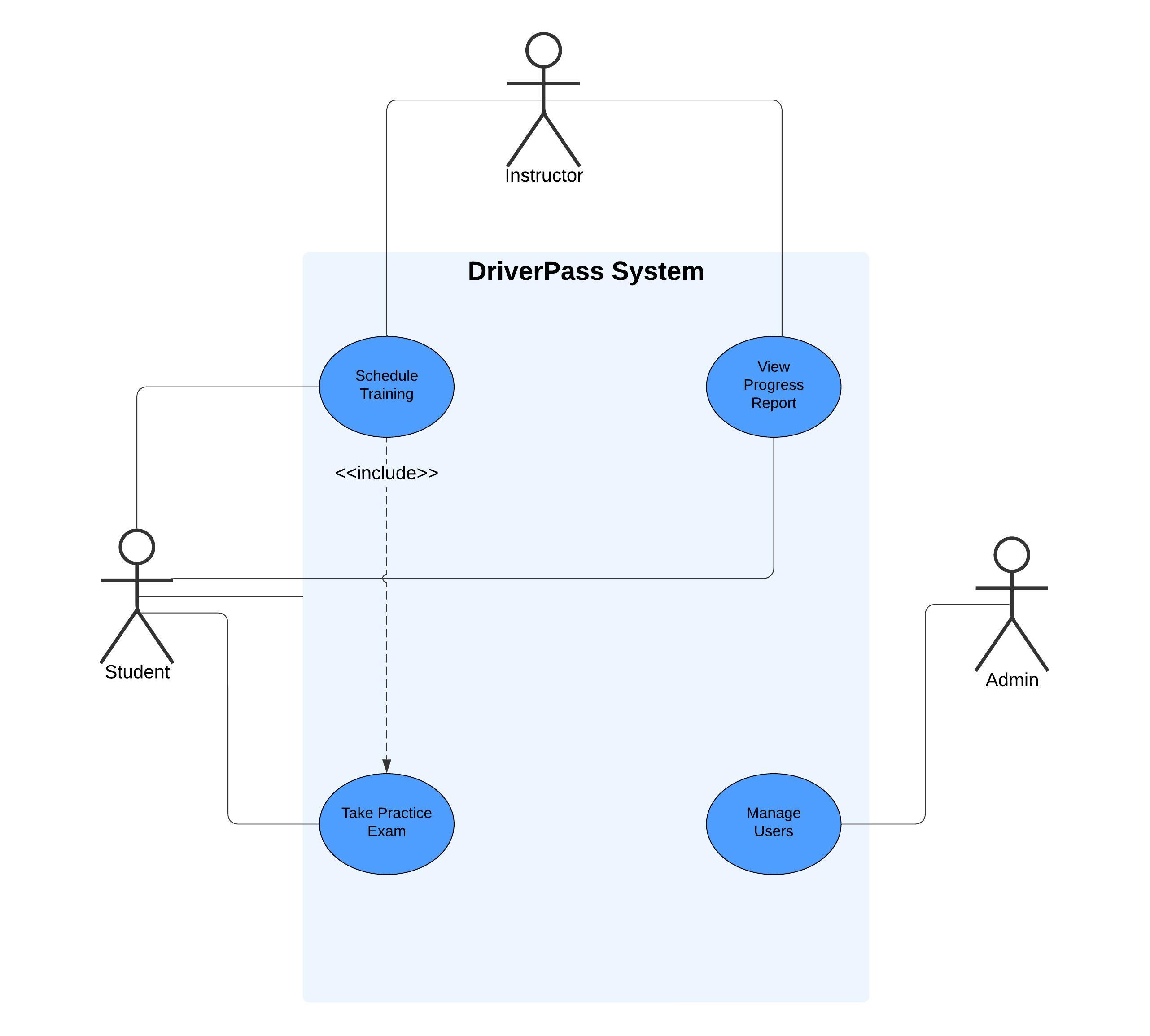
# CS 255 System Design Document Template

