# CS 255 System Design Document Template

This template lays out all the different sections that you need to complete for Project Two. Each section has guidance to prompt your thinking. You will need to continually reference the interview transcript as you work to make sure that you are addressing your client’s needs. There is no required length for the final document. Instead the goal is to complete each section based on what your client’s needs are. Remove this note when you are finished, and replace all bracketed text with the relevant information.

## UML Diagrams

## **UML Diagrams**

### **UML Use Case Diagram**

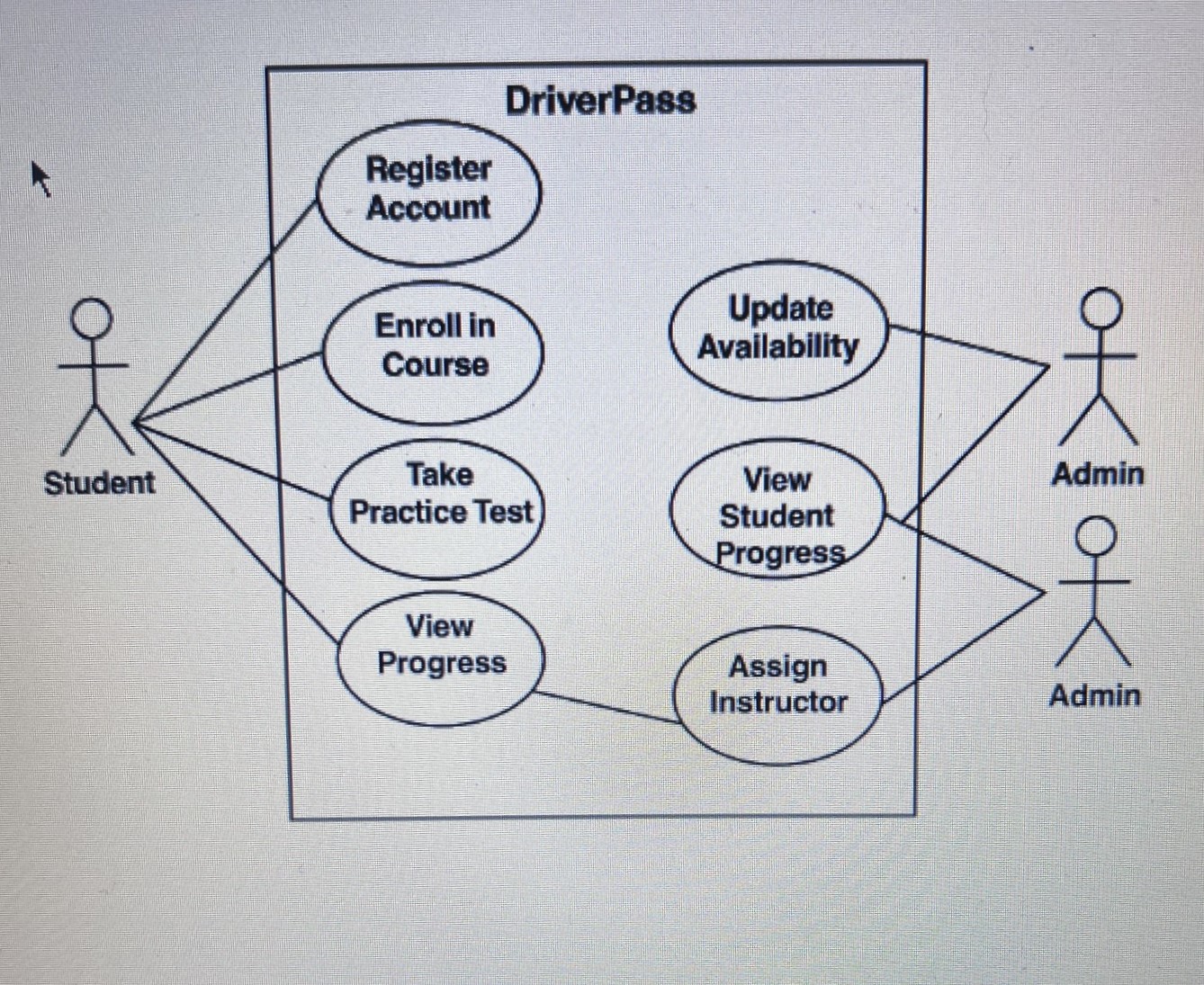
**Purpose:** Illustrate the interactions between users (actors) and the system's functionalities.

**Actors:**

* **Student:** Register, log in, select packages, schedule/cancel lessons, take practice tests.
* **Secretary:** Register students, schedule/cancel lessons on behalf of students.
* **Admin (Ian):** Manage user accounts, reset passwords, disable packages.
* **Driver:** View schedule, add lesson notes.

