additionally, KNN may face challenges in environments with complex structures, sparse data, or non-uniform distributions. Hence, it is important to carefully consider the specific characteristics and requirements of the indoor localization problem before applying KNN as a classifier.

3.4.4 RF as a classifier for indoor localization

Firstly, Random Forest can handle complex relationships and interactions between features in the indoor environment. In indoor localization, there can be a multitude of factors that contribute to determining the location, such as Wi-Fi signal strength, Bluetooth beacons, and physical landmarks. Random Forest is capable of capturing and utilizing these complex relationships effectively by building a large number of decision trees. Each decision tree learns from different subsets of the training data and features, and their predictions are combined through majority voting. This ensemble approach enables Random Forest to model the intricate relationships among various indoor localization features.

Secondly, Random Forest is robust against overfitting and noise in the data. Overfitting occurs when a model becomes too specific to the training data, failing to generalize well to unseen instances. In indoor localization, noise in the data can be introduced by signal interference, multipath effects, or other environmental factors. Random Forest reduces the risk of overfitting by randomly selecting subsets of features and data samples for each decision tree, ensuring that different trees learn different aspects of the problem. By aggregating the predictions of multiple trees, Random Forest achieves better generalization and can handle noisy data more effectively.

Neural network arch (simplified)

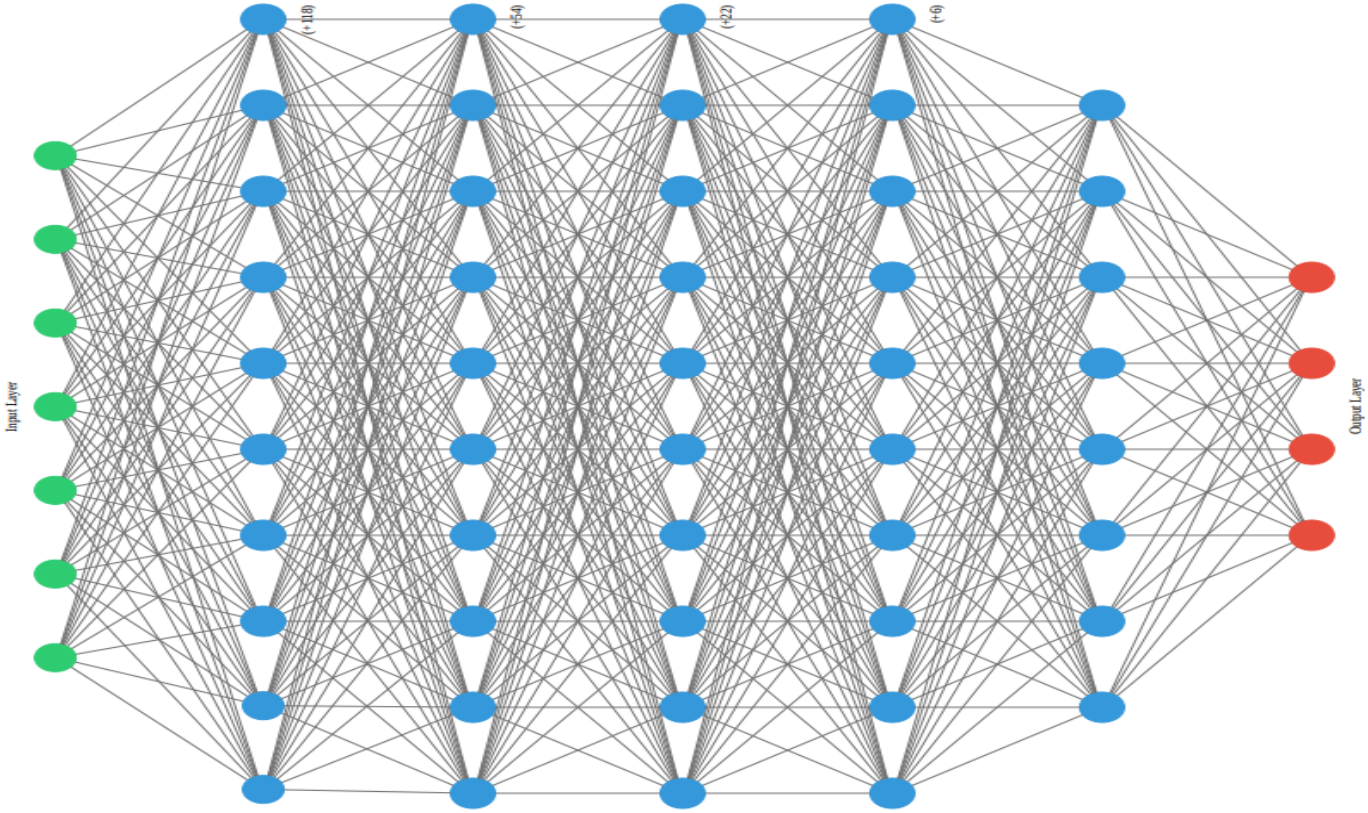


Figure 3.30 ANN

3.4.5 The use of synthetic data to increase accuracy and resolution

A GAN generates synthetic data that resembles real data. This synthetic data can be used to supplement the training set of an ANN, improving its accuracy by providing additional diverse examples for learning, this can also add examples that weren't within the dataset and overall help the model be fit.

Process of Creating synthetic data using a GAN

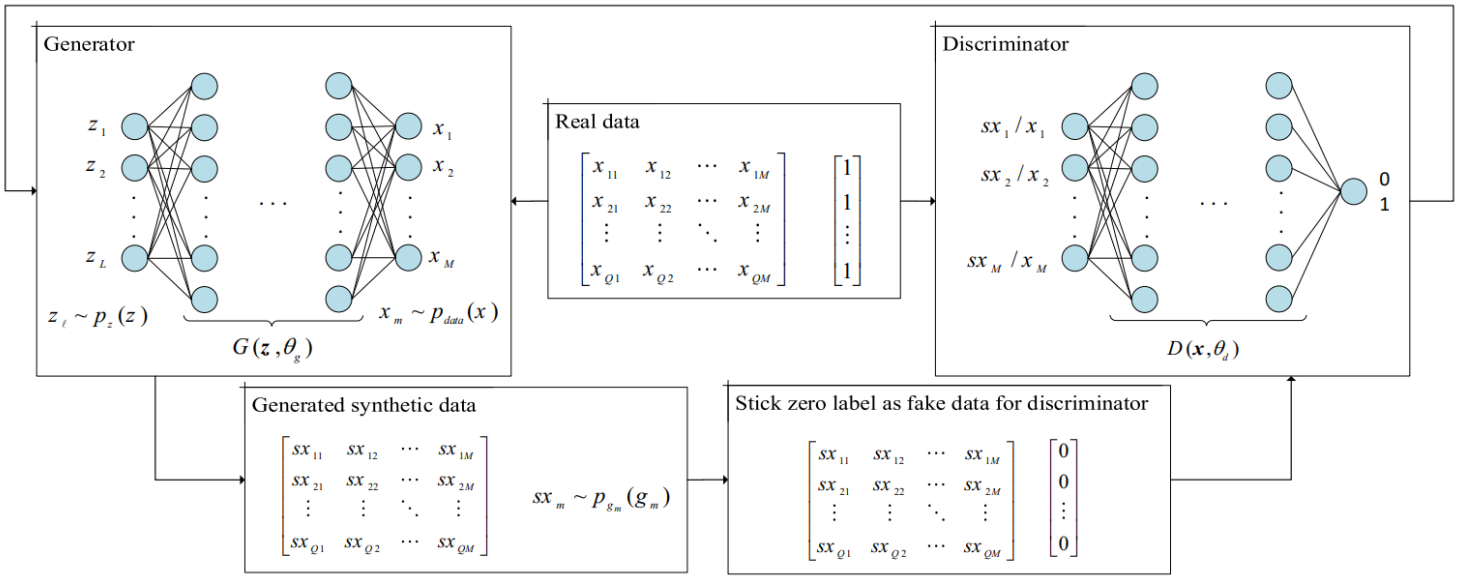


Figure 3.31 GAN architecture

After training the generator learns to generate synthetic data that closely matches the dataset collected, and the discriminator learns to label the synthetic data. This process is relatively fast after training.

Tools used:

- Sci-kit learn
- Numpy
- Pandas
- TensorFlow
- Keras

Initial testing and different model testing

Model Name	Validation Accuracy (15%) Average over 50 tests	Log Loss
KNN	86.2%	1.10
RF	84.5%	1.33
KNN+RF	92.8%	0.42
SVM	79.1%	2.27
Linear Reg	81.2%	1.77
ANN without GAN & RSSI prediction	92.4%	0.22
ANN with GAN	95.8%	0.21
ANN with GAN & RSSI prediction	97.4%	0.09

Table 3.1 Initial testing and different model testing

3.5 AR Experience

3.5.1 AR figures

The AR section simply implies a couple of features, the first feature is that it tries to detect the user's approximate location, thus when existing in a specific museum for example, a AR 3D model corresponding to the culture of that specific museum shall be shown, and it shall not be shown except in that specific place and will disappear when this place is left, and this happens to most museums, having a 3D model dedicated for every museum, or every place in the museum.

