# Southern New Hampshire University

By: James Michael Martin

System Design

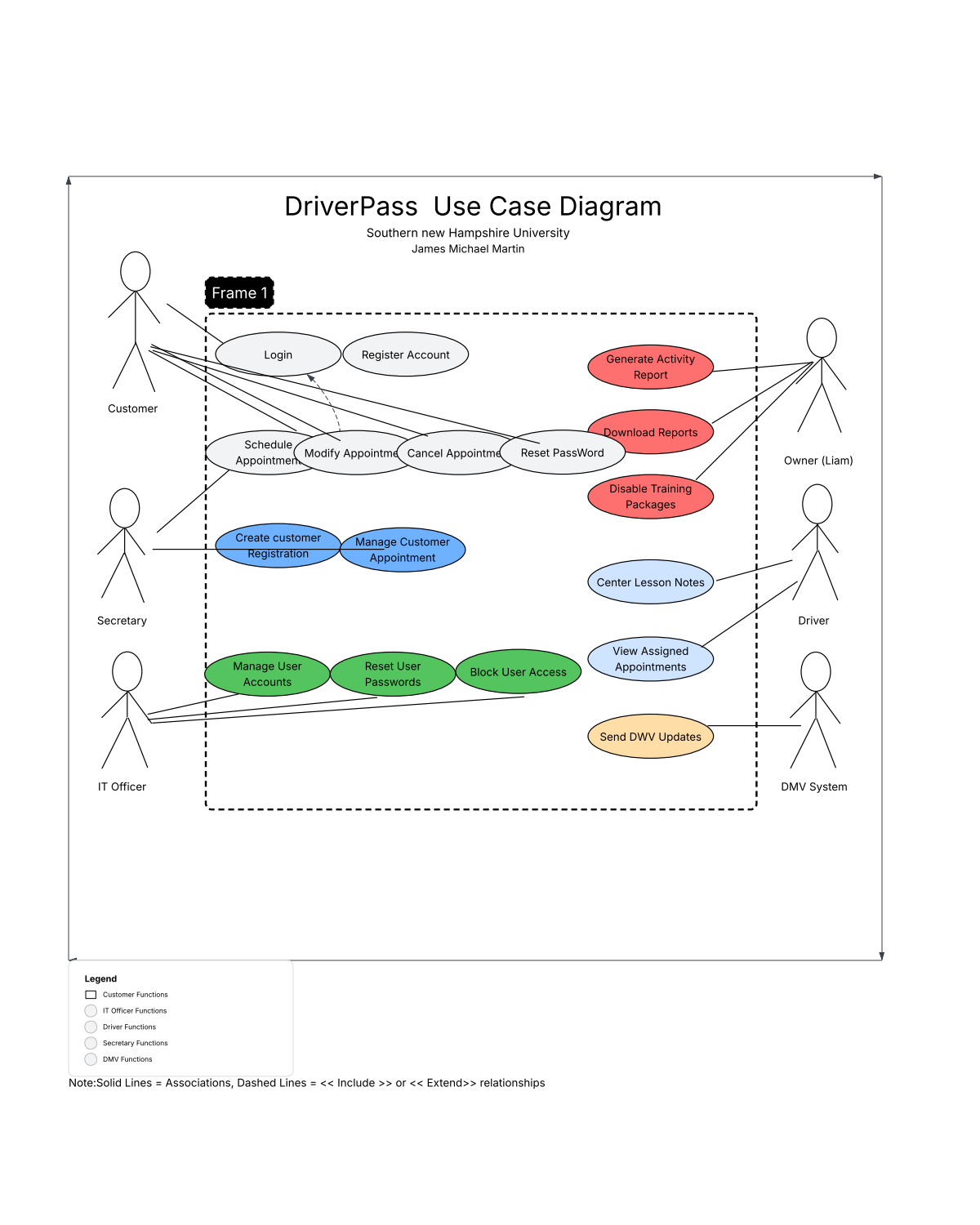
# CS 255 System Design Document Template

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 Use Case Diagram

**Diagram Description:** This diagram illustrates all the major functionalities of the DriverPass system from the perspective of its different users. It shows how customers can schedule lessons and take tests, how the secretary can manage appointments, how the IT officer can control user access, and how the owner can access business reports, ensuring all user needs are accounted for.

