**Week 2 Assignment: UML Design Modeling**

Shaun Hoadley

CST499: Capstone for Computer Software Technology

Professor Amr Elchouemi

January 23, 2022

**UML Design Modeling**

**Introduction**

It is critical to test software throughout the development lifecycle to ensure a quality product. Testing can save on costs, reduce the development time, improve the quality of code, improve maintainability, locate and correct faults earlier in the process, increase the product robustness, and ensure that the customer receives a quality product that meets their needs and desires when it is performed effectively. According to Spillner et al., there are four levels of testing, component testing, integration testing, system testing, and acceptance testing (2014). In component testing, the software is broken down into functional units and is tested as to whether or not these units meet the required specifications. When groups of related components are tested together to locate faults in the interfaces and interactions between components, this is integration testing (GeeksForGeeks, n.d.). System testing "specifically focuses on testing the functional and non-functional aspects of the software in a more comprehensive manner including security, usability, performance and compatibility" (Suffian et al., 2016). Acceptance testing ensures that the software works in a production-level environment and meets the customer's needs.

**Component Testing**

With regard to the Student Course Enrollment System, the component testing occurs for each functionality illustrated in the use case diagram below. As an example, the software has components carrying out each of the individual functions: displaying the schedule, generating the dropdowns dynamically that allow for course searches, generating the results list based on the form input data, adding courses to the schedule, dropping courses from the schedule, and adding students to the waitlist for a course that is full. All of the functions and classes need to be tested thoroughly, independently from each other. Component-level testing helps to ensure that the code is maintainable, performant, robust, effective, and is free of faults.

**Integration Testing**

In addition to component testing, the Student Course Enrollment System needs to go through integration testing. Integration testing allows developers and testers to "discover faults and bugs in the interaction between integrated components" (Ali et al., 2018). After all of the components are tested and deemed passed on their own, test cases need to be generated to evaluate the components working together. For example, the sequence diagram below illustrates the classes necessary in a sequence of events that allows students to register for a course. While each of these classes might have passed testing at the component level, that does not guarantee whether or not the control flow and data flow will be successful among the interactions between components.

**System Testing**

After integration tests are complete, the next level of testing for the Student Course Enrollment System will be system tests. System tests assess whether or not the application as a whole meets functional and non-functional requirements. For example, does the application contain all of the functionalities listed in the SRS? Is the application fast, reliable, maintainable, secure, and available? During system testing, testers and developers need to view the system from a user perspective and test the system in an environment that is as close to production-level as possible.

**Acceptance Testing**

Lastly, the Student Course Enrollment System must go through acceptance testing. This process will involve the client to ensure that the system provides them with all of the functionality they require for students to rapidly search for, register for, and drop classes. The client also tests that the administrators can quickly and easily generate reports, aid students with their accounts, and add, remove, and update course offerings. Acceptance testing needs to be done using different operating systems and devices to ensure that everyone has a positive user experience.

**Conclusion**

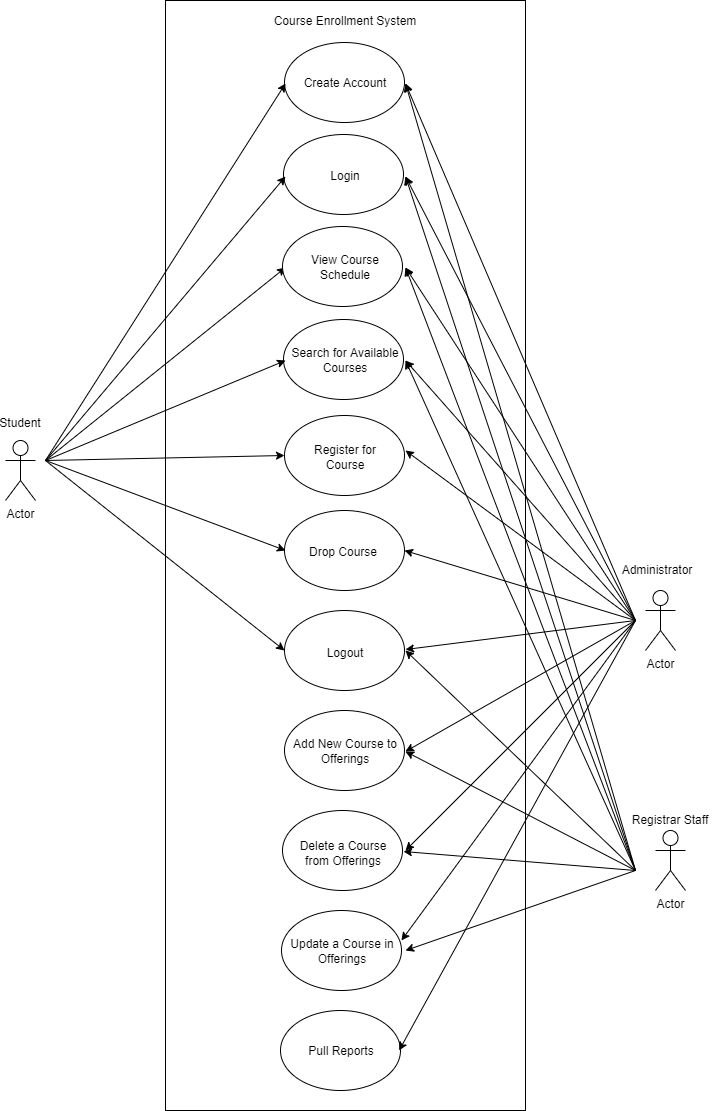
By performing and completing the four levels of testing (component, integration, system, and acceptance), the development team is aided in delivering a high-quality product that meets the needs of the client and does so as quickly and efficiently as it can be done.

