
**Al-Hussein Bin Talal University**
**Information Technology Collage**

# Project name (WebWiz)

**Team Members:**
**Abedalaziz Abuawwad (Team leader)**
**Mohammad Hanoun**
**Wesam Barqawe**
**Nawras Aldahabi**
**Ra'ad Alreqeb**
**Aya Altamimi**
**Supervisor: (Professor Bassam Al-Qaralleh)**

**BACHELOR'S DEGREE IN INFORMATION TECHNOLOGY**
**AL-HUSSEIN BIN TALAL UNIVERSITY**
**[2024-2025]**

***Permission to Use***

*In presenting this project as part of the fulfilment of the requirements for a Bachelor's Degree in Information Technology from Al-Hussein Bin Talal University, we agree that the University Library may make it freely available for public access. Additionally, we consent that permission for copying any portion of this project for scholarly purposes may be granted by our supervisor(s) or, in their absence, by the College of Information Technology.*

*It is understood that any reproduction, publication, or use of this project or any of its parts for financial gain is strictly prohibited without our written consent. Proper acknowledgment must be given to us and Al-Hussein Bin Talal University for any scholarly use of materials derived from this project. Requests for permission to copy or utilize materials from this project, in whole or in part, should be directed to the College of Information Technology.*

*College of Information Technology*
*Al-Hussein Bin Talal University*
*Ma'an, Jordan*

# Abstract

Web development is expanding in both technologies and problem-solving approaches, and it has become a popular learning path for beginners. This has led to an abundance of learning resources; however, most of them lack practical application. As a result, learners often struggle to know what to practice.

Most front-end designs can be implemented in various ways, which can confuse learners about what the best practices are and when to apply different approaches. Therefore, most learners need guidance during project practice to understand the most suitable methods and when to use them.

WebWiz makes practicing web development and applying specific skills easier and more straightforward for learners at all levels by providing an environment with structured tasks to practice.

Learners can choose tasks based on difficulty from simple exercises and fundamental refreshers to advanced topics in HTML, CSS, JavaScript, and React. This helps learners improve their skills by applying them directly within the WebWiz platform.

WebWiz will utilize advanced AI models to support learners in applying their skills by providing an AI-powered evaluation system. This system will ensure that designs are created using best practices and valid implementation methods.

Furthermore, WebWiz will reduce the time and effort required for practicing web development skills by offering skill-based tasks that ensure progressive learning for all levels.

By leveraging AI models, the platform will provide learners with personalized guidance, evaluate their solutions, and offer corrections when needed making the learning process easier, faster, and more efficient. WebWiz aims to be an all-in-one solution for front-end web development education.

# Acknowledgement

First, we praise God for our success in this journey. First, we raise our hands in gratitude to God for our success in this journey.

In addition to thanking God, we would like to express our deep appreciation to our father, our great guide, and teacher, Professor Bassam Al-Qraleh. His role went far beyond teaching, guiding, or advising us throughout this journey. He stood by us emotionally, preparing us mentally before practically, encouraging us to enjoy the work, discover the spirit of teamwork, and understand that sometimes supporting each other is more important than merely achieving the goal.

We also extend our heartfelt thanks to the team members themselves, who always strived for excellence and worked with all their hearts. They chose to endure sleepless nights and hardships over shortcuts in this work, not only for their own future but out of respect for everyone who stood by them starting from their families and friends, to their supervisor, and the entire academic staff who played a vital role in bringing us to where we are today.

A special thank you goes to our families and friends, who have been a constant source of support, not only in this moment but in every step of our journey. They have always pushed us forward, helping us grow and become who we are today.

Finally, we offer our deepest gratitude to our esteemed professors and supervisors, and to the members of the discussion committee, who provided us with valuable guidance and constructive feedback that helped improve this work, as well as to all those whose names we may not recall, but whose contributions have been priceless. This work is the result of collective effort, built on mutual understanding, faith, and the shared drive to achieve success.

# Table of contents

Table of figuers

**Table of tables**



Table of abbreviation

| abbreviation | The full term |
|---|---|
| **AI** | Artificial inelegance |
| **API** | Application Programming Interface |
| **SRS** | Software requirement specification |
| **IT** | Information technology |

# Chapter 1
## INTRODUCTION

The continuous expansion of the Information Technology (IT) sector has created a growing demand for skilled professionals, particularly in web development—one of the most dynamic and in-demand areas in the industry. While countless learning resources and online courses are available, many learners struggle to move from theoretical understanding to practical application. This gap between knowledge and hands-on experience often prevents aspiring developers from reaching job readiness.

WebWiz was created to address this challenge by offering an interactive, skill-based learning environment focused on front-end development. Through structured coding tasks in HTML, CSS, JavaScript, and React, the platform enables learners to apply what they've learned in a realistic, task-driven context. By incorporating AI-powered solution evaluation, level-based task design, and clear feedback, WebWiz supports learners in building strong technical foundations and becoming confident, job-ready developers.

In addition to skill development, WebWiz enhances the overall learning experience by integrating features that support motivation, progress tracking, and accessibility. Learners can engage in friendly competition through a dynamic leaderboard. Moreover, they work through a personalized task roadmap that arranges coding challenges in a logical order starting from the beginner level, helping them know exactly where to start and how to progress. Additionally, learners can write and test code directly in a Live Coding Editor, making the practice immediate and interactive.

## 1.1 Project Background

In our time, the Information Technology (IT) sector has become an essential part of daily life due to its positive impact on various industries. This sector has witnessed rapid growth, creating numerous job opportunities and encouraging many individuals to pursue careers in IT and develop their skills in the field.

One of the most important areas within IT is Web Development. It focuses on building and developing websites and applications, encompassing both the programming and design aspects of front-end and back-end interfaces to ensure efficient user interaction and optimal website performance.

Web development is generally divided into two main branches:

- Front-end Development: Focused on designing the visual and interactive interface of the website.
- Back-end Development: Responsible for data processing and server-side management.

Web development is among the fastest growing and most widely used fields in the IT sector. According to a 2022 survey conducted by the well-known Stack Overflow platform which includes thousands of programming professionals worldwide 25.96% of the 61,302 participants specialized in front-end development. This places it third among developer types, following back-end and full-stack development. ***[1]***

These statistics highlight the significance and high demand for front-end development in the job market, as it plays a vital role in enhancing user experience and interaction with the web. Due to this increasing demand, competition among learners has also grown, requiring developers to attain a high level of skill and competence.

This emphasizes the importance of continuous learning and staying up to date with technological advancements. Learning alone is not enough it must be reinforced with frequent practice to gain real-world experience and valuable skills.

Our project was developed to bridge the gap between theoretical knowledge and practical application, empowering learners to enter the job market with confidence and solid capabilities.

## 1.2 Problem Statement

Despite the abundance of learning resources in the Information Technology (IT) sector ranging from online courses to university programs, many of these resources primarily focus on theoretical knowledge. This leaves a critical gap in hands-on, task-based learning, especially in front-end development, where practical experience is essential for mastering real-world skills.

Learners often face challenges such as the absence of structured learning paths, unclear starting points, and the overwhelming variety of tools and technologies. For beginners, this lack

of guidance leads to wasted time, confusion, and slow progress. Moreover, most platforms fail to provide interactive tasks that match the learner's current skill level, further hindering their ability to gain confidence and grow.

As the demand for skilled front-end developers continues to rise, so does the competition. Learners need accessible, high-quality resources that combine guidance, practical application, and personalized challenges to help them build solid, job-ready skills effectively .

WebWiz addresses these issues by offering a structured, interactive learning environment focused on front-end development, empowering learners to bridge the gap between theory and practice and achieve tangible growth.

## 1.3 Project Objectives

The primary goal of WebWiz is to help learners transform their knowledge into practical, job-ready skills in front-end development, empowering them to confidently enter the job market.

To achieve this, the platform offers coding tasks in HTML, CSS, JavaScript, and React, tailored to different skill levels based on each learner's background and experience. This hands-on approach enables learners to progressively enhance their abilities through practical application.

WebWiz also fosters problem-solving skills by evaluating submitted solutions once a task is completed correctly, helping learners write clean, professional code while understanding best practices in utilizing modern technologies effectively.

Moreover, the platform strengthens learners' grasp of front-end logic and user interface behavior, equipping them to build dynamic, responsive, and user-friendly web applications with confidence and accuracy.

## 1.4 Project Significance

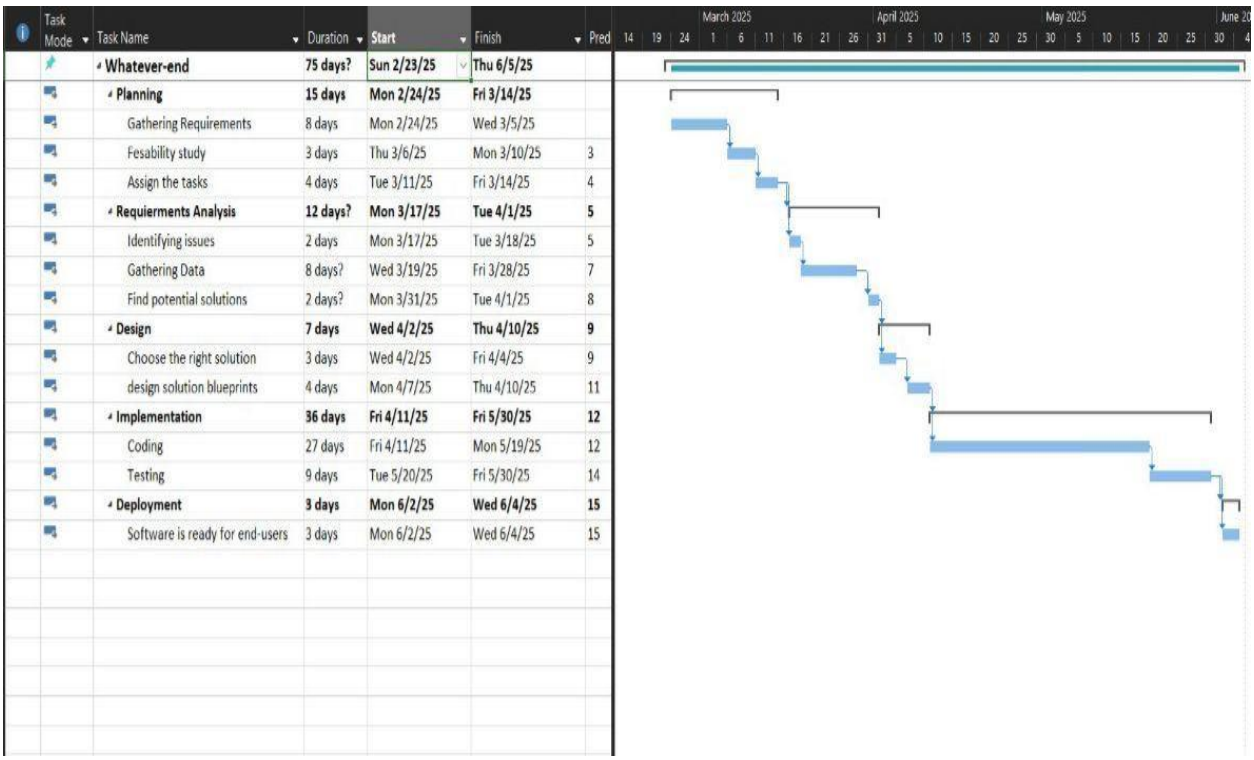
The WebWiz platform addresses the critical shortage of practical, level-based learning resources in front-end development by offering an interactive and structured environment. It bridges the gap between theoretical knowledge and hands-on application, enabling learners to develop comprehensive and job-ready skills.

Catering to both beginners aiming to build a strong foundation and experienced developers seeking to advance their expertise, WebWiz provides a variety of coding tasks in HTML, CSS, JavaScript, and React, tailored to different skill levels.

By evaluating submitted solutions upon task completion, the platform fosters problem-solving skills and reinforces best practices in modern front-end technologies. This approach accelerates learners' progress and ensures a deeper understanding of efficient and professional coding standards.

Ultimately, WebWiz equips learners with practical experience and mastery of front-end development concepts, preparing them to compete effectively in the job market and meet industry demands.

## 1.5 Project Gantt Chart



| Task Mode | Task Name | Duration | Start | Finish | Pred |
|---|---|---|---|---|---|
| | **Whatever-end** | **75 days?** | **Sun 2/23/25** | **Thu 6/5/25** | |
| | **Planning** | **15 days** | **Mon 2/24/25** | **Fri 3/14/25** | |
| | Gathering Requirements | 8 days | Mon 2/24/25 | Wed 3/5/25 | |
| | Fesability study | 3 days | Thu 3/6/25 | Mon 3/10/25 | 3 |
| | Assign the tasks | 4 days | Tue 3/11/25 | Fri 3/14/25 | 4 |
| | **Requierments Analysis** | **12 days?** | **Mon 3/17/25** | **Tue 4/1/25** | **5** |
| | Identifying issues | 2 days | Mon 3/17/25 | Tue 3/18/25 | 5 |
| | Gathering Data | 8 days? | Wed 3/19/25 | Fri 3/28/25 | 7 |
| | Find potential solutions | 2 days? | Mon 3/31/25 | Tue 4/1/25 | 8 |
| | **Design** | **7 days** | **Wed 4/2/25** | **Thu 4/10/25** | **9** |
| | Choose the right solution | 3 days | Wed 4/2/25 | Fri 4/4/25 | 9 |
| | design solution blueprints | 4 days | Mon 4/7/25 | Thu 4/10/25 | 11 |
| | **Implementation** | **36 days** | **Fri 4/11/25** | **Fri 5/30/25** | **12** |
| | Coding | 27 days | Fri 4/11/25 | Mon 5/19/25 | 12 |
| | Testing | 9 days | Tue 5/20/25 | Fri 5/30/25 | 14 |
| | **Deployment** | **3 days** | **Mon 6/2/25** | **Wed 6/4/25** | **15** |
| | Software is ready for end-users | 3 days | Mon 6/2/25 | Wed 6/4/25 | 15 |



*Figure 1-1: Gantt Chart*

# Chapter 2

# LITEATURE REVIEW

## 2.1 Introduction

A literature review is a fundamental step in any project. Its purpose is to gather, analyze, and interpret information related to the project topic by comparing similar platforms and applications. While there may be other tools and websites that address related goals, our project stands out due to its unique features and innovative characteristics that differentiate it from existing solutions.

## 2.2 External services

**WebWiz** relies on external services to extend its core functionality and enhance the overall learning experience. These services are integrated to provide intelligent feedback, manage media assets, and support seamless performance. The main external services used in the system include an advanced AI integration for evaluating learner code and cloud storage solutions for securely handling task-related resources.

### 2.2.1 *WebWiz AI Integration*

The AI model used in WebWiz is a specialized implementation of Gemini 2.0 Flash, a cutting-edge multimodal AI technology renowned for its high efficiency in processing complex contextual inputs. This model is purpose-built to enhance the learning experience on WebWiz by providing immediate, intelligent, and structured feedback based on learner-submitted solutions.

The AI operates in a focused role as a code evaluator and advisor, where it analyzes the learner's submitted code in relation to a specific task description, the corresponding design (provided as an image), and configuration details (in a JSON format). The prompt sent to the AI is carefully structured to ensure the response adheres to a predefined JSON schema. This response is then parsed and displayed to the learner through an interactive UI component.

By leveraging the capabilities of Gemini 2.0 Flash, WebWiz delivers precise evaluations, suggesting improvements and highlighting best practices that align with both coding standards

and the intended design outcome. This allows learners to receive real-time, actionable guidance that fosters deeper understanding and higher code quality.

**Key Features and Capabilities of the WebWiz AI Integration**

**Specialized AI Model:**

Built on the Gemini 2.0 Flash API, the model excels at understanding code, tasks, and visual designs simultaneously, enabling accurate and contextual analysis.

**Multimodal Input Processing:**

The AI evaluates task descriptions, code submited, and supporting JSON data to deliver comprehensive and relevant responses.

**Structured Output Format:**

All responses follow a consistent JSON schema, allowing seamless integration with the frontend UI for clear and user-friendly presentation.

By combining the advanced reasoning capabilities of Gemini 2.0 Flash with WebWiz's targeted learning goals, the AI model empowers learner's to grow their development skills with clarity, confidence, and immediate support.

### 2.2.2 Cloud Storage Services

The WebWiz system utilizes cloud storage services to securely store task-related images and data. This ensures that all media files are safely backed up and easily accessible, without putting extra load on the local server. Leveraging cloud storage enhances WebWiz performance, reliability, and scalability, providing learners with a seamless experience when uploading and accessing images associated with their coding tasks.

