Addis Ababa Institute of Technology

School of Information Technology and Engineering

Department of IT/SW Eng.

Code Land

Team Members

1. Kidus Yohannes	ATR/7242/11
2. Mikael Teshome(IT Add)	ATR/1408/11
3. Naboni Abebe	ETR/0714/11
4. Yohannes Addisu	ATR/1256/11
5. Yonatan Merkebu	ATR/4308/11

Advisor: Mr. Yoseph Abate

June 2023

Addis Ababa Institute of Technology
School of Information Technology and Engineering

Code Land

This Project documentation submitted in partial fulfillment of the requirements for the Degree of Bachelor of Science in **Software Engineering and IT**.

Project Advised by:

Mr. Yoseph Abate

Name and signature of Members of the examining board:

<u>NameTitleSignatureDate</u>

1.	___________________	Advisor	______________	______________
2.	___________________	Chairperson	______________	______________
3.	___________________	Examiner	______________	______________
4.	___________________	Examiner	______________	______________
5.	___________________	Examiner	______________	______________

June 2023

Declaration

We declare that this project is our original work and has not been presented for a degree in any other university.

<u>NameSignatureDate</u>

1. Name of Student 1___________________ ___________________
2. Name of Student 2___________________ ___________________
3. Name of Student 3___________________ ___________________
4. Name of Student 4___________________ ___________________
5. Name of Student 4___________________ ___________________

This project documentation has been submitted for examination with my approval as university advisor:

Mr. Yoseph Abate__________________________

June 2023

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who have contributed to the successful completion of our final year software project, Code Land.

First and foremost, we would like to thank our project advisor, Mr. Yoseph Abate, for their guidance, support, and valuable feedback throughout the project. We are also grateful to the faculty members of the Software Engineering department for providing us with the necessary resources and facilities to carry out our project.

We would like to extend our appreciation to our fellow students who have provided us with their feedback and suggestions, which have helped us to improve the quality of our project.

Finally, We would like to express great appreciation to the School of Information, Technology and Engineering for enabling us to undertake our final project.

ABSTRACT

CODE LAND is a platform for students who want to learn Data Structures and Algorithms. It aims to eliminate the difficulties associated with understanding the concepts of data structures and algorithms. To make the topics easier to comprehend, it will provide graphic implementations of the majority of the algorithms and concepts (Recursion, Tree, Graph, etc.).

Additionally, it will provide a selected set of questions on certain subjects to give students hands-on practice. There will be thorough descriptions on both the subject and answers to the questions.

It will also have a code playground(Integrated Development Environment) with almost every language support where students can easily test their code. It will also enable pair-programming with other students on the same code, giving the ability to learn and solve together.

Table of Content

ACRONYMS

SRS - Software Requirements Specification SDS - Software Design Specification

IDE – Integrated Development Environment IT – Information Technology

UUID - Universally Unique Identifier

UML – Unified Modeling Language

SDLC – Software Development Life Cycle, is a set of steps used to create software applications.

AAiT: Addis Ababa Institute of Technology

SITE: School of Information Technology and Engineering

Chapter 1: INTRODUCTION

1.1 Background

Learning data structures and algorithms allow us to write efficient and optimized computer programs and that’s why it is included in our curriculum as a subject.

Even though our group, and we think many other students, were quite enthusiastic about the course, the learning process wasn't a smooth ride. When attempting to understand new concepts and put them into practice, we encountered several obstacles.

In addition, the way the course was taught was by using only one programming language. This was a constraint for students with knowledge of other programming languages.

That’s why we are proposing what we think is the best way to learn the course, by using CODE LAND.

CODE LAND will be the first learning and practicing platform for students (not only for university students) in Ethiopia to easily learn Data Structures and Algorithms guided by visual effects and practice within the platform itself without being constrained to a single programming language.

We aim to complete the project within 3-4 months with no cost in the development phase.

Our plan is to provide the platform to our university (Addis Ababa University) at the end and help both teachers and students on the way to successfully completing the course.

1.2 The Existing System

Currently, our university (AAIT) is not using any special platform to teach students about Data Structures and Algorithms. The course is taught on a whiteboard and using slides which is one of the reasons that made it difficult for students to grasp concepts.

The fact that Python is the sole programming language used in the course is another major source of difficulty. Python is the preferred programming language for the majority of students, however it is not taken into account while teaching other students how to write in other programming languages.

1.3 Statement of the Problem

Students (IT and Software Engineers) need Data Structures and Algorithms skills so that they can be successful on their job. They could learn the concept in Universities or they could learn on their own. The aim of CODE LAND is to make the learning process of these concepts easy. To do so, CODE LAND must solve two main problems:

- Change the way the course is thought (Stopping the usage of only slides and white boards)
- Remove programming language constraints (No single programming language)

These two problems have existed since the course was included in the curriculum and no one has done anything to address them.

Element	Description
The problem of ...	Unsuitable way of teaching of the course
Affects ...	Students are affected
And results in ...	Better way of learning the course
Benefits of a solution ...	It will help students learn concepts faster and practice more with their desired programming language

1.4 Objective of the Project

Upon completion of this project, students will learn and practice Data Structures and Algorithms easily.

1.4.1 General Objective

This project will provide a platform where students learn and practice Data Structures and Algorithms either individually or in groups with their peers using the platform provided by this project.

1.4.2 Specific Objective

The specific objective of this project is to make the teaching and learning process of Data Structures and Algorithms courses in universities a smooth and enjoyable ride. No student should struggle with a topic and become lost due to imprecise explanations.

Learning of new concepts with no difficulty, with visual guidance and practice areas with student's choice of programming language is the main objectives of CODE LAND.

1.5 Proposed System

The system we are proposing is a web based platform with a handful of features. The system will have the features listed below:

- Authentication system to uniquely identify students
- Questions with detailed explanation and answers
- Code Visualization to easily understand the topic
- A code submission system to run code against a question
- A code playground with pair programming feature
- Profile status tracking

1.6 Feasibility Study

1.6.1 Economic Feasibility

The project will have no cost for the end user. There won't be any tools or documents to download in order to use the website. The only thing the end user needs to have is an internet connection.

1.6.1.1. Developmental cost

The development of the project is taken care of using free tools and a local development environment which is totally free of cost.

1.6.1.2. Operational Cost

If adopted, the system will have all front-end, back-end and database deployed so there will be charge in order to be functional.The costs will heavily depend on active users because in order to serve more users, the system might need to scale up and down.

1.6.2 Technical Feasibility

This project used Next.js for the front-end, Node.js for the back-end and MongoDB for the database. The project also has Micro services architecture.

The project also used many third-party libraries to achieve many of the functionalities.

1.6.3 Schedule Feasibility

From project inception through product demonstration, the project is anticipated to take months, and we want to complete the project on schedule.

1.7 Scope

We believe CODE LAND will be the go to platform for students who want to learn Data Structures and Algorithms. To make this a reality we solved all the hardship that arises when learning the course which are:

- Unsuitable way of teaching of the course
- Programming language constraint

Besides solving the inconveniences, CODE LAND has extra features like questions to solve, pair programming, visualization and practice areas in which students can learn and try to solve problems together.

1.8 Methodology

For developing the project, we used Waterfall methodology.

Waterfall model — is a sequential development process that flows like a waterfall through all phases of a project (analysis, design, development, and testing, for example), with each phase completely wrapping up before the next phase begins.

For the first step, which is the collecting requirement, we didn't need to interview students or send out questionnaires because our team has passed through those difficult times of learning the course. This means that we clearly have the picture of the problem and how we must improve it. For the planning and designing phase, we worked on it for 3 weeks and immediately moved to the implementation phase.

Of all the phases, the implementation phase takes more time. We planned to develop a highly performant and maintainable project.

Finally, we will test the project if it meets the requirements listed above and also get feedback from target users which are students.

We plan to develop an application that supports all device types and can be accessed with ease. So, the project will be a web based application so it will use Next.js framework for the front-end, Node.js for the back-end and MongoDB for the database. The project will also use Micro services architecture.

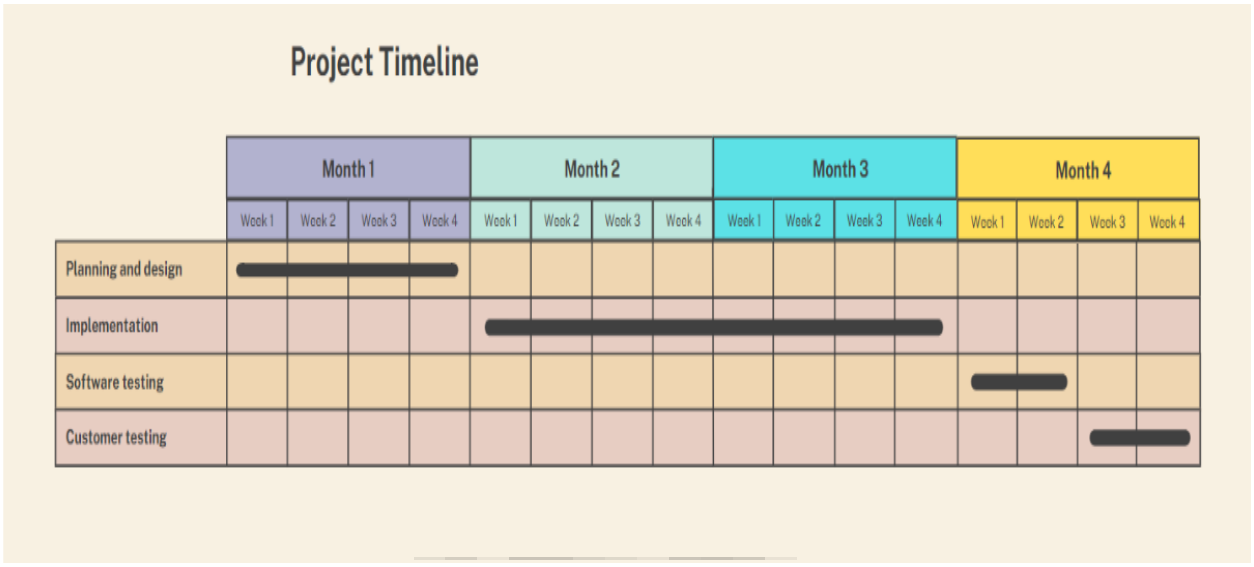
The project will also use third-party libraries to achieve many of the functionalities.

After completing the project, we plan on testing it with various methods to measure its success. While integration testing is the main part of the testing phase, the real success of the project will be determined by actual students in our university. We plan on distributing the project to beta testers to receive feedback from them.

1.9 Project Management plan

1.9.1.Time Management plan

The project timeline starts immediately after the proposal gets accepted and we are done with creating SRS and SDS documents.



Project Timeline

	Month 1				Month 2				Month 3				Month 4			
	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4
Planning and design	■	■	■	■												
Implementation					■	■	■	■	■	■	■	■				
Software testing													■	■		
Customer testing															■	■

Table 1.1: project Timeline

1.9 .2 Quality Management Plan

The project is risk-free, but we'll make sure the outcome lives up to expectations.

The quality of the product is measured based on if the provided method (solution) ends up making the previous hustle go away or add more to it.

That’s why we tested the product with target users at the end of the development phase and gathered feedback.

1.9 .3. Communication Management Plan

The team consisted of five students who were able to complete the assigned tasks with their own insights. Team members cooperated and planned to meet in person 2 days a week to discuss progress and resolve any difficulties. In addition, we used a task management system called Trello to manage the tasks assigned to each team member. If necessary, team members and consultants could also view the task progress, complete the task and give comments in the Trello.

Type of Communication	**Method / Tool**	**Frequency/ Schedule**	**Information**	**Participants / Responsibilities**
Internal Communication:				
Project Meetings	Teleconference/in person meetings	2 days a week	Review project status	Team Members
Sharing of project data	Teleconference/Github	2 days a week	All project documentation and reports. Discuss what each team member did and what they'll do.	Team Members
Milestone Meetings	Teleconference/ In person meeting/ Google meet	Before milestones	Project status (progress)	Team Members
Final Project Meeting	Teleconference/ In person meeting	M6	Wrap-up Experiences	Team Members
External Communication and Reporting:				
Project Report	Google sheet	once a week	Project status - progress - forecast - risks	Team Members
SteCo Meetings	Teleconference/ Email / Google Doc	Monthly		Team Members

Table 1.1: Communication Management Plan

Chapter 2: Requirement Analysis

2.1 Introduction

This paper outlines the software requirements for our capstone project, "Code Land," a website being created for students. This document seeks to gather information, analyze it, and provide a comprehensive overview of the entire application we are creating. An overview of the system, specific requirements, modeling specifications, diagrams, and a description of the prototype that will be created to demonstrate the system's capabilities are all included in this chapter.

2.2General Description

The SRS provides a detailed description of the purpose and scope of the software, the characteristics of the users who will be using the software, the operating environment in which the software will be used, and any assumptions, dependencies, and constraints that may affect the development or implementation of the software.

2.3Product Perspective

CodeLand appears to be a unique project that combines visualizers and an integrated development environment (IDE) to help teach data structures and algorithms to students. There are several other products and projects that aim to teach these concepts, but they tend to focus on one aspect of the learning experience, such as interactive visualizations or text-based coding tutorials.

One example of a similar product is VisuAlgo, which also uses visualizations to help students understand data structures and algorithms. However, VisuAlgo does not have an integrated development environment for students to practice coding.

Another example is Codecademy, which is an online platform that provides interactive coding tutorials for a variety of programming languages and concepts, including data structures and algorithms. However, Codecademy does not provide visualizations to help students understand these concepts.

Additionally, there are several open-source projects such as Jupiter that allow users to create interactive notebooks that mix code, text and visualizations. These projects can be used to create interactive tutorials and lessons but don't have the integrated development environment features.

Overall, CodeLand appears to be unique in its combination of visualizations and an integrated development environment for students to practice coding. This approach may provide a more comprehensive learning experience for students

2.4Product Functions

When finished, the platform will be a one-stop platform for students and any individuals who would like to learn and practice Data structures and algorithms with detailed descriptions and visual aids.

2.5User Characteristics

The platform only has one target customer which is an individual who would like to learn Data structures and algorithms.

The users will interact with the platform in various ways. Starting from registering, learning new topics and practicing questions alone or in pairs.

2.6General Constraints

The following are considered to be the general constraint for the system we are going to develop:

(1) **Reliability Requirement** - the system must be available 24 hours a day 7 days a week and 365 days a year. The functionalities of the system should always be working.
(2) **Criticality of the system** - the system is important to all users to keep updated with ongoing activities. Especially updates shall be available.
(3) **Safety and Security consideration** - the system and user data shall be kept on a server and necessary precautions have to be taken for data not to be compromised.
(4) **Interface to other applications** - The platform will depend on a third party API for the code execution part.

2.7Assumptions and Dependencies

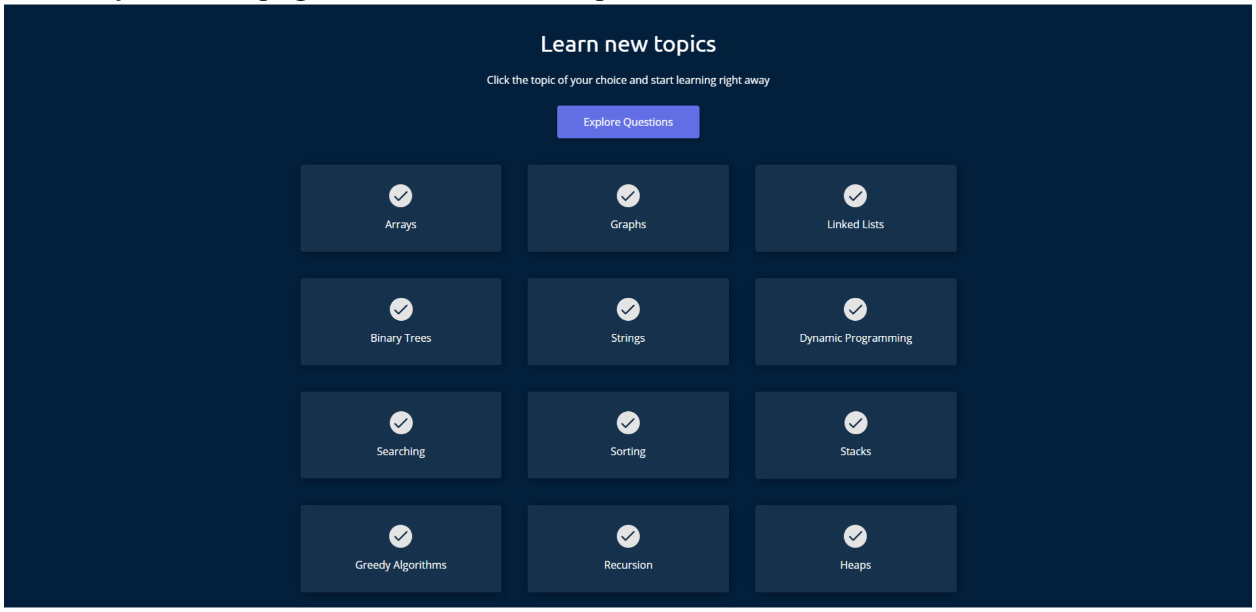
There are only two assumptions. The first being, users know how to use the internet and the second is, they have a browser application installed and can access our website.

Since the platform is web based, it won't have any dependency with the user's machine.

