**System Design Document (SDD)**

**Title: System Design for Cross-Platform Scheduling Application**

**1. Introduction**

**1.1 Purpose** The purpose of this document is to provide a detailed system design for a cross-platform scheduling application that synchronizes daily schedules between a Windows PC and an Android device.

**1.2 Scope** This document covers the system architecture, design considerations, and detailed design of the components and modules of the application.

**1.3 Definitions, Acronyms, and Abbreviations**

* SDD: System Design Document
* UI: User Interface
* API: Application Programming Interface
* SDK: Software Development Kit
* JSON: JavaScript Object Notation

**1.4 References**

* Firebase Documentation: https://firebase.google.com/docs
* Android Development Documentation: <https://developer.android.com/docs>
* Windows Application Development Documentation: <https://docs.microsoft.com/en-us/windows/apps/>
* React Documentation: https://reactjs.org/docs/getting-started.html

**2. System Architecture**

**2.1 High-Level Architecture**

The system architecture is divided into three main components:

* **Frontend**: Responsible for the user interface and interaction with the user.
  + Android Application
  + Windows Application
* **Backend**: Responsible for data storage, user authentication, and synchronization.
  + Firebase Services (Authentication, Firestore, Cloud Functions)
* **Synchronization**: Ensures real-time data synchronization between the Android and Windows applications.

**2.2 Component Diagram**

**3. Design Considerations**

**3.1 Performance**

* Optimize data synchronization to ensure minimal latency.
* Implement efficient data retrieval and storage mechanisms.

**3.2 Scalability**

* Design the system to handle up to 1,000 active users.
* Utilize Firebase's scalable infrastructure.

**3.3 Security**

* Ensure data encryption during transmission and storage.
* Implement secure authentication mechanisms.

**3.4 Usability**

* Provide an intuitive and user-friendly interface.
* Ensure consistency across Android and Windows applications.

**4. Detailed System Design**

**4.1 Frontend Design**

**4.1.1 Android Application**

* **UI Components**:
  + Task List View: Displays a list of tasks.
  + Task Detail View: Displays detailed information about a task.
  + Task Editor: Allows users to create and edit tasks.
* **Technology Stack**:
  + Kotlin/Java for development
  + XML for UI design
  + Firebase SDK for authentication and database interactions

**4.1.2 Windows Application**

* **UI Components**:
  + Task List View: Displays a list of tasks.
  + Task Detail View: Displays detailed information about a task.
  + Task Editor: Allows users to create and edit tasks.
* **Technology Stack**:
  + C# and WPF for development
  + XAML for UI design
  + Firebase SDK for authentication and database interactions

**4.2 Backend Design**

**4.2.1 Firebase Authentication**

* **Functions**:
  + User Registration
  + User Login
  + Password Reset

**4.2.2 Firestore Database**

* **Data Structure**:

{

"users": {

"userId": {

"tasks": {

"taskId": {

"title": "Task Title",

"description": "Task Description",

"startTime": "Timestamp",

"endTime": "Timestamp",

"recurring": "Daily/Weekly/Monthly"

}

}

}

}

}

**4.2.3 Cloud Functions**

* **Functions**:
  + Data Synchronization: Ensure data consistency across devices.
  + Notification Triggers: Send push notifications for upcoming tasks.

**4.3 Synchronization Design**

**4.3.1 Real-Time Synchronization**

* Use Firestore's real-time capabilities to sync data between devices.
* Implement listeners on both Android and Windows applications to detect changes and update the UI accordingly.

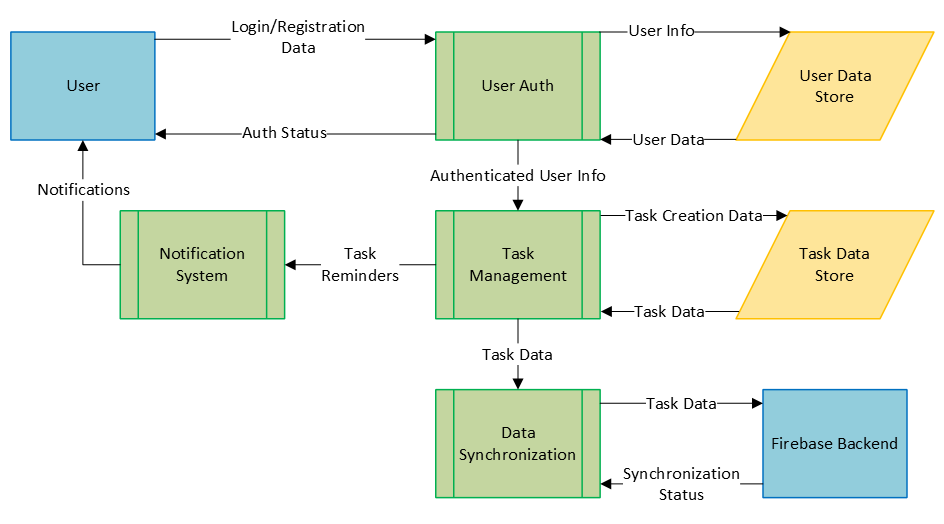
**5. Data Flow Diagrams**

**5.1 Level 0 DFD (Context Diagram)**

A diagram of a software application

Description automatically generated

**5.2 Level 1 DFD (Detailed Diagram)**



**6. User Interface Design**

**6.1 Android Application UI Mockups**

* **Task List View**
* **Task Detail View**
* **Task Editor**

**6.2 Windows Application UI Mockups**

* **Task List View**
* **Task Detail View**
* **Task Editor**

**7. Testing and Validation**

**7.1 Unit Testing**

* Write unit tests for individual components and functions.
* Ensure high code coverage.

**7.2 Integration Testing**

* Test the interaction between frontend and backend components.
* Ensure data synchronization works seamlessly.

**7.3 User Acceptance Testing (UAT)**

* Conduct UAT with a group of users.
* Gather feedback and make necessary improvements.

**8. Deployment**

**8.1 Android Application**

* Deploy the application on the Google Play Store.
* Provide installation instructions for side-loading during testing phases.

**8.2 Windows Application**

* Create an installer for the Windows application.
* Provide installation instructions for testing and final deployment.

**9. Maintenance and Updates**

**9.1 Bug Fixes**

* Monitor user feedback and error logs.
* Release timely updates to fix bugs and improve performance.

**9.2 Feature Enhancements**

* Gather user suggestions for new features.
* Prioritize and implement new features based on user demand and feasibility.