Safety Report Mobile System

Software Architecture Document
Version 1.0

Revision History

Date	Version	Description	Author
16/08/2023	1.0	Software Architecture Initial Document	Kovalev Alexey

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a high-level overview of the system's architecture, using multiple views to represent different aspects of the system. It aims to capture and communicate the key architectural decisions that have been made for the system.

1.2 Scope

This document provides an architectural overview of the SafeReport Mobile system. SafeReport Mobile is a system being developed to support proactive safety management for a technical services company, according to client needs and preferences.

2. Architectural Goals and Constraints

The following are some key requirements and system constraints that have a significant bearing on the architecture:

- Accident Reporting: Allows users to report accidents, capturing location, description, and actions taken.
- Safety Concern Reporting: Enables users to report safety concerns or propose improvements.
- View All Reports (Admin Only): Administrators can view and manage all submitted reports.
- Data Export (Admin Only): Administrators may require the ability to export reports for analysis.
- Push Notifications: Enable notifications for users/administrators to receive updates or reminders.
- User Authentication: Implements secure login mechanisms to ensure privacy and data security.
- Emergency Call: Users can initiate an emergency call with one click.
- Offline Support: Works offline with data synchronization when internet connectivity is available.
- Responsive UI: Provides an intuitive user interface.

Image Name: Demo Screenshots

3. Use-Case View

A description of the use-case view of the software architecture. The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

3.1 Architecturally Significant Use Cases

- Report Safety Concern (User)
- Report Accident (User)
- Make Emergency Call (User)
- User Authentication (User, Administrator)
- View All Reports (Administrator)
- Export Data (Administrator)
- Customize Reporting Form (Administrator)
- View Analytics Dashboard (Administrator, Management)

These use cases are initiated by the user, or the administrator actors.
In addition, interaction with external actors; management and IT department occur.

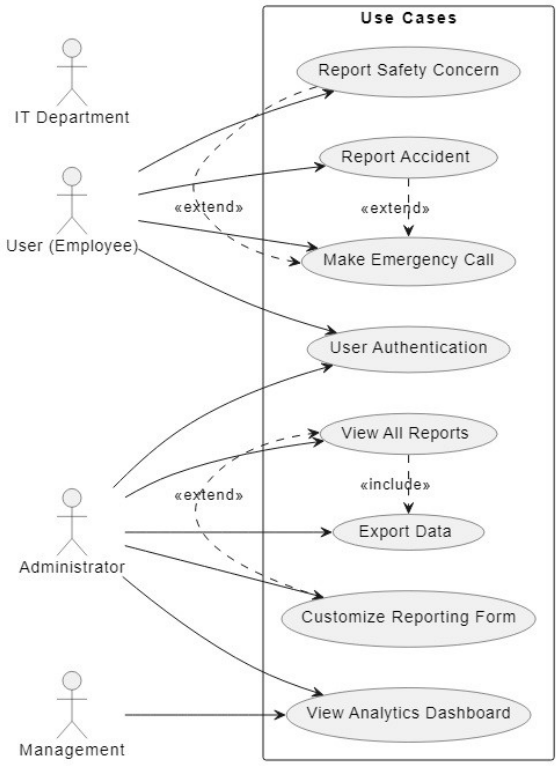


Diagram Name: Architecturally Significant Use-Cases

4. Model-View-ViewModel Architecture

The "SafeReport Mobile" application employs the Model-View-ViewModel (MVVM) architectural pattern to structure its design. MVVM is a design pattern that separates the user interface (View) from the underlying data (Model) and introduces a middle layer called ViewModel to manage and expose the data to the View.

4.1 Key Components of MVVM in the "SafeReport Mobile" App

4.1.1 Model: The Model represents the data and business logic of the application. It includes entities like User, AccidentReport, and others. In the app, the Room database handles data persistence, and entities are defined in Kotlin data classes.

4.1.2 View: The View is responsible for the user interface and how data is presented to the user. It includes activities like LoginActivity, HomeActivity, ListReportsActivity, and more. XML layout files define the UI components.

