**Week 5 Final Project – Final Software Project**

Vicki Kelm

The University of Arizona Global Campus

CST 499: Capstone for Computer Software Technology

Dr. Amjad Alkilani

January 28, 2024

Software Requirements Specification

for

Student Enrollment System

Version 1.0 approved

Prepared by Vicki Kelm

University of Arizona Global Campus

January 28, 2024

Table of Contents

Table of Contents iii-iv

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1-2

2. Overall Description 2

2.1 Product Perspective 2-3

2.2 Product Functions 3

2.3 User Classes and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 3

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3. External Interface Requirements 4

3.1 User Interfaces 4

3.2 API Interfaces 4

3.2 Hardware Interfaces 4

3.2 Software Interfaces 5

3.2 Communication Interfaces 5

4. System Features 5

4.1 Administration Features 5

4.2 Account Features 5

4.3 Application Services and Technical Support…….…………………………………………………5

4.4 Help Desk Support………………………………………………………………………………….5

4.5 System hardware fail over and routine back up 6

5. Other Nonfunctional Requirements 6

5.1 Performance Requirements 6

5.2 Safety Requirements 6

5.3 Security Requirements 6

5.4 Other Non-functional Requirements 6

6. Functional Requirements 6

6.1 Student Self-Service Functionalities 6

6.2 Registration Functionalities 6-7

7. Test Management Strategy 5

7.1 Test Teams 7

7.2 Test Roles 7-8

7.3 Exit Criteria 8-9

7.4 Test Estimated Effort 9

7.5 Test and Risk 9-10

7.6 Incident Reporting 10

7.7 Defect Classification 10-11

7.8 Configuration Management 11-12

8. Test Tools 12

8.1 Introduction 12-14

8.2 Test Management and Control Tools 14

8.3 Test Specification Tools 14-15

8.4 Static Testing Tools 15

8.5 Dynamic Testing Tools 15-16

8.6 Non-Functional Testing Tools 16

Appendix A: Glossary 17

Appendix B: Analysis Models 18-22

Appendix C: Landing, Login, and Enrollment Pages 23-24

Appendix D: MySQL Database and Class Registration 25-26

Appendix E: PHP Codes 26-32

Appendix F: To Be Determined List 32

References 33-34

# Introduction

## Purpose

The purpose of this document is to outline the software requirements for the development of a user registration and online course enrollment system. The system will allow users to register, create accounts and profiles, and enroll in courses. Additionally, it will manage course enrollment limits, waiting lists, and user interactions.

## Document Conventions

The SRS document is divided into 5 sections that include the headers for Introduction, Overall Description, External Interface Requirements, System Features, and Other Non-Functional Requirements. Each header contains sub-headers that provide specific details pertaining to each section of the development process.

## Intended Audience and Reading Suggestions

This document is to provide a clear and complete description of the functionality, features, design, limitations, and goals of the system. It is intended for the following audiences:

* The client, who is the university that requested the system.
* The developers, who will design, implement, and test the system.
* The testers, who will verify that the system meets the requirements
* The users, who are the students and instructors who will use the system.

## Product Scope

This system will cover the entire process of user registration, account and profile creation, course enrollment, and waiting list management. It will facilitate the interaction between users and the available courses across three semesters in a year.

## References

Bose, S. (2022, June 2). *What to consider when evaluating a Test Automation Tool: Checklist*. <https://www.browserstack.com/guide/test-automation-tool-evaluation-checklist>

GeeksforGeeks. (2020, June 04). *Defect Severity*. <https://www.geeksforgeeks.org/defect-severity/>

GeeksforGeeks. (2023, July 19). *Software Engineering | Black box testing*. <https://www.geeksforgeeks.org/software-engineering-black-box-testing/>

Mikoluk, K. (2013, September 18). [XAMPP tutorial: How to use XAMPP to run your own web serverLinks to an external site.](https://blog.udemy.com/xampp-tutorial). Udemy. https://blog.udemy.com/xampp-tutorial

NYU. (n.d.) *System Requirement Specifications (SRS)*. <https://cs.nyu.edu/~jcf/classes/CSCI-GA.2440-001/handouts/Assignment1SampleSolution.pdf>

Software Testing Help. (2023, June 22). *How To Choose The Best Automation Testing Tool (A Complete Guide)*. <https://www.softwaretestinghelp.com/automation-testing-tutorial-4/>

Spillner, A., Linz, T., & Schaefer, H. (2014). *Software Testing Foundations: A Study Guide for the Certified Tester Exam (4th ed.)*. <https://ashford.redshelf.com/app/ecom/shelf/course-section/6094369>

Titov, M. (2023, February 02). *How to Write An SRS Document, or Software Requirements Specification*. <https://www.altigee.com/magazine/how-to-write-an-srs-document-or-software-requirements-specification>

Tsui, F., Karam, O., & Bernal, B. (2018). [Essentials of software engineering](https://uagc.instructure.com/courses/126521/modules/items/6439323)(4th ed.). Jones & Bartlett Learning.

Wiegers, K. E. (1999). [*Software requirement specifications for <project>*Links to an external site.](https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc) [Template]. <https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc>

W3schools. (n.d.). PHP MySQL Delete Data. <https://www.w3schools.com/php/php_mysql_delete.asp>

# Overall Description

## Product Perspective

The system is designed to provide a platform for users to register, create accounts and profiles, and enroll in online courses offered during four semesters per year, winter, spring, summer, and fall. It will include features such as unique user IDs, password protection, course listing, enrollment limits, waiting lists, and cancellation procedures.

## Product Functions

1. User Registration and Account Creation:
   1. Unique user IDs and passwords for each user.
   2. Prevention of duplicate user IDs during registration.
2. User Profile:
   1. Name
   2. Email
   3. Password
   4. User Type
   5. Enrolled courses
   6. Other necessary information stored in user profiles.
3. Course Enrollment:
   1. Users can list courses offered during any semester.
   2. Maximum enrollment limits for each house.
4. Waiting List:
   1. Users can add themselves to a waiting list if a course is full.
   2. Automatic notification to the first user in the waiting list when a spot becomes available.
5. Enrollment Cancellation:
   1. Users can cancel their enrollment in a course.
   2. Notification to the first user in the waiting list (if any) for enrollment opportunity.
6. User Authentication: Users can log in using their unique user ID and password

## User Classes and Characteristics

There will be two types of users for the system: customers and admins. Customers will be able to use all the system’s features, while admins will have access to additional features such as user management, system updates, and data management.

## Operating Environment

1. **Hardware Interfaces**:
   1. Operating System: Windows and Mac
   2. Hard Disk: 40 GB RAM: 512 MB
   3. Processor: Pentium®Dual-core CPU
   4. Network Interface Card (NIC)
2. **Software Interfaces**
   1. XAMPP
   2. Notepad++
   3. MySQL Server
3. **Communication Interfaces**: The customer must connect to the Internet in order to access the Website using broadband internet connection via Internet provider.

## Design and Implementation Constraints

* System must run in windows operating system environment.
* System will use MySQL database for all data management tasks.

## User Documentation

* The website will include troubleshooting suggestions for login access, class enrollment, and course selection.
* In addition, there will be a developing FAQ (Frequently Asked Questions) section for the website that will be maintained by the university.
* Documents will be modified by system administrators as needed.

