**Week 5 Assignment**

**Final Software Project**

Steven Warnock

The University of Arizona Global Campus

CST 499 Capstone for Computer Software Technology

Charmelia Butler

11/2/2024

**SRS Document**

**Software Requirements Specification**

**for**

**Online Student Enrollment**

**Version 1.0 approved**

**Steven Warnock**

**UAGC**

**10/7/2024**

**Software Requirements Specification (SRS) Document**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to provide a description of the online course registration system. This document outlines the functional and non-functional requirements of the system, focusing on user registration, course management, and enrollment processes. It is intended for developers, testers, and stakeholders to ensure all requirements are implemented correctly.

**1.2 Scope**

The system allows users to register, create profiles, and enroll in courses offered during three academic semesters (spring, summer, fall). The system manages course offerings, enrollments, and waitlists, ensuring students can access available courses and be informed if a space becomes available. The system will be used by students and administrators.

**1.3 Definitions, Acronyms, and Abbreviations**

* **User**: Any individual who registers for an account in the system (e.g., students).
* **ID**: Unique identifier for a user.
* **Profile**: Personal information associated with a user.
* **Course**: An educational class offered online.
* **Waitlist**: A queue of users waiting to enroll in a course if it is full.
* **SRS**: Software Requirements Specification.

**1.4 References**

* IEEE SRS Standard 830-1998.
* Project documentation and design guidelines.

**1.5 Overview**

This document is divided into multiple sections. Section 2 provides an overview of the system, its context, and product functions. Section 3 specifies detailed functional and non-functional requirements.

**2. Overall Description**

**2.1 Product Perspective**

The online course registration system is a self-contained system designed to allow students to manage their course enrollments efficiently. The system will be web-based and accessible through standard web browsers.

**2.2 Product Functions**

* User Registration and Login.
* Profile Management.
* Course Listing by Semester.
* Course Enrollment.
* Waitlist Management.
* Enrollment Cancellation and Waitlist Notification.

**2.3 User Characteristics**

The primary users are students looking to enroll in online courses. Users are expected to have basic computer literacy skills.

**2.4 Constraints**

* The system should enforce unique user IDs during registration.
* Course capacities must be strictly adhered to, and no over-enrollments should occur.
* The system must ensure secure handling of user credentials.

**2.5 Assumptions and Dependencies**

* The system assumes that users have internet access.
* Course offerings are predetermined by administrators.

**3. Specific Requirements**

**3.1 Functional Requirements**

1. **User Registration**
   * The system shall allow new users to register by providing personal information such as name, email, phone number, and a chosen password.
   * The system shall ensure that the user ID is unique. If a duplicate ID is found, the system will prompt the user to choose a different ID.
   * Upon successful registration, the system shall create a profile for the user with a unique ID and password.
2. **User Login**
   * The system shall allow registered users to log in using their unique ID and password.
   * The system shall verify user credentials before granting access.
3. **Profile Management**
   * The system shall store user profiles containing information such as name, phone number, and email address.
   * The user shall be able to update their profile information at any time.
4. **Course Management**
   * The system shall display a list of courses available for each semester (spring, summer, fall).
   * Each course listing shall include the course name, description, and maximum number of enrollments.
5. **Course Enrollment**
   * The system shall allow users to enroll in courses with available seats.
   * If the course is full, the system shall allow users to join a waitlist.
   * The system shall notify the user if a seat becomes available and they are next on the waitlist.
6. **Waitlist Management**
   * The system shall maintain a waitlist for each course that has reached its maximum enrollment.
   * The system shall automatically add users to the waitlist when attempting to enroll in a full course.
   * When a user drops a course, the system shall notify the first user on the waitlist to enroll.
7. **Enrollment Cancellation**
   * The system shall allow users to cancel their enrollment in a course.
   * The system shall update the course enrollment status and notify the next user on the waitlist.

