ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

**NEUMANN JÁNOS**
**FACULTY OF INFORMATICS.**

Annex 5

# DEGREE PROJECT
# THESIS

**OE-NIK** Student's Name: **Vishav Lakhtia**
**2022** Student's registration number: **T/007240/Fl12904/N**

Óbudai University
John von Neumann Faculty of Informatics
Institute of Biomatics and Applied Artificial Intelligence

# THESIS WORK SPECIFICATION

| | |
|---|---|
| Name of the student: | **Vishav Lakhtia** |
| Identification number of the student: | T/007240/FI12904/N |
| Neptun's code: | [redacted] |

Title of the thesis:

**Automated Arbitrage for AI technologies in modern application development on mobile platforms**
**MI technológiák automatizált arbitrázsa a modern alkalmazásfejlesztésben mobil platformokon**

| | |
|---|---|
| Internal supervisors<br>External supervisor: | Dr. Katalin Szenes, Bence Tureczki |
| Deadline: | 15 December 2021 |
| Subjects of the final exam: | Computer Architectures<br>Cloud service technologies and IT security specialization |

**The task:**

The goal is to develop an authenticating system for mobile phones and using special barcodes. We use Android which is one of the most popular operating system for mobile.
The thesis aims to integrate the autonomous machine learning models in android applications. This application will use the Google Machine Learning kit and CameraX library to scan the barcodes to authenticate guest tickets for the events organized by the company "Erasmus Life Budapest". Various tools and libraries will be utilized in the creation of the application.

**The thesis must contain:**

- Introduction of the target company and the product.
- The main features of the Android operating system.
- The advantages of the Google Machine Learning Kit.
    - Why we use Machine Learning.
    - Comparing the Google Machine Learning Kit to other Machine Learning systems.
- The system plans
    - The way of implementation of Model-View-ViewModel design pattern.
    - Descriptions of user interface components.
    - Description of storing and accessing data from the database.
    - Description of data transfer communication between different modules of the application.
    - The security of the system.
        - The security aspects to be considered.
        - Contradictions between the chosen aspects.
        - Fulfilling the chosen security criteria.
        - Enhancing the data security when communicating with different android applications.
- Special Tools and Technologies to be used.
- Test plan and test results.
- User Interface Tests.
- Integration tests.
- Unit Testing.
- Possible directions of future development.
- Glossary of Terms.
- List of Abbreviations.
- Bibliography

Place of stamp

.....
Dr. György Eigner
head of institute

This specification is valid until: **15 December 2023**
(According to OE TVSz 55.§)

I consider the thesis suitable for submission:

.....
external supervisor

.....
internal supervisor

STUDENT'S DECLARATION

I the undersigned student hereby declare that this degree project / thesis is the result of my own work; I have disclosed the references and tools used in an identifiable manner. The results indicated in my degree project / thesis completed may be used by the university and the institution announcing the task for their own purposes free of charge.

Dated: Budapest, 12 MAY 2022.

.....
student's signature

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Faculty of Informatics
Annex 9

# CONSULTATION LOG

Student's name: Vishav Lakhtia

Neptun code: [redacted]

Specialty: Cloud Computing Services

Phone number:

Mailing address (e.g. domicile): [redacted]

Degree project / thesis[1] title in Hungarian:

MI technológiák automatizált arbitrázsa a modern alkalmazásfejlesztésben mobil platformokon

Degree project / thesis[2] title in English:

Automated Arbitrage for AI technologies in modern application development on mobile platforms

Institutional supervisor: Dr. Katalin Szenes, Bence Tureczki

External supervisor:

Please write the data in printed capital letters!

| Occ. | Date | Content | Signature |
|---|---|---|---|
| 1. | 2021.03.22. | MODELING OF THE PROBLEM SPACE | |
| 2. | 2021.10.22. | DEFINING TEST PLAN VARIATIONS | |
| 3. | 2021.11.22. | COMPARING AI TOOLKITS | |
| 4. | 2021.12.06. | RETROSPECTIVE OF DEVELOPMENT | |

The Consultation log is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Degree project I / Degree project II (BSc) or Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4[3], and is allowed to proceed for reporting / defense[4].

Dated: Budapest, ...DECEMBER 9... 2021.

.....
institutional supervisor

[1] Underline as appropriate
[2] Underline as appropriate
[3] Underline as appropriate
[4] Underline as appropriate

# CONSULTATION LOG

Student's name: Vishav Lakhtia

Neptun code: [redacted]

Specialty: Cloud Computing Services

Phone number: [redacted]

Mailing address (e.g. domicile): [redacted]

Degree project / thesis[1] title in Hungarian:

MI technológiák automatizált arbitrázsa a modern alkalmazásfejlesztésben mobil platformokon

Degree project / thesis[2] title in English:

Automated Arbitrage for AI technologies in modern application development on mobile platforms

Institutional supervisor: Dr. Katalin Szenes, Bence Tureczki

External supervisor:

Please write the data in printed capital letters!

| Occ. | Date | Content | Signature |
|---|---|---|---|
| 1. | 2022.02.15 | PLANNING DATA TRANSFER API MODEL | |
| 2. | 2022.02.22. | DASHBOARDING OF INTERFACE FUNCTIONS | |
| 3. | 2022.05.08. | EVALUATIONS OF SECURITY CONTRADICTIONS | |
| 4. | 2022.05.10. | COMPARISON OF TESTING FRAMEWORK | |

The Consultation log is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Degree project I / Degree project II (BSc) or Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4[3], and is allowed to proceed for reporting / defense[4].

Dated: Budapest, 12 MAY 2022

institutional supervisor

[1] Underline as appropriate
[2] Underline as appropriate
[3] Underline as appropriate
[4] Underline as appropriate

# Contents

# Chapter 1
# Introduction of Target Company

## 1.1 Target company

Erasmus life Budapest is a privately held small business in Budapest that was formed in 2016. For exchange students visiting Budapest, the firm provides services and merchandise.

The company's major activity is the arranging of leisure activities, but they also strive to assist international students in various sectors that come to Hungary.

Students from abroad, not just Erasmus + scholarship university students or Erasmus interns, but also the Stipendium Hungaricum scholarship holders are the major target group. The activities, however, are open to the public. Many foreigners working, residing, or visiting Hungary, as well as Hungarian students assisting foreign students, attend the activities.

As the organization expands daily, technology will be required to help them manage their workload. As a result, they began developing various online and mobile applications to automate their operations. One of the applications is for scanning ticket barcodes to authenticate a guest's admittance into a certain event.

## 1.2 Introduction

Organizing events with a large number of guests and manually verifying the tickets can be a hectic task. For the events organized by Erasmus Life Budapest, tickets can be purchased either online or in the office. While generating the ticket, the system creates a barcode for the ticket number which is present on the ticket. This barcode is usually of the format CODE-39 as the company uses this as the standard barcode format. This code contains the ticket number, and the information of the guest is stored in the company system. Mobile engagement with items and product-centric information services is enabled by scanning these barcodes with smartphone cameras.

Several new mobile apps are produced frequently as a result of current technological advancements and the combination of mobile and barcode technologies. Electronic barcodes are an excellent way to store data in a portable manner. Barcodes are machine-readable data that can be stored, transmitted, and calculated quickly. As they are printed and processed by machines, barcodes have the potential to increase the efficiency and quality of almost all applications. They provide a better degree of precision over conventional data entry. Barcodes are used for a variety of purposes, including object identification, stock tagging, cargo container labeling and packaging, and more. Barcodes not only give a straightforward and low-cost way to show a variety of business data, but they also improve the responsive design by decreasing the number of human inputs.[1]

## 1.3 The goal of development

The development's purpose is to produce an application that can be used regularly by event organizer firms to handle everything easily. To verify tickets, the application will use the camera on the mobile phone to snap live photographs, which will then be analyzed in real-time by Google's machine learning model. The model will extract the ticket number from the QR (Quick Response) code and compare it to the database. SQLite database will be used to record all data, such as history, events, tickets, and many more items.

The company has its database for selling tickets; to synchronize both databases, a REST API (Representation State Transfer) will be developed, which will operate as a bridge between the apps for data transfer. The purpose of designing the application is also to extensively test the functioning of the program using various approaches to provide a satisfactory user experience. In addition, I will investigate many elements of security measures that might improve user and corporate privacy without revealing data on the internet.

I solve the problem by putting together different technologies available into a thoroughly designed android application. Mobile phones can function as scanners, barcode readers, and portable storage media, network access. 2-D Barcodes serve as a tag to connect the physical and virtual worlds when used in conjunction with camera phones.[2]

One of the key concerns of the companies is how software can improve the measurement of their data and its processing. Furthermore, data must be available at all times and from any location. For reading such images as, e.g.: the QR codes, the mobile phone is an ideal tool. Besides mobile phones being small and easy to use, practically everyone has one, making them accessible. This is the main reason which motivated me and the company to develop such an application that will meet the requirements according to the needs of the company. The followings are the preliminary goal of this work:

1. To integrate the automated machine learning models into modern applications.
2. To process the data using a mobile camera and barcodes.
3. To demonstrate how barcodes may be used to create a variety of applications that can aid users in making decisions, analyzing, and comparing data.
4. To illustrate how user data can be kept more secure while being connected to the internet

To effectively create an application that is scalable and extendable in the future according to the needs of the company changes, we need to isolate different components which are thus easily understandable. The below figure 1.1, depicts the overall workflow of the application and how it will communicate with other devices and data which is stored in the database.

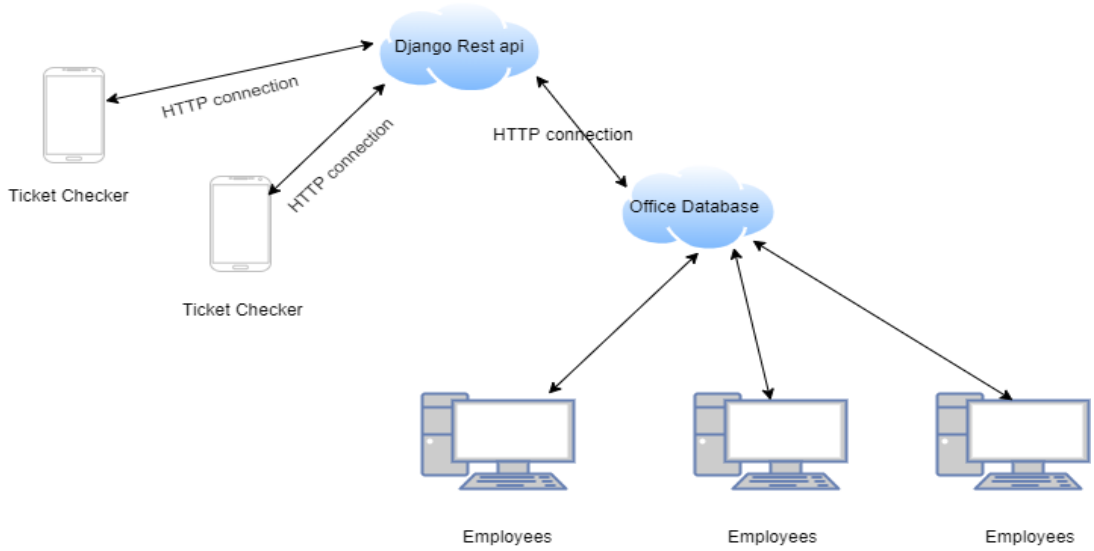


*Figure 1.1 Application Workflow*

As the primary motive of the application is to ease the process of validating the tickets the company sells for the event. To automate the process, the main database of the company which stores the information about the guests, events, and tickets will be connected to the REST API through the endpoints. So, whenever there is a new ticket sold or a new event taking place that information will be passed to the API. As the android application also makes use of the REST API endpoint then the data that is updated will be made available to the android application also. From which employees can validate the ticket by scanning the barcodes.

# Chapter 2

## Tools and Technologies

To create an application that stores data, monitors it, and keeps the user informed of any changes that occur. To create such an application, modern tools and technologies are necessary. This chapter will concentrate on the tools and technology utilized to create this application.

### 2.1 Android Studio

Android Studio is the official IDE (Integrated Development Environment) for android. It's designed specifically for android to help you speed up development and create high-quality apps for every android device. It was announced on May 16, 2013, at the google I/O conference. Starting with version 0.1 in May 2013, Android Studio was in early access preview mode, then moved to beta mode with version 0.8, which was published in June 2014. Starting with version 1.0, the first stable build was published in December 2014. Android studio was designed specifically for android development. So, we can either get an android studio on windows as well as on Mac. It has replaced the eclipse android development as the primary development platform for Google. [3]

| Feature | Android Studio | Eclipse ADT |
|---|---|---|
| Build system | Gradle | Apache Ant |
| Graphical layout | yes | Yes |
| Build variants and multiple APK generation | Yes | No |
| Advanced Android Code | yes | No |
| APK signing and Keystore management | yes | yes |
| NDK Support | Yes | Yes |
| Build dependencies based on Maven | Yes | No |

*Table 2.1: Eclipse ADT vs Android Studio*

### 2.2 Android Architecture

The Android operating system architecture was investigated and came to the solution that android is a Linux-based operating system that consists of a hierarchy of software components separated into five parts and four primary layers, as illustrated in the

architectural schematic below, figure 2.1 . While giving a call interface to the higher encapsulation, each layer of the lower encapsulation

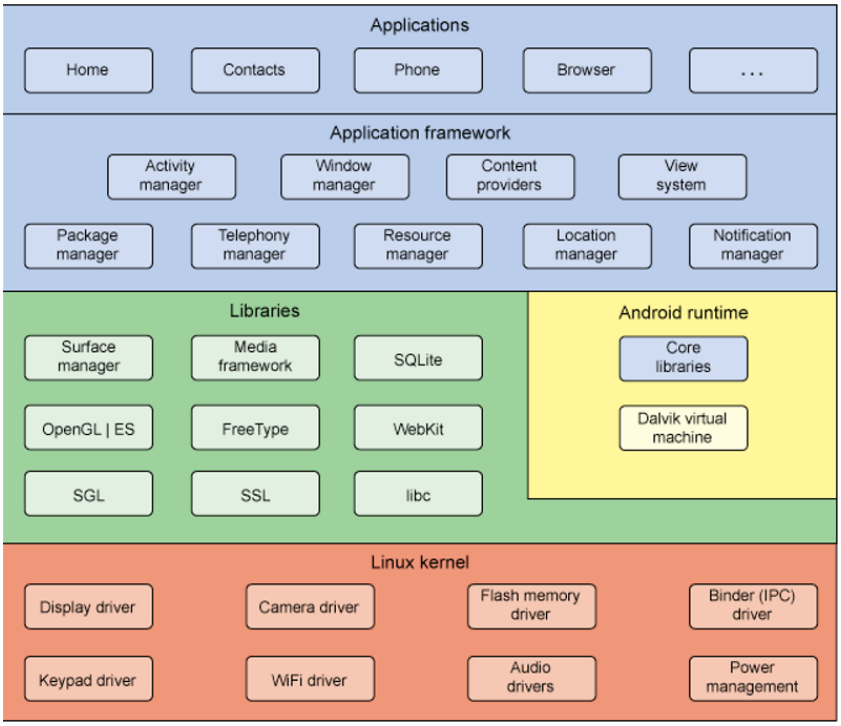


*Figure 2.1 Android Architecture [4]*

### 2.2.1 Applications

At the upper most layer you will be located to all the android applications. You can only accomplish this on this layer if you wish to install your program or build your application code.[5]

### 2.2.2 Application Framework

The application framework simplifies the reuse of its components. If you wish to access the functional components of another android application, you must first determine if the application has been released or not, and you must adhere to the framework's security. Similarly, users can utilize this reuse mechanism to replace program pieces. Moreover, all the API frameworks are very easily accessible by the developers and can modify according to their requirements[4].

