**Automated Attendance System Flutter Project**

**Supervisor**

* Engineer Abdullah Khalid
* Dr. Aamir Irshad

**Authors**

* Muhammad Ahsan Aftab, Air University
* Engineer Abdullah Khalid

1. **Introduction**

**Project Overview**

The Automated Attendance System is an innovative Flutter-based application designed to modernize and streamline the process of tracking attendance. Traditional attendance methods, which often rely on manual sign-ins or timecards, can be inefficient and prone to errors. This application offers a digital solution, allowing users to mark their attendance automatically through a user-friendly interface. Leveraging Firebase for real-time data storage, the system can be easily integrated with various other databases, providing flexibility for different organizational needs.

**Objectives**

The primary objectives of this system are:

* **Efficiency:** Automating attendance tracking to save time and reduce administrative burden.
* **Accuracy:** Minimizing errors associated with manual entry methods, ensuring accurate attendance records.
* **Comprehensive Monitoring:** Providing administrators with a powerful tool to track and manage employee attendance, including real-time data access and historical records.
* **Security:** Implementing secure user authentication and data management to protect sensitive information.
* **Scalability:** Designing the system to be adaptable, allowing for future enhancements or integration with additional services.

1. **Application Architecture**

**High-Level Architecture Diagram**

*(Insert a visual diagram of the system components and their interactions here, such as user devices, Firebase services, and the facial recognition API)*

**Components Description**

**Frontend**

The frontend is built using **Flutter**, an open-source UI software development kit by Google. Flutter enables the creation of natively compiled applications for mobile, web, and desktop from a single codebase. Key components of the frontend include:

* **Login System:** A secure login system with role-based access control.
* **Home Screen:** A user-friendly interface where employees can mark their attendance or view their attendance history.
* **Employee Records:** Detailed records that can be accessed by administrators, showing individual and overall attendance trends.

**Backend**

The backend is managed using **Firebase**, a platform developed by Google for creating mobile and web applications. Firebase handles several critical backend functionalities:

* **Authentication:** Managing user sign-ins, role assignments, and secure access to different sections of the app.
* **Data Storage:** Storing user data, attendance records, and other related information in a structured manner.
* **Cloud Functions:** Serverless functions that perform automated tasks like updating records, sending notifications, or generating reports.

**Database**

The primary database used is **Firebase Firestore**, a flexible, scalable database for mobile, web, and server development. Firestore provides real-time updates, allowing the application to respond immediately to changes in data. Key aspects of the database include:

* **User Data:** Storing user profiles, including personal information, roles, and authentication details.
* **Attendance Records:** Maintaining logs of attendance, including timestamps, user IDs, and verification results from the facial recognition system.

1. **Key Features**

**Login System**

The application incorporates a robust login system to authenticate users securely. This system differentiates between two main user roles:

* **Admins:** Users with full access, capable of managing employee records, adding or removing users, and viewing detailed attendance reports.
* **Normal Users:** Employees who can access their own attendance records and update personal information.

**Code Example for User Sign-In**

*(Insert the Flutter/Dart code snippet for the user sign-in process, highlighting role-based access control)*

**Role-Based Access Control (RBAC)**

The system implements Role-Based Access Control (RBAC) to ensure that users can only access the functionalities permitted by their roles. This enhances security and ensures that sensitive data is protected from unauthorized access.

**Password Security**

To ensure data security, passwords are not stored as plain text. Instead, they are hashed using secure algorithms before being stored in the database. This prevents unauthorized access even if the database is compromised.

1. **System Functionality**

**Attendance Marking**

One of the standout features of this application is its attendance marking system, which uses facial recognition technology powered by Google's ML Kit. The process involves:

* **Facial Recognition:** Employees take a photo using their device’s camera. The photo is processed to verify their identity using machine learning algorithms.
* **Attendance Logging:** Once verified, the system logs the attendance automatically, including the date, time, and user ID.
* **Error Handling:** The system includes fallback mechanisms for cases where facial recognition fails, ensuring that attendance can still be marked through alternative methods like manual entry by an admin.

**Profile Management**

Users have access to a profile management system where they can:

* **View Profile Information:** Including full name, email, phone number, and profile picture.
* **Edit Profile Details:** Users can update their contact information and profile picture.
* **Role Management (Admin):** Admins can update user roles, granting or restricting access as necessary.

**Real-Time Updates**

Thanks to Firebase Firestore's real-time capabilities, any changes made to user data or attendance records are instantly reflected across the application. This ensures that all users have access to the most up-to-date information without needing to refresh or reload the app.

**Notifications**

The system can be extended to include a notification feature, where users are alerted about important updates, such as changes to their attendance records, upcoming deadlines, or system maintenance.