**UML Diagrams**

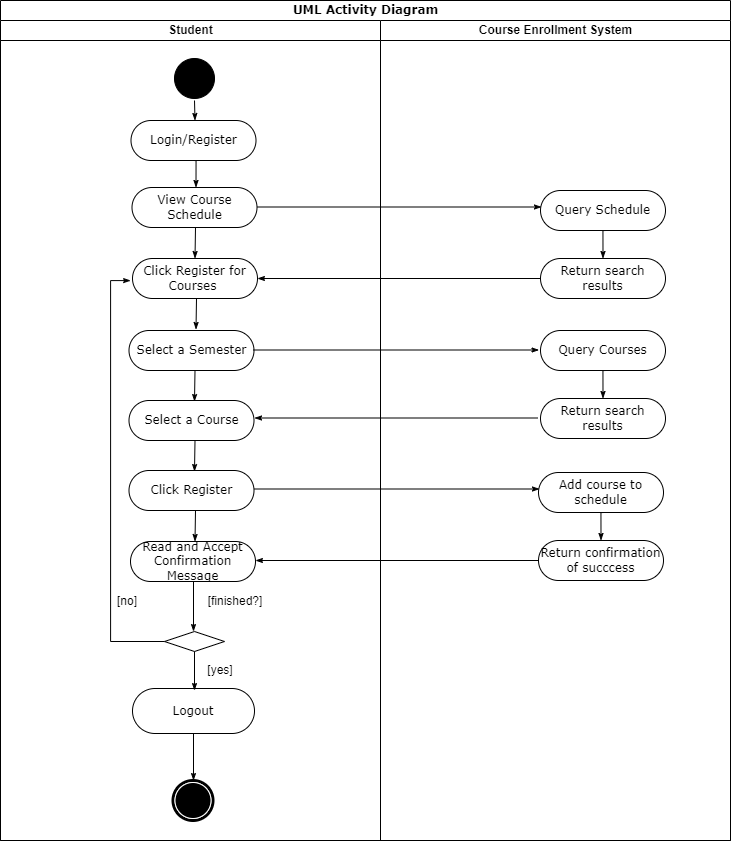
**Graphical user interface, text, application

Description automatically generated**

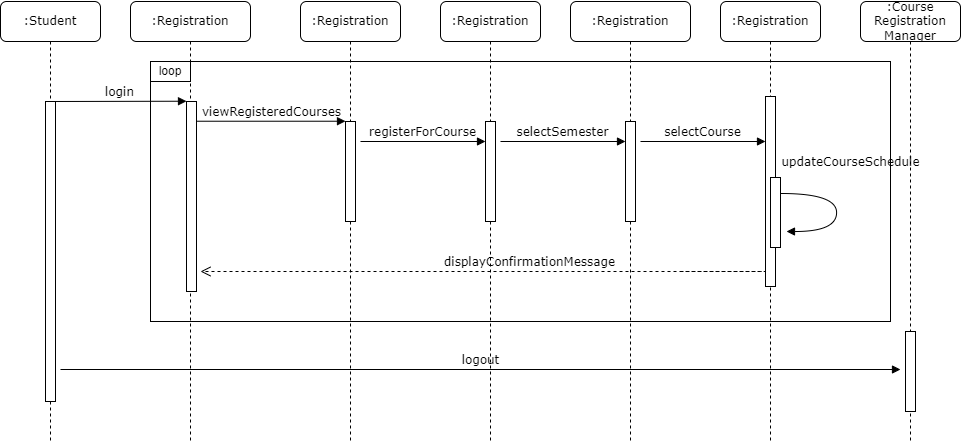
**Figure 1: UML Class Diagram for Student Course Enrollment System**

****

**Figure 2: Use Case Diagram for Student Course Enrollment System**

****

**Figure 3: UML Activity diagram for Student Course Enrollment System**

****

**Figure 4: UML Sequence Diagram for Student Course Enrollment System**

**References**

Ali, S., Imran, M., Hafeez, Y., Abbasi, T.R., Haider, W., & Salam, A. *Improving Component Based Software Integration Testing Using Data Mining Technique*. 2018 12th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS), 2018, pp. 1-6, doi: 10.1109/MACS.2018.8628368.

GeeksForGeeks.com. (n.d.). *Software Engineering: Control Flow Graph (CFG)*. Retrieved from https://www.geeksforgeeks.org/software-engineering-control-flow-graph-cfg/

Spillner, A., Linz, T., & Schaefer, H. (2014). *Software testing foundations: A study guide for the certified tester exam (4th ed.)*. Rocky Nook.

Suffian, M. D. M., Rizal, F. F., Loo, F. A., Aman, N. F., and Bajuri, N. "Software capability rating using system testing scores," *2016 IEEE Conference on Open Systems (ICOS), 2016, pp. 105-110*, doi: 10.1109/ICOS.2016.7881997.