 Elite Education Institute

**ASSIGNMENT COVER SHEET**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UNIT NO: ICT304** | **UNIT NAME:** | | **Final project Android** [Assignment 3] | | |
| **FAMILY NAME: Waleed** | | | | **FIRST NAME: Muhammad** | |
| **STUDENT NO: 20221057** | | | | **CONTACT PHONE NO:** | |
| **ASSIGNMENT/TOPIC TITLE:**  Mobile application | | | | | |
| **TUTOR*’*S NAME:** | | | | | **DATE RECEIVED STAMP** (Office use only) |
| **DUE DATE**   * **EXTENSION GRANTED** * **APPROVAL ATTACHED TO COVER SHEET** | | **SUBMITTED WORD COUNT:** | | |
| **TURNITIN NUMBER:** | | |

Student Declaration

I certify the following:

* This assignment is my own work, based on my personal study and/or research
* I have acknowledged all material and sources used in the preparation of this assignment, including any material generated in the course of my employment
* If this assignment was based on collaborative preparatory work, as approved by the teachers of the unit, I have not submitted substantially the same final version of any material as another student
* Neither the assignment, nor substantial parts of it, have been previously submitted for assessment in this or any other institution
* I have not copied in part, or in whole, or otherwise plagiarized the work of other students
* I have read and I understand the criteria used for assessment
* The use of any material in this assignment does not infringe the intellectual property/copyright of a third party
* I have kept a copy of my assignment and this coversheet
* The assignment is within the word limit and page limit as specified in the unit outline
* I understand that this assignment may undergo electronic detection for plagiarism, and a copy of the assignment may be retained on the database and used to make comparisons with other assignments in future. To ensure confidentiality all personal details would be removed.
* I have read and understood the information on plagiarism. For the Institute’s Student Academic Integrity Policy, please refer to the following website [http://ee.edu.au/download/policies/Student-](http://ee.edu.au/download/policies/Student-Academic-Integrity-Policy.pdf) [Academic-Integrity-Policy.pdf](http://ee.edu.au/download/policies/Student-Academic-Integrity-Policy.pdf)

Signature of Student: **Waleed**  Date: 02/12/2024

# OVERVIEW

The Employee Management System is the mobile application to manage employee records easily and quickly. This allows the user to access the user interface to insert, modify, and delete employee details painlessly. A secure authentication mechanism is integrated into the system to authenticate personnel that are permitted to get access and maintain the records of employee.

It is possible to store and manage records locally or remotely (so users can enter and record even if they are offline). The structure of the system is based on an architecture utilizing the MVVM (Model-View-ViewModel) for maintainability, scalability and high performance as well.

# ARCHITECTURE & TECHNOLOGIES

The Employee Management System is created using modern Android development practices such as Jetpack Compose, Room Database, with REST API integration. Lastly, MVVM architecture leads to a clean separation of concerns, library to library, and therefore aids in creating modular and maintainable codes.

## 1. Model Layer

* Responsible for handling data operations, both locally and remotely.
* It has stored onto Room Database for local storage and it allows to keep employee record in offline mode.
* It maintains data synchronization between the multiple devices by fetching and updating the employee records through REST API calls.

## 2. View Model Layer

* Acts as an intermediary between the Model and View layers.
* It uses Live Data to observe and update UI elements as efficiently as possible.
* It is used to manage coroutines responsible for handling long running operations like network requests and database transaction in the background.

## 3. View Layer

* It was developed using Jetpack Compose for a modern UI experience.
* It performs dynamic UI components and responds to their state changes.
* It contains UI elements which contains buttons, text fields, lists, and dialogs in order to interact with user.

## Key Functionalities User Authentication & Security

* Access to the application is dependent on the user must first register and then log in.
* JWT token based authentication secures the authentication mechanisms and only the authorized users are able to interact with the system.
* Passwords are also hashed with an encryption to secure them from unauthorized access.
* A multi session management makes sure the user is logged in between devices.