Figure 3.32 AR 3D model rendered in a specific place

3.5.2 Augmented Images

Another feature we have is Augmented images, where we can play videos for the user in a different way than the traditional way, and instead of letting them search for media and so , the augmented images help us detect specific images in the physical world , for example, a user can detect a specific piece of art, a hanging drawing or drawings on a temple wall, and then when those surfaces are detected, a specific video talking about the history of this view and related to the topic of it will be played, thus making the user entertained with a different way in viewing media.

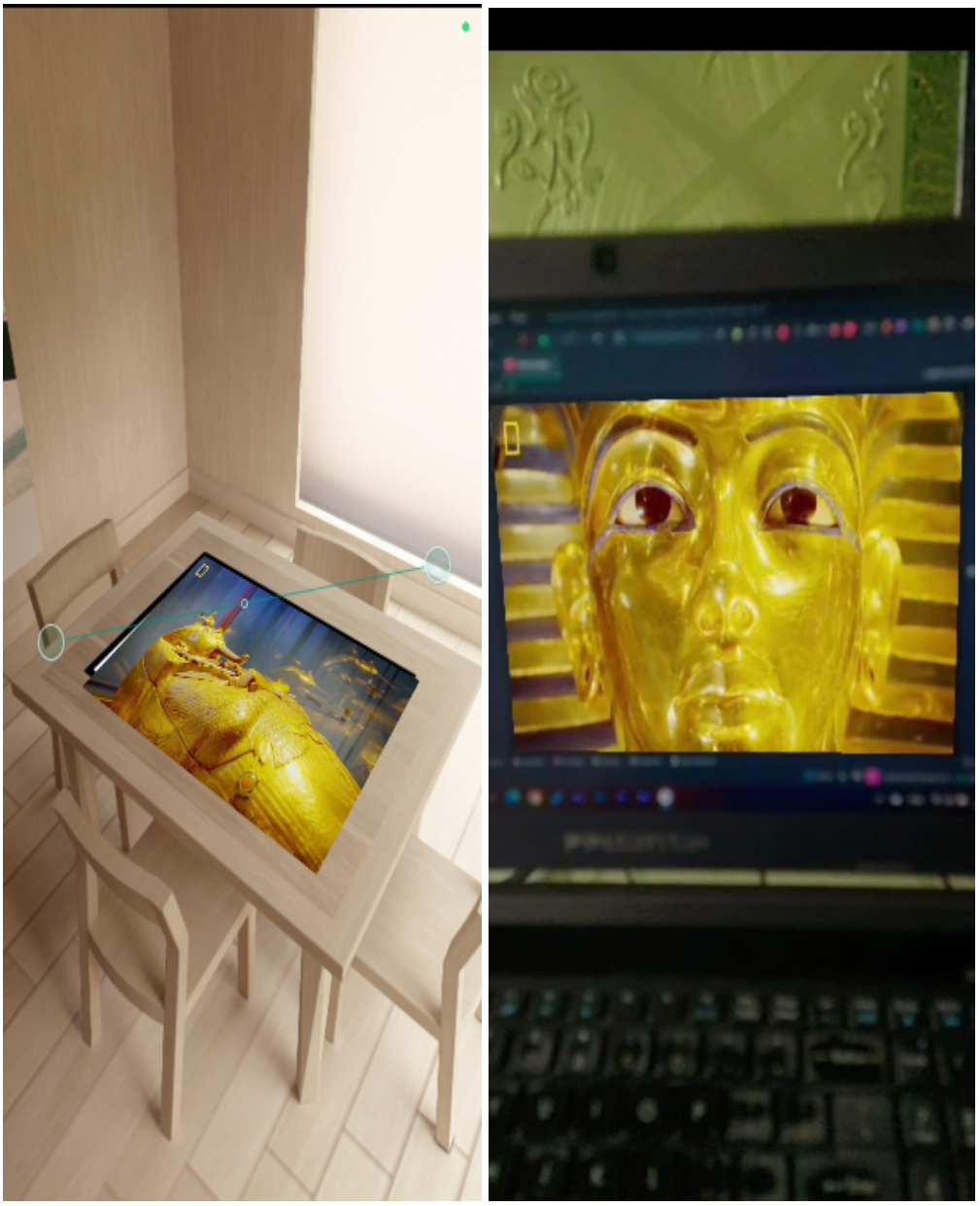


Figure 3.33 Augmented video played on a detected surface, a photo, in Emulator and the physical world

A challenge we face is that those 3D might be taking space which might be high, which is bad for performance, so we have all the models stored in a Firebase remote storage using and will cache every model in the device storage when it is needed, thus to save resources and improve performance, the following flow chart might clarify the flow of the process.

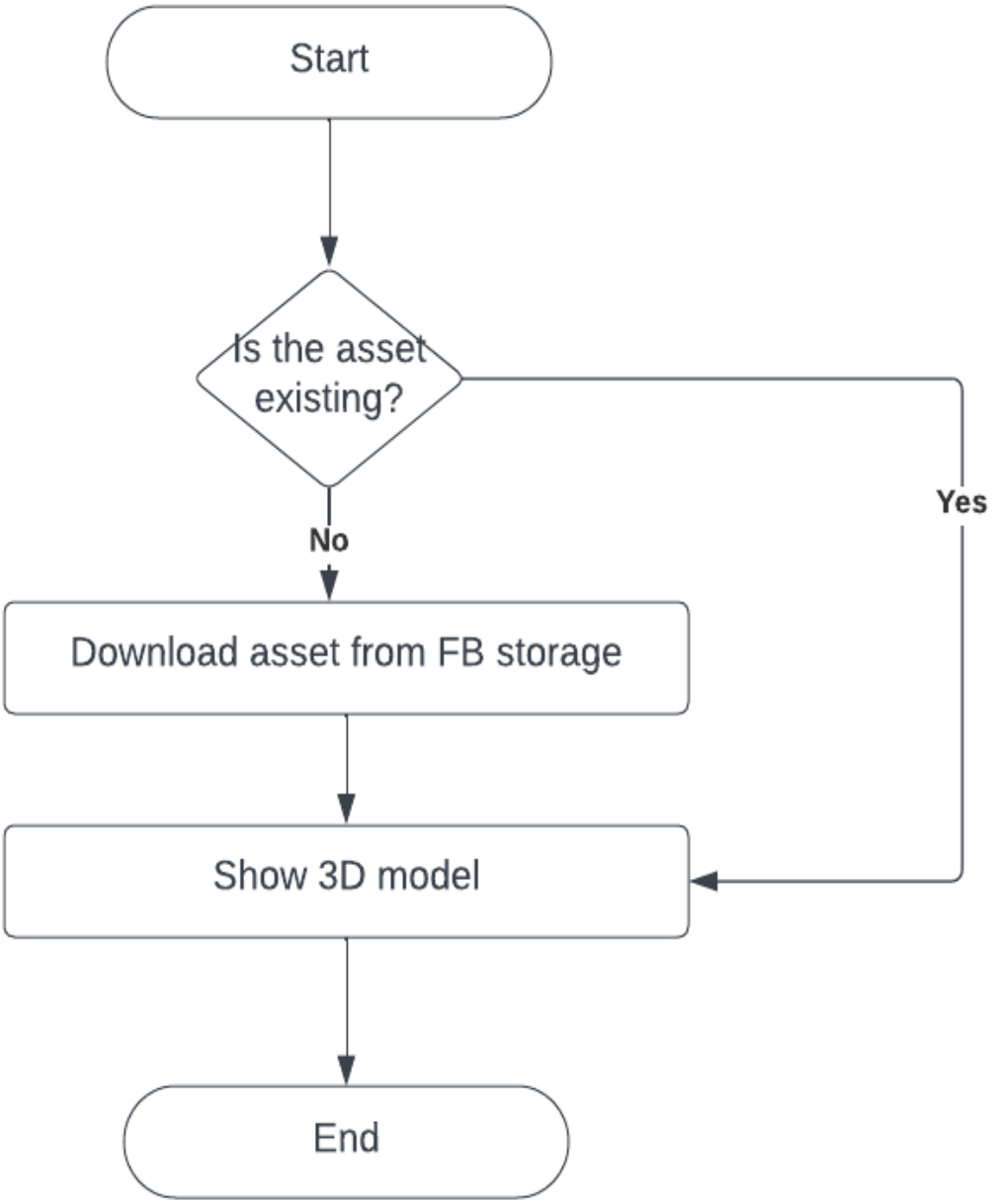


Figure 3.34 Flow chart illustrating how fetching from FB works

3.6 Extractive Question Answering and Fine Tuning BERT

Extractive question answering involves identifying and extracting the answer to a given question directly from a given text. This approach is akin to using a highlighter to mark relevant parts of the text and then extracting the required information.