**3.2 Non-Functional Requirements**

1. **Usability**
   * The system interface shall be intuitive and user-friendly, allowing students to easily navigate and manage enrollments.
2. **Performance**
   * The system shall handle up to 500 simultaneous users without performance degradation.
3. **Security**
   * The system shall use encryption protocols to protect user credentials.
   * Passwords shall be stored using secure hashing techniques.
4. **Reliability**
   * The system shall have an uptime of 99.5% to ensure availability during peak registration periods.
5. **Scalability**
   * The system should be scalable to accommodate a growing number of courses and students.

**4. System Features**

**4.1 Registration and Login Module**

* **Description**: Allows users to create accounts and log in using secure credentials.
* **Inputs**: User-provided data such as name, email, phone number, ID, and password.
* **Outputs**: Confirmation messages and user-specific dashboards.

**4.2 Course Management Module**

* **Description**: Displays available courses for the selected semester.
* **Inputs**: Semester selection by the user.
* **Outputs**: Course listings with current enrollment status.

**4.3 Enrollment Management Module**

* **Description**: Enables users to enroll, drop, or join waitlists for courses.
* **Inputs**: Course selection and user action (enroll/drop).
* **Outputs**: Enrollment confirmation or waitlist position.

**5. Appendices**

**A screenshot of a computer

Description automatically generated**

**UML Design Model**

**A diagram of a person

Description automatically generated**

**UML Class Diagram**

**A diagram of a course

Description automatically generated**

**Activity Diagram**

**A diagram of a student's course

Description automatically generated**

Testing is a critical component of software development, ensuring that systems function as intended and meet user expectations. For the online student enrollment portal, testing must be conducted at various levels to identify and resolve issues early, guarantee smooth module integration, and confirm that the system fulfills business and user requirements. The best approach for testing is to use the V-model in which each development phase corresponds directly to a testing phase, ensuring that testing is integrated throughout the entire development process rather than being a separate, final phase. As noted by Firesmith (2013), the v-model testing procedure “clearly represents the primary engineering activities in a logical flow that is easily understandable and balances development activities with their corresponding testing activities”. This approach includes different testing levels: component testing, integration testing, system testing, and acceptance testing. Each level addresses specific aspects of the system's development, from individual modules to the fully integrated solution.

Componenttesting is used to verify individual components or modules of the enrollment portal work as expected in isolation. Components to be tested include the user registration module, course selection module, and payment processing module. User registration module will test the form validation for input fields, including student ID and email. Course selection module testing will ensure the course list is displayed correctly. Payment processing testing will verify the payment flow and ensure correct amounts and transactions are completed.

Integration testing is used to test the interaction between integrated components or modules to ensure they work together as a cohesive system. As noted by Geeks for Geeks (2024), “Integration testing is performed in the Architecture design phase”. Integration scenarios include student registration and document submission, course enrollment and payment gateway, and error handling across modules. Student registration and document submission verifies that once the student completes registration, the document upload feature is triggered, and the data is saved correctly in the system. Course enrollment and payment gateway testing checks that after the student selects a course and the admin approves it, the payment process flows smoothly between the enrollment module and the payment gateway. Error handling across modules tests how the system behaves if a failure occurs in one module (e.g., payment failure after course registration).

System testing is done to test the entire enrollment portal as a complete and integrated system to validate that it meets all specified requirements. System test scenarios include the end-to-end enrollment process, system usability, security testing, and performance testing. The end-to-end enrollment process tests the complete flow from student registration to course selection, document submission, admin approval, and payment completion. System usability ensures the system is user-friendly, including navigation, page load times, and responsiveness. Security testing tests the login process, encryption of sensitive information (like payments and personal data), and proper access controls for different user roles of student, admin, and registrar. Performance testing simulates multiple concurrent users to ensure the system performs under high traffic conditions without significant delays or failures.