### 2.2.3 Libraries and Android Runtime

So as referenced in the Diagram above, the library part is divided into the two main elements android runtime and libraries. The android system library strengthens the application framework and serves as a link between the application framework and the Linux kernel. This system library has been expanded in C or C++. These libraries can also be accessed by the various components of the android system. The application framework provides services to developers. As far as android runtime is concerned core libraries provide the core libraries of java and for specific development, we have the DVM which stands for Dalvik virtual machine.[5]

### 2.2.4 Linux Kernel

The last layer is considered to be the Linux kernel in android architecture and is considered the heart of the architecture. It mostly focuses on the management of the driver like camera, BT (Bluetooth), audio, etc. Furthermore, the kernel handles all the things that Linux excels at, such as networking and a huge number of device drivers, which make connecting to peripheral hardware a breeze.[4]

## 2.3 Django

Django is the high-end framework for python that will enable the rapid development of secure and easily maintainable websites or applications. It is free and open-source, has a robust and active community, excellent documentation, and several free and paid-for support alternatives[6]. Django is a complete product that the developer needs while developing. It is compatible with any client-side framework and can send material in nearly any format (including HTML, RSS feeds, JSON, XML etc) [7]. Moreover, it's a most secure framework in that it secures the user credentials in the right manner that instead of the cookies it contains a key, and the data is stored in the database. It follows an architecture that is independent of the others and that results in proving Django as scalable. Django Rest Framework is a powerful and reliable toolkit that I have used for creating the Web API. In the later chapter, I will go through the implementation of the REST API in detail.[8]

## 2.4 Amazon Web Services

AWS stands for Amazon Web Services, which is a public cloud developed by Amazon. It is a secure cloud service platform that offers plenty of services, compute power and content delivery. AWS's major offerings include EC2 (Elastic Computing), Amazon's virtual machine service, Glacier, a low-cost cloud storage service, and S3, Amazon's storage system. It helps us to run the web applications servers in the cloud that host websites so that it is available all around the world. Moreover, it is a highly secured cloud that keeps an eye on your files and can be accessed anywhere. AWS used the databases like MYSQL, PostgreSQL, and Oracle servers to store the information. I have used EC2 Instance which is most popular across this product which are just virtual machines in the cloud on which we have OS-level control. And for the storage S3, a Simple storage service is used where we can store the objects like files, folders, images, documents, etc. [11]

## 2.5 Docker

Docker is an open-source platform for running applications and making the development and distribution process easier. Docker-built programs are bundled with all necessary dependencies into a common form known as a container. These containers continue to execute in isolation on top of the kernel of the operating system. The additional layer of abstraction may have an impact on performance. So, it provides the facility to automate

the applications when we deploy them into containers. Docker is meant to provide a speedy and lightweight environment in which code may be executed quickly, as well as an added facility of the competent work process to take the code from the computer for testing before production. Docker has collaborated with AWS to give developers easy delivery of modern applications to the cloud. AWS services like AWS Fargate, Amazon ECS, etc make the developer life easy to run and manage the docker container at scale [9]. A detailed description of how I delivered my application into docker will be described in future chapters.

### 2.5.1 Virtual Machine vs Docker

Virtualization is an older principle that has been employed in cloud computing since IaaS (Infrastructure as Service) was regarded as a critical approach for system construction, resource provisioning, and multi-tenancy. Virtualized resources are crucial in resolving difficulties utilizing cloud computing's primary methodology. The below figure 2.2, the architecture of the virtual machine is described.

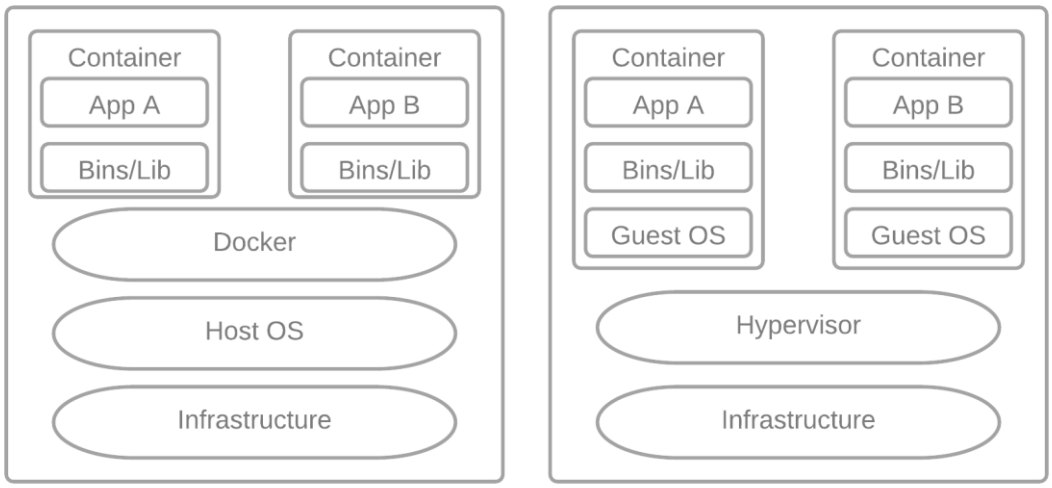


*Figure 2.2 Docker vs Virtual Machine [9]*

The hypervisor sits between the host and guest operating systems. It is a virtual platform that manages many operating systems on the server. It communicates with the operating system and the CPU(Central Processing Unit). It is divided into two sections by virtualization: the first is Paravirtualization, and the second is Full Virtualization. Containers are an abstraction at the app layer that groups together code and dependencies. Multiple containers can operate on the same computer and share the OS kernel, each executing as an isolated process in user space. Containers take up less space than virtual machines (container images are often tens of megabytes in size), can accommodate more applications, and need fewer virtual machines and operating systems.

| VM’S | CONTAINERS |
|---|---|
| Heavyweight | Lightweight |
| Limited performance | Native Performance |
| Each VM runs in its own OS | All containers share the host OS |
| Start-up time slower | Start-up time in Milliseconds |
| Require more memory | Require less Memory |
| More secure | It's less secure than VM’S |

*Table 2.2 VM vs Containers [10]*

# Chapter 3
## Design Patterns

Design patterns are conventional solutions to common challenges in software development. They can be considered as blueprints that you can alter to remedy recurring design issues in the code. In other words, it is a template for solving an issue that may be applied to a variety of scenarios.
The concept of Design pattern was first introduced in software development when four authors Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides released a book called “Design Patterns - Elements of Reusable Object-Oriented Software”. Gang of Four (GOF) is the collective name for these authors.

### 3.1 Types of design pattern

There are 23 design patterns that may be categorized into three groups, according to the design pattern reference book Design Patterns - Elements of Reusable Object-Oriented Software: Creational, Structural, and Behavioural

| **Creational Patterns** | **Structural Pattern** | **Behavioural Patterns** |
|---|---|---|
| These design patterns give a mechanism to construct things without obscuring the creation logic. This allows the software more freedom in determining which objects are required for a specific use case. | It's all about Class and Object construction in these design patterns. Inheritance is used to construct interfaces. These patterns define how to put objects together to get additional functionality. | These patterns mainly focus on the communication of the class objects. They are primarily concerned with object-to-object communication. |

*Table 3.1 Types of Design Pattern*

There are plenty of advantages of implementing the design pattern in software development. They make it easier to create highly coherent modules with less coupling. They separate any potential fluctuation in system needs, making the whole system easier to comprehend and manage. Implementing the design patterns will give you a technique to address software development problems. Next, we will discuss the design patterns that were used when developing this application.

### 3.1.1 Design patterns used

As per guidelines by Google, it is recommended to integrate the repository with the MVVM design pattern in the application.

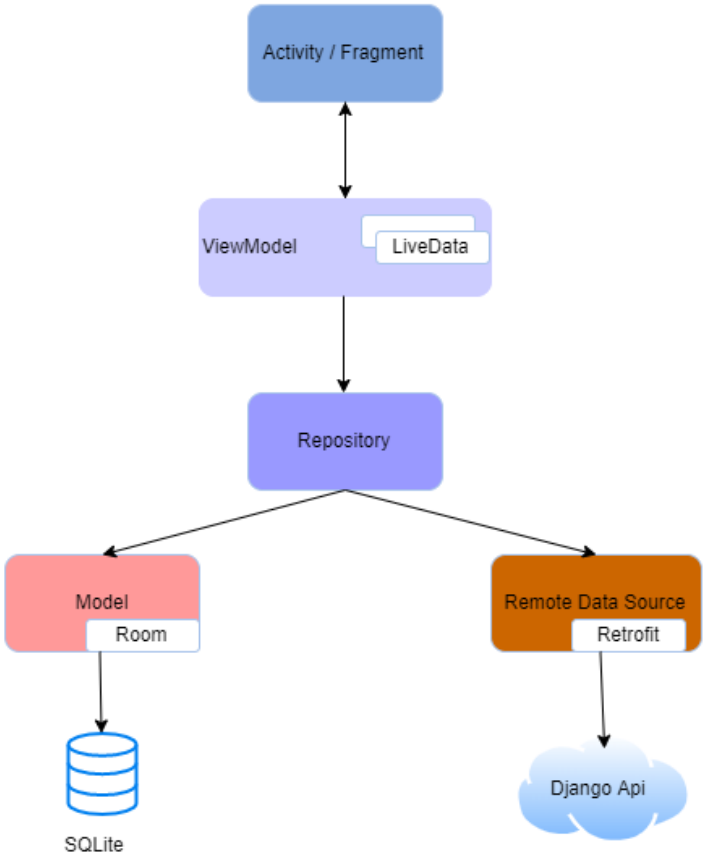


*Figure 3.1 Design Pattern*

## 3.2 MVVM (Model-View-View Model)

The MVVM pattern is a software architecture paradigm that abstracts the business logic from the user interface controls. This design pattern was introduced by Ken cooper and John Gossman. It is classified as a type of structural design pattern. Beyond attaining segregation and the effectiveness gained after implementation, the main feature is permitting genuine separation between View and Model.[12]

Like other design patterns, MVVM contributes to organizing the code and separating applications in different modules. Thus, by doing this it fastens the development process. In the near future, it also makes it easier to update, or reuse the code. MVVM makes it easier to understand the code for other developers who can be new to the project.

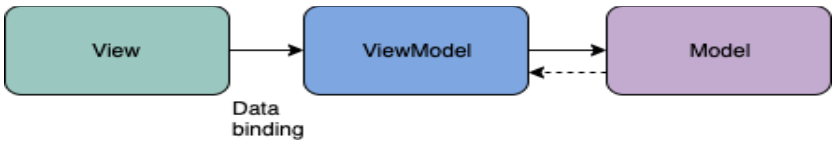


*Figure 3.2 MVVM [12]*

From figure 3.2, it is quite predictable that the MVVM pattern is divided into different sub-modules which are then connected to send updates to data or the user interface.

### 3.2.1 Model

The abstract form of the data structure and qualities that are necessary for specific data is called a model. It is extremely usual for one model to refer to another. The Model layer is responsible for validating and acting on the data it gets from the user. If the validation is successful, the data should be saved in the database; otherwise, the user should be notified of the issues.

### 3.2.2 View

This is the user-friendly interface, in other words, it is the collection of user interface components that displays data to the user. This is the interface that the user interacts with. Users can alter the data by clicking buttons or using swipe movements. This module is responsible for receiving user input and acting on it. To make the view which is presentable to the user the Application uses XML language.

### 3.2.3 View Model

The view model is a presentation of a view that reveals the public attributes and commands. This layer links the View and Model layers together. This layer ties the data to the display and uses Live Data to transmit changes whenever the database is updated. The view then receives Live Data and monitors the changes.

## 3.3 Repository pattern

"*Repository Mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects.*"

**by Edward Hieatt and Rob Mee**

As per the above definition, the repository pattern links the data from different sources. In context with our application, the repository pattern provides the single point entry to data. Thus, it becomes the responsibility of the repository which data should be passed when requested. Thus, it can be a key to the problem of managing complex data and data queries. A repository pattern is a special kind of Facade Pattern that is a structural design pattern. It isolates the data access logic from business logic. Interfaces are used to communicate between the data and business layers.[13]

# Chapter 4
# Database

Databases are essentially data containers. We may also claim that a library is a database of books since it holds books. Databases, on the other hand, are computer systems that save, organize, safeguard, and distribute data. A database system is an integrated collection of connected files, as well as information on the data's interpretation. A database system is essentially nothing more than a computer-based record-keeping system, that is, a system whose fundamental aim is to store and retain information/data.[14]

A database management system (DBMS) is a piece of software that allows users to access data stored in a database. The goal of a database management system (DBMS) is to provide a simple and efficient way of defining, storing, and retrieving data from a database. The database management system (DBMS) communicates with the application programs so that the database's data may be accessed by numerous applications and users. Furthermore, the DBMS maintains centralized database control, prevents fraudulent or unauthorized users from accessing data, and assures data privacy[15].

## 4.1 Database component

There are basically five main components of the database.

