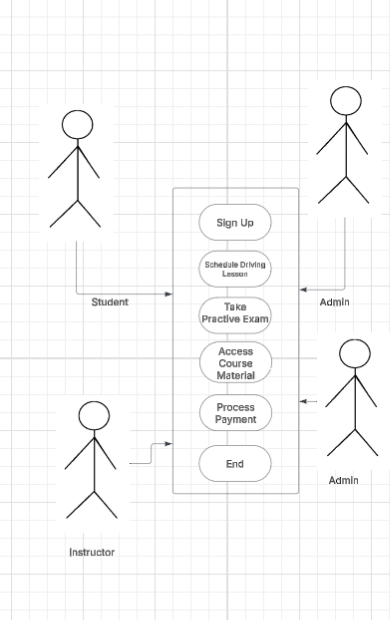
# CS 255 System Design Document Template

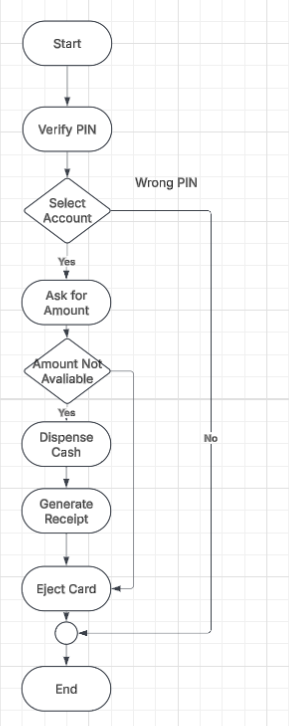
## UML Diagrams

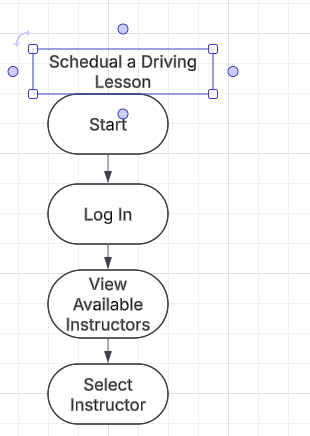
### UML Use Case Diagram

**

This use case diagram shows how different users interact with the DriverPass system. It includes major features such as scheduling lessons, taking practice tests, making payments, and accessing course materials.

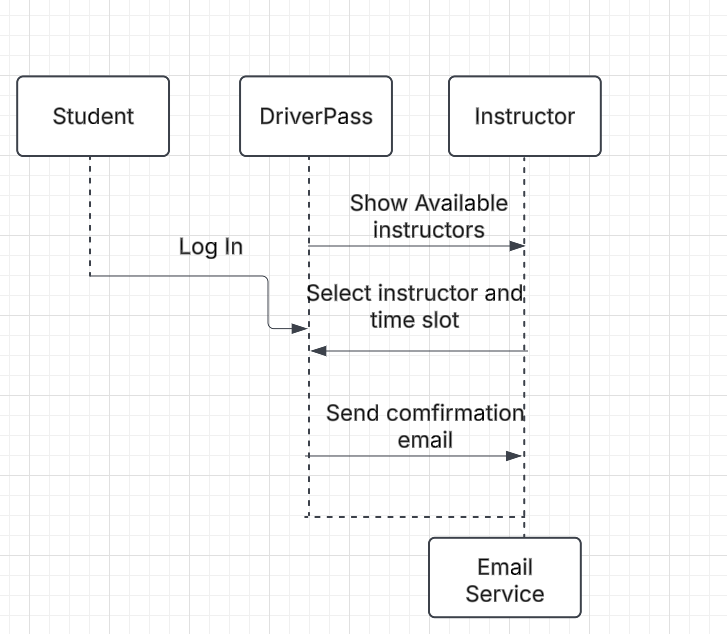
### UML Activity Diagrams

**This activity diagram outlines the step-by-step process of withdrawing cash from an ATM. It starts with PIN verification, account selection, and amount entry. It also includes decision points for incorrect PINs and cash availability. If the transaction is successful, the system dispenses cash and ejects the card. This diagram shows a secure and user-friendly withdrawal process.

**

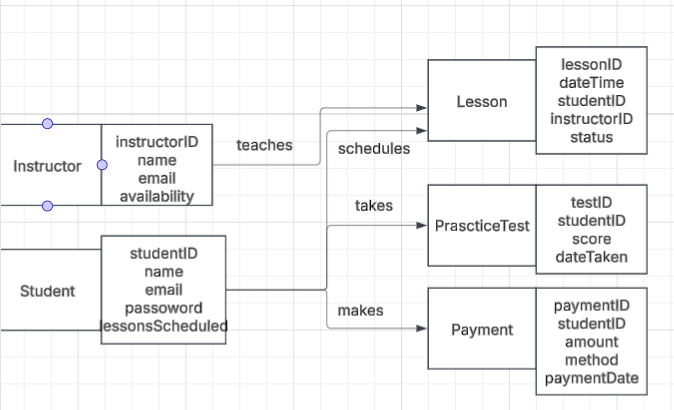
This diagram illustrates the process of scheduling a driving lesson in the DriverPass system. It starts with a student logging in, browsing available instructors, and selecting one along with a time slot. The process ends after confirming the appointment. This workflow supports DriverPass’s goal to simplify lesson scheduling and reduce manual coordination.

### UML Sequence Diagram

**

This sequence diagram shows the interaction between a student, the DriverPass system, an instructor, and the email service during the process of scheduling a driving lesson. It illustrates the order of operations, including login, selecting a lesson slot, and receiving confirmation via email. This interaction reflects the automation and ease-of-use requested by the client.

### UML Class Diagram

**

This class diagram defines the structure of the DriverPass system, organizing key entities such as Student, Instructor, Lesson, Payment, and PracticeTest. Each class includes essential attributes for storing and managing system data. This structure ensures efficient data relationships and supports the core system fun

ctionality requested by the client.

## Technical Requirements

To build the DriverPass system, we’d need a setup that supports both web and backend functionality, since the platform is meant to be accessible to students, instructors, and admins online. Based on the use cases and diagrams, here’s what would be needed to make the system work smoothly:

**Hardware:**

* A cloud-based web server (like AWS or Heroku) to host everything
* A database server (MySQL or PostgreSQL) for storing user accounts, lesson data, test scores, etc.
* Basic scalability options — like load balancing or storage backups — to handle growth if the platform gets more users

**Software & Tools:**

* **Frontend**: HTML, CSS, JavaScript (probably with React or another framework to make it responsive and user-friendly)
* **Backend**: Node.js with Express or Python with Django to handle all the logic and APIs
* **Database**: MySQL or PostgreSQL to manage all the structured data
* **Other Tools**:  
  + GitHub for version control
  + Lucidchart for creating all UML diagrams
  + PowerPoint for the client presentation

**Infrastructure Needs:**

* HTTPS for secure browsing
* Login system with user roles (students, instructors, admins) so access is limited appropriately
* API endpoints to connect the frontend with the backend
* Email integration (like SendGrid) to send confirmations after lessons are scheduled

Overall, the system needs to be fast, secure, and reliable — especially since it involves real user data and transactions. This setup gives DriverPass everything they need to meet client expectations and make the platform work efficiently.