Acceptance testing is used to verify that the system meets the business and user requirements and is ready for deployment. This is the final testing phase, usually done by the client or end-users. As mentioned by Mathur and Malik (2010), “business requirements are used to guide the user acceptance testing”. Acceptance testing scenarios include business requirement validation, user experience testing, and when applicable, regulatory compliance.Business requirement validation ensures that the portal fulfills the use cases as specified by the business, such as successful student enrollment, payment processing, and report generation. User experience testing is done to gather feedback from actual users to ensure that the system is easy to use and meets user expectations. Regulatory compliance verifies that the system complies with any applicable regulations, such as data protection laws or educational standards.

Testing at multiple levels—component, integration, system, and acceptance—is essential for the successful deployment of the online student enrollment portal. Each testing phase plays a distinct role in identifying and addressing potential issues, from individual modules to the complete system. By following this layered approach, the system can be thoroughly validated, ensuring a seamless user experience and meeting both technical and business requirements before it is rolled out to students and administrators.

**Landing, Login, and Enrollment Pages**

**Introduction**

Developing a student registration page is a simple part of any web-based system. This process involves creating a user-friendly page where students can easily register by providing authentication details, including username, password, and email. These details must be securely handled, stored, and linked to a database. In this week’s assignment, I will build three components of a user registration system: the landing page, the login page, and the registration page. Also, I will also create the necessary database structure using MySQL, connecting the web pages to a back-end database for data storage and access.

**How to run a PHP file in XAMPP**

To run a PHP file in XAMPP, follow these steps, first download and install XAMPP from [Apache Friends](https://www.apachefriends.org) and install it on your computer. Next, launch XAMPP, and click *start* next to Apache and MySQL. Next, a project folder should be created. Navigate to the htdocs directory within the XAMPP installation folder, and created a new folder within htdocs for your project. Inside the new project folder, create a PHP file using a text editor. After the file is created with PHP code, run the PHP file by opening your web browser and type <http://LocalHost/foldername/filename.php>.

**MySQL DB Functions**

I used several functions for the creation of the database. First, I used mysqli\_connect() to establish a connection to the database. I also used mysqli\_query() to perform the SQL queries. Then I used mysqli\_real\_escale\_string () to return result rows in an array.

**Registration Page Development**

The layout of the registration page is designed to collect essential information that the system will use to authenticate and notify the registering user. Information collected includes a Username, Password, and Email. Below the text entry fields, a submit button completes the registration. Errors will occur if the user enters an invalid email address, or doesn’t complete all fields.

**Registration PHP Source Code**

<?php

// Database connection file

include\_once 'Database.php';

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$username = $\_POST['username'];

$password = password\_hash($\_POST['password'], PASSWORD\_DEFAULT);

$email = $\_POST['email'];

// Connect to database

$database = new Database();

$db = $database->getConnection();

// SQL insert statement

$sql = "INSERT INTO users (username, password, email) VALUES (?, ?, ?)";

$stmt = $db->prepare($sql);

$stmt->bind\_param("sss", $username, $password, $email);

// Execute the statement and validate

if ($stmt->execute()) {

echo "Registration successful!";

} else {

echo "Error: " . $stmt->error;

}

}

?>

**Table that saves user info in DB**

The users table structure stores the user info. Fields include id, username, password, and email. The id type is INT that auto increments, while the rest are VCHAR.

**Steps taken to create the registration page and save user info to DB**

An HTML form was created to gather the user's username, password, and email. Input validation, that checks required fields, is done by HTML5 form attributes. A custom class is used to perform the database connection, ensuring that the code is modular and easy to maintain. This class establishes a connection to the MySQL database using mysqli. The mysqli\_real\_escape\_string() function is used to prevent SQL injection, and passwords are hashed using PHP’s password\_hash() function for secure storage. After sanitizing and hashing the user input, an SQL query inserts the data into the users table. A prepared statement (bind\_param()) is used to prevent SQL injection. The user is notified of the registration's success or failure by a simple message displayed on the screen.