4.1.3 ViewModel: The ViewModel acts as a bridge between the View and the Model. It holds and manages the UI-related data that the View needs to display. The ViewModel exposes methods and LiveData for the View to interact with. For instance, the ListReportsViewModel could provide methods to retrieve user-specific or all reports.

4.2 Advantages of MVVM in the "SafeReport Mobile" App

4.2.1 Separation of Concerns: MVVM separates UI-related logic from the data and business logic, leading to better code organization and maintainability.

4.2.2 Testability: With MVVM, the ViewModel holds the app's logic, making it easier to test without involving the UI components.

4.2.3 Reusability: ViewModel instances are more reusable as they can be used across different Views without modifications.

4.2.4 LiveData: MVVM often utilizes LiveData to manage UI-related data changes. LiveData provides data updates to the View components, ensuring an up-to-date user interface.

4.2.5 Data Binding: MVVM can be combined with data binding to streamline the UI code and minimize the need for manual updates between the View and ViewModel.

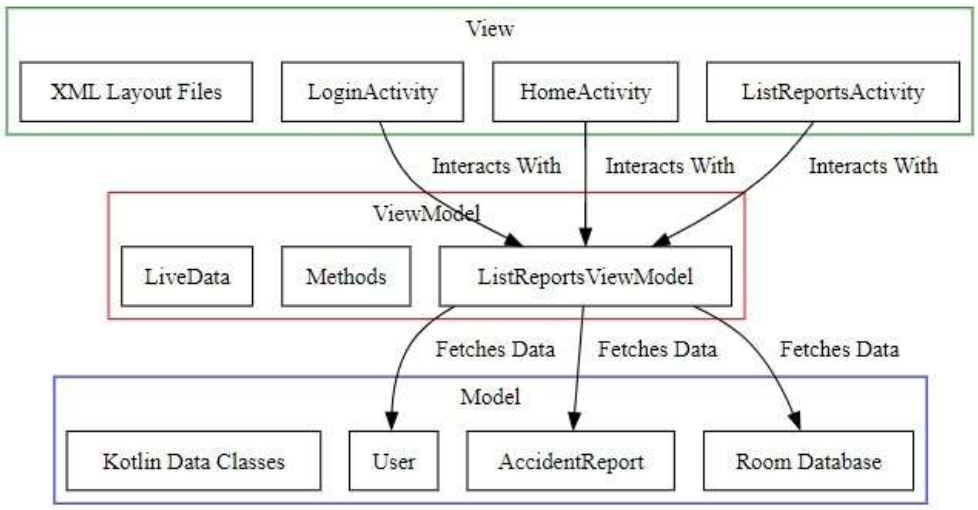


Diagram Name: MVVM Pattern

5. Client-Server Architecture

This architectural pattern divides the system into two major components: the client and the server. Here's a description of how this pattern is implemented in the app:

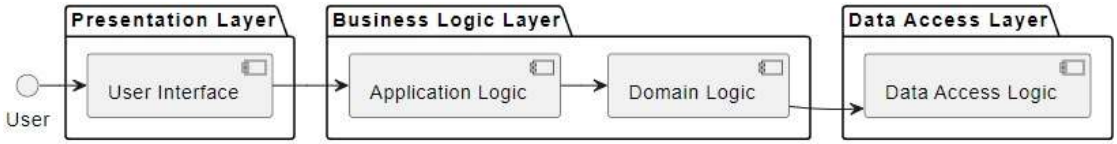
5.1 Client Component:

The client component is the user interface where users interact with the application. It includes activities, UI elements, and user interactions.
The app's activities, such as Login, Home, Report Accident, List Reports, and others, form the client component.
Users interact with the app through various UI elements, buttons, forms, and navigation.

5.2 Server Component:

The server component is responsible for processing requests from clients, performing business logic, and providing necessary data and services.
In the app, the server component is represented by the back-end services that handle operations like user authentication, data storage, and data retrieval.
For example, the Room database handles data persistence, while the user authentication logic is implemented in the server component.