## UML Diagrams

**UML Use Case Diagram**

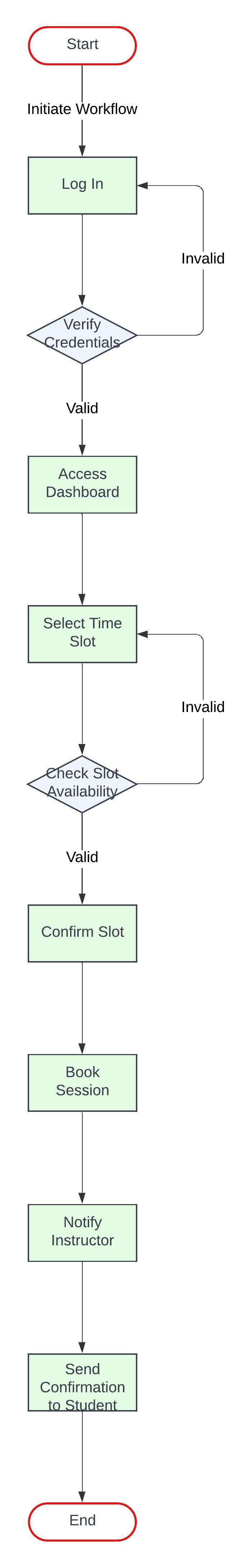
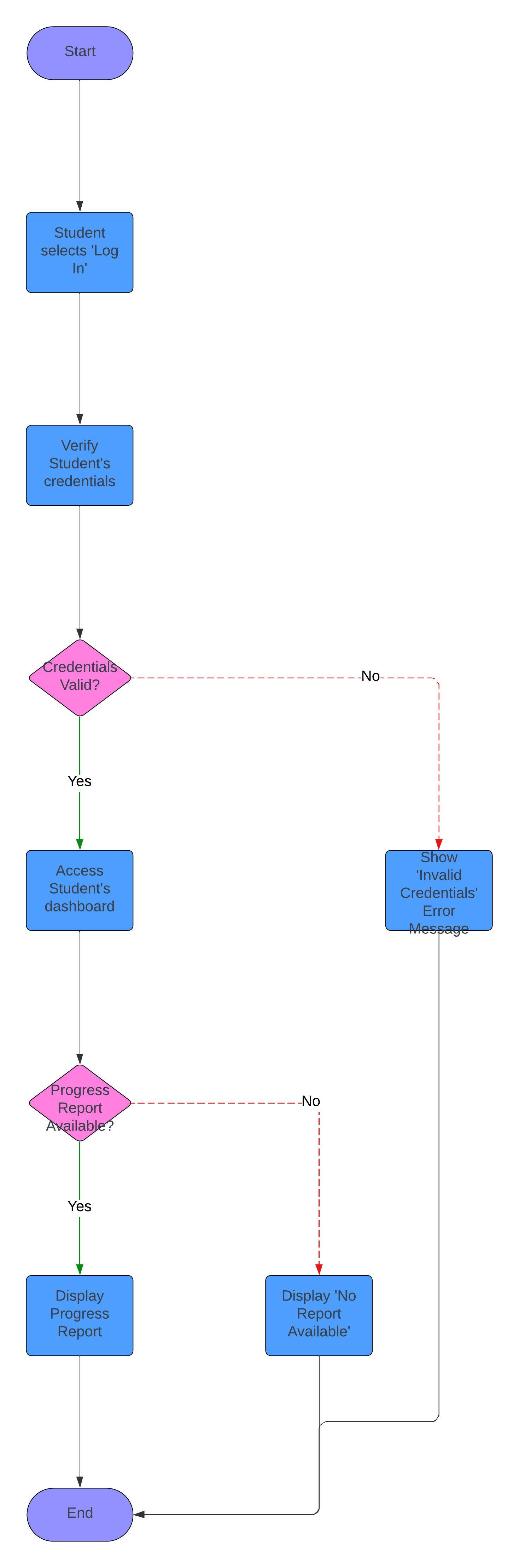
The use case diagram outlines the primary functionalities of the DriverPass System based on user roles and their interaction with the system. In the provided use case diagram, we can observe the system divided into different user categories such as **Admin**, **Instructor**, **Student**, and **DriverPass System**. Each actor has specific use cases, such as:

* **Student**: Logging in, viewing sessions, taking exams, etc.
* **Instructor**: Scheduling sessions, viewing student progress.
* **Admin**: Managing users, generating reports. The use case diagram is instrumental in visualizing how the users interact with the system and provides insight into their specific actions and the system's response