2.8 External Interface Requirements

2.8.1 User Interfaces

Portion of the home page where users select topics to learn

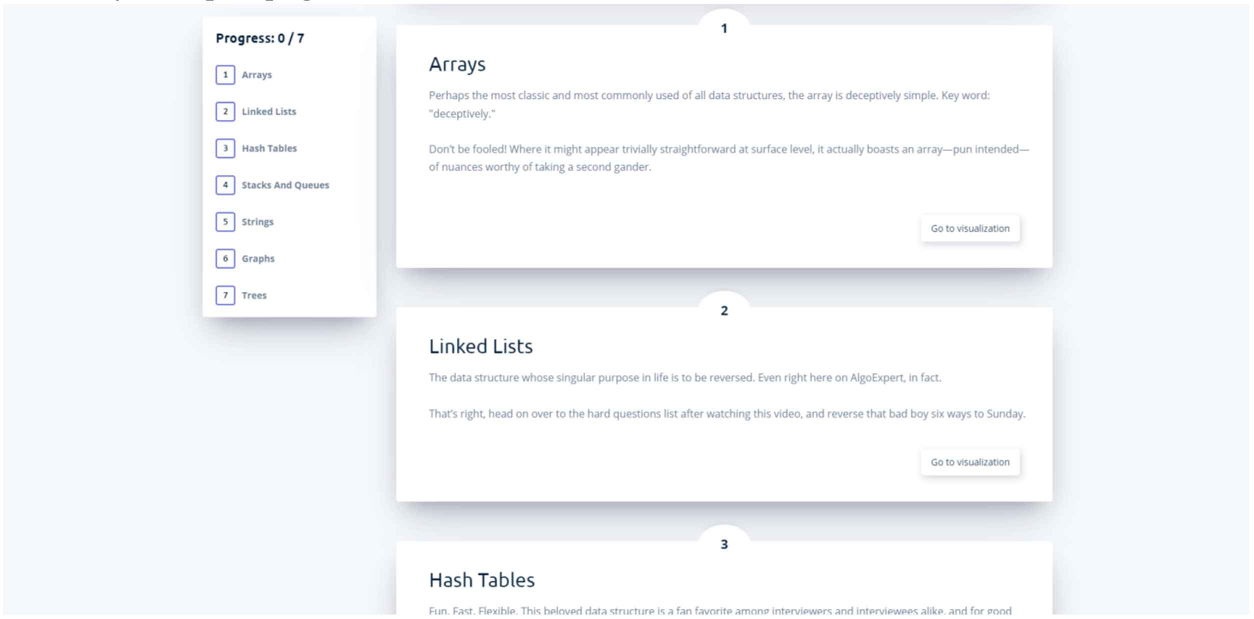


Learn new topics

Click the topic of your choice and start learning right away

Figure 2.1. Topics Page

Portion of the topics page



Progress: 0 / 7

1 Arrays
2 Linked Lists
3 Hash Tables
4 Stacks And Queues
5 Strings
6 Graphs
7 Trees

1

Arrays

Perhaps the most classic and most commonly used of all data structures, the array is deceptively simple. Key word: "deceptively."

Don't be fooled! Where it might appear trivially straightforward at surface level, it actually boasts an array—pun intended—of nuances worthy of taking a second gander.

Go to visualization

2

Linked Lists

The data structure whose singular purpose in life is to be reversed. Even right here on AlgoExpert, in fact.

That's right, head on over to the hard questions list after watching this video, and reverse that bad boy six ways to Sunday.

Go to visualization

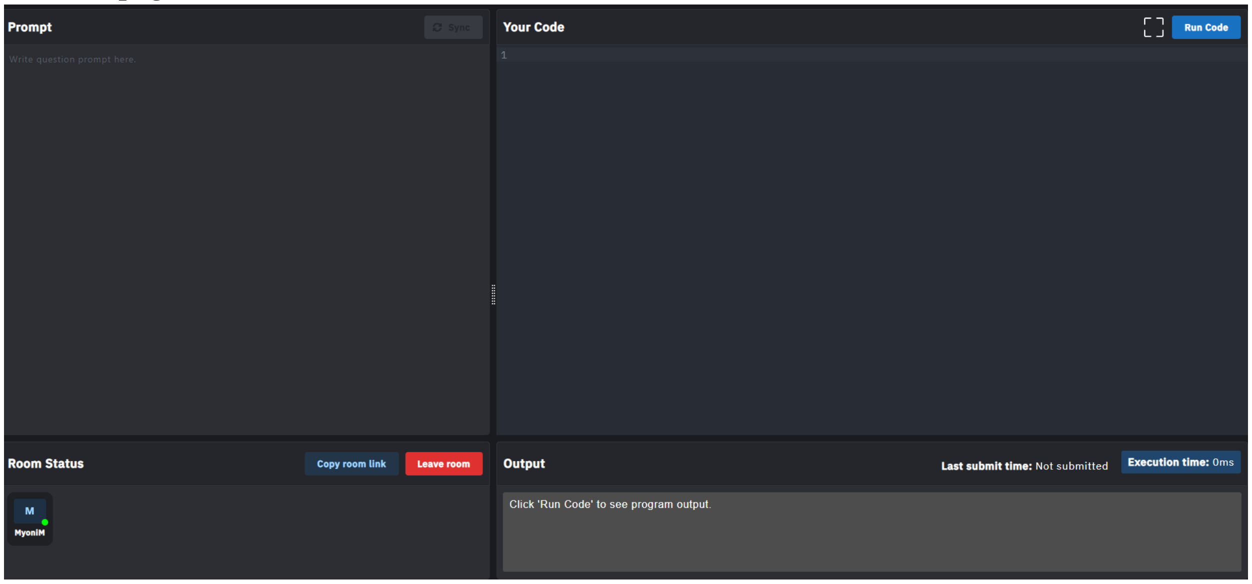
3

Hash Tables

Fun. Fast. Flexible. This beloved data structure is a fan favorite among interviewers and interviewees alike, and for good

Figure 2.2: Specific Topic Page

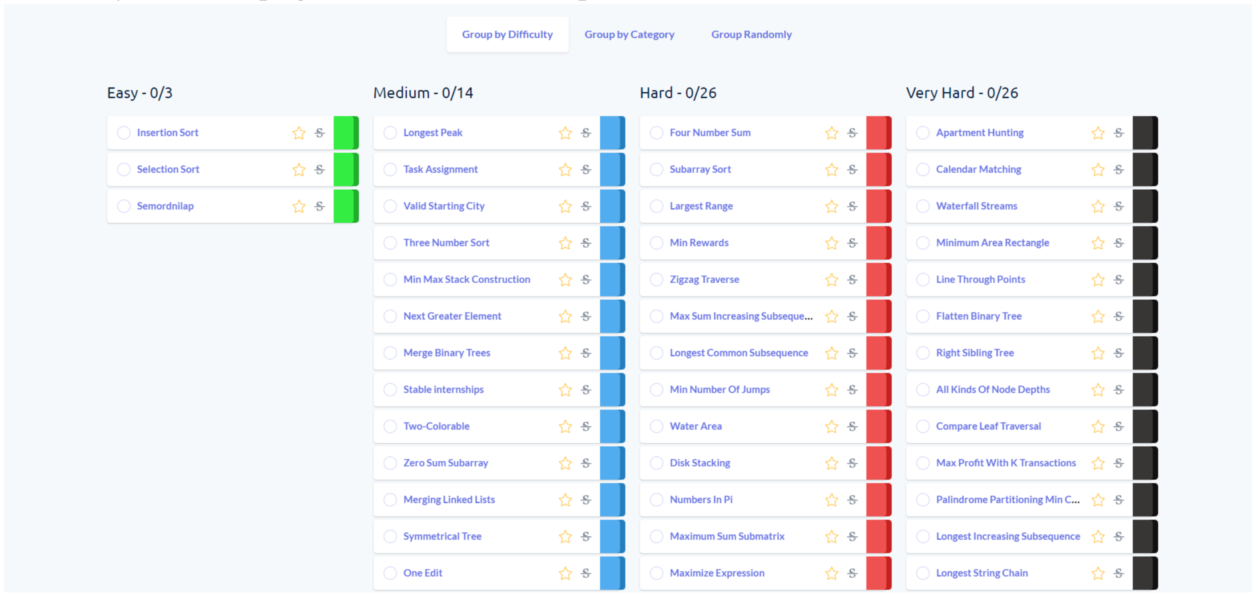
The IDE page



Click 'Run Code' to see program output.

Figure 2.3: The IDE page

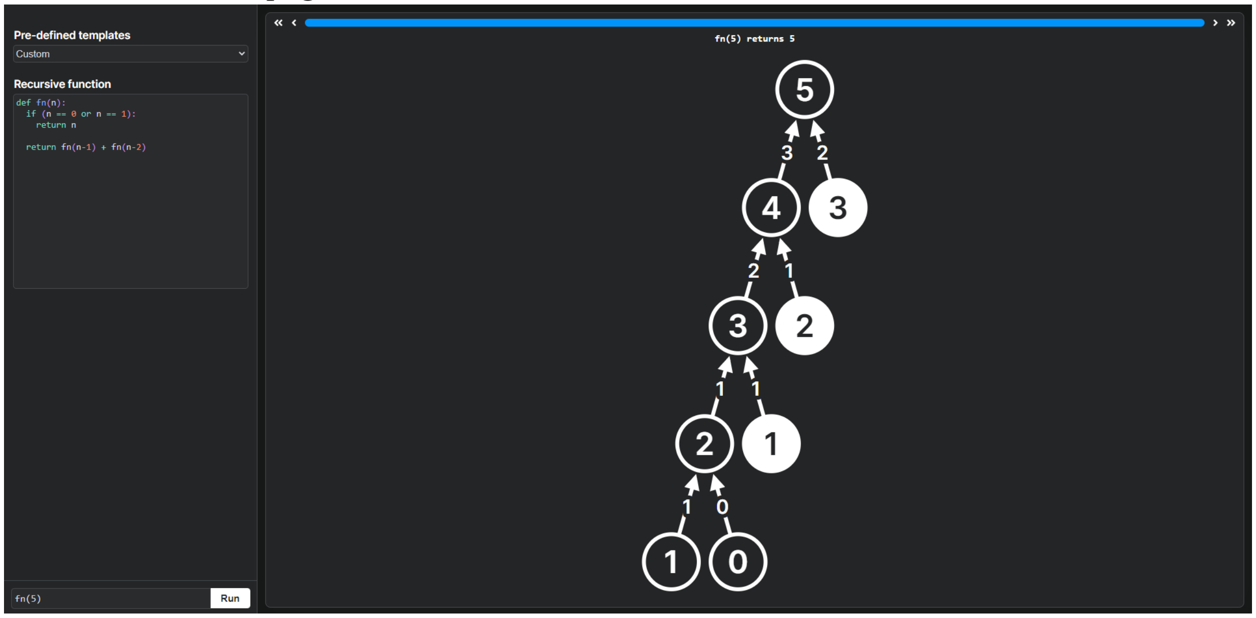
Portion of the home page where users select question to solve



Easy - 0/3	Medium - 0/14	Hard - 0/26	Very Hard - 0/26
Insertion Sort	Longest Peak	Four Number Sum	Apartment Hunting
Selection Sort	Task Assignment	Subarray Sort	Calendar Matching
Semordnilap	Valid Starting City	Largest Range	Waterfall Streams
	Three Number Sort	Min Rewards	Minimum Area Rectangle
	Min Max Stack Construction	Zigzag Traverse	Line Through Points
	Next Greater Element	Max Sum Increasing Subseque...	Flatten Binary Tree
	Merge Binary Trees	Longest Common Subsequence	Right Sibling Tree
	Stable internships	Min Number Of Jumps	All Kinds Of Node Depths
	Two-Colorable	Water Area	Compare Leaf Traversal
	Zero Sum Subarray	Disk Stacking	Max Profit With K Transactions
	Merging Linked Lists	Numbers In Pi	Palindrome Partitioning Min C...
	Symmetrical Tree	Maximum Sum Submatrix	Longest Increasing Subsequence
	One Edit	Maximize Expression	Longest String Chain

Figure 2.4: Questions Page

Recursion visualizer page



```
def fn(n):
  if (n == 0 or n == 1):
    return n

  return fn(n-1) + fn(n-2)
```

```
fn(5)
```

```
fn(5) returns 5
```

Figure 2.5. Recursion Visualizer Page

Path finder visualizer page

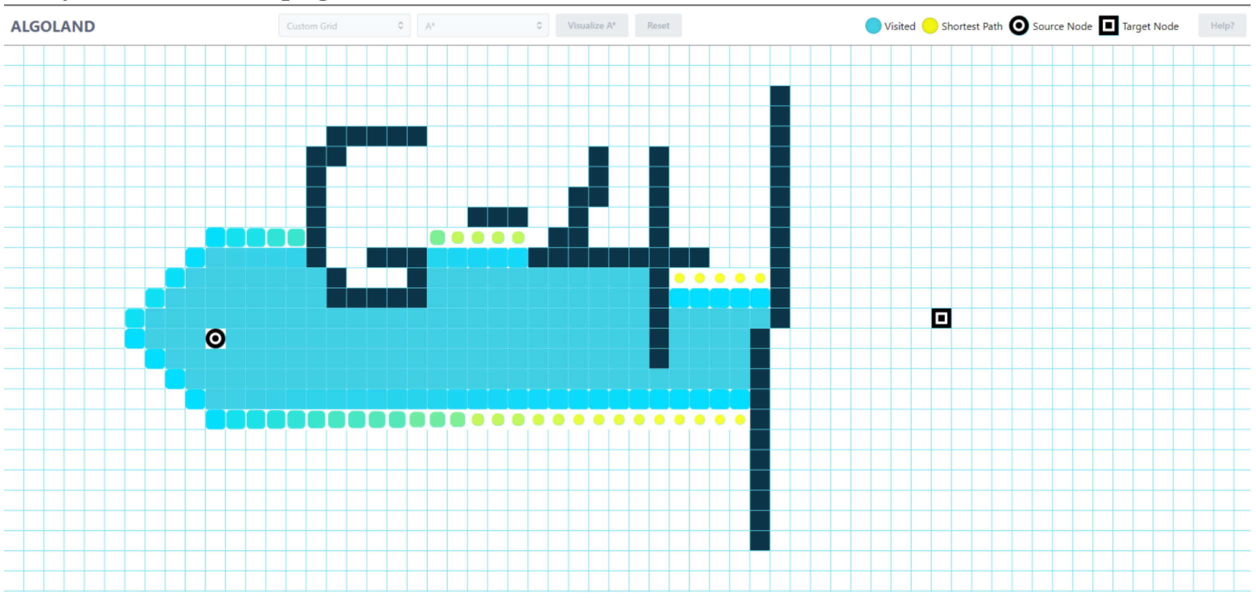
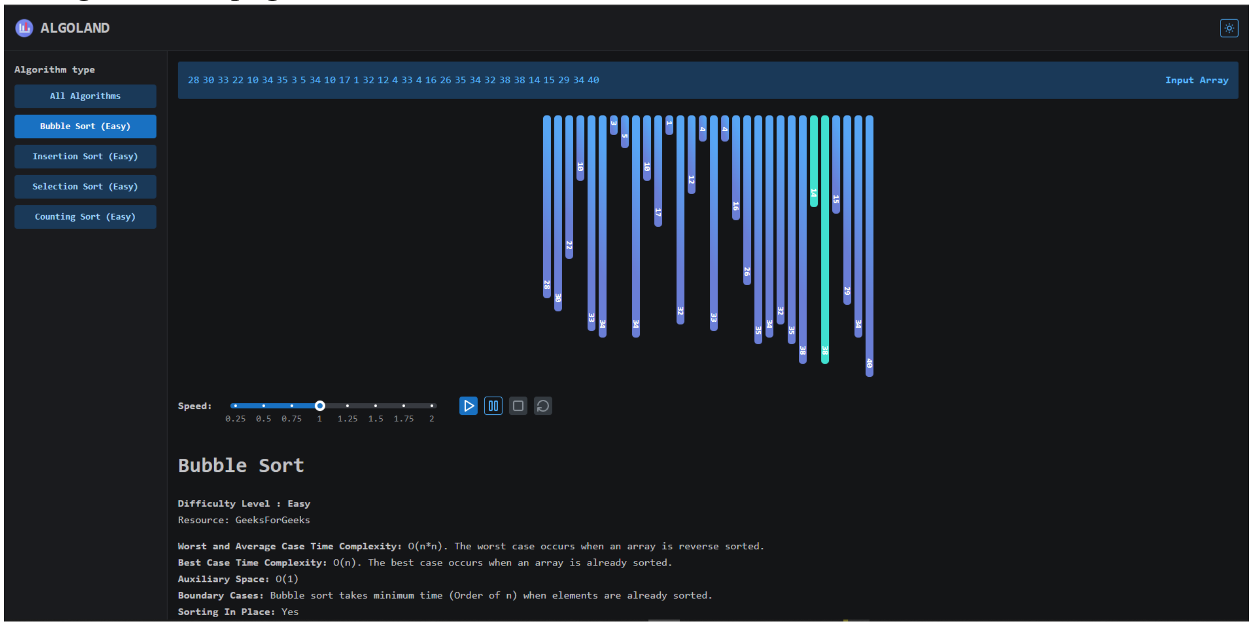


Figure 2.6. Path finder Visualizer Page

Sorting visualizer page



28 30 33 22 10 34 35 3 5 34 10 17 1 32 12 4 33 4 16 26 35 34 32 38 38 14 15 29 34 40

Bubble Sort

Difficulty Level : Easy
Resource: GeeksForGeeks

Worst and Average Case Time Complexity: O(n*n). The worst case occurs when an array is reverse sorted.
Best Case Time Complexity: O(n). The best case occurs when an array is already sorted.
Auxiliary Space: O(1)
Boundary Cases: Bubble sort takes minimum time (Order of n) when elements are already sorted.
Sorting In Place: Yes

Figure 2.7. Sorting Visualizer Page

2.8.2 Hardware Interfaces

The platform has no hardware interface requirements

2.8.3 Software Interfaces

The platform has no customer defined software interface requirements

2.8.4 Communications Interfaces

Since the platform is built to be used by users from different areas, an internet connection shall be used for communication.