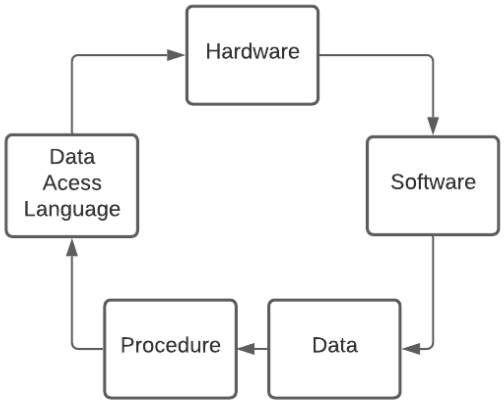


*Figure 4.1 Database Components[14]*

**Hardware:** Physical, electronic equipment such as computers, I/O devices, storage devices, and so on comprise the hardware. This serves as a bridge between computers and real-world systems.[14]

**Software:** This is a collection of applications used to administer and govern the database as a whole. This comprises the database software, the operating system, the network

software used to distribute data among users, and the application applications needed to access data in the database.[14]

**Data:** Data is a raw and unstructured fact that must be processed in order to be relevant. Data may be basic and disorganized unless it is arranged. In general, data includes facts, observations, perceptions, numbers etc.[14]

**Procedure:** Procedures are a list of procedures and rules that assist you in using the database management system. It is the process of developing and administering a database using defined techniques so that the users who operate and maintain it may be guided[14].

**Database Access Language:** It is used to access data in and out of the database[14], as well as to input new data, change current data, and retrieve necessary data from the database management system. In a database access language, the user writes certain particular instructions and sends them to the database.

### 4.1.1 Why do we need database?

On the one hand, the companies store every business- and some of the personal, sensitive data in databases. On the orher hand, large amounts of data can be processed in data bases, even on a daily basis. Using spreadsheet or any other tool is not so feasible. If correctly handled, the information stored in the databases is traceable, and correct, at the same time. For updating the data data manipulation languages like SQL (Structured Query Language), NOSQL etc are available. Moreover, there are various methods that ensure such data security requirements, as an aspect of confidentiality, that only authorized users are allowed to access the database. For the research of our data database we can use such query languages that let us to search in the database and to perform different operations.

The database schema for our application is shown in figure 4.2. This diagram depicts the kind of information and details that our program will require in order to function properly and efficiently. We need to store tickets, events, histories, ticketlists, and much more information, as shown in the figure. This database schema serves as the foundation for both our android application and REST API.

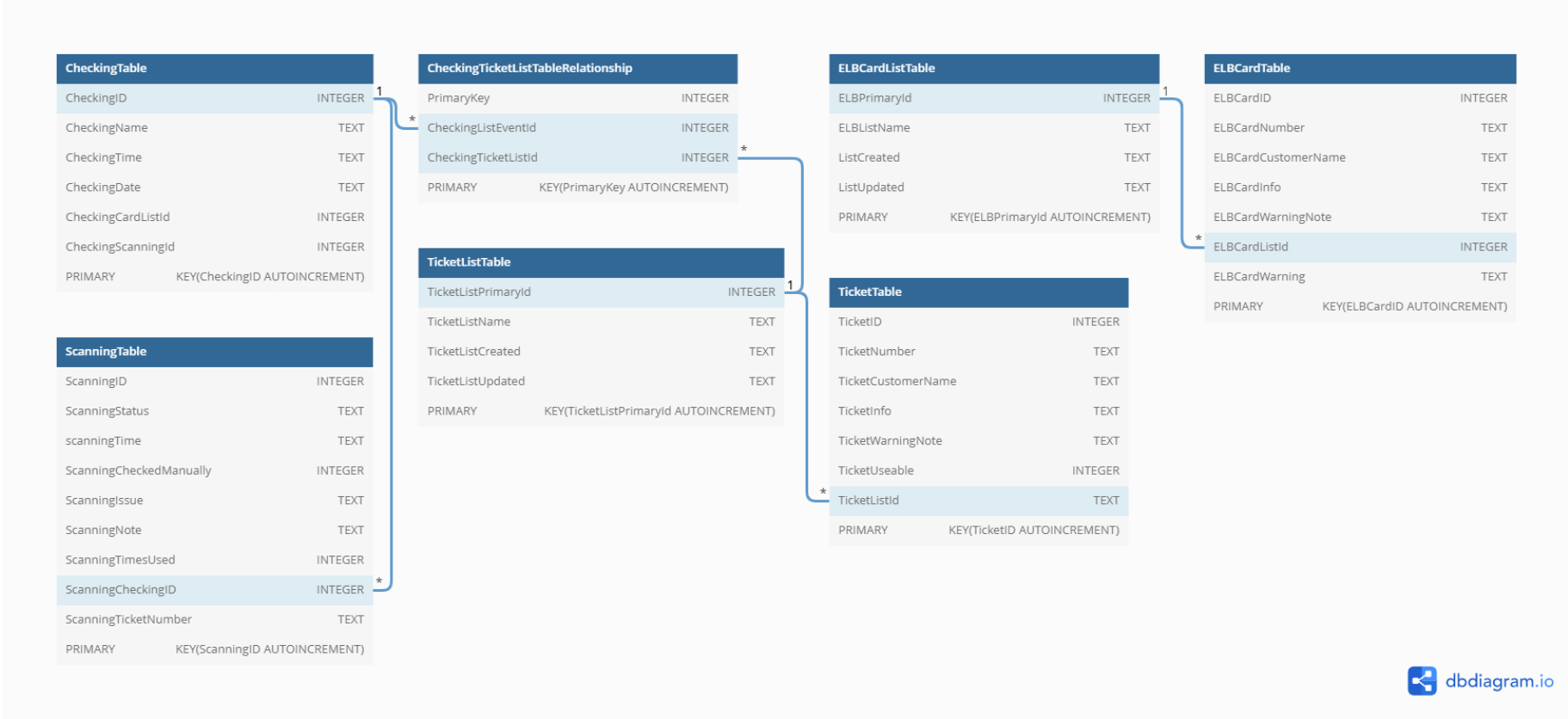


*Figure 4.2 Database Schema*

## 4.2 Relational vs Non-Relational Database

| Non-Relational Database | Relational Database |
|---|---|
| It does not store data in tabular schema | It stores the data into rows and column like a spreadsheet |
| It has high data throughput | It has low data throughput |
| We can increase performance by enhancing caching data | Caching can only be done via special infrastructure |
| We can insert data at any time without defining a schema | Data must be fit in predefined tables or structure |
| We can store single index, key value | Indices are available on multiple columns |
| Provide the BASE properties | Provides ACID properties |
| It is highly scalable | It is less scalable |
| It mostly compromises on consistency | It provides much better consistency than non-relational database |
| Data duplication is allowed which threatens data integrity. | The relational database design process eliminates record duplication, |
| Searching in non-relational database is much harder that are based on multiple criteria | Relational database organization allows a query language SQL to move around the database |
| For example: HBase, MongoDB, Cassandra | For example: MySQL, SQL Lite, PostgreSQL |

*Table 4.1 Relational vs non-Relational*

## 4.3 Non-Relational database types

Non-relational databases, unlike relational databases do not store data in tabular form. They generally store their data in form of key-value pairs or graph employing vertices and edges. There are different types of non-relational database types which are discussed below.

1. **Columnar Database:** As the name implies, data in this form of database is kept in columns rather than rows (like in regular relational database). This type selects only the necessary columns and performs analysis without storing a huge number of rows in memory [16]. It is efficient but inserting or updating data in the future will be tough. For instance, HBase and Cassandra.

2. **Document Database:** In this sort of database, data is kept in the form of key-value pairs within documents. Arrays, ints, and other data types can be preserved within a document [16]. A collection is a group of documents. Typically, the same type of document is kept within a collection to make it more efficient and scalable. There may be several duplicates of the same document in another collection. As a result, it is often difficult to update each document whenever there is a change in any of the documents. For instance, Firebase Firestore and MongoDB.
3. **Graph Database:** A graph database is often used to store data in a graph format. They store data and build relationships using nodes and edges. Graph databases are extremely efficient in searching for data or relationships within a database[16]. However, when the database increases in size, it might become challenging to maintain and scale the data as it evolves into a more complicated graph. As an example, consider Neo4.
4. **Key-value database:** This is the most fundamental non-relational database. Data is saved in this manner as a key value pair, much like a dictionary. Where each key is paired with a unique value [16]. For instance, consider the Firebase database and Amazon DynamoDB.

## 4.4 SQLITE

SQLite is an in-process library that creates a transactional SQL database engine that is self-contained, serverless, and requires no setup. SQLite's code is in the public domain, which means it can be used for any purpose, commercial or personal. SQLite is the most extensively used database on the market, with an uncountable number of applications, including some high-profile projects. I have used SQLite relational database in my application because it is simple, lightweight, performance with simple architecture and highly readable code.

### 4.3.1 SQLite architecture

To Understand in detail about the SQLite Architecture I have created the flow diagram about the structure.

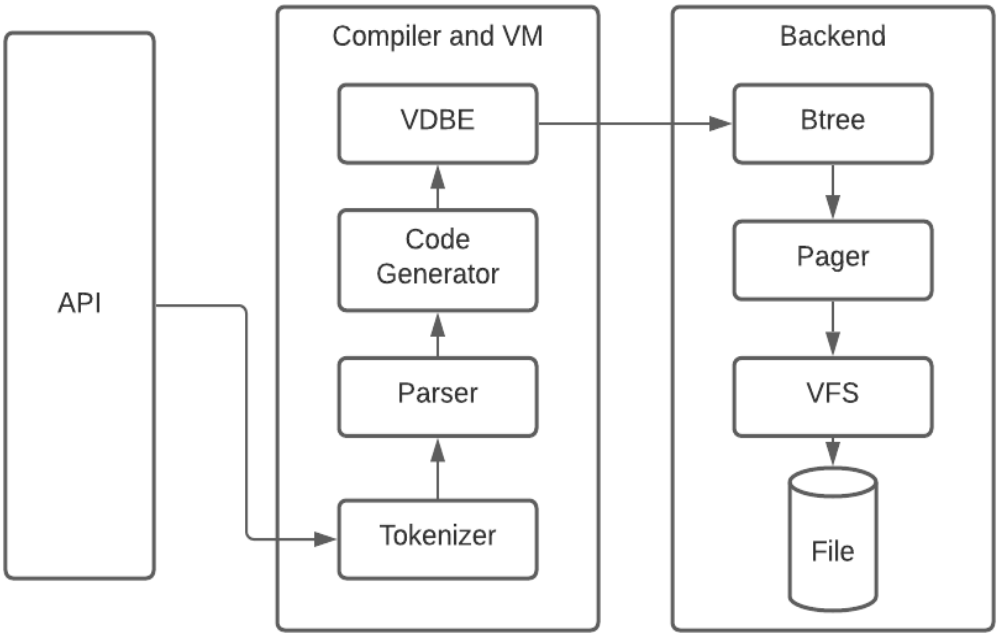


*Figure 4.3 SQLite Architecture [14]*

SQLite architecture as seen in figure 4.3, is basically split up into two different sections named core and backend. Core section basically focuses on Interface, Code Generator, Tokenizer and VS machine which create execution orders. To access the file system, the backend comprises B-tree, Pager, and an OS interface. The compiler, which consists of a tokenizer, parser, and code generator, creates a set of opcodes that operate on a virtual machine. [17]

## 4.5 Room

The Android Jetpack includes a persistent library called Room. The room persistent library has many advantages over raw SQLite. It is built on top of SQLite. The latest update is 2.3.0. SQL queries can also be validated at compile time. This implies that if a SQL query error occurs, an application will not build. This protects the developer from having to deal with run-time issues.

To better understand the implementation of the Room DB in your application, I will talk about room database main components and steps to get it initiated.

1. Create Entity - A mapping table will be created in the database for each model class that has foreign keys, indices, primary keys, table name annotations.

```
@Entity(tableName = "TicketListTable")
public class TicketListTable implements Serializable {

   @PrimaryKey(autoGenerate = true)
   @SerializedName("id")
   @ColumnInfo(name = "TicketListPrimaryId", index = true)
   private long id;

   @ColumnInfo(name = "TicketListName")
   @SerializedName("ticketListName")
   private String ticketListName;

   @ColumnInfo(name = "TicketListCreated")
   @SerializedName("ticketListCreated")
   private String ticketListCreated;

   @ColumnInfo(name = "TicketListUpdated")
   @SerializedName("ticketListUpdates")
   private String ticketListUpdated;

   public TicketListTable(long id,String ticketListName, String
ticketListCreated, String ticketListUpdated) {
       this.id = id;
       this.ticketListName = ticketListName;
       this.ticketListCreated = ticketListCreated;
       this.ticketListUpdated = ticketListUpdated;
   }

}
```

*Example 4.1 TicketListTable Entity*

Use the annotations to turn the class into a database Entity and supply the table name. Following that, describe the properties of the class that will be used as the table's column, as well as its data type and name.

2. DAO: DAO is the function that we need after we created our table. These functions are used to query our table. To query the database, the DAO interface includes three primary variables.

- @Insert is a function that inserts data into a table. The database is updated with the item supplied into the method that is annotated with this function.

```
@Insert(onConflict = OnConflictStrategy.IGNORE)
void insert(TicketTable ticketTable);
```

*Example 4.2 Insert DAO*

- @Query - When writing a SQL statement, this is the syntax to use. As an argument, it receives a SQL statement. This annotation was used to acquire all of the to insert items and to remove one of them. Because there is no specific annotation for deleting a single item, we use @query to remove it.

```
@Query("SELECT * FROM TicketTable Where TicketNumber =
:tickNumber")
TicketTable searchTicket(String tickNumber);
```

*Example 4.3 Query*

- @Delete - This annotation is used to delete the table or the specific entry in the table

3. Create the Database: In this we will define the abstract class that will extend the Room Database because the class needs to have a singleton feature.

```
@Database(entities = {CheckingTable.class, … TicketTable.class},
version = 13)
public abstract class AppDatabase extends RoomDatabase {
   ...
   public abstract TicketListDao ticketListDao();
   public static AppDatabase getDatabase(final Context context){
       if(INSTANCE == null)
       {
           synchronized (AppDatabase.class)
           {
               if (INSTANCE == null) {
                   INSTANCE =
Room.databaseBuilder(context.getApplicationContext(),
                           AppDatabase.class, "app_database")
                           .openHelperFactory(factory)
                           .allowMainThreadQueries()
                           .build();
               }
           }
       }
       return INSTANCE;
   }
}
```

*Example 4.4 Database Creation*

The above class, Example 4.4, restricts the use of the Database to a single instance in the application. Developers can construct a database instance by calling Room.databaseBuilder(). And allowMainThreadQueries () allows queries to be run on the main threads. This abstract class should be created as a database by using annotations and specifying table classes to establish the database's structure.

