**Course Registraation System:**

**UML Design Modeling & Testing Levels**

Angelica Topete

University of Arizona Global Campus

CST499: Capstone for Computer Software Technology

Joseph Rangitsch

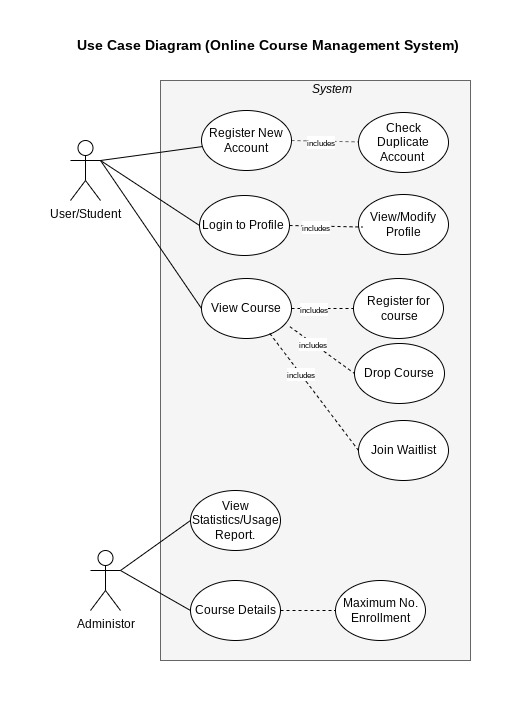
July 3, 2023

**UML Design Modeling**

The UML models below depict the proposed course registration system. These models give us a clear picture of how the system works and help meet its specific requirements. This section explores several types of UML diagrams, such as user, class, sequence, state, and deployment diagrams. Each diagram serves a unique purpose and offers insights into the developing system's requirements.

**Figure 1**

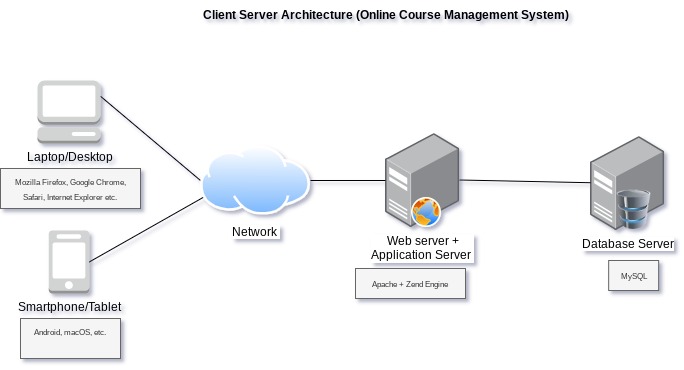
*Use Case Diagram*



Note. The use case diagram for the course registration system serves as a visual representation of the system's various roles and interactions. This diagram provides a clear overview of the distinct roles involved and illustrates how these roles interact with the system.

