Thoth Requirements Specification Document

November 2023

1 Introduction

1.1 Document Purpose

This document is intended to provide a detailed description of system requirements and the constraints that should be satisfied. The targeted audience is the team members to provide agreement between them on what should be implemented.

1.2 Product Scope

This system will be an Interactive Programming Tutorial (C++ for a start). It will be different than other alternatives because it will provide a smoother interface. In addition, it will depend on multiple APIs that will be used to enhance the learning experience.

1.3 Definitions, Acronyms, and Abbreviations

The project's Definitions, Acronyms, and Abbreviations are currently under development

1.4 Document Overview

This document will include the product functions summarized and the product constraints. In addition, It will describe the user's Characteristics, the assumptions and dependants on the projects, and how the requirements will be transformed into function in the system(Apportioning of Requirements). Moreover, It will cover the user, hardware, and software interfaces of the system. In addition to functional and Non-functional requirements.

2 Product Overview

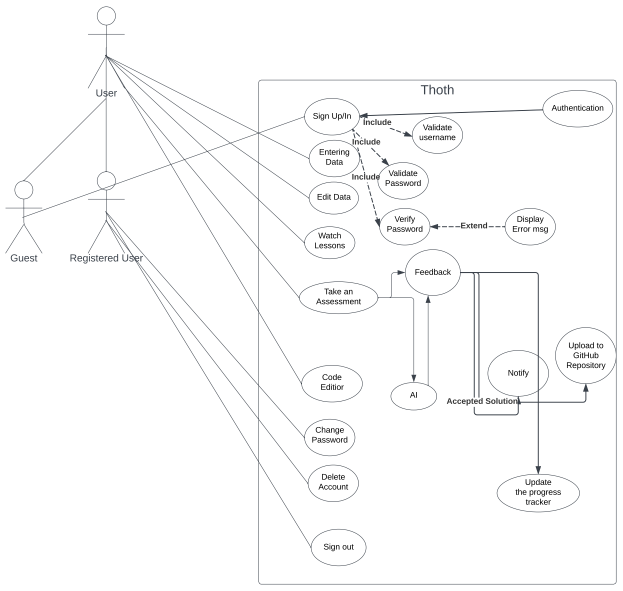
2.1 Product Functions

The users shall be able to sign up and track their progress across devices. In addition, the users shall be able to edit personal information and add friends to their accounts. The main aim of the app is to enable the users to write, compile, and run C++ code; feedback on this code should be sent to the users and notify them. The users shall be able to upload this solution to their GitHub accounts.

2.2 Product Constraints

The developers will opt for a smooth user interface that may be more complex to implement. For budget constraints, a less accurate AI server will be used to generate the solution feedback. To ensure security, the accounts will be linked to GitHub accounts which will limit the signing up process if the user does not have a GitHub account.

2.3 User Characteristics



As shown in the use case, guest users shall be able to sign up (sign up/in). Registered users shall be able to sign in (sign up/in) and delete the account, change the password, and sign out. All users shall be able to enter data, edit data, and watch lessons. They also shall be able to write, modify, and compile code (code editor). They shall be able to take the assessment and then receive AI feedback which is represented by (Take an assessment) then (AI) then (feedback). And they shall be notified if the solution is accepted which is represented by the (Notify) branching from (feedback). In addition, they shall be able to track their progress across devices(update the progress tracker) and upload the solution on GitHub(Upload on GitHub repository).

2.4 Assumptions and Dependencies

The app will depend in the first place on GitHub as the accounts will be linked to it. In addition, the users' solutions should be uploaded to their GitHub account. Moreover, the solutions' feedback accuracy will depend on the AI server's accuracy. The usability of the app will be dependent on the APIs.

2.5 Apportioning of Requirements

Function	Implantation Priority
As an end user, I want to have multiple options to sign-up and sign-in so that I can easily and securely access my account based on my preferences.	P0
As a user of the platform, I should have the ability to add and update my personal information, such as my username, password, and profile picture ensuring that my profile remains personalized.	P1
As an end user, I want to easily manage my lessons and track my progress so I can optimize my learning experience.	P1
As an end user, I want to take assessments covering the topics I learned in the lesson to test my understanding.	P1
As an end user, I want to be able to receive feedback on my code to have a dynamic method of interaction with the lesson and make the most use out of it.	P1
As an end user, I want to be able to utilize the built-in code editor to write, compile, and run C++ code, enabling me to try examples and solve problems.	P1
As an end user, I want to be able to upload my code on my personal Github account once I solve a coding problem successfully to display my effort, codes, and progress to the public.	P2

As an end user, I want to have control over my notifications and personalize the platform's theme, so I can receive updates on my progress and achievements while customizing the platform's appearance to suit my preferences.	P2
As an end user, I want to seamlessly view my progress across multiple devices, so I can transition between devices and continue my learning journey without interruption.	P3

3 Requirements

3.1 External interfaces

3.1.1 User interfaces

The users will send data to the system through a simple GUI that will include space to write code and a button to submit their code to the system. In general, users will be only using the provided GUI to communicate with the system.