4. Create a repository: When we have multiple sources of data, we use repositories that is remote and local sources. This class will need to hold reference to the data access object.

```
public class ScanningTableRepo {
   private ScanningTableDao scanningTableDao;
   private LiveData<List<ScanningTable>> allHistory;
   private DataService dataService;

   public ScanningTableRepo(Application application)
   {
       AppDatabase db = AppDatabase.getDatabase(application);
       scanningTableDao = db.scanningTableDao();
       allHistory = scanningTableDao.getAllHistory();
       dataService =
RetrofitConnection.getRetroFitInstance().create(DataService.class);

   }
  …

  …

   public void insert(List<ScanningTable> scanningTables)
   {
       try {
           scanningTableDao.insert(scanningTables);
       }
       catch (Exception e)
       {
           Log.i("scanning exception", e.getMessage() +"");

       }
   }
```

*Example 4.5 Scanning Table Repo*

This repository's constructor retrieves the database instance and creates the Dao object to conduct the supplied queries. The insert () function gets the list of scanningTables and inserts them into the database.

# Chapter 5
# Machine Learning

## 5.1 Introduction

Machine learning is a subset of Artificial Intelligence that uses data presented to software to efficiently anticipate output without expressly training it to do so.

Machine learning is a critical component of the rapidly expanding discipline of data science. Models are implemented to generate classifications or predictions using statistical approaches, revealing crucial insights in data mining initiatives. These insights are then used to drive decision making within programs and enterprises, hopefully influencing key growth indicators. As big data expands and grows, so will the industry demand for data analysts, who will be required to aid in the discovery of the most important business issues and, ultimately, the data to answer them.[18]

Machine learning models are being employed in almost every industry. Models are increasingly utilized to forecast a company's future growth or stock price predictions. In apps, machine learning may assist software in understanding user behaviour and personalizing the program accordingly.

In this application, Google ML kit is utilized to decode the barcodes and extract the required information. Using ML kit resulted in optimising the scanning of barcodes without much error. Apart from this, it is perfectly documented how one can use this kit in their application. I preferred Google’s ML kit as it is backed by one of the most advanced companies and they continuously optimized the performance of their models to meet the requirements and to decrease the probability of encountering the errors. Further I will try to give the overview insight of Google ML kit and how it is embedded in this application.

## 5.2 Google ML kit

ML Kit is a sophisticated and easy-to-use tool that delivers Google's machine learning capabilities to mobile developers. With solutions that are tailored to operate on device, you can make your iOS and android apps more entertaining, customized, and useful.[19]

### 5.2.1 Why ML kit?

Some of the reasons why it is recommended to process the information or embed the machine learning models are stated:

1. **Designed with mobile in Mind**
   Google explicitly made this kit especially for mobile developers keeping optimization in mind. The ML Kit's processing takes place entirely on the device. This makes it quick and allows for real-time applications such as processing camera information. It may also be used to process photos and text that must be kept on the device while offline.[19]

2. **Google experience was used**
   Because these machine learning models are conceived, developed, and built by Google personnel with years of machine learning and optimization experience.[19]

3. **Integration and Ready to Use**
   Because this kit was designed exclusively for android apps, it's straightforward for developers to integrate and understand how things work. Its ready-to-use functionality allows developers to focus on developing rather than training models, since they can simply incorporate these models by following the guidelines.[19]

The ML kit is further subdivided into categories, which are explained below.

1. **Vision API**
   This API is used to analyse images and to label. Apart from this, it can decode the barcodes and extract the information from them.

2. **Natural Processing API**
   APIs for identifying and translating between 58 languages, as well as providing response ideas.

Each of these APIs has a variety of models that may be accessed through endpoints. These APIs will offer the outcome to the user based on the information provided by the user.

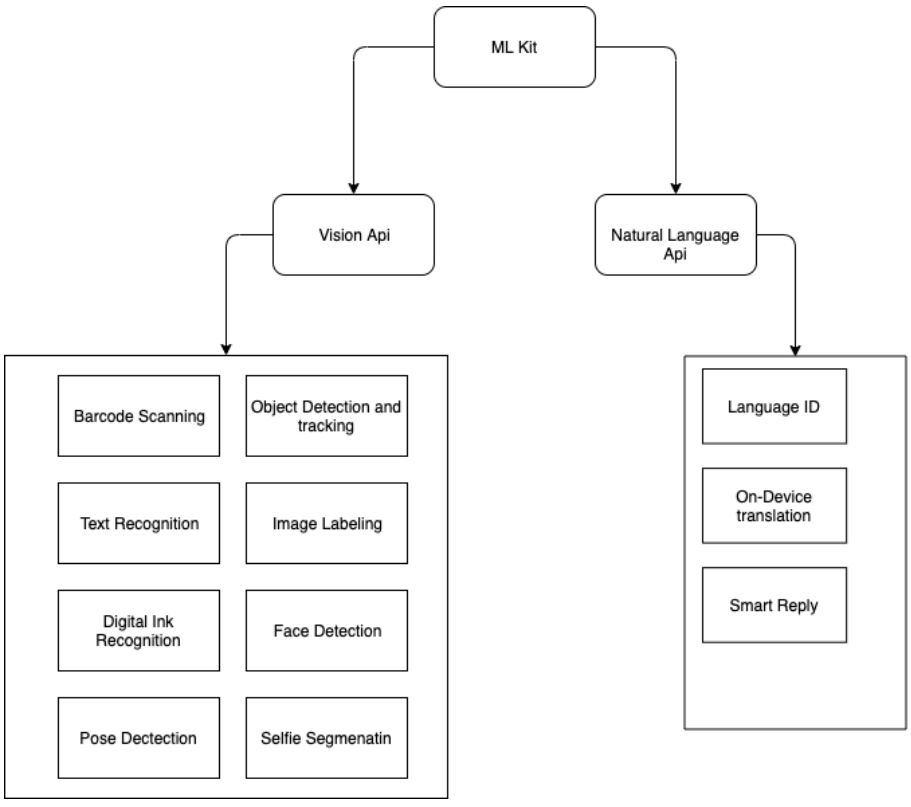


*Figure 5.1 Google Ml Kit Models*

As we all know, each of these models has its own set of applications. The Barcode Scanning model, which is utilized in the application to extract information encoded in barcode or QR codes, will be the subject of this summary. This API works with a variety of barcodes.

### 5.2.2 Barcode scanning model

You may read data stored in most popular barcode codecs using ML Kit's barcode scanning API. The processing of barcodes takes place on the device itself, and there is no need for a network connection.[20]

Barcodes are a quick and easy method to get data from the real world into your app. Structured data, such as contact details or WIFI details, can be encoded using 2D formats such as QR codes. Because ML Kit recognizes and parses this data automatically, your application can react smartly when a user scans a barcode.[20]

Some of the features of this models: -

1. Read most common barcode formats.
2. Format detection is done automatically.
3. Information that is organized is retrieved.
4. Any rotation can be used.

It is a complete bundle with several features. This model also has several flaws that should be considered before employing it in applications. This vision model doesn't work with 1D barcodes or the barcodes which are encoded with FNC2, FNC3 or FNC4.

You may incorporate barcode scanning in one of two ways: by including the model in your application or by utilizing an unbundled model that relies on Google Play Services. Your app will be smaller if you choose the unbundled approach, but the bundled model provides performance advantages over the unbundled model.

**Implementation**

This section will show you how to integrate the machine learning kit into your android app. The packaged version will be used since it is significantly quicker and more accurate. To begin, a barcode scanning dependency must be established in the application via which we will access all of the model's capabilities. We will utilize the CameraX library to capture the image in real time, which will be detailed later. Output from the CameraX library will be used as input to our ml model.

```
dependencies {
   implementation 'com.google.mlkit:barcode-scanning:17.0.0'
}
```

*Example 5.1 Installing Dependencies*

We may define the type of barcode our program should scan using the code above. The speed of the detector will be increased by selecting certain barcodes. Then, the client of the barcode scanner can be initiated and pass the options as arguments.

```
BarcodeScannerOptions barcodeScannerOptions =
        new BarcodeScannerOptions.Builder()
                .setBarcodeFormats(
                        Barcode.FORMAT_CODE_39
                ).build();
BarcodeScanner scanner = BarcodeScanning.getClient(barcodeScannerOptions);
```

*Example 5.2 Barcode Scanning Options*

Following that, we can transfer the picture we acquired from the CameraX library to the scanner, which will begin analyzing the image and attempting to find the barcode. We can retrieve information from a barcode once it has been successfully recognized.

```
Task<List<Barcode>> result = scanner.process(image)
        .addOnSuccessListener(this::processResult)
        .addOnFailureListener(e -> Toast.makeText(getContext(),
e.getMessage(),Toast.LENGTH_SHORT).show())
        .addOnCompleteListener(task -> {
            mediaImage.close();
            imageProxy.close();
        });
```

*Example 5.3 Processing Image*

Finally, we can check whether the information (in this scenario, the Ticket Number) is recorded in the database, validating the guest's admittance into the event.

## 5.3 Alternatives

### 5.3.1 Zxing

Zxing also known as Zebra Crossing is a Java-based barcode image recognition package with portability to other languages. It can read 1D product barcodes, 1D industrial barcodes, and 2D barcodes.[21]

**Advantages of Zxing**

1. It's possible to utilize it with Intents.
2. It may be incorporated in an activity for more sophisticated user interface and logic customisation.
3. Scanning may be done in either landscape or portrait orientation.
4. For quick start-up, the camera is controlled in a background thread.[21]

However, because Google's machine learning kit has been proven in a variety of scenarios, it is more accurate, quick, and simple to integrate into the application. In comparison to Google's ML kit, the documentation for the Zxing library is not as clear.

# Chapter 6
## Core Application Layer

This chapter will focus on the development of REST API and android applications. Later on, the process of integrating REST API will be explained in detail.

### 6.1 REST API core

API stands for Application Programming Interface. REST, which stands for "Representational State Transfer," is a set of principles for describing and retrieving data in your system as interconnected objects and collections. In other words, REST API acts as the bridge between backend and frontend of the application whose prime role is to transfer data to frontend from backend [22]. In this application, REST API will retrieve the data from the database and provide it to the barcode application. Apart from this application can save data to the database via REST API.

Applications can send different types of requests through API, which are described below.

- **GET**: - This request is used to fetch the data from the REST API for the specific resource.

- **POST**: - This request is used to send the data to the REST API and save it to the database for future usage.

- **DELETE**: - This request is used to delete the specific resource from the database.

- **UPDATE**: - This request is used to update the data of specific resources. In this method, User must provide the identifier to identify the data which has to be updated.

#### 6.1.1 Components of django API

**Models:** Models are the heart of the Django projects; it basically defines the fields and behaviour of the data. Generally, one model represents the table in the database. Models have the attributes which define the constraints and datatype of the database field. It is the object type which is saved into the database [23].

```
class TicketTable(models.Model):
   ticketNumber = models.CharField(max_length=50)
   ticketCustomerName = models.CharField(max_length=50)
   ticketInfo = models.CharField(max_length=100, blank=True)
   ticketWarningNote = models.CharField(max_length=100, blank=True)
   ticketUseable = models.IntegerField()
   ticketWarning = models.CharField(max_length=50, blank=True)
   ticketListId = models.ForeignKey(TicketListTable,
on_delete=models.CASCADE, related_name="listTable_tickets")

   def __str__(self):
       return self.ticketNumber

   class Meta:
       app_label = "backend"
       db_table = "TicketTable"
```

*Example 6.1 Ticket Table Model*

The above code is the model for the Tickets, thus inherited from the models.Model class which possesses the different attributes for the models' field. Each field has its data type described which it will store while saving the data. Developers can also define the relations between two tables to query the data easily. In the class meta, the app name this table belongs to, and the verbose name of the table is mentioned.

**Serializers**: Complex tasks, like query sets and model objects, may be serialized to pure Python object types, which is then readily transformed into JSON, XML, or other content types. Deserialization is also handled by serializers, which allows interpreted data to be turned back into complex types after analysing the data received [24]. In our scenario, the API receives the JSON object from the android application which must be converted into the python object to save it in the database, this is deserialization. But when the API fetches the data, it is a python object which has to be interpreted in the JSON object which will be handled by our android application, this process is Serialization.

```
class TicketSerializer(serializers.ModelSerializer):
   id = serializers.ReadOnlyField()

   class Meta:
       model = TicketTable
       fields = ['id', 'ticketNumber', 'ticketCustomerName', 'ticketInfo',
                 'ticketWarningNote', 'ticketUseable', 'ticketWarning',
                 'ticketListId']
```

*Example 6.2 Ticket Serializer*

The above written code is the Serializer class for the ticket table, which inherits the serializers.ModelSerializer class because our a API will serialize the python object to json

object. In subclass meta, model name is defined which has to be serialised and the fields of the tables which have to be considered by process the objects.

**Views:** The views are the functions or the class which handles different types of the request to the website or the API. In short, Views return responses to the requests made by the user. The response can vary according to the needs of the developer. It can be an HTML page or the redirect to some other or some json object or anything else [25]. According to the coding conventions defined by Django, the views should be written in the views.py file. The example of how the view looks is given below.

```
# TicketList class to return data from database or to save data.
class TicketList(GenericAPIView):
    serializer_class = TicketListSerializer

    def get(self, request):
        ticketsList = TicketListTable.objects.all()
        serializer = TicketListSerializer(ticketsList, many=True)
        return Response(serializer.data)

    def post(self, request):
        serializer = TicketListSerializer(data=request.data)
        return saveToDb(serializer)
```

*Example 6.3 Ticket List Model*

The above view handles the requests which are related to the Tickets Table. The view is inherited from the GenericAPIView which specifies that this is the API view and provides the actions for the standard list and detailed views. As, it is crystal clear that this class handles GET, and POST requests made to the ticket table through the API. For the GET request, the get method fetches all the tickets which are stored in the database and passes the list to Serializer which then serializes the python object to JSON response. After that, the serialized data is returned as Response to user which can be interpreted according to the needs. Whereas the post method, it receives the data from the user request which is in our case JSON object. Thus, Serializer then deserializes the object to a python object and the data is saved to the Database.