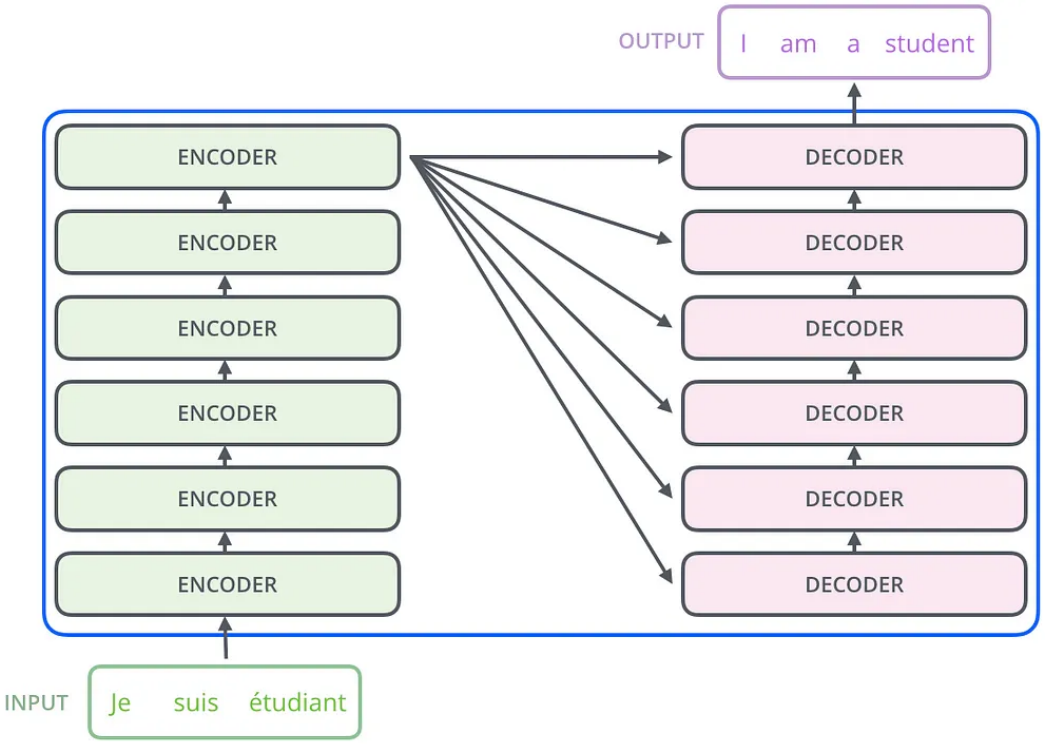
How does Extractive process work? the entire process is divided into 5 main steps:

1. Question processing: The model leverages natural language processing techniques to break down the question into smaller components, extracting and identifying the most relevant keywords to help generate the final answer. Techniques such as named entity recognition and part-of-speech parsing are employed in this step.
2. Text segmentation: The model dissects the text document into smaller units, such as sentences or paragraphs. This allows for the analysis of each unit separately, identifying those that are more likely to contain the answer. Segmenting the text also reduces computational complexity, making the process more efficient.
3. Data analysis: Once the text is segmented, the model analyzes each unit using various techniques, including named entity recognition, part-of-speech tagging, and syntactic parsing. These techniques help identify essential entities, grammatical structures, and relationships between words and phrases in the text.
4. Answer extraction: Finally, the model selects the highest-scoring unit and extracts the answer from it. The answer can be a single word, a phrase, or a complete sentence, depending on the question and the information available in the text. The extracted answer is then presented to the user as the model's response to the question.

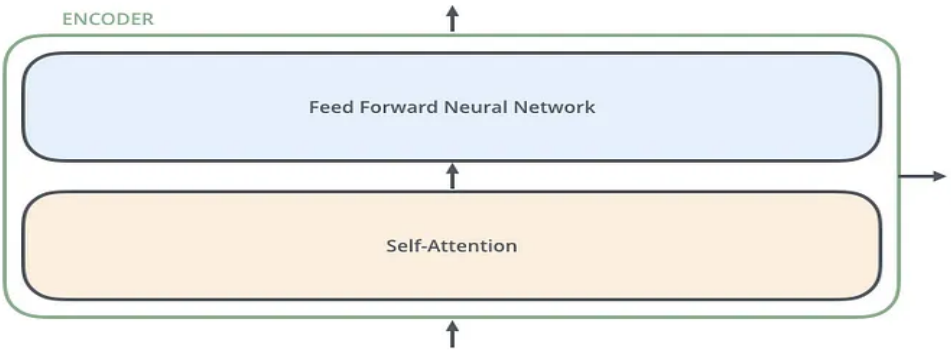
3.6.1 Transformer Working theory:

Let's take a look at how Transformer works. Transformer is a model that uses attention to boost the speed. More specifically, it uses self-attention.

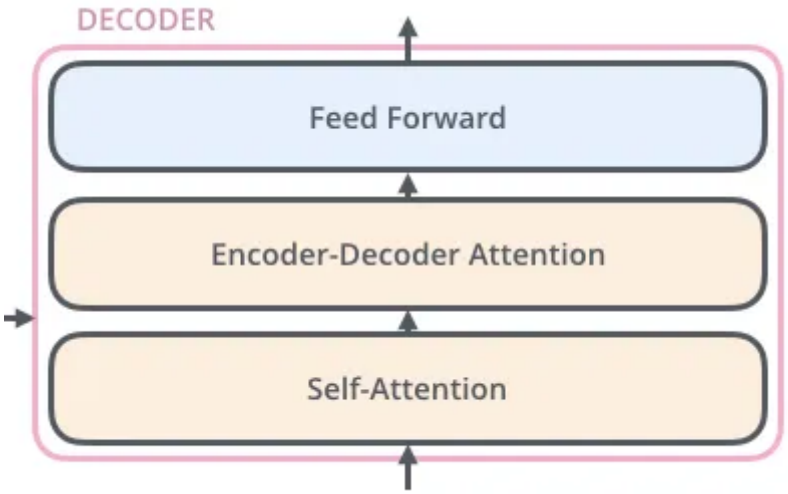
Internally, the Transformer has a similar kind of architecture as the previous models above. But the Transformer consists of six encoders and six decoders.



Each encoder is very similar to each other. All encoders have the same architecture. Decoders share the same property, i.e. they are also very similar to each other. Each encoder consists of two layers: Self-attention and a feed Forward Neural Network.



The encoder's inputs first flow through a self-attention layer. It helps the encoder look at other words in the input sentence as it encodes a specific word. The decoder has both those layers, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence.



3.7 Implementation using HuggingFace Transformers & Bert

The base Model is RoBERTa, it's a Pretrained model on English language using a masked language modeling (MLM) objective. This model is case-sensitive: it makes a difference between english and English. This model includes 124M parameters, and was trained on wikipedia and bookcorpus. Research paper where bert is based off.

The model was then fine tuned for the Question Answering (Extractive) task, where it was training on the Squad_v2 dataset, it was also validated and evaluated on the same dataset using the following hyperparameters:

batch_size = 96
n_epochs = 2
base_LM_model = "roberta-base"
max_seq_len = 386
learning_rate = 3e-5
lr_schedule = LinearWarmup
warmup_proportion = 0.2
doc_stride=128
max_query_length=64

Evaluation Metrics:

Performance

Evaluated on the SQuAD 2.0 dev set with the official eval script.