## Assumptions and Dependencies

* The system will be web-based and accessible through a standard browser.
* The system will us a relational database to store the data.
* The system will have a backup and recovery.
* Assumption that end-users of this software have basic computer knowledge.
* System Admins must be diligent in data management to avoid the deletion or modification of any data in the system which could lead to inconsistencies in the database.

# External Interface Requirements

## User Interfaces

The system will provide user interfaces for:

* User login.
* Registration and account creation.
* Profile management.
* Course listing and enrollment.
* Waiting list management.

## API Interfaces

The system may expose APIs for integration with external systems.

The customer must connect to the Internet in order to access the Website using broadband internet connection via Internet provider.

## Hardware Interfaces

The student enrollment system will be compatible with many devices that include, desktop workstations, laptops, tablets, and mobile devices. HTTPS (Hypertext Transfer Protocol Secure) will be utilized for user access in order to encrypt the connection and hide user history, and any other metadata.

## Software Interfaces

The website will offer a connection to the student enrollment system database that houses all user access information, available courses, and enrollment status. Additionally, the website will be compatible with the most popular operating systems such as, Windows, Mac, and Linux.

## Communication Interfaces

Users will have access to communication notifications provided from the website by using their assigned school email address. Example: student enrolls or cancels enrollment of a course, this triggers an automatic notification will be sent to the user via school email address with details of transaction.

# System Features

## Administration Features

System security and access levels are provided in the online system. There are varying levels of system access and functional authority. Each student’s access is limited to his/her own registration records. Only authorized system administrator(s) has access to all student registration records.

## Account Features

Users will create an account by selecting the registration button from the Home page. This account is required and will allow users access to the website.

## Application Services and Technical Support

Programmers and application developers will have access to source code to address bugs or system enhancements as deemed necessary. Network Administrator and Database Admin support is also required to maintain a 24x7 system uptime.

## Help Desk Support

System users have a 24x7 access to telephone assistance for questions that are technical in nature, such as, slow or sluggish system response time, incompatible browser features, application errors, system downtime inquiries, account lock-out assistance, etc.

## System hardware fail over and routine back up

Computer operations center will handle system hardware tasks such as data tape back-up, hardware maintenance, fail over, scheduled system patches and maintenance.

# Other Nonfunctional Requirements

## Performance Requirements

1. The system shall handle concurrent user registrations without performance degradation.
2. Response times for user actions (e.g., enrollment, cancellation) shall be within acceptable limits.

## Safety Requirements

1. Personal account information will be protected using encryption for user passwords.
2. The system will follow FERPA and ADA guidelines.
3. Session will disconnect after 5 minutes of no activity.

## Security Requirements

1. User passwords shall be securely stored using industry-standard encryption.
2. The system shall implement measures to prevent unauthorized access to user accounts.

## Other Non-functional Requirements

1. The system shall be user-friendly with clear instructions and error messages.
2. The system shall be scalable to accommodate an increasing number of users and courses.

# Functional Requirements

The online registration system is “self-service style” system that shall initially address the student registration needs.

## Student Self-Service Functionalities

* Create userid and password
* Create profile
* View account

## Registration Functionalities

* View registration status
* Course details
* Student schedule
* Add or drop a course

# Test Management Strategy

## Test Teams

“An independent testing team is beneficial for test quality and comprehensiveness” (Spillner, Linz, & Schaefer, p. 32). A tester is able to view the test object without bias. They will need to possess necessary knowledge regarding the test object to create test cases, but will not have any assumptions or misunderstandings that the developer may develop. A developer should not be one to test their own product due to being blind to their own code. “Because there is tendency to be blind to our own errors, it is much more efficient to let different people perform testing and development and to organize testing independently as possible from development” (Spillner, Linz, & Schaefer, p. 169). For this reason, designated testers on the development team should perform component testing. Additionally, “test personnel will consist of three to five IT and test specialists responsible for integration activities, non-functional tests, test automation, and support of test tools (“technical test”)” Spillner, Linz & Schaefer, p. 171). The ultimate leader of the test team will be the test manager who provides the test planning and test control.

## Test Roles

The following test roles should be assigned to individuals when conducting testing of the Student Course Enrollment System:

* **Test Manager**: Leader, responsible for test planning and test control experts, possessing knowledge and experience in the fields of software testing, quality management, project management, and personnel management
* **Test Designer**: Test analyst, expert in test methods and test specification, has knowledge and experience in the fields of software testing, software engineering, and formal specification methods.
* **Test Automator**: Test automation expert(s) with knowledge of testing basics, programming experience, and deep knowledge of the testing tools and script languages.
* **Test Administrator**: Expert(s) for installing and operating the test environment (system administrator knowledge).
* **Tester**: Expert(s) for executing tests and reporting failures, IT basics, basic knowledge of testing, using test tools, understanding test object.

## Exit Criteria

Exit criteria along with test entry “is an important part of test planning” (Spillner, Linz, & Schaefer, p. 179). Both assist with defining the point at which testing will start and stop. The importance of exit criteria is to prevent the test work is not ended prematurely or by chance. Each test case should contain specific exit criteria to guide the testers for determining when a test is complete. Some examples of test criteria and indicators are listed below:

* **Achieved Test Coverage**: Tests run, covered requirements, code coverage
* **Product Quality**: Defect density, defect severity, failure rate, and reliability of the test object
* **Residual Risk**: Tests not executed, defects not repaired, incomplete coverage of requirements or code
* **Economic Restraints**: Allowed cost, project risks, release deadlines, market chances

Any project-specific test exit criteria will be included in the test plan as defined by the Test Manager. The exit criteria will be measured regularly and evaluated during test execution to serve as the decision basis by test and project management.

## Test Estimated Effort

The Test Manager will provide in the test plan any details pertaining to estimating test effort and test costs to include; re-estimating, and re-planning the testing tasks during later testing work. It is the Test Manager’s responsibility to initiate test effort estimation during the planning phase to ensure that resources are assigned and distributed properly. “Task-driven test effort estimation tends to underestimate the testing effort. Estimating based on the data of experience of similar projects or typical values usually leads to better results” (Spillner, Linz, & Schaefer, p. 184).

## Test and Risk

Test and risk are combined as Risk-Based Testing which uses information on project and product risks and directs testing to areas with high risk. “Risk-based prioritization of the tests ensures that risky product parts are tested more intensively and earlier than parts with lower risk” (Spillner, Linz, & Schaefer, p. 189). This will aid in finding severe problems early on reducing delays or the need for corrective rework. The demand for systemic risk management is included in the standards for quality assurance and test plans that include the following:

* Regularly identifying what can go wrong (risks)
* Prioritizing identified risks
* Implementing actions to mitigate or fight those risks

Testing will decrease any uncertainty about risks, will help to estimate risks, and will ultimately identify new risks.