**URLs**: This file stores the URL pattern of how the REST endpoints will be exposed to users. These are the points to which the user makes a request and the view related to that will perform this task based on the type of request received. From below we can see that methods related to the ticket table can be accessed from the ticket endpoint of our URL.

```
urlpatterns = [
    path('ticketLists/', views.TicketList.as_view()),
    path('ticketLists/<str:pk>', views.TicketListDetail.as_view()),
    path('ticket/', views.Ticket.as_view()),
    path('ticket/<str:pk>', views.TicketDetail.as_view()),]
```

*Example 6.4 URL Patterns*

## 6.2 Deployment

As indicated in Chapter 2, AWS and Docker will be utilized to deliver the REST API. An introduction to the topic was provided in Chapter 2. This part will solely cover the implementation and deployment procedure. To prepare our REST application for deployment, we must create the docker file.

### 6.2.1 Dockerfile

This file is placed in the application's root folder. To begin, it utilises a pre-built image (in our case, it is python). The docker file for our application is provided below.

```
FROM python:3.8.9
ENV PYTHONUNBUFFERED=1
WORKDIR /BarcodeApi
ADD . /BarcodeApi
COPY requirements.txt /BarcodeApi/
RUN pip install -r requirements.txt
COPY . /BarcodeApi/
```

*Example 6.5 Dockerfile*

The code above generates a docker image of the REST API. It retrieves the python image from the Docker registry and builds the BarcodeApi folder within it. The project files are then moved to the image folder, and the prerequisites for running the container are installed.

1. Create the EC2 Instance on AWS cloud. For our application, I initiated the Linux ec2 instance.

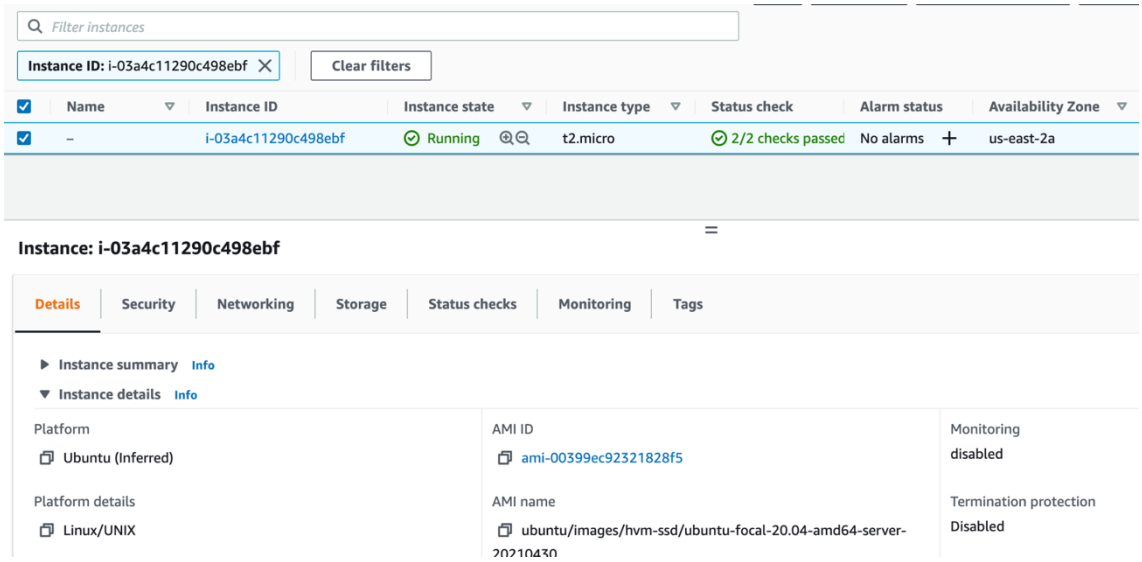


*Figure 6.1 AWS EC2 Instance*

2. Transfer all the application files to AWS cloud using the SSH (Secure Shell) connection or any third-party software like FileZilla or another.

```
ssh -i "restapi.pem" ubuntu@ec2-3-13-208-251.us-east-
2.compute.amazonaws.com
```

*Example 6.6 SSH Connection*

3. Build the Docker image using the below command.

```
$ docker build .
```

*Example 6.7 Build Command*

4. And finally run the image in the container.

```
$ docker-compose up
```

*Example 6.8 Run Container*

After running the container, the application can be accessible from the elastic IP in our case it is http://ec2-3-13-208-251.us-east-2.compute.amazonaws.com:8000/ .

## 6.3 Android user interface

The user interface that houses the user interface controls or widgets that will display on the screen of an android application or activity screen is defined by Android Layout. In general, every application is made up of a View and a ViewGroup. An android application, as we all know, includes a big number of activities, each of which may be considered a distinct app page. As a result, each activity includes a range of user interface elements, such as View and ViewGroup instances.[26]

### 6.3.1 User interface architecture

The flow chart can be simple or complex that will be based on our requirement. You can use android's default widgets and layouts, or you can make your own custom Views. The most fundamental component of a user interface is represented by this class. A View is a rectangle on the screen that is responsible for drawing and event processing. View is the base for all the user interface components (Image field, text view, buttons etc) as seen in figure below.[27]

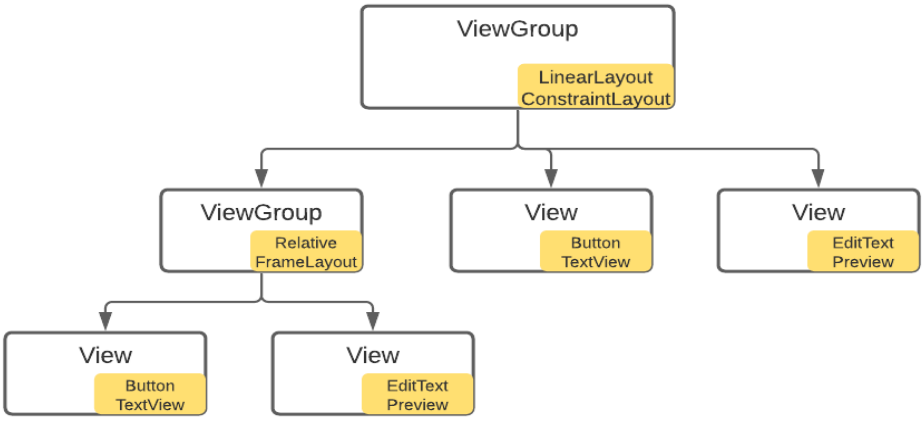


*Figure 6.2 View Hierarchy*

## 6.4 Android screen user interface components

In this we will discuss the elements that are displayed on the android screen and various layouts that are available in android to position the components.

### 6.4.1 Textview

TextView is a graphical user interface control that is used to configure and show text to the user depending on our specifications. The textview control will behave similarly to a label control and will not enable users to modify the content.[28]

```
<TextView
    android:layout_width="wrap_content"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    android:layout_margin="8dp"
    android:id="@+id/ticketNumber"
    android:text="Ticket Number"
    android:layout_height="wrap_content"/>
```

*Example 6.9 TextView*

### 6.4.2 Edit text

EditText is a user interface control that allows the user to enter or change text. We must use the inputType property when utilizing the edittext control in our android applications to indicate the type of data that the text field may receive.[28]

```
<EditText
    android:layout_width="match_parent"
    android:hint="Login"
    android:textColorHint="@color/colorAccent"
    android:inputType="text"
    android:id="@+id/username"
    android:layout_height="wrap_content"/>
```

*Example 6.10 EditText*

### 6.4.3 Button

A button is a user interface element that, when clicked or pressed, performs some action. It is a highly popular widget on android, and many designers make use of it [28].

```
<Button
   android:layout_width="wrap_content"
   app:layout_constraintTop_toBottomOf="@id/floatPassword"
   app:layout_constraintRight_toRightOf="parent"
   android:layout_margin="8dp"
   android:text="Login"
   android:textColor="#fff"
   android:background="@color/colorPrimary"
   android:id="@+id/login"
   app:layout_constraintLeft_toLeftOf="parent"
   android:layout_height="wrap_content"/>
```

*Example 6.11 Button*

### 6.4.4 Check box

A checkbox is a square button with a check to indicate the current status of the button On &Off. The user can choose one or more things from a list using checkboxes. Checkboxes can be used to enable or disable a feature. Changes in the states of one checkbox normally do not influence the states of other checkboxes, unlike radio buttons[29].

```
<CheckBox
   android:layout_width="0dp"
   android:layout_height="wrap_content"
   android:layout_weight="1"
   android:layout_margin="8dp"
   android:id="@+id/checkbox" />
```

*Example 6.12 Checkbox*

### 6.4.5 Radio button

A radio button is a plugin that allows you to select from many options. At any one moment, the user may only select one choice. Each choice is a radio button, and the issue's alternatives are collectively referred to as the Radio Group [29].

```
<RadioButton
   app:layout_constraintTop_toTopOf="parent"
   app:layout_constraintLeft_toLeftOf="parent"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/checkingNameLayout"
   />
```

*Example 6.13 RadioButton*

### 6.4.6 Linear layout

Linear layout is a view group that is most frequently used that aligns the children in the single direction. It can be on the developer requirement whether it display it as horizontal or vertical direction. here are piled one after other, so vertical layout contains only one child per row and same for the horizontal list.[30]

1. **Vertical**: All of the children are positioned vertically in a line one after the other like this. We defined orientation "vertical" in the code snippets below so that the children/views of this layout are shown vertically.

```
<LinearLayout
    android:id="@+id/detail_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:background="@drawable/rect_box"
    android:layout_margin = "4dp"
    android:padding="8dp"
    app:layout_constraintBottom_toBottomOf="parent"/>
```

*Example 6.14 Vertical Linear Layout*

2. **Horizontal:** Every one of the children are positioned horizontally in a line one after another. We set alignment "horizontal" in the source code below so that the children/views of this layout are shown horizontally.

```
<LinearLayout
    android:id="@+id/layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintBottom_toBottomOf="parent"/>
```

*Example 6.15 Horizontal Linear Layout*

### 6.4.7 Constraint layout

ViewGroup's new type of layout is Constraint layout. Constraint Layout is a ViewGroup that enables you to design huge and complicated layouts with a flat view hierarchy, as well as position and size widgets in a very dynamic way. It was designed to aid in the reduction of view nesting while also improving the efficiency of layout files. It was released in 2016 and is most reliable and flexible to work with on the android studio editor. We can perform animation on constraint layouts, and it improve the performance over the other layouts. [31]

```
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".Fragments.HomeFragment"/>
```

*Example 6.16 Constraint Layout*

### 6.4.8 Table layout

Android TableLayout will divide view groups into columns and rows. To create a row in the table, you will utilize the table row element. The columns, rows, and cells in table layout containers do not have a border line. The number of columns in a table is equal to the number of cells in the row with the most cells.

### 6.4.9 Frame layout

One of the most basic layouts for organizing view controls is the Frame Layout. They're made to cover a portion of the screen. Because it might be hard to show single views at a specified location on the screen without overlapping one another, Frame Layout is being used to store child views. Attributes that are used in frame layout are android:id, android: foreground, android: foregroundGravity, android: measureAllChildren. Multiple children can be added to a frame layout and their position can be controlled by assigning gravity to each child.

```
<FrameLayout
   app:layout_constraintTop_toBottomOf="@id/divider"
   android:layout_width="match_parent"
   android:layout_height="0dp"
   app:layout_constraintBottom_toTopOf="@id/detail_layout"
   android:id="@+id/cameraLayout">
</FrameLayout>
```

*Example 6.17 Frame Layout*

### 6.4.10 Activity

One of the cornerstones of Android OS is activity. In a simplified way, an activity is a screen with which the user interacts. In android, every activity has a lifespan that includes being generated, launched, resumed, paused, halted, or deleted. The activity lifecycle is the name given to these many phases. Like in most of the programming languages we have (main method) as its entry point of the whole execution, therefore the android OS is initiated code in an activity instance. Each one of the activities has a layout that I listed above these layouts have user interfaces that interact with the user. Listed below methods are the android activity lifecycle.

| Methods | Description |
|---|---|
| OnCreate | When a new activity is created, it is termed. |
| OnStart | When activity becomes visible to the user, this function is invoked. |
| OnResume | called when activity will start interacting with the user. |
| OnPause | When the user can't see what's going on, this method is used. |
| OnStop | When a user's activity is no longer visible, this method is implemented. |
| OnRestart | called after your action has come to a halt, but once it begins. |
| OnDestroy | Before the activity is deleted |

*Table 6.1 Activity Lifecycle*

## Transfer data across activities

The application's user interface design allows the user to transition between activities. The Intent() class is used to begin the new activity. As a result, there are times when data must be moved across operations. For example, if a user fills out a form in one activity but then clicks the submit button in another, the application must display the data input by the user in the other activity. If this is the case, data transmission should be implemented. Any type of data, such as text, int, or object, can be exchanged across activities. The key-value pair combination is used to convey data. As specified using intent other activities can be initiated. And data is transferred by adding the data to intent with the help of extras.

Sending Activity: The activity which sends the data to other activities. In the this activity following code has to included.

```
Intent intent = new Intent(getActivity(), ScannerActivity.class);
intent.putExtra("event", selectedEvent);
startActivity(intent);
```

*Example 6.18 Sending Data*

Receiving Activity: The activity which receive the data from other activities. To receive the data successfully following should be implemented.

```
Intent intent = getIntent();
CheckingTable event = (CheckingTable) intent.getSerializableExtra("event");
```

*Example 6.19 Receiving Data*

### 6.4.11 Fragment:

Fragment is an activity component that allows for more flexible activity creation. It is not inappropriate to consider a fragment as a type of sub-activity. In an activity, it depicts a behaviour or a piece of the user interface. Many fragments may be used in a single activity to create a multi-panel user interface, and a fragment can be reused in multiple Activities. Fragments must always be embedded in an activity, and the lifespan of the host activity has a direct impact on the lifecycle of the fragment. We may combine several fragments in a single activity by utilizing fragments. Each fragment has its own set of events, layouts, and life cycle. It gave users more freedom and removed the restriction of only having one activity on the screen at a time.[32]

## 6.5 Android core

This part will concentrate on the android application's heart. This section will cover topics such as utilizing libraries, recording live video, and connecting to the REST API.