5.3 Architecture Pattern Overview – Package and Subsystem Layering



5.3.1 Presentation Layer

This layer is responsible for interacting with the user and presenting the data to them. It includes the user interface (UI), such as the screens and menus that the user sees.

5.3.2 Business Logic Layer

This layer implements the business rules of the system. It determines what actions can be performed by the user and how the data is processed.

5.3.3 Data Access Layer

This layer provides access to the data stored in the system. It interacts with the database or other data sources to retrieve and store data.

6. Process and Class View

A description of the process view and class hierarchy of the architecture. Describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. Also describes the allocation of objects and classes to tasks.

6.1 Class and File Diagrams:

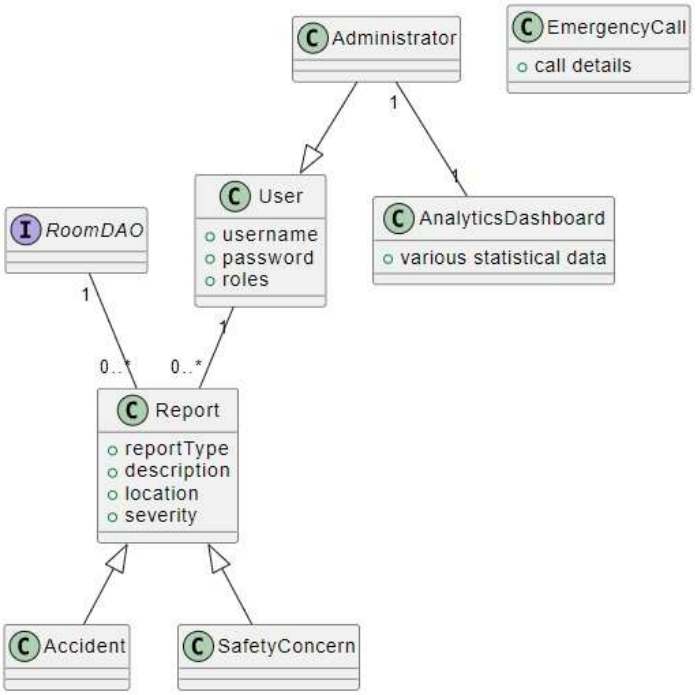


Diagram Name: Classes Diagram

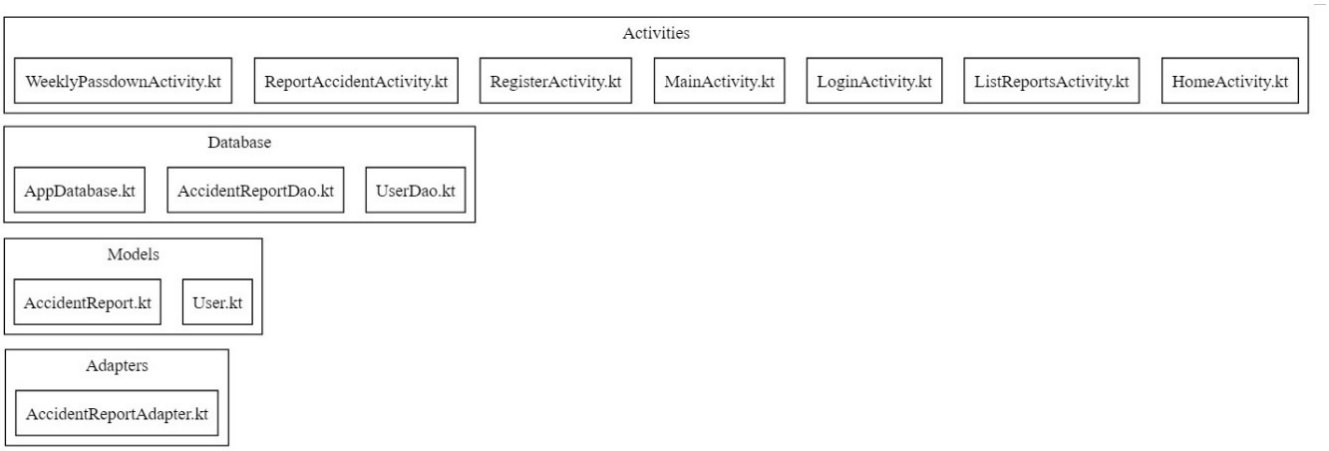
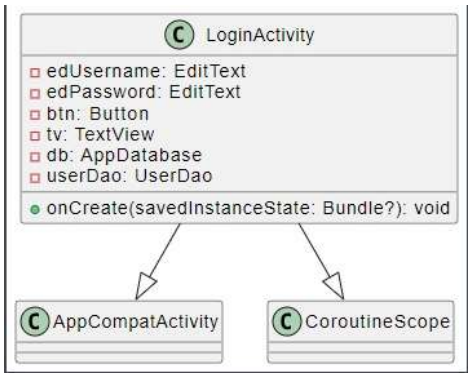