### UML Activity Diagrams

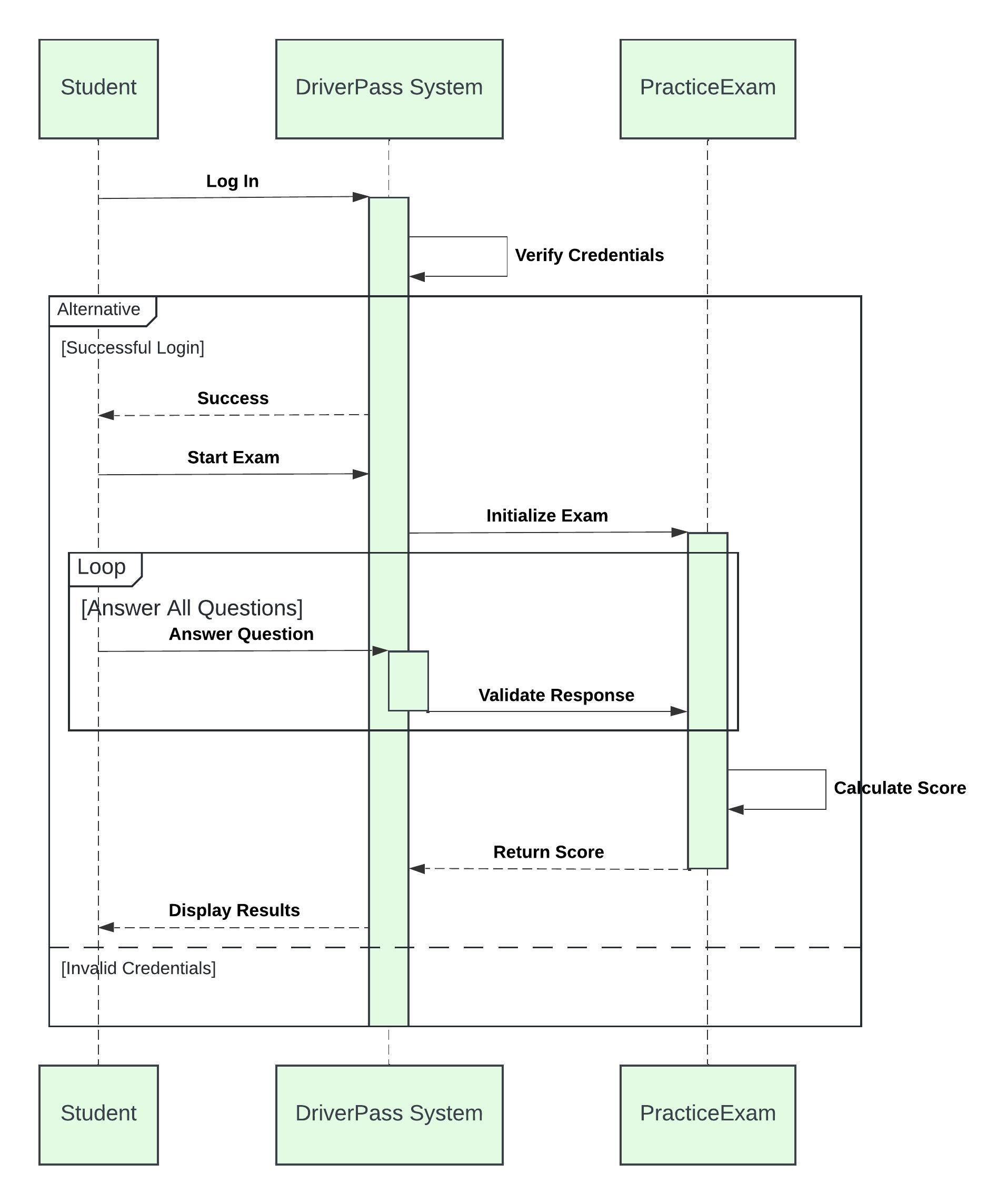
Activity diagrams illustrate the dynamic flow of control during key processes within the system. Two major activity diagrams were created:

* **Student Login and Progress Report View (left image)**: This diagram outlines the steps a student follows to log into the system and check their progress report. It highlights decision points such as whether the login credentials are valid and whether a report is available for the student to view.
* **Student Scheduling Workflow (right image)**: This diagram illustrates the sequence of activities involved when a student selects and books a time slot for a session. It includes steps such as verifying credentials, checking slot availability, and booking the session. This helps to ensure that the student can successfully interact with the scheduling system.