### 6.5.1 CameraX

CameraX is a Google library that is included with the jetpack library. This makes it simple for developers to incorporate live view and create camera apps. This library is more stable than others on the market and is simple to incorporate. This library is a significantly enhanced version of camera2, addressing one of the device compatibility difficulties[33]. The advent of this library decreased the amount of code required to write for camera applications. There are different use cases that can be used by the developers which are defined below:

1. Preview: This case shows a live glimpse of the surface from the device camera.
2. Image Analysis: This gives you access to the CPU buffer, which is used to scan images for machine learning.
3. Image Capture: This enables users to include the option to snap and store photos.

We employ the Preview and Image Analysis use case in our application since our application will seek for the barcode and scan it, which is then provided to Google ML Kit to analyse. Following that, we'll look at how these two scenarios were incorporated into the application.

```
@RequiresApi(api = Build.VERSION_CODES.P)
void bindPreview(ProcessCameraProvider cameraProvider)
{
   Preview preview = new Preview.Builder()
           .build();
   CameraSelector cameraSelector  = new CameraSelector.Builder()
           .requireLensFacing(CameraSelector.LENS_FACING_BACK)
           .build();
   preview.setSurfaceProvider(cameraPreview.getSurfaceProvider());
   camera = cameraProvider.bindToLifecycle((LifecycleOwner)this
           ,cameraSelector,preview,imageAnalysis);
}
```

*Example 6.20 Bind Preview*

Above java method, build the surface which will be used to display the live updates from the camera. As a result, a preview object is produced initially, followed by the camera to be set (in our example, the Back Lens). Following that, all of the essential settings are bound to the life cycle of cameraProvider in order to obtain the Camera object.
Once the camera is attached to preview, our program will begin receiving LiveView using the camera that was picked during configuration. Now we must develop the image analyzer, which will generate the picture that will be supplied to the Barcode Scanner ML model.

```
@RequiresApi(api = Build.VERSION_CODES.P)
private void configImageAnalysis(){
   if(getActivity() != null) {

imageAnalysis.setAnalyzer(ContextCompat.getMainExecutor(getActivity()),
               imageProxy -> {
                   int rotationDegrees =
imageProxy.getImageInfo().getRotationDegrees();
                   @SuppressLint({"UnsafeExperimentalUsageError",
"UnsafeOptInUsageError"}) Image mediaImage = imageProxy.getImage();
                   if (mediaImage != null) {
                 InputImage image =
InputImage.fromMediaImage(mediaImage,rotationDegrees);
                   }
               });
   }

}
```

*Example 6.21 Image Analyzer*

The preceding code generates an image analyzer from the context. The picture may then be rotated, and the raw image obtained from the image proxy. InputImage may be produced from the picture and supplied to our scanning model.

### 6.5.2 Retrofit

Square created Retrofit, an HTTP (Hypertext Transfer Protocol) client for Android applications. Our program will use this to make synchronous or asynchronous API requests. This library simplifies the conversion of JSON objects to Java objects and vice versa, which explains its type-safe feature. If anything in JSON is not defined in the Java class, retrofit will not generate problems.

### Setting up

Dependency must be established in the gradle file to include the retrofit client in our application.

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.3.0'
```

*Example 6.22 Dependencies*

Now sync the project to obtain the required classes and components for the retrofit connection.

### Connecting to REST API

A retrofit instance must be generated for the connection to be successful. We need a URL to which API requests will be performed in order to construct the object.

```
public class RetrofitConnection {
   private static Retrofit retrofit;
   private static final String BASE_URL = "http://ec2-3-13-208-251.us-east-
2.compute.amazonaws.com:8000/api/";
   public static Retrofit getRetroFitInstance()
   {
       if(retrofit == null)
       {
           retrofit = new Retrofit.Builder().baseUrl(BASE_URL)
                   .addConverterFactory(GsonConverterFactory.create())
                   .build();
       }
       return retrofit;
   }
}
```

*Example 6.23 Retrofit Connection*

REST API is deployed to an AWS EC2 instance, which can be accessed at http://ec2-3-13-208-251.us-east-2.compute.amazonaws.com:8000/api/. Create the retrofit object and provide the base URL (Uniform Resource Locator), then add the converter and construct the client to convert the object. This client will now be used to begin all API queries.

## Service

Now that we have the retrofit client, we need the Service interface to execute the request, which will use the REST endpoints and make the appropriate requests. For which various annotations are used to define distinct URL components.

```
public interface DataService {
   @Headers({"Authorization: Token " + BuildConfig.API_KEY})
   @POST("scanning/")
   Call<ScanningTable> createScanning(@Body ScanningTable scanningTable);

   @Headers({"Authorization: Token " + BuildConfig.API_KEY})
   @GET("scanning/")
   Call<List<ScanningTable>> getScannings();

   @Headers({"Authorization: Token " + BuildConfig.API_KEY})
   @PUT("scanning/{id}")
   Call<ScanningTable> updateScanning(@Path("id") long id, @Body
ScanningTable scanningTable);
}
```

*Example 6.24 Data Service*

The POST, GET, and PUT annotations indicate the type of request that will be generated by this function, as well as the various parameters that will be required to make the request successful. The headers annotation will place the parameters in the URL's header, in our case authorization, which is the API key. (described in Chapter 7).

Everything is now in place to make the request. So, build the dataService instance and perform the relevant calls based on the application's requirements.

```
dataService= RetrofitConnection
             .getRetroFitInstance()
             .create(DataService.class);
```

*Example 6.25 DataService Instance*

The preceding code makes use of the getRetroFitInstance() function, which is seen in the figure 6.21. Then, as seen below, call the request.

```
Call<List<TicketListTable>> call = dataService.getTicketLists();
```

*Example 6.26 Calling the REST API endpoint*

After a successful execution, the application will receive a response from the API in the form of a json object, which will be transformed to a java object.

# Chapter 7
## Security

Businesses have always been concerned about security, and this issue is amplified when it comes to mobile apps. Almost every major company or product now has a mobile app to make it easier to communicate with their clients. With the rapid growth of mobile devices and the expanding usage of public cloud, where data is stored on servers and available from almost anywhere, the applications that operate on these platforms must be safe, durable, and tough. They must withstand cyberattacks because, as history has proven, hackers follow the information and go where they can acquire it. The battleground is moving to mobile devices. One of the most important obligations of an android developer is to think about security while designing apps. To effectively protect against this, one must know security mechanisms and how to create safe apps.[34]
Data is kept in a docker container on AWS EC2 in our application, making it publicly accessible. To preserve data, we must implement some security measures that will keep it safe. Another thing to keep in mind is that while our app is installed on an android device, a third-party program may attempt to contact it and utilize or steal data. To keep it safe locally, we'll use a technique to encrypt our database, making it unreadable from the outside.

### 7.1 SQLChiper

SQLCipher is a toolkit that encrypts SQLite database files with aes-256 Encryption in a straightforward and safe manner. Many proprietary and open-source solutions have chosen SQLCipher as a highly secured technology, making it among the most popular encoded data stores for android, Micro, and Software applications. SQLCipher is a SQLite database cybersecurity module that makes it easier to create protected databases. It inserts a listener into the pager system that may operate on database pages right before they are stored to it and retrieved from storage, using the core SQLite Encoder API. The reason we encrypt our database is that some hackers may attempt to reverse engineer your program, giving them access to data stored on the local device. [35]

### Features

1. The whole database file seems to include useless data once it is encoded.
2. When feasible, SQLCipher locks memory and wipes it before releasing it
3. Each information page is independently decrypted. The default page size is 4096 bytes, although this can be changed at runtime to speed up particular query types.
4. Encryption latency is as low as 5-15 percent, resulting in lightning-fast performance.
5. The database file's data is completely secured.
6. Uses zero-configuration and application-level encryption, as well as strong security practices.
7. The OpenSSL crypto module provides algorithms that have been reviewed regularly.

### 7.1.1 Implementation

```
dependencies {
   implementation "net.zetetic:android-database-sqlcipher:4.4.3"
}
```

*Example 7.1 Dependencies*

To begin encrypting our Room database for an android application, we must first define the dependency that will give the database's encryption algorithms and methods. SQLChiper is compatible with Room since it is an extension of SQLite. To begin encoding, we must define the encryption during database construction.

```
final byte[] passphrase =
SQLiteDatabase.getBytes("nasjkdfnuiweub".toCharArray());
final SupportFactory factory = new SupportFactory(passphrase);
if (INSTANCE == null) {
   INSTANCE = Room.databaseBuilder(context.getApplicationContext(),
           AppDatabase.class, "app_database")
           .openHelperFactory(factory)
           .allowMainThreadQueries()
           .build();
```

*Example 7.2 Room Instance*

As in the code above, a master key must be defined before the data can be encrypted. The developer is responsible for securely embedding the key in the database. With this simple integration, now our data is secured and unreadable from the outside. Next, we will focus on other techniques to secure our applications.

## 7.2 API tokens

APIs, as previously mentioned, are used to transfer data from one end to the other. As a result, it's critical to protect these APIs against attacks, and only authenticated users should be able to access the data they contain. There are several ways to obtain this degree of security. Some of these are verifying input, requesting credentials, and allowing users to use tokens. Each of them has its own set of advantages and disadvantages. The focus of this description is on utilizing API tokens to contact REST endpoints and retrieve data from them. [36]

Tokens are a natural progression from email and passwords. They are distinct for every user, who wants to access data over the internet. It's an HTTP authentication system that uses tokens. When a user requests data, the user must provide the Token in the header, which is verified to see whether it exists and if it does, the user is authorized.

We used the Django REST framework to create the REST API for our application, which includes token-based authentication. As a result, it is easy to set it without putting in a lot of effort. [36]

To start with, we need to add the rest_framework.authtoken in the list of installed apps in order to use the functions in our project. Then, we need to specify the authentication classes that will be utilized in the project that is

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.TokenAuthentication',
    ),
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.IsAuthenticated',
    ),
}
```

*Example 7.3 Rest Framework Configs.*

After that, in order to access data via the API, the user must supply a valid token in the URL header, which will then authenticate the user into the system.

## 7.3 Securing the secret keys

During the development of a project, a secret key might be anything from a database password to an API key or anything else that grants root access to the user who possesses it. To access the data, these keys may need to be hardcoded into the program. As a result, hardcoding the keys without safeguarding them makes the program vulnerable to data theft. There are a variety of approaches for securing these kinds of keys. Following that, we'll look at how to include master keys into web and android applications.

### 7.3.1 Android application

We have hardcoded the API key in the android application, which will be in the header of the URL when collecting data from the API. This implies that if someone else has access to the key, they can take the information. There are also cases where developers utilize REST APIs created by other programmers, in which case some of them pay for the services they offer. As a result, if someone obtains your API key, they may use it in their app, and you must pay for the services or requests that their app makes. Developers must pay close attention to adequately safeguarding their keys in order to reduce the possibility of data theft and pay a premium.[37]

To secure the API key in android application

1. Specify your key in the local.properties file

```
apiKey="YOUR_SECRET_KEY"
```

*Example 7.4 API Key*

2. Then go to the app module gradle file and encode the key.

```
defaultConfig {
    buildConfigField "String", "API_KEY", localProperties['apiKey']
}
```

*Example 7.5 Gradle File*

3. Then rebuild your project, this will generate the BuildConfig.java file which will have our key encoded. The file will look something like this

```
public final class BuildConfig {
 public static final boolean DEBUG = Boolean.parseBoolean("true");
 public static final String APPLICATION_ID = "com.vishav.barcode";
 // Field from default config.
 public static final String API_KEY = "YOUR_SECRET_KEY";
}
```

*Example 7.6 BuildConfig File*

After following all these steps, developers can access the key anywhere in the code using the BuildConfig.API_KEY in our case.

### 7.3.2 Django Rest API

In every Django project, there is a secret key which is used to perform the cryptographic signature. In most cases this key is used to sign session cookies. If this key is obtained, the user will be able to alter the cookies supplied by the program. For Django projects it is recommended to hide the keys using the environment variables. [38]
To accomplish this,

1. Create a .env file in the root folder of your application and specify your key and its name.
2. After that, go to manage.py file and load the environment variables by doing the following.

```
def main():
    """Run administrative tasks."""
    dotenv.load_dotenv()
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'BarcodeApi.settings')
```

*Example 7.7 Loading the key from Env*

3. Now all the variables are loaded and can be accessed using *os.getenv("SECRET_KEY").*

Now all of the keys are safe hence, which means the data of the application is secure.

## 7.4 Enforcing security requirements

| Prior to Implementing Security Measures | After Applying Security measure |
|---|---|
| **Android Application** | |
| Reverse engineering will reveal the API token that hackers can use to obtain access to databases or misuse API tokens for their own advantage. | All API tokens are saved in locally. properties generated by Gradle and reverse engineered will render the key unusable. |
| The data isn't encoded. | SQLChiper is used to encrypt the database, which can only be decrypted with the private key. |
| For retrieving data from the REST API, the API key was hardcoded into the url. | The API key was inserted using local.properties to prevent hackers from intercepting traffic and gaining access to the key. |
| **REST API** | |
| Because the Rest API database was accessible over the internet, it was susceptible. | To authenticate the user and access the data, a Rest Api token was incorporated. |
| The Django secret was hardcoded in the settings. | The secret key was saved in a separate file that could be accessed via a virtual environment. |
| **Docker Container** | |
| The AWS container was reachable through the web application's public IP address. | Only HTTP requests were permitted to get past the firewall, and traffic was monitored. |

## 7.5 Mesaures serving the goal of the project

This chapter covered several security strategies and libraries that may be used to prepare an application for deployment. Deploying the program without considering security makes it susceptible and less engaging since users do not believe that their data is secure. After implementing all of the steps outlined in this chapter, developers should encrypt databases using several layers and reply to only legitimate requests with information. The API keys should be appropriately concealed because if someone gains access, they can use it in their application and the developer must incur the expense, or they can access the user's sensitive data, which might lead to a security breach. Overall, every developer should endeavor to encrypt or hide any private key or data that may pose a security risk. We have now implemented all of the approaches that we mentioned. However, when it comes to elevating the security of a system or application, "More is always less."

