Software Engineering and Web Technologies Laboratory

Integrated Project on

WHO Quality of Life

Bachelor of Engineering

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

Team No: 13

Div: E

Name	SRN	RollNo.
Bibijan A Matte	01fe21bcs287	562
Tanushree Kabbur	01fe21bcs262	558
Aditya Manjinal	01fe21bcs324	540
Sanjana Murgod	01fe21bcs357	566

Faculty In charges

Dr. P.G. Sunita Hiremath, Miss. Neha T

SCHOOL OF COMPUTER SCIENCE & ENGINEERING
HUBLI–580 031 (India).

Academic year 2023-24

Chapter 1
Introduction

1.1 Preamble

The World Health Organization (WHO) underscores the pivotal role of human well-being by establishing a comprehensive framework devoted to evaluating and enhancing Quality of Life (QoL). Anchored in principles of equity and inclusivity, this document guides global efforts toward holistic development, viewing QoL as a multidimensional construct encompassing physical health, mental well-being, social relationships, and environmental sustainability. The WHO commits to collaborative endeavors, research, and policies fostering a world where every individual, regardless of location or socio-economic status, can lead fulfilling lives. Pledging strong partnerships, the WHO aims to empower individuals and societies, fostering a global landscape characterized by robust health, enduring happiness, and harmonious coexistence. This commitment extends to evidence-based policies addressing diverse global needs, championing innovative solutions for collective prosperity. Embracing a vision of a healthier, happier world, the WHO invites nations, organizations, and individuals to join in shaping a future where well-being is a universal priority, contributing to a tapestry of collective prosperity.

1.2 Problem Statement

Design and develop a website that assesses and evaluates the quality of life for individuals. The platform should provide comprehensive and personalized insights to empower users in making informed decisions to enhance their overall quality of life.

1.3 Objectives

- To assess an individual's quality of life.
- Calculating the domain summary scores
- Evaluate the results of the test.
- Help user understand their results and how they can use them to feel better or get better help if they need it.

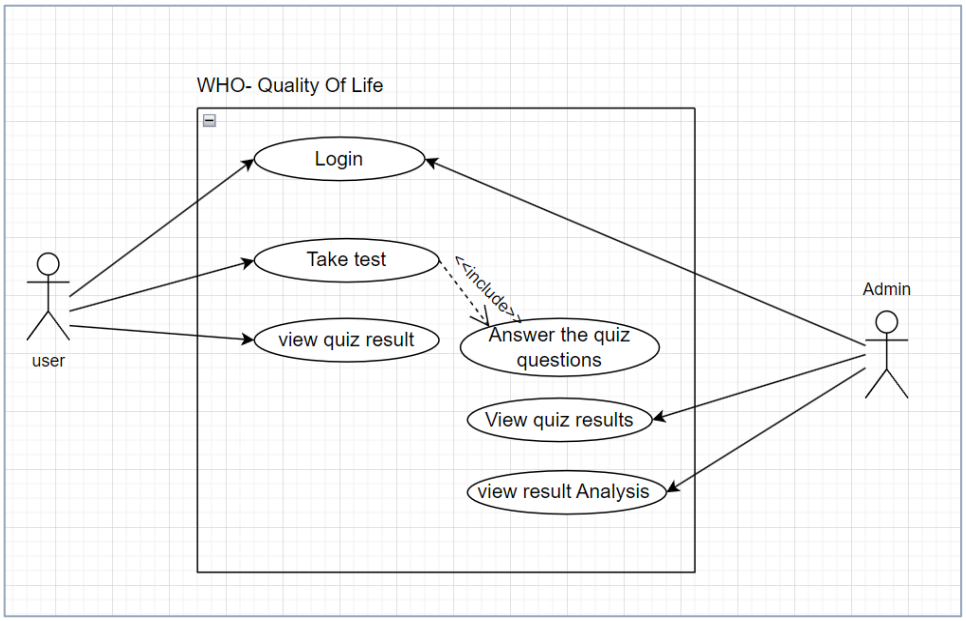
Chapter 2

Software Requirement Specifications

2.1 Functional Requirements and Use case Diagram

1. The system shall provide a secure login interface for users.
2. The system shall display the instructions for attempting the test.
3. The system shall display all the questions.
4. The system shall display Likert scale for every question ranging from 1 to 5.
5. The system shall calculate and display the end score by taking the average score of all the domains.
6. It shall display individual scores for each domain (physical health, psychological well-being, social relationships, environment).
7. The system shall calculate and display the questions with the least score from all the domains. The system shall display the domain with the lowest score from the user's responses.
8. It shall compare the user's individual domain scores with average scores across all domains.