## 2.3 Related Works of WebWiz website

Many platforms are designed in this field such as:

**Skill Development Platforms:**

1. **PlayCode**

   **PlayCode** is an interactive browser-based coding platform that lets users write and execute code instantly without installing a local environment. It is particularly useful for

both beginners and developers who want to test and experiment with code quickly. The platform also includes educational content covering HTML, CSS, JavaScript, and React.[2]


*Figure 2-1: PlayCode - Image 1*

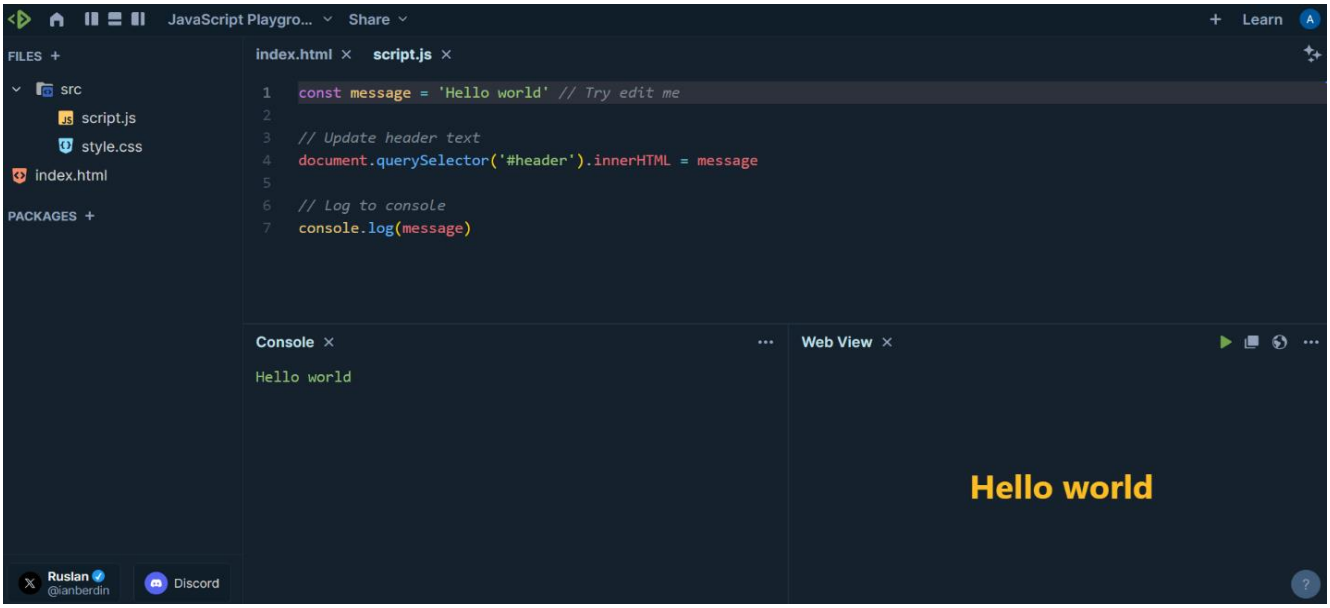



*Figure 2-2: PlayCode - Image 2*

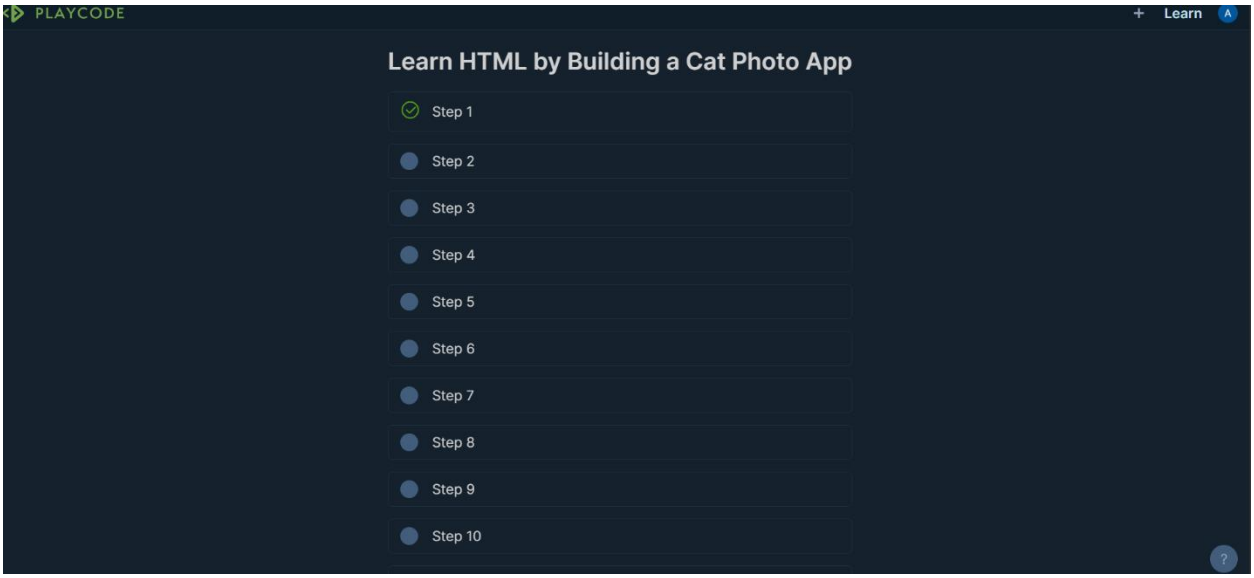



*Figure2-3:PlayCode - Image 3*

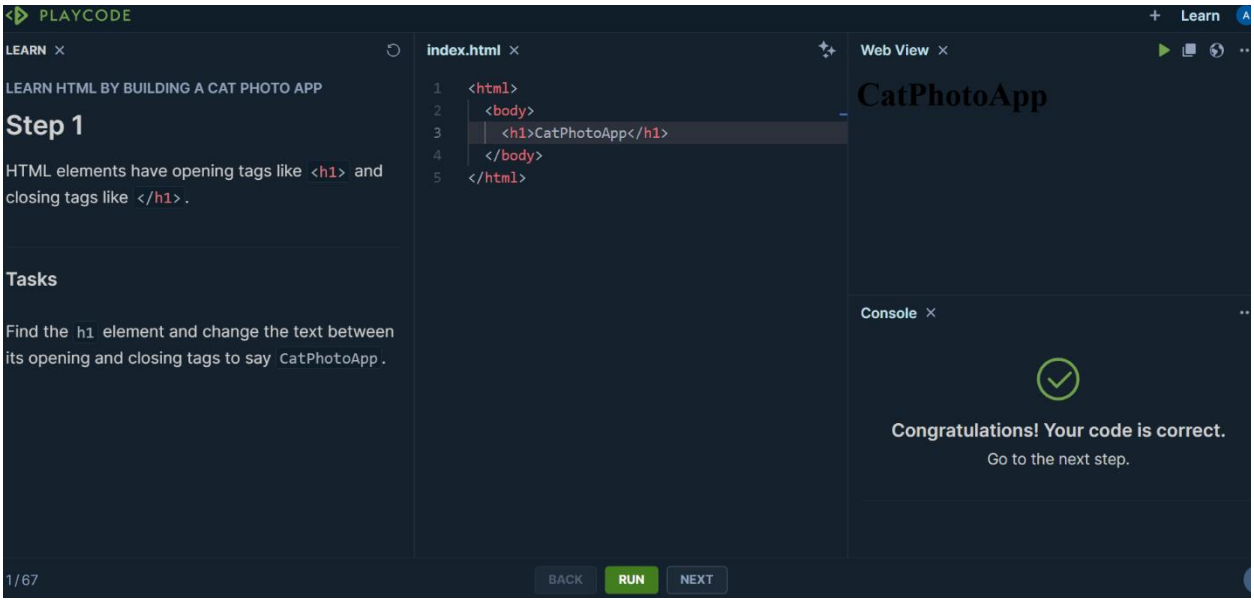



*Figure2-4:PlayCode - Image 4*

## 2. Frontend Mentor

**Frontend Mentor** is a platform that helps developers enhance their front-end development skills through practical, real-world challenges. It offers design-to-code projects that allow learners to build websites from provided UI designs while following structured learning paths.[3]

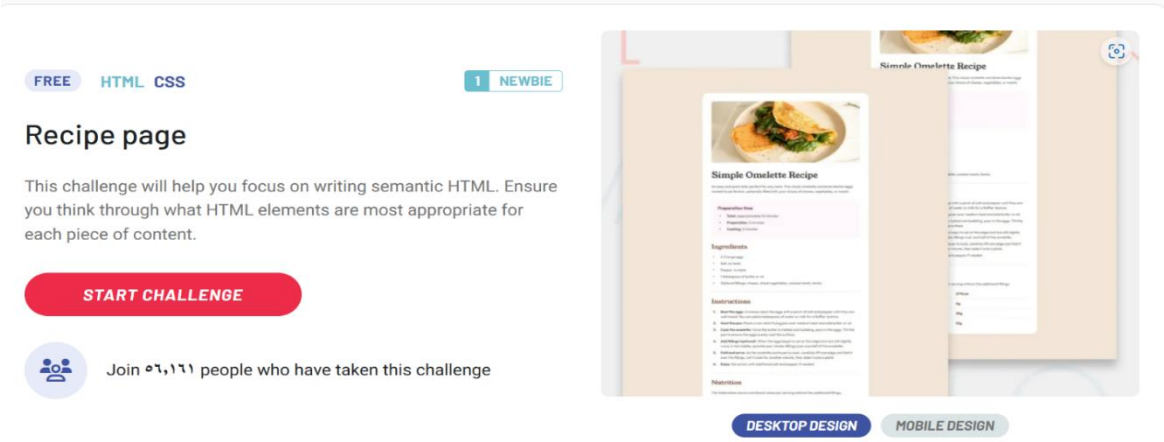



*Figure 2-5: Frontend Mentor – Image1*

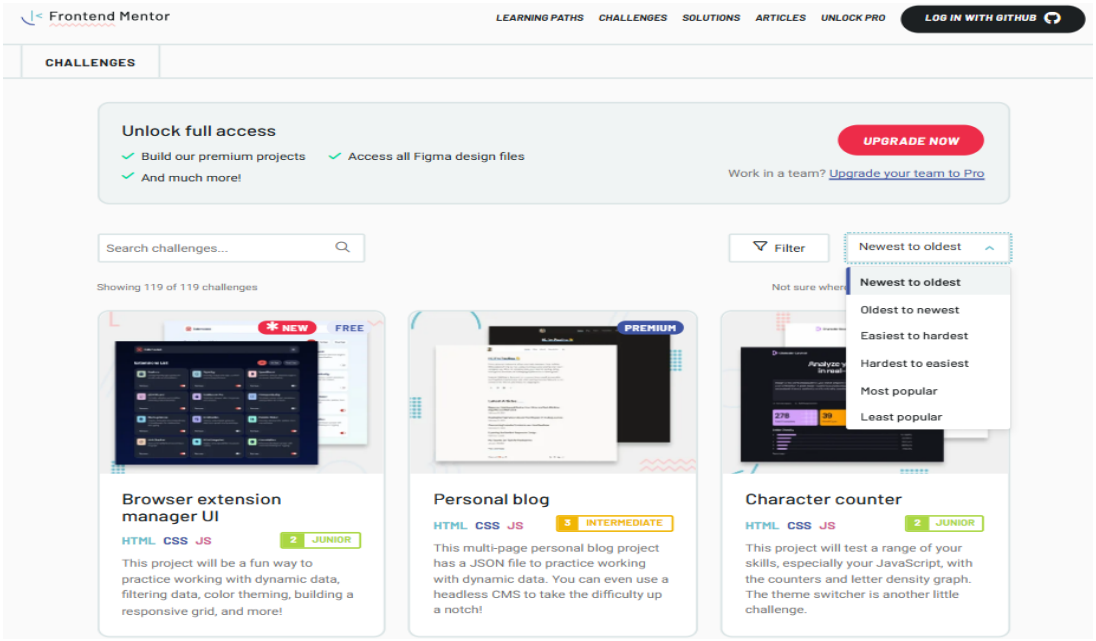



*Figure 2-6: Frontend Mentor – Image2*

## 3. LeetCode

**LeetCode** is an online platform designed to enhance programming skills and solve technical problems. The site primarily focuses on preparing for technical job interviews, offering challenges in areas such as programming, data structures and algorithms.[4]

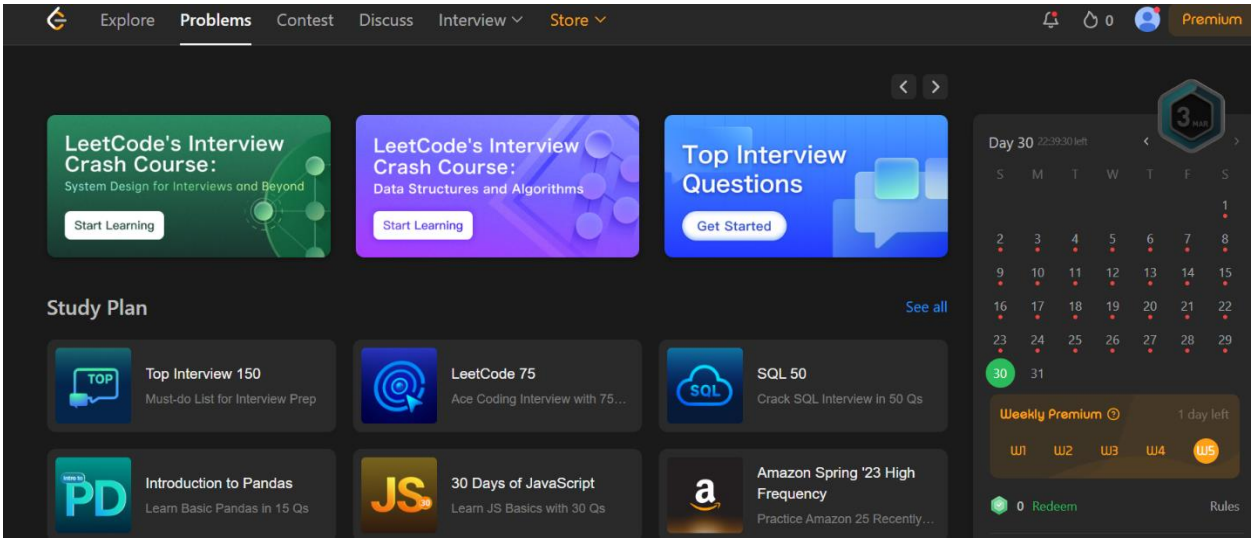



*Figure 2-7: LeetCode– Image1*

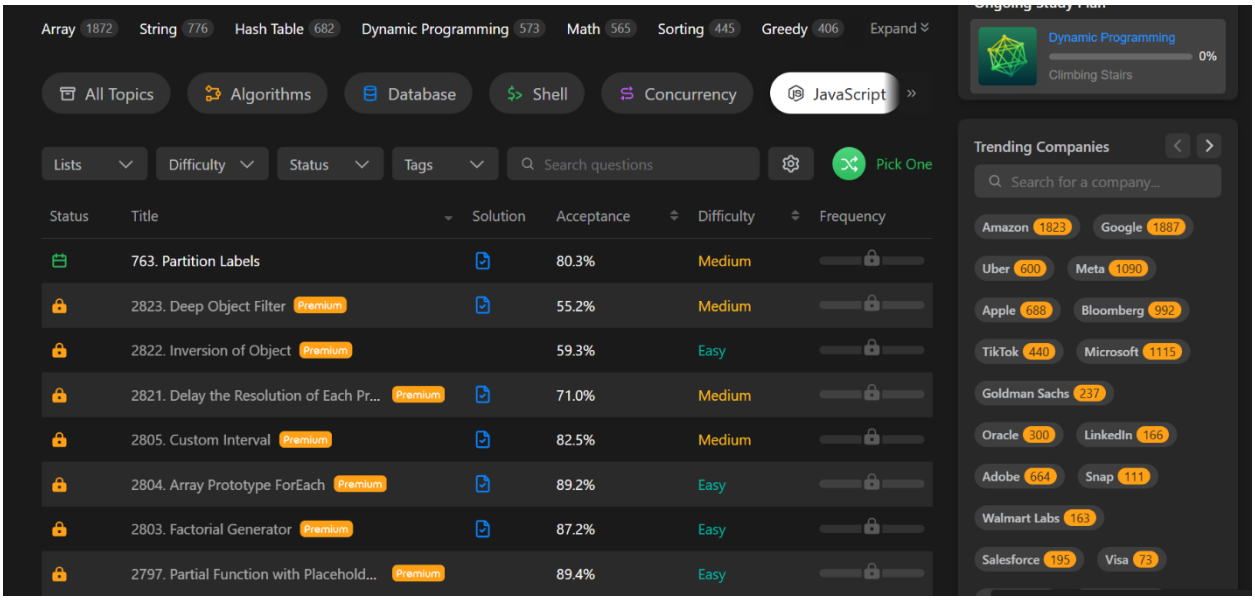



*Figure 2-8: LeetCode– Image2*


*Figure 2-9: LeetCode– Image3*

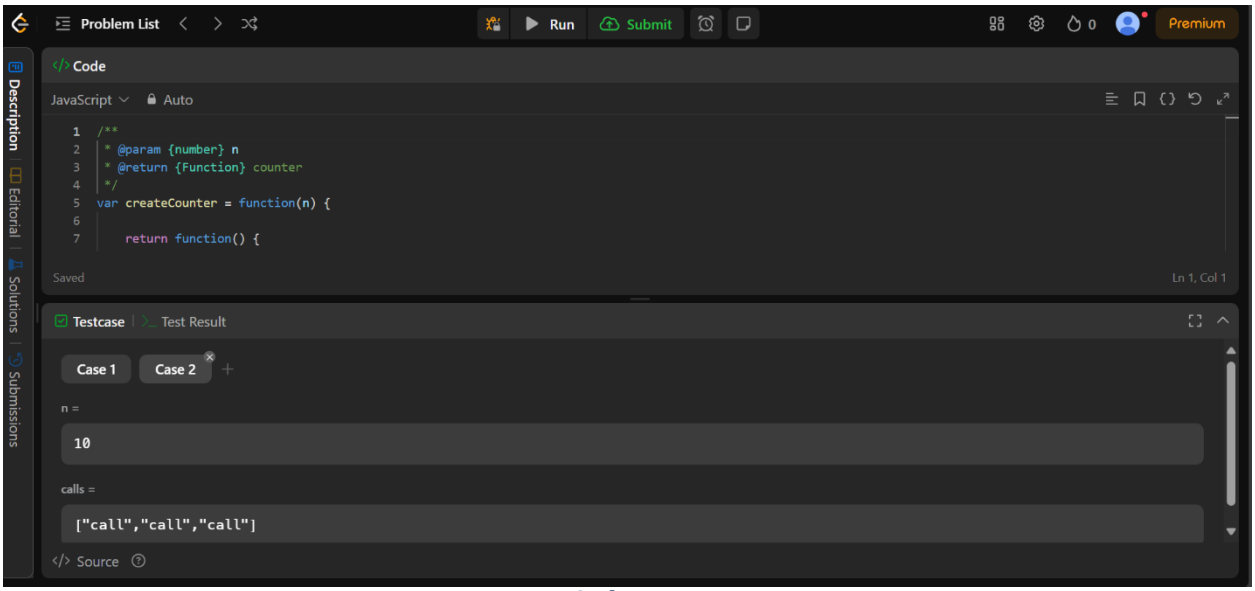



*Figure 2-10: LeetCode– Image4*

| *Feature* | *Solution validation with AI* | *Real-time code preview* | *Structured learning roadmap* |
|---|---|---|---|
| *PlayCode* | ✗ | ✓ | ✗ |
| *Frontend Mentor* | ✗ | ✗ | ✗ |
| *LeetCode* | ✗ | ✗ | ✗ |
| *WebWiz* | ✓ | ✓ | ✓ |

*Table **Error! No text of specified style in document.**- Comparison Table*

# Chapter 3

# METHODOLOGY

The **WebWiz** project will adopt an agile methodology for software development. This approach emphasizes flexibility, continuous feedback, and iterative progress, making it ideal for a dynamic and user-focused project like **WebWiz**.

## 3.1 Agile Methodology

The agile methodology is a flexible and iterative approach to software development. It breaks down the development process into small, manageable cycles or sprints. Each sprint delivers a working increment of the product, enabling continuous refinement based on user feedback. Agile focuses on collaboration, adaptability, and delivering value quickly, ensuring that the software evolves in response to changing requirements and user needs.

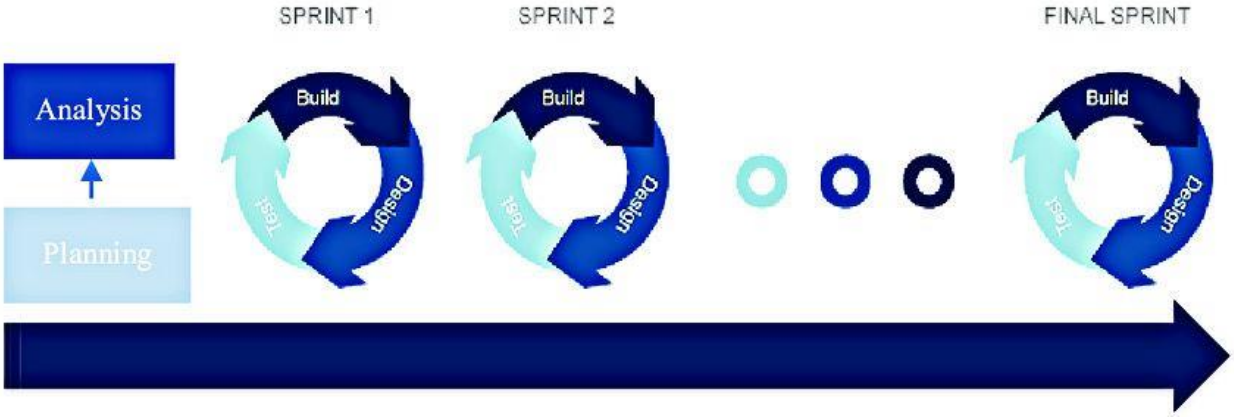



*Figure **Error! No text of specified style in document.**-1: Agile methodology*

### *3.1.1 WebWiz Analysis*

In the initial phase, we gathered detailed software requirements through discussions with potential users, such as IT students. This phase focused on defining the core functionalities of WebWiz, including user authentication and authorization, task delivery, an interactive live code editor, solution evaluation, progress tracking, code saving, and a structured learning roadmap. By understanding user needs, we established a foundation for the system that aligns with both business goals and end-user expectations.