2.9 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

2.9.1 Registration

Introduction	The system allows the user to register into the system.
Inputs	Users shall enter their information: - Full name - Email - Password
Processing	The system shall take the input and record it in the database
Outputs	The system shall notify users after registration and redirect to the desired page.
Error Handling	The system shall notify users upon failed requests

Table 2.1. Registration functional requirement

2.9.2 Log In

Introduction	The system allows the user to login into the system.
Inputs	Users shall enter their information: - Email - Password
Processing	The system shall take the input and check their validity.
Outputs	The system shall redirect to desired page if successful
Error Handling	The system shall notify users upon failed requests

Table 2.2. Login functional requirement

2.9.3 Run code

Introduction	The system allows the user to run code on the IDE.

Inputs	Users only need to write code in the IDE to solve a given question from the platform or while practicing on their own.
Processing	The system shall take the code and execute it.
Outputs	The system shall send the output of the code back to the user.
Error Handling	The system shall notify users upon failed requests.

Table 2.3. Run code functional requirement

2.9.4 Join Room

Introduction	The system allows the user to join another person's room if they have an invitation link.
Inputs	Users need to provide the invite link to a room.
Processing	The system shall take the link and check if it is valid.
Outputs	The system shall redirect the user to a specific room.
Error Handling	The system shall notify users upon failed requests.

Table 2.4. Join Room functional requirement

2.9.5 Leave Room

Introduction	The system allows the user to leave a room.
Inputs	No required inputs.
Processing	The system shall remove user from a room.
Outputs	The system shall redirect the user to home page.
Error Handling	The system shall notify users upon failed requests.

Table 2.5. Leave room functional requirement

2.9.6 Invite to Room

Introduction	The system allows the user to invite their friends to a common room.
Inputs	No required inputs.
Processing	The system shall generate an invite link.
Outputs	The system output the invitation link to the user who wants to invite.
Error Handling	The system shall notify users upon failed requests.

Table 2.6. Invite to room functional requirement

2.9.7 Filter Questions

Introduction	The system allows the user to filter question based on difficulty, category, and randomly.
Inputs	No required inputs.
Processing	The system shall re-arrange the questions based on the given filter criteria.
Outputs	No output.
Error Handling	The system shall notify users upon failed requests.

Table 2.7. Filter Questions functional requirement

2.9.8 Select Question

Introduction	The system allows the user to select a question to solve.
Inputs	No required inputs.
Processing	No required processing.
Outputs	The system shall redirect the user to the IDE page with the question as a prompt.

Error Handling	The system shall notify users upon failed requests.

Table 2.8. Select question functional requirement

2.9.9 Select Topic

Introduction	The system allows the user to select a topic to learn.
Inputs	No required inputs.
Processing	No required processing.
Outputs	The system shall redirect the user to a page where they can learn the topic of choice.
Error Handling	The system shall notify users upon failed requests.

Table 2.9. Select topic functional requirement

2.9.10 Visualize Topic

Introduction	The system allows the user to select a topic to visualize.
Inputs	No required inputs.
Processing	No required processing.
Outputs	The system shall redirect the user to a page where they can visualize the topic of choice.
Error Handling	The system shall notify users upon failed requests.

Table 2.10. Visualize topic functional requirement

2.9.11 Edit visualization parameters

Introduction	The system allows the user to edit and modify the visualization environment. Detailed requirements for every visualization environment listed below.
Inputs	Inputs may vary based on the topic selected.
Processing	Processing may vary based on the topic selected.
Outputs	Output may vary based on the topic selected.
Error Handling	The system shall notify users upon failed requests.

Table 2.11. Edit visualization parameters functional requirement

2.9.12 Sorting Visualizer

Introduction	The system allows the user to view and interact with the sorting visualizer.
Inputs	User may provide an input for the visualizer if they don't want to use the randomly generated value.
Processing	The system shall sort the given input.
Outputs	The system shall visualize the process of sorting the given input.
Error Handling	The system shall notify users upon failed requests.

Table 2.12. Sorting visualizer functional requirement

2.9.13 Path finder Visualizer

Introduction	The system allows the user to view and interact with the path finder visualizer.
Inputs	Users pass input by drawing on the provided environment. Input consists of starting and ending point, walls (paths), and search algorithm type.
Processing	The system shall find the best path to reach the end destination.

Outputs	The system shall visualize the process of finding the best path.
Error Handling	The system shall notify users upon failed requests.

Table 2.13. Path finder visualizer functional requirement

2.9.14 Recursion Visualizer

Introduction	The system allows the user to view and interact with the recursion visualizer.
Inputs	Users provide a recursive function as a code in the specified field or choose from an already specified functions.
Processing	The system shall create a recursive tree that will be used to visualize the invoking steps.
Outputs	The system shall visualize the process of invoking the function with its arguments and return values.
Error Handling	The system shall notify users upon failed requests.

Table 2.14. Recursion visualizer functional requirement

2.9.15 Favorite Question

Introduction	The system allows the user to add a question as a favorite.
Inputs	No required inputs.
Processing	The system shall add the question to the user’s favorite list.
Outputs	The system shall display the question as favorite.
Error Handling	The system shall notify users upon failed requests.

Table 2.15. Favorite question functional requirement

2.9.16 Add Discussion to a Question

Introduction	The system allows the user to start a discussion on a question.
Inputs	User must submit: - title of the discussion - tag - description
Processing	The system shall save the new discussion to the database.
Outputs	The system shall display the discussion inside the discussions tab of the question.
Error Handling	The system shall notify users upon failed requests.

Table 2.16. Add discussion functional requirement

2.9.17 Add Comment to a Discussion

Introduction	The system allows the user to add a comment to a discussion.
Inputs	User must submit: - Description of the comment
Processing	The system shall save the new comment to the database.
Outputs	The system shall display the comment inside the comment tab of the discussion.
Error Handling	The system shall notify users upon failed requests.

Table 2.17. Add Comment functional requirement

2.9.18 Mark a question as complete

Introduction	The system allows the user to mark a question as completed.
Inputs	No required inputs.

Processing	The system shall save the completed question add it to your progress.
Outputs	The system shall display the question as completed.
Error Handling	The system shall notify users upon failed requests.

Table 2.18. Mark question as completed functional requirement

2.10 Use Cases

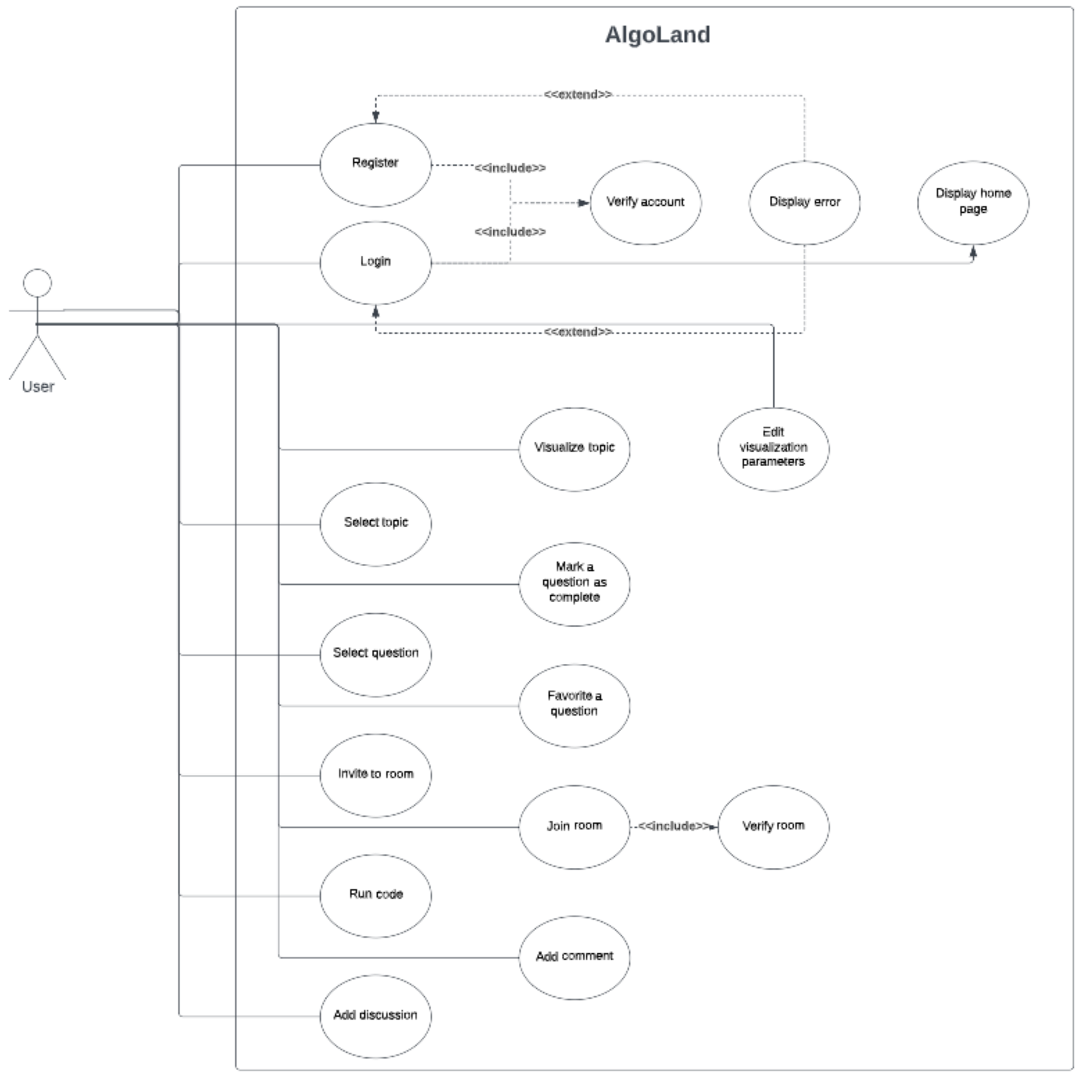


Figure 2.8. Use case diagram

2.10.1 Register

Goal	To register the users
Primary Actor	Users
Included Use Case	The system validates the user's credentials.

Extended Use Case	The systems displays registration error
Preconditions	Users visit the website
Success Scenario	1. The user visits the website 2. The user clicks the register button 3. The registration window or page displays the full name, email, password fields. 4. The user fills out the required fields 5. The user submits their credentials 6. The system validates the user's credentials 7. (Extension : The system displays "Register Error " message) 8. The system verifies the user 9. The system sends a successful registration message 10. The system displays the user homepage

Table 2.19. Register use case.

2.10.2 Login

Goal	To Login to the system
Primary Actor	Users
Included Use Case	The system validates the user's credentials.
Extend Use Case	The systems displays login error
Preconditions	The user has an account.
Post-conditions	The system shall display the homepage.
Success Scenario	1. The user clicks the login button 2. The login page displays the email and password fields. 3. The user enters email and password

	4. The user submits their credential 5. The system triggers to validate the user credential 6. (Extension: The system displays a "Login Error" message) 7. The system verifies the user 8. The system displays homepage

Table 2.20. Login use case

2.10.3 Select topic

Goal	To select a topic to learn
Primary Actor	Users
Preconditions	The user must be logged in.
Post-conditions	The system shall display the learning page related with the topic.
Success Scenario	1. The user selects the topic they want to learn 2. The system displays the appropriate learning page

Table 2.21. Select topic use case

2.10.4 Visualize topic

Goal	To visualize a selected topic
Primary Actor	Users
Preconditions	The user must be logged in.
Post-conditions	The system shall display the visualization page related with the topic.
Success Scenario	1. The user clicks the visualize topic button 2. The system displays the appropriate visualization page

Table 2.22. Visualize topic use case

2.10.5 Edit visualization parameters

Goal	To edit and configure the visualizer
Primary Actor	Users
Preconditions	The user must be logged in.
Post-conditions	The system shall re-configure the visualization environment.
Success Scenario	1. The user edits and configures the environment 2. The system re-adjust to the new configurations

Table 2.23. Edit visualization parameters use case

2.10.6 Select questions

Goal	To select a question to solve
Primary Actor	Users
Preconditions	The user must be logged in.
Post-conditions	The system shall display the IDE page with the question as a prompt.
Success Scenario	1. The user selects a question to solve 3. The system displays the IDE page with the question as a prompt.

Table 2.24. Select questions use case

2.10.7 Mark question as complete

Goal	To mark a question as complete
Primary Actor	Users
Preconditions	The user must be logged in.
Post-conditions	The system shall mark the question as complete.

Success Scenario	1. The user clicks the “Mark as complete” button 2. The system shall mark the question as complete.

Table 2.25. Mark question as completed use case

2.10.8 Mark questions as favorite

Goal	To mark a question as favorite
Primary Actor	Users
Preconditions	The user must be logged in.
Post-conditions	The system shall mark the question as favorite.
Success Scenario	1. The user clicks the Favorite icon 2. The system shall mark the question as favorite.

Table 2.26. Mark question as favorite

2.10.9 Invite to room

Goal	To invite users to a room
Primary Actor	Users
Preconditions	The user must be logged in.
Post-conditions	The system shall generate an invitation link.
Success Scenario	1. The user clicks the “Copy room link” button 2. The system shall generate an invitation link.

Table 2.27. Invite to room use case

2.10.10 Join room

Goal	To join another person’s room
Primary Actor	Users
Included Use Case	The system validates if room exists.

Preconditions	The user must be logged in.
Success Scenario	1. The user clicks “Join room” button 2. The user adds room link in the provided space 3. The system verifies the room 4. The system displays the IDE page with the user who invited the current user.

Table 2.28. Join room use case

2.10.11 Run code

Goal	To run code for a given question
Primary Actor	Users
Preconditions	The user must be logged in.
Post-conditions	The system shall take the code and run it.
Success Scenario	1. The user clicks the “Run code” button 2. The system shall take the code and run it. 3. The system returns the output for the execution

Table 2.29. Run code use case

2.10.12 Add discussion

Goal	To add discussion to a particular question
Primary Actor	Users
Preconditions	The user must be logged in.
Post-conditions	The system shall save the new discussion.
Success Scenario	1. The user clicks the “Add discussion” button 2. The system shall display a page to add discussion. 3. The system adds the discussion to the database

Table 2.30. Add discussion use case

2.10.13 Add comment

Goal	To add comment to a particular discussion
Primary Actor	Users
Preconditions	The user must be logged in.
Post-conditions	The system shall save the new comment.
Success Scenario	1. The user clicks the “Add comment” button 2. The system shall display a tab to add discussion. 3. The system adds the comment to the database

Table 2.31. Add comment use case

2.11 Non-Functional Requirements

2.11.1 Performance

The application’s load time should be no more than 5 seconds, given internet access with an adequate bandwidth. As for the implementation, since we are building the project using the React framework, it is expected to run smoothly across all supported browsers.

2.11.2 Reliability

Users generally always have access to their data and can do actions enabled by the program. We will evaluate and put better methods into practice as we develop the application to lower the error margin. Incompatibility between program features and network concerns should be anticipated.

2.11.3 Availability

We expect our platform to have an availability of three nines.

2.11.4 Security

In terms of security, we'll offer user authentication to give each user access to a secure environment and user permission to prevent anyone from accessing information and content that is outside of their privilege.

2.11.5 Maintainability

In the event that a feature update, bug fix, or general maintenance is required, we will minimize the downtime.

2.11.6 Portability

The platform will be very portable with responsive designs.

2.12 Inverse Requirements

The system will not verify users code for a particular question, meaning it will not test the code with additional test cases, it is up to the user to identify whether the solution is correct or not.

2.13 Design Constraints

The only constraint is internet connection.

2.14 Logical Database Requirements

The platform will use PostgreSQL as a database.

Database Table	Attributes	Data Types	Description
User	-id -fullName -gender -birthDay -location -email -password -github -linkedin	UUID Text Text Text Text Text Text Text Text	This table holds the information for a user.
Question	-id -description -difficulty -title -topicId	UUID Text Number Text UUID	This table holds the information for a question.
Topic	-id -title -description -visulizerId	UUID Text Text Text	This table holds the information for a topic.
Discussion	-id -userId	UUID UUID	This table holds the information for a discussion.

	-questionId -tag -title -description -seenCount -upVoteCount -downVoteCount	UUID Array<Enum> Text Text Number Number Number	
Comment	-id -userId -discussionId -text	UUID UUID UUID Text	This table holds the information for a comment.
Favorite	-id -questionId -userId -questionTitle	UUID UUID UUID Text	This table holds the information for a favorite question.
Solution	-id -userId -questionId -topicId -questionTitle -topicTitle -solutionCode -difficulty	UUID UUID UUID UUID Text Text Text Number	This table holds the information of user's progress.

Table 2.32. Logical Database Requirements

2.15 Other Requirements

Training-related Requirements
The technology will be simple to use and will not require any training.

Packaging Requirements
There won't be any packing specifications. It only requires an internet connection and a device to run on.

Legal Requirements
Not required

2.16 Change Management Process

All members of the team must agree to any requests to change the project's requirements and scope. Changes won't be made until the majority of the team approves of them. In this case, team members must reflect the changes in the SRS document and note the date of modification in the file. This modification request must be sent to the team by the client or anybody else besides the team. Any time one of these requests is made, the team will assess its viability while taking into account the time limits and structural limitations of the current modules, and will then develop an implementation plan. The group will then carry on implementing the new specifications.

Chapter 3: SYSTEM DESIGN

3.1 General Overview

CodeLand is an online platform designed to help students learn about Data Structures and Algorithms in a more interactive and intuitive way. The platform aims to eliminate the difficulties associated with understanding the concepts of data structures and algorithms by providing user-configurable graphic implementations of the majority of the algorithms and concepts such as Recursion, Tree, and Graph. Additionally, the platform will provide a selected set of questions on certain subjects to give students hands-on practice. The platform will also have detailed descriptions of both the subject and answers to the questions.