Use Case Diagram



2.4 Test Cases

Test Case ID	Steps	Expected Output	Actual Output	Test Case Result
TC_01	Valid details in the QoL assessment form	Login Successful	Login Successful	Pass
	Valid details in the QoL assessment form	Login Successful	Unsuccess Login	Fail
TC_02	Attempt to assess QoL with incomplete details	Display error message indicating invalid input	Invalid user details	Pass
TC_03	Attempt to move to the next question without selecting an option	Display message to select an option	Select the option before moving	Pass
TC_04	Select an option and attempt to move to the next question	Next question is displayed	Next question displayed	Pass
TC_05	Admin logs in with valid credentials	Successful login	Access to admin dashboard	Pass
TC_06	Admin logs in with invalid credentials	Display error message indicating invalid login	Invalid Username or Password	Pass
TC_07	Test user verification time	User	1 sec	Pass

		verification completed within 1 minute		
TC_08	Verify web pages load within specified time	Web pages load within 5 seconds on average	5 sec	Pass
TC_09	Check system response time during regular usage	Response time within 3 seconds under normal load	1.00 min	Pass

2.2 Non-functional Requirements

1. Web pages shall load within 3 seconds on average, considering an average user's internet speed.
2. The system shall verify the user within 1 min.
3. The system shall handle a minimum of 10,000 concurrent users without a decrease in performance.
4. Within 1 minute after submitting the answers, the system should display the result.

2.3 Hardware and Software Requirements

Hardware Requirements,

- Processor: Dual-core or higher
- RAM: 8 GB
- Storage: SSD storage for improved performance
- Network: High speed internet connection

Software Requirements

- Visual Studio Code

Chapter 3

System Design & Implementation

3.1 MVC Architecture

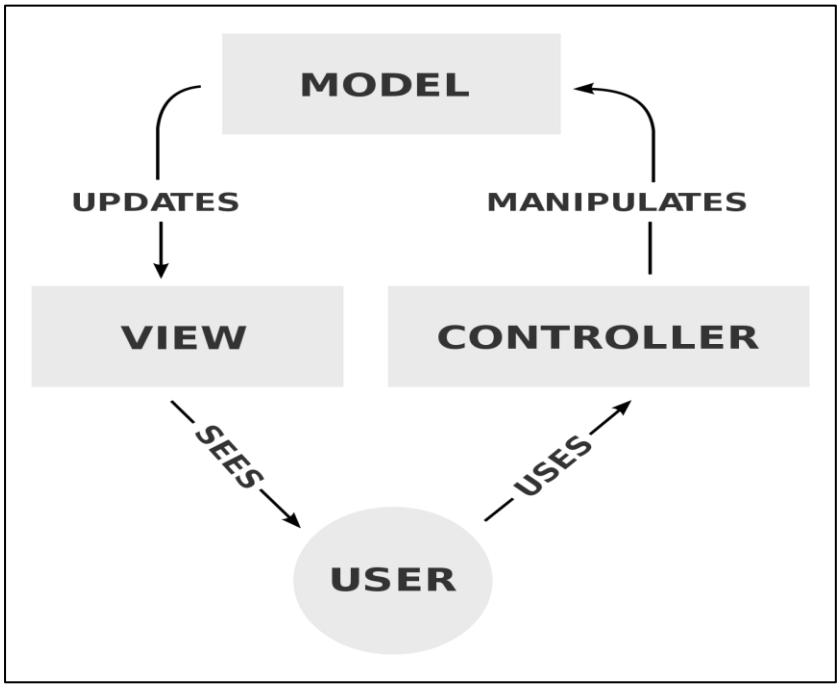


Figure 3.1 MAV Architecture

Model (M)

The Model represents the application's data and business logic. It is responsible for managing the data, processing user inputs, and responding to queries from the View or Controller. The Model is independent of both the View and Controller, ensuring that changes in one component do not directly affect the others.