### *3.1.2 WebWiz System Design*

During the system design phase, the product is divided into smaller, modular components for easier development. This includes both logical design, such as defining the system architecture, and physical design, detailing the software components. For **WebWiz**, our team plans to create UML diagrams, including  use case and sequence diagrams. Additionally, we plan to use the Next.js framework for the front end and the .NET framework for the back end to ensure a scalable and maintainable platform structure. We also aim to implement Clean Architecture, adhering to the SOLID principles (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, and Dependency Inversion) to ensure code maintainability, scalability, and flexibility for future development.

### *3.1.3 WebWiz Implementation and Incremental Testing*

The implementation of **WebWiz** is guided by the foundational requirements and modular design established during the analysis and design phases. The development process follows an incremental approach, where the system is built in small, manageable components. Each feature is coded and tested in isolation to ensure stability and functionality before being integrated into the larger system.

In the initial increments, the focus will be on implementing the core features identified during the requirements analysis:

- **User Authentication:** Ensuring secure access for all users.
- **Task Delivery:** Delivering tasks categorized by the learner's skill level.
- **Live Code Editor:** Displaying the task design, description, code editor, output screen, and a submission button.

- **Solution Evaluation:** Utilizing an external AI API to check if the submitted solution matches the required design.
- **Progress and Code Saving:** Saving user progress and written code for each task.
- **Structured Learning Roadmap:** Providing a structured roadmap with organized tasks to facilitate an easier and more ordered learning process.

The implementation process will strictly adhere to the Clean Architecture principles defined in the design phase. This approach ensures maintainable, scalable, and flexible development, leveraging the Next.js framework for the frontend and .NET for the backend.

**Testing Strategy**:

1. **Unit Testing**: Verifying the functionality of individual components, such as the authentication module and data collection mechanisms.
2. **Integration Testing**: Ensuring seamless interaction between the front-end, back-end, and AI-based features, such as solution correction and providing the feedback.
3. **Adherence to Quality Standards**: By implementing rigorous testing processes, we aim to identify and resolve issues early, ensuring a stable and reliable system that aligns with user expectations and business goals.

This phased implementation strategy ensures that **WebWiz** evolves into a robust, user-focused platform that meets the dynamic demands of learning web development.

## 3.2 Advantages of Agile Methodology

**Flexibility**: Agile allows for frequent changes based on user feedback and evolving requirements.

**Quick Delivery**: Working software is delivered early and continuously throughout the development cycle.

**Risk Management**: Frequent testing and iterations help identify and resolve issues promptly.

**Improved Collaboration**: Continuous interaction between the development team and stakeholders ensures alignment with project goals.

## 3.3 Disadvantages of Agile Methodology

**Potential Scope Creep**: Continuous changes can lead to an ever-expanding project scope.

**Requires Strong Team Coordination**: Agile relies heavily on effective communication and collaboration.

**Frequent Testing**: Each sprint requires rigorous testing, which can increase time and resource consumption.

# Chapter 4

# ANALYSIS AND DESIGN

## 4.1 Analysis

The analysis phase is the initial stage of the Software Development Life Cycle (SDLC),where the system's requirements are thoroughly understood and defined. In this phase, the necessary information is gathered to accurately document the system's requirements, ensuring that the system under development meets the needs and expectations of the stakeholders.

### *4.1.1Requirements Determination*

The requirements for a system are the descriptions of what the system should do the services that it provides and the constraints on its operation. These requirements reflect the needs of customers for a system that serves a certain purpose.
Carrying out, analyzing, documenting and checking these services and constraints is called requirements engineering (RE).***[5]***

#### *4.1.1.1 Questionnaire*

**WebWiz** Need Assessment Questionnaire:

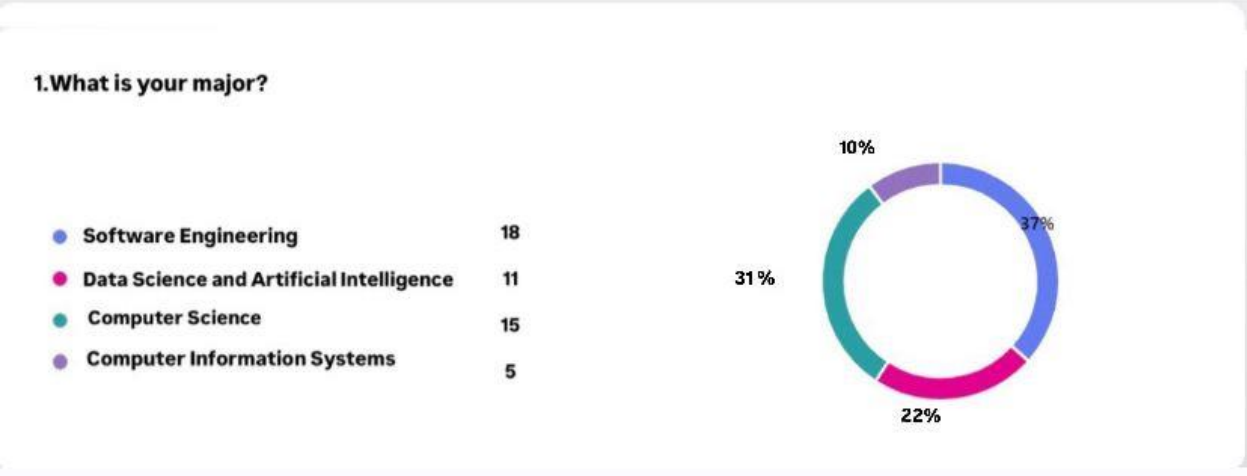



*Figure 4-1 WebWiz Need Assessment Question*


*Figure 4-2 WebWiz Need Assessment Question*

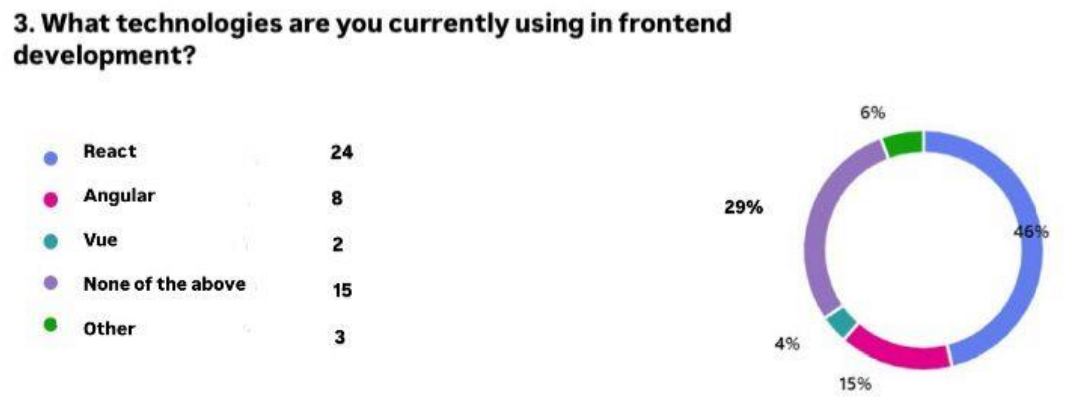



*Figure 4-3 WebWiz Need Assessment Question*

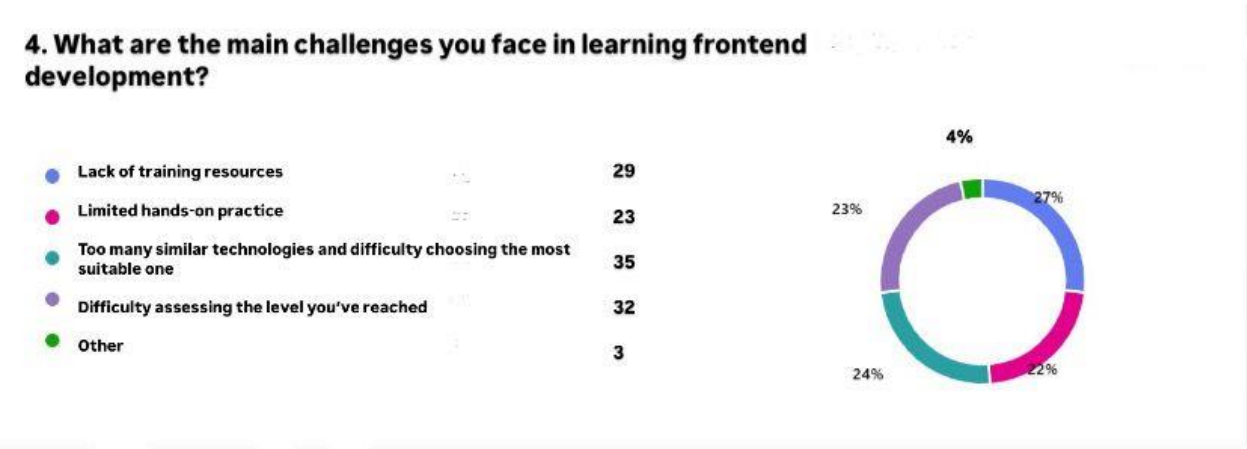



*Figure 4-4 WebWiz Need Assessment Question*

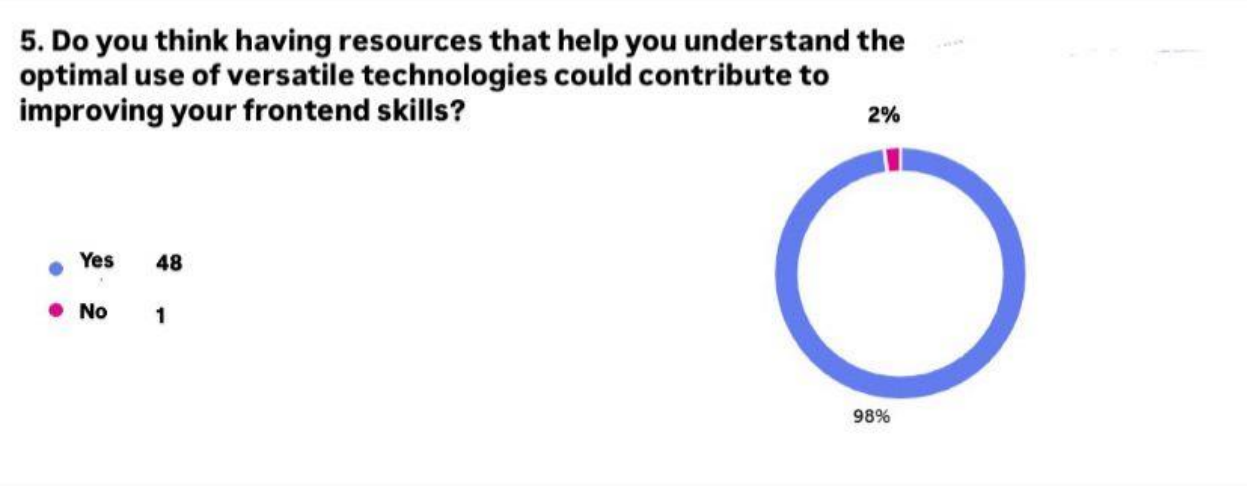



*Figure 4-5 WebWiz Need Assessment Question*

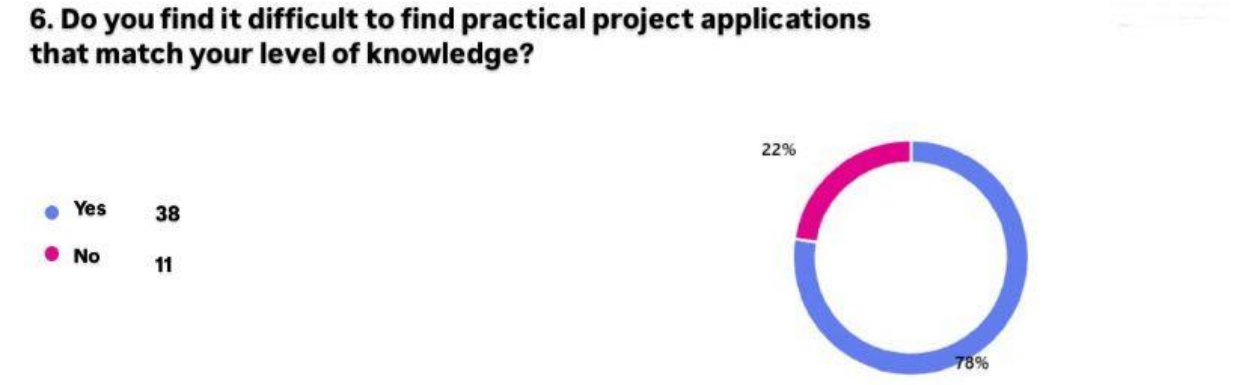



*Figure 4-6 WebWiz Need Assessment Question*

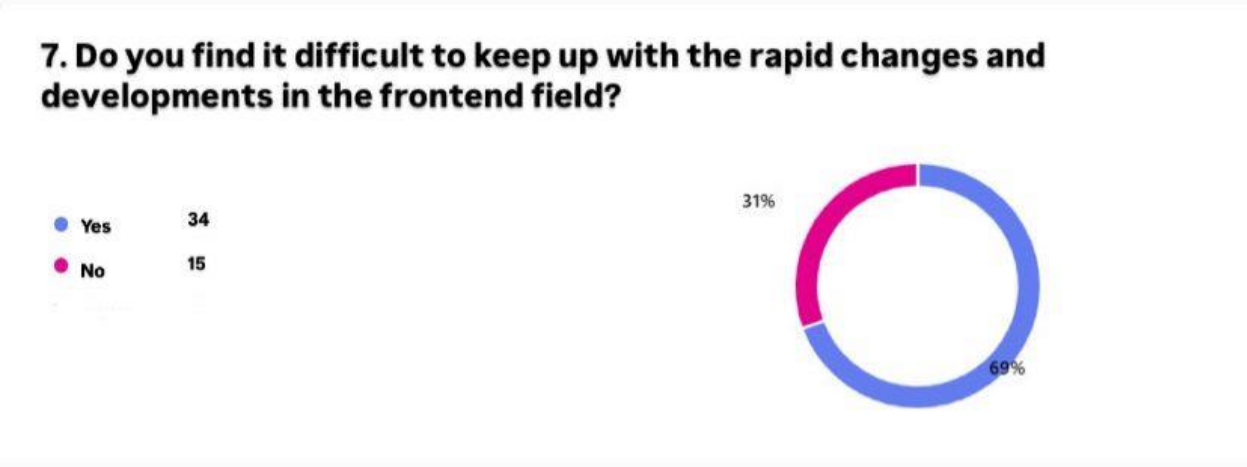



*Figure 4-7 WebWiz Need Assessment Question*

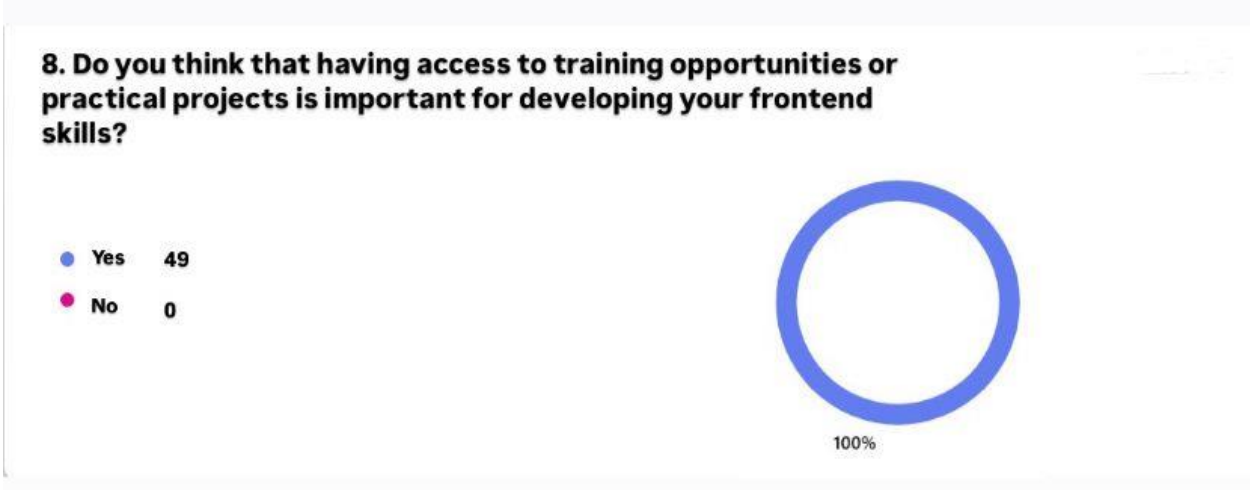



*Figure 4-8 WebWiz Need Assessment Question*

### *4.1.2 Software Requirements Specification (SRS):*

A software requirements specification (SRS) is a detailed description of a software system to be developed, including its functional and non-functional requirements. The SRS is developed based on the agreement between customers and contractors. It may include use cases describing user interaction with the software system. The SRS outlines all requirements necessary for project development. Understanding the software system clearly is essential for successful implementation. This is achieved through continuous communication with customers to gather complete and accurate requirements.**[6]**

#### *4.1.2.1 User Requirements*

When the learner enters the website, the first page they encounter is the Home page, which provides a welcoming interface for easy navigation throughout the platform. From this page, the learner can log into their account if they already have one or create a new account via the Register page. Learners can also use Google and GitHub accounts to log in to the site. They can also access the Tasks page, Structured Learning Roadmap, Leaderboard and Code Playgrounds from the Home page.

The system identifies whether the learner has permission to create a task. If the learner has this permission, a button appears on the Home page, granting access to the Create Task page. This page allows them to create new tasks or delet existing ones. If the learner lacks this permission, the option is hidden.

One of the key features of the learner's experience is when they enter the Live Code Editor, where they will find a Task button to access task details. The learner can directly write the required code in the code editor within the Live Code Editor. However, access to this page requires the learner to be logged in.

The Live Code Editor also includes an Output section, which displays the results of the written code in real time, providing the learner with an interactive and realistic experience.

Once the learner has completed a task, they can click the Submit button to send their solution for review and receive feedback.

The learner can also navigate to the Tasks page, which contains a collection of tasks designed to enhance their programming skills. These tasks are categorized by programming languages such as HTML, CSS, JavaScript, and React, and are further divided by difficulty levels: Beginner, Intermediate, Advanced, and Wizard. The learner can easily browse through these tasks and select those that align with their current skill level. Additionally, they can track the status of each task, whether it is completed or not yet started.