One of the key features of CodeLand is the built-in code playground, or Integrated Development Environment, which supports almost every language. This allows students to easily test their code and receive feedback in real-time. The platform also enables pair-programming, allowing students to collaborate and learn from one another.

The system uses micro service architecture which is a software architecture pattern that structures an application as a collection of small, independent services that communicate with each other using a lightweight protocol, typically HTTP/REST. This approach allows for greater flexibility, scalability, and maintainability of the system.

Overall, CodeLand is a comprehensive and interactive platform for students who want to learn about Data Structures and Algorithms. With its visualizations, practice questions, code playground, and pair-programming capabilities, it aims to make the learning experience more engaging and effective.

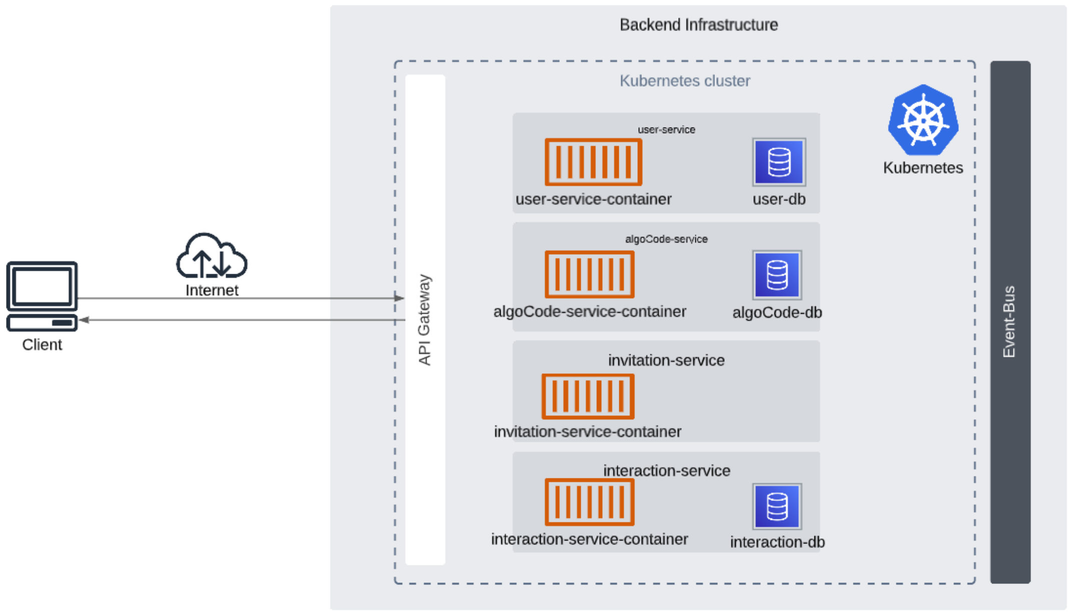


Figure 3.1. System Architecture Diagram.

3.2Development Methods & Contingencies

The Software Development Lifecycle, a series of procedures, is being used as a generic method for building the system (SDLC). Planning, analysis, design, implementation, and maintenance are the steps of the SDLC. This widely used method is appropriate for the project's methodology. Nevertheless, there may be more techniques to support the system analysis and design process.

Because the model is not rigid, the system will use an iterative approach. After reaching a goal, the system's overall effectiveness will be evaluated. Surveys and other strategies for information collecting will be utilized to help determine user requirements. Throughout the entire cycle of

software development. The core concept, however, will remain true to the users' original request. We'll keep collecting information from beta-testers to figure this out.

3.3 System Architecture

3.3.1 Subsystem decomposition

UML Component Diagram

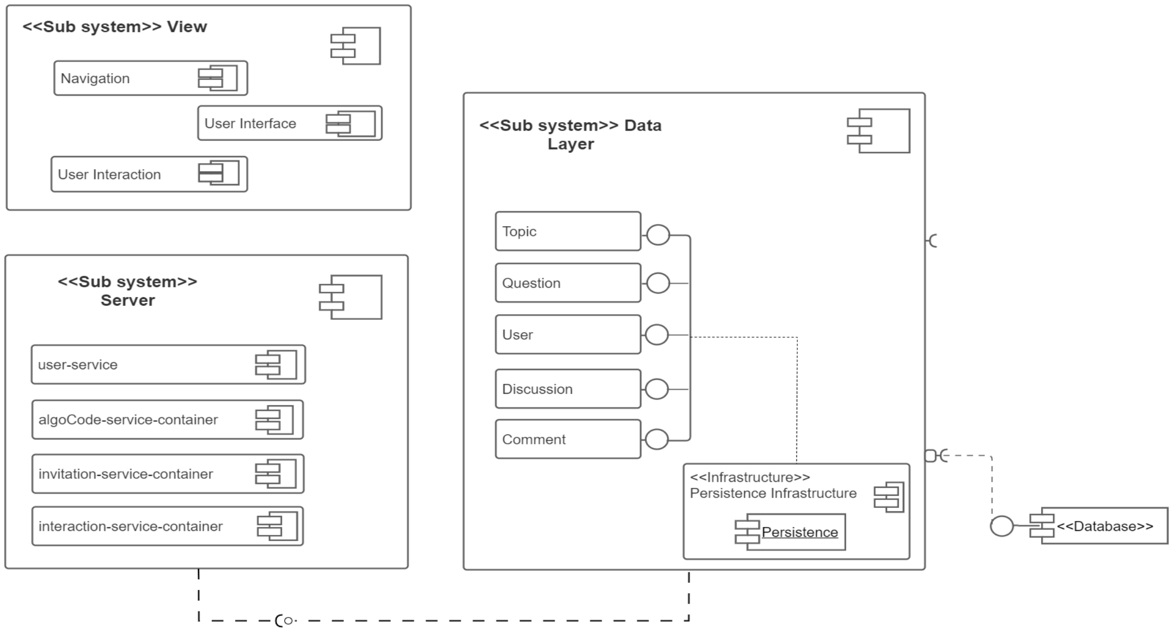

<<Sub system>> View
Navigation
User Interface
User Interaction
<<Sub system>> Server
user-service
algoCode-service-container
invitation-service-container
interaction-service-container
<<Sub system>> Data Layer
Topic
Question
User
Discussion
Comment
<<Infrastructure>> Persistence Infrastructure
Persistence
<<Database>>

Figure 3.2. UML Component Diagram

3.3.2 Hardware/software mapping

UML Deployment diagram

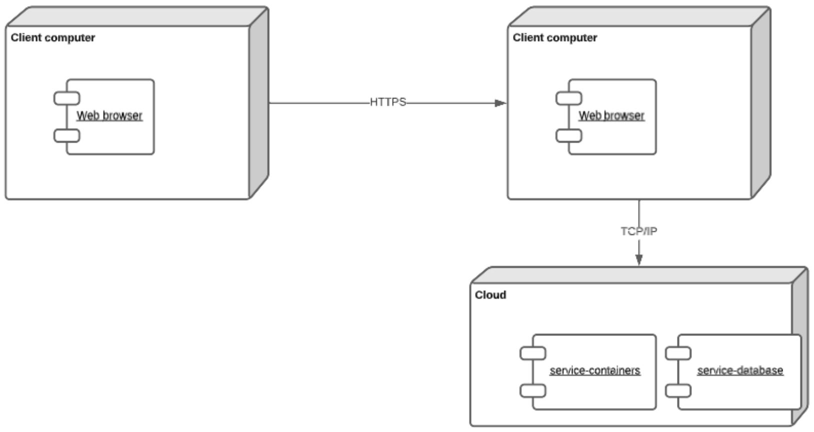
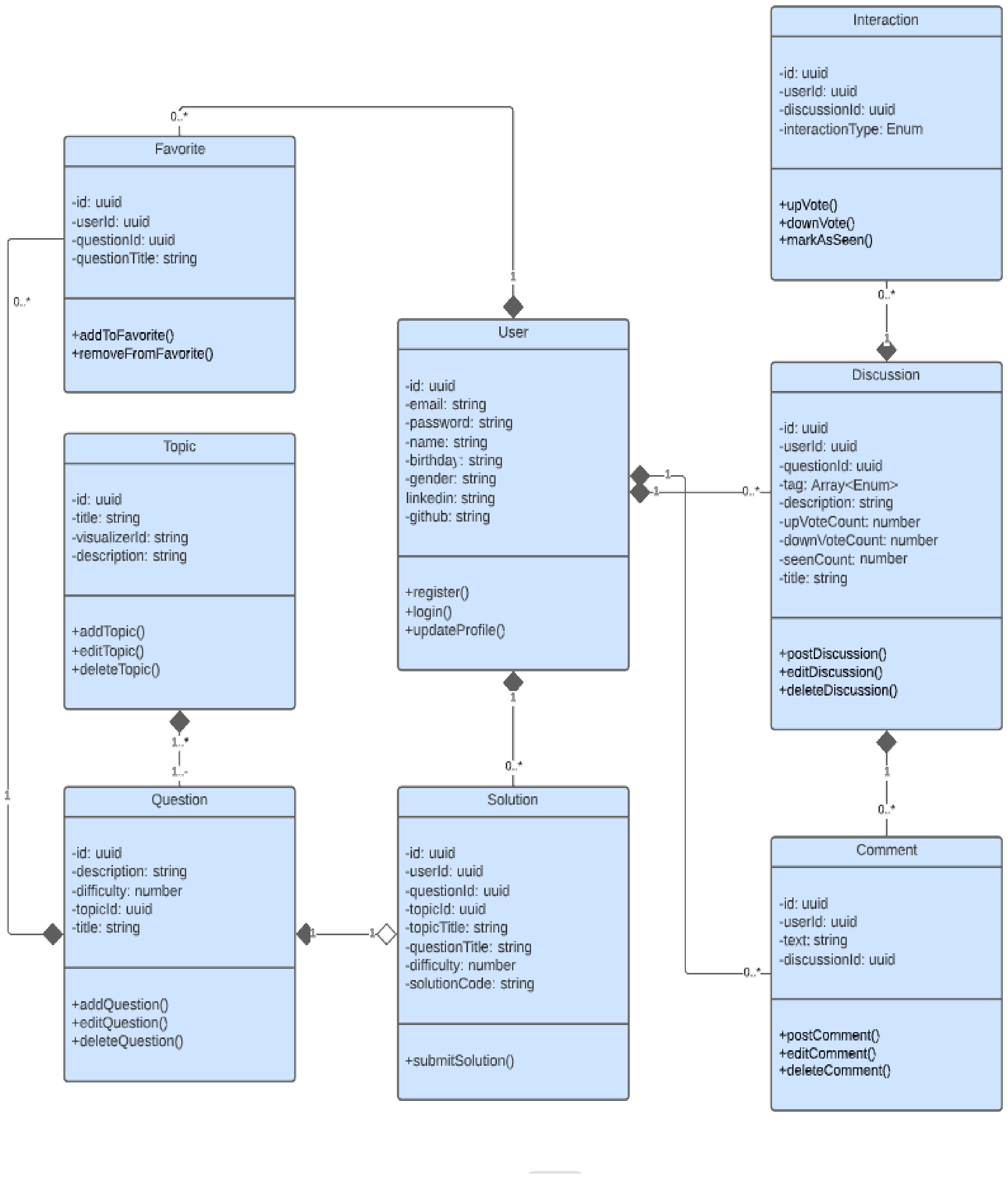


Figure 3.3. UML Deployment Diagram

3.4 Object Model

3.4.1 Class Diagram



3.4.2 Sequence Diagram

Show how processes operate with one another and in what order. Depict the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

3.4.2.1 User Registration Sequence Diagram

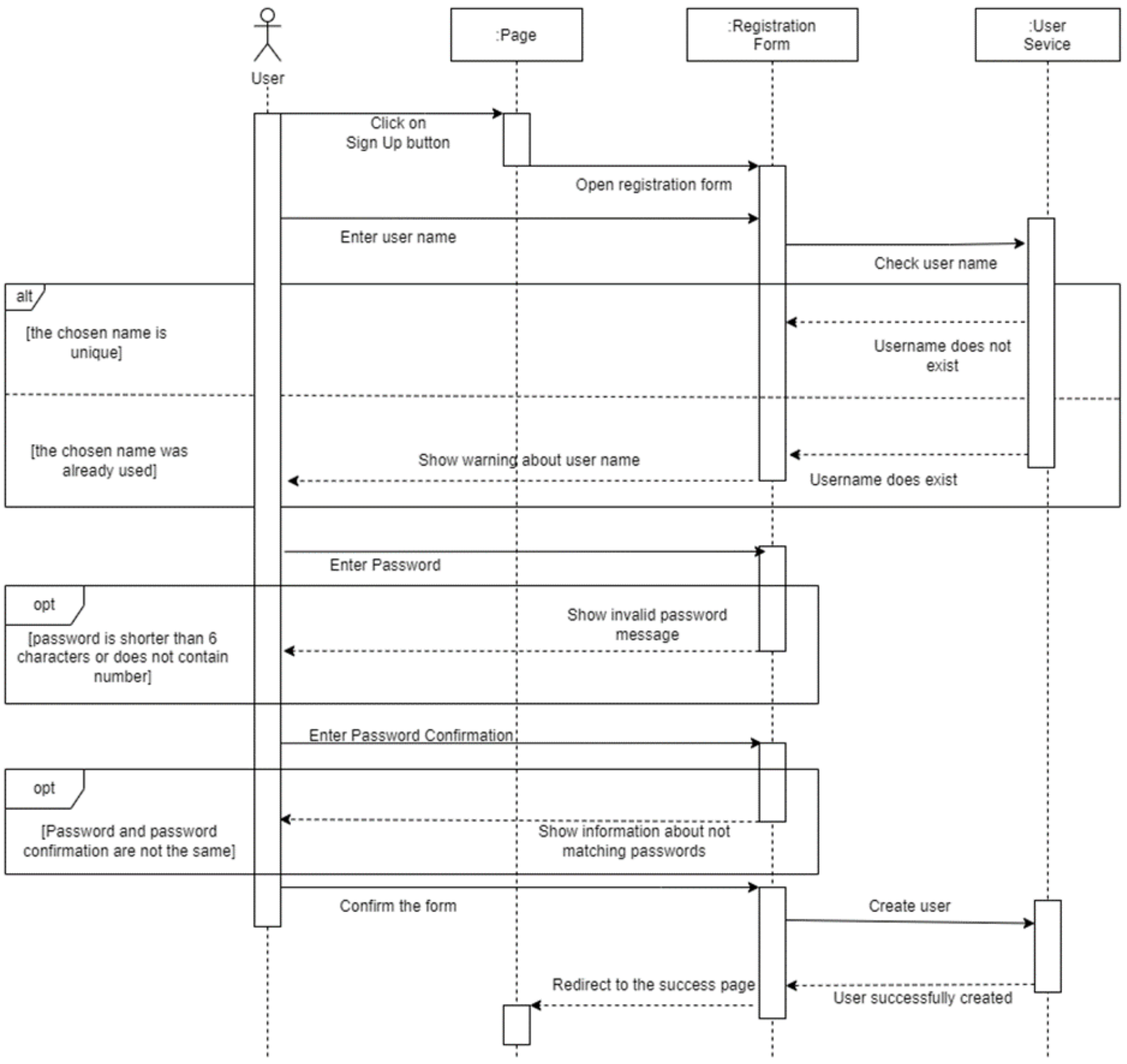

User
:Page
:Registration Form
:User Sevice
Click on Sign Up button
Open registration form
Enter user name
Check user name
alt
[the chosen name is unique]
Username does not exist
[the chosen name was already used]
Show warning about user name
Username does exist
Enter Password
opt
[password is shorter than 6 characters or does not contain number]
Show invalid password message
Enter Password Confirmation
opt
[Password and password confirmation are not the same]
Show information about not matching passwords
Confirm the form
Create user
Redirect to the success page
User successfully created

