# CS 255 System Design Document Template

## UML Diagrams

### UML Use Case Diagram

A diagram of a computer

AI-generated content may be incorrect.

### UML Activity Diagrams

*A diagram of a workflow

AI-generated content may be incorrect.* A diagram of a flowchart

AI-generated content may be incorrect.

### UML Sequence Diagram

*A diagram of a diagram

AI-generated content may be incorrect.*

### UML Class Diagram

*A diagram of a program

AI-generated content may be incorrect.*

## Technical Requirements

DriverPass is designed for Android smartphones and tablets running API level 21 or higher. Devices must support local data storage and SMS functionality to enable core features. Development targets API level 28 to ensure compatibility with advanced system components such as Camera and Telephony APIs.

The app is built using Java within Android Studio and uses SQLite for persistent, local data management. The database scheme includes foreign key constraints to maintain relational integrity between tables—such as linking student profiles to scheduled appointments and test results.

System diagrams, created in Lucid chart, illustrate the relationships between UI components, controllers, and database layers. These diagrams follow a layered architecture, where user interactions are captured by the interface, processed by logic controllers, and stored asynchronously in the database to optimize performance.

Material Design principles guide the user interface for consistency and accessibility, while testing frameworks like JUnit and Espresso are used to validate functionality and ensure reliability across devices.

Runtime permissions are required for SMS and storage access, and the modular design supports future scalability, including potential integration with cloud-based services for backup and synchronization.

For example, the UML Class Diagram for DriverPass is built on modular, scalable infrastructures that support both local performance and future cloud integration. The system is designed to ensure data integrity, user accessibility, and efficiency component interaction across the layers.

Core Components:

* UI Layer (captures user input and displays feedback)
* Controller layer (processes logic, validates input, and coordinates between UI and database)
* Database Layer (Stores persistent data using SQLite, enforcing relations via foreign keys)