

Functional Requirements

1 Elicitation Techniques Used

- Interviews
- Surveys
- Brainstorming among group members

2 Requirements

1. User Authentication

(a) Login and Sign Up

Users should be able to create an account using their email ID and choose a password, or log in if they already have an account.

(b) Login with Google

Users should be able to login using Google's OAuth authentication as it is faster and easier for them.

(c) Login with GitHub

Users should be able to directly login using their Github account.

(d) Reset Password

If a user forgets his password, he should be able to recover his account. For this purpose a reset password link shall be sent to the registered emailID.

2. Roadmap

(a) Roadmap Generation

The system should be able to generate roadmaps for any relevant coding topic given by the user.

(b) Considering user's experience level

The system should generate roadmaps based on user's experience level (i.e beginner, intermediate or advanced)

(c) Career focused roadmaps

Users should be able to generate roadmaps focused on their career goals, such as becoming a web developer, data scientist, or software engineer.

(d) Topic based modules

Each Roadmap should be divided into smaller topic wise modules, allowing the user to learn in a structured manner.

(e) **Difficulty tags**

The modules should have difficulty tags to help users understand how challenging a module is before starting it.

(f) **View prerequisites**

Each module should display topics that are prerequisites, allowing users to be well prepared before starting a module.

3. Course Management

(a) **Course Organization**

Users should be able to access, manage, delete, pin and organise their generated courses.

4. Progress Tracking

(a) **Progress bar**

A progress bar should be shown to the users which tracks how many modules in the roadmap they have completed.

5. AI Assistance

(a) **Giving Hints**

Instead of directly giving the final answer, AI should give hints step by step, allowing users to think and solve on their own.

(b) **Feedback on code**

The AI should analyze the user's code and give feedback on its quality and suggest improvements..

(c) **Optimization**

The AI should be able to rewrite or optimize users code while also explaining the changes, to help users learn better practices.

6. Integrated IDE

(a) **IDE**

An integrated IDE will be provided to allow users to write and run code. The user shall be able to give input and output must be displayed.

(b) **File Management**

Users should be able to create, write, save and delete files within the IDE for proper management.

(c) **Time and Space Complexity Tool**

The IDE will analyze user's written code and provide its time and space complexity.

(d) **Code Formatting, Debug and Autocomplete**

The IDE should include features such as formatting user written code, provide debugging tools and autocomplete the code.

7. Evaluations after each module

(a) **Quizzes**

Each module will contain quiz which include multiple choice questions or coding questions to test the user's understanding.

(b) **Questions categories**

All questions will be tagged as Easy, Medium, or Hard, allowing users to progress at their own level.

8. External Sources

(a) **Videos and Articles**

AI will fetch relevant videos and articles for each topics in the module from the web, which could be helpful to users for understanding.

(b) **External Sources**

The resources fetched will be sorted based on their quality and usefulness, so that users can see the most relevant videos and articles first.

9. Notes

(a) **Standard and Markdown format**

Users can write plain text in notes or can use markdown too for proper formatting. The markdown code should be previewed side by side.

(b) **Import and Export Notes**

Users should be able to save notes locally and should be able to import and export notes.