Figure 3.5. User Registration Sequence Diagram

3.4.2.2 User Login Sequence Diagram

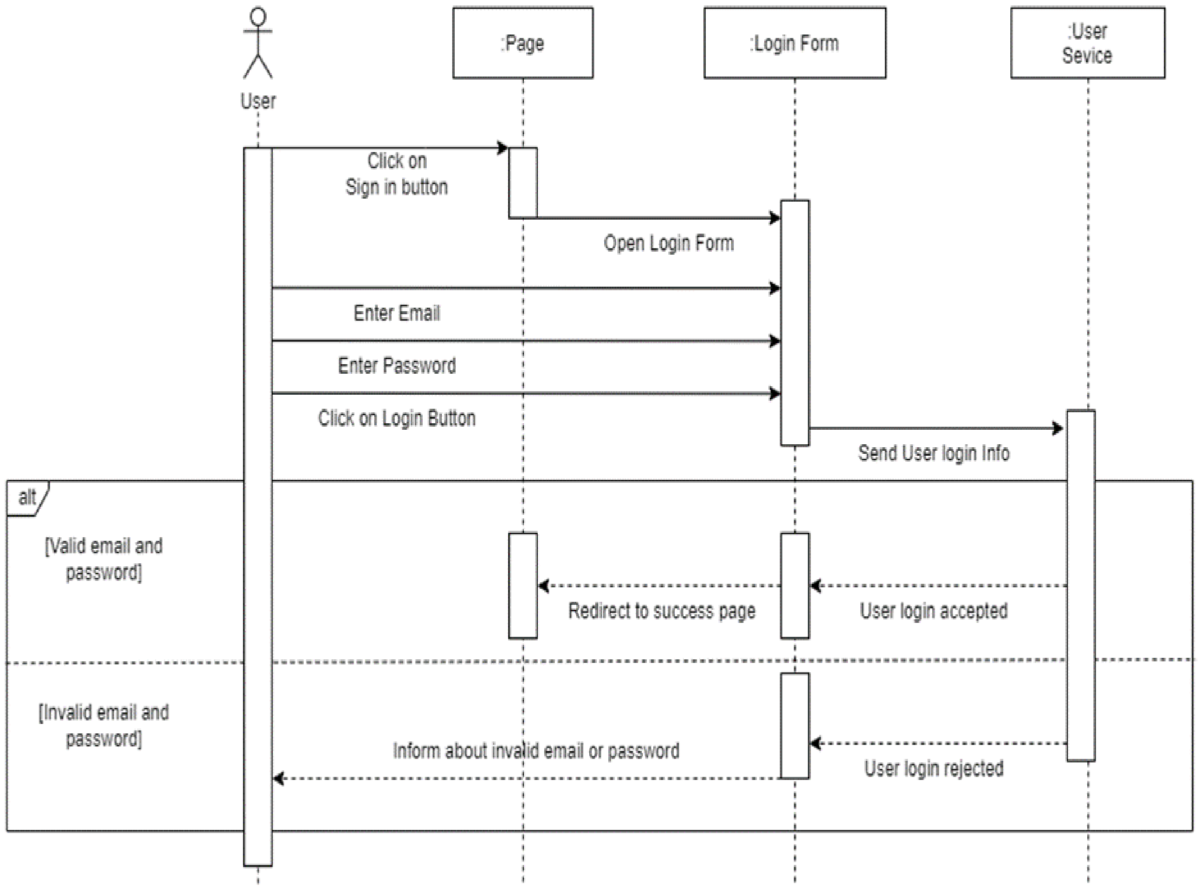


Figure 3.6. User Login Sequence Diagram

3.4.2.3 Mark Question as Complete Sequence Diagram

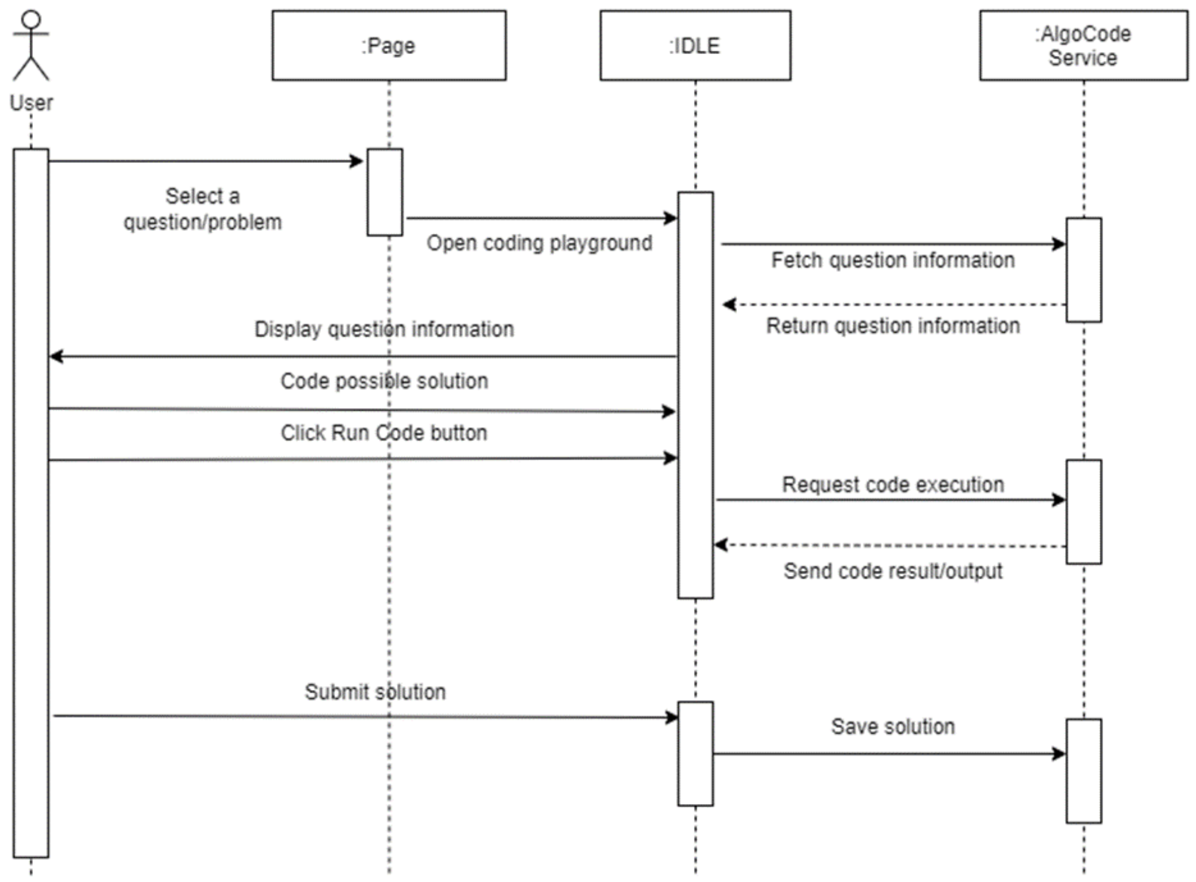


Figure 3.7. Mark question as Complete Sequence Diagram

3.4.2.4 Pair Programming Sequence Diagram

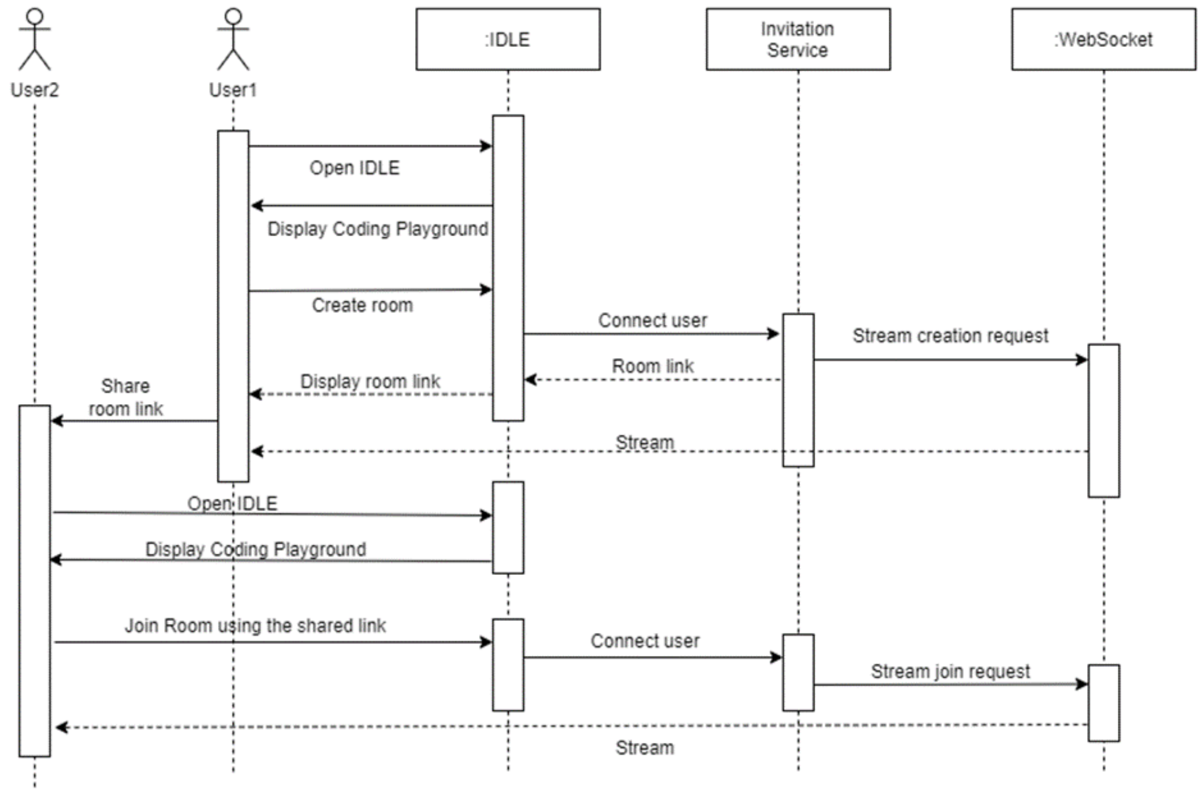


Figure 3.8. Pair Programming Sequence Diagram

3.4.2.5 Post Discussion Sequence Diagram

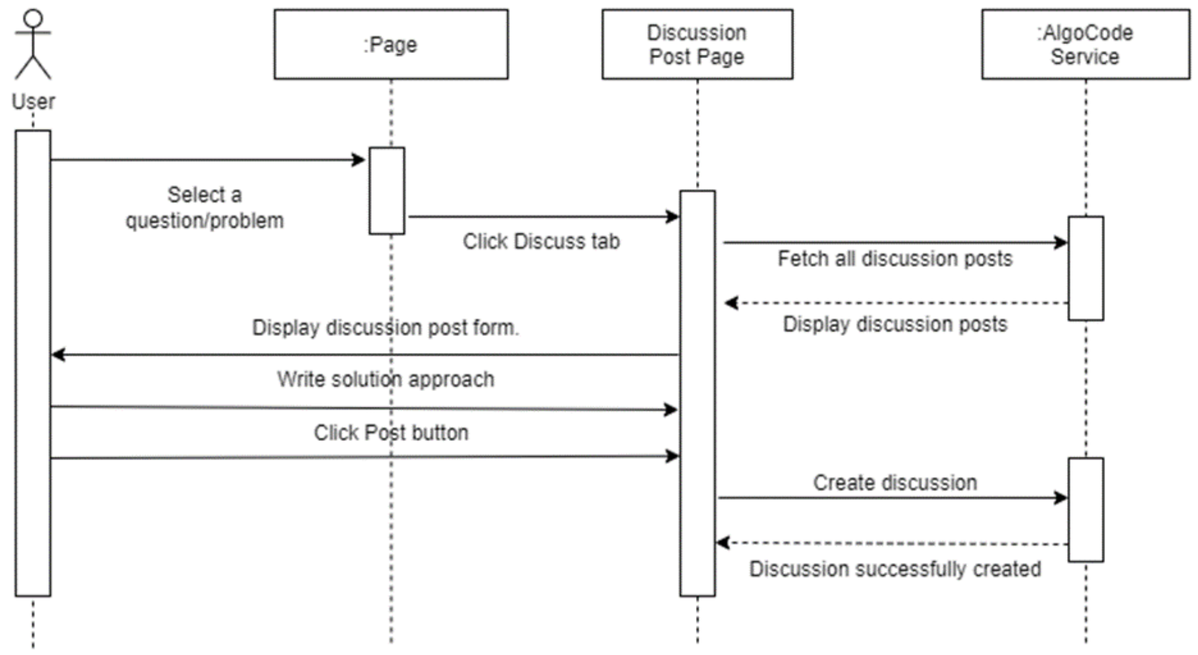


Figure 3.9. Post Discussion Sequence Diagram

3.4.2.6 Topic Visualizer Sequence Diagram

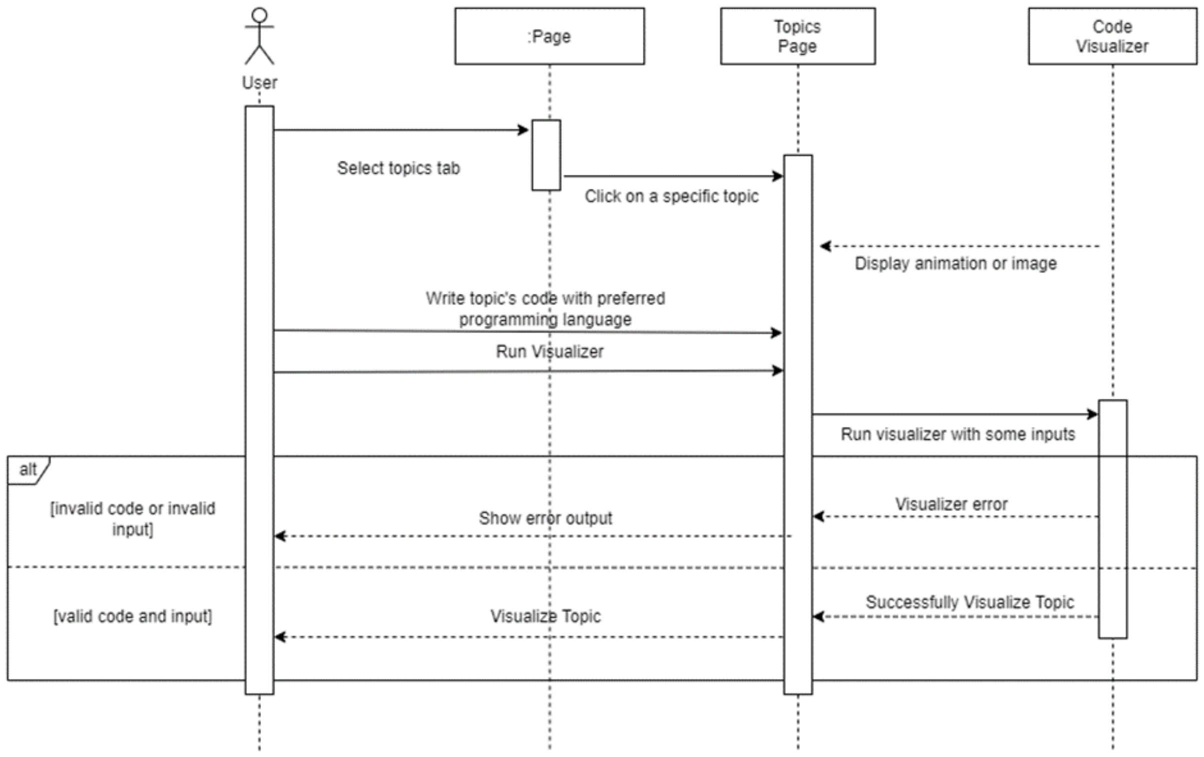

User
:Page
Topics Page
Code Visualizer
Select topics tab
Click on a specific topic
Display animation or image
Write topic's code with preferred programming language
Run Visualizer
Run visualizer with some inputs
alt
[invalid code or invalid input]
Visualizer error
Show error output
[valid code and input]
Successfully Visualize Topic
Visualize Topic

Figure 3.10. Topic Visualizer Sequence Diagram

3.4.2.7 User Update Profile Sequence Diagram

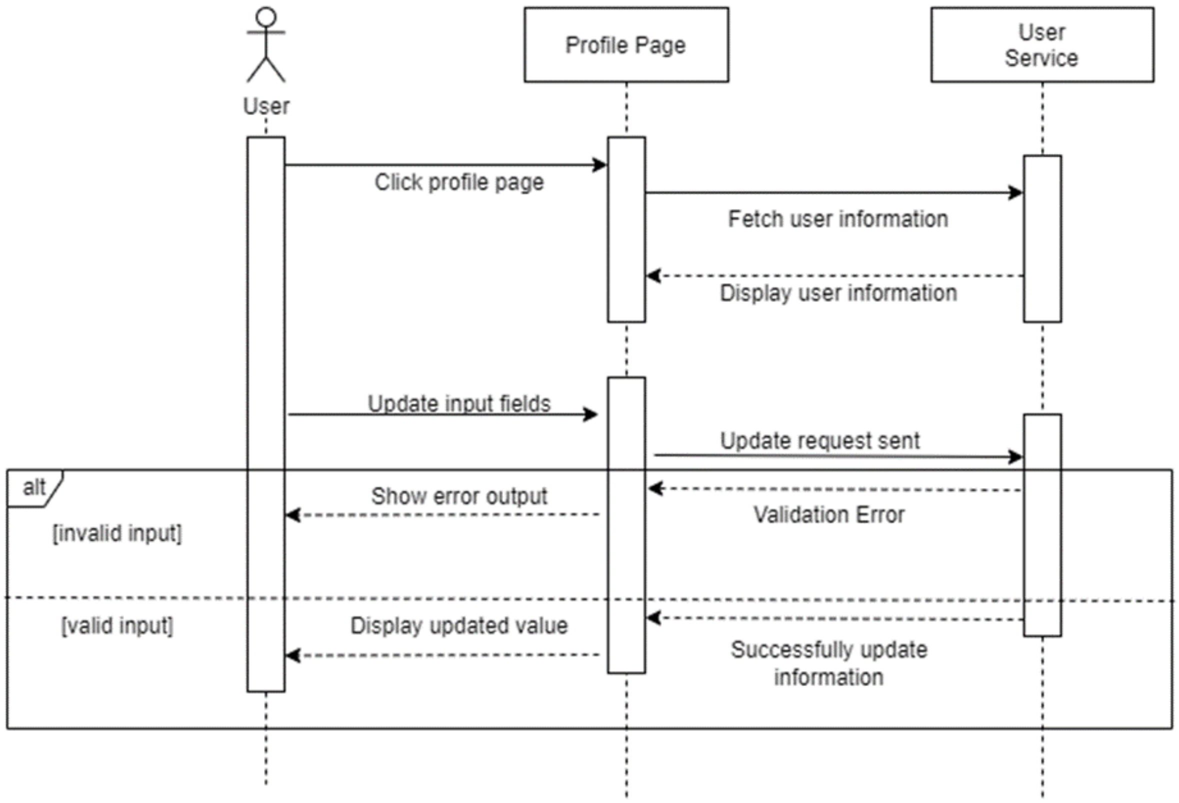


Figure 3.11. User Update Profile Sequence Diagram

3.5 Detailed Design

Table: 1 User Entity

User
- id: uuid - email : string - password : string - userName : string - linkedin : string - github: string - gender : string - birthDate: date - location: string

+ Registration()
+ Login()
+ updateProfile()

Table 3.1. User Entity

Table: 1.1 Attributes description for User class

Attribute	Type	Visibility	Invariant
id	uuid	Private	id <> NULL and must be a unique identifier
email	String	Private	email<> NULL and it must be a string, contain a valid email domain and @ symbol
password	String	Private	password<>NULL and must contain strings and numbers or special characters
userName	String	Private	userName<>NULL and must be a string
linkedin	String	Private	Linkedin can be NULL and must be a string
github	String	Private	Github can be NULL and must be a string
gender	Enum	Private	gender<>NULL and must be a male or female
birthDate	Date	Private	birthDate<>NULL and must contain a date object
location	String	Private	Location<>NULL and must be a string

Table 3.2. Attributes description for User class

Table: 1.2 Operation description for User class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
register	Public	void	fullName, email, password, userName, linkedin, github, gender, birthdate, location	The user has input a non-existent account name.	A user account with the name specified by the user or suggested by the system must be created
login	Public	void	userName or email and password	The account has to be already existing one and the user must provide the correct credentials	The user has to get logged in.
updateProfile()	Public	void	userId and the information needed to update.,	A valid input data has to be provided and the user must be signed in	The user account's profile must be updated to the new one set by the user. The system should display a success/failure message

Table 3.3. Operation description for User class

Table: 2 Question Entity

Question
- id: uuid - questionTitle : string - questionDescription : string - questionDifficulty : string - topicId : string

+ addQuestion()
+ editQuestion()
+ deleteQuestion()

Table 3.4. Question Entity

Table: 2.1 Attributes description for Question class

Attribute	Type	Visibility	Invariant
id	uuid	Private	id <> NULL and must be a unique identifier
questionTitle	String	Private	questionTitle<>NULL and must be a string
questionDescription	String	Private	questionDescription<>NULL and must be a string
questionDifficulty	String	Private	questionDifficulty<>NULL and must be a string
topicId	uuid	Private	topicId<>NULL and must be a uuid

Table 3.5. Attributes description for Question class

Table: 2.2 Operation description for Topic class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
addQuestion	public	void	questionId, questionTitle, questionDescription, questionDifficulty, topicId	The user must be logged in.	It will create a new question.
editQuestion	Public	void	questionId, questionTitle, questionDescription, questionDifficulty, topicId	The user must have a previous question and they have to be logged in.	It will edit a question.
deleteQuestion	Public	void	questionId	The user must have a previous question and they have to be logged in.	It will delete a question.