## Incident Reporting

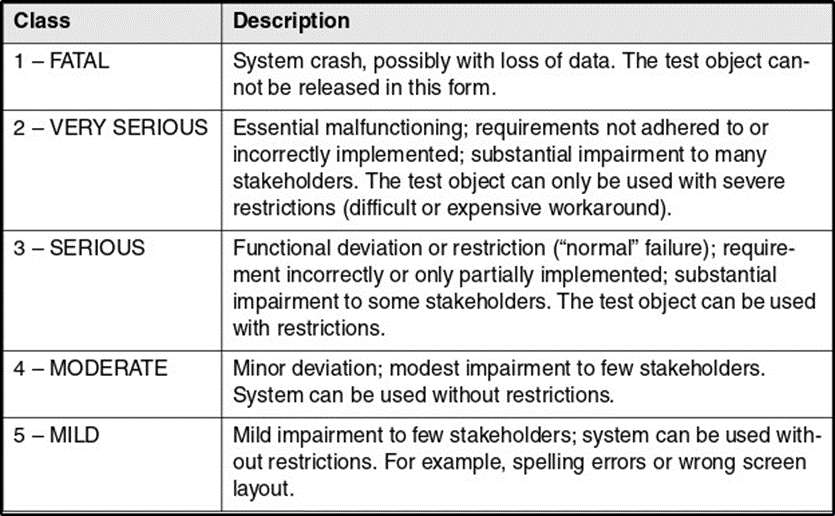
A test incident report is a document/report generated after the culmination of the software testing process. The various incidents and defects are reported and logged by the team members to maintain transparency among the team members and to take essential steps to resolve these issues. An incident report is created for the test object that caused the problem and is done for every unexpected behavior or observed deviation from the expected results found in the test log (Spillner, Linz, & Schaefer, p. 193). The incident report is a formal document that describes any unexpected event, issue, or problem that is encountered during the testing process. A central database is created for each project in order to manage each incident that is discovered. Incidents may be reported by testers, Quality Assurance Engineers, or other stakeholders of the project. The report will include information regarding the tested software, test environment, tester’s name, defect classification, defect prioritization, and any relevant information to reproduce or locate the defect.

## Defect Classification

Defect classification, also known as defect severity, “is the classification of a defect based on its level of destructive impact on the requirements specification of the software” (Defect Severity, 2020). The defects, also known as bugs or faults, are anything that causes threading to the software's value, quality, and aim. The tester can identify it through the variance of expected results and actual results of execution of test cases. The classification includes levels of severity class as well as fault priority. There are five class levels of severity: 1 – FATAL, 2 – VERY SERIOUS, 3 – SERIOUS, 4 – MODERATE, 5 – MILD. There are four levels of priority: 1 - IMMEDIATE, 2 – NEXT RELEASE, 3 – ON OCCASION, 4 – OPEN. The diagrams below provide the levels and details of each as provided in the course text:

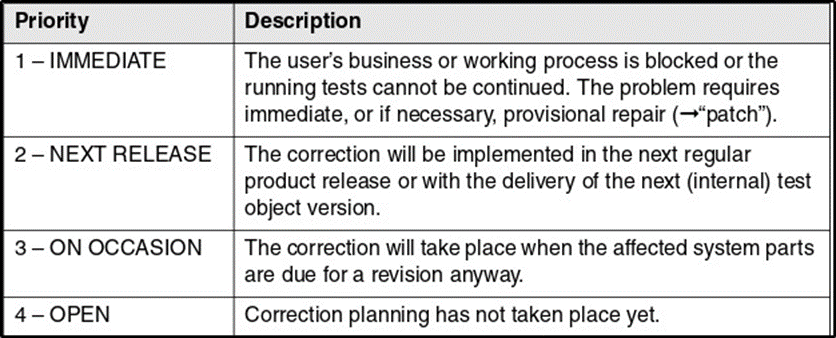
**Figure 1**

Failure Severity Class Table



**Figure 2**

Fault Priority Table



## Configuration Management

In order to have sufficient and successful configuration management, several requirements must be met. These requirements include: Version Management, Configuration Identification, Incident and Change Status Control, and Configuration Audits.

* **Version Management**: Provides cataloguing, filing, and retrieving the different versions of a configuration item, and includes securing comments on the reason for a particular change.
* **Configuration Identification**: The identification and management of all files (configuration objects) in the particular version, comprises a subsystem (configuration) and requires version management as a prerequisite.
* **Incident and Change Status Control**: Documenting of incident reports and change requests and possibilities to reconstruct their application on the configuration objects.
* **Configuration Audits**: Used to check the effectiveness of configuration management, useful to organize configuration audits. Audits offer the possibility to check whether the configuration management documented all software components, whether configurations can be correctly identified, etc.

Tech Target states, “Configuration management heavily depends on policy, process and automation, which must be integrated into the CM tool or platform”. It is important to revisit, audit and test the configuration management policy, and regularly review and update the process and automation elements to ensure that tool and its usage maintain consistency with IT and business goals.

# Test Tools

## Introduction

There are many tools that provide critical support to the different phases of testing. “Test tools are normally used for these purposes: improving test efficiency, enabling tests, and improving test reliability” (Spillner, Linz, & Schaefer, p. 205). The general categories of tools commonly used in various testing phases are:

* **Test planning tools**: Help testers to define the objectives, scope, strategy, and schedule of the testing project.

Examples: Microsoft Project, Jira, Trello, and TestRail

* **Test design tools**: Help testers to create test cases, test scripts, test data, and test automation scripts based on the requirements and specifications of the software being tested.

Examples: TestComplete, Selenium, Appium, and Postman

* **Test execution tools**: Help testers to run the test cases, test scripts, and test automation scripts on the software under test and collect the test results.

Examples: Perfecto LambdaTest, Jenkins, and Bugzilla

* **Test reporting tools**: Help testers to analyze the test results, generate test reports, and communicate the test findings and recommendations to the stakeholders.

Examples: TestRail, Allure, Kibana, and Power BI

Some key characteristics for test tools that allow for objective evaluation include: cost, usability, functionality, and support. The tool should fit within the budget of the organization and provide a good return on investment. “The cost should include not only the licensing fee, but also the operational, maintenance, and support costs” (Bose, 2022). The tool should be easy to use and learn for the testers, and provide a user-friendly interface and documentation. “The tool should also be compatible with the existing tools and technologies used by the organization” (Software Testing Help, 2023). The tool should support the testing needs and objectives of the organization, and provide the required features and capabilities. “The tool should also be reliable, stable, and scalable, and support various types of testing, such as functional, performance, and security (Software Testing Help, 2023). The tool should have a good customer service and technical support, and provide regular updates and bug fixes.

## Test Management and Control Tools

“Test management tools provide mechanisms for easy documentation, prioritization, listing, and maintenance of test cases” (Spillner, Linz, & Schaefer, p. 206). These tools provide the ability to document and evaluate details for if, when, and how often a test case has been executed. Advanced test management tools support requirements-based testing by capturing system requirements and link them to the tests, which tests the corresponding requirements. Some of the leading test management tools include: Tuskr, PractiTest, and Jira.

* Tuskr: Powerful. Aesthetically pleasing and intuitive cloud-based test management tool, allows for easy organization of test cases into projects, suites, and sections.
* PractiTest: End to end test management tool, enables full visibility into testing process, provides broader understanding of testing results, fully customizable and flexible.
* Jira: Performs automatic build, test, and release in a single place, supports many technologies and languages.

## Test Specification Tools

Tools for test specification allow for test cases to be reproducible, the pre- and postconditions as well as test input data and expected results need to be specified. Some tools used for test specification are: QACoverage, TestCaseLab, and Test Collab.

* QACoverage: Test design module to create manual tests cases, monitor completeness of requirements traceability, test case execution.
* TestCaseLab: Test cases, test plans, test runs, integrations, intuitive to use, create unlimited projects and users.
* Test Collab: Test cases categorized/managed in one place, test execution report, custom reports, integrated bug tracker, requirements linked with test cases.