**Use Cases:**

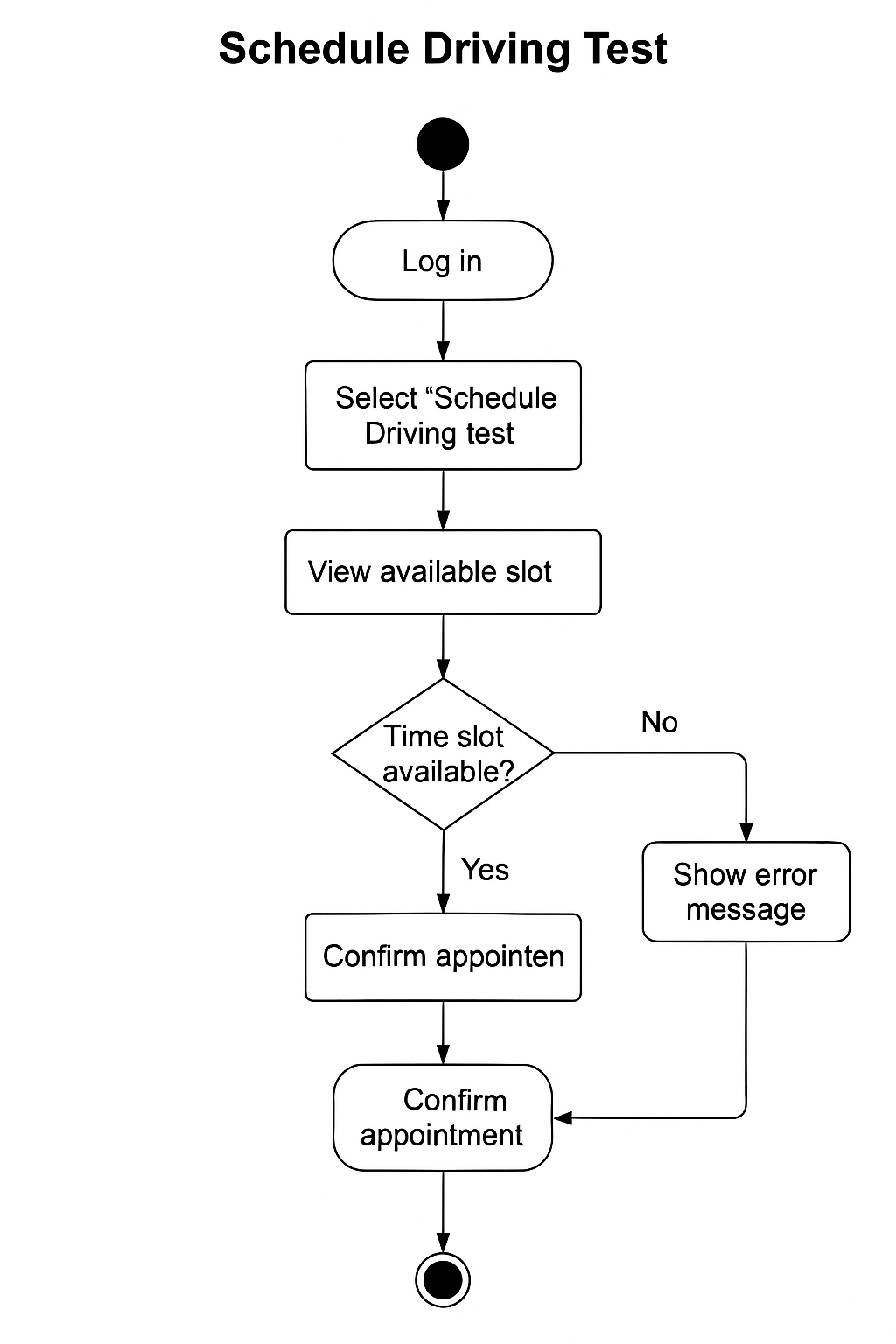
* **Register Account:** Students and secretaries can create new student accounts.
* **Log In:** Users authenticate to access the system.
* **Select Package:** Students choose from available training packages.
* **Schedule Lesson:** Students or secretaries schedule driving lessons.
* **Cancel Lesson:** Students or secretaries cancel scheduled lessons.
* **Take Practice Test:** Students access and complete online practice tests.
* **Manage Accounts:** Admin manages user accounts and access rights.
* **Add Lesson Notes:** Drivers input notes post-lesson.



**Activity Diagram 2: Scheduling a Driving Lesson**

**Flow:**

1. Student logs into the system.
2. Navigates to the scheduling section.
3. Selects preferred date and time.
4. Chooses a driver and vehicle (if applicable).
5. Confirms the appointment.
6. System checks for availability.
7. If available, appointment is scheduled; confirmation is sent.
8. If not, prompts student to select a different slot.



