**UML Design Modeling**

Nicholi D’Amato

The University of Arizona Global Campus

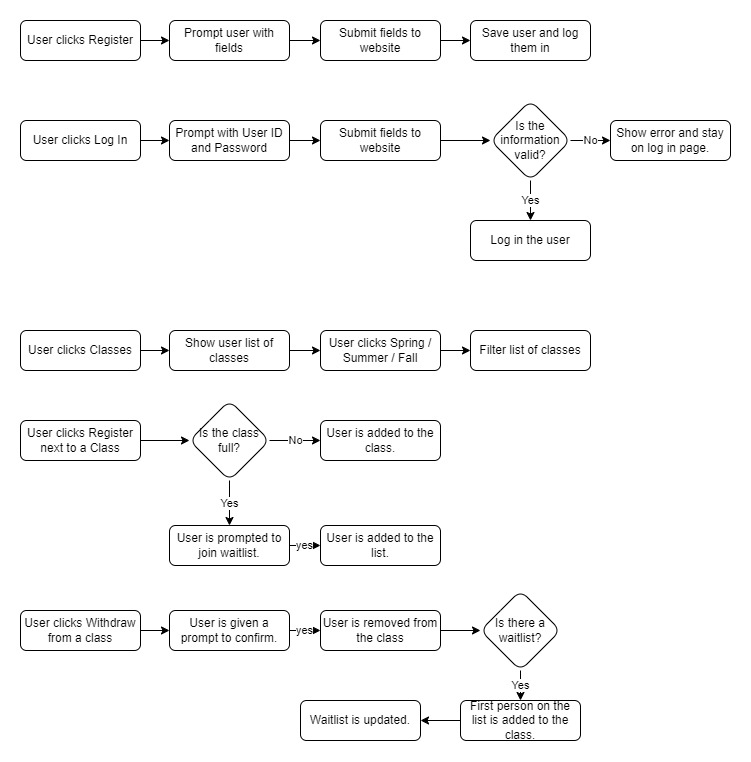
CST 499: Capstone for Computer Software Technology

Instructor: Joseph Rangitsch

April 22, 2024

**A diagram of a website

Description automatically generatedFigure 1: Use Case Diagram**

**Figure 2: Workflow Diagrams**

**A diagram of a class

Description automatically generated**

**Figure 3: Class Diagrams**

Testing is one of the most essential elements of software development. Without proper testing how do we know the software does what it is supposed to do? How do we know that it performs well and meets requirements? Each of the different types of testing such as Component Testing, Integration Testing, System Testing, and Acceptance Testing has their own use, but I believe that Acceptance Testing is the most important testing for software delivery.

For Component Testing we are testing the behavior of a small section of code, but not necessarily a single method or class (Tsui et al., 2018). An easy way to think of component testing would be testing a step, or a few steps of a use case. For example, if I need to register a user with a website, the step of registering the user would be a component, testing the UI of the registration page would be another, and logging in with that user another. Going through the entire use case would generally fall under system testing or integration testing. Sometimes Component Testing can be done with automation in the code itself like Unit Testing, but not always.

Unit Testing is the process of testing small sections of code such as a calculation or saving something to a database. The purpose of these tests is to ensure that as changes to code is made for fixing bugs or adding features, that the essential purpose of that code does not break. The simplest example of this might be a method that adds two numbers together. The test might try 2+2 = 4, and if for any reason that method returns something other than four, then we know it is broken. Unit testing does have a specific meaning, but in common usage often means both the unit testing described here, as well as component testing, or even sometimes integration and system testing. Any automated testing can fall under the term Unit Testing colloquially.

Integration and System Testing covers larger tests than components, instead testing the entirety of the use case. If we continue the example of registering a user with a website, the system testing will cover the entirety of the registration process, from the UI to the back-end performance and behavior. Integration testing is testing the behavior between any two systems. So, if the registration page needs to access a service that stores the registration, that would be covered as an Integration Test (*Integration Testing: What it is, Best Practices & Examples,* n.d.*).* Integration and system testing let us find if there are any issues in how systems, methods and classes are working together that may not be caught by only testing one component.

Acceptance testing is the last stage of testing something before delivery to the customer, regardless of if they are an internal or external customer. Acceptance testing involves working with the customer and ensuring that everything is behaving as it should be and meets the requirements of the project (Collins, 2022). Ideally the project had clear requirements laid out, and the software was tested periodically in the other steps, and the acceptance testing goes well. If the project comes up short in some way, decisions can be made about how to address those problems. If they are small then perhaps a patch can be delivered later, or if they are large then more resources can be assigned to the project, or deadlines pushed. Delivering an untested project to a customer is an easy way to set everyone up for failure.

Ideally, testing will take place throughout the software development process and usually does. Generally, a developer will test their components before pushing code to others to make sure it works. Where things get more difficult is once components and integrations get larger and need to be tested together in Integration and System tests. This step can often be overlooked in small dev teams but can prevent larger issues later. No matter the size of the team, acceptance testing is a must for delivering a completed project. The less of the other testing done, the more likely you are to have issues with acceptance testing, but at least you can catch bugs here. Attempting to deliver a project without any testing, and especially without acceptance testing, will almost certainly result in major problems.

References:

*Integration Testing: What it is, Best Practices & Examples.* (n.d.) TestSigma <https://testsigma.com/guides/integration-testing/>

Collins, Tom. *What is Acceptance Testing? (Importance, Types & Best Practices)*. (December 2022). BrowserStack. <https://www.browserstack.com/guide/acceptance-testing>

Tsui, F., Karam, O., & Bernal, B. (2018). [Essentials of software engineering](https://uagc.instructure.com/courses/130505/modules/items/6659009) (4th ed.). Jones & Bartlett Learning.