Table 3.6. Operation description for Topic class

Table: 3 Topic Entity

Topic
- id: uuid - topicTitle : string - topicDescription : string - visualizerId : uuid

+ addTopic () + editTopic () + deleteTopic ()

Table 3.7. Topic Entity

Table: 3.1 Attributes description for Topic class

Attribute	Type	Visibility	Invariant
id	uuid	Private	id <> NULL and must be a unique identifier
topicTitle	String	Private	topicTitle<>NULL and must be a string
topicDescription	String	Private	topicDescription<>NULL and must be a string
visualizerId	uuid	Private	visualizerId<>NULL and must be a uuid

Table 3.8. Attributes description for Topic class

Table: 3.2 Operation description for Topic class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
addTopic	public	void	topicId, topicTitle, topicDescription, visualizerId	The user must have to be logged in.	It will create a new topic.
editTopic	Public	void	topicId, topicTitle, topicDescription, visualizerId	The user must have a previous topic and they have to be logged in.	It will edit a topic.
deleteTopic	Public	void	topicId	The user must have a previous topic and they have to be logged in.	It will delete a topic.

Table 3.9. Operation description for Topic class

Table: 4 Discussion Entity

Discussion
- id: uuid - userId : uuid - questionId : uuid - tag : string - title : string - description : string - upVoteCount : int - downVoteCount : int - seenCount : int
+ postDiscussion() + editDiscussion () + deleteDiscussion ()

Table 3.10. Discussion Entity

Table: 4.1 Attributes description for Discussion class

Attribute	Type	Visibility	Invariant
id	uuid	Private	id <> NULL and must be a unique identifier
userId	uuid	Private	userId<>NULL and must be a uuid
questionId	uuid	Private	questionId<>NULL and must be a uuid
tag	String	Private	tag<>NULL and must be a string
title	String	Private	title<>NULL and must be a string
description	String	Private	description<>NULL and must be a string
upVoteCount	int	Private	upVoteCount<>NULL and must be a int
downVoteCount	int	Private	downVoteCount<>NULL and must be a int
seenCount	int	Private	seenCount<>NULL and must be a int

Table 3.11. Attributes description for Discussion class

Table: 4.2 Operation description for Discussion class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
PostDiscussion	public	void	userId, questionId, title, description	The user must have to be logged in.	It will create new discussion.
editDiscussion	Public	void	userId, questionId, title, description	The user must have a previous discussion and they have to be logged in.	It will edit a discussion.
deleteDiscussion	Public	void	userId, questionId	The user must have a previous discussion and they have to be logged in.	It will delete a discussion.

Table 3.12. Operation description for Discussion class

Table: 5 Comment Entity

Comment
- id: uuid - userId : uuid - commentText : string - discussionId : uuid

+ postComment()
+ editComment()
+ deleteComment()

Table 3.13. Comment Entity

Table: 5.1 Attributes description for comment class

Attribute	Type	Visibility	Invariant
id	uuid	Private	id <> NULL and must be a unique identifier
userId	uuid	Private	userId<>NULL and must be a uuid
commentText	String	Private	commentText<>NULL and must be a string
discussionId	uuid	Private	discussionId<>NULL and must be a uuid

Table 3.14. Attributes description for comment class

Table: 5.2 Operation description for comment class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
postComment	Public	void	userId, discussionId, commentText	The user must have to enter a comment into the comment section and they must have to be logged in.	It will post a comment for the discussion.
editComment	Public	void	userId, commentId, commentText	The user must have a previous comment and they have to be logged in.	It will edit a comment for the discussion.
deleteComment	Public	void	userId, commentId	The user must have a previous comment and they have to be logged in.	It will delete a comment for the discussion.

Table 3.15. Operation description for comment class

Table: 6 Favorite Entity

Favorite
- id: uuid - userId : uuid - questionId : uuid - questionTitle : string

+ addToFavorite()
+ removeFromFavorite()

Table 3.16. Favorite Entity

Table: 6.1 Attributes description for favorite class

Attribute	Type	Visibility	Invariant
id	uuid	Private	id <> NULL and must be a unique identifier
userId	uuid	Private	userId<>NULL and must be a uuid
questionId	uuid	Private	questionId<>NULL and must be a uuid

questionTitle	String	Private	questionTitle<>NULL and must be a string

Table 3.17. Attributes description for favorite class

Table: 6.2 Operation description for favorite class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
addToFavorite	public	void	userId, questionId, questionTitle	The user must have to be logged in.	The user will add a question into their favorite list.
removeFromFavorite	public	void	userId, questionId, questionTitle	The user must have to be logged in.	The user will remove a question from their favorite list.

Table 3.18. Operation description for favorite class

Table: 7 Solution Entity

Solution
- id: uuid - userId : uuid - questionId : uuid - solutionText : string
+ submitSolution()

Table 3.19. Solution Entity

Table: 7.1 Attributes description for solution class

Attribute	Type	Visibility	Invariant
id	uuid	Private	id <> NULL and must be a unique identifier
userId	uuid	Private	userId<>NULL and must be a uuid
questionId	uuid	Private	questionId<>NULL and must be a uuid
solutionText	String	Private	solutionText<>NULL and must be a string

Table 3.20. Attributes description for solution class

Table: 7.2 Operation description for solution class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
submitSolution	Public	void	userID, questionId, solution	The user must have to enter some solution or can't submit a null solution .	The solution will be submitted

Table 3.21. Operation description for solution class

Table: 8 Interaction Entity

Interaction
- id : uuid - userId : uuid - discussionId: uuid - interactionType : string

+ upVote()
+ downVote()
+ markUsSeen()

Table 3.22. Interaction Entity

Table: 8.1 Attributes description for interaction class

Attribute	Type	Visibility	Invariant
id	uuid	Private	id <> NULL and must be a unique identifier
userId	uuid	Private	userId<>NULL and must be a uuid
discussionId	uuid	Private	discussionId<>NULL and must be a uuid
interactionType	String	Private	solutionText<>NULL and must be a string

Table 3. 23. Attributes description for interaction class

Table: 8.2 Operation description for interaction class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
upVote	public	int	userId, discussionId, interactionType	The user must have to be logged in and press the upvote button.	It will upVote the discussion.
downVote	public	int	userId, discussionId, interactionType	The user must have to be logged in and press the downVote button.	It will downVote the discussion.
markUsSeen	Public	void	userID, questionId	The user must have to see the discussion.	The discussion seen count will be incremented.

Table 3.24. Operation description for interaction class

Chapter 4: Testing

4.1 Introduction

The purpose of this test plan document is to provide a comprehensive and structured approach to testing the Code Land website, ensuring that all features and functionalities are thoroughly tested and any issues or defects are identified and resolved before the website is launched. The test plan document outlines the testing strategy, objectives, scope, and approach for the website, and serves as a guide for the testing team to ensure that all aspects of the website are tested in a systematic and efficient manner. The primary objectives of the testing process are to ensure the accuracy and effectiveness of the algorithms, test the user interface and navigation, ensure the website's performance and security, and ensure the website's compatibility with different browsers, devices, and operating systems. The test plan document will be used by the testing team to plan, execute, and report on the testing process, and will be updated as necessary throughout the testing process to ensure that all testing objectives are met.

4.2 Features to be tested/not to be tested

5.2.1 Features to be tested

All the features of Code-Land websites which were defined in software requirement specs need to be tested.

Feature to be tested	**Reference**
Recursion Visualizer	FR: 01
Pair Programming	FR: 02
Run Solution	FR :03
user create discussion	FR:04
Admin create *Question*	FR:05
Admin create *Topic*	FR: 06
admin post acceptable solution	FR: 07

Table 5.1: Features to be tested

4.2.2 Features not to be tested

These feature are not be tested because they are not included in the software requirement specs

- External integrations with third-party services
- Advanced algorithm implementation not supported by the platform

4.3. Pass/Fail criteria

5.3.1 Pass criteria

User is able to perform assigned task and receives expected results

5.3.2 Fail Criteria

User is not able to perform assigned task or does not receive expected results

4.4. Approach/Strategy

We are going to use the following approach to test our project

- The testing method we are going to use is Black box testing method
- Cover as much specified behavior as possible
- Based on a description of the software(specification)

4.5 Test cases with specifications

Table 4.2: Test case specification for Recursion Visualizer

Name: Recursion Visualizer				
Purpose: to visualize proper recursion algorithm				
Test Data= Custom code(Empty, Valid, Invalid)				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty code	“Empty code”	No code	“Empty code”	Pass
Valid Code	“Submit code”	Valid code	“Submit code”	Pass
Invalid Code	“Code error”	Invalid code	“Code error”	Pass

Table 4.3: Test case specification for Pair programming

Name: Pair Programming				
Purpose: for pair programming over a particular question				
Test Data= Invite Key (Empty, Valid, Invalid)				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty Invitation key	“Empty key”	No invitation key	“Empty key”	Pass
Valid Invitation key	“Joins the room”	Valid invitation key	“Joins the room”	Pass
Invalid Invitation key	“Key error”	Invalid invitation key	“Key error”	Pass

Table 4.4: Test case specification for Run Solution

Name: Run Solution				
Purpose: to verify that user runs appropriate code				
Test Data= User code (Empty, Valid, Invalid)				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty code	“Empty code”	No code	“Empty code”	Pass
Valid Code	“Submit code”	Valid code	“Submit code”	Pass
Invalid Code	“Code error”	Invalid code	“Code error”	Pass

Table 4.5: Test case specification for user create discussion

Name: user create discussion				
Purpose: for creating a discussion properly				
Test Data= Discussion Text(Empty, Valid, Invalid)				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty item	"fill the required fields" error message displayed	empty item	"fill the required fields" error message displayed	Pass
Valid item	"creates a question"	any valid item	"creates a question"	Pass
Invalid item	"invalid input" error message displayed	any invalid item	"invalid input" error message displayed	Pass

Table 4.6: Test case specification for admin create question

Name: admin create question				
Purpose: for creating a question properly				
Test Data= Question Text(Empty, Valid, Invalid)				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty item	"fill the required fields" error message displayed	empty item	"fill the required fields" error message displayed	Pass
Valid item	"creates a question"	any valid item	"creates a question"	Pass
Invalid item	"invalid input" error message displayed	any invalid item	"invalid input" error message displayed	Pass

Table 4.7: Test case specification for admin create topic

Name: admin create topic				
Purpose: for creating a topic properly				
Test Data= Topic Text(Empty, Valid, Invalid)				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty item	“fill the required fields” error message displayed	empty item	“fill the required fields” error message displayed	Pass
Valid item	“creates a topic”	any valid item	“creates a topic”	Pass
Invalid item	“invalid input” error message displayed	any invalid item	“invalid input” error message displayed	Pass

Table 4.8: Test case specification for admin post acceptable solution

Name: admin post acceptable solution				
Purpose: for posting acceptable solution				
Test Data= Solution Text(Empty, Valid, Invalid)				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty item	“fill the required fields” error message displayed	empty item	“fill the required fields” error message displayed	Pass
Valid item	“Posts a solution”	any valid item	“Posts a solution”	Pass
Invalid item	“invalid input” error message displayed	any invalid item	“invalid input” error message displayed	Pass

Chapter 5: User Manual

5.1 scope

The manual covers the following:

- How to start the system
- How to learn specific topic
- How to visualize each topic
- How to pair program with peers
- How to post a discussion
- How admin manage question
- How admin manage topic
- How admin manage Solutions

5.2 Installation and configuration

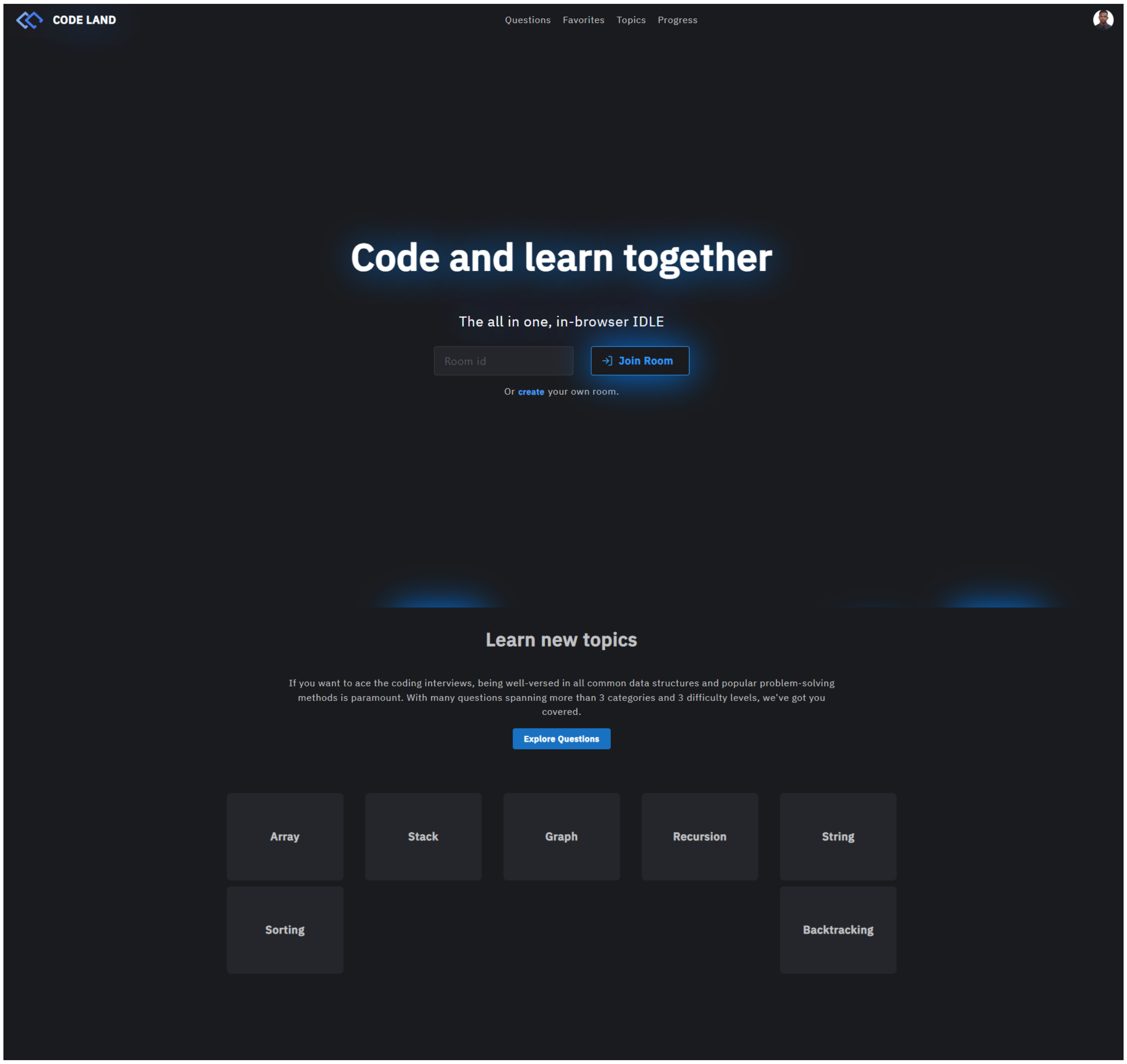
This is a website, thus no installation is required. To access the website, one will need to go to the following URL: https://code-land.vercel.app