Diagram Name: File Hierarchy

6.1 LoginActivity Class



6.1.1 **edUsername**: An EditText object that represents the input field for the username.

6.1.2 **edPassword**: An EditText object that represents the input field for the password.

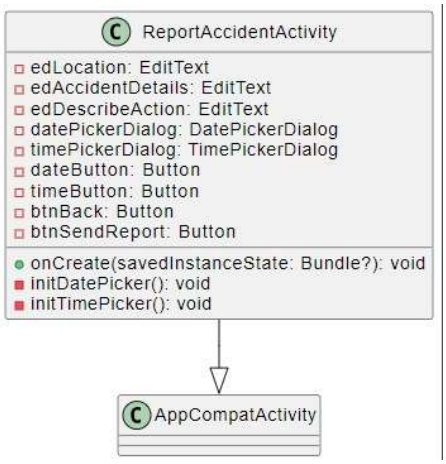
6.1.3 **btn**: A Button object that represents the login button.

6.1.4 **tv**: A TextView object that represents the text view for new users.

6.1.5 **db**: An AppDatabase object that represents the application's database.

6.1.6 **userDao**: A UserDao object that represents the data access object for users.

6.2 ReportAccidentActivity Class

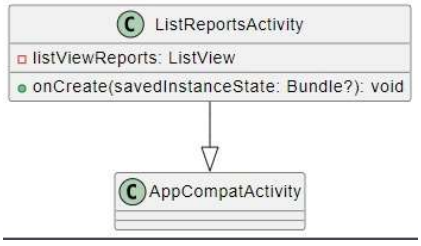


edLocation: An EditText object that represents the input field for the accident location.
edAccidentDetails: An EditText object that represents the input field for the accident details.
edDescribeAction: An EditText object that represents the input field for the action taken.
datePickerDialog: A DatePickerDialog object that represents the date picker dialog.
timePickerDialog: A TimePickerDialog object that represents the time picker dialog.
dateButton: A Button object that represents the date button.
timeButton: A Button object that represents the time button.
btnBack: A Button object that represents the back button.
btnSendReport: A Button object that represents the send report button.

6.3 ListReports Class

6.3.1 **listViewReports**: A ListView object that represents the list view for displaying the accident reports.

6.3.2 **onCreate(savedInstanceState: Bundle?):** This method is called when the activity is first created. It initializes the user interface elements, sets up the list view, and handles user interactions such as clicking the back button. It retrieves the logged-in username from shared preferences and fetches the accident reports based on the user's role. If the user is a manager, it fetches all accident reports; otherwise, it fetches only the user's own accident reports. The fetched reports are then displayed in the list view using an AccidentReportAdapter.



6.3.3 **Inheritance**
The ListReportsActivity class extends the AppCompatActivity class, which is a base class for activities that use the support library action bar features. This allows the activity to use features from the Android support library, such as backward-compatible action bars.

7. Sequence Diagrams

7.1 Login Activity:

7.1.1 User Authentication Attempt:
User enters username and password and clicks "Login."
Empty fields prompt a "fill all details" message.