## Static Testing Tools

Before there are executable programs, static analysis can be executed on source code or on specifications before there are executable programs. “Tools for static testing can therefore be helpful to find faults in early phases of the development cycle” (Spiller, Linz, & Schaefer, p. 210). Some tools for static testing include: Checkstyle, Soot, and SourceMeter.

* Checkstyle: Helps to write Java code, automates process of checking Java code, finds class/method design problems, ability to check code layout and formatting issues.
* Soot: Java optimization framework, provides four representations for analyzing and transforming Java bytecode.
* SourceMeter: Built for precise static source code analysis of C/C++, Java, C#, Python, and RPG projects, finds weak spots of a system under development from source code.

## Dynamic Testing Tools

Dynamic testing tools reduce the mechanical work involved in test execution. “These tools send input data to the test object, record its reaction, and document test execution” (Spillner, Linz, & Schaefer, p. 211). Some tools include: Selenium, JUnit, and Daikon.

* Selenium: An open-source test automation framework that supports multiple programming languages and browsers.
* JUnit: A unit testing framework for Java programming language that allows developers to write and run tests.
* Daikon: A dynamic testing/analysis tool that helps to identify and remediate bugs in Java, C, and C++ programs.

## Non-Functional Testing Tools

Tools for nonfunctional tests provide support for load and performance tests. Some tools for non-functional testing are listed below:

* **Load Testing Tools**: Tools such as Apache JMeter, Gatling, and LoadRunner help simulate high user loads and measure system performance under heavy traffic.
* **Security Testing Tools**: Security tools like OWASP ZAP, Burp Suite, and Nessus aid in identifying vulnerabilities and weaknesses in the system’s security.
* **Performance Monitoring Tools**: Performance monitoring tools like AppDynamics, New Relic, and Dynatrace help monitor the performance of applications and infrastructure.
* **Usability Testing Tools**: Usability testing tools like UserTesting, UserZoom, and Optimizely help test the usability of applications and websites.
* **Test Management Tools**: Test management tools like TestRail, Zephyr, and qTest help manage test cases, test plans, and test runs.
* **Compliance Testing Tools**: Compliance testing tools like Compliance Sheriff, Siteimprove, and AChecker help ensure that web content meets accessibility standards.