5.3 How to Operate the system

5.3.1 User

A. List of Topics

● User can view list of topics in the Home page of CodeLand



Code and learn together

The all in one, in-browser IDLE

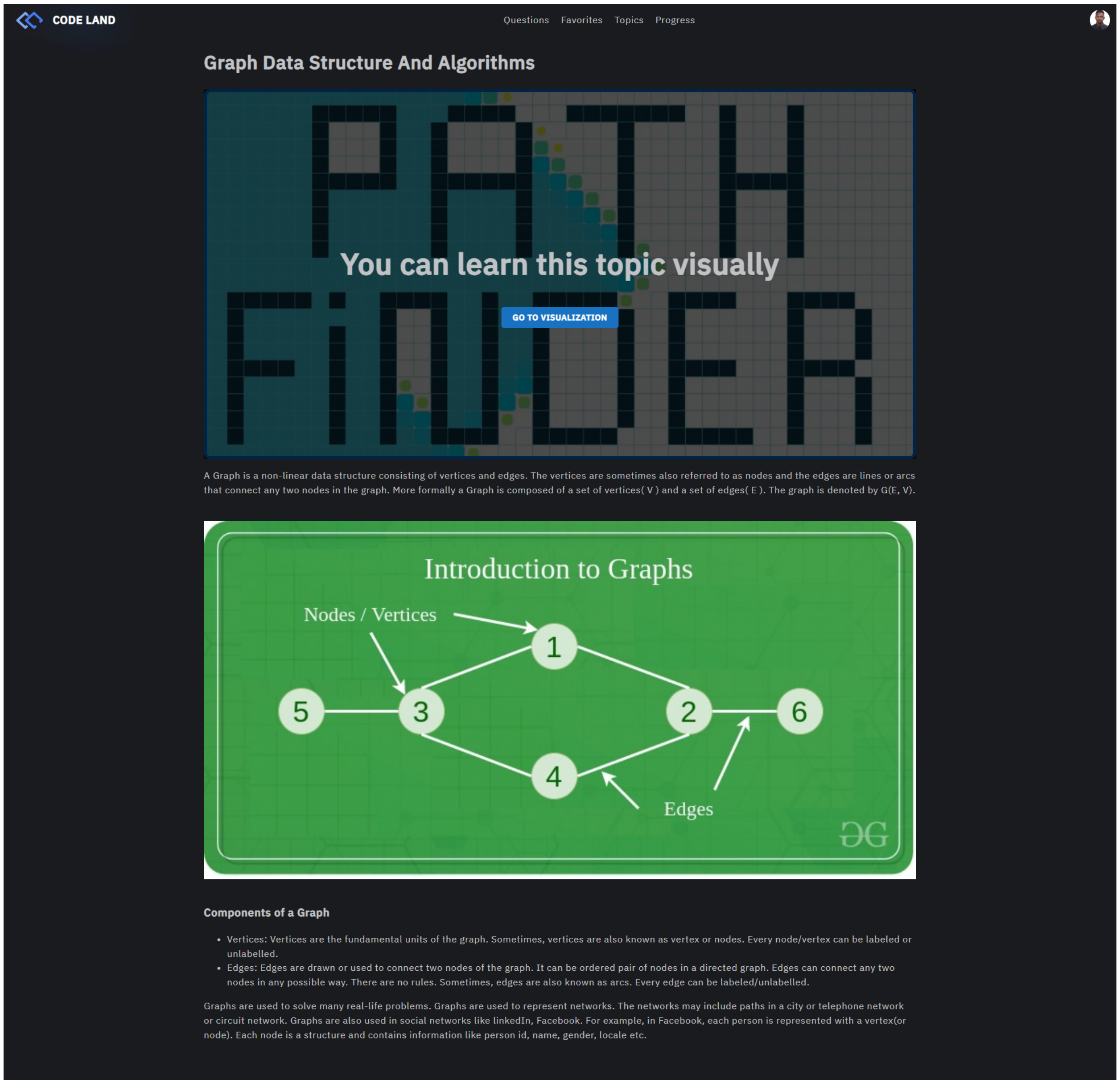
Or create your own room.

Learn new topics

If you want to ace the coding interviews, being well-versed in all common data structures and popular problem-solving methods is paramount. With many questions spanning more than 3 categories and 3 difficulty levels, we've got you covered.

B. Topics page

● Users can view each topic page in CodeLand by pressing respective topic button



Graph Data Structure And Algorithms

A Graph is a non-linear data structure consisting of vertices and edges. The vertices are sometimes also referred to as nodes and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph is composed of a set of vertices(V) and a set of edges(E). The graph is denoted by G(E, V).

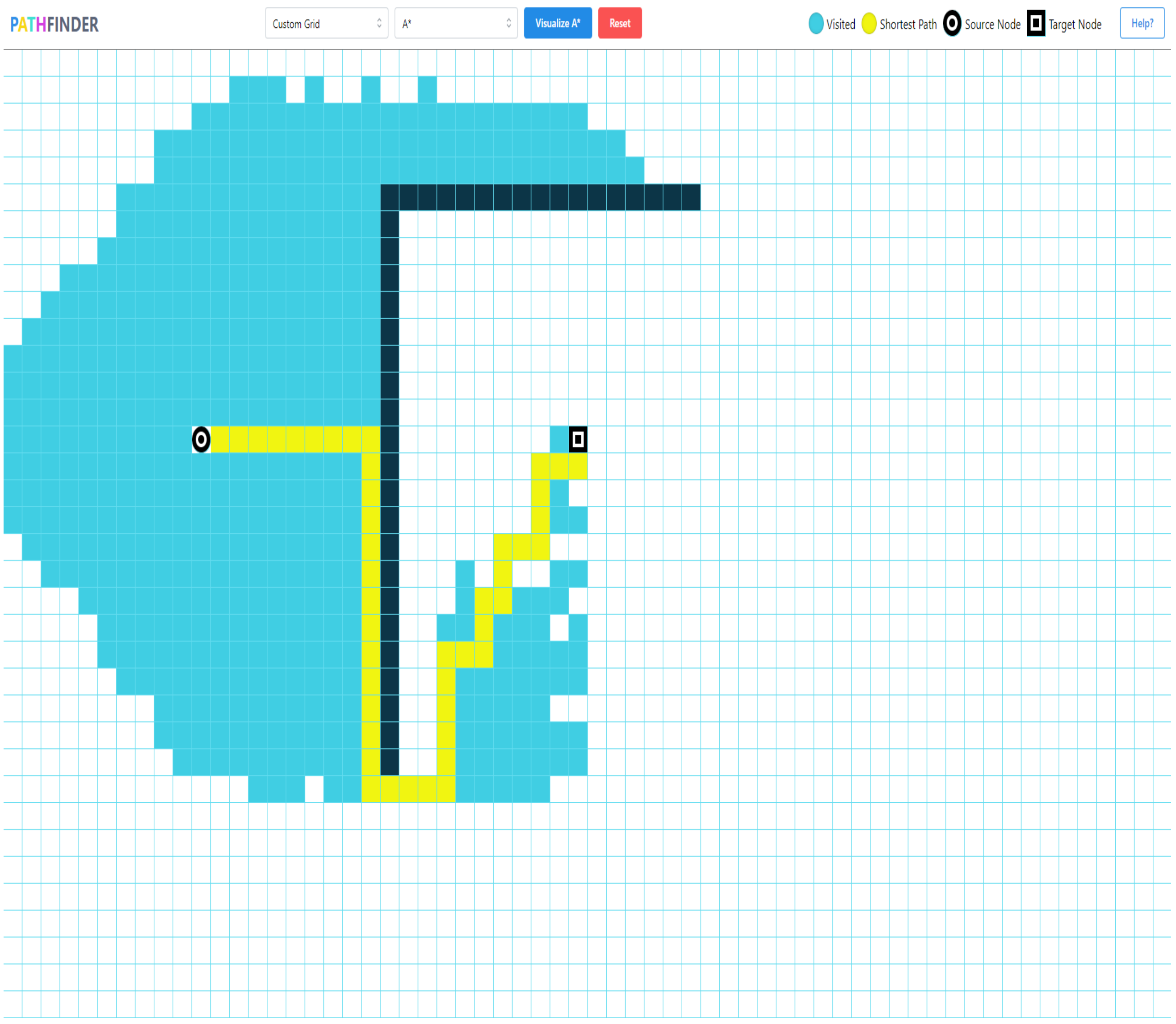
Components of a Graph

- Vertices: Vertices are the fundamental units of the graph. Sometimes, vertices are also known as vertex or nodes. Every node/vertex can be labeled or unlabelled.
- Edges: Edges are drawn or used to connect two nodes of the graph. It can be ordered pair of nodes in a directed graph. Edges can connect any two nodes in any possible way. There are no rules. Sometimes, edges are also known as arcs. Every edge can be labeled/unlabelled.

Graphs are used to solve many real-life problems. Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, Facebook. For example, in Facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender, locale etc.

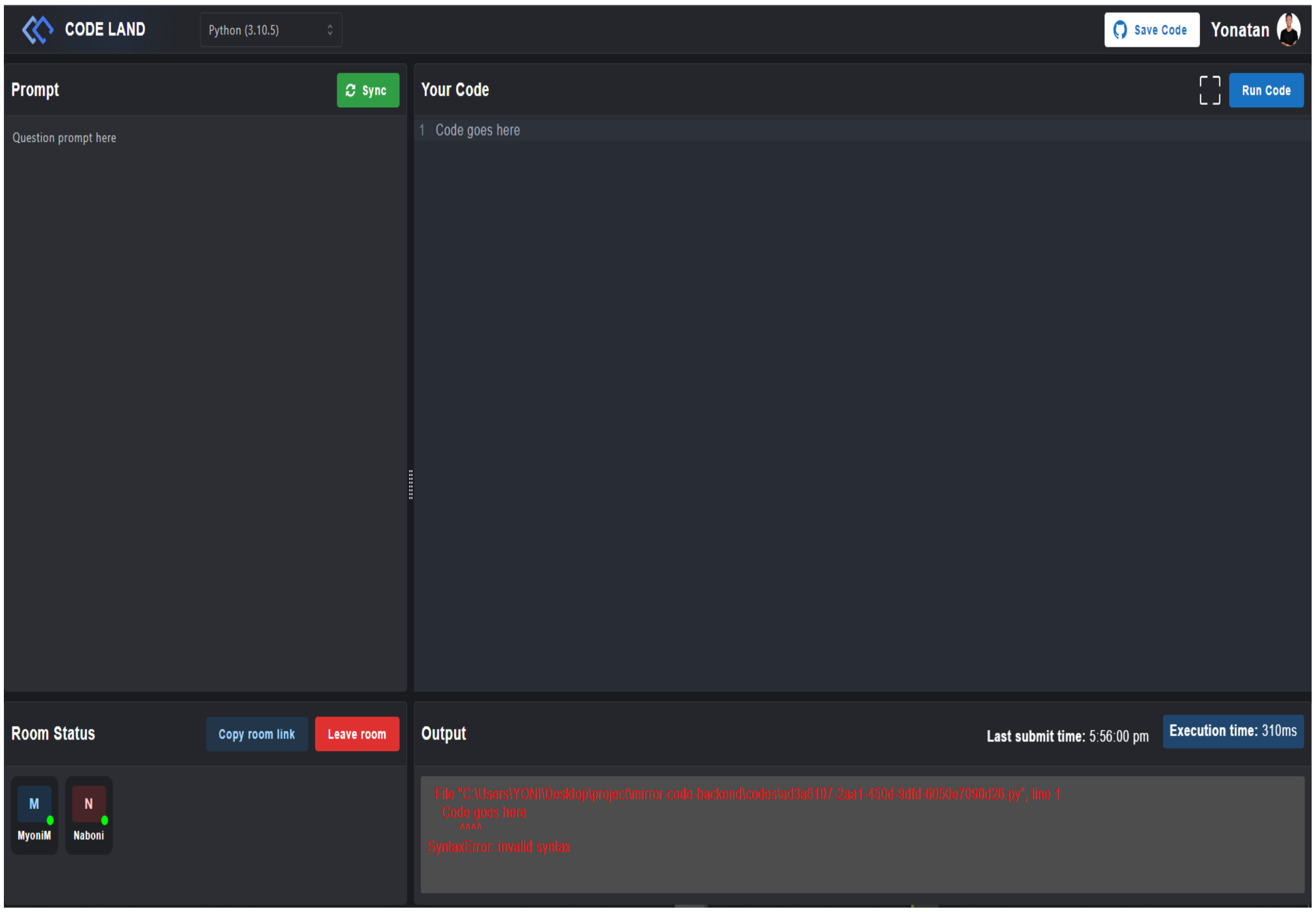
C. Visualize topic

● Users can learn each topic through Visualizing by pressing the go to visualization button in each topic.



D. Pair Programming

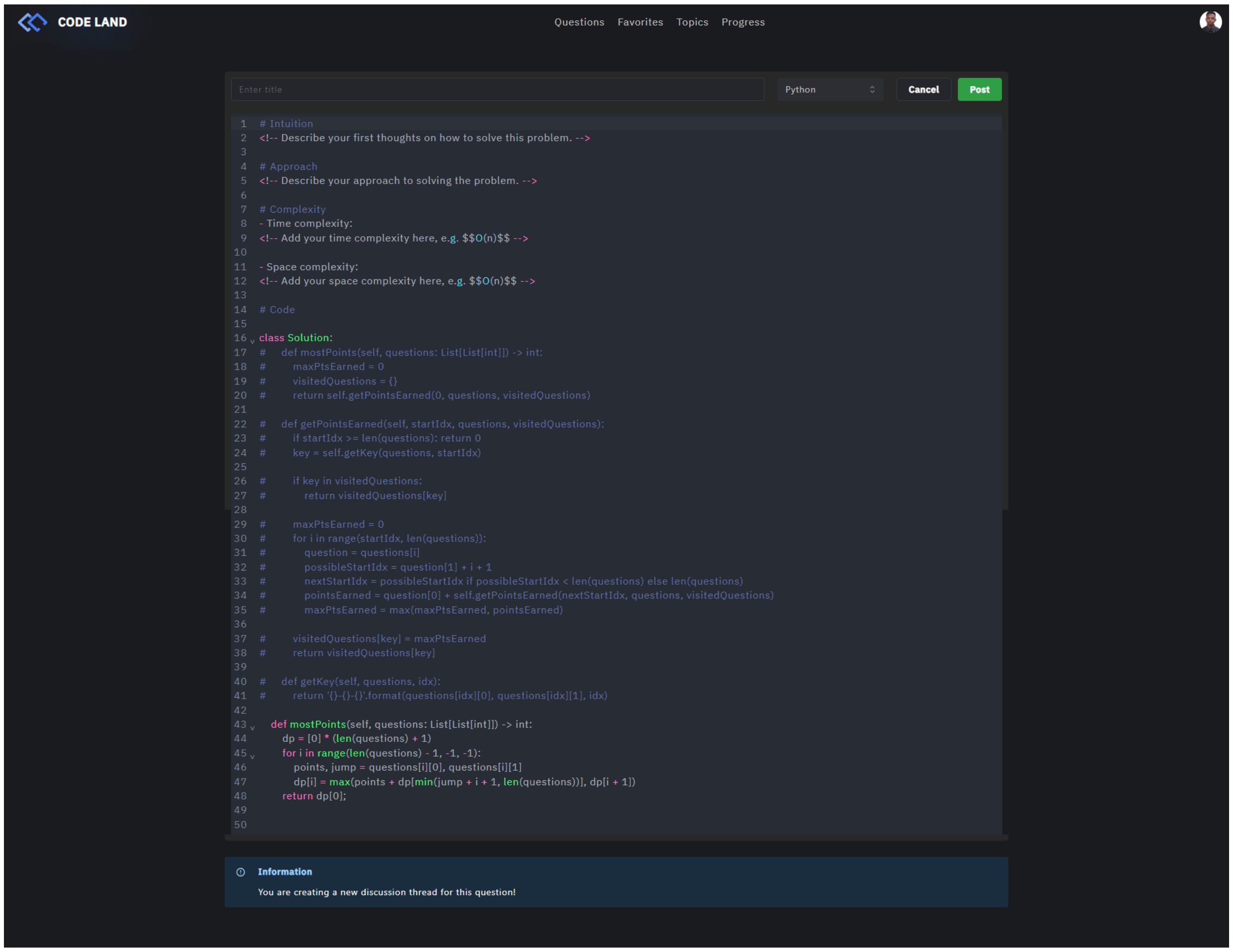
- users can invite their peers and code together using different programming languages by pressing create invite link.



```
 File "C:\Users\YONI\Desktop\project\mirror-code-backend\codes\ad3a6107-2aa1-450d-9dfd-6050e7090d26.py", line 1
  Code goes here
       ^^^^
SyntaxError: invalid syntax
```

E. Post Discussion

- Users can Post Discussion by pressing the post button.