**Conclusion**

The development of the student registration page includes front-end design and back-end integration to ensure a simple and secure user experience. By creating the landing, login, and registration pages, and linking them to the MySQL database, I completed a fully functional registration system. Each step, from designing the HTML forms to implementing secure password storage, ensures that user data is handled responsibly. This approach facilitates easy student registration and also forms the base for future added functionalities including user authentication and data management within academic systems.

**Screenshots**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A computer screen shot of a program

Description automatically generated

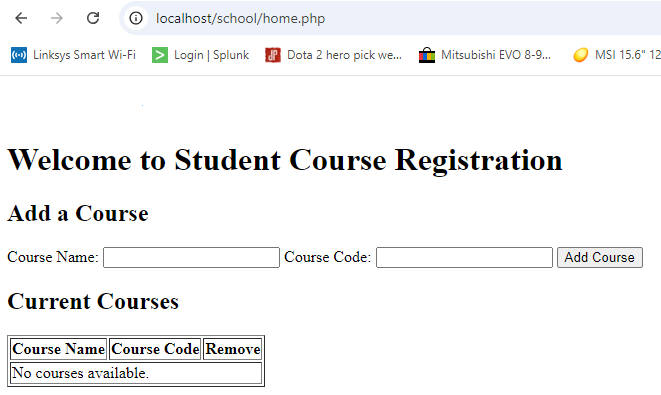
A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