Appendix A: Glossary

ADA: Americans with Disabilities Act

API: Application Programming Interface

FERPA: Family Educational Rights and Privacy Act

HTTPS: Hypertext Transfer Protocol Secure

SRS: Software Requirements Specifications

Appendix B:Analysis Models

**Figure 3**

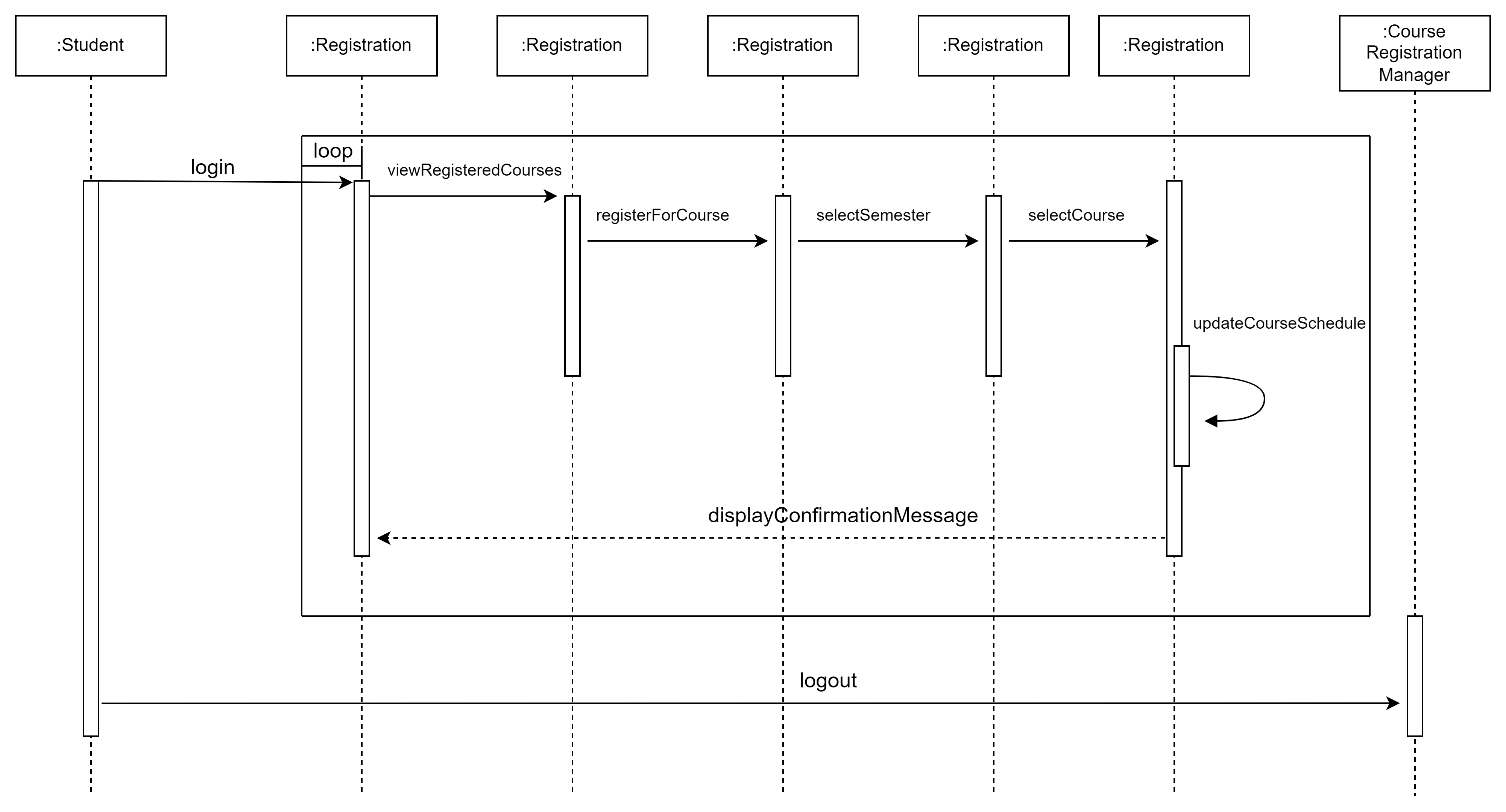
Class Diagram



Note. UML Class Diagram for Student Registration System

**Figure 4**

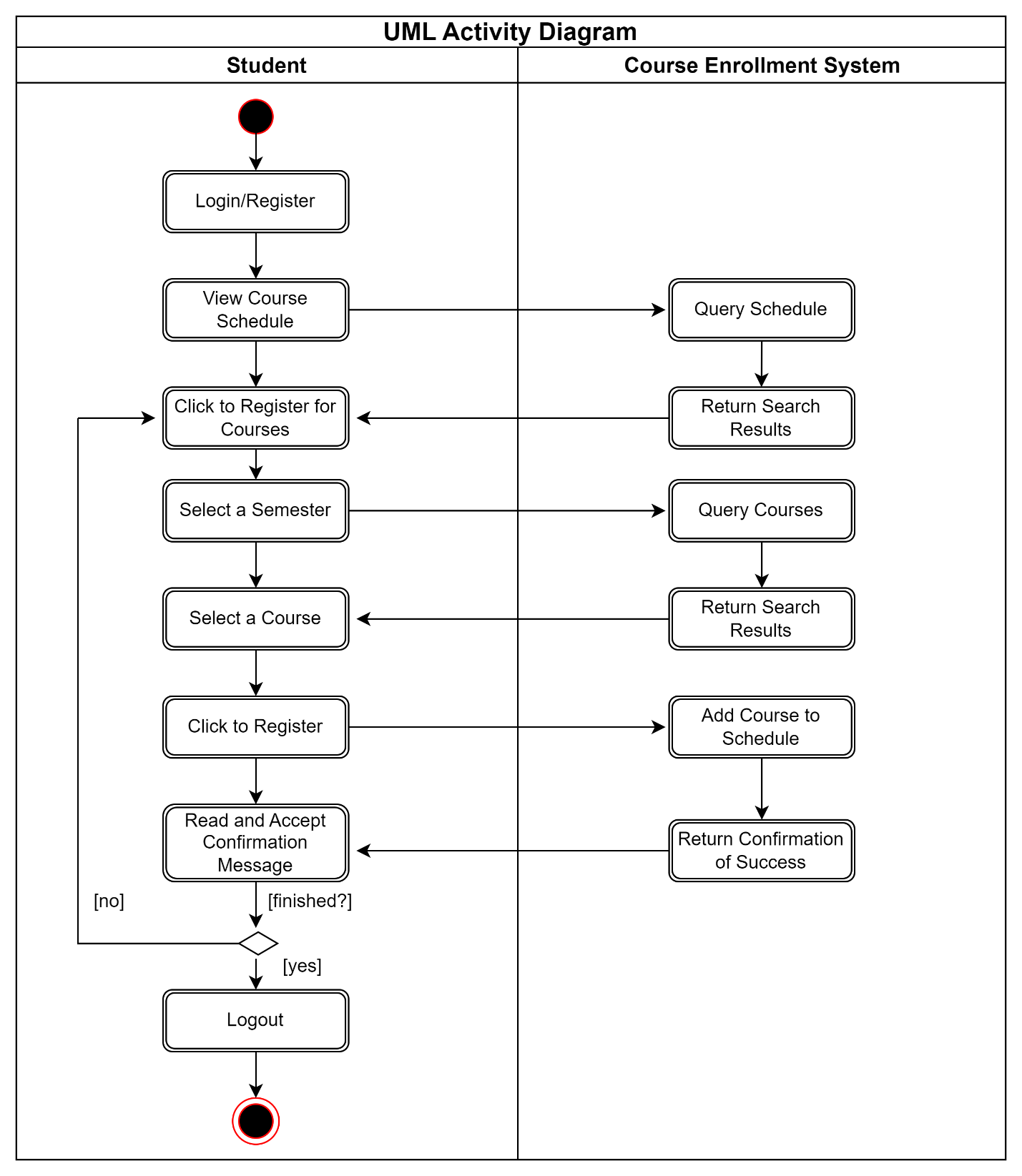
Sequence Diagram



Note. UML Sequence Diagram for Student Registration System

**Figure 5**

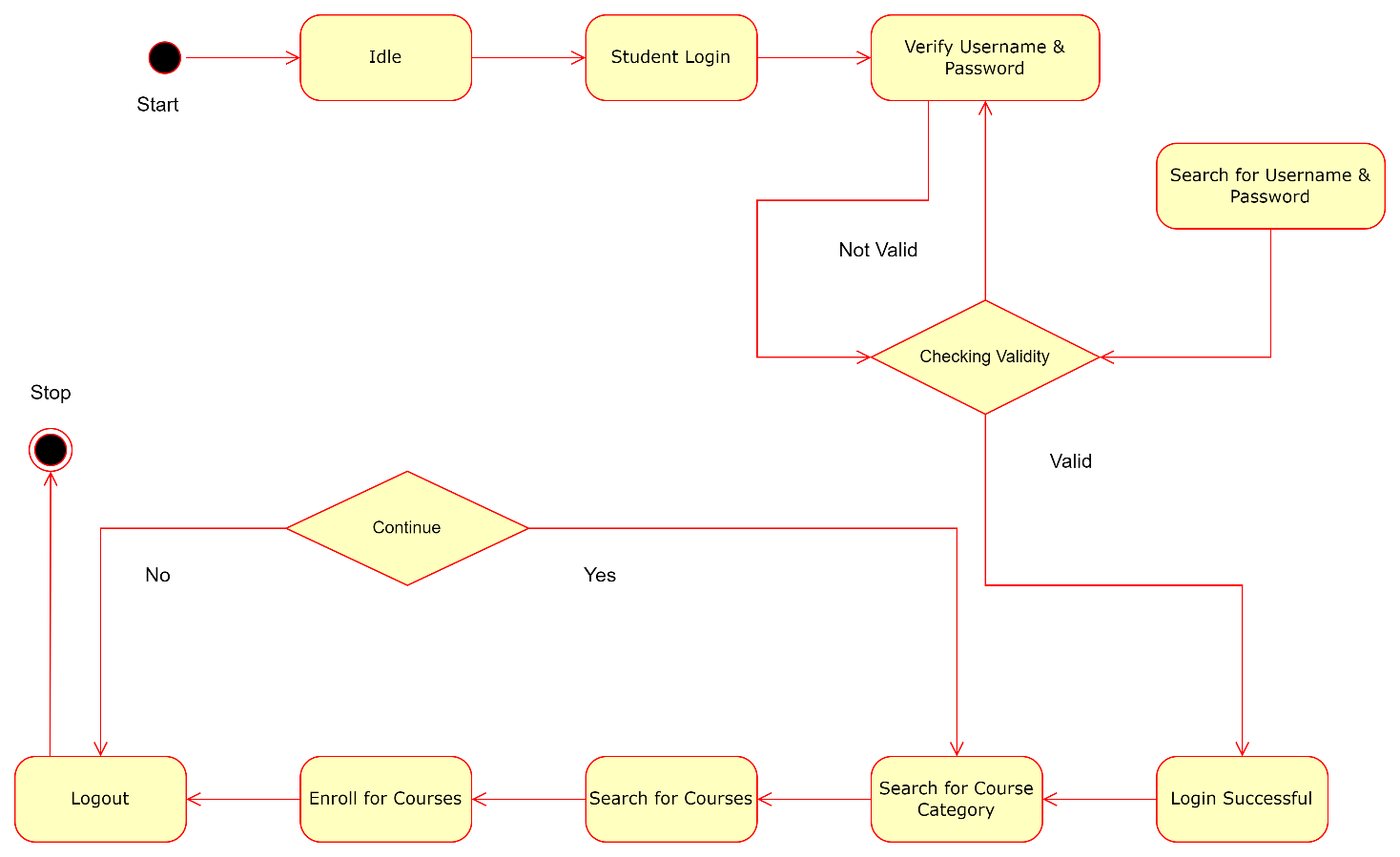