**

### UML Activity Diagrams

*[You were asked to choose* ***two*** *use cases and create* ***two*** *activity diagrams, one for each use case. Please insert* ***both*** *of your activity diagrams here. Check to make sure that you included appropriate components and symbols and that your design meets the client’s needs.]*

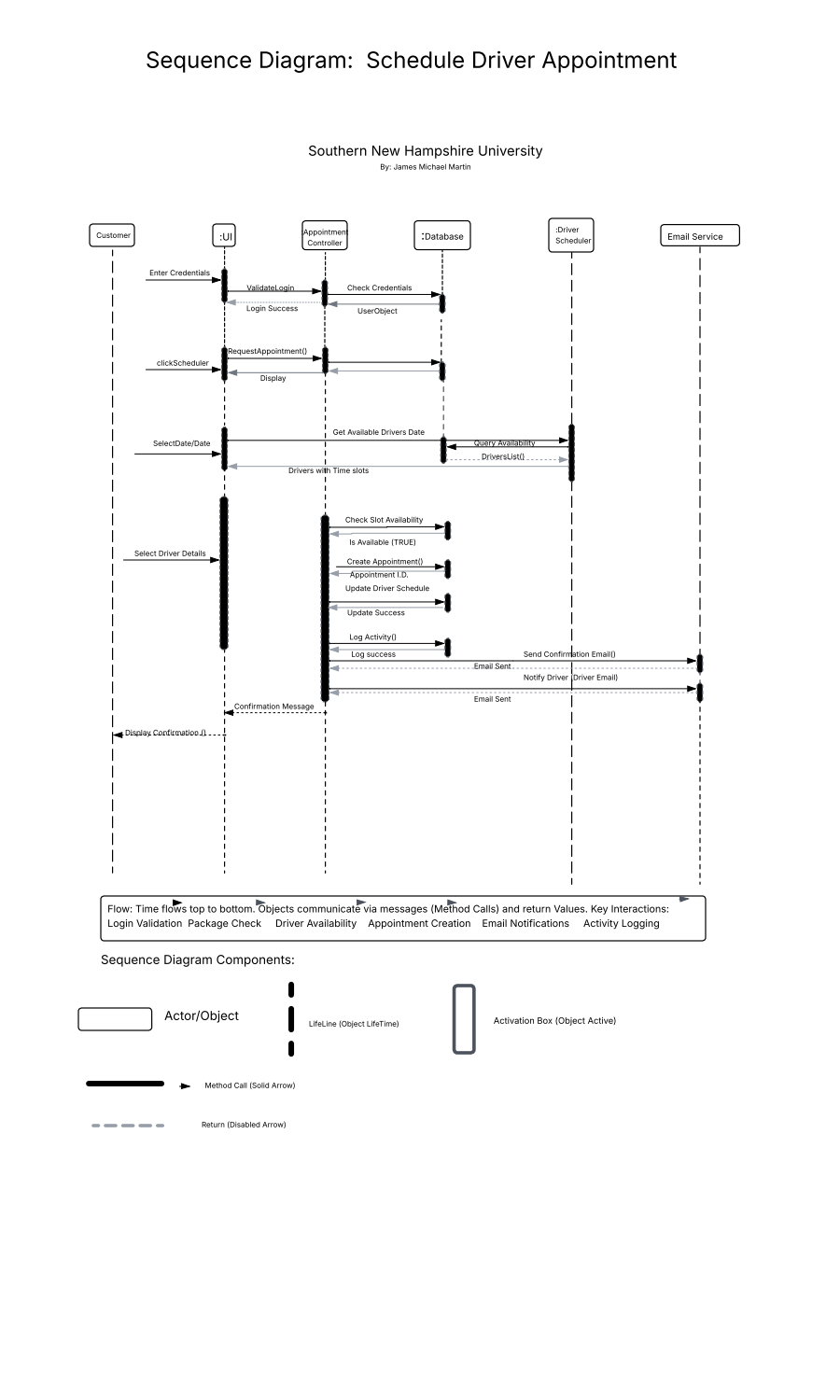
### UML Sequence Diagram

****Activity Diagram 1: Schedule a Driving Lesson****  
This diagram visualizes