# Chapter 8
## Testing

Software testing is a technique for determining if the actual application meets the expected criteria and ensuring that it is defect-free. It entails the use of manual or automated methods to assess one or more attributes of interest by executing application components. In comparison to real requirements, software testing's goal is to find mistakes, gaps, and inconsistencies. Software testing is a commonly utilized technique since it is required to test all software before it is released. There are different types of testing which are used to detect the flaws in the system. To test the functionality different types of tests are recommended by Google, which will help in analysing the error or to debug them. In this chapter, we will look into these types and how they are implemented in android testing [39].

### 8.1 Unit testing

Unit testing is a way of examining the smallest testable pieces, such as methods and classes, that can be run individually and independently. They are run on a local Computer or in a development environment without the need for a physical device. I was unable to complete this testing since my code requires context, which is obtained after commencing action on the actual device[40].

### 8.2 Integrated testing

Integration tests evaluate interactions across different layers of the application inside modules or interactions between different modules. These tests are performed on android emulators or actual devices to determine how the components will function on a real device. Integration testing is a technique for verifying the interaction of a different section of code[41].

```
@Test
public void updateTicket()
{

   TicketTable ticket = new TicketTable("15000M",
           "Vishav"
           ,"Nem","Nem",2,
           1,"No");
   ticketTableDao.insert(ticket);
   ticketTableDao.updateTicketUseable(0,1);
   List<TicketTable> ticketTables = ticketTableDao.getAll();
   TicketTable ticketInDb = ticketTables.stream().filter(x-
>x.getTicketNumber()
           .equals("15000M"))
           .findFirst()
           .orElse(ticket);
   Assert.assertEquals(0, ticketInDb.getTicketUseable());
}
```

*Example 8.1 Integration Unit testing*

Example 8.1, verifies that the update capability of the ticket tables works as intended. To validate this, we first generate a ticket and enter it into the database. The update Ticket function is then used to update the ticket in the database. Following that, we obtain the tickets and filter for the one we changed. Also, check to see whether the ticket has been updated.

## 8.3 User interface testing

User Interface Tests allow verifying that the application maintains a high standard of quality and functional requirements. There are different methods to test the user interface of the application.

1. Manual Testing: In this way, the user manually configures the program by following a series of instructions. This process can take days to test the functioning of the program and check it on multiple devices, which might lead to additional mistakes and bad test results.
2. Automated Testing: These methods test the android components in the automated way. Nowadays, there are two different approaches under this category.
    - ➢ Creating user interface tests that can be run repeatedly on an android emulator or device. For this developer is responsible for writing functional tests. The same tests may be run on various devices multiple times. And the logs for the faults discovered will be automatically tracked.
    - ➢ Robot user interface Testing: Several cloud services now allow you to test apps across several devices with an automated robot. During this testing, a robot will try to explore the application by clicking or scrolling on the user interface components. They also record video so that the testers may see how the exact problem was identified or what sequence of activities were used to test the application.

For the user interface testing of my application, I used the first approach of user interface testing that is writing the tests. During the testing different activities and user interface components were considered that users can interact with.

### 8.3.1 Espresso

For testing the user interface of the application, Espresso Library was utilized. It is highly recommended by Google for its concise, beautiful, and reliable user interface testing. The fundamental API is compact, stable, and simple to understand, but it is also available for customization. Espresso tests simply explain objectives, behaviours, and conclusions without being distracted by generic text, customized architecture, or complicated technical requirements. Espresso tests are rapid! [42] It allows the user to manage your queues, synchronize, sleep, and surveys behind while manipulating and asserting on the app user interface when it is idle.

Before, we could test the login activity developers have to create the application using the ActivityScenario which can be done with the code provided below.

```
@Before
public void setUp()
{
   ActivityScenario<LoginActivity> scenario
           = ActivityScenario.launch(LoginActivity.class);
   scenario.moveToState(Lifecycle.State.RESUMED);
}
```

*Example 8.2 Setup Testing Environment*

After the activity is created and launched all the components within that activity are all available for testing. The code below is to check if the login credentials are correct then the expected Activity is launched.

```
@Test
public void testInteraction()
{
   Intents.init();

Espresso.onView(withId(R.id.username)).perform(ViewActions.typeText("admin"
));

Espresso.onView(withId(R.id.password)).perform(ViewActions.typeText("admin"
));
   Espresso.closeSoftKeyboard();
   Espresso.onView(withId(R.id.login)).perform(click());
   Intents.intended(hasComponent(MainActivity.class.getName()));
}
```

*Example 8.3* User Interface *Test using Espresso*

This test initiates the intent to keep track of the activity that is launched during the testing. The tester may then use the withId () function to locate the component within the activity and conduct the operation as needed. Following the provision of the credentials, the activity executes the defined logic, and if the credentials are correct, the appropriate activity is launched. The activity's launch is monitored by intents, which confirms that the activity was initiated.

## 8.4 Postman API

The REST API, which keeps data synchronized across various android devices, serves as the application's backend. So, the postman API was used to test the Django REST API. Essentially, postman API is the client that allows testers to easily test the API by starting various sorts of calls to the API. The user can see the API answer in a variety of ways. Postman API testing makes it easy for developers to debug the application in an effective manner. A variety of queries were sent to the application API in order to test the various functions of the API.

### 8.4.1 Testing API

**POST request**: Post requests, as we all know, are used to provide data to a website and subsequently store it to a database. To test this, we may define the url to which the POST request must be performed, then enter the data in the Body using JSON format and click the send button.

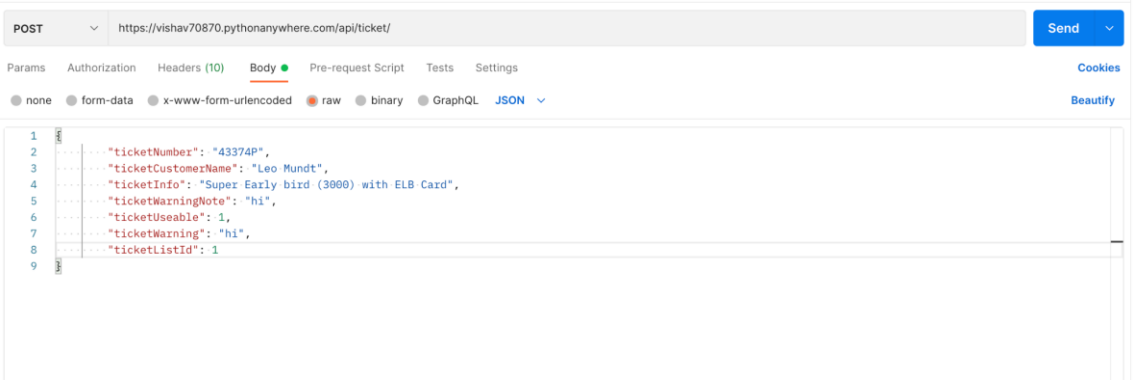


*Figure 8.1 POST request*

Then the postman API client will try to make the request specified and will provide the response we got form the API. As, I executed this request the following response was generated by the API.

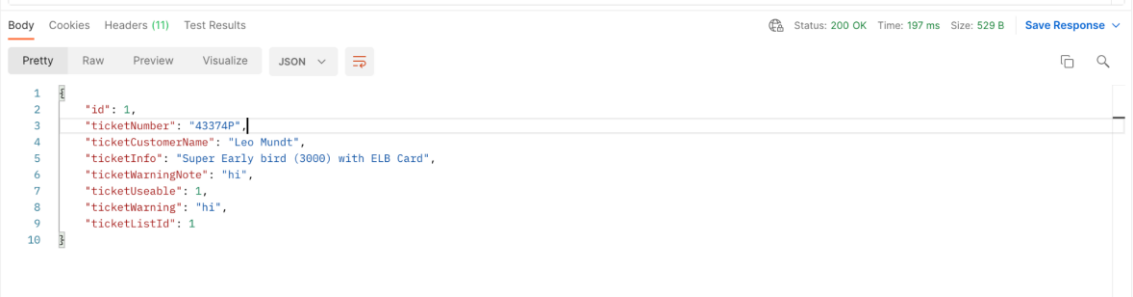


*Figure 8.2 POST request result*

The answer contains the id property, which was not supplied when the data was sent. This validates that the data was correctly saved in the database. Aside from that, there is time, which indicates how long it takes to request the API to do the specific operation and provide a response. Furthermore, a Status code of 200 indicates that the connection between the client and the REST endpoint was successful.

**GET request**: Get request is used to fetch the data from the API endpoint. The response we get can vary by the format we specified. To execute this test same post client windows was used as in figure 8.2, just have to change the POST request to GET request from the windows and click on the send button which will fetch all the tickets saved in the database.

**Testing Authorization:** To improve the API's security, token authorization was used to determine whether our data is available to the public. Using a false token, we may attempt to make a request to the database.

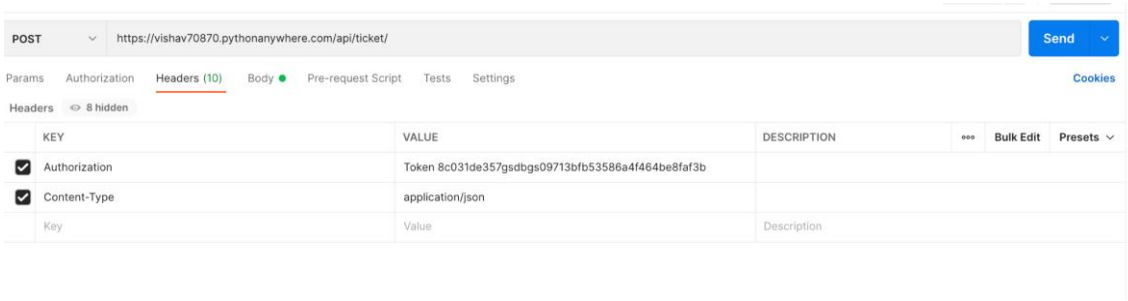


*Figure 8.3 Testing API token*

The above figure depicts an API header, one being authorization, which contains the API Token. When we submit this request to API, it fails as predicted since the token supplied in the request does not exist in our application. The outcomes were as follows:

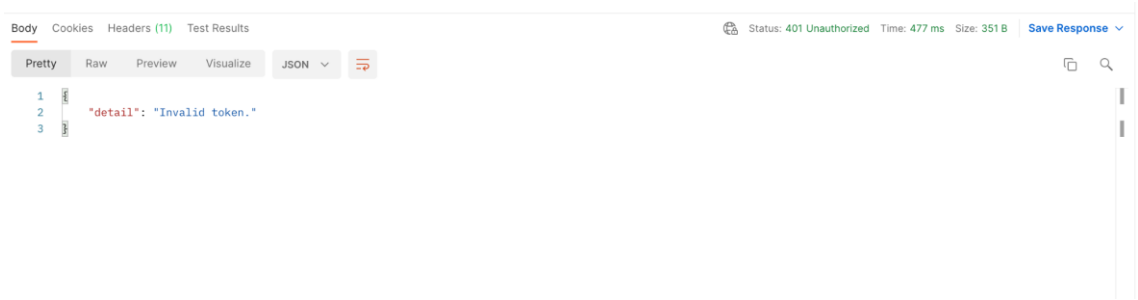


*Figure 8.4 Invalid Token result*

As anticipated, the API request failed, and the API returned a json response with the message "Invalid Token." And it's worth noting that the API's status response code was 401, which implies Unauthorized. As a result, by not allowing an external user to see the data saved, we can confirm that the token authorisation is operating as planned.

# Chapter 9
# Summary

## 9.1 The goal of the development

The goal of this thesis was to integrate automated machine learning models into android applications. This was accomplished by creating an android application that was embedded with Google ML Kit. Based on the studies and tools used, it is possible to conclude that incorporating an automated machine learning model can make development easier and more dependable. Using the Google ML kit, developers can create android apps that can execute a variety of complicated tasks quickly and address real-world issues. There is also a full description and implementation of various tools and technologies. Following development, the program was tested on real-world events by the firm, where it performed flawlessly and as planned. There were several challenges and confusions concerning the library to be utilized throughout development, all of which are addressed in this synopsis. Because the organization wanted the application to integrate with their software, Django REST API was created and deployed on the AWS cloud. This method allows the database to be synchronized across two apps. After utilizing the program in a real-world setting, the process of validating tickets for events was sped up by 50%, and the risks of error when validating were greatly decreased. Furthermore, the automated data update from the enterprise application to the android application saved a significant amount of time. In addition, other development methodologies, such as MVVM, were employed to preserve the code's quality. The code is more legible and intelligible after MVVM implementation. As a result, future options to extend the code are now available. This synopsis also focuses on the security elements of data security. Various data encryption technologies and approaches were employed to ensure security. Even after reverse engineering the program, data was not readable, and unauthorized users were unable to access crucial information. This project assisted me in learning about the various technologies available and how to apply them into the apps. The indicated future development will be completed in order to save time while developing.

## 9.2 Possibilities of Future Development

The program is complete and functional, but there are still enhancements and features that need to be updated. In this area of my thesis, I will outline the changes that can be done to improve the user experience or to include new features.

1. **CI/CD**

   The abbreviations CI and CD are often used in current development methods and DevOps. Continuous integration (CI) refers to a core DevOps professional standard in which developers often integrate changes to the system into a centralized source where automated builds and tests are executed. However, CD can refer to either continuous delivery or continuous deployment [44].

   - CI: Continuous integration (CI) is a method in which developers make minor modifications and tests to their code. Because of the magnitude of the criteria and the number of stages required, this process is streamlined to guarantee that teams can create, test, and package their applications in a reliable and repeatable manner. CI helps to simplify code changes, giving developers more time to make changes and contribute to better products [43].

   - CD: Continuous Delivery (CD) is the automatic distribution of finished code to contexts such as test execution. CD enables the delivery of code to various settings in an automatic and consistent manner. After test results are positive here comes the next step deployment. CD helps to deploy the code into production which has passed the tests thus saving the time for the developers [43].

   In the future the plan is to integrate CI/CD on the REST API to automate the process to code testing and deployment. By implementing this, the changes in the code will be automatically deployed in the production after running it against the tests written for automation. This will save the time of deploying the application manually. Tools which could be used to achieve this are Jenkins, GitHub, Docker.

2. **User Management**

   For the time being, in our application, there is only one user admin who may execute any task such as adding, editing, updating, or deleting. In the future, roles should be defined for the user based on the company's needs, and particular permissions should be enabled for these duties. This would need the usage of user management. Django, on the other hand, gives the finest and most centralized solution to handle users. We have a user group in Django that allows developers to set the permissions that a certain user has, and the group can be allocated to employees based on their job in the firm. User Management is a mechanism for managing different users for the program by setting their data access. User

management will also maintain track of the history, which implies which user contributed or edited which item in the database.