**Figure 2**

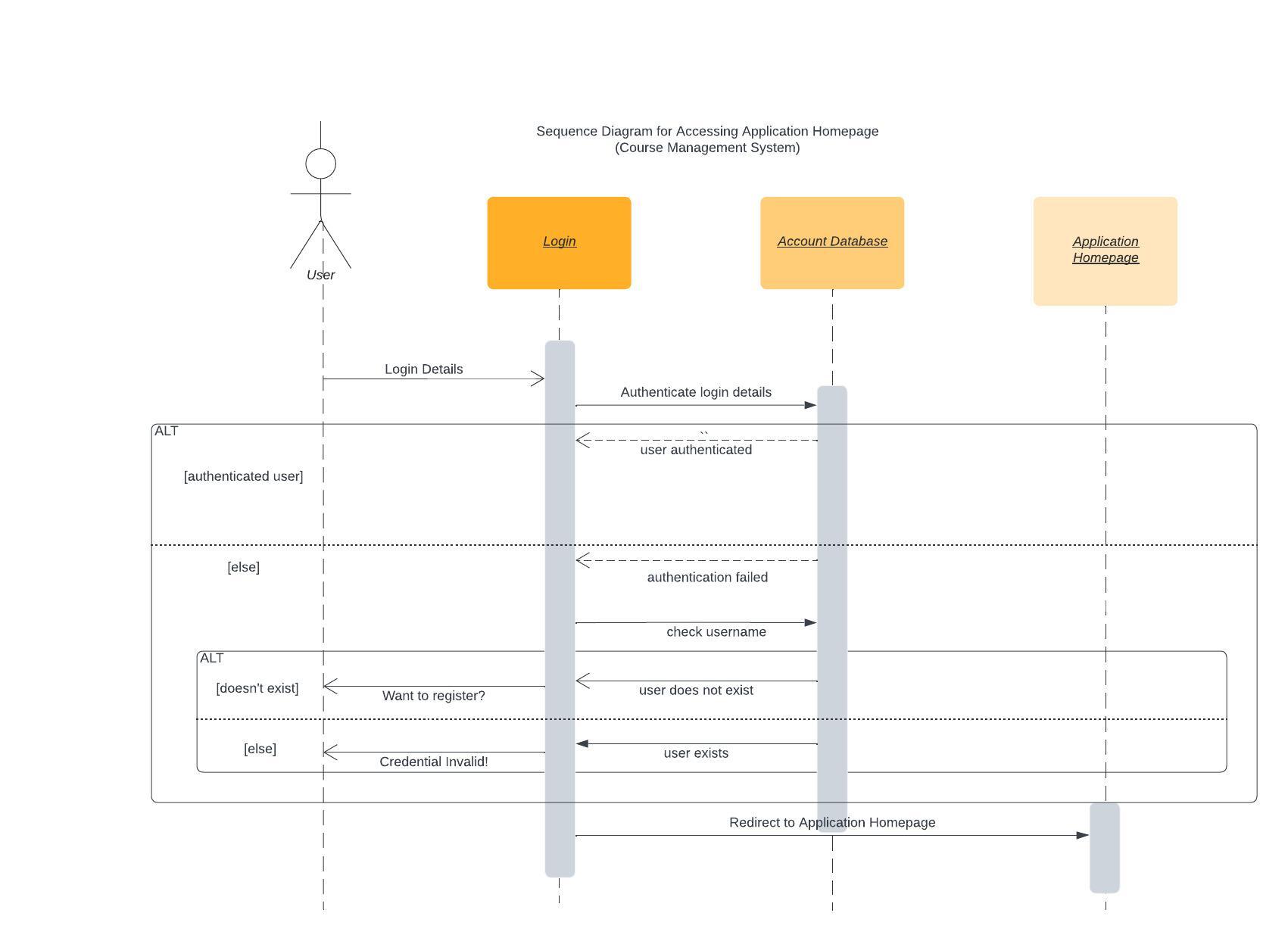
*Deployment Diagram*



Note. The deployment diagram for the course management system shows the system's architecture, illustrating the hardware and software components and their environment. This diagram visually represents how the website is deployed and demonstrates the connections and relationships between the elements.

**Figure 3**

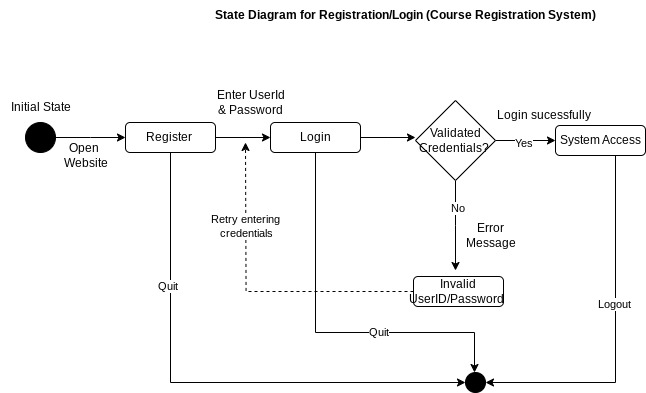
*Sequence Diagram*



Note. The sequence diagram for the course registration system provides a visual representation of how objects interact within a specific use case. In this case, the deployment diagram models the flow of interaction between the system when a user accesses certain functions.