the process flow for a customer (or secretary) scheduling a driving lesson.

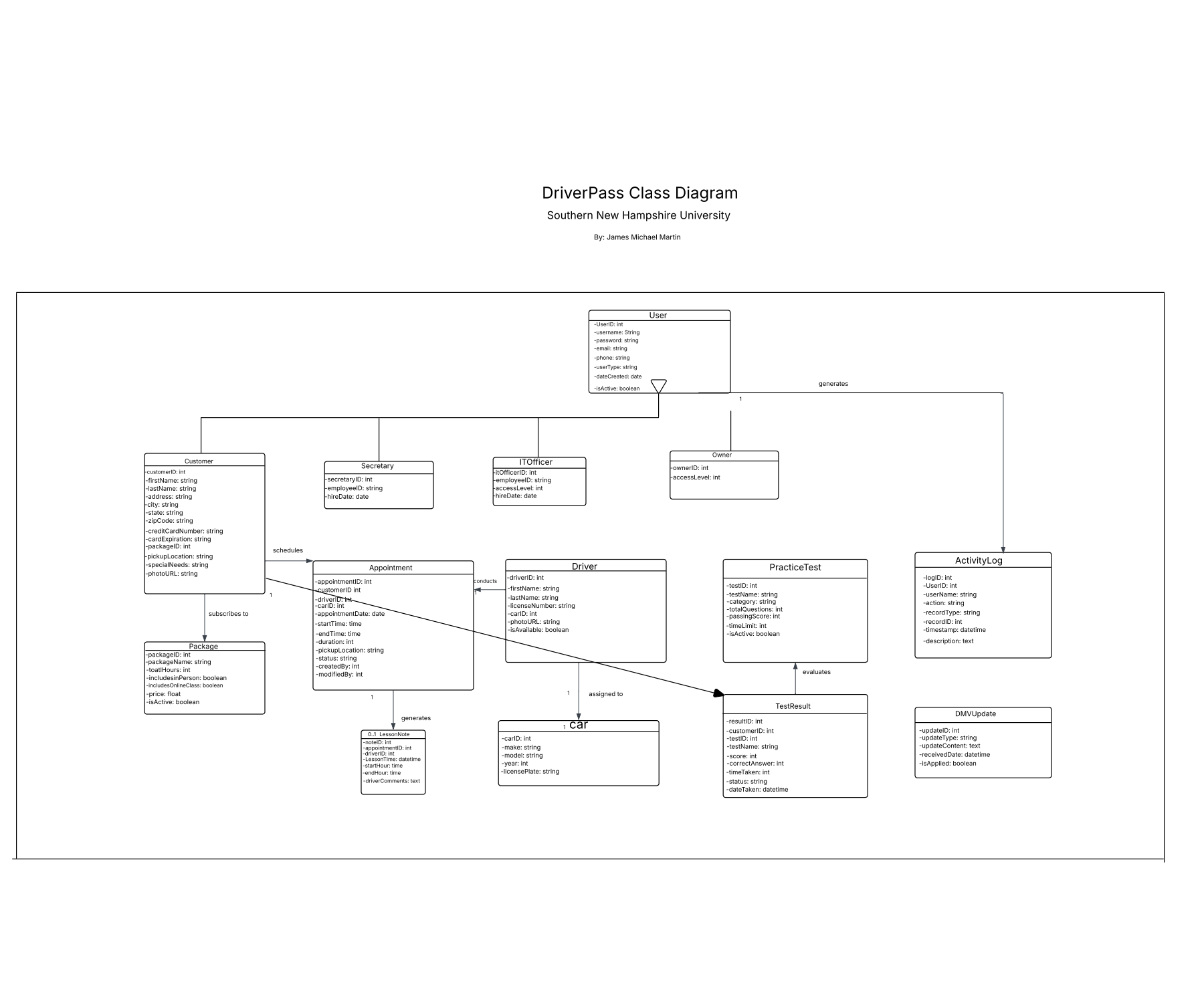
* ****Description:**** The process begins when a user logs in. The system checks their package to ensure they have remaining hours. The user then selects a date and time. The system checks driver and vehicle availability. If available, the appointment is booked, and the user receives a confirmation. If not, the user is prompted to choose a different time.