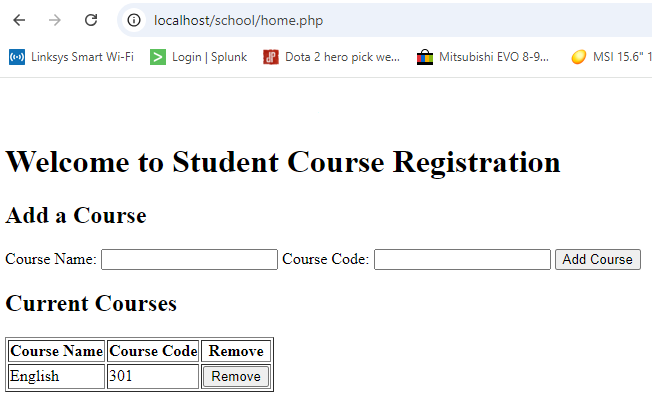
Description automatically generated

**MySQL Database and Class Registration**

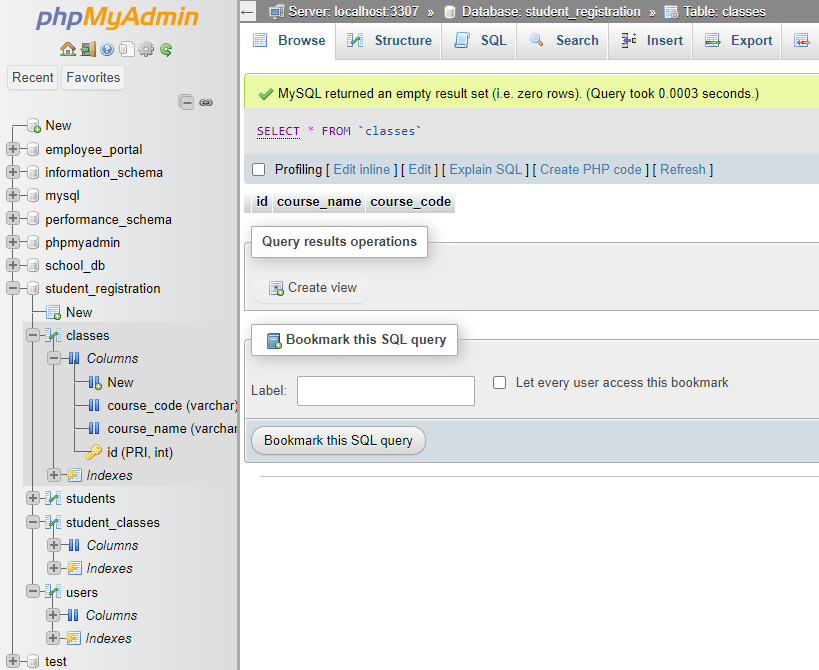
After sign in



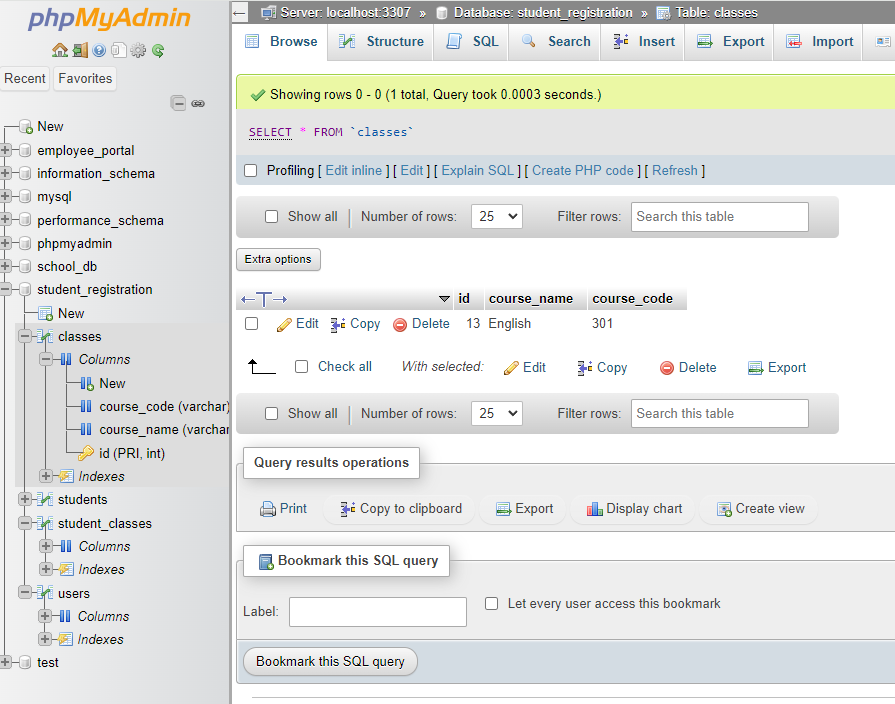
Adding a class



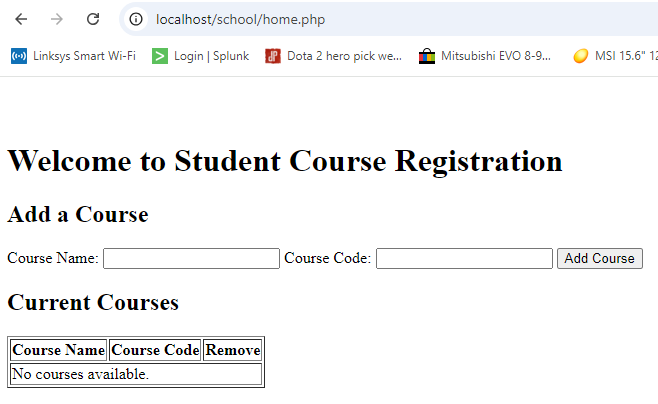