"exact": 79.87029394424324,

"f1": 82.91251169582613,

"total": 11873,

"HasAns_exact": 77.93522267206478,

"HasAns_f1": 84.02838248389763,

"HasAns_total": 5928,

"NoAns_exact": 81.79983179142137,

"NoAns_f1": 81.79983179142137,

"NoAns_total": 5945

Result

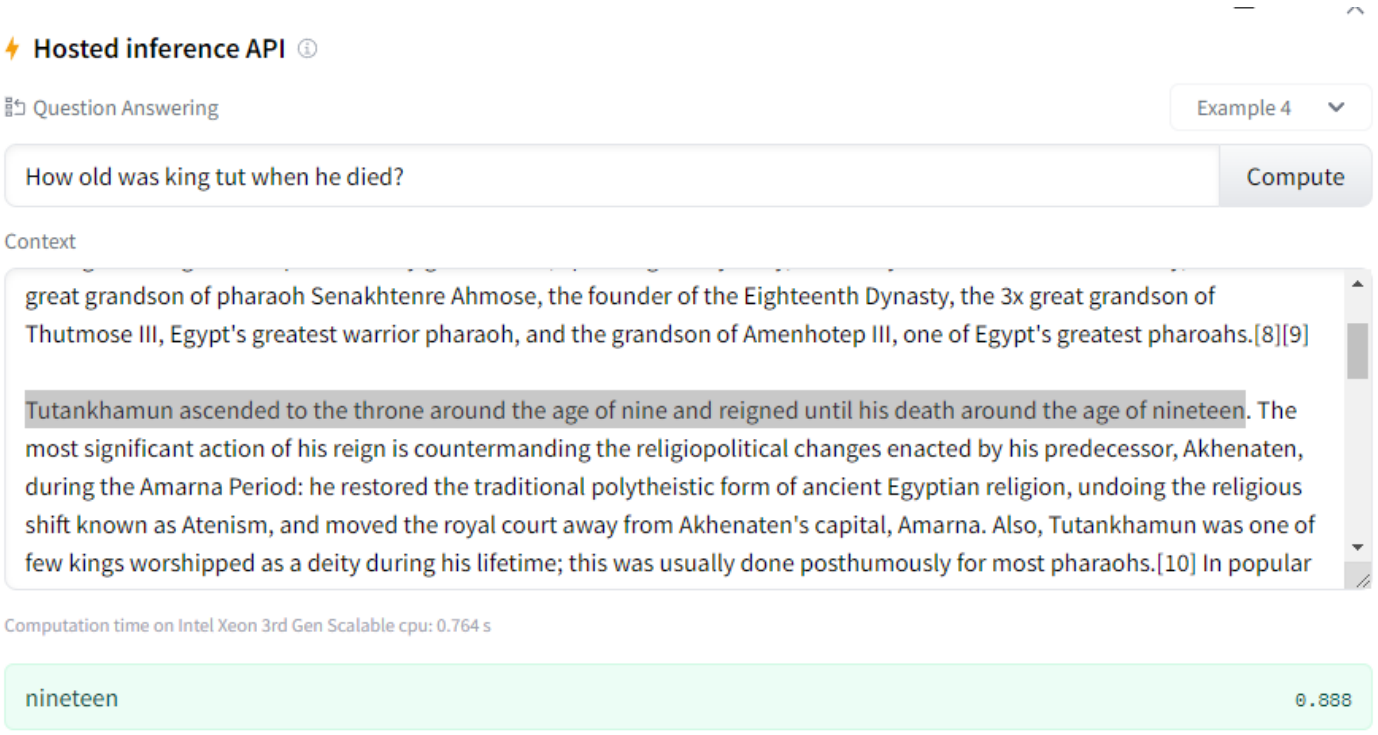


Figure 3.35 Chatbot results

This model is hosted on huggingface hub:
https://huggingface.co/PavlySafwat/ExtractiveQABertBase
It also provides an API access point which hosts the model and does all processing server side, which helps the model be accessible by many device types including mobile phones

3.8 Community

3.8.1 Show Posts

In this screen, the user can see all the post's posts and like, share and comment on any post.

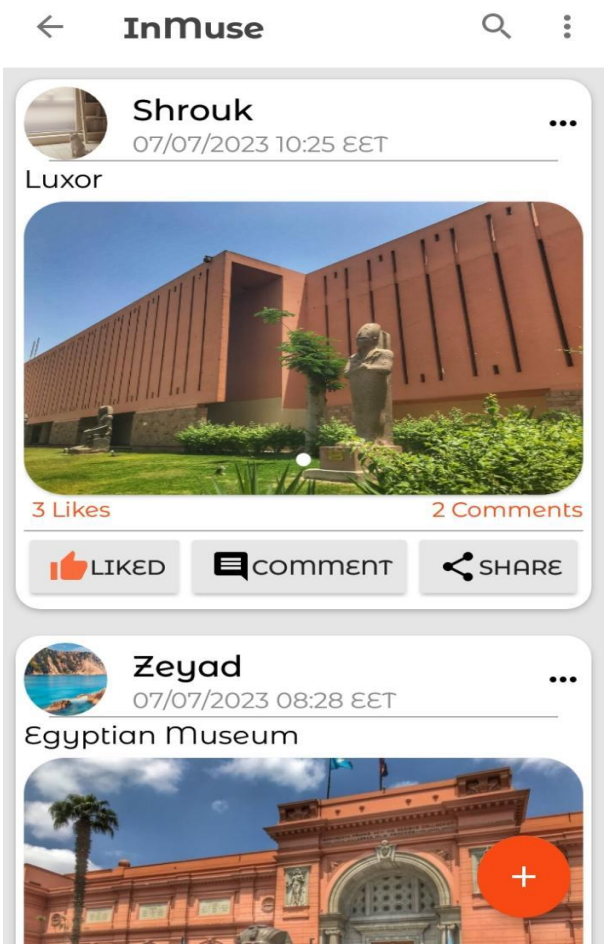


Figure 3.36 Blogs

3.8.2 Edit & Delete Posts

In this screen the user can edit the description and images. Also he can delete any of his own posts.

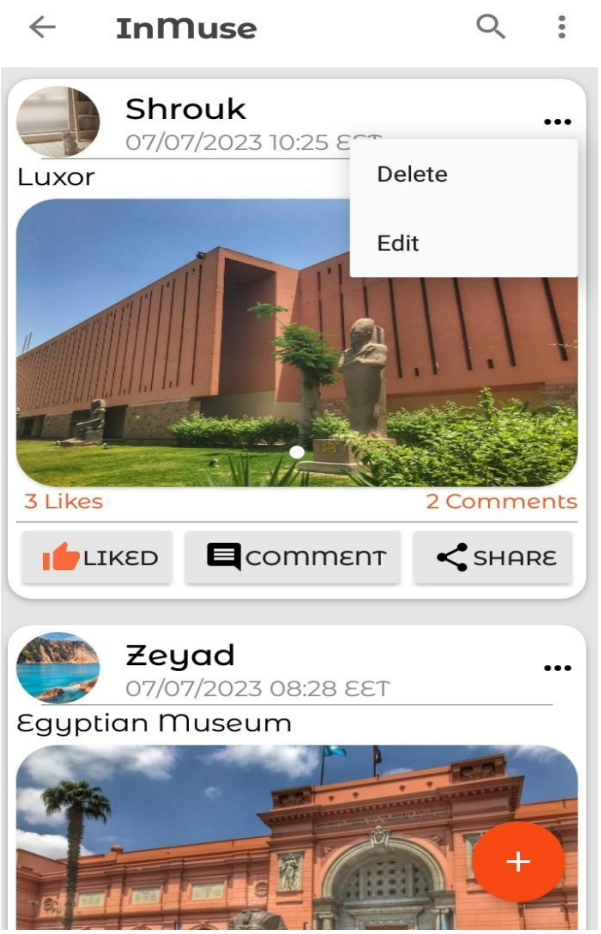


Figure 3.37 Edit & Delete Posts

3.8.3 Add New Post

In this page the user can add a new post with many images and add description.

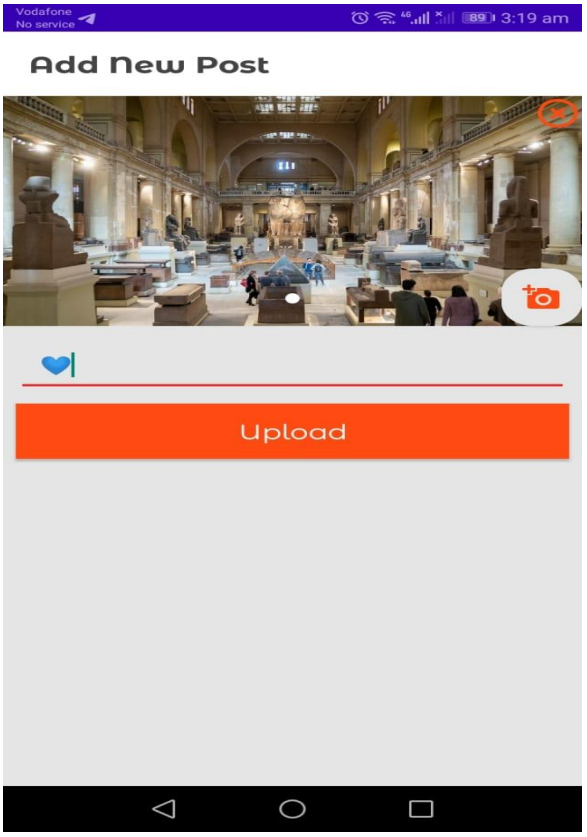


Figure 3.38 Add new posts

3.8.4 Update Post

In this page the user can update the images and the description of the post.