7.1.2 Authentication Process:
Entered credentials are sent to the database for verification.
User DAO checks for a matching user.

7.1.3 Authentication Outcome Handling:
If valid user found: Admin status determines redirection.
User's username stored in shared preferences.
Invalid credentials prompt a "Invalid Username and/or password" message.

7.1.4 Register Link Option:
User can click "Register" to navigate to Register Activity.

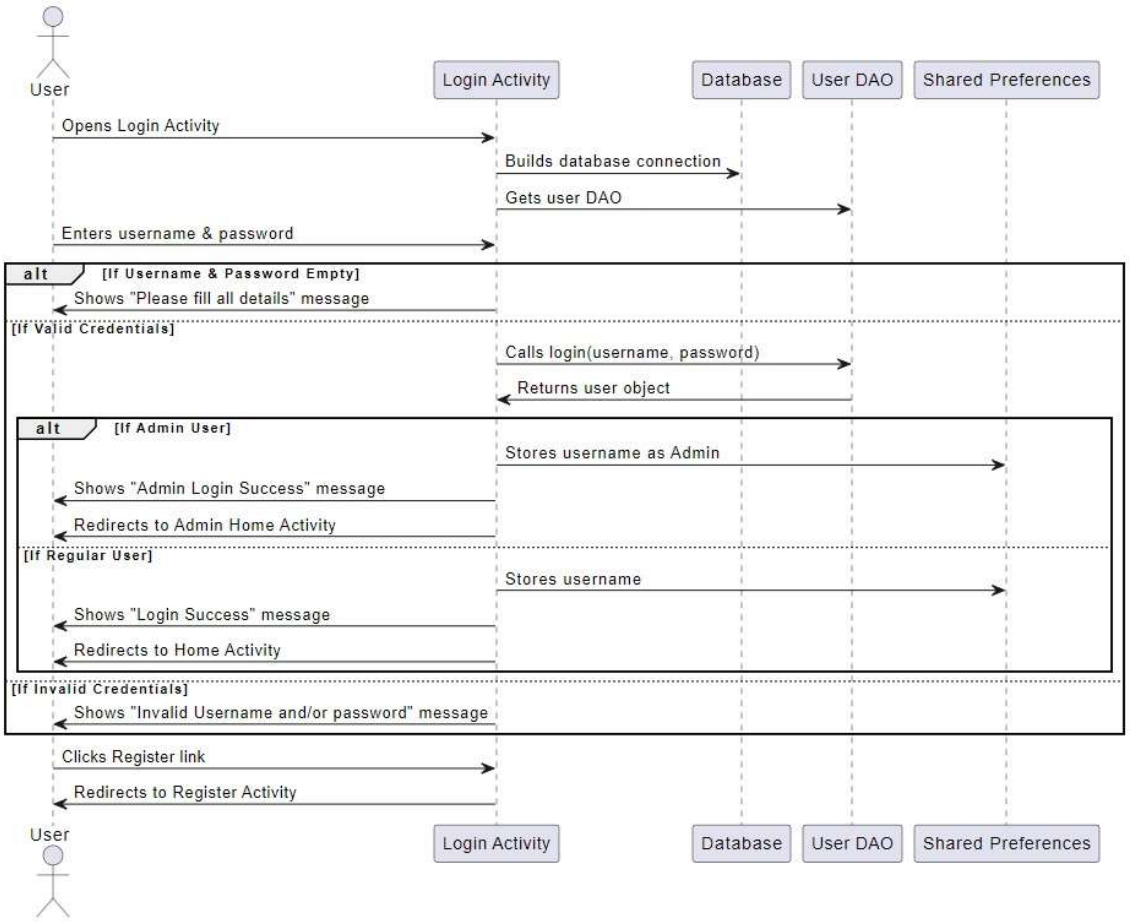


Diagram Name: Login Activity Sequence Diagram

7.2 Home Activity:

7.2.1 User Home Screen Entry:

Upon Successful login, the user is directed to the Home Activity, where they can access various features.