****Activity Diagram 2: Take a Practice Test****  
This diagram visualizes the process flow for a customer completing an online practice test.

* ****Description:**** The customer selects a practice test to take. The system presents questions one by one. The customer answers each question, and the system records the response. After completing all questions, the system calculates the score, updates the customer's progress status (e.g., "Passed" or "Failed"), and saves the results, including the time taken.

**

### UML Class Diagram

**

****Sequence Diagram: Schedule a Driving Lesson****  
This diagram details the object interactions required to execute the "Schedule a Driving Lesson" use case.

* ****Description:**** The sequence starts with the Customer actor clicking "Schedule" on the User Interface. The interface sends a checkAvailability(request) message to the Scheduler object. The Scheduler then queries the Database to getDriverAndVehicleAvailability(request). Based on the response, the Scheduler confirms availability and the User Interface prompts for confirmation. Upon confirmation, the User Interface tells the Appointment object to createNewAppointment(details), which finally saves the data to the Database. This sequence ensures a clear, traceable process for creating an appointment.

****UML Class Diagram****  
This diagram defines the static structure of the system, showing the main classes, their attributes, and relationships.

* ****Description:**** The core classes include:
  + User (with attributes like userID, password, firstName, lastName), which is a generalization for Customer, Employee, ITOfficer, and Owner.
  + Customer (with attributes like address, packageType, remainingHours).
  + Appointment (with attributes like appointmentID, date, time, duration).
  + Vehicle and Driver.
  + Test (with attributes like testID, title, maxScore).
  + TestResult (with attributes like score, status, timeTaken).  
    The relationships show that a Customer can have multiple Appointments and TestResults, and an Appointment is associated with one Driver and one Vehicle.

****Technical Requirements****  
Based on the functional and nonfunctional requirements from the BRD, the following technical specifications are proposed:

* ****Hardware & Infrastructure:****
  + The system will be deployed on a ****cloud infrastructure**** (e.g., AWS, Azure, or Google Cloud) to eliminate the need for on-premise servers and leverage the provider's built-in backup and security services.
  + No specific client hardware is required beyond devices capable of running a modern web browser.
* ****Software & Tools:****
  + ****Front-end:**** A responsive web application built using HTML5, CSS3, and JavaScript to ensure compatibility across desktops, tablets, and mobile devices.
  + ****Back-end:**** A server-side technology stack (e.g., Node.js, .NET Core, or Python/Django) to handle business logic.
  + ****Database:**** A relational database management system (RDBMS) such as PostgreSQL or MySQL to store user data, appointments, test results, and driver notes.
  + ****Security:**** Implementation of HTTPS/SSL for all data in transit. Database encryption and PCI-DSS compliant payment gateways for storing/processing credit card data.
* ****Integration:****
  + Development of a secure API endpoint to receive update notifications from the DMV systems regarding rule and test question changes.