The learner can navigate to the Structured Learning Roadmap selection page through the Start Learning button, where they can choose between “Frontend Fundamentals” or “React Development.” Each roadmap provides a set of sequential tasks. The learner must solve the first task to unlock the second one, and so on.

#### *4.1.2.2 System Requirements*

Software system requirements are often classified as functional or non-functional requirements:

##### *4.1.2.2.1 Functional Requirements*

**1- Authentication and Account Management Section:**

- **Login:**
    - The system shall allow users to log in using email and password.
    - The system shall allow users to log in using Google OAuth.
    - The system shall allow users to log in using GitHub OAuth.

- **Sign up:**
  The system shall allow users to register using email and password.

- **User Role Identification:**
    - The system shall identify the learner's permission based on their email address.
    - The system shall check whether the user has permission to create a task.
    - If the user has such permission, a "Create Task" button will appear on the Home page, whereas learners without permission do not see this button.

**2-. Tasks preview section:**

- **The system allows learners to access programming tasks:**
    - The system shall categorize tasks based on the programming languages.
    - The system shall classify tasks according to difficulty levels.
    - The system shall display the status of each task. (e.g., completed, or not started).

**3- Structured learning roadmap section:**

- The website shall provide a learning roadmap that has logically ordered tasks that help the learner get into frontend development and the JavaScript framework react .

**4- Task Creation section:**

- The "Create Task" button shall be available only to authorized learners, and it shall provide access to the task creation page.

- The system shall allow learners with permission to create and delet tasks.

**5- Live code editor section:**

- **Access Control:**

  The system shall restrict access to the live code editor page to logged-in learners only.

- **Code Editor:**

  The live code editor shall allow learners to write and modify source code.

- **Task Details Button:**

  The live code editor shall include a button that, when clicked, displays the task details.

- **Real-Time Output Display**
  The live code editor shall provide an output section that displays the result of the executed code in real time during development.

- **Submit Button**

  The live code editor shall include a "Submit" button that allows learners to submit their solution for evaluation with the help of AI.

**6-Submission and AI**

- **AI-Powered Code Submission**
  The "Submit" button shall send the written code for evaluation via an AI-powered verification process.
- **AI-Powered**
  The system shall display the results of the evaluation for the submitted task, including a score or points that reflect the quality, correctness, and efficiency of the submitted code.

- **Code Quality Guidance**
  The AI system shall guide the learner towards writing cleaner and more efficient code by providing feedback

#### *4.1.2.2.2 Non-Functional Requirements*

1. **Performance**: The system must process user inputs and display code output live. The code should be automatically executed, and the output displayed within 1.5 seconds of a pause in user input.
2. **Usability**: The system has an intuitive and user-friendly interface that can be easily navigated by users with varying levels of technical proficiency.
3. **Reliability**: The application has a high responsiveness under different workloads.
4. **Interoperability**: The system must be able to integrate with external services such as Google and GitHub to allow seamless login and interaction with user accounts on these platforms, integration with AI (Gemini AI) through standard APIs, and integration with cloud services for saving images .
5. **Availability**: The website available for use all time 24/7.

## *4.2 Design*

Software design is a creative activity in which you identify software components and their relationships, based on a customer's requirements.**[5]**

### *4.2.1 Logical Design*

A logical design is a conceptual, abstract design. You do not deal with the physical implementation details yet; you deal only with defining the types of information that you need.**[7]**

#### *4.2.1.1 Use Case Diagram*

When the website opens, the learner is presented with the Home Page, which enables smooth navigation between the different sections of the website. The user has the option to register a new account either by using third-party authentication services such as Google or GitHub or by manually completing a registration form. After successful registration, the user can log in by entering their credentials. If the user already has an existing account, they can simply use the Sign in option to access their personal account without needing to register again.

When a user registers, the system verifies whether they are using a professional email address. If the email is valid and meets the professional criteria, the user is granted permission to create new tasks.

Learners who register with a professional email address are granted special privileges within the system. These include the ability to create and delet tasks, upload design images related to each task, and provide an optimal solution. The design images are stored securely in cloud storage, while the submitted optimal solutions are sent to the integrated AI for evaluation.

Regardless of their permission to create tasks, all learners have access to the Tasks Page. This page presents a diverse collection of programming tasks, organized by programming language and difficulty level. Built-in filtering tools allow learners to efficiently browse and locate tasks that align with their current skill level or areas of interest.

When a learner selects a task, they are automatically redirected to the Live Code Editor. This integrated environment offers a clean and user-friendly workspace, where learners can view detailed task information including descriptions and associated images write code directly in the editor, and see real-time output as they type. To prevent data loss, the system continuously auto-saves the learner's progress.

Once a learner completes a task, they can submit their solution by clicking the "Submit" button. This action initiates an AI-powered evaluation process. The system analyzes the submitted code and then evaluates the solution for the learner, helping them improve their skills and understanding.

The learner can enroll in a structured learning roadmap that provides a sequence of tasks. Each task on the roadmap is locked by default, except for the first one, which is unlocked. As the learner solves each task, the next task is automatically unlocked, continuing this process until the entire roadmap is completed.

While solving a task, the learner's progress is automatically tracked. Each successfully completed task is recorded in the learner's profile. Additionally, every time a task is submitted, the score earned is added to the learner's total points. These accumulated points are displayed

on the learner’s profile and used to rank learners on the leaderboard, which showcases the top 10 learners with the highest total scores.

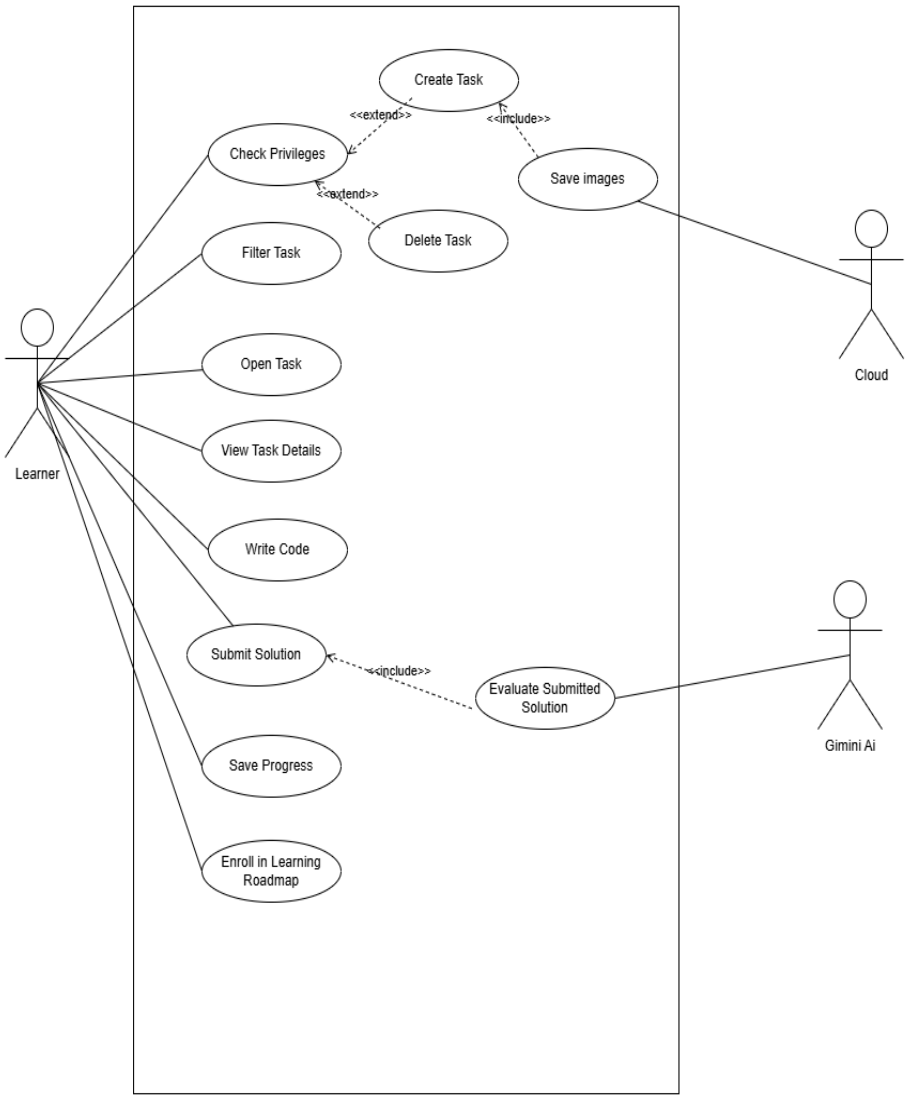



*Figure 4-9 Use Case*

#### *4.2.1.2 Sequence Diagrams*

In **WebWiz**, the sequence diagram follows two scenarios for any use case or service that any **WebWiz** user can perform and in the event of any error occurring in any of the servers, controllers, or repositories, the error handler middleware handles it and sends the appropriate response to the user. The two scenarios are as follow:

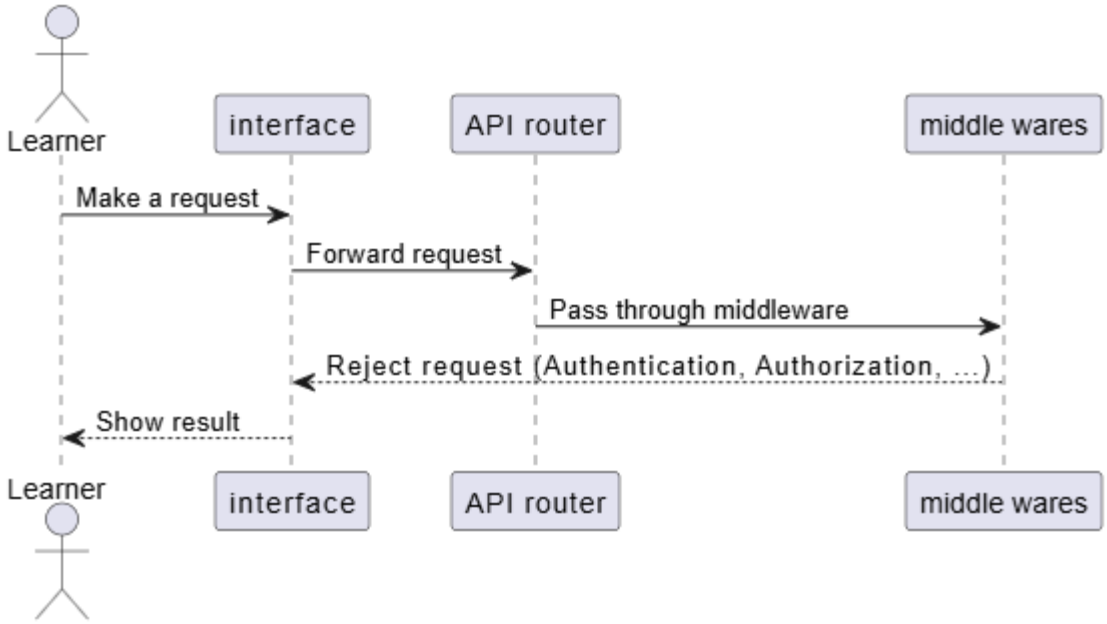



*Figure 4-10: Sequence middleware scenario*

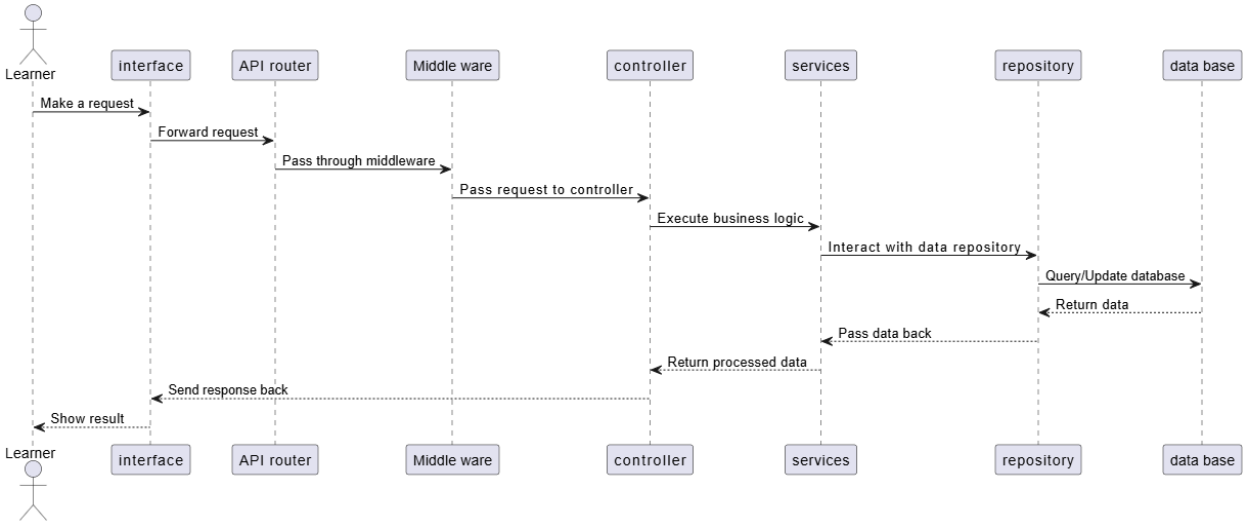



*Figure 4-11: General Sequence diagram scenario*

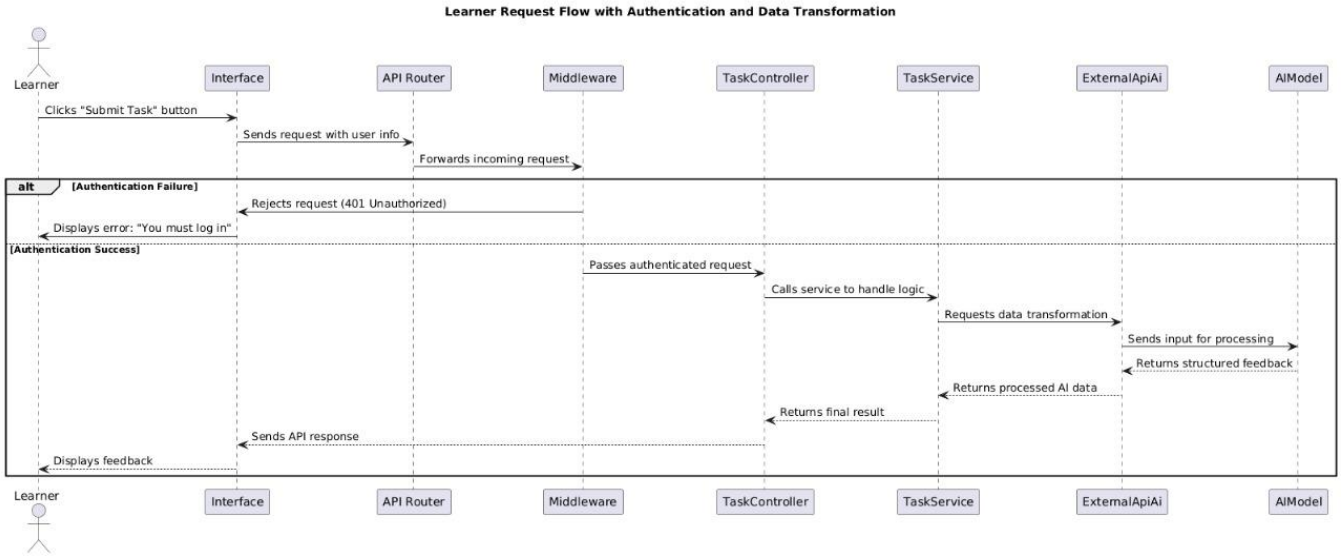



*Figure 4-12: Submit task Sequence*

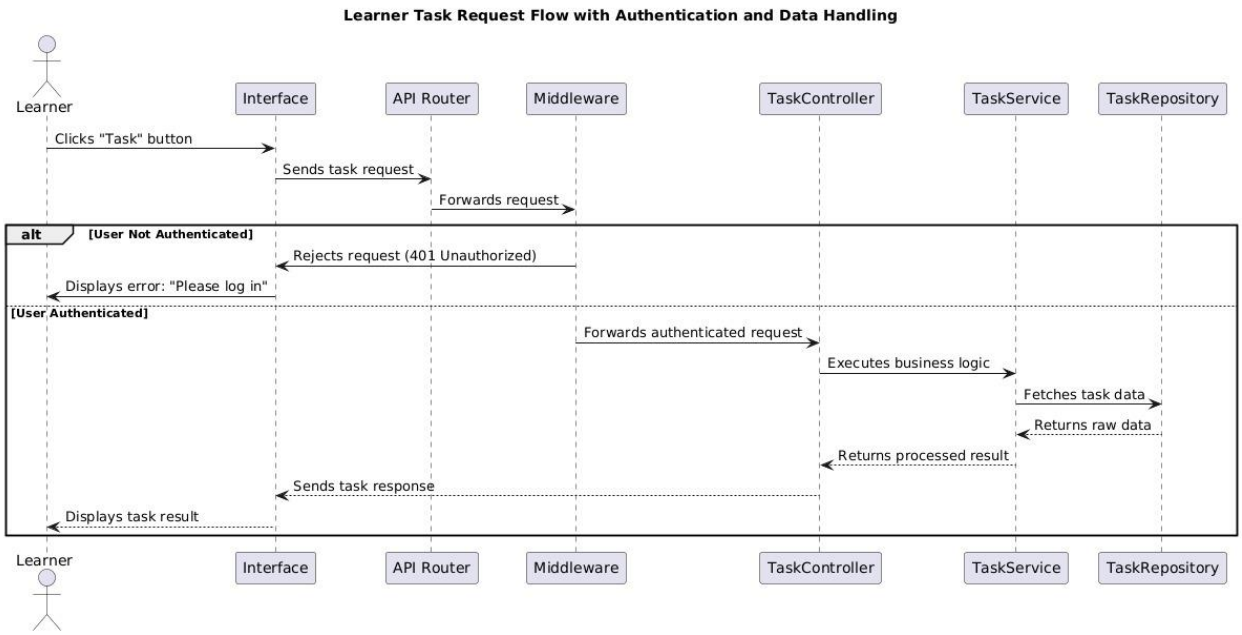



*Figure 4-13:Task Sequence*

### *4.2.1.3 Entity Relationship Diagram*

It is a data modelling tool that uses a high-level conceptual design to achieve the goal of describing data or depicting it in an abstract way and describing the relationship of Entities with each other and helps reduce errors and redundancies in the database, among. The main concepts in this model are entity, attribute, and relationship.

