**Personal Project Report**

|  |
| --- |
| **Application Name** |
| Hospital Appointment System |
| **Application Explanation**   * What your application? * App usage scenario |
| The Hospital Appointment System is an android application designed to streamline the process of scheduling appointments with doctors across various specialties. The app enables users to view available doctors, select preferred specialists and book appointments at convenient times.   **User Registration and Login**: Users create an account or log in to an existing account.   **Specialty Selection**: Users select the medical specialty they require.   **Doctor Selection**: Users choose a doctor from the list of available doctors in the selected specialty.   **Appointment Booking**: Users select a preferred time slot and book an appointment.   **Appointment Management**: Users can view and manage their appointments. |
| **Activity Explanation**   * What activities is in your application? * Role of each activity * Design of each activity in aspect of UI/UX |
| **Activities and their roles:**   * **LoginActivity**: Facilitates user authentication, registration, and navigation to the main app. * **MainActivity**: Serves as the central hub for navigating to other parts of the app like booking appointments, viewing appointments, and searching for doctors. * **AppointmentFragment**: Provides a user interface for selecting specialties, doctors, and booking appointment times. * **DoctorSearchActivity**: Offers search functionality to find doctors based on user input. * **AppointmentListActivity**: Displays and manages the appointments booked by the user.  Design of each activity in aspect of UI/UX:  * **LoginActivity**: Simple and intuitive design with text fields for email and password, and buttons for login and registration. * **MainActivity**: Dashboard-style layout with buttons for quick access to various functionalities like booking appointments and viewing appointments. * **AppointmentFragment**: Uses a combination of GridLayout for specialties and CardViews for time slots, ensuring a clean and organized appearance. * **DoctorSearchActivity**: Features a search bar at the top and a RecyclerView to display search results dynamically. * **AppointmentListActivity**: List-based UI with clear, readable appointment details and options for managing appointments. |
| **API Usage**   * From which point your application connect with Open API(or your own server)? * Why you used it? |
| The application connects with **Firebase** for authentication, real-time database management, and cloud storage. This integration occurs primarily in the login, registration, and appointment booking functionalities.  Firebase was chosen for its robust, scalable, and easy-to-integrate backend services, including:   * **Authentication**: Secure and simple user authentication. * **Realtime Database**: Efficient real-time data synchronization across multiple clients. * **Cloud Storage**: Seamless management of user data. |
| **Implementation Challenge**   * From where you took time while implementing your application? * What is your special-care-point?  (code optimization, user-friendly UI, … / specify code file&line or activity name) |
| From where did you take time while implementing your application?  * **User Authentication**: Implementing secure and efficient user authentication using Firebase. * **UI/UX Design**: Ensuring a user-friendly and intuitive design across all activities. * **Real-time Data Sync**: Handling real-time updates and synchronization of appointment data.  What is your special-care-point?  * **Code Optimization**: Ensuring efficient and maintainable code, especially in AppointmentFragment.kt where appointment booking, and specialty filtering logic are implemented. * **User-friendly UI**: Emphasis on clean and intuitive interfaces across all activities to enhance user experience. |
| **Application Design**   * Overall design of your android application * It can be design pattern, code separation, or so on. |
| Overall design of your android application: The application follows an MVVM (Model-View-ViewModel) architecture:   * **Model**: Represents the data layer, including Firebase database and authentication. * **View**: Comprises the UI components (activities and fragments) that render data to the user. * **ViewModel**: Manages the communication between the model and the view, handling business logic and data processing.  Design pattern, code separation, or so on:  * **Separation of Concerns**: Different activities and fragments handle specific tasks, promoting modularity. * **Data Binding**: Used to bind UI components to data sources, facilitating a reactive UI. * **LiveData**: Ensures the UI reflects the latest data updates. |
| **Server Design (Optional)**   * (Write only you used your own server like AWS) Overall design of your server * It can be design pattern, code separation, or so on. |
| * **Serverless Architecture**: Utilizes Firebase for serverless backend services, reducing the need for dedicated server management. * **Real-time Database**: Firebase Realtime Database provides real-time data synchronization. * **Authentication**: Firebase Authentication ensures secure and scalable user management.  Design pattern, code separation, or so on  * **Microservices**: Firebase functions can be used to create microservices for handling specific tasks like sending notifications or processing data. |

* You can freely add the row in the table above.