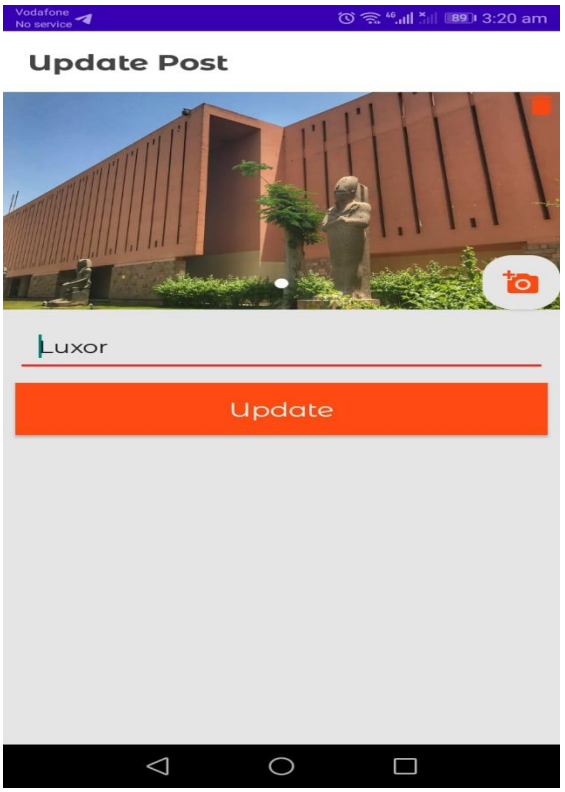


Figure 3.39 Update posts

3.8.5 Share Posts

In this page the user can share any post he want and edit its description.

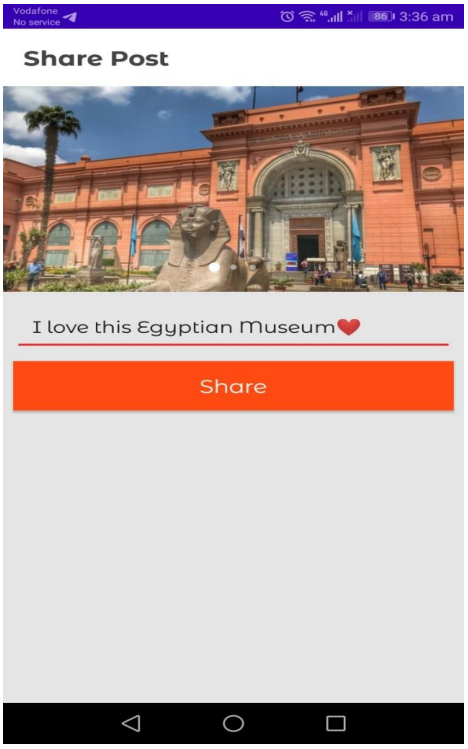


Figure 3.40 Share posts

3.8.6 Add Comments

In this page the user can add comments to any post.

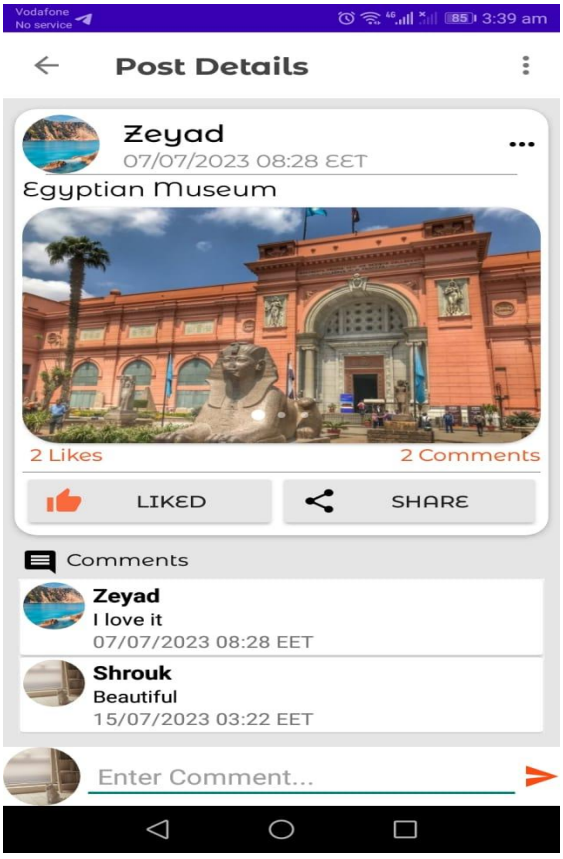


Figure 3.41 Add comments

3.9 Audio Broadcasting

Audio broadcasting is built on UDP socket technology. By sampling and encoding the broadcaster sound with PCM16 format.where PCM refers to Pulse Code Modulation that uses a 16-bit representation for each sample. In PCM 16, the analog signal is sampled at regular intervals, and each sample is quantized into a 16-bit binary number.
This format has many advantages, some of them are stated; easy digital processing, robustness against noise and interference, and compatibility with digital systems. It is the basis for various audio formats, including WAV (Waveform Audio File Format) and CD (Compact Disc) audio.

PCM has many variants depending on the bit representations of the audio samples; it may be PCM 8 (8-bit), PCM 24 (24-bit), and PCM 32 (32-bit). The choice of the appropriate bit depth depends on the specific application requirements and the desired level of audio fidelity.

UDP multicasting/broadcasting requires special configurations in the hosting router which may be difficult or overhead for the broadcaster to configure as the expected mode of operation is the broadcaster will open his personal mobile hotspot and connect the subscribers to his phone.

UDP (User Datagram Protocol) is often considered suitable for audio broadcasting over TCP (Transmission Control Protocol) due to several reasons:

1. **Low latency:** Video conferencing requires real-time transmission of audio and video data. UDP's connectionless nature and lack of overhead allow for lower latency compared to TCP. With UDP, there is no need for the additional handshaking and acknowledgments that TCP requires, resulting in faster delivery of packets. This reduced latency helps maintain a smooth and responsive video conferencing experience.

2. **Reduced retransmissions:** In real-time streaming , a small amount of packet loss is generally tolerable, but frequent retransmissions can significantly impact the quality and real-time nature of the communication. TCP is designed to ensure reliable data transmission by retransmitting lost packets, which can introduce additional delays. UDP, on the other hand, does not have built-in retransmission mechanisms.

3. **Control over data transmission:** UDP provides applications with greater control over the transmission process. In real-time streaming, developers can design their own error detection, loss recovery, and congestion control mechanisms tailored specifically for their needs. This flexibility allows for optimizations that are more suitable for real-time performance, adapting to varying network conditions and delivering an optimized real-time streaming experience.

4. **Scalability:** UDP is a lightweight protocol compared to TCP, requiring fewer system resources and incurring less processing overhead. This makes it more suitable for handling multiple simultaneous real-time streaming, where scalability is essential to support a large number of participants.

While UDP offers advantages in terms of low latency, reduced retransmissions, control over data transmission, and scalability, it is important to note that UDP does not guarantee the **same level of reliability and congestion control as TCP.** Therefore, it is crucial for real-time streaming using UDP to implement appropriate **error-handling mechanisms** at the application level to ensure the best possible user experience.

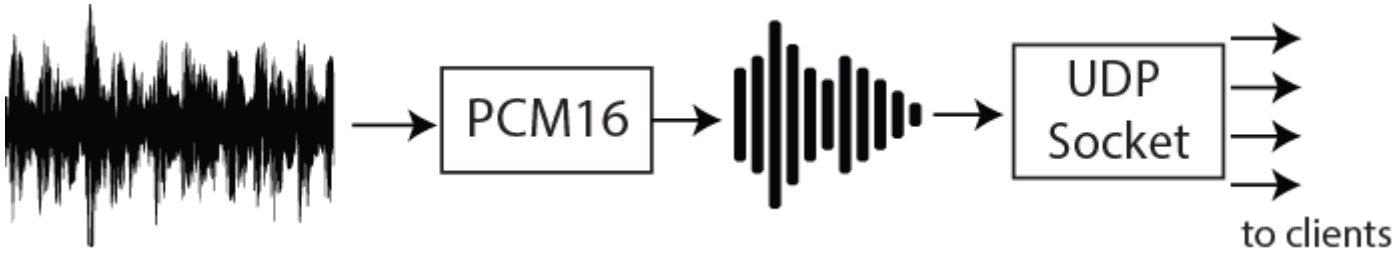


Figure 3.42 transmitter side of broadcaster

To subscribe to a broadcaster the subscriber will scan a QR code image from the host's mobile phone. The QR code encodes the IP address and port number of the host's mobile phone, then the subscriber's mobile phone will send a "notify to connect" message to the host phone. The host's phone will register the IP address of the sender to the subscriber's list and upon broadcasting the host phone will send to all connected subscribers.