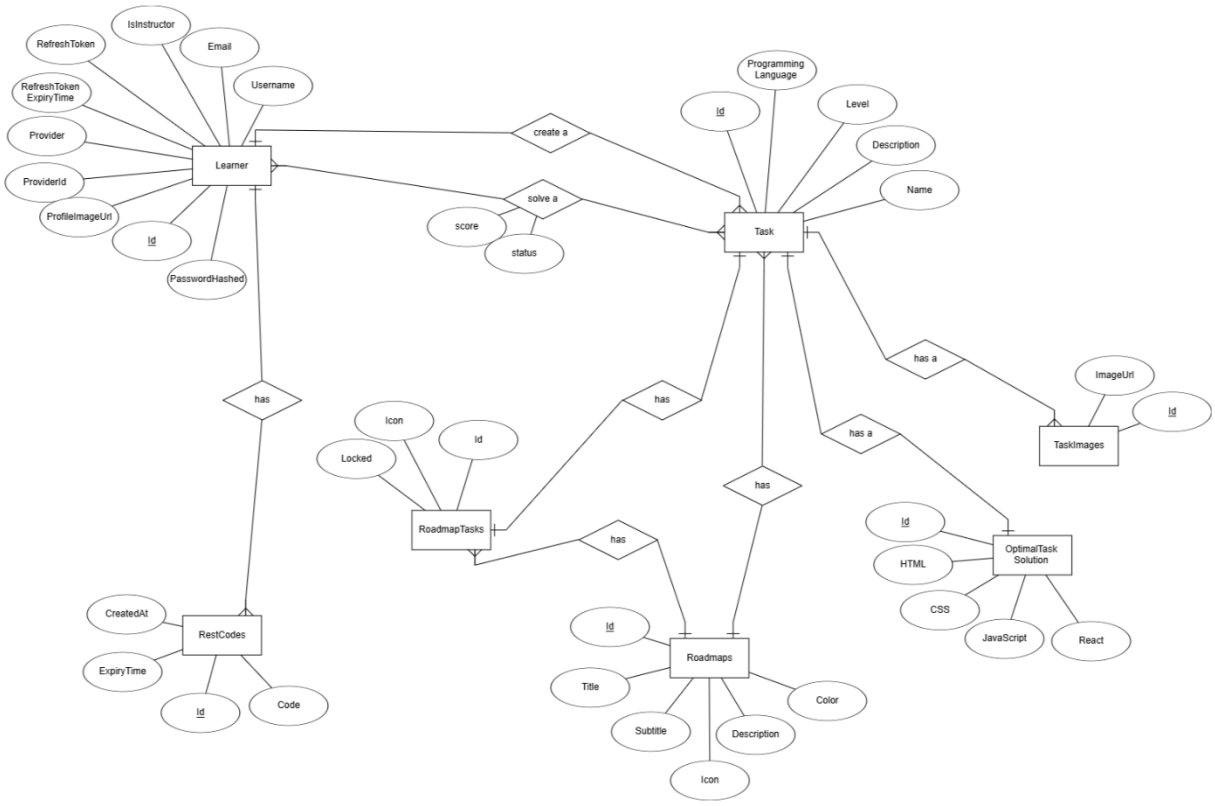



*Figure 4-14 ER-Diagram*

### *4.2.1.4 Database Schema*

A database schema defines the structure of a database using a formal language typically supported by a relational database management system. It serves as a blueprint that organizes data and outlines how the database is designed.


*Figure 4-15 Schema*

#### *4.2.1.5 WebWiz architecture*

- (Clean Architecture):

In the **WebWiz** platform, we adopted the Clean Architecture in our backend application to adhere to SOLID principles and implement high-quality, testable code. This architecture provides a clear separation of concerns, ensuring maintainability and scalability.

Key Layers of Clean Architecture

1. Domain Layer
   This is the core of the application, containing the business logic. It is independent of external changes, ensuring that modifications to other layers do not affect it. The domain represents the unchanging foundation of the system.
2. Application Layer
   This layer contains use cases and the main logic for executing application workflows. It relies on the domain layer and may include helper services. Changes here are more frequent than in the domain layer due to evolving business requirements.
3. Presentation Layer
   Responsible for presenting data and interacting with clients, such as through APIs. It serves as the interface between
4. the client and the application. This layer is prone to frequent changes, such as migrating to
5. a new framework or altering the delivery mechanism (e.g., from APIs to MVC).

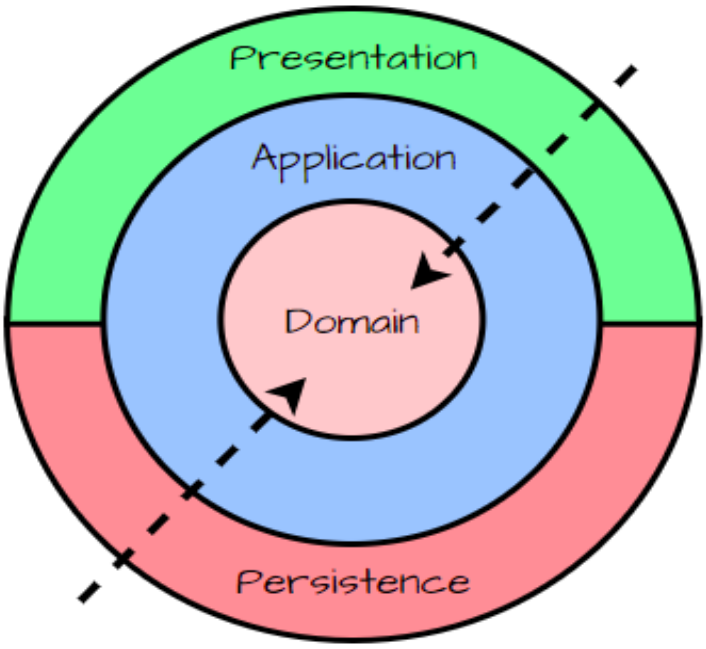



*Figure 4-16: Clean-Architecture*

6. Infrastructure Layer
   Manages external interactions like databases, logging systems, and email services. It is the most change-prone layer,

   designed to ensure updates do not ripple through the entire system.

**Dependency Rule**

In Clean Architecture, dependencies flow inward. Outer layers depend on inner layers, never the reverse. This design ensures that changes in volatile layers (e.g., infrastructure and presentation) do not impact the stable core layers (domain and application).

**Preserving SOLID Principles**

The architecture ensures compliance with SOLID principles:

1. **Single Responsibility Principle (SRP)**
   Each layer has a distinct responsibility: core business logic in the domain, use cases in the application, client interaction in the presentation, and external operations in the infrastructure.

2. **Open/Closed Principle (OCP)**
   Components are designed to be open for extension but closed for modification. For instance, new services can be added without altering existing domain logic.

3. **Liskov Substitution Principle (LSP)**
   Interfaces and abstract classes enable substitutability, ensuring components can evolve without breaking the system.

4. **Interface Segregation Principle (ISP)**
   Each layer exposes only the necessary interfaces, preventing unnecessary dependencies.

5. **Dependency Inversion Principle (DIP)**
   High-level modules depend on abstractions, not concrete implementations, ensuring flexibility and resilience to change.

By adhering to these principles, we maintain a robust, scalable, and testable system architecture.

### *4.2.2 Physical Design*

Meanwhile, physical design deals with the process of converting the logical design into a more technical specification of the system development. In designing the physical part of the system, all diagrams that were produced in the logical design were turned into a structured systems design. During physical design, the researcher determined which programming language and database system will be used as well as the determination of which UI design, hardware platform, operating system and network environment the system will run under. The specifications are portrayed in table 4.1 Physical design

| **Purpose** | Hardware – Software Specifications |
|---|---|
| UI design | The website was designed using Next.js |
| Functionality and backend | Application functions and API are created using the ASP .NET core web API<br>As for the database SQL server was used |
| Environment and write code | Visual studio code<br><br>Visual studio community 2022 |
| Operating systems | Desktop: Windows, Linux, Mac<br>Mobile: Android, Ios, tablet |

*Table **Error! No text of specified style in document.**.1 Physical design*

# Chapter 5

# FINDINGS

## 5.1 *Introduction*

Many programming languages have been used in this project in addition to different work environments, and these languages, tools, environments will be clarified in this chapter in addition to reviewing the user interfaces in **WebWiz** Website.

The Challenge in Implementation

When implementing Clean Architecture, one major challenge arises in the interaction between the application layer and the infrastructure layer. Typically, an API call flows like this:

1. The router passes the request payload to the service.
2. The service processes the business logic.
3. The service writes or retrieves data from the database.

However, in Clean Architecture, the service (application layer) cannot directly interact with the database (infrastructure layer) due to dependency rules. This creates a problem: how do we allow the service to access the database while maintaining clean dependency boundaries?

The Solution: Using Interfaces and Dependency Injection

To solve this, we use an interface to abstract database interactions and follow a service-repository pattern. Here's how it works:

1. **Define an Interface**: The interface is placed in the domain (or application) layer to define the contract for database operations.
2. **Implement the Interface**: A concrete class (repository) in the infrastructure layer implements this interface to handle real database operations.
3. **Inject the Dependency**: The API layer creates an instance of the repository and injects it into the service via the constructor.

This ensures the application layer depends only on abstractions (the interface) and not on concrete implementations (the database).

**Example Code Implementation**

Below is an example of how this pattern is implemented:

```typescript
// * src/domain/IRepos/IUserRepo.ts
interface IUserRepo {
  create(user: IUser): IUser; // Defines the contract for the repository
}

// * src/infrastructure/repos/UserRepo.ts
class UserRepo implements IUserRepo {
  create(user: IUser): IUser {
    // ! Real implementation with a database call
    return {} as IUser; // Example return value
  }
}

// * src/application/service/UserService.ts
class UserService {
  private userRepo: IUserRepo;

  // * Injects the repository into the service
  constructor(userRepo: IUserRepo) {
    this.userRepo = userRepo;
  }

  // Example service method
  createUser(user: IUser): IUser {
    return this.userRepo.create(user);
  }
}

// * src/API/Controller/UserController.ts
// Instantiate the repository and inject it into the service
const userRepo = new UserRepo();
const userService = new UserService(userRepo);

// Now the service can handle requests while adhering to Clean Architecture principles
```

*Figure 5-1 Example Code Implementation*

### *5.1.1 Programming languages*

**JavaScript**

- Overview and Definition
  JavaScript is a high-level, interpreted programming language primarily used for making web pages interactive. It can be executed in web browsers (client-side) or on servers (using Node.js). It is an object-oriented and event-driven scripting language that supports dynamic and responsive web functionalities such as animations, form validation, and interactive interfaces.

**C#**

- Overview and Definition
  C# (pronounced "see sharp") is a general-purpose, high-level programming language that supports multiple programming styles. It includes features like static and strong typing, lexical scoping, and supports imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming.

### *5.1.2 Frameworks*

***Next.js***

- **Overview**
  Next.js is a web development framework built on top of React. It helps developers build fast and user-friendly websites and web apps. Next.js supports features like server-side rendering, static site generation, and API routes, making it easy to build both simple websites and complex full-stack applications. It also comes with built-in tools for routing, performance optimization, and deployment.

***ASP .NET Core***

- **Overview**
  ASP.NET Core is a modern, open-source web framework developed by Microsoft for building high-performance web applications and APIs. It is a part of the .NET platform and is designed to be fast, flexible, and cross-platform, meaning it can run on Windows, macOS, and Linux.

  ASP.NET Core makes it easy to build secure and scalable backend services for web and mobile apps. It supports features like routing, middleware, dependency injection, and model binding out of the box.

***Jest***

- **Overview**
  Jest is a JavaScript testing framework developed by Meta (formerly Facebook) that is widely used for testing applications built with JavaScript and its related ecosystems. It is a powerful, flexible, and easy-to-use framework that supports unit testing, integration testing, and end-to-end testing.

***xUnit***

- **Overview**
  xUnit is a popular open-source testing framework for .NET applications. It is designed to be simple, flexible, and efficient for writing unit tests and verifying that your application behaves as expected. xUnit is widely used in the .NET ecosystem and is the default testing framework for .NET projects created with modern templates.

### *5.1.3 technology & code environment*

***node.js***

- **Overview**
  Node.js is an open-source, cross-platform runtime environment built on Chrome's V8 JavaScript engine. It allows developers to run JavaScript code outside a browser, primarily for backend development. Node.js uses a single-threaded, event-driven architecture, making it ideal for

***GitHub***

- **Overview**
  GitHub is a widely-used web-based platform for version control and collaboration. Built on Git, it allows developers to manage, share, and collaborate on projects efficiently. GitHub supports a wide range of features including repository hosting, branch management, pull requests, and code review. It is also integral to many open-source projects and professional development workflows, making it a critical tool for modern software engineering.

***Visual Studio Code***

- **Overview**
  Visual Studio Code (VS Code) is a free, open-source integrated development environment (IDE) developed by Microsoft. It supports various programming languages and is available across multiple platforms, including Windows, macOS, and Linux. VS Code offers features such as debugging, syntax highlighting, intelligent code completion, and integrated Git control, making it a versatile tool for developers.

***Visual Studio Community***

- **Overview**
  Visual Studio Community is a free, full-featured development environment made by Microsoft. It's used to build applications for Windows, web, mobile, and the cloud. It supports many programming languages like C#, C++, Python, and more. Visual Studio Community includes helpful tools for writing, debugging, and testing code, and it's a great choice for students, individual developers, and small teams.

**Jira**

- **Overview**
  Jira, developed by Atlassian, is a versatile project management and issue-tracking tool widely used by teams to plan, track, and manage software development projects. It supports various methodologies, including Agile, Scrum, and Kanban, facilitating efficient workflow management and team collaboration.

***Vercel***

- **Overview**
  Vercel is a cloud platform optimized for deploying, hosting, and scaling modern web applications, specifically those built with frontend frameworks like Next.js, React, and others. It provides developers with a seamless workflow for building high-performance web projects, enabling fast deployment, scalability, and great user experiences.

### *5.1.4 Database Management System*

***SQL Server 2022***

- **Overview**
  SQL Server 2022 is the latest version of Microsoft's powerful relational database management system (RDBMS). It is designed to store, manage, and query large volumes of structured data efficiently. SQL Server 2022 offers high performance, advanced security features, and deep integration with Microsoft Azure for hybrid cloud capabilities.

### *5.1.5 Hardware*

- Laptop
- Smart Phone

### *5.1.6 Operating system*

- Windows
- Ubuntu

## 5.2 *WebWiz interfaces and their description*

The following fingers show the secerns of our website:

**5.2.1 logo**


*Figure 5-2* ***logo***

### 5.2.2 user login page

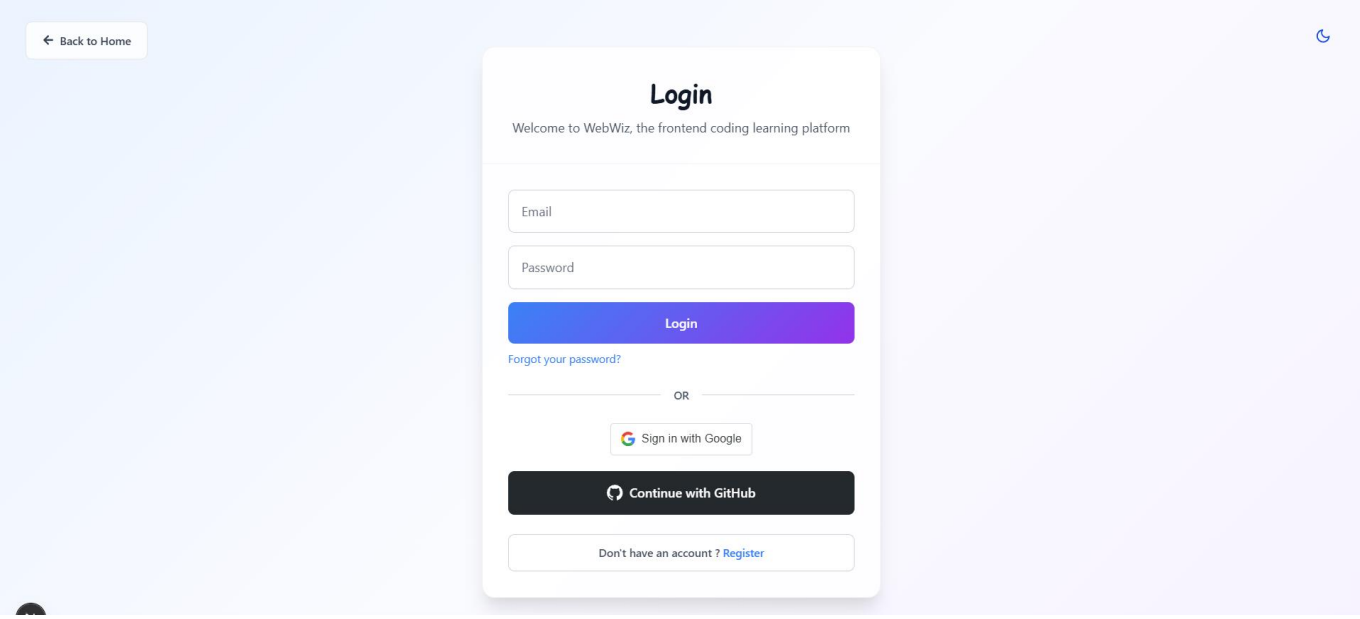



*Figure 5-3* ***user login page***

### 5.2.3 user register page

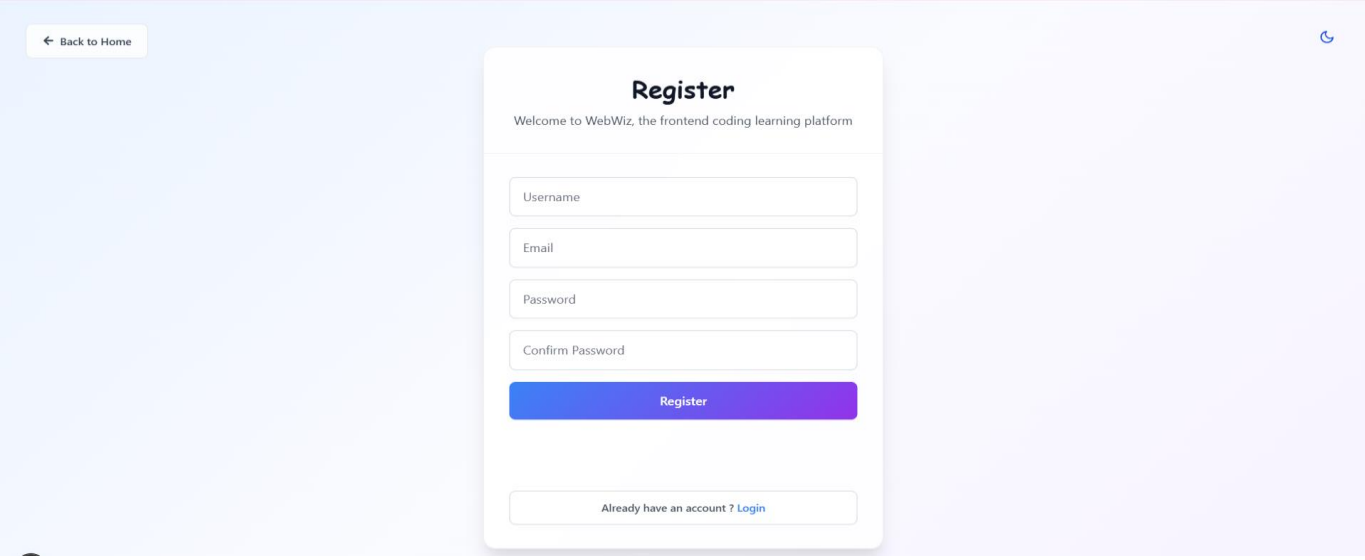



*Figure 5-4* ***user register page***

### 5.2.4 Task page

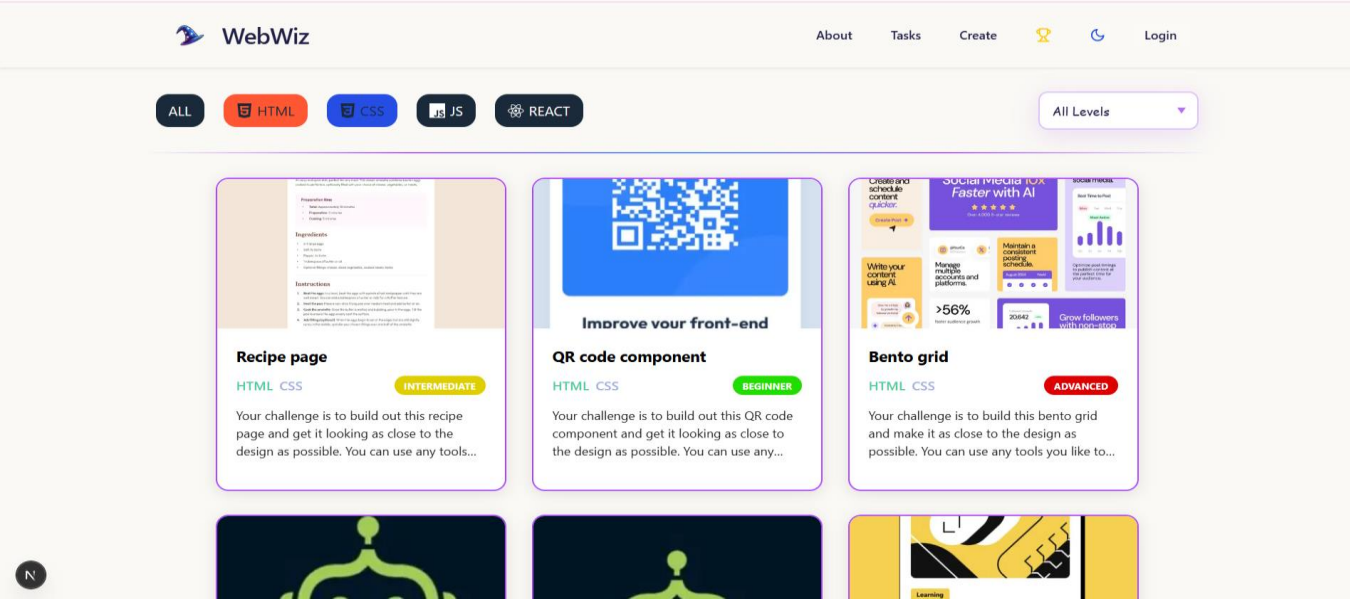



*Figure 5-5* ***Task page***

### 5.2.5 Landing page

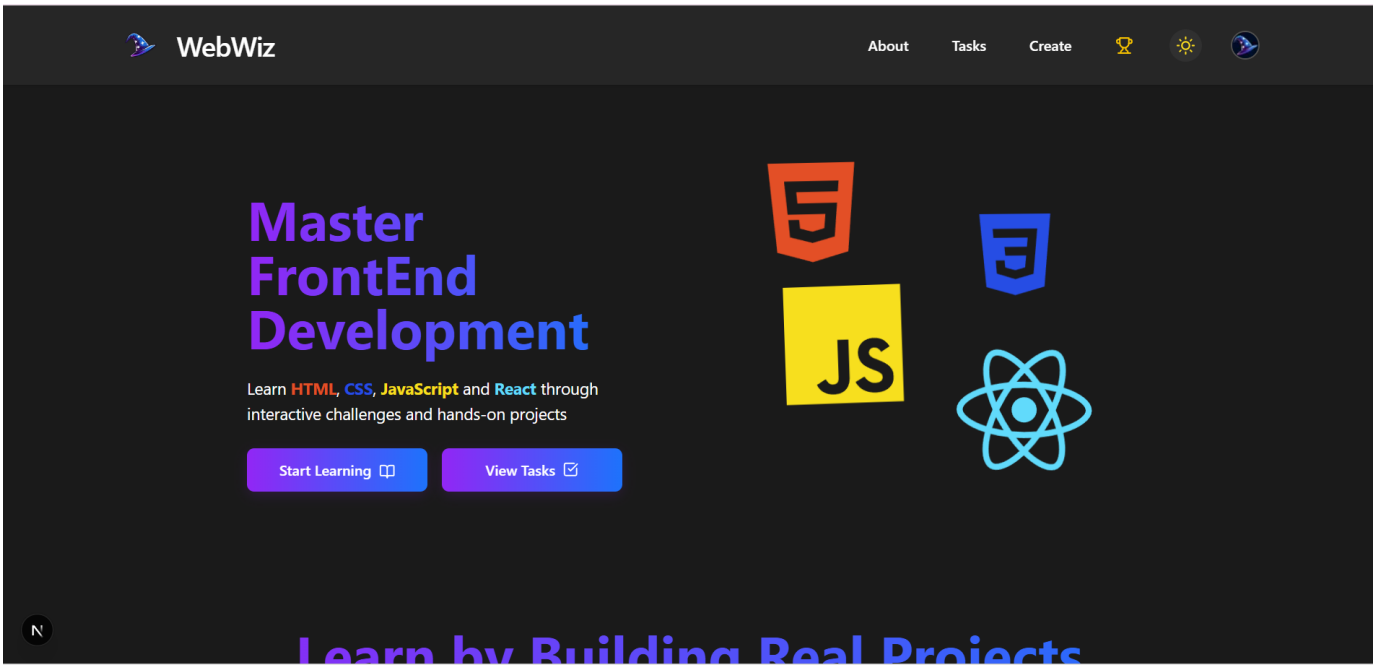



*Figure 5-6* ***Home page***

## 5.2.6 structured learning roadmap selection page


*Figure 5-7* ***Roadmap page***

## 5.2.7 structured learning road map for Frontend fundamentals

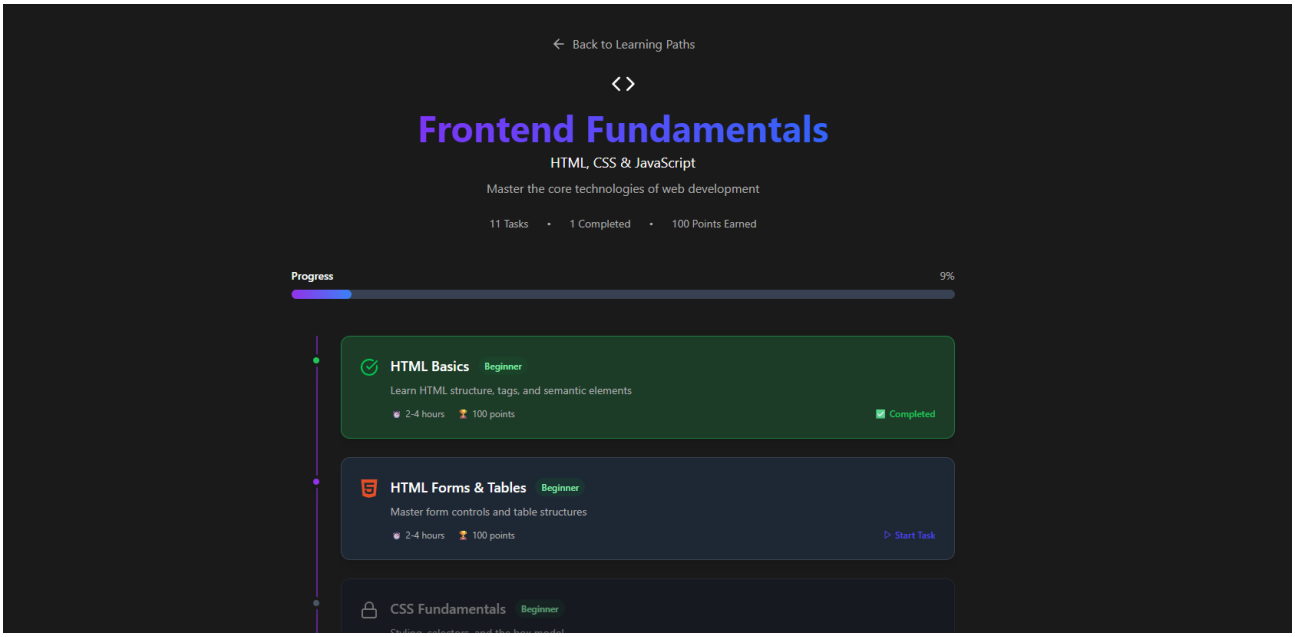



*Figure 5-8* ***Roadmap page***

### 5.2.8 structured learning road map for React development

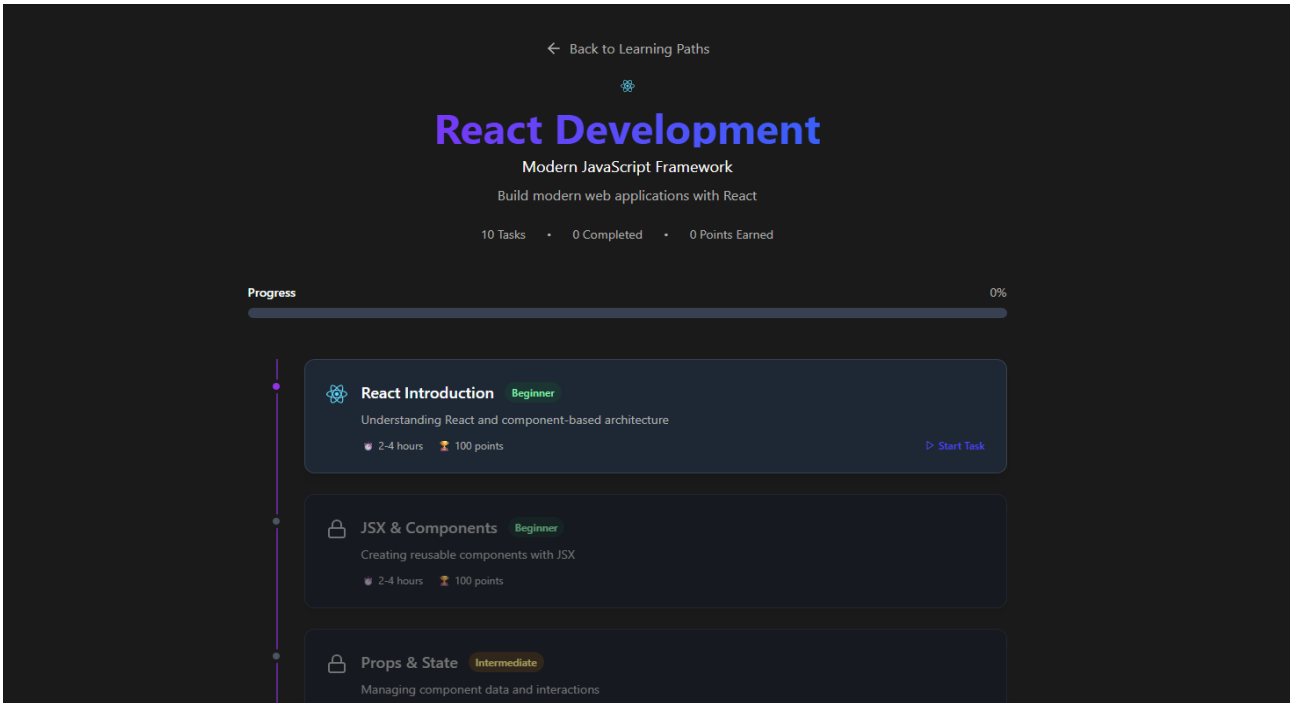



*Figure 5-9* ***Roadmap page***

### 5.2.9 Leaderboard screen

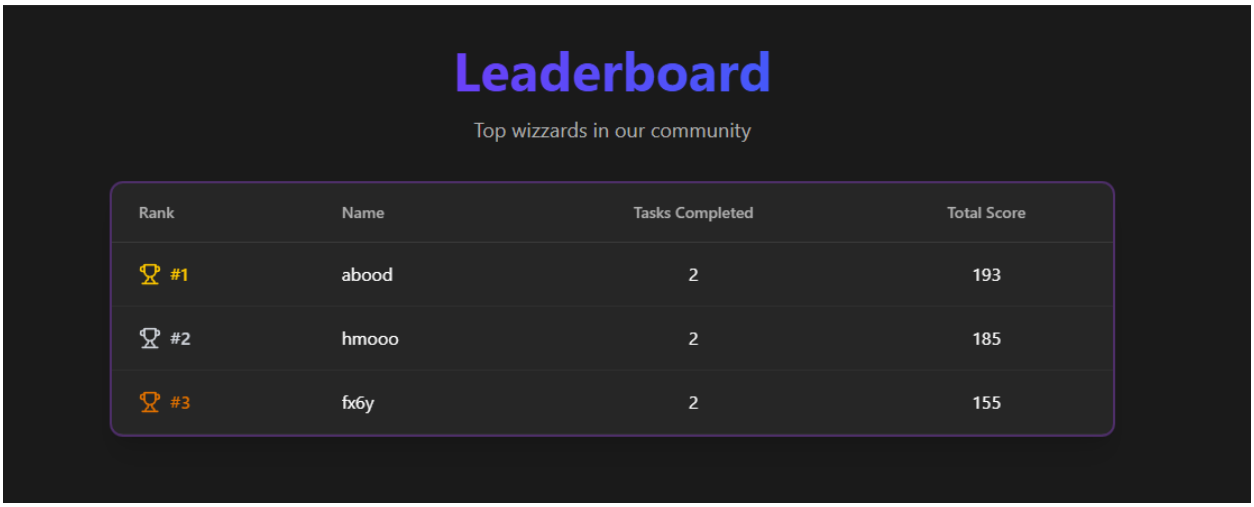



*Figure 5-10* ***leaderboard page***

### 5.2.10 Profile page

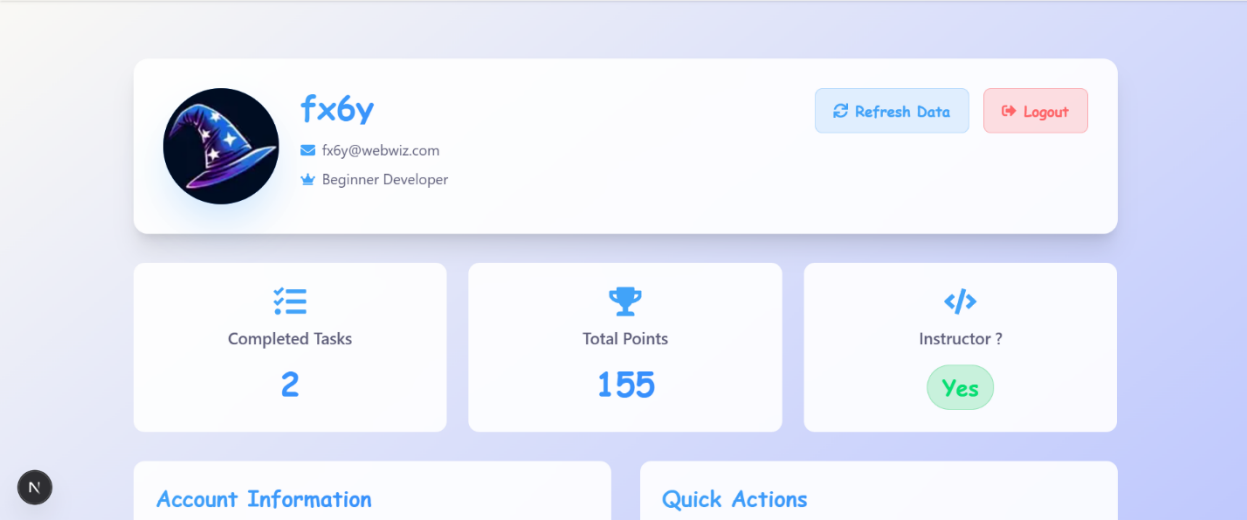



*Figure 5-11* ***profile page***

### 5.2.11 Task modal in live code editor

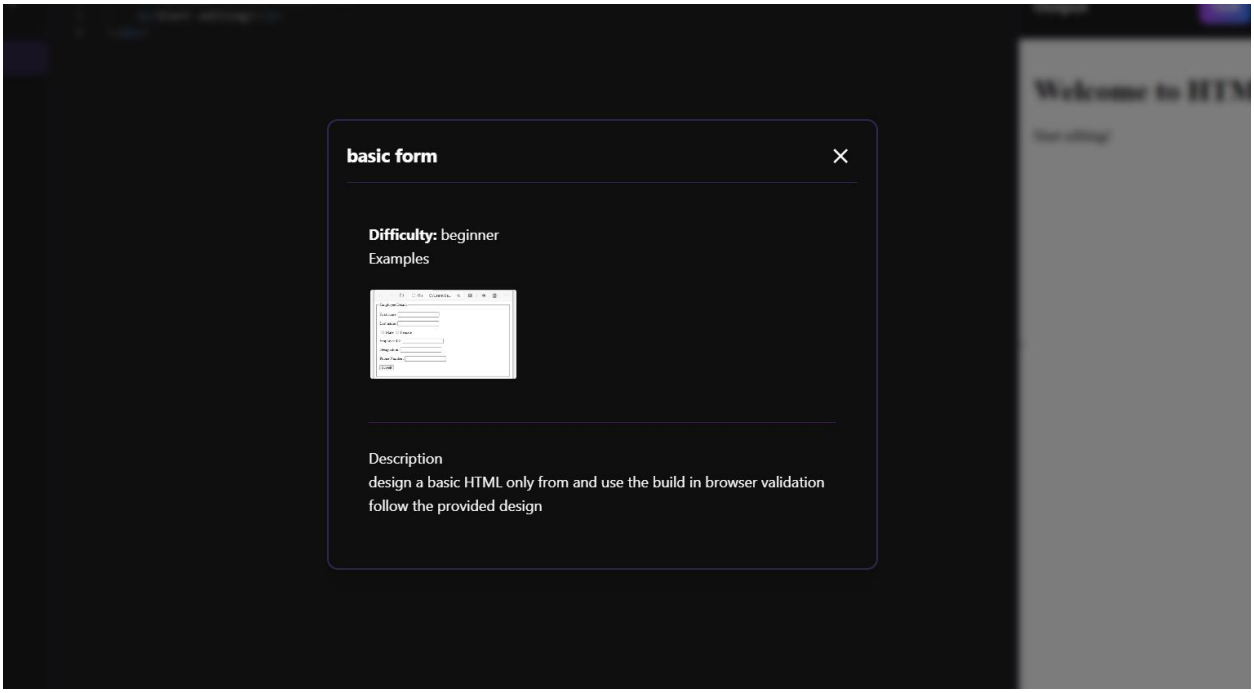



*Figure 5-12* ***Live Code Editor page***

### 5.2.12 Live code editor

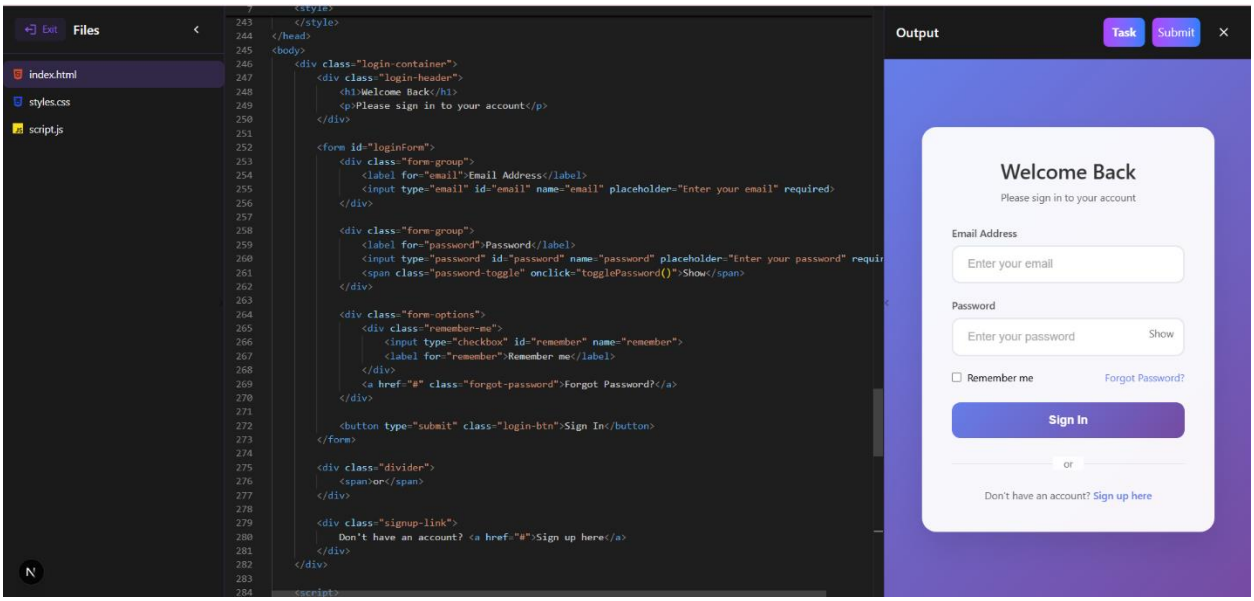



*Figure 5-13* ***Live Code Editor***

### 5.2.13 AI validation page

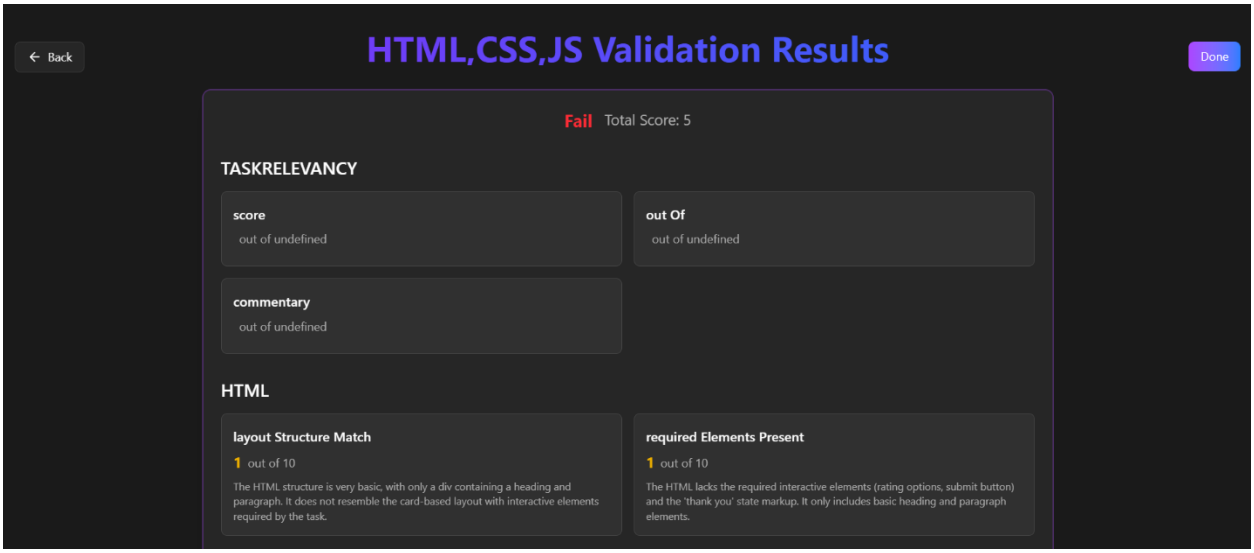



*Figure 5-14:****Validation***

### 5.2.13 Create task multistep form

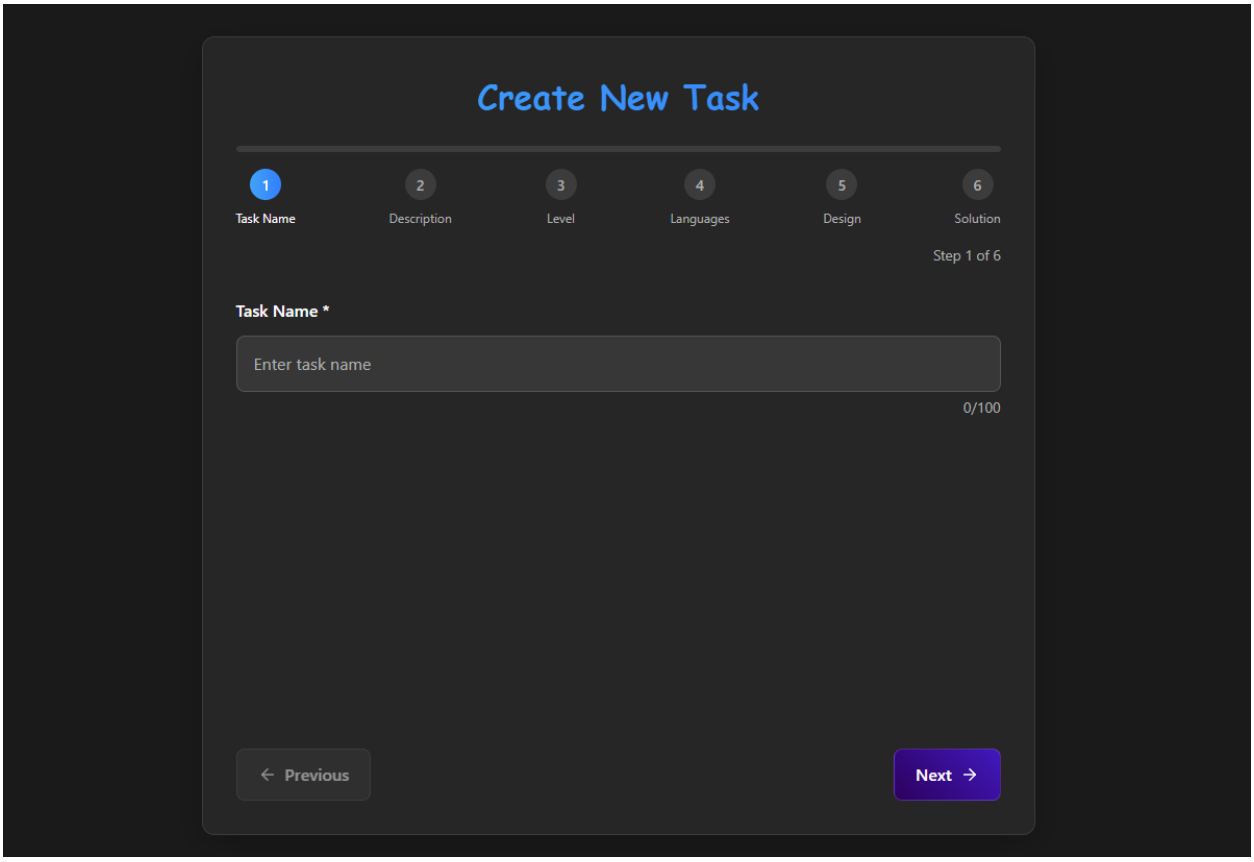



*Figure 5-15* ***Create new task form(step 1)***

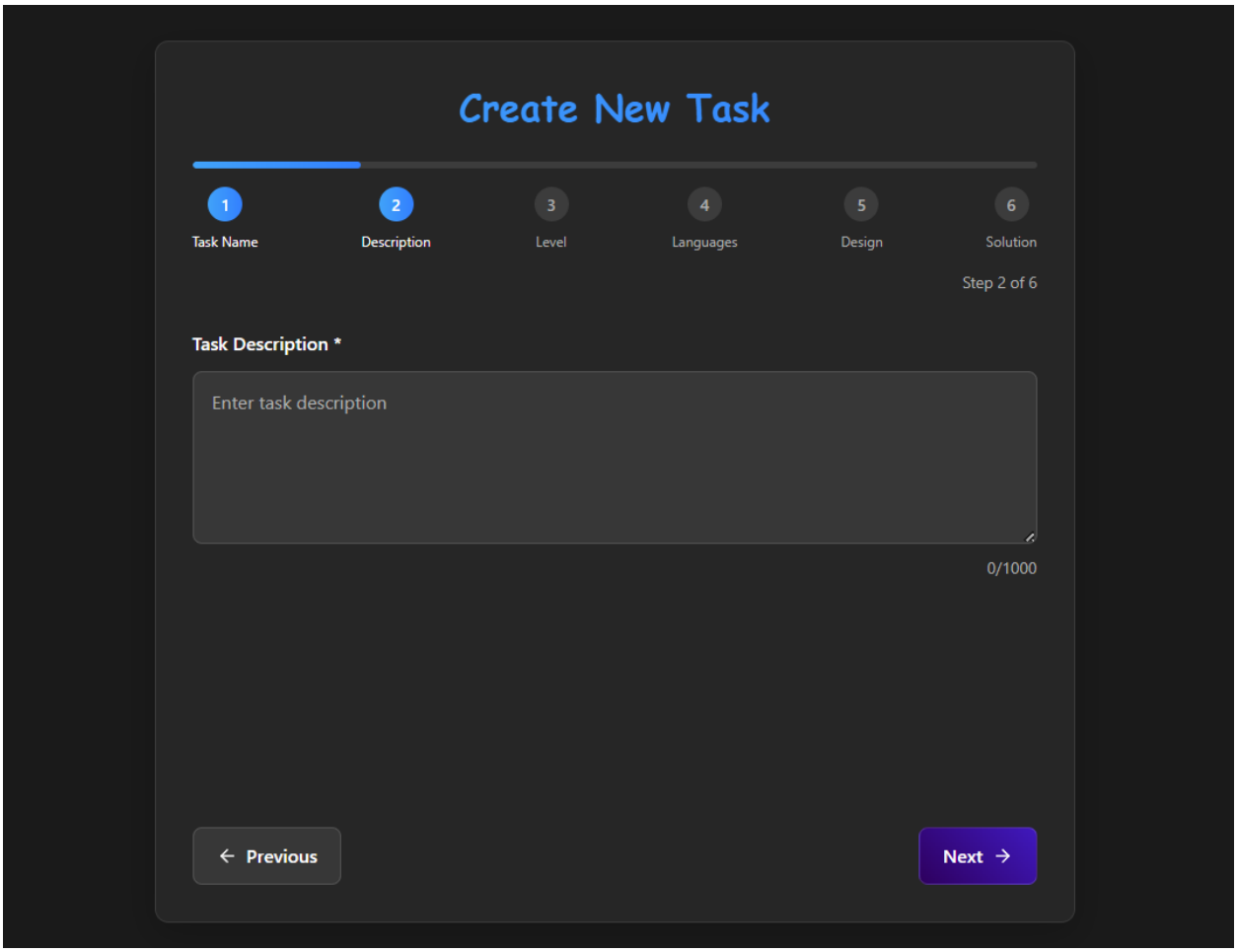



*Figure 5-15 Create new task* ***form(step 2)***

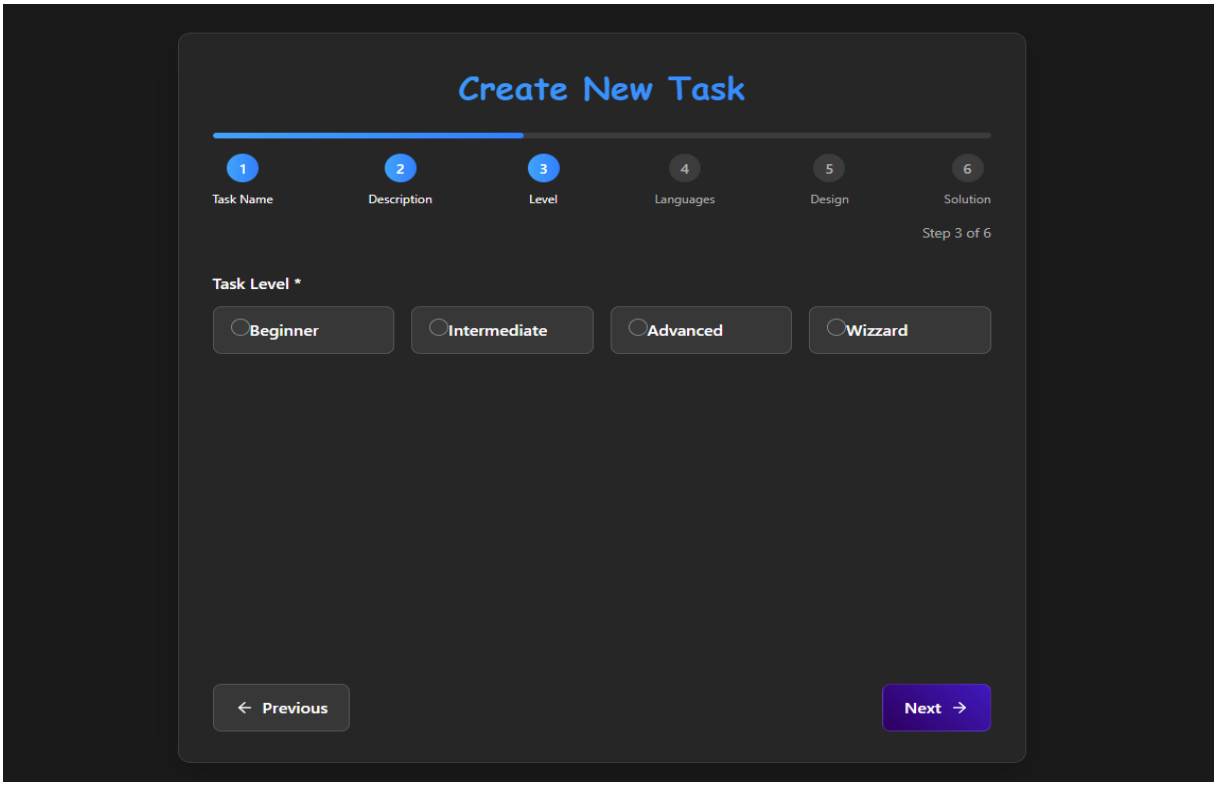



*Figure 5-17* ***Create new task form(step 3)***

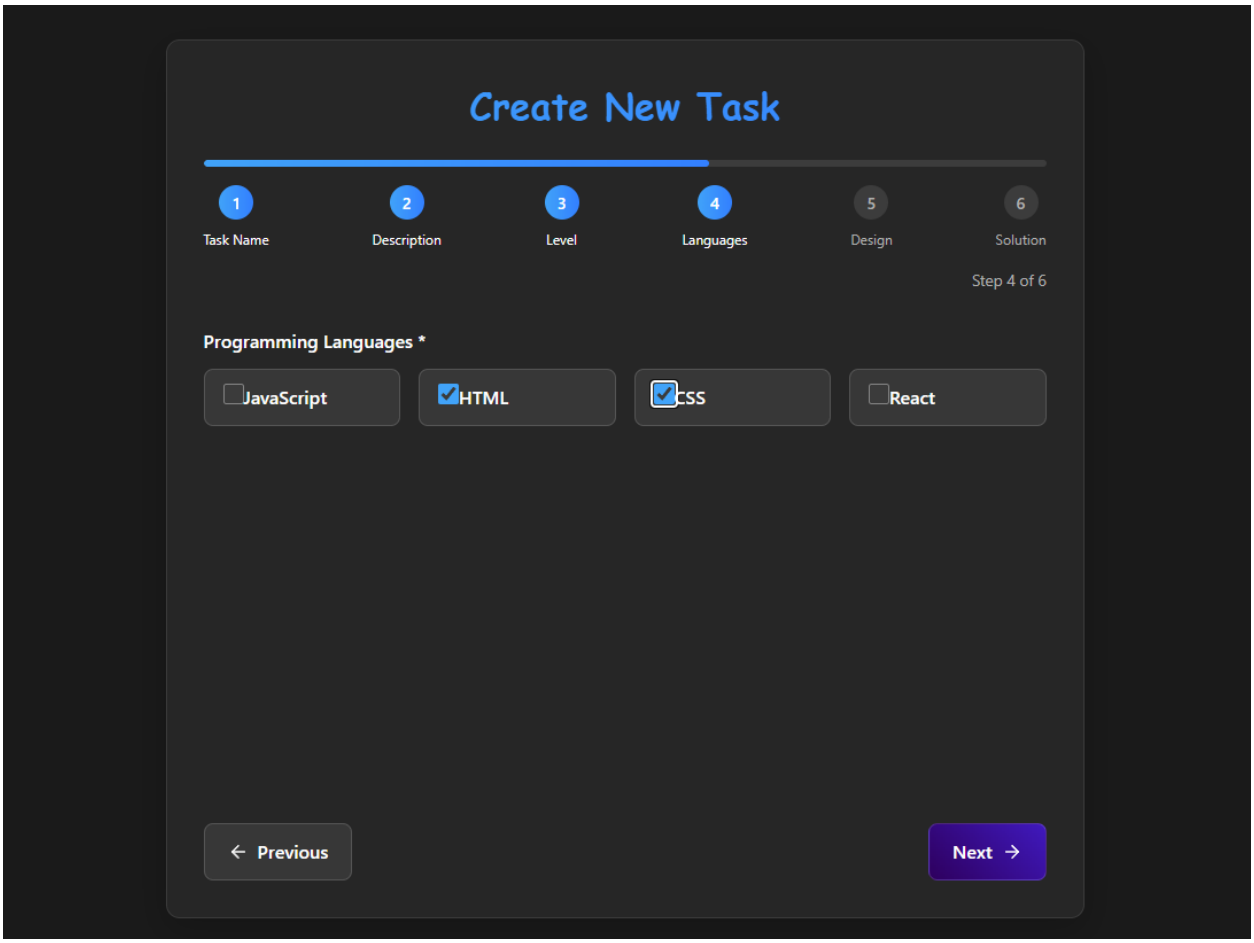



*Figure 5-18:**Create new task form(step 4)***

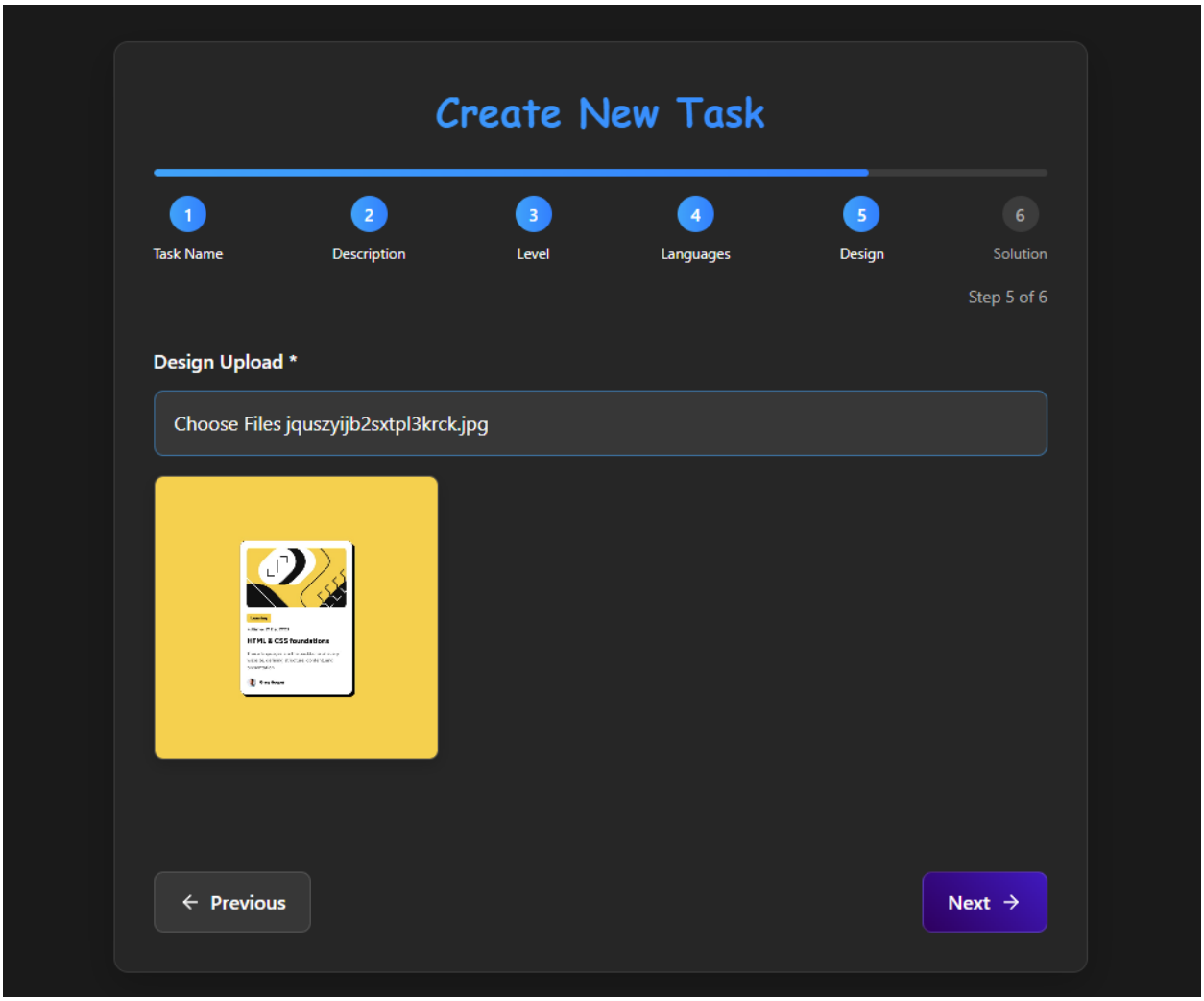



*Figure 5-19* ***Create new task form(step 5)***

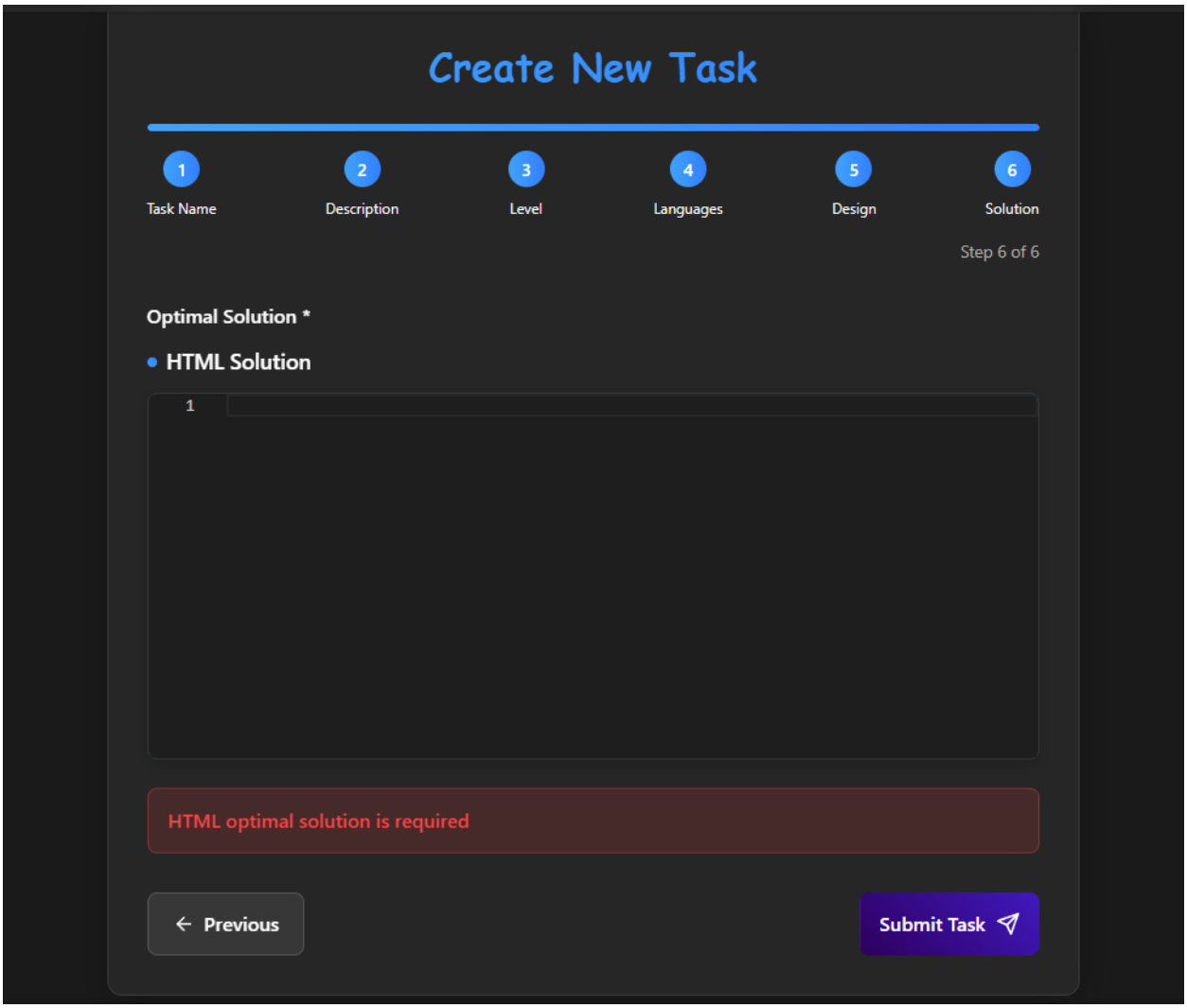



*Figure 5-20 Create new task* ***form(step 6)***

# Chapter 6

# EVALUATION AND TESTING

## 6.1 Introduction

Testing is essential in software development to ensure code quality, reliability, and functionality. It helps detect bugs and issues early, reducing costs and improving user satisfaction. By validating software behavior under various scenarios, testing enhances stability and minimizes risks during updates or feature additions. It also aims to root out undesirable behaviors such as system crashes, incorrect computations, data corruption, and unwanted interactions with other systems. Moreover, testing ensures compliance with project requirements and standards, fostering stakeholder trust. Ultimately, it is a critical investment in the success and maintainability of any software project.

## 6.2 Testing Levels

During the development of **WebWiz**, a structured manual testing approach was followed, encompassing multiple levels of testing to ensure the system's reliability and correctness:

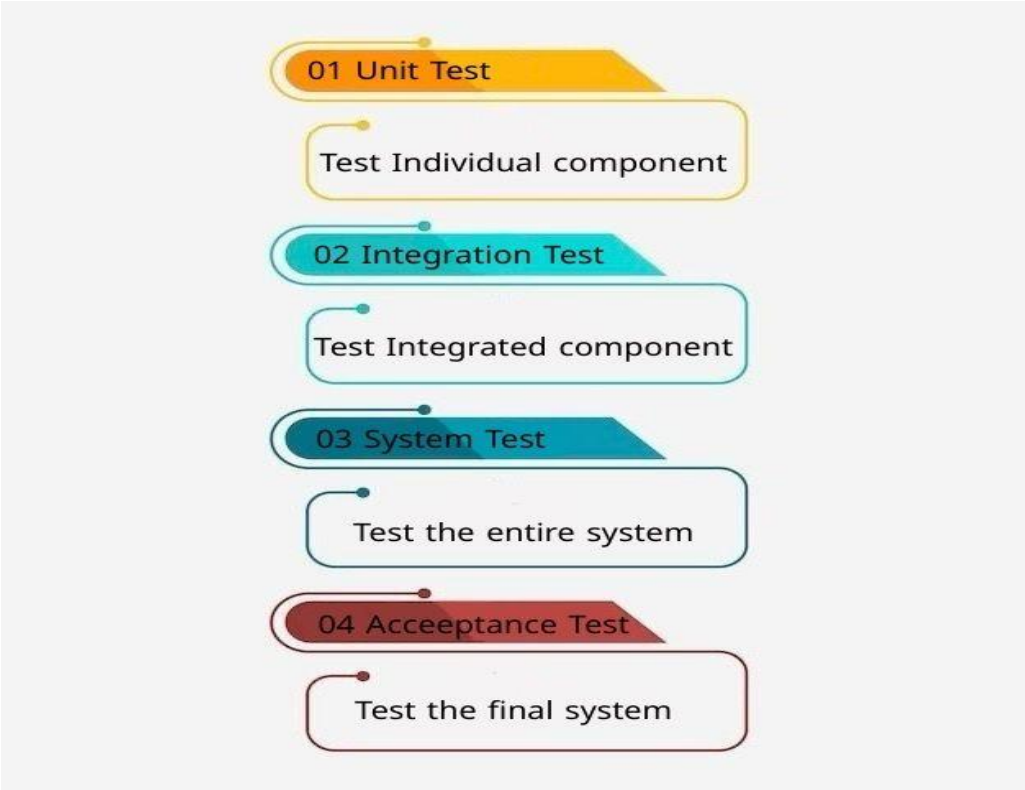



*Figure 6-1: level of testing*

### *6.2.1 Unit/Component Testing*

Unit testing is done at the code level, where each component is tested individually to ensure their impartiality and analyze their functionality.

### *6.2.2 Integration Testing*

Integration Testing enables software testers to test group units integrated into a system or subsystems, it helps identify any bugs or issues arising from coding errors or integrations between modules.

### *6.2.3 System Testing*

System testing is performed on an integrated environment comprising the whole application, where all components are assessed against specific business requirements.

### *6.2.4 Acceptance Testing*

Acceptance testing involves testing the system's and functional Non-functional aspects, such as performance, security, usability, accessibility, compatibility, and reliability.

### *6.2.5 Acceptance Testing and evaluation*

User Acceptance Testing involves engaging real users, often from the programming community, to assess the application's ease of use and overall performance. This testing phase provides crucial feedback from end users, enabling the development team to align the platform with users' needs and expectations.

## 6.3 Types of Testing

### *6.3.1 Functional Testing*

Testing all functions within the application to ensure they operate efficiently and effectively.

### *6.3.2 Compatibility Testing*

Test the application on various devices and operating systems as necessary, ensuring consistent efficiency across all platforms.

### *6.3.3 Performance Testing*

We conducted this testing to guarantee the system's responsiveness to user interactions, establishing a maximum response time of 5 seconds for each interaction.

### *6.3.4 Security Testing*

This testing aims to verify the security of all incoming data into the system, ensuring protection against potential attacks.

### *6.3.5 Usability Testing*

The purpose is to ascertain that the system is user-friendly and straightforward for the intended audience. We validate this through an evaluation process, with further details to be provided later.

### *6.3.6 Test Automation*

It refers to automating recurring tests to streamline testing procedures and minimize time requirements. However, we found no necessity to incorporate it into the project.

## 6.4Questionnaire

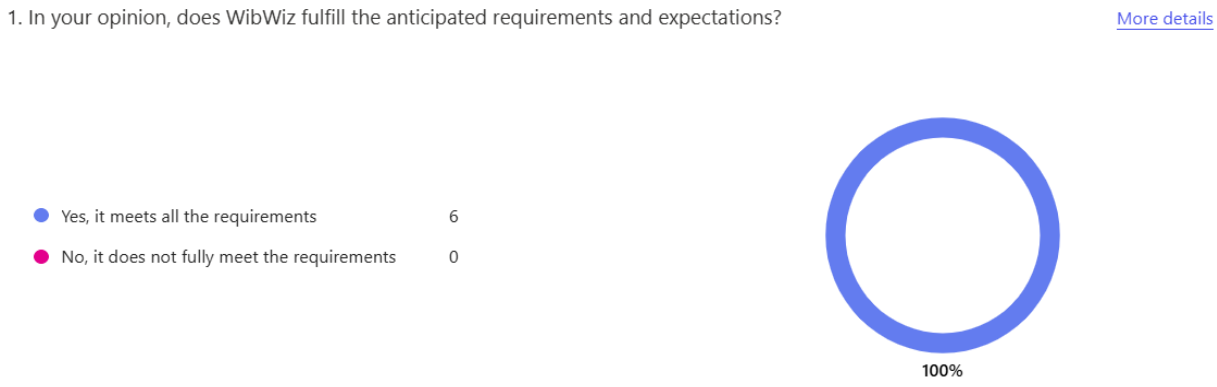


*Figure 6-2questionnaire*


*Figure 6-3 questionnaire*


*Figure 6-4questionnaire*

4. Was the information displayed on the site sufficient to understand its functionality?

More details


*Figure 6-5 questionnaire*

5. Did you feel that the site provided real value and helped you develop your skills?

More details


*Figure 6-6 questionnaire*