7.2.2 Personalized Greeting and Shared Preferences:

The Home Activity retrieves the username stored in shared preferences.
A personalized welcome message is displayed, addressing the user by their username.

7.2.3 Card Interaction - Log Out:

The user can interact with the "Log Out" card by tapping it.
Upon interaction, shared preferences are cleared, the user is redirected to the Login Activity.

7.2.4 Card Interaction - Emergency Call:

The user can interact with the "Emergency Call" card by tapping it.
An implicit intent is triggered, initiating a dialer to call emergency services.

7.2.5 Card Interaction - Report Accident:

The user can interact with the "Report Accident" card by tapping it.
Upon interaction, the user is redirected to the Report Accident Activity to submit an a report.

7.2.6 Card Interaction - All Reports:

The user can interact with the "All Reports" card by tapping it.
Upon interaction, the user is redirected to the List Reports Activity to view accident reports.

7.2.4 Card Interaction - Weekly Passdown:

The user can interact with the "Weekly Passdown" card by tapping it.
Upon interaction, the user is redirected to the Weekly Passdown Activity to access passdown information.

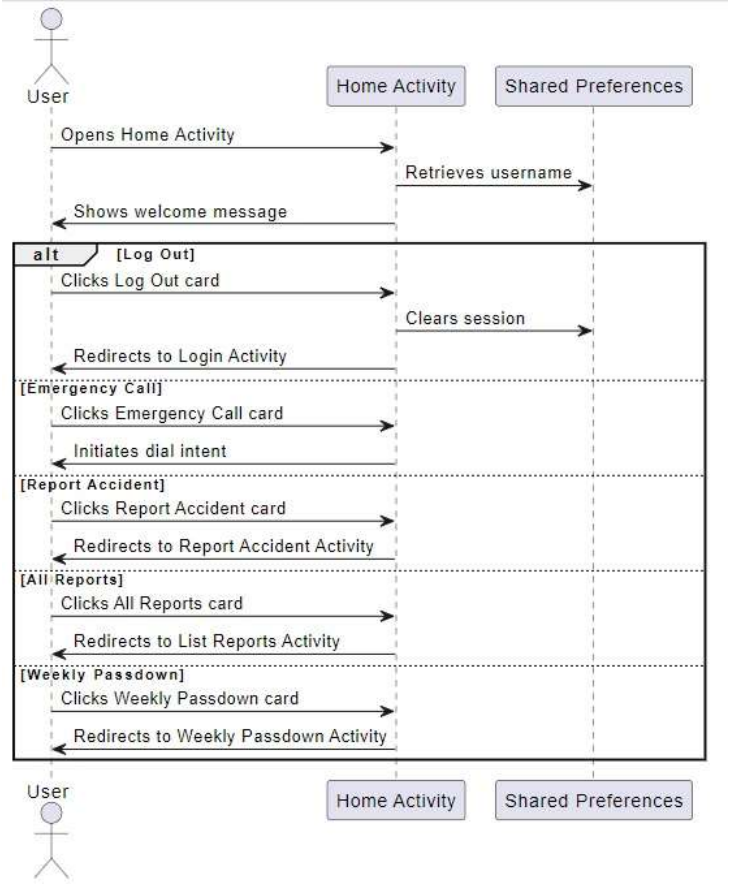
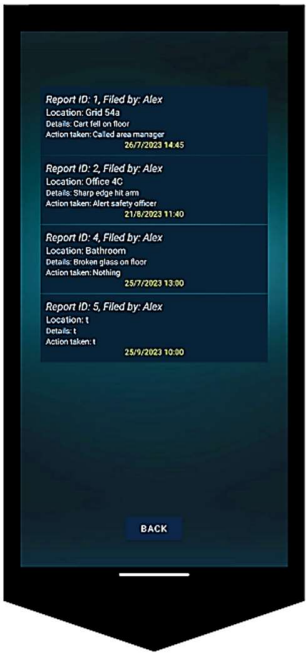


Diagram Name: Home Activity Sequence Diagram

7.3 List Report Activity:

7.3.1 User Enters List Reports Activity:

Upon selecting the "All Reports" card from the Home Activity, users, both regular and admins, are directed to the List Reports Activity.