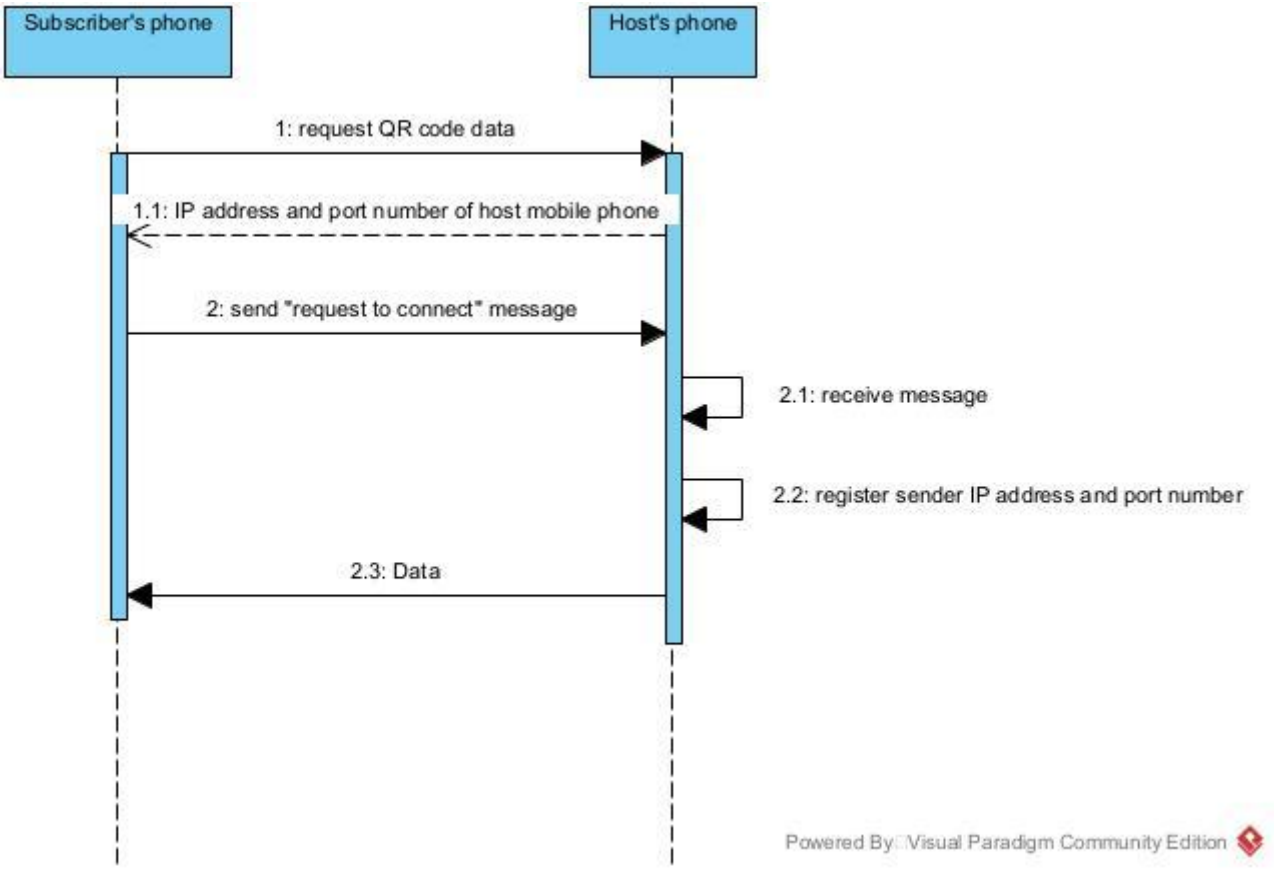


Figure 3.43 sequence of broadcasting

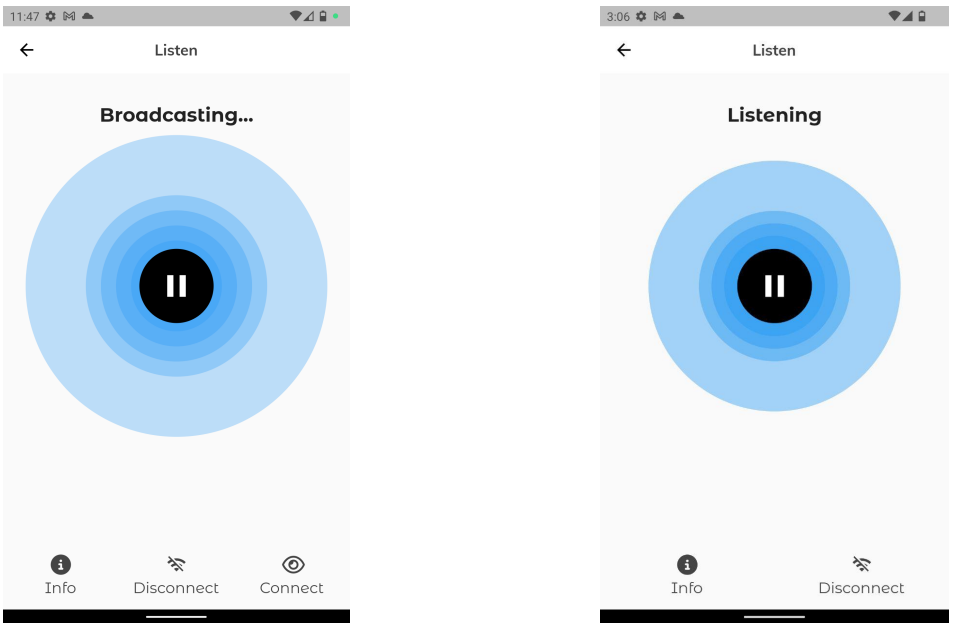


Figure 3.44 sender and receiver screens

At the receiving side it will initialize a media player instance then it will listen to its socket upon receiving new data it will pass these bits to the media player instance.

The sender can pause or stop broadcasting and the same for the receiver.

CHAPTER FOUR
DESIGN SPECIFICATIONS

4.1 System Block diagram

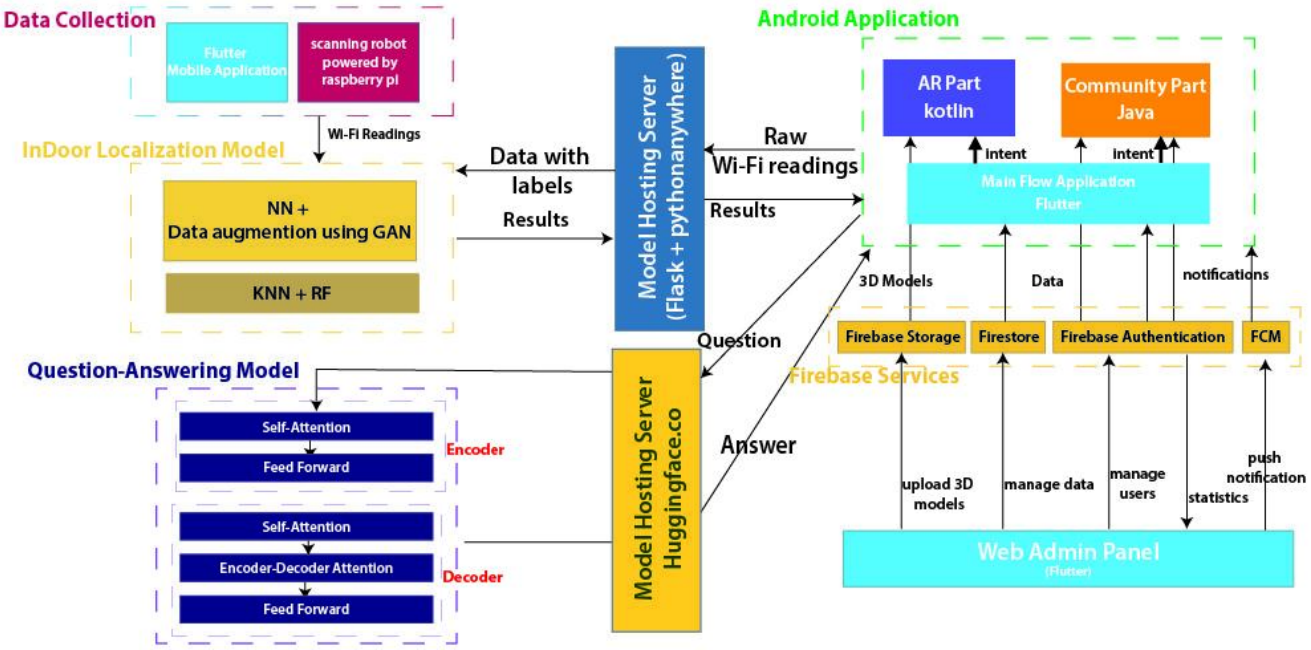


Figure 4.1 System block diagram

A block diagram shows the overall system block diagram it can be concluded from this diagram that:

- Android Application
  - Main component of the system where it is the only component that user can interface with it directly
  - It consists from three different components each developed with different framework
  - It is interfaced with other system components through proxy layers to achieve reliability and scalability as the system can be changed in one component without affecting the other and can replicate other components.