6. Are there any missing features you wish to have?

More details

6
Responses

Latest Responses
"No"
"Providing communication between learners"
"No"
...

*Figure 7-6 questionnaire*

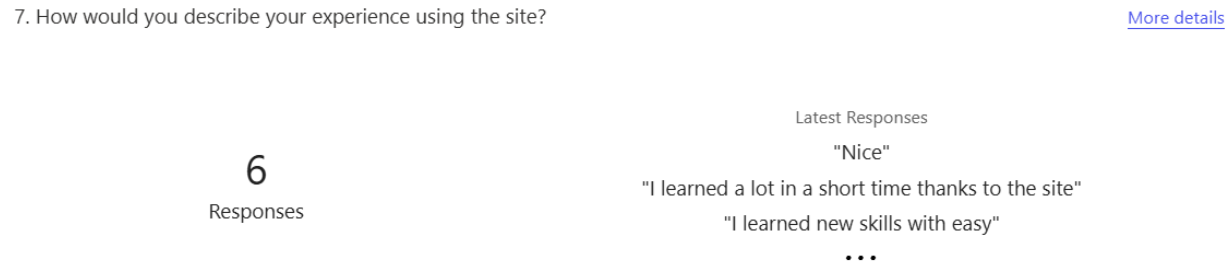



*Figure 6-8:questionnaire*

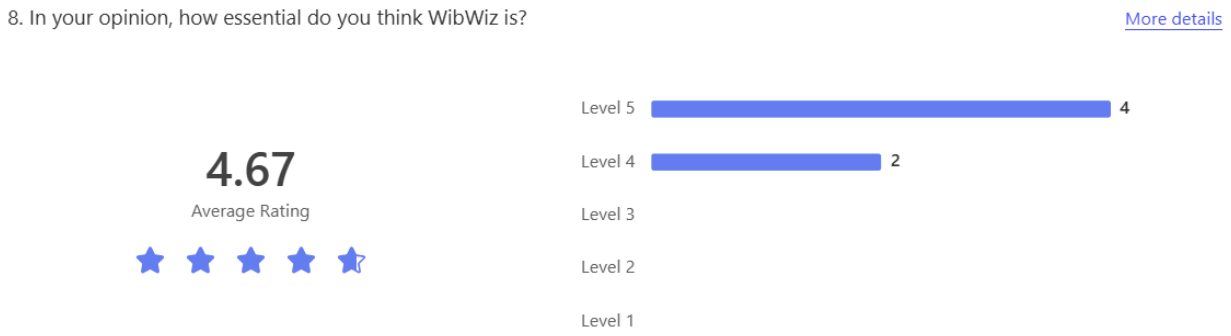



*Figure 6-9: questionnaire*

# Chapter 7

# CONCLUSION

## 7.1 Conclusion

In a world where progress is measured by the speed of learning, the smartest path is the one that combines proper guidance with intelligent tools.
**WebWiz** has successfully reduced learning time and increased learner engagement by providing smart, personalized learning experiences tailored to each user's level.

This project doesn't just offer technical solutions  it redefines the learning experience from the ground up, centering it around the learner, adapting to their needs, and optimizing their time and effort.

We chose not to be just another resource in the vast ocean of content, but rather a smart digital mentor that guides learners step by step toward mastery. It provides the necessary support, direction, corrections, and real-time feedback. Through features like automatic task correction and optimal solution suggestions powered by AI, learners can instantly learn from their mistakes and overcome technical challenges efficiently and smoothly.

This project doesn't merely create learners  it builds confident developers, ready to enter the job market armed with skill and structured knowledge.
It is an invitation to anyone seeking a smart beginning and a professional end.

## 7.2 Future Works

1. Implementing an admin system to provide enhanced control over platform management.

2. Introducing additional frontend frameworks (e.g., Angular, Vue).

3. Supporting the organization and management of coding competitions.

4. Expanding into full-stack development by introducing backend tasks (e.g., designing API endpoints).

## 7.3 References


***[1]*** *Stack Overflow. Stack Overflow Developer Survey 2022. Stack Overflow, 2022,* ***https://survey.stackoverflow.com/2022/.*** *Accessed 17 March 2025.*

***[2]*** *PlayCode. "PlayCode – JavaScript Playground." PlayCode, 2025,* ***https://playcode.io/.*** *Accessed 17 March 2025.*

***[3]*** *Frontend Mentor. "Frontend Mentor – Front-end Coding Challenges." Frontend Mentor, 2025,* ***https://www.frontendmentor.io/.*** *Accessed 17 March 2025.*

***[4]*** *LeetCode. "LeetCode – Coding Challenges and Technical Interview Preparation." LeetCode, 2025,* ***https://leetcode.com/.*** *Accessed 17 March 2025.*

***[5]*** *Sommerville, Ian. Software Engineering. 10th ed., Pearson, 2016.*

***[6]*** *Sommerville, Ian. Software Engineering. 9th ed., Addison-Wesley, October 2009.*

***[7]*** *Gregor, Shirley, and David Jones. "The Anatomy of a Design Theory." ANU Research Publications, 22 Mar. 2007.*