**Reporting**

Admins can generate detailed attendance reports, which can be exported for further analysis or archival purposes. Reports can include daily, weekly, or monthly summaries, and can be filtered based on various criteria such as departments, individual employees, or specific time periods.

1. **Future Enhancements**

While the current system meets the basic requirements of an attendance tracking system, there are several potential enhancements:

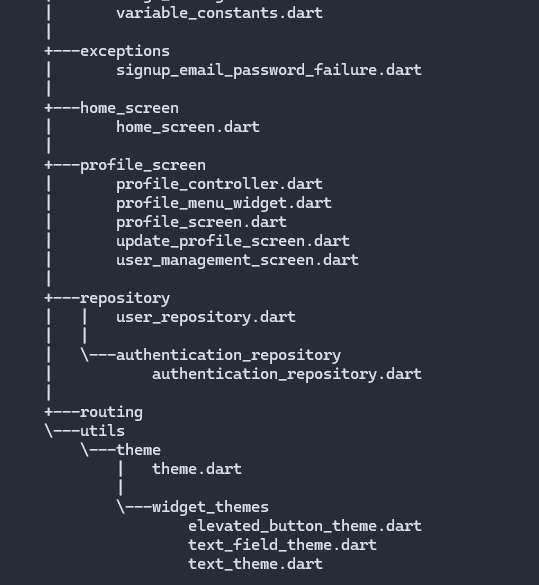
* **Geofencing:** Integrating location-based services to ensure that attendance can only be marked from specific locations.
* **Multi-Factor Authentication:** Adding additional security layers to the login process, such as OTP or biometric authentication.

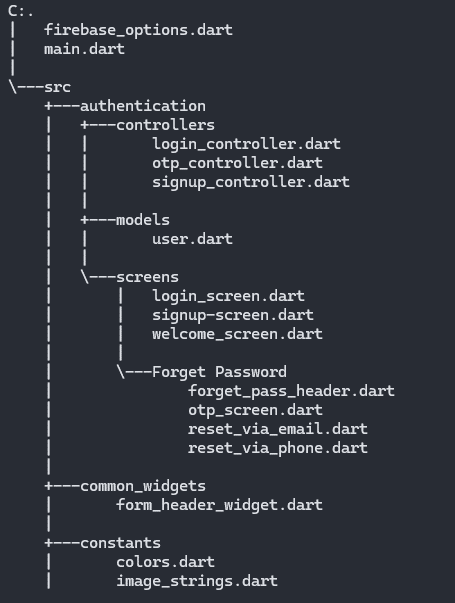
1. **Project Flow**

**Folder Structure:**

The folder structure for the Automated Attendance System project is organized as follows:

### **Overview of Key Directories and Files**





* **firebase\_options.dart:** Contains Firebase configuration options required for initializing Firebase services.
* **main.dart:** The entry point of the Flutter application, setting up initial routing and app configurations.

**/src/authentication**

* **controllers:** Contains business logic for authentication, such as login\_controller.dart, otp\_controller.dart, and signup\_controller.dart.
* **models:** Holds the user.dart model, defining the structure of user data.
* **screens:** UI components for authentication, including login\_screen.dart, signup\_screen.dart, and welcome\_screen.dart.
* **Forget Password:** Components for handling password reset functionality, including OTP verification and resetting via email or phone.

**/src/common\_widgets**

* **form\_header\_widget.dart:** Reusable widgets for consistent form headers across different screens.

**/src/constants**

* **colors.dart, image\_strings.dart, variable\_constants.dart:** Define reusable constants for colors, image paths, and other variables, ensuring consistency across the app.

**/src/exceptions**

* **signup\_email\_password\_failure.dart:** Custom exceptions related to sign-up failures, improving error handling and debugging.

**/src/home\_screen**

* **home\_screen.dart:** The main dashboard screen where users interact with the core features of the application.

**/src/profile\_screen**

* **profile\_controller.dart:** Handles business logic related to user profiles.
* **profile\_menu\_widget.dart:** Custom widget for displaying the profile menu.
* **profile\_screen.dart, update\_profile\_screen.dart, user\_management\_screen.dart:** UI components for managing user profiles and user information.

**/src/repository**

* **user\_repository.dart:** Centralizes data access related to user information, separating the data layer from business logic.
* **authentication\_repository.dart:** Manages authentication-related data, providing a clear separation of concerns.

**/src/routing**

* Handles navigation and routing within the app, ensuring a modular and maintainable structure.

**/src/utils/theme**

* **theme.dart:** Centralized theme management for consistent styling.
* **widget\_themes:** Contains individual themes for widgets like elevated buttons, text fields, and text, providing a unified look and feel.

This folder structure ensures a modular and organized codebase, making the project easier to maintain, scale, and debug. Each directory and file has a clear purpose, contributing to the overall functionality and user experience of the Automated Attendance System.