Activity Diagram



Note. UML Activity Diagram for Student Registration System

**Figure 6**

State Diagram



Note. UML State Diagram for Student Registration System

**Figure 7**

Use Case Diagram



Note. UML Use Case Diagram for Student Registration System

Appendix C: Landing, Login, and Enrollment Pages

Figure 8

Landing Page

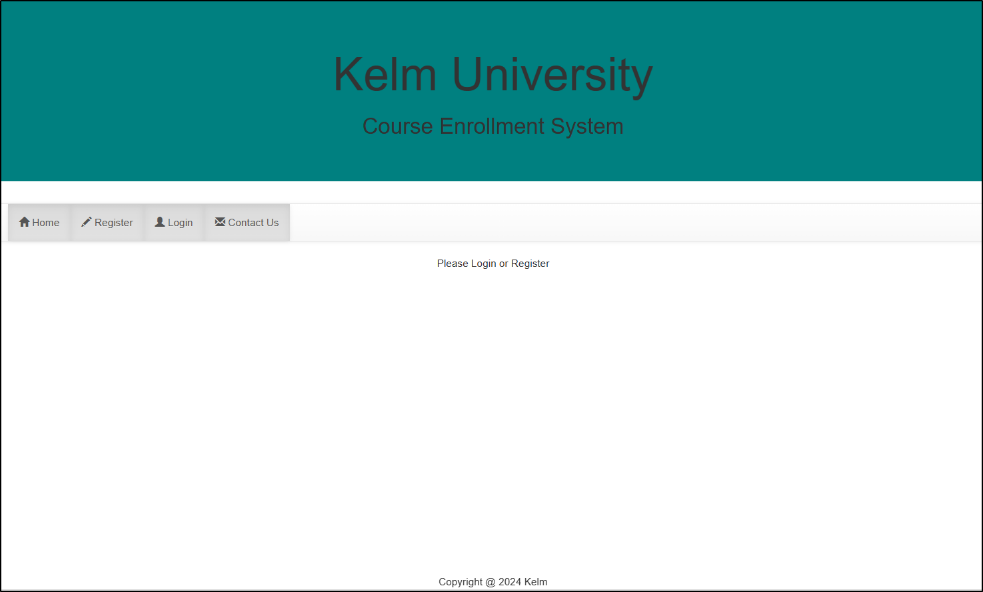


Figure 9

Login Page

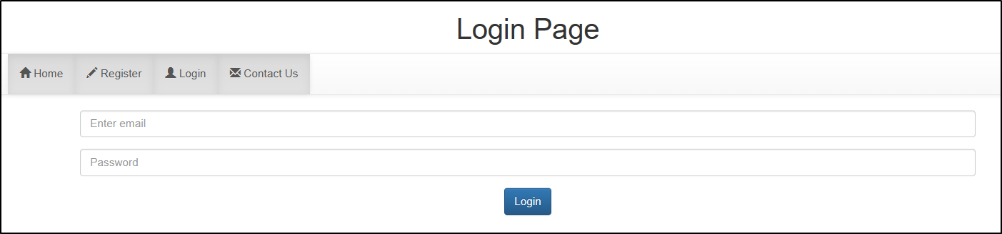


Figure 10

Registration Page

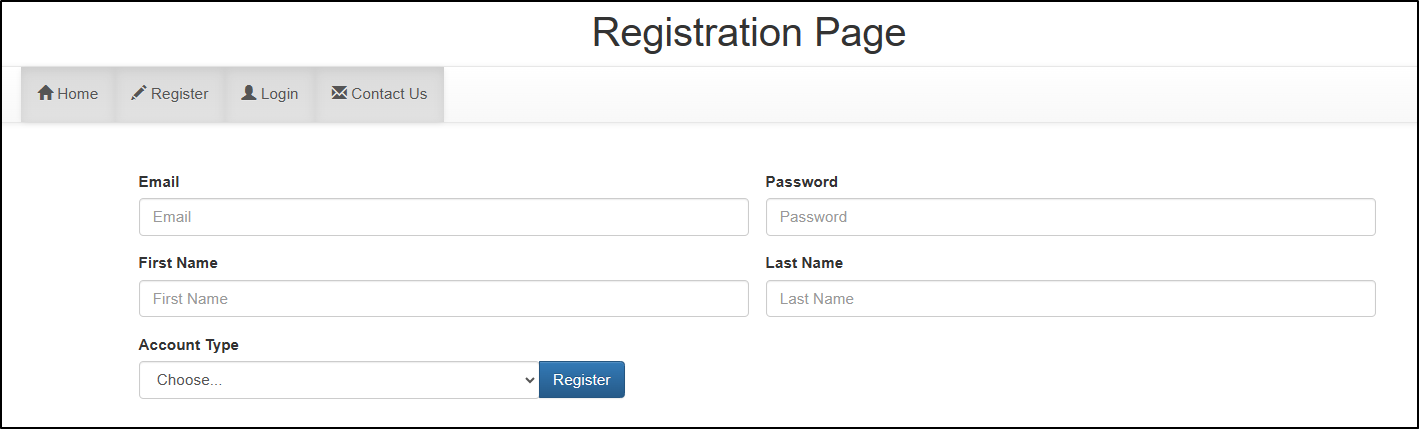


Figure 11

Profile Page – after student enrolls in course(s)