7.3.2 Shared Preferences Retrieval:

The List Reports Activity retrieves the username of the logged-in user from shared preferences, catering to both regular users and admins.

7.3.3 Database Query for User Reports:

A coroutine initiates a database query, focusing on retrieving accident reports linked to the logged-in user's username.

7.3.4 User-Specific Reports Display:

The List Reports Activity presents the fetched accident reports specific to the user in a ListView, powered by the AccidentReportAdapter.
Each report's details, encompassing location and description, are showcased.

7.3.5 Admin Privilege - Comprehensive Report Access:

Admin users benefit from a distinctive privilege in this phase.
They are granted access to a comprehensive view, enabling them to explore and analyze all reports submitted by all users.

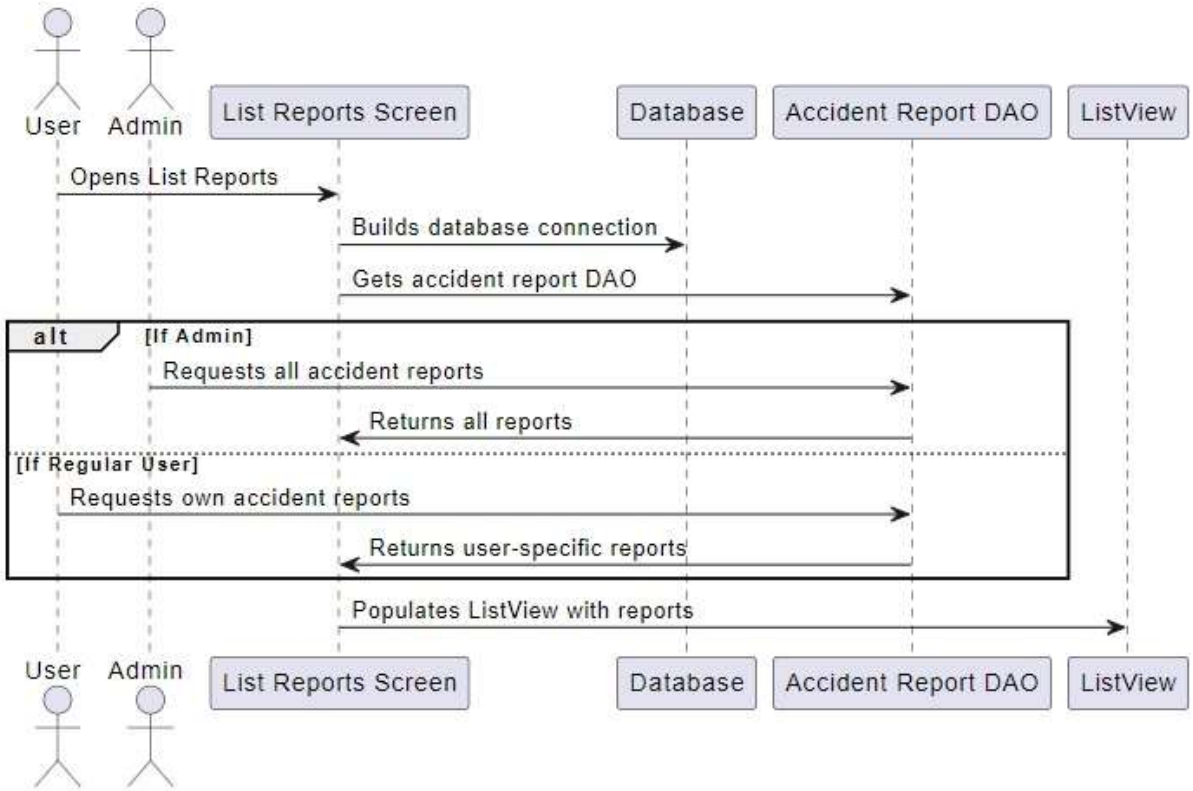


Diagram Name: List Reports Activity Sequence Diagram