### UML Sequence Diagram

The sequence diagram for **Student taking a Practice Exam** was created to demonstrate the interaction between the **Student**, **DriverPass System**, and **PracticeExam**. It visualizes the messages exchanged during the login, exam initiation, question answering, validation of responses, and score calculation. This diagram emphasizes how the system responds to user actions in real time. It also highlights the use of loops and activation bars to show the system's state as the student answers multiple questions. This sequence diagram provides a clear view of how each system component interacts to complete a process.



### UML Class Diagram

The class diagram defines the structure of the system in terms of classes, their attributes, and the relationships between them. In the DriverPass System, classes include **Student**, **Instructor**, **Admin**, **DriverPass System**, **PracticeExam**, and **TrainingSession**.

* **Student** has attributes like **studentID** and **progressReport**, and methods like **viewSessions()** and **takeExam()**.
* **Instructor** has **instructorID**, **availability**, and methods like **scheduleSession()** and **viewStudentProgress()**.
* **TrainingSession** and **PracticeExam** represent core functionalities, storing session details and exam data.

The class diagram captures the structure and behavior of the system’s entities.

## 

## Technical Requirements

Based on the UML diagrams, the following technical requirements are necessary for the system:

1. **Hardware Requirements**
2. The DriverPass System requires the following hardware:
   1. **Servers**: Reliable servers with sufficient processing power to handle multiple users, exams, and student data.
   2. **Workstations**: The system should be accessible on any device (desktop, tablet, mobile) to allow users to log in and access their sessions, reports, and exams from anywhere.
3. **Software Requirements**

The system should run on a modern web application platform, with:

* 1. **Operating System**: Cross-platform compatibility (Windows, Linux, macOS).
  2. **Database Management**: A relational database such as MySQL or PostgreSQL to store user data, exam results, and session details.
  3. **Programming Languages**:
     1. **Frontend**: HTML5, CSS3, and JavaScript (with frameworks such as React or Angular for interactivity).
     2. **Backend**: Python (Django or Flask), JavaScript (Node.js), or Java for server-side processing and API development.
  4. **Authentication**: Secure login system, potentially using OAuth 2.0 or JWT for token-based authentication.
  5. **Version Control**: Git for source code management.

1. **Tools and Infrastructure**
   1. **Development Tools**: Integrated Development Environments (IDEs) like Visual Studio Code or PyCharm for coding.
   2. **API Integration**: RESTful APIs to connect different parts of the system (e.g., connecting the DriverPass System to the PracticeExam).
   3. **Hosting**: A cloud hosting service such as AWS, Google Cloud, or Azure to host the application and database.
   4. **CI/CD**: Continuous Integration/Continuous Deployment tools like Jenkins, GitHub Actions, or CircleCI for streamlined code deployment.
2. **Network Infrastructure**
   1. **Web Hosting and Security**: Ensure SSL certificates for encrypted data transmission, along with firewalls to protect against unauthorized access.
   2. **Backup and Recovery**: Implement regular backup procedures to ensure data integrity and recovery in case of failure.
3. **User Interface and User Experience**
   1. The system must have an intuitive user interface (UI) that allows easy navigation through logging in, selecting sessions, taking exams, and viewing reports.
   2. Mobile responsiveness for users to access the system via smartphones or tablets.
4. **Maintenance and Scalability**

### The system should be scalable to handle increasing numbers of students and instructors. Maintenance should be regularly planned to ensure that the system stays up-to-date and bug-free. **Final Thoughts**

The complete design includes use case, activity, sequence, and class diagrams that work together to represent the workflow and structure of the DriverPass System. The technical requirements ensure that the system is robust, scalable, and secure, providing a smooth experience for users across different devices and platforms. This design aims to meet the needs of all users—students, instructors, and admins—by streamlining their interactions and ensuring the system’s scalability, security, and overall usability.