### **UML Sequence Diagram**

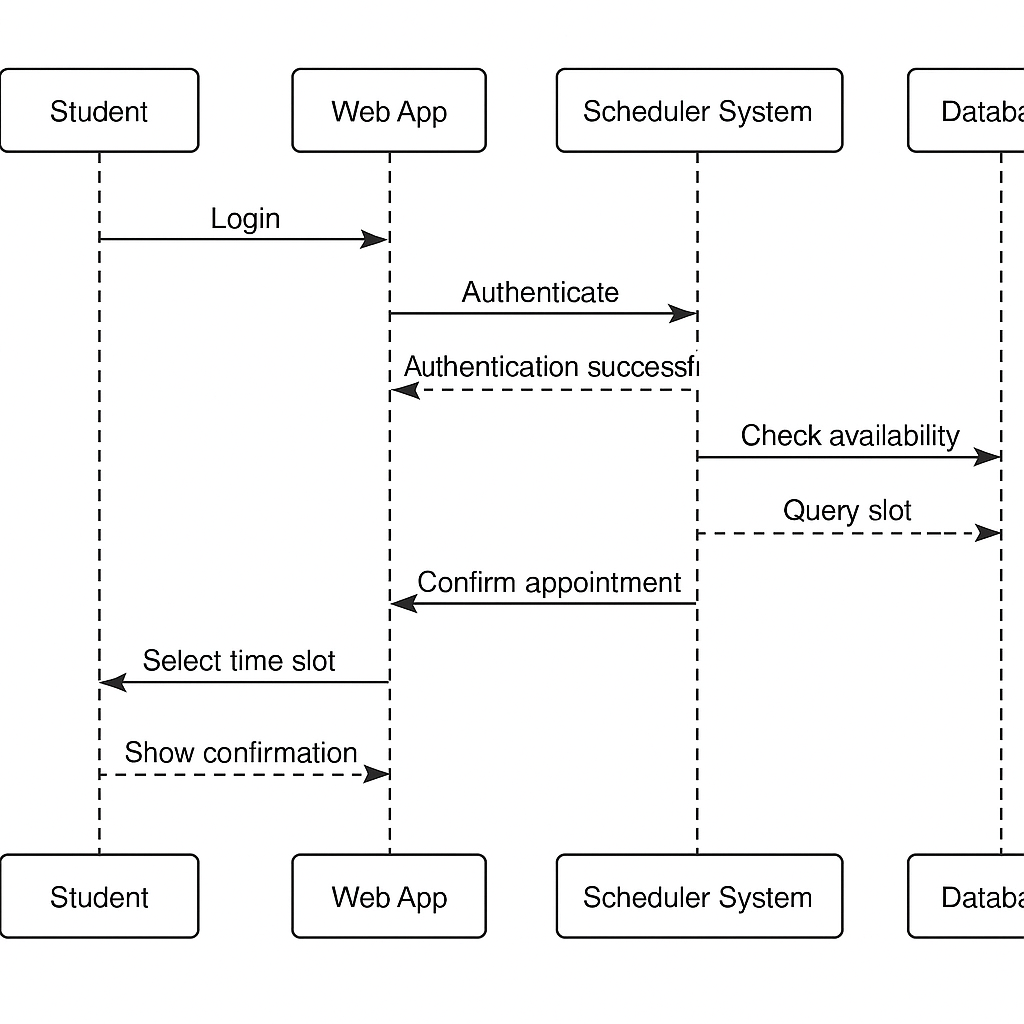
**Scenario: Student Scheduling a Driving Lesson**