View (V)

View is responsible for presenting the data to the user in a human-readable format. It displays information retrieved from the Model and sends user inputs to the Controller for processing. It represents the user interface (UI) components of the application. Views can be anything from HTML pages and templates to graphical user interfaces (GUIs).

Controller (C)

The Controller handles user input and acts as an intermediary between the Model and the View. It processes user requests, updates the Model accordingly, and selects the appropriate View for displaying the updated data. It serves as the bridge between the user interface and the application's logic. Controllers interpret user actions and manipulate the Model and View accordingly.

3.2 MERN Framework

The MERN stack is a popular technology stack for building web applications. It's an acronym that stands for MongoDB, Express.js, React.js and Node.js. Each component in the stack plays a specific role in the development process.

1. **MongoDB:** MongoDB is a NoSQL database that stores in a flexible, JSON-like format called BSON (Binary JSON). It is well-suited for handling large amounts of unstructured or semi-structured data. MongoDB is used as the database layer in the MERN stack.
2. **Express.js:** Express.js is a web application framework for Node.js. It facilitates the creation of APIs (Application Programming Interfaces) and handling HTTP requests and responses. It acts as the backend framework in the MERN stack.
3. **React.js:** React.js is a JavaScript library developed by Facebook for building user interfaces. It follows a component-based architecture, allowing developers to build reusable UI components.
4. **Node.js:** Node.js is a JavaScript runtime that allows developers to run JavaScript on the server side. It is designed to be lightweight, efficient, and scalable. Node.js enables the execution of JavaScript code outside the browser, making it suitable for building server-side applications. It serves as the runtime environment for the MERN stack.

MERN Stack Workflow:

1. **Frontend (React.js):**

- Develop the user interface using React.js components.
- Manage application state using Reacts state management.
- Interact with the backend through API calls.

2. **Backend (Node.js and Express.js):**

- Create RESTful APIs using Express.js for communication with the frontend.
- Implement business logic and handle data processing on the server side.
- Communicate with the database (MongoDB) to store and retrieve data.

3. **Database (MongoDB):**

- Store and manage application data in MongoDB.
- Utilize MongoDB's flexible document-oriented structure.

4. **Runtime Environment (Node.js):**

- Run server-side JavaScript code using Node.js.
- Enable the server to handle incoming requests and respond with data.

3.3 Detailed Database Description (MongoDB)

MongoDB is a popular, open-source NoSQL database management system. Unlike traditional relational databases, MongoDB is classified as a document-oriented database, belonging to the family of NoSQL databases.it stores data in flexible, JSON-like documents, which makes it easier to handle and scale for certain types of applications. It is widely used in various applications, especially in scenarios where flexibility, scalability and quick development cycles are essential, such as content management systems, mobile applications and real-time analytics.

Key Features:

1. **Document-Oriented:** MongoDB stores data in BSON (Binary JSON) documents, which can have varying structures, making it more flexible than traditional relational databases.
2. **Schema-less:** MongoDB doesn't require a predefined schema, allowing for easy and dynamic modification of data structures.
3. **Scalability:** It is designed to scale horizontally, handling large amounts of data by distributing it across multiple servers.
4. **Query Language:** MongoDB uses a JSON-style query language that is expressive and powerful, allowing for complex queries and indexing.
5. **Indexing:** Supports secondary indexing, improving query performance.
6. **Aggregation Framework:** Provides a powerful and flexible aggregation framework for data processing and analysis.
7. **Geospatial Indexing:** Offers geospatial indexing for location-based data.
8. **Replication and High Availability:** Supports replica sets for data redundancy and high availability.

3.4 Modules description:

1. **CSS:** CSS, or Cascading Style Sheets, is a style sheet language used to describe the presentation of a document written in HTML or XML. It defines how elements should be displayed on a webpage, specifying aspects like layout, colors, fonts, and spacing. CSS enables the separation of content from design, making it easier to manage and style web pages.

   Implementation Details:

   - **Selectors:** CSS uses selectors to target HTML elements for styling.