**Figure 4**

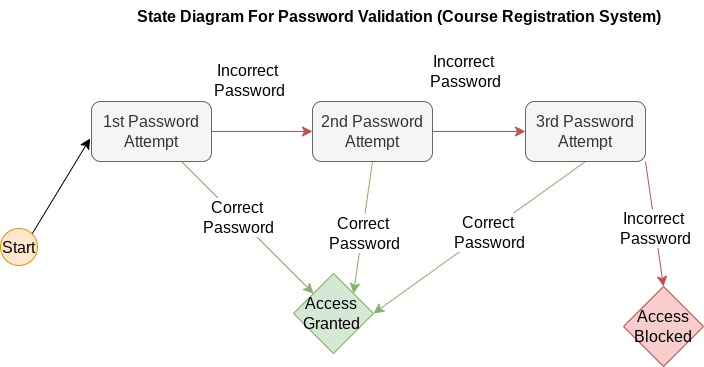
*State Diagram*



Note. The state diagram for the course registration system is designed to illustrate the different states that the system can be in. In this particular diagram, we focus on the control flow within the system, specifically based on the user's input.

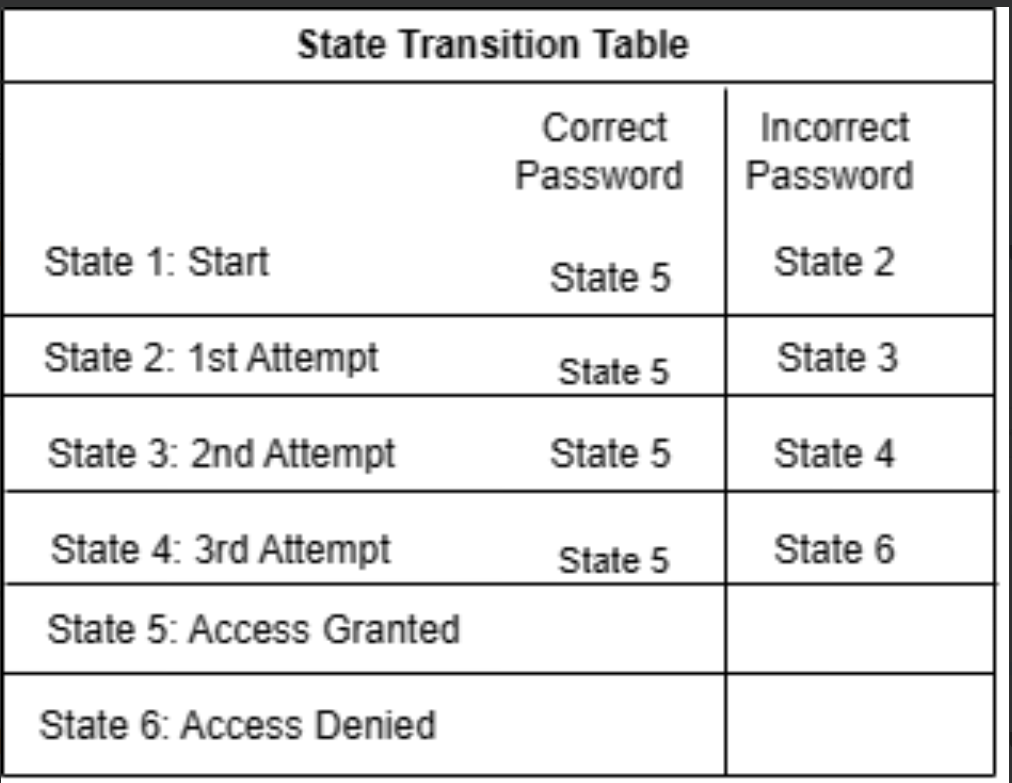
**Figure 5 (A)**

State Diagram for Password Verification



Note. The state diagram for the system's password verification feature illustrates a security feature that blacks account access after the 3rd consecutive failed password attempt. Access to the system is granted if the correct password is entered before the third attempt.

