SOFTWARE REQUIREMENTS SPECIFICATION

for

Thoth, a C++ Online Learning Platform

Version 4.0

Prepared by: Omar Bahgat
Shaza Ali
Ziad Hassan
Bemen Girgis
Dana Alkhouri
Layla Hussein

December 10, 2023

Contents

1 Introduction

1.1 Document Purpose

The primary purpose of this document is to outline the system requirements, both the functional and non-functional, and constraints that should be fulfilled throughout the project lifecycle. It serves as a comprehensive guide intended for the team members to provide agreement on what should be implemented.

1.2 Intended Audience

This document is meant to a diverse audience engaged in various aspects of the product development. It targets software developers working on the project, project managers, and stakeholders. Additionally, it serves as a reference for external contractors involved in the project. By providing comprehensive details on the software's requirements and functionalities, it aligns the understanding of all parties involved in managing, validating, and supporting the product.

1.3 Project Scope

Our product is an interactive programming tutorial platform centered around C++, aimed at providing a supportive environment for individuals venturing into programming for the first time. The core focus lies in creating a user-friendly experience that ensures accessibility to coding skills, catering specifically to newcomers in the programming domain. Our primary goal is to make programming education accessible to everyone, irrespective of their background or prior experience. This platform will offer a diverse range of lessons and engaging practice problems carefully planned to facilitate a gradual learning curve. By providing comprehensive