Database without classes



Database after class add



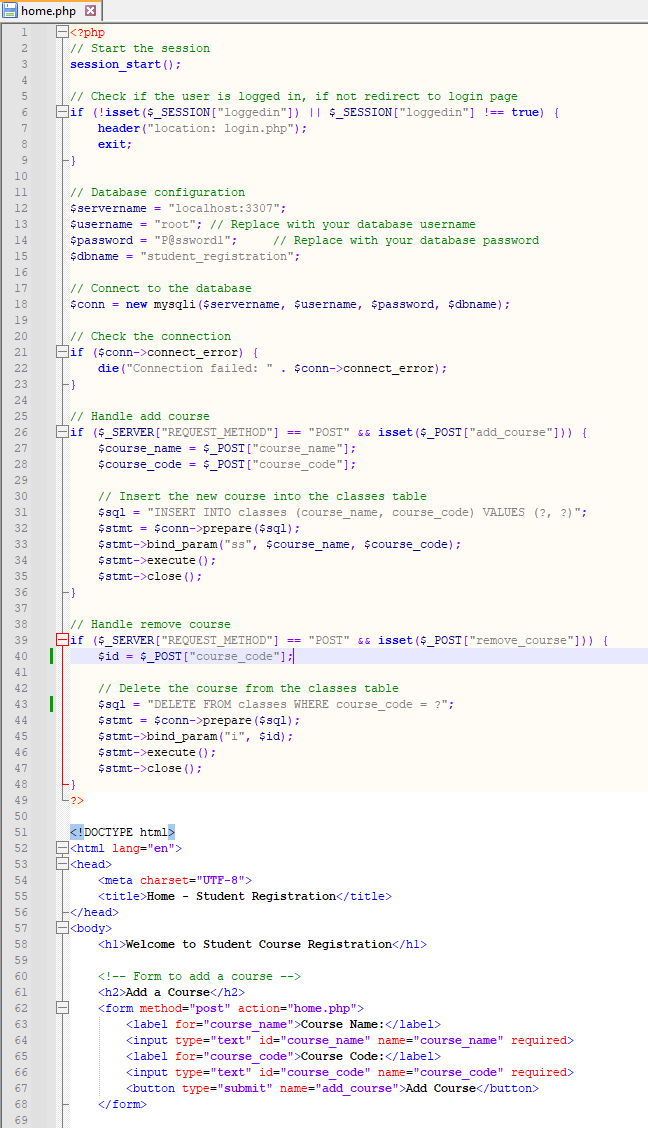
Click Remove, class removed

v

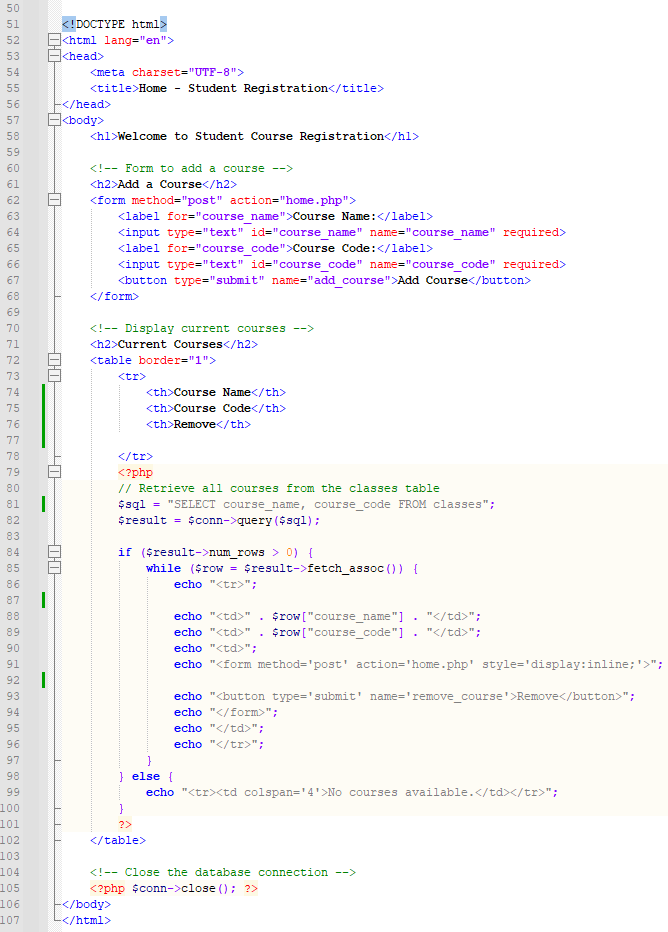
**PHP Code**

Home.php, including course registration list, add, and remove.