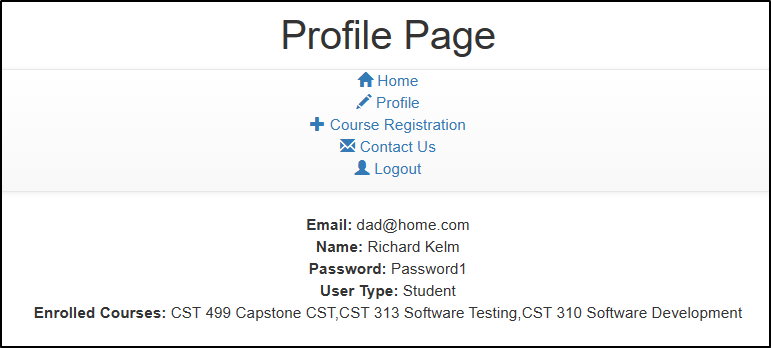


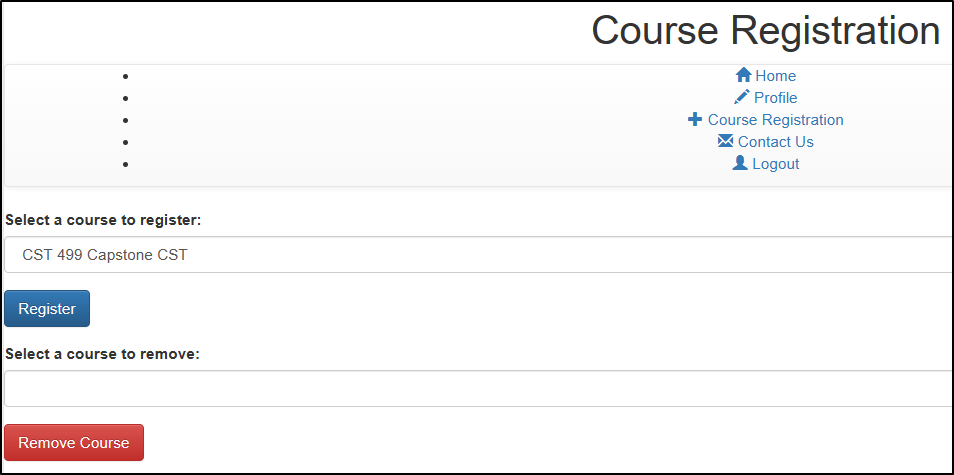
Figure 12

Profile Page – after removing course from student enrollment



Figure 13

Course Registration Page – add/remove courses



Appendix D: MySQL Database and Class Registration

Figure 14

MySQL Database

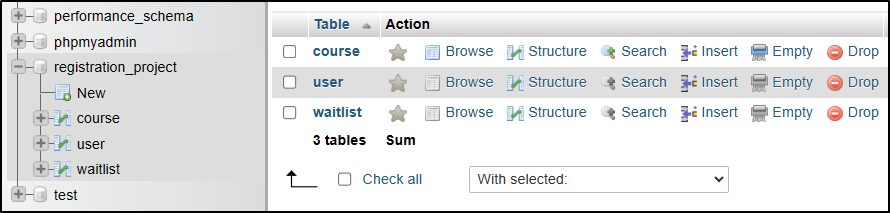


Figure 15

User Table – before courses are enrolled



Figure 16

User Table – after courses are enrolled

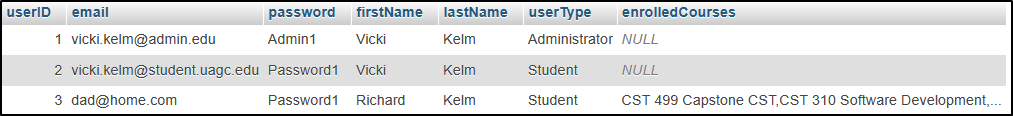


Figure 17

User Table – after removed course: CST 499 Capstone CST

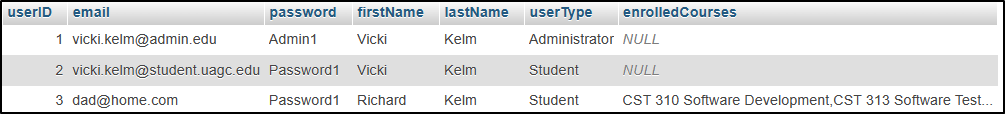


Figure 18

Course Table – before students are enrolled

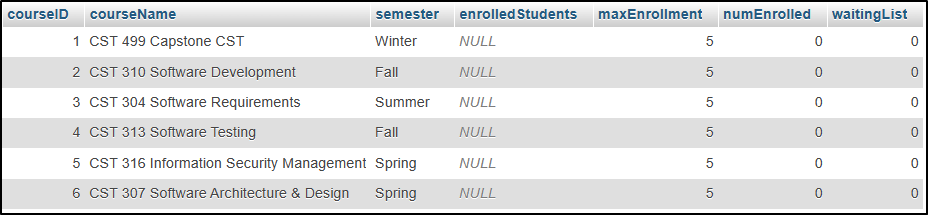


Figure 19

Course table – after students are enrolled

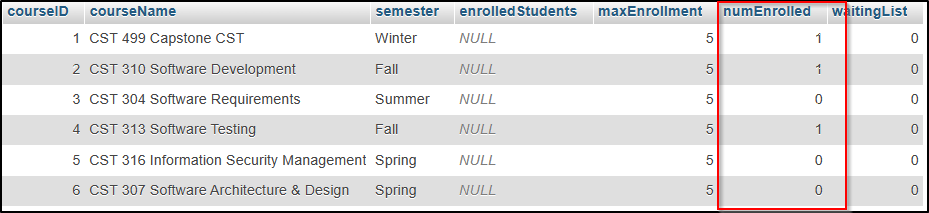
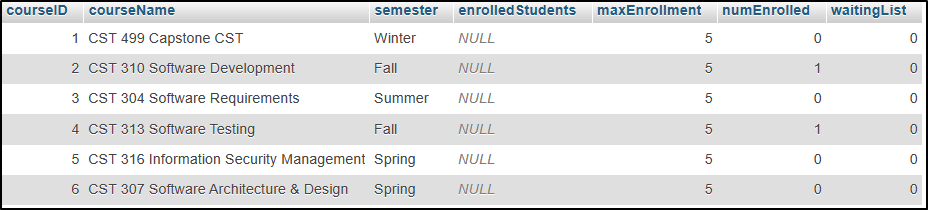