- Firebase services
  - The service provider layer that provides services and data including; Firestore, cloud messaging, cloud storage, authentication and hosting.
- Web admin panel
  - Interfaced with firebase services layer which provides communication between client application and admin panel.
  - Used to manage content and client application.
- Model hosting server

    - A layer between the client application and the indoor localization ML models.
    - Provides scalability that models can be changed without affecting the client's mobile application.
- Data collection
    - Combination of mobile application and scanning robot.
- Indoor localization models
    - Seralized pkl files that are deserialized by the hosting server.
    - These models are given input in a suitable format -np array- and outputs a prediction which is sent to the client mobile application.

- Question/Answering model
    - Models hosted by hugging face.
    - Feed with context it will analyze the question and provide the answer.

4.2 Use-Case diagram

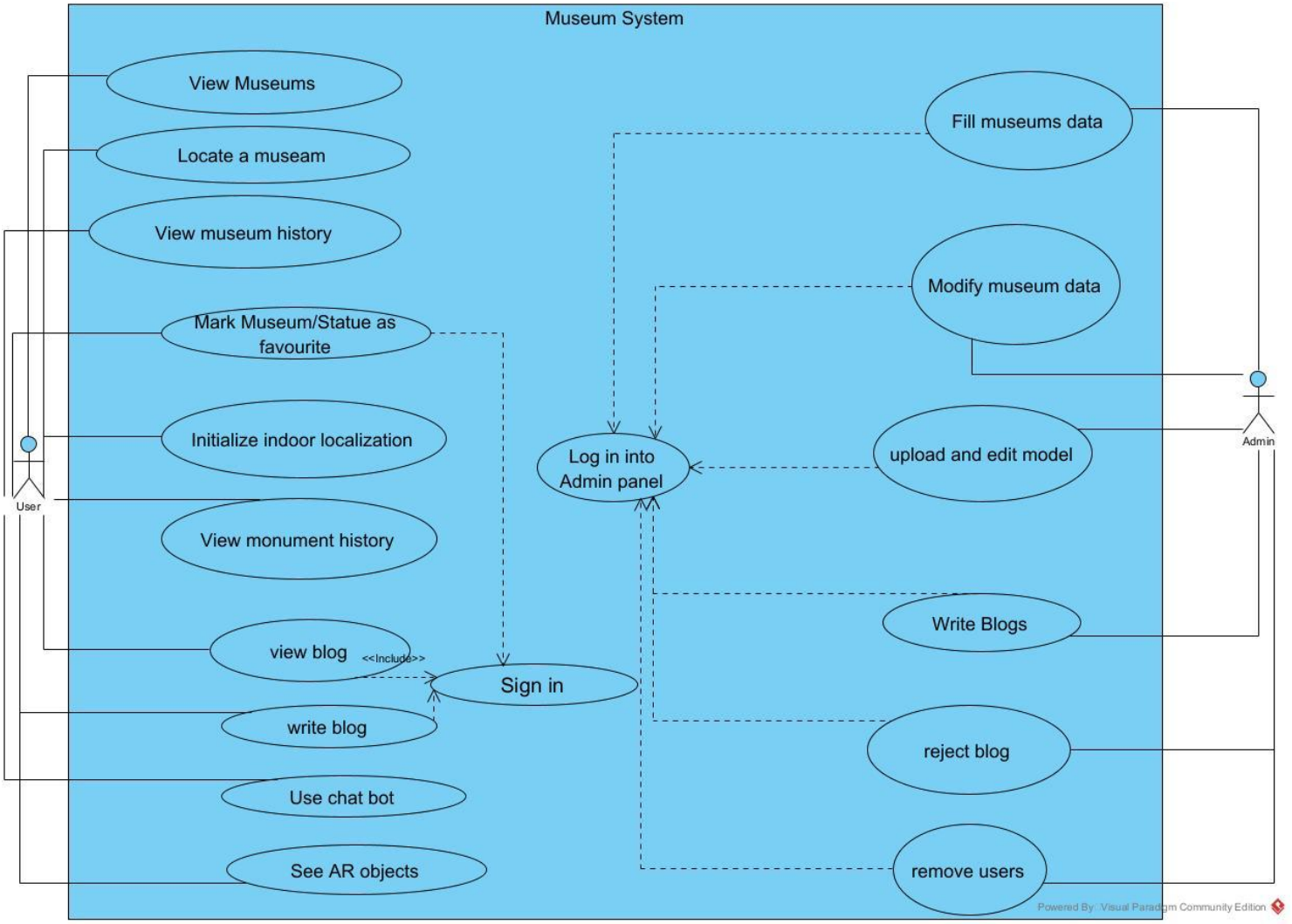


Figure 4.2 System use case diagram

In the developed system we have two actors a user and an admin, mainly user utilizes the main system features and the admin manages the content of the application.

An admin has its admin panel (as mentioned in Chapter 3) with limited access that limits admin function without login.

For a user the application is partially limited access as some features are available to all users such as viewing museums or communicating with the host in audio broadcasting some others require login including user authentication in some processes like commenting or accessing the community.

4.3 Sequence diagram

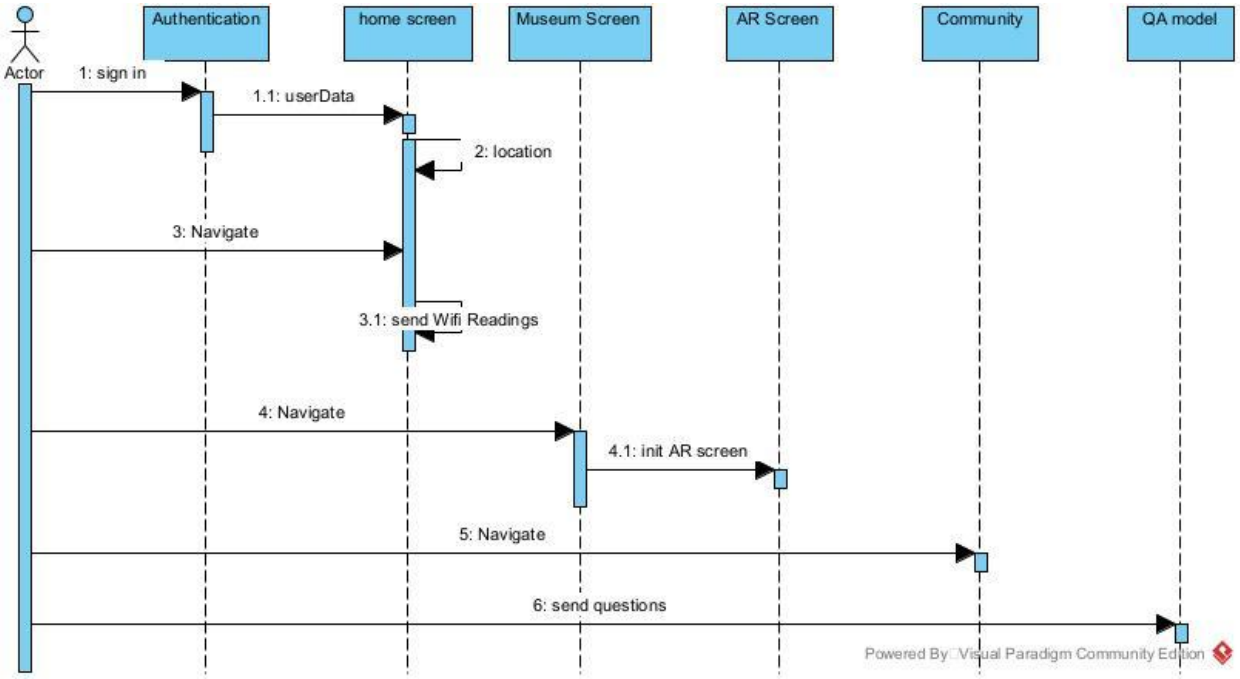


Fig 4.3: sequence diagram

Shown is a sequence diagram for the general mode of operation for user and using most of the features of the system