**Participants:**

* Student
* Web Interface
* Scheduling System
* Database

**Sequence:**

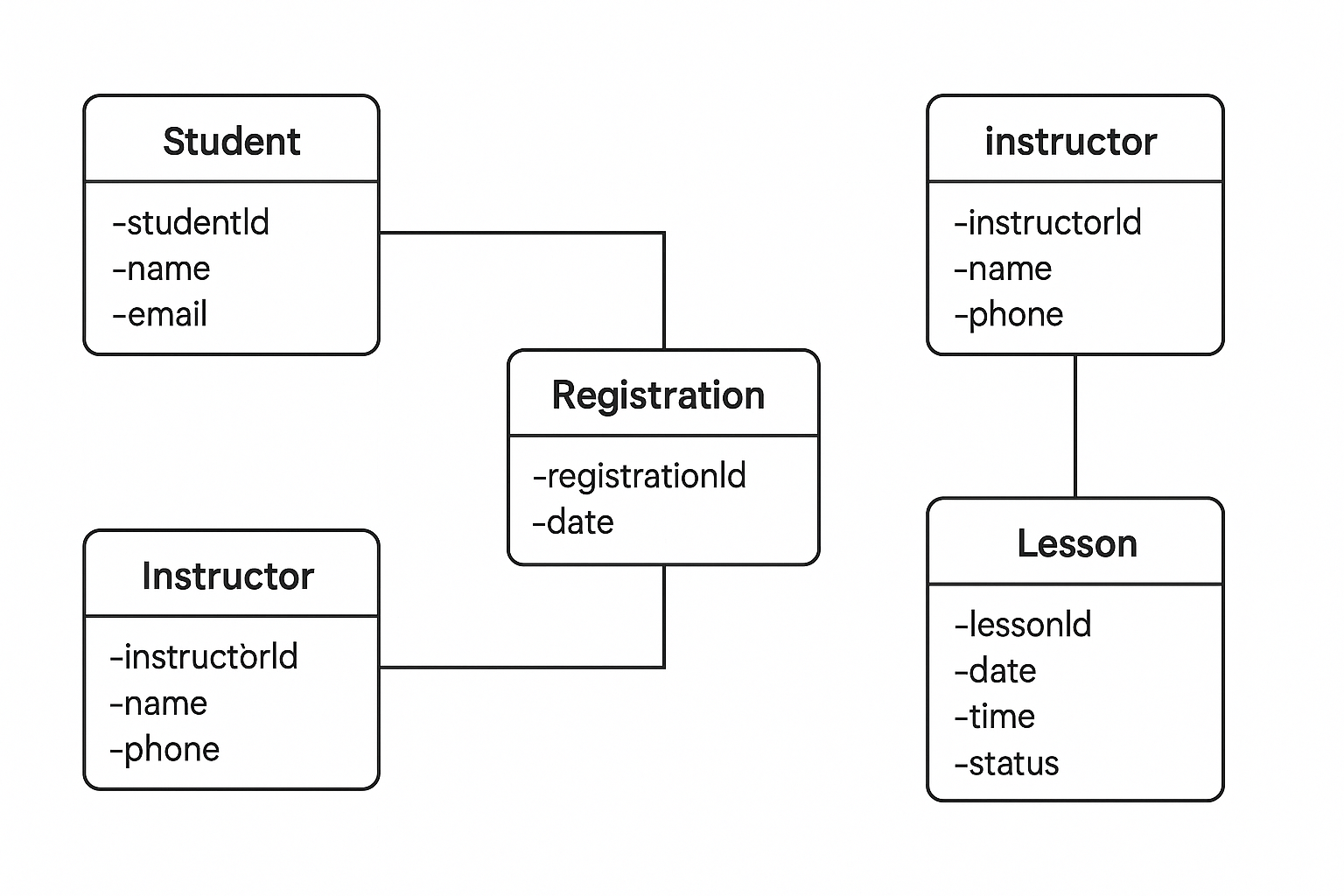
1. Student initiates login via the web interface.
2. System authenticates credentials against the database.
3. Upon successful login, student navigates to the scheduling module.
4. Student inputs desired date and time.
5. Scheduling system queries the database for availability.
6. If slot is available, system reserves the slot.
7. Confirmation is displayed to the student.



### **UML Class Diagram**

**Classes and Attributes:**

* **Student**
  + Attributes: studentID, name, address, contactInfo, loginCredentials
* **Package**
  + Attributes: packageID, description, duration, price, availabilityStatus
* **Lesson**
  + Attributes: lessonID, dateTime, studentID, driverID, vehicleID, status
* **Driver**
  + Attributes: driverID, name, licenseNumber, contactInfo
* **Vehicle**
  + Attributes: vehicleID, make, model, licensePlate
* **Admin**
  + Attributes: adminID, name, role, permissions



## Technical Requirements

* **Hardware Requirements:**
* Web servers to host the DriverPass application and manage client-server interactions
* A centralized relational database server to store user data, test schedules, course materials, and progress tracking
* Client devices such as PCs, laptops, tablets, or smartphones with internet access for students
* PCs or laptops for instructors and administrators to access scheduling and monitoring tools
* **Software Requirements:**
* Server operating system: Linux-based (Ubuntu Server) or Windows Server
* Client operating systems: Windows, macOS, iOS, or Android
* Frontend development: HTML5, CSS3, JavaScript (using React or Angular)
* Backend development: Java, Python (Django or Flask), or Node.js
* Relational database: MySQL or PostgreSQL
* Browser support: Chrome, Firefox, Safari, Edge
* **Tools:**
* Lucidchart for UML diagramming and system modeling
* Git with GitHub or GitLab for version control and source code management
* Visual Studio Code, IntelliJ IDEA, or Eclipse as development environments
* Selenium for automated web testing and Postman for API testing
* Jira or Trello for Agile project tracking and collaboration
* **Infrastructure:**
* Cloud hosting using AWS, Microsoft Azure, or Google Cloud Platform
* Services to support load balancing, auto-scaling, and system backups
* HTTPS encryption for secure web communication
* User authentication with role-based access control
* Secure database connections and daily backup scheduling
* Email service integration for test confirmations and notifications (e.g., SendGrid,