½



2/2



**Summary through implementation phase**

I wrote additional php code to add a home.php site that would load upon successful authentication at the login.php page. Once on the home page, the logged in users class list was shown. If a class is requested to be added, the user inputs the course name and course ID, and after clicking Register, the class is added to the bottom. Clicking Remove removes the class from the user’s class list. All classes are stored on the database of the specific user that is logged in to the session.

Going through the implementation was a hands-on experience that brought the project’s theoretical aspects into actual use. It required working closely with the database setup and user interface to ensure successful interaction. Early on, I identified the importance of session management for login authentication, which was critical in restricting access and managing user sessions effectively.

Debugging was essential, especially when handling form data and ensuring input names matched their expected references in PHP. This also reinforced the need for clean and consistent code because small mismatches in variable names and form handling led to warnings, highlighting the importance of detail-oriented coding practices. Also, troubleshooting connection issues with the database configuration helped refine my approach to error handling and testing across different environments.

Overall, this project greatly helped me improve my skills in PHP, MySQL, and session management while providing valuable insights into how detailed testing and debugging led to a much better application.

**References**

(2024) Classification of Software Requirements – Software Engineering. *GeeksforGeeks.* <https://www.geeksforgeeks.org/software-engineering-classification-of-software-requirements/>

(2024) SDLC V-Model – Software Engineering.  *Geeks for Geeks*. <https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/>

Bolloju, N., Alter, S. (2016) Better Use Case Diagrams by Using Work System Snapshots. *The University of San Francisco.* <https://repository.usfca.edu/cgi/viewcontent.cgi?article=1088&context=at>

Connolly, R., & Hoar, R. (2022). [*Fundamentals of web development*](https://uagc.instructure.com/courses/137971/modules/items/7033225) (3rd ed.). Pearson.

Firesmith, D. (2013) Using V Models for Testing. *Software Engineering Institute of Carnegie Mellon University.* <https://insights.sei.cmu.edu/blog/using-v-models-for-testing/>

IEEE Recommended Practice for Software Requirements Specifications. *IEEE Std 830-1998* , vol., no., pp.1-40, 20 Oct. 1998, doi: 10.1109/IEEESTD.1998.88286. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=720574>

Krishna, M., Gaur, B., Verma, A., & Jalote, P. (2024) Using LLMs in Software Requirements Specifications: An Empirical Evaluation. *2024 IEEE 32nd International Requirements Engineering Conference (RE).* [*https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10628461*](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10628461)

Mancordis, S. (1999, September 27). [*Topics in object-oriented design patterns*Links to an external site.](http://research.cs.queensu.ca/home/ahmed/home/teaching/CISC322/F08/files/Slides_Spiros_Patterns.pdf) [Presentation slides]. Retrieved from <http://research.cs.queensu.ca/home/ahmed/home/teaching/CISC322/F08/files/Slides_Spiros_Patterns.pdf>

Mathur, S., Malik, S. (2010) Advancements in the V-Model. *International Journal of Computer Applications.* <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=04aca97824d178d7ca3688bbed2118d0115dfaba>

Nasir, O. (2023) 7 Tips to identify Software Requirements for a Successful Development Cycle. *Tkxel.* <https://tkxel.com/blog/7-tips-to-identify-software-requirements-for-a-successful-development-cycle/>

Psinas, M. (2012, March 12). [Role based access control in PHPLinks to an external site.](https://www.sitepoint.com/role-based-access-control-in-php/). Retrieved from https://www.sitepoint.com/role-based-access-control-in-php/

Tsui, F., Karam, O., & Bernal, B. (2018). [*Essentials of software engineering*](https://uagc.instructure.com/courses/137971/modules/items/7033225)(4th ed.). Jones & Bartlett Learning.