## Employee Records Management

* **Allow Add Employee:** It allows users to add a new employee with the details as a name, designation, department, contact and profile picture of that employee.
* **Updating Employee:** It will get the details of the existing employee to maintain differences (in the role, contact details or the department).
* **Permanent Delete Employee:** Allows a user to delete an employee’s records and is confirmed against accidental deletions.
* **Data Persistence:** Employee records are persisted using Room Database so that employee records can be accessed offline and the user experience is efficient.

## REST API Integration

* Implements synchronizing of the employee data between a mobile application and a

remote server.

* It uses HTTP methods (GET, POST, PUT and DELETE), to issue fetch, update and delete of employee records.
* The user authentication (login/signup) and employee data retrieval are being handled by API calls.
* Uses Retrofit for easy and seamless API interactions and parsing JSON.

## Asynchronous Processing with Coroutines

* We use Kotlin Coroutines to easily work with the database queries and network requests.
* It makes sure the main thread isn’t blocked for any long duration, hence improving the UI responsiveness.
* Accesses and fetches data in a way that optimizes with structured concurrency and suspend functions.

# COMPONENTS & FUNCTIONALITY

## 1. Login/Signup Screen

* Users authenticate with the username and password.
* Provides error messages for incorrect credentials.
* Once authenticated successfully, it takes the user to the Employee List Screen.

## 2. Main Screen (Employee List)

* A list of employees is displayed through a scrollable card-based user interface.
* Each employee card portrays details like name, department, and designation.
* Touching an employee card will lead to the Employee Details Screen.

## 3. Add Employee Screen

* User enters employee details that are saved in a database.
* There are validations to prevent missing or faulty data.

## 4. Employee Details Screen

* Displays detailed employee information.
* Along with the option to edit or delete the employee record.
* Shows a confirmation alert before deleting an employee.

## 5. Edit Employee Screen

* Allows users to edit existing employee details.
* Updates changes both on the local and remote server.

## 6. Employee Deleting Functionality

* A confirmation alert is shown so that employees are not inadvertently deleted.
* When the confirmation is given, the employee is removed from both local storage and the remote database.

# CHALLENGES & Solutions

## Challenge 1: Implementing Secure Authentication with REST API

**Issue:** Problem-Well secured handling of authentication tokens, user sessions, and API requests. **Solution:**

* Used JWT tokens for session management.
* Implemented secure API headers to send authentication credentials.
* Used Postman to send test requests to the APIS before implementation into the app.

## Challenge 2: Ensuring Offline Data Access

**Issue:** Problem-Users should be able to access and amend employee records even in offline mode.

**Solution:**

* Room Database for local storage.
* Once connection is restored, a syncing mechanism is implemented to update records.

## Challenge 3: API Errors and Network Failures Handling

**Issue:** The API failure may halt the UI response or even leave some transactions incomplete.

**Solution:**

* Implemented error handling via try-catch blocks in coroutines.
* User-friendly toast messages and alerts for error reporting.

# APP SETUP

## 1) Environment Configuration for Development

* Install Android Studio (latest recommended version).
* Ensure that Kotlin and Jetpack Compose are enabled.

## 2) Setting up the Project

* Unzip the project folder and open it in Android Studio.
* Connect to GitHub for version control (this step can be skipped).
* Sync project dependencies via Gradle:

File -> Sync Project with Gradle Files

* Each of the necessary dependencies (Retrofit, Room, Jetpack Compose) should now be installed.

## 3) Application Execution

* Connect a physical Android device or fetch up the emulator (AVD).
* Run the application from Android Studio by clicking on the run button.
* Grant necessary permissions (internet access, local storage, etc.).

## 4) Test Authentication and Data Management

* To log in, use test credentials, and check if the user is authenticated.
* Ensure that all add, edit, and delete operations concerning employee records work as they are expected.
* API logs can be checked by using Logcat to debug the network transactions.