- **Properties:** Define styling properties like color, font, margin, padding, etc.
- **Classes and IDs:** Apply styles selectively using classes and IDs.
- **Box Model:** Understand and leverage the CSS box model for layout.
- **Responsive Design:** Implement media queries to create responsive designs for various screen sizes.
- **Flexbox and Grid:** Utilize Flexbox and Grid layout systems for advanced layout structures.
- **Transitions and Animations:** Add dynamic effects using CSS transitions and animations.

2. **Node.js:** Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It allows developers to run JavaScript code on the server side, enabling server-side scripting and the development of scalable network applications. Node.js is event-driven and non-blocking, making it suitable for building fast and efficient applications.

   Implementation Details:

   - **Server Creation:** Use Node.js to create a server that handles HTTP requests and responses.
   - **NPM (Node Package Manager):** Manage project dependencies using NPM.
   - **Modules:** Leverage the module system to organize and modularize code.
   - **Express.js:** Build web applications and APIs using the Express.js framework.
   - **Middleware:** Implement middleware for tasks like logging, authentication, etc.
   - **Asynchronous Programming:** Utilize asynchronous programming patterns with callbacks, promises, and async/await.
   - **Routing:** Define routes to handle different endpoints in a web application.

3. **React.js:** React.js is a JavaScript library for building user interfaces. Developed by Facebook, react simplifies the process of building interactive and dynamic UIs. It follows a component-based architecture, allowing developers to create reusable UI components that update efficiently in response to changes.

   Implementation Details:

   - **Components:** Build UI elements as React components.
   - **JSX (JavaScript XML):** Use JSX to write HTML-like code in JavaScript files.
   - **React Router:** Set up routing for single-page applications.
   - **State Management:** Utilize state management libraries like Redux or Context API for managing application state.
   - **Hooks:** Leverage hooks like useState and useEffect for functional component logic.
   - **Virtual DOM:** Understand and benefit from Reacts virtual DOM for efficient updates

Chapter 4

Results and Discussions

4.1 Results/Snapshots with Description (Query/Search Results):

Result Snapshot:

After the user successfully completes the test, the result of the test will be displayed as shown in Figure 4.1.1. The result gives detailed information about the scores scored in the four domains. This domain is used for analysis showing how well the user is doing in each domain.