Figure 20

Course table – after course removed from student



Appendix E: PHP Code

Figure 21

Database and table: registration\_project DB, user Table, course Table

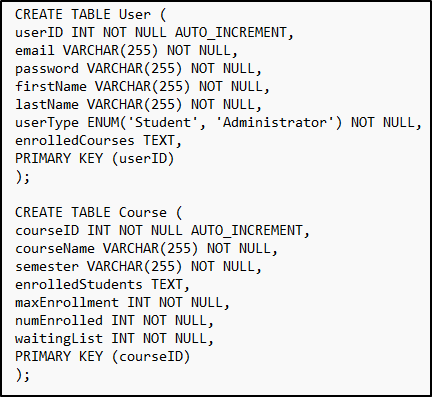


Figure 22

Profile Page PHP Code



Figure 23

Course Registration Page PHP Code: Add or Remove Courses





Figure 24

Connection PHP Code

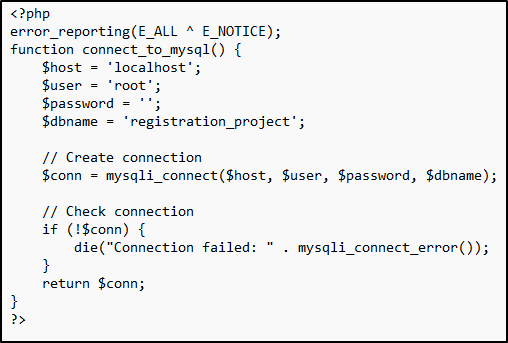


Figure 25

Course Enrollment PHP Code



Figure 26

Landing Page PHP Code

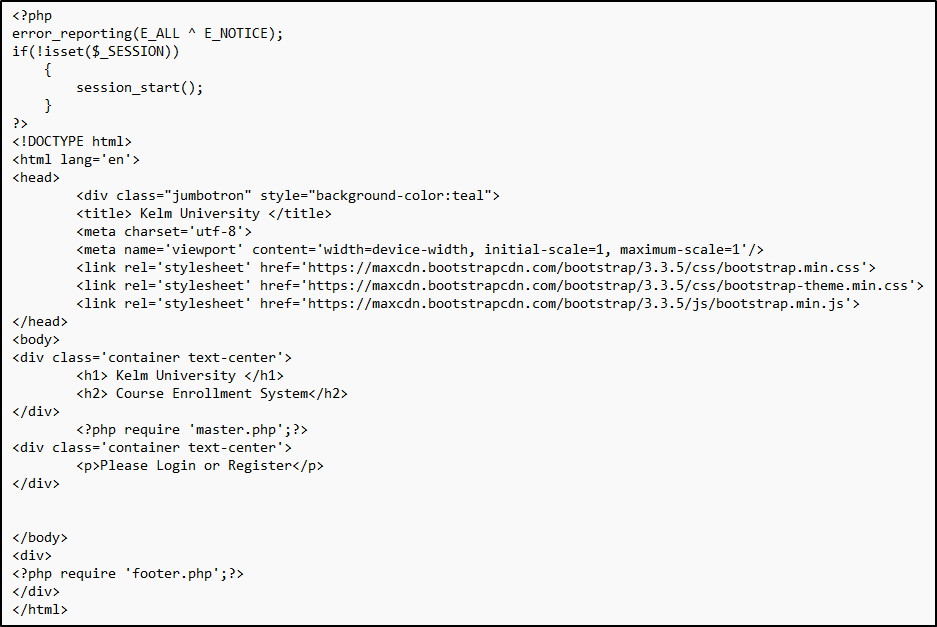


Figure 27

Footer PHP Code

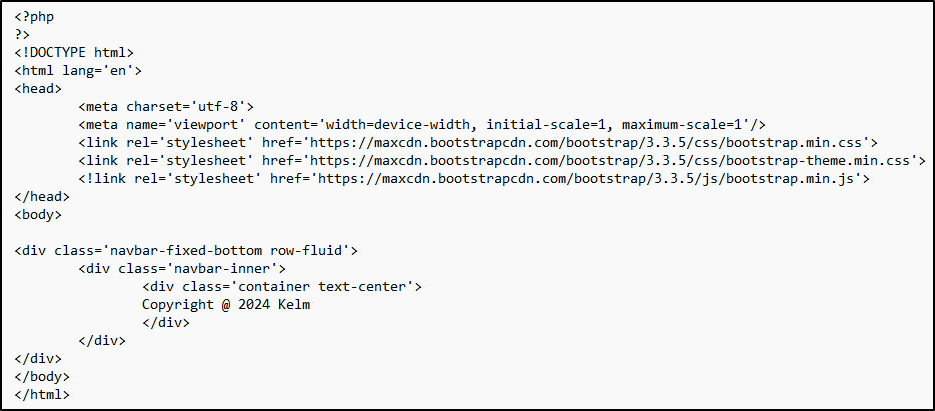


Figure 28

Login Page PHP Code



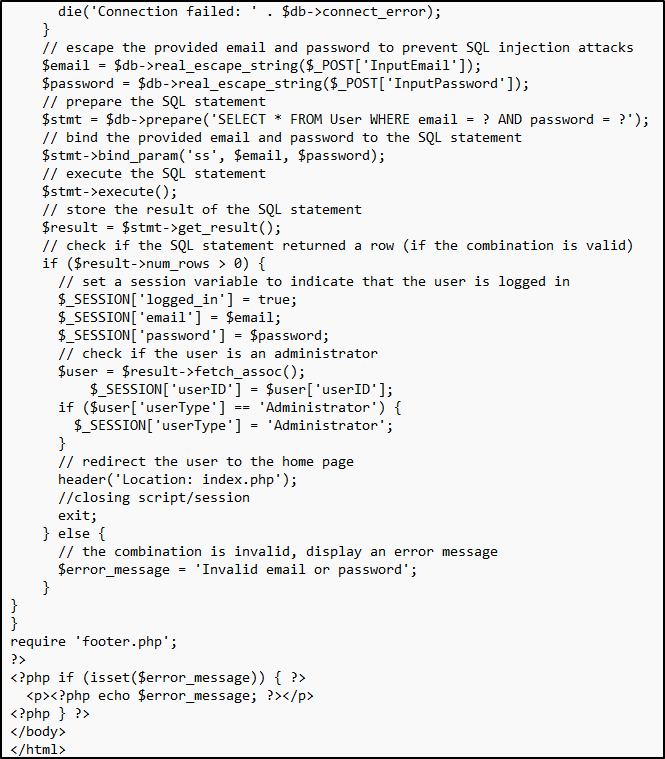
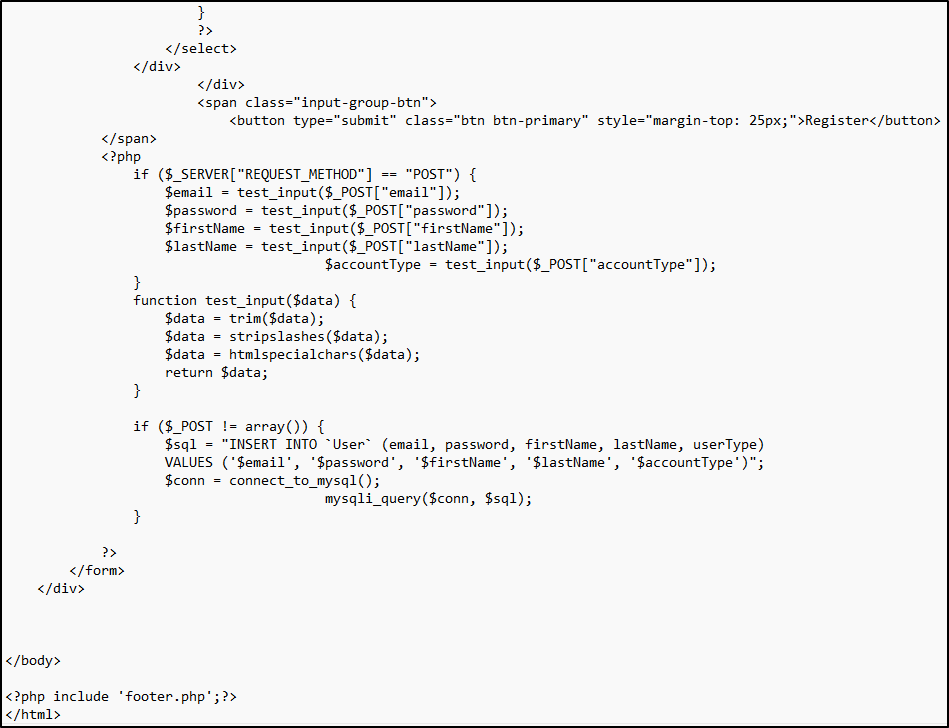


Figure 29

Registration Page PHP Code





Appendix F: To Be Determined List

Collect a numbered list of the TBD (to be determined) references that remain in the SRS to be tracked to closure.

**References**

Bose, S. (2022, June 2). *What to consider when evaluating a Test Automation Tool: Checklist*. <https://www.browserstack.com/guide/test-automation-tool-evaluation-checklist>

GeeksforGeeks. (2020, June 04). *Defect Severity*. <https://www.geeksforgeeks.org/defect-severity/>

GeeksforGeeks. (2023, July 19). *Software Engineering | Black box testing*. <https://www.geeksforgeeks.org/software-engineering-black-box-testing/>

Mikoluk, K. (2013, September 18). [XAMPP tutorial: How to use XAMPP to run your own web serverLinks to an external site.](https://blog.udemy.com/xampp-tutorial). Udemy. https://blog.udemy.com/xampp-tutorial

NYU. (n.d.) *System Requirement Specifications (SRS)*. <https://cs.nyu.edu/~jcf/classes/CSCI-GA.2440-001/handouts/Assignment1SampleSolution.pdf>

Software Testing Help. (2023, June 22). *How To Choose The Best Automation Testing Tool (A Complete Guide)*. <https://www.softwaretestinghelp.com/automation-testing-tutorial-4/>

Spillner, A., Linz, T., & Schaefer, H. (2014). *Software Testing Foundations: A Study Guide for the Certified Tester Exam (4th ed.)*. <https://ashford.redshelf.com/app/ecom/shelf/course-section/6094369>

Titov, M. (2023, February 02). *How to Write An SRS Document, or Software Requirements Specification*. <https://www.altigee.com/magazine/how-to-write-an-srs-document-or-software-requirements-specification>

Tsui, F., Karam, O., & Bernal, B. (2018). [Essentials of software engineering](https://uagc.instructure.com/courses/126521/modules/items/6439323)(4th ed.). Jones & Bartlett Learning.

Wiegers, K. E. (1999). [*Software requirement specifications for <project>*Links to an external site.](https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc) [Template]. <https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc>

W3schools. (n.d.). PHP MySQL Delete Data. <https://www.w3schools.com/php/php_mysql_delete.asp>