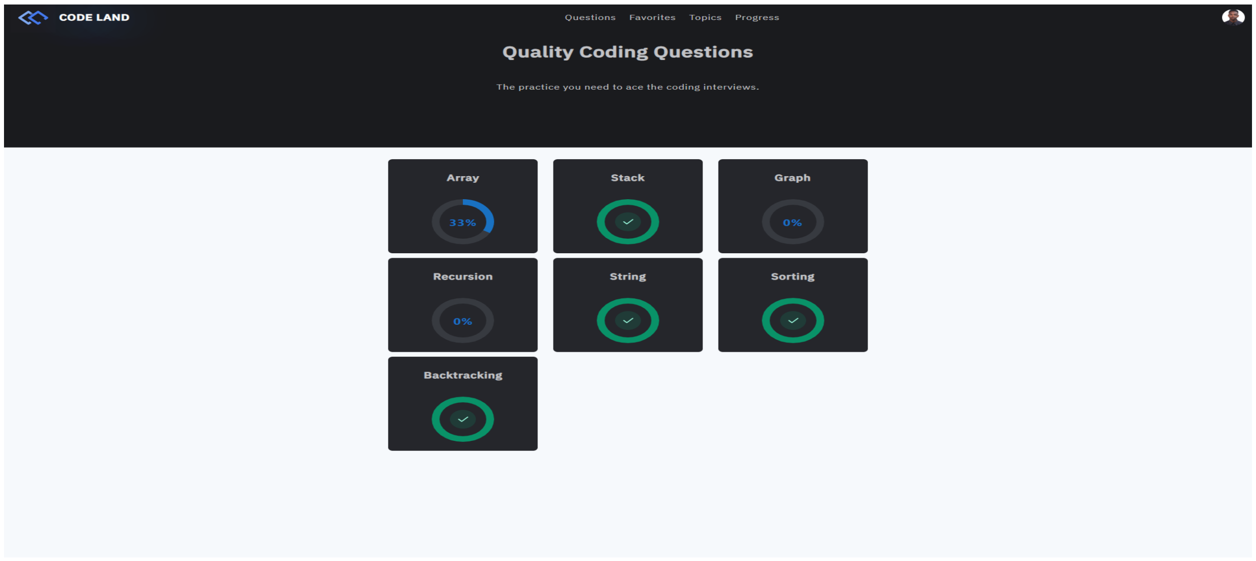
```
# Intuition
<!-- Describe your first thoughts on how to solve this problem. -->

# Approach
<!-- Describe your approach to solving the problem. -->

# Complexity
- Time complexity:
<!-- Add your time complexity here, e.g. $$O(n)$$ -->

- Space complexity:
<!-- Add your space complexity here, e.g. $$O(n)$$ -->

# Code

class Solution:
#    def mostPoints(self, questions: List[List[int]]) -> int:
#        maxPtsEarned = 0
#        visitedQuestions = {}
#        return self.getPointsEarned(0, questions, visitedQuestions)

#    def getPointsEarned(self, startIdx, questions, visitedQuestions):
#        if startIdx >= len(questions): return 0
#        key = self.getKey(questions, startIdx)

#        if key in visitedQuestions:
#            return visitedQuestions[key]

#        maxPtsEarned = 0
#        for i in range(startIdx, len(questions)):
#            question = questions[i]
#            possibleStartIdx = question[1] + i + 1
#            nextStartIdx = possibleStartIdx if possibleStartIdx < len(questions) else len(questions)
#            pointsEarned = question[0] + self.getPointsEarned(nextStartIdx, questions, visitedQuestions)
#            maxPtsEarned = max(maxPtsEarned, pointsEarned)

#        visitedQuestions[key] = maxPtsEarned
#        return visitedQuestions[key]

#    def getKey(self, questions, idx):
#        return '{}-{}-{}'.format(questions[idx][0], questions[idx][1], idx)

    def mostPoints(self, questions: List[List[int]]) -> int:
        dp = [0] * (len(questions) + 1)
        for i in range(len(questions) - 1, -1, -1):
            points, jump = questions[i][0], questions[i][1]
            dp[i] = max(points + dp[min(jump + i + 1, len(questions))], dp[i + 1])
        return dp[0];
```

Information

You are creating a new discussion thread for this question!

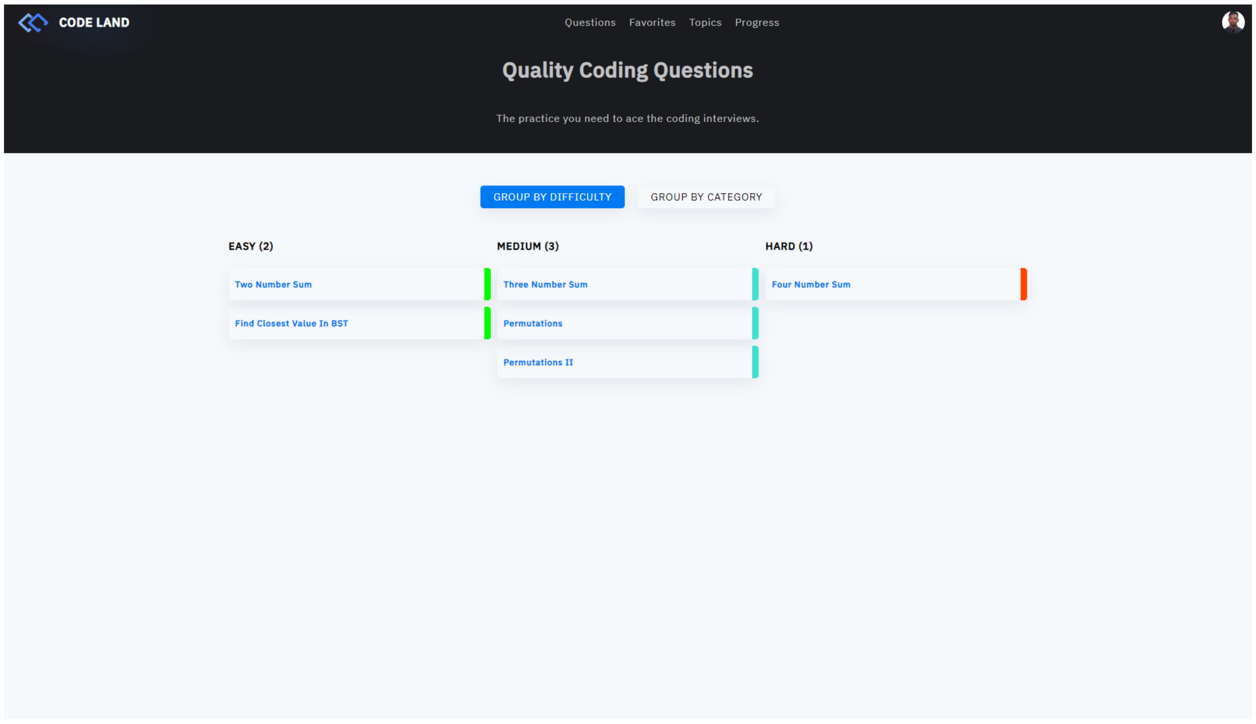
F Topic Status

- Users can see topic status



G. Questions Page

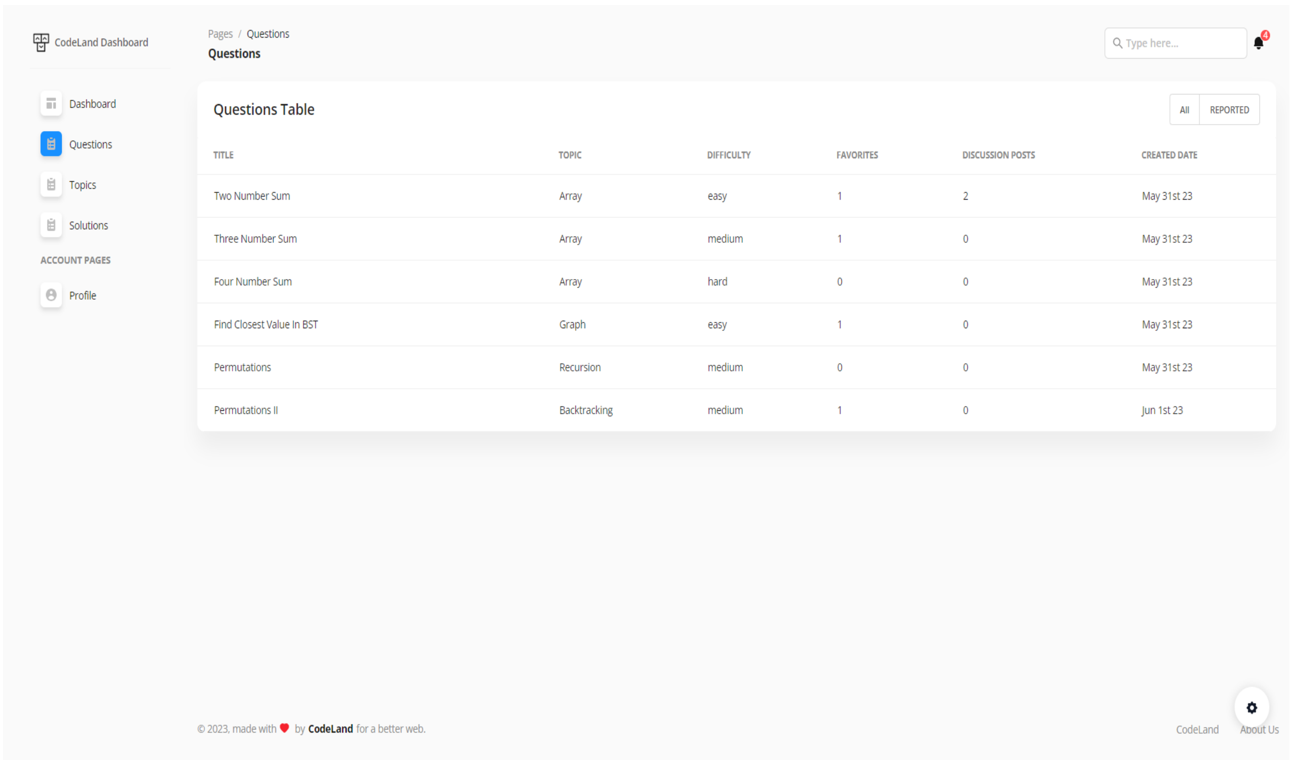
- Users can see list of questions by pressing Questions button from navbar



5.3.2 Admin

A. Question

- By pressing question button from side navbar you can manage list of questions



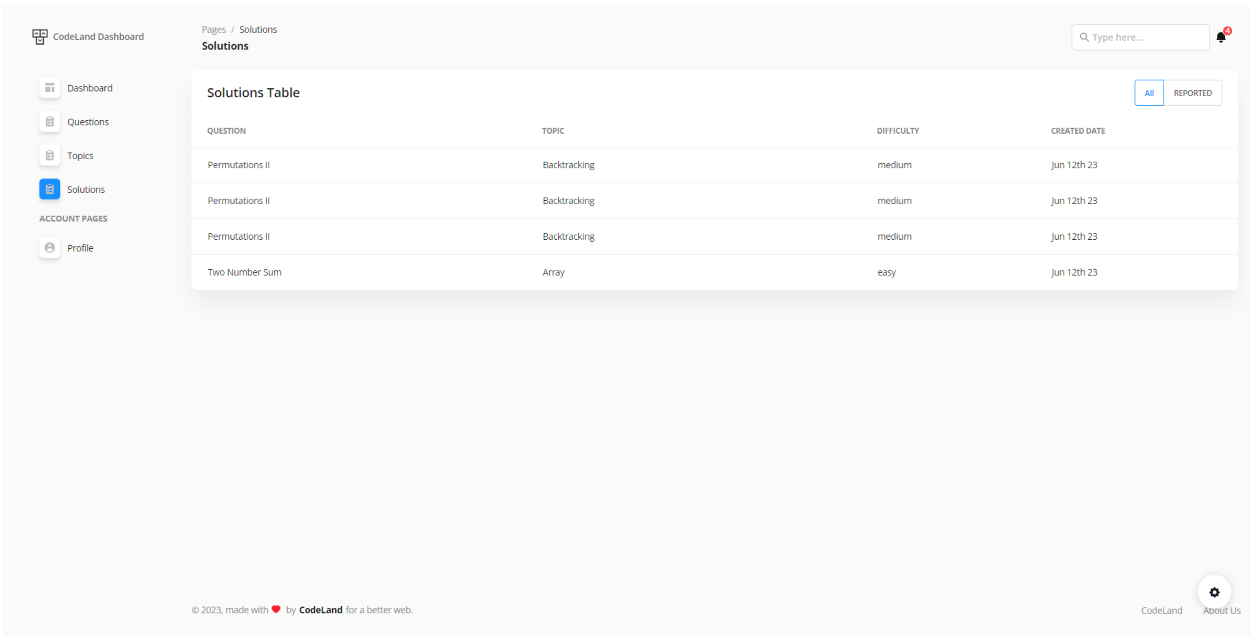
Questions Table

TITLE	TOPIC	DIFFICULTY	FAVORITES	DISCUSSION POSTS	CREATED DATE
Two Number Sum	Array	easy	1	2	May 31st 23
Three Number Sum	Array	medium	1	0	May 31st 23
Four Number Sum	Array	hard	0	0	May 31st 23
Find Closest Value In BST	Graph	easy	1	0	May 31st 23
Permutations	Recursion	medium	0	0	May 31st 23
Permutations II	Backtracking	medium	1	0	Jun 1st 23

© 2023, made with ♥ by **CodeLand** for a better web.

B. Solution

- By pressing the Solution button from the side navbar you can manage a list of Solutions.

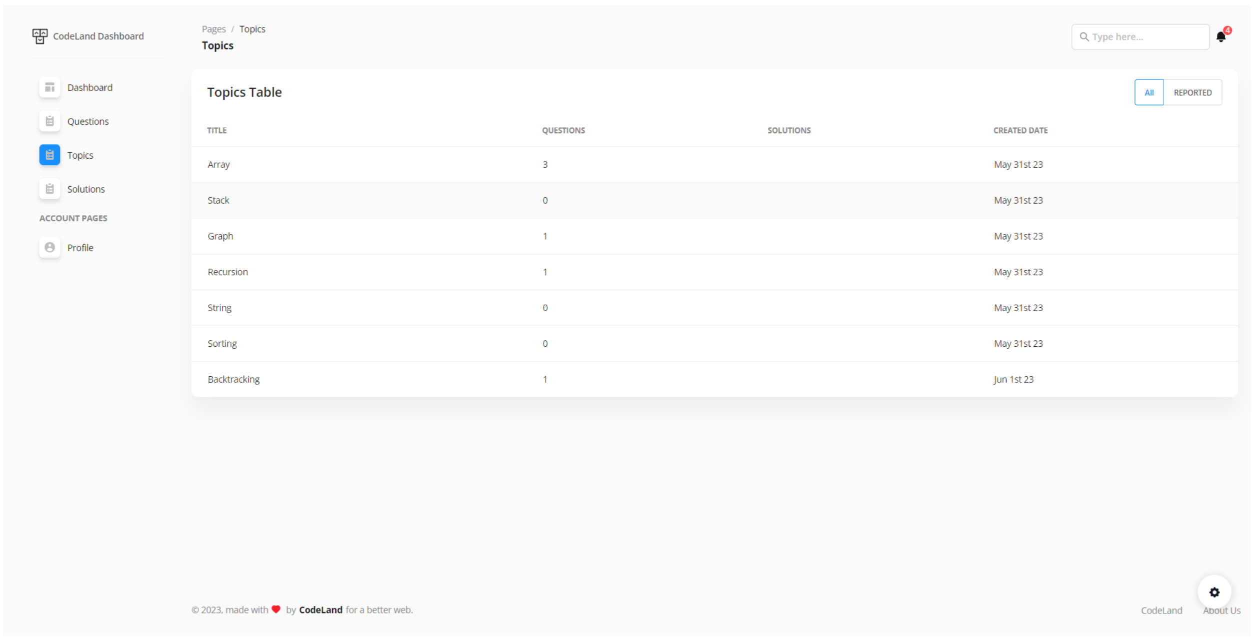


Solutions Table

QUESTION	TOPIC	DIFFICULTY	CREATED DATE
Permutations II	Backtracking	medium	Jun 12th 23
Permutations II	Backtracking	medium	Jun 12th 23
Permutations II	Backtracking	medium	Jun 12th 23
Two Number Sum	Array	easy	Jun 12th 23

C. Topic

- By pressing the topic button from the side navbar you can manage a list of topics.



Topics Table

TITLE	QUESTIONS	SOLUTIONS	CREATED DATE
Array	3		May 31st 23
Stack	0		May 31st 23
Graph	1		May 31st 23
Recursion	1		May 31st 23
String	0		May 31st 23
Sorting	0		May 31st 23
Backtracking	1		Jun 1st 23

© 2023, made with ❤ by **CodeLand** for a better web.

Chapter 6: CONCLUSION AND RECOMMENDATION

6.1 Conclusion

In conclusion, the development of a data structure and algorithm teaching software is a crucial step towards enhancing the learning experience of university students. The software provides a platform for students to learn and practice data structures and algorithms in a structured and interactive manner. The software's features and capabilities, as outlined in the Software Requirements Specification (SRS), are designed to meet the needs of both students and instructors, ensuring that the software is effective, efficient, and user-friendly. The successful implementation of the software will not only benefit the students but also the university, as it will enhance the quality of education and produce graduates who are well-equipped with the necessary skills to succeed in the industry. Overall, the development of a data structure and algorithm teaching software is a significant contribution to the field of computer science education, and we look forward to its successful implementation and adoption.

6.2 Recommendation

We recommend the use of this project for these main reasons:

- To learn data structure and algorithm easily with visualization
- To develop their data structure and algorithm knowledge using different questions
- To develop their coding skill by programming with their peers in one code base.

BIBLIOGRAPHY

[1]Use Case Diagram, https://www.lucidchart.com/pages/uml-use-case-diagram, Dec 2022

[2]Data structure and algorithm Notes, https://www.geeksforgeeks.org, Jun 2023

[3]Data structure and algorithm Question, https://www.geeksforgeeks.org, Jun 2023

[4] Class Diagram Tutorial, https://creately.com/blog/diagrams/class-diagram-relationships/, Dec 2022

[5] Sequence Diagram Tutorial, https://www.lucidchart.com/pages/uml-sequence-diagram, Dec 2022

APPENDIX