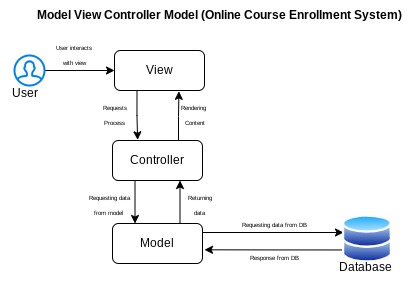
**Figure 5 (B) State Transition Table**



Note. The state transition table shows the transition states for the password verification system.

**Figure 6**

*Model Controller View (MCV) Model*



Note. The Model Controller View (MCV) model divides the application into three interconnected components the model, the view, and the controller.

**Figure 7**  
*Class Diagram*

A diagram of a course management system

Description automatically generated

Note. The class diagram provides a comprehensive representation of the structure of the course registration system. It visualizes the different classes within the system, including their attributes, methods, and relationships.

**Testing Levels**

It is essential to validate that the requirements and specifications are accurate and appropriate for the software being developed (Spillner, Linz & Schaefer, 2014). The V-model development process is the most suitable model for developing the course registration system since it integrates design and test phases. This section identifies several different testing activities for the development activities, including component tests, integration tests, system tests, and acceptance tests.

*Component Tests*

The first level of testing is component tests. This type of testing verifies whether each software component fulfills the specification by "checking that the entire functionality of the test object works correctly and completely as required by its specification" (Spillner, Linz & Schaefer, 2014, P. 46). An example of a test case for component testing in the course registration website, is validating the input/output behavior of the user when accessing the website. Another example would be testing the specific behavior of the website when a user requests certain data, such as logging in to the system or enrolling in a course. Testing and ensuring single components function appropriately guarantees the system can function properly.

*Integration Testing*

The second level of testing is integration testing. This type of testing involves testing a larger group of structural units and subsystems to ensure that all the components collaborate correctly (Spillner, Linz & Schaefer, 2014). To carry out this type of testing, the tester can choose from several different approaches, including big bang integration testing, top-down integration testing, bottom-up integration testing, or mixed integration testing (GeeksforGeeks, 2022a). Big bang integration testing is considered the simplest form since all the modules or components are combined, and the functionality is verified after component testing. Since the course registration website is a small and less complex system, this approach is more favorable than the other approaches.

*System Testing*

The third level of testing is system testing**.** System testing is performed on the completed system. It is used to detect any issues between the combined units. According to GeeksforGeeks (2022c), the testing process is as follows,

1. Create a test environment (in this case, the testing environment would be the actual e-commerce website)
2. Create a test case (an example would be to ensure the user can complete a purchase)
3. Create test data.
4. Execute the test case.
5. Report the defects.
6. Fix the defects.
7. Retest the system.

System testing can be visualized as an iterative approach (if defects are present); however, ensuring the system functions guarantees that the client's requirements are properly met.

*Acceptance Testing*

The fourth and final level of testing is acceptance testing. This type of testing occurs before the software is executed and presented to the client. The environment of acceptance tests is the user's environment. The end-user generally performs acceptance testing (in this case, the student enrolling for a course). Its purpose is to verify whether the developed system meets the users' needs and to see if it is ready to be used in the real world (GeeksforGeeks, 2022b). This type of testing verifies if there are any bugs in the system before the official release.

**References**

GeeksforGeeks. (2022a). Software engineering: Integration testing. GeeksforGeeks. December 20, 2022, from <https://www.geeksforgeeks.org/software-engineering-integration-testing/>

GeeksforGeeks. (2022b). Software engineering: SDLC V-model. GeeksforGeeks. <https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/>

GeeksforGeeks. (2022c). System testing. GeeksforGeeks. <https://www.geeksforgeeks.org/system-testing/>

Spillner, A., Linz, T., & Schaefer, H. (2014). *Software testing foundations: A study guide for the certified tester exam* (4th ed.). Rocky Nook. <https://platform.virdocs.com/r/s/0/doc/567697/>