4.4 Class diagram

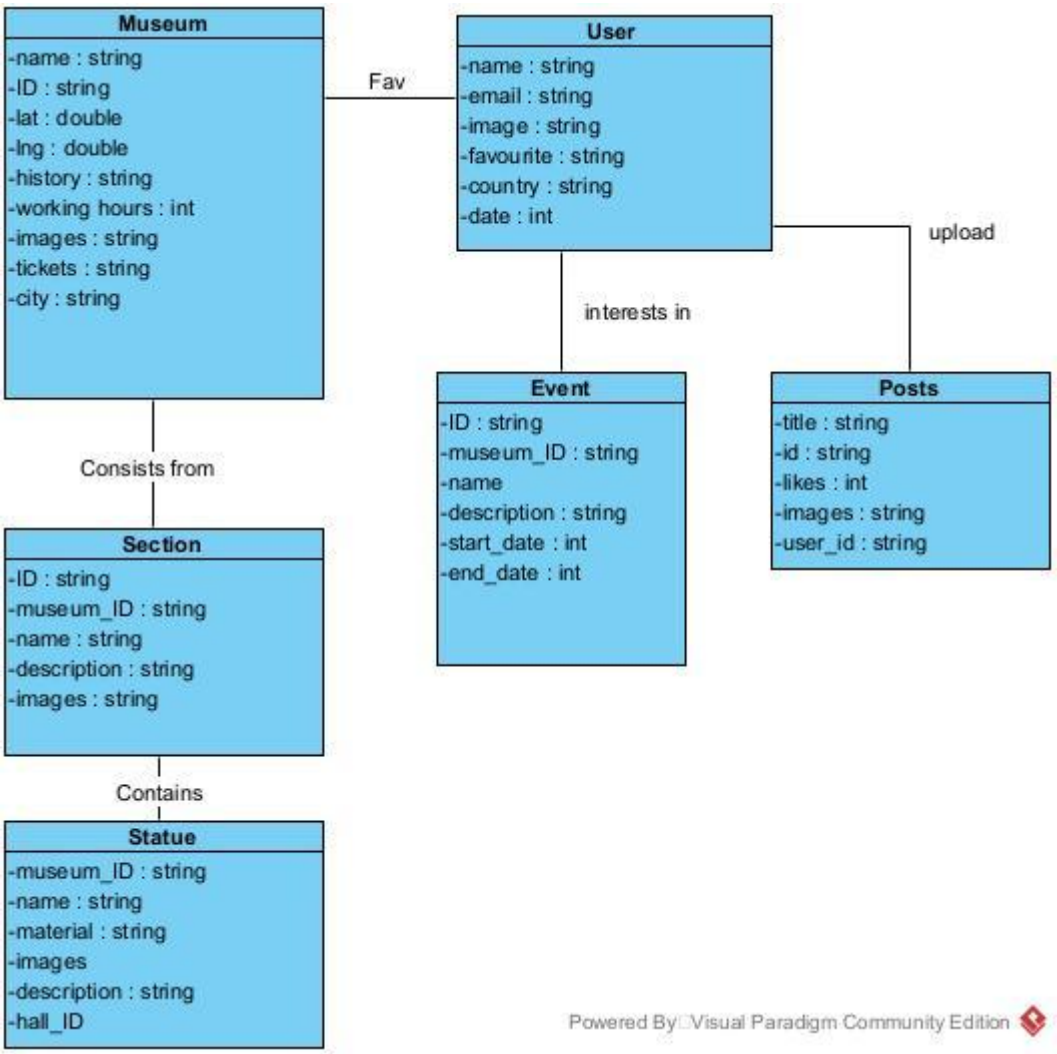


Fig 4.4 class diagram

The shown diagram is the class diagram for the classes created by the system designers, many other classes are used and depend on from the developing frameworks (Android Java, Android Kotlin, and Flutter)

4.5 Entity Relationship diagram

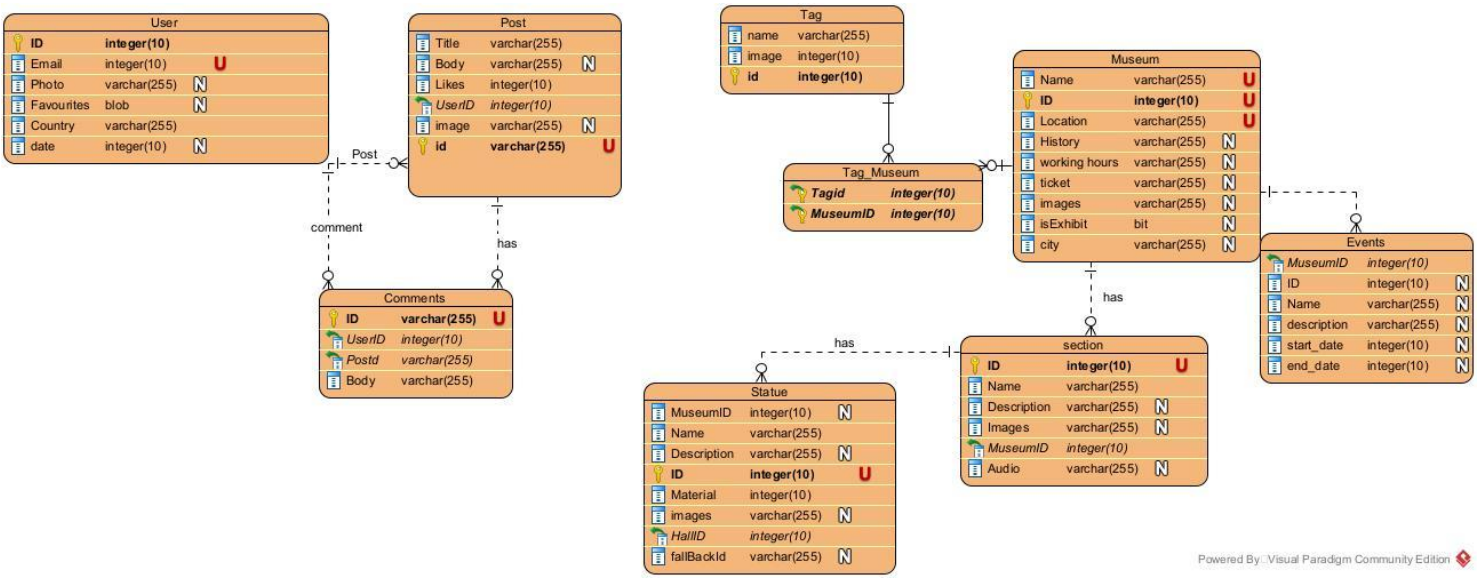


Figure 4.5 Entity relationship diagram

CHAPTER FIVE
CONCLUSIONS

5.1 conclusions

Indoor localization can be implemented with different techniques, we choose to use artificial neural networks and to augment the similar data with new generated data by GAN after using Random forest and K-nearest neighbor and comparing results.

Android applications can be integrated with such models by using a hosting model server that acts as middleware layer this results in a better experience.

A Single Android application can be implemented with several frameworks which are Android Java, Android Kotlin and Flutter. This can be achieved by providing common data structure between languages and class serialization thanks to the common environment that android application (APK -android packaging kit-) provides.

AR technology has many variants including displaying AR objects in certain regions defined by GPS or displaying Augmented images or videos on some defined images by using techniques of computer vision and Sceneview.

Community is an important part to enhance application, and innovative methods should be used to store data in NoSQL Firestore to reduce the number of reading and writing.

Audio broadcasting can be achieved on a local network with defining sockets on both sender and receiver.

5.2 Limitations

AR is not supported on all android devices, but for the new mobile phones they support the AR.

Indoor localization models should be specific to regions

5.3 Future Work

Recommendation system is ready to implement on posts and museums as every museum is classified into a set of tags which may help in classifying museums.

IOS and web versions are ready to be developed since a part of this application is created by cross-platform flutter

5.4 References

[1]*Using synthetic data to enhance the accuracy of Fingerprint-Based Localization: a deep learning approach*. (2020, April 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8981805

[2]Shang, S., & Wang, L. (2022). Overview of WiFi fingerprinting‐based indoor positioning. *Iet Communications*, *16*(7), 725–733. https://doi.org/10.1049/cmu2.12386

[3]Feng, X., Nguyen, K. B., & Luo, Z. (2021). A survey of deep learning approaches for WiFi-based indoor positioning. *Journal of Information and Telecommunication*, *6*(2), 163–216. https://doi.org/10.1080/24751839.2021.1975425

[4]*Wi-Fi and Bluetooth contact tracing without user intervention*. (2022). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/9866766

[5]*Machine Learning based indoor localization using Wi-Fi RSSI Fingerprints: an overview*. (2021). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9531633

[6]SceneView. (n.d.). *GitHub - SceneView/sceneview-android: SceneView is a 3D and AR Android Composable and View with Google Filament and ARCore. This is a Sceneform replacement in Kotlin*. GitHub. https://github.com/SceneView/sceneview-android#readme

[7]Slocum, J. (n.d.). *James Slocum*. https://jamesslocum.com/blog/post/77759061182

[8]Wallarm. (2023, April 21). *What is gRPC? Meaning, Architecture, Advantages*. https://www.wallarm.com/what/the-concept-of-grpc#:~:text=gRPC%20Architecture,-In%20the%20following&text=The%20gRPC%20client%20makes%20the,library%20in%20the%20local%20machine

[9]Rosencrance, L., & Matturro, B. (2021). Remote Procedure Call (RPC). *App Architecture*. https://www.techtarget.com/searchapparchitecture/definition/Remote-Procedure-Call-RPC

[10]*Firestore | Firebase*. (n.d.). Firebase. https://firebase.google.com/docs/firestore