## 3. Improving UI/UX

The user interface (UI) is the component with which the user interacts with the application, whereas the user experience (UX) is the experience users have while using the program. To engage their users more, developers or designers should take UI/UX in mind while creating an application. Developers are continually striving to improve the user experience. The application's user interface (UI) is vital in improving UX. For the time being, our android application makes use of the android support library for user components. In the future, I'd like to replace it with Material Design 3, which Google has released in 2021. Following the launch, I went over the user components supplied by this library. These components are both appealing and user-friendly, which will entice more people to utilize the program. Aside from the user interface, there may be many other aspects that may be improved to make users more comfortable while using the program. Others are reducing application loading times, making apps perform more tasks through threading, and saving the user time. Furthermore, if a user complains about something, designers may analyse the request and change it to meet their demands. There is a need for the organization to upgrade the UI/UX since it will make the program more flexible for the workforce. Employees will test the application in order to obtain feedback.

## 4. Real Time Updates:

The process of changing application material in Realtime is known as Realtime update. This approach is now used in WhatsApp and many other software’s. For the time being, the application includes a button that, when pressed, syncs the database with the REST API. In the future, I will remove this button, and the app will receive real-time changes from the API. When the data in the application is changed, the REST API will send a notice. When the notice is received, the program will immediately retrieve the data.

# Glossary of Terms

| **Word** | **Meaning** |
|---|---|
| Android | It is a linux-based operating system intended specifically for mobile devices. |
| Android Studio | IDE for designing and developing high quality android applications. |
| API | Application Programming Interface |
| AWS | Amazon Web Service |
| Barcodes | A pattern that encodes information using bars and spaces that a machine can read. |
| Button | User interface to perform some actions |
| CPU | Central Processing Unit |
| Database | The collection of data using a well-defined way. |
| DBMS | Database Management System |
| DELETE | To delete the specific resource. |
| Dependencies | These are the links between two projects. |
| Django | It is a python-based web framework. |
| Docker | It is used to deploy applications using containers. |
| Dockerfile | The files used to create the docker image. |
| EC2 | Elastic Computing |
| EditText | User interface to take input from the user. |
| Endpoints | The entry point through which two applications can communicate with each other. |
| Entity | It is the table in the database. |
| Espresso | The user interface test library provides |

| Word | Meaning |
|---|---|
| | simple techniques to test applications. |
| Framework | It is usually a basic structure to provide the generic functionality of software or programming languages to solve common problems. |
| GET | To fetch the data from website |
| GOF | Gang of Four |
| Google ML kit | The machine learning kit developed by Google to provide models to android developers. |
| Hardware | Physical electronic equipment. |
| IDE | Integrated Development Environment |
| JSON | JavaScript Object Notation |
| Libraries | These are predefined code or templates which can be used in the applications |
| LiveData | It observes the data to listen to the updates to the database. |
| Machine Learning | It is a branch of Artificial Intelligence that uses data to train computers. |
| ML | Machine Learning |
| Model | The generic form of data structure using attributes. |
| MVVM | Model-View-ViewModel |
| OS | Operating System |
| POST | To send the data to the website. |
| Postman API | The client which calls the requests on the REST API for testing. |
| QR | Quick Response |
| Repository | It unifies the data from different sources |

| Word | Meaning |
|---|---|
| | and passes it to the viewmodel. |
| REST | The architecture style defines the World wide web. |
| REST | Representational State Transfer |
| Retrofit | A type-safe HTTP client for Android and Java turns HTTP API into a Java interface. |
| Room | Jetpack library built on the top of SQLite. |
| Schema | The generic structure of how data will be saved in the database. |
| Serializers | To convert the object to any other content type. |
| Software | Set of instructions to complete the task. |
| SQL | Structured Query Language |
| SQLChiper | The security tool to encrypt the database. |
| SSH | Secure Shell |
| Testing | The process validates the functionality and methods of application. |
| Token Authentication | The REST API tokens are unique for every user for the authorization process. |
| UI | User Interface |
| UI Testing | These tests mainly test the UI controls by invoking the actions. |
| UPDATE | To update the data of specific resource. |
| URL | Uniform Resource Locator |
| User Interface | It contains the UI controls and widgets with which the user can interact with the application. |
| UX | User Experience |

| Word | Meaning |
|---|---|
| View | The user-interface to interact with the application. |
| ViewModel | It links the View and Model layer. And supply data from Model to View for presenting. |
| VM | Virtual Machine |
| XML | Extensible Markup Language |
| Zxing | It is an alternative to Google ML kit. |

# List of Abbreviations

| | |
|---|---|
| ML | Machine Learning |
| MVVM | Model-View-ViewModel |
| REST | Representational State Transfer |
| API | Application Programming Interface |
| IDE | Integrated Development Environment |
| JSON | JavaScript Object Notation |
| XML | Extensible Markup Language |
| HTTP | Hypertext Transfer Protocol |
| HTML | HyperText Markup Language |
| VM | Virtual Machine |
| CPU | Central Processing Unit |
| OS | Operating System |
| AWS | Amazon Web Service |
| GOF | Gang of Four |
| SQL | Structured Query Language |
| UI | User Interface |
| UX | User Experience |
| DBMS | Database Management System |
| QR | Quick Response |
| Zxing | Zebra Crossing |
| URL | Uniform Resource Locator |
| EC2 | Elastic Computing |
| SSH | Secure Shell |

# Bibliography

[1] Karpischek, S. (2012). Mobile barcode scanning applications for consumers. ETH. *https://doi.org/10.3929/ethz-a-009774955*

[2] Adelmann, R. (2007). Mobile Phone Based Interaction with Everyday Products - On the Go. The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST 2007), 63–69. *https://doi.org/10.1109/ngmast.2007.4343402*

[3] Google. (n.d.). Android Platform. Android Developers. Retrieved October 23, 2021, from *https://developer.android.com/about*

[4] Singh, A., Sharma, S., & Singh, S. (2016). Android Application Development using Android Studio and PHP FrameworkAndroid Application Development using Android Studio and PHP Framework. IJCA Proceedings on Recent Trends in Future Prospective in Engineering and Management Technology, RTFEM 2016, 5–8. *https://www.ijcaonline.org/proceedings/rtfem2016/number1/25479-5109*

[5] Android - Architecture. (n.d.). Tutorialspoint. Retrieved October 29, 2021, from *https://www.tutorialspoint.com/android/android_architecture.htm*

[6] The web framework for perfectionists with deadlines | Django. (n.d.). Djangoproject. Retrieved October 31, 2021, from *https://www.djangoproject.com/*

[7] Ahuja, S. S. (2021). FlexFashion: E-Commerce with Advance Features. International Journal for Research in Applied Science and Engineering Technology, 9(12), 1015–1019. *https://doi.org/10.22214/ijraset.2021.39113*

[8] Christie, T. (n.d.). Home - Django REST framework. Django Rest Framework. Retrieved November 1, 2021, from *https://www.django-rest-framework.org/*

[9] What is Docker? | AWS. (n.d.). Amazon Web Services, Inc. Retrieved November 3, 2021, from *https://aws.amazon.com/docker/*

[10] Clancy, M. (2021, October 20). What's the Diff: VMs vs. Containers. Backblaze Blog | Cloud Storage & Cloud Backup. Retrieved November 3, 2021, from *https://www.backblaze.com/blog/vm-vs-containers/*

[11] Yadav, K. (2022, May 1). What is AWS and What can you do with it - Kunal Yadav. Medium. Retrieved November 3, 2021, from *https://medium.com/@kunalyadav/what-is-aws-and-what-can-you-do-with-it-395b585b03c*

[12] MVVM Advantages. (n.d.). Tutorialspoint. Retrieved November 7, 2021, from *https://www.tutorialspoint.com/mvvm/mvvm_advantages.htm*

[13] Bergman, P. (2019, May 7). Repository Design Pattern - Per-Erik Bergman. Medium. Retrieved November 9, 2021, from *https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30*

[14] Osagie, J. (2020, November 25). What is a Database? Linkedin. Retrieved November 9, 2021, from *https://www.linkedin.com/pulse/what-database-jonathan-osagie/*

[15] Oracle. (n.d.). What is a Database. Retrieved November 9, 2021, from *https://www.oracle.com/database/what-is-database/*

[16] MongoDB. (n.d.). Types Of NoSQL Databases. Retrieved November 9, 2021, from *https://www.mongodb.com/scale/types-of-nosql-databases*

[17] Madushan, D. (2018, December 19). How the SQLite Database Works. Dzone.Com. Retrieved November 9, 2021, from *https://dzone.com/articles/how-sqlite-database-works*

[18] Bansode, A., & Nagpure, S. (2021). Sign Language. International Journal Of Advance Scientific Research & Engineering Trends, 5(12), 702–706. *http://ijasret.com/VolumeArticles/FullTextPDF/1075_132.SIGN_LANGUAGE.pdf*

[19] Google. (n.d.). ML Kit. Google Developers. Retrieved November 11, 2021, from *https://developers.google.com/ml-kit*

[20] Google. (n.d.). Scan Barcodes with ML Kit on Android |. Google Developers. Retrieved November 11, 2021, from *https://developers.google.com/ml-kit/vision/barcode-scanning/android#java*

[21] Sakare, L. (2019, December 30). Zxing Vs Google Vision - lalitkumar sakare. Medium. Retrieved November 11, 2021, from *https://medium.com/@lkumar.sakare/zxing-vs-google-vision-fc3be8d83ace*

[22] RedHat. (2020, May 8). What is a REST API? Redhat.Com. Retrieved November 11, 2021, from *https://www.redhat.com/en/topics/api/what-is-a-rest-api#:%7E:text=A%20REST%20API%20(also%20known,by%20computer%20scientist%20Roy%20Fielding*

[23] Django. (n.d.). Models | Django documentation | Django. Retrieved November 12, 2021, from *https://docs.djangoproject.com/en/4.0/topics/db/models/*

[24] Christie, T. (n.d.). Serializers - Django REST framework. Django Rest Framework. Retrieved November 11, 2021, from *https://www.django-rest-framework.org/api-guide/serializers/*

[25] Django. (n.d.). Class-based views | Django documentation | Django. Retrieved November 16, 2021, from *https://docs.djangoproject.com/en/4.0/topics/class-based-views/#:%7E:text=A%20view%20is%20a%20callable,by%20harnessing%20inheritance%20and%20mixins*

[26] Parveen. (2019, November 11). Android UI Layouts. GeeksforGeeks. Retrieved November 16, 2021, from *https://www.geeksforgeeks.org/android-ui-layouts/*

[27] Rajkumar, A. S. (2013). A Study Paper On Android UI. International Journal of Enterprise Computing and Business Systems, 2(1). *http://www.ijecbs.com/January2013/1.pdf*

[28] Tutorialspoint. (n.d.). Android - UI Controls. Retrieved November 16, 2021, from *https://www.tutorialspoint.com/android/android_user_interface_controls.htm*

[29] Hazem, A. A. R., & Samy, S. A.-N. (2018). Android Applications UI Development Intelligent Tutoring System. International Journal of Engineering and Information Systems, 2(1), 1–14. *https://www.researchgate.net/publication/322757183_Android_Applications_UI_Development_Intelligent_Tutoring_System*

[30] Abhi. (2019, May 18). Linear Layout Tutorial With Examples In Android | Abhi Android. Abhiandroid.Com. Retrieved November 18, 2021, from *https://abhiandroid.com/ui/linear-layout*

[31] Abhi. (n.d.). Constraint Layout Tutorial With Example In Android Studio [Step by Step] | Abhi Android. Abhiandroid.Com. Retrieved November 18, 2021, from *https://abhiandroid.com/ui/constraintlayout*

[32] Google. (n.d.). Fragments. Android Developers. Retrieved November 19, 2021, from *https://developer.android.com/guide/fragments*

[33] Google. (n.d.). CameraX overview. Android Developers. Retrieved November 21, 2021, from *https://developer.android.com/training/camerax*

[34] Six, J. (n.d.). Application Security for the Android Platform. O'Reilly Online Learning. Retrieved November 21, 2021, from *https://www.oreilly.com/library/view/application-security-for/9781449322250/ch01.html*

[35] SQLCipher. (n.d.). SQLCipher Design - Zetetic. Zetetic. Retrieved November 21, 2021, from *https://www.zetetic.net/sqlcipher/design/*

[36] Christie, T. (n.d.). Authentication - Django REST framework. Django Rest Framework. Retrieved November 23, 2021, from *https://www.django-rest-framework.org/api-guide/authentication/*

[37] Google. (n.d.). App security best practices |. Android Developers. Retrieved November 23, 2021, from *https://developer.android.com/topic/security/best-practices*

[38] Django Secret Key | GitGuardian documentation. (n.d.). Gitguardian.Com. Retrieved November 23, 2021, from *https://docs.gitguardian.com/secrets-detection/detectors/specifics/django_secret_key*

[39] Nanda, V. (2021, July 13). Static Testing Vs Dynamic Testing. Tutorialspoint. Retrieved November 25, 2021, from *https://www.tutorialspoint.com/static-testing-vs-dynamic-testing*

[40] Gill, N. (2021, October 28). Unit Testing Techniques and Best Practices. XenonStack. Retrieved November 26, 2021, from *https://www.xenonstack.com/insights/what-is-unit-testing#:%7E:text=Concluding%20Unit%20Testing-,What%20is%20Unit%20Testing%3F,and%20produces%20a%20single%20output*

[41] Hamilton, T. (2020, January 10). Integration Testing: What is, Types, Top Down & Bottom Up Example. Guru99. Retrieved November 26, 2021, from *https://www.guru99.com/integration-testing.html*

[42] Google. (n.d.). Espresso. Android Developers. Retrieved November 27, 2021, from *https://developer.android.com/training/testing/espresso*

[43] Sacolick, I. (2020, January 17). What is CI/CD? Continuous integration and continuous delivery explained. InfoWorld. Retrieved November 30, 2021, from *https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html*

[44] Klocwork. (n.d.). Learn How to Optimize CI/CD Pipelines [E-book]. Perforce. Retrieved November 30, 2021, from *https://www.perforce.com/p/success/kw/ci-cd-best-practices-for-embedded-software*