3.1.2 Hardware interfaces

The exact list of supported devices is not yet specified.

3.1.3 Software interfaces

1. The problem set and the test cases for each problem shall be stored on an external database server.
2. The users' information shall be stored in an encrypted format in an external database server.
3. Vim text editor software shall be embedded in the system allowing for typing code. Nvim API will be used to connect to this software interface.
4. The system shall connect to an external judge API to be able to run the submitted codes against the test cases. The exact judge used is not yet specified.
5. The system shall connect to an external algorithm graphical visualiser API to view some of complicated complicated algorithms. The exact algorithm visualizer is not yet specified.

3.2 Functional requirements

1. The user shall create a new account by providing a username, a password, and an email address.

2. The user shall sign-in or sign-up using either their email, GitHub account, or Google account.
3. The user must confirm his password when signing-up.
4. After successful registration, the user receives a confirmation email for account verification.
5. The user can reset the password by requesting a password reset link via email.
6. A user-friendly interface shall be found for addition and modification of personal information, including username, password, and profile picture.
7. Any changes made by the user shall be promptly updated across the platform, with the system performing validation checks to ensure data accuracy.
8. The user shall browse the available lessons and enroll in any of them.
9. The user shall view all the lessons they are enrolled in.
10. The user shall access all the content of a lesson they are enrolled in.
11. The user shall see which lessons they have completed.
12. The user shall view a clear roadmap of the recommended order to complete the lessons.
13. The user shall distinguish between completed and pending lessons using visual indicators or labels.
14. The user shall view a percentage progress bar which tracks the number of lessons completed.
15. The assessment shall include a coding component and an MCQ component.
16. The assessment shall cover every topic in the lesson, ensuring comprehensiveness.
17. Each problem in the assessment shall offer one or more hints to assist users in solving the given problem.
18. For coding problems, sample input and sample output shall be given to make sure the user clearly understands the problem.
19. An optional timer shall be available in the assessment to track the time taken to solve it.
20. MCQ questions must include an explanation for the correct answer.

21. Feedback for a coding question shall consist of a verdict (accepted, wrong answer, run-time error, time-limit exceeded, memory-limit exceeded).

22. Each coding problem shall have an editorial that explains the solution and provides the corresponding code.

23. AI-generated feedback shall offer hints for the problem, detect logical errors, and suggest potential optimizations.

24. The code editor shall support the latest C++ version.

25. Code auto-completion shall be available to users while writing code.

26. The code editor shall support undo and redo operations.

27. Feedback for the user indicates whether the code compiled and ran successfully. In case of unsuccessful compilation or execution, the correct error message is provided.

28. The system shall only initiate the GitHub code upload process for coding problems, and not for MCQ problems.

29. Code is uploaded to the user's personal GitHub account only after the coding problem is accepted

30. The system should use a designated script to handle the code upload process seamlessly.

31. The script should access the user's GitHub account securely, requiring proper authentication.

32. The uploaded code shall be organized and properly labeled within the user's GitHub with the problem's name and lesson number.

33. In case of any upload failure or issues, the system should provide appropriate error messages to the user.

34. Users shall have the option to enable or disable the GitHub code upload feature from their account settings.

35. When accessing the platform on a new device, the user should be prompted to login and verify their identity.

36. The platform should allow users to access a lesson from several devices.

37. Changes made on one device shall be immediately reflected across all other devices.

38. The user shall be notified of their assessment results upon completion.

39. The user shall be notified upon completion of a lesson.

40. The user shall be able to opt in for email notifications.
41. Notifications shall be clear and concise - does not exceed two sentences.
42. The user shall be able to choose between light, dark, or colorblind themes.
43. Theme customization shall be accessible from the account settings.

3.3 Non-functional requirements

1. **Security:** All user data, including passwords and personal information, must be encrypted during storage and transmission.
2. **Performance:** The system shall provide feedback for the submitted solutions in a time period ranging from **five seconds** to no more than **three minutes**, depending on the problem and the size of the test cases.
3. **Availability:** Any updates should be done during non-peak hours to minimize user disruption.
4. **Customization:** Theme changes should apply to the entire platform ensuring consistency.
5. **Notification Delivery:** Notifications should be delivered promptly in no more than **two minutes**.

4 Verification

Static verification will be done to ensure that the requirements are correct, consistent, complete, and conform.

for this sprint verification is done by peer reviewing.

5 References

The project's list of references is currently under development.

6 Appendices