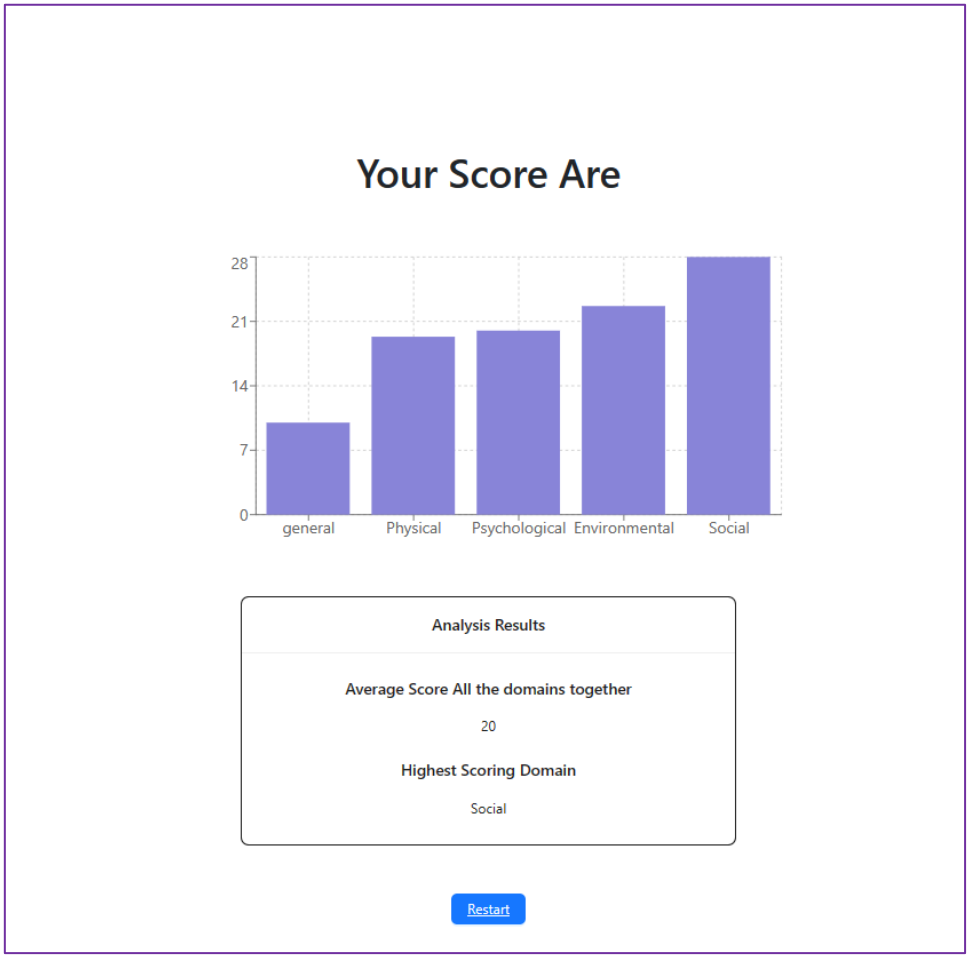


Figure: 4.1.1 Result Page

Admin page Snapshot:

Figure: 4.1.2 shows how the admin can perceive the analysis of the tests taken by the users. This will help the admin to take measures respectively. Dashboard shows detailed information about the marks scored gender wise.

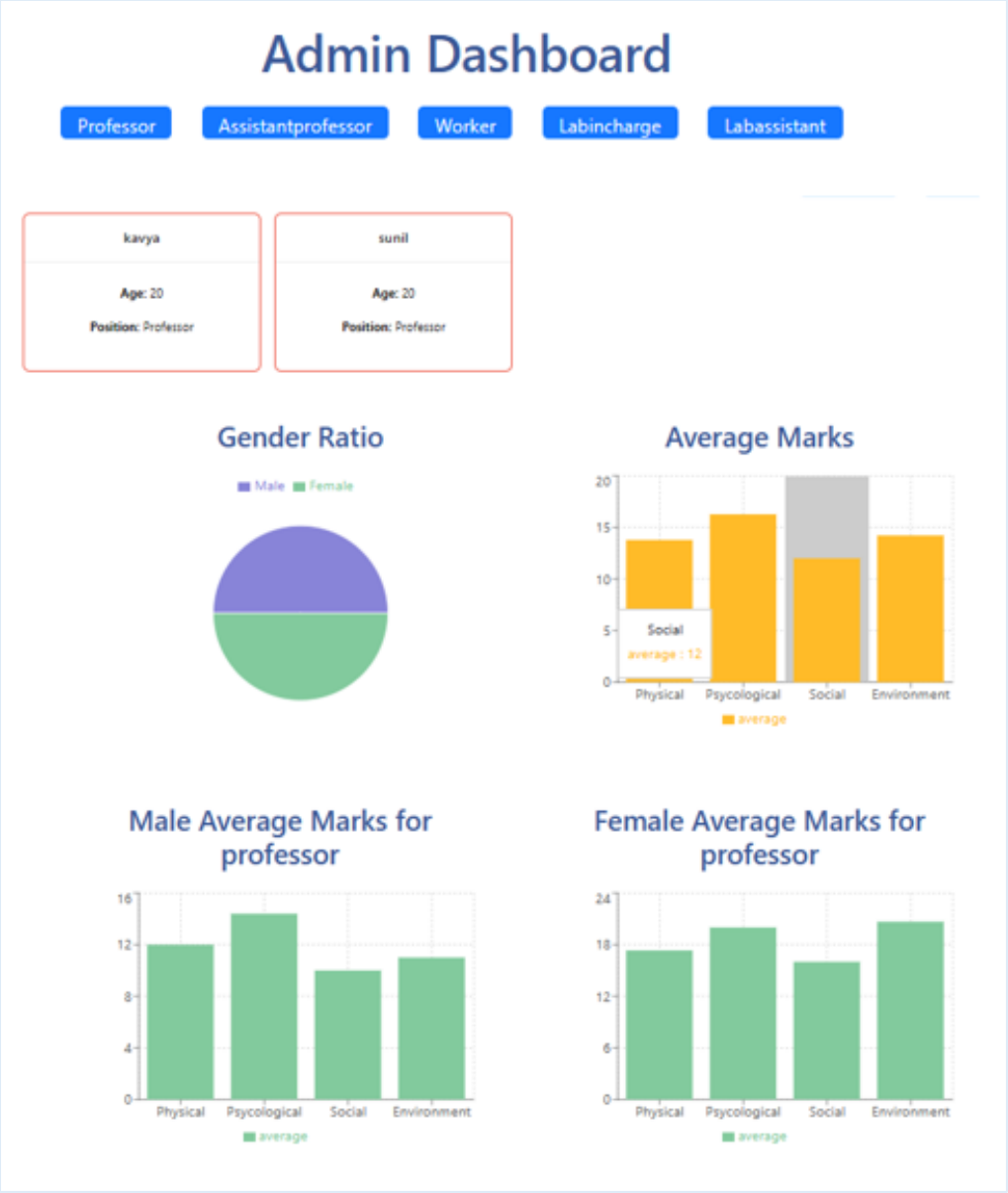


Figure: 4.1.2 Admin dashboard

4.2 Testing Report

Test 1: Verification of user

Passed: Input validation for user details.
Failed: If the user inputs his/her name other than alphabets, alert message will pop up

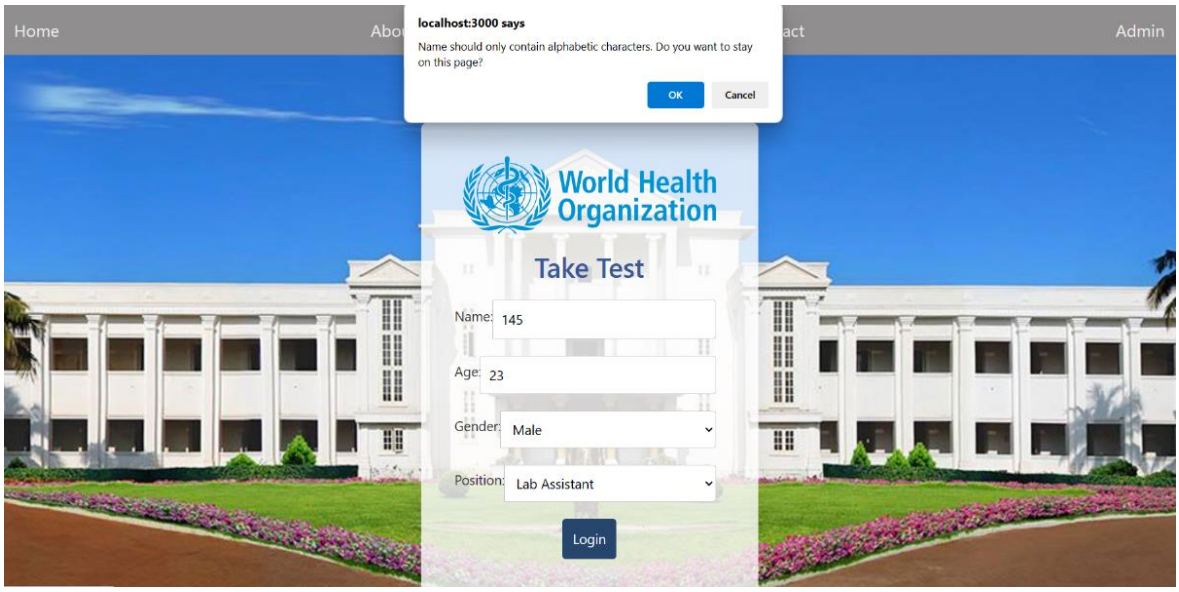


Figure: 4.2.1 Invalid Input

Test 2: Verification of Admin

Passed: Correct admin password can access the user dashboard.

Failed: If the admin inputs incorrect admin password, alert message would pop up.

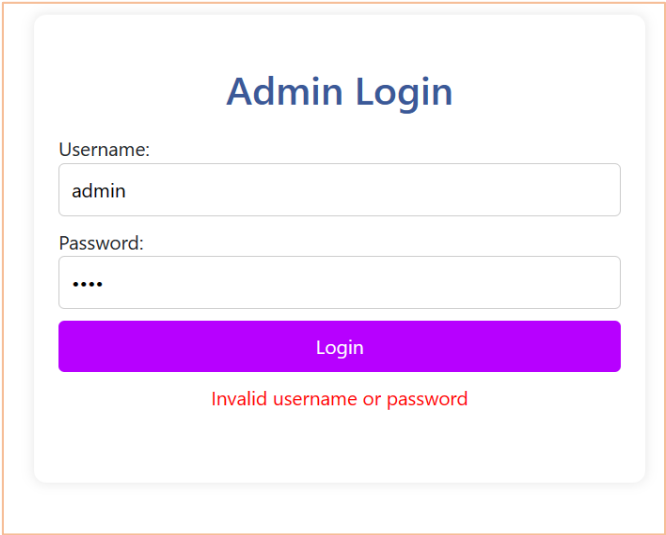


Figure: 4.2.2 Invalid Input

Test 3: Selecting the option

Passed: Option is selected for the given question

Failed: If the user does not select any option and try to move to the next question, an alert message will pop up. This will ensure that user answers all the question

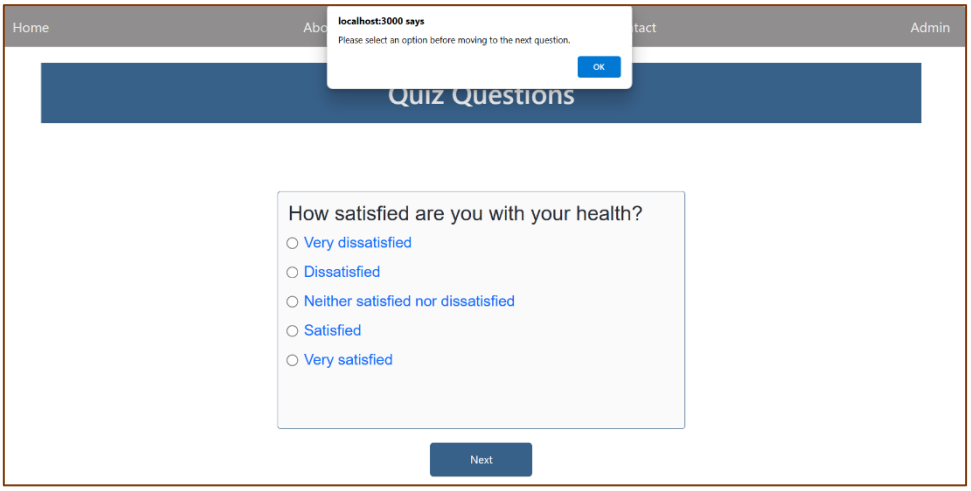


Figure: 4.2.3 Option not selected

4.3 Testing Tool:

Selenium: Selenium is a powerful and widely used open-source testing framework for web applications. It provides a suite of tools for automating web browsers, allowing developers and testers to perform automated testing, interact with web elements, and validate the functionality of web applications.

Testing Tool Used: Selenium for automated testing.

Discussion:

Selenium played a crucial role in automating various testing processes, ensuring thorough coverage and accuracy. Its integration significantly enhanced the testing phase's efficiency.

4.4 Continuous Integration and Continuous Delivery (CI/CD)

Jenkins: Jenkins, a robust CI/CD tool, automates testing and deployment processes, ensuring a streamlined and efficient development lifecycle.

Discussion:

Jenkins plays a pivotal role in maintaining code quality and ensuring the continuous delivery of stable releases. Its integration aligns with modern development practices, promoting efficiency and reliability.

Chapter 5: Conclusion & Future Scope

Conclusion:

In summary, the WHO Quality of Life website effectively addresses the goal of assessing and enhancing individuals' quality of life. The project meets its objectives, offering a comprehensive platform for users to gain valuable insights into their well-being.

Future Endeavors:

Additional Features:

Introduce personalized recommendations based on the assessment results.

Implement a feature for users to track their progress over time.

Usability Refinement:

Conduct user feedback sessions for insights into improving the user interface.

Implement accessibility features for a more inclusive user experience.

Performance Optimization:

Optimize database queries and implement caching mechanisms for improved performance.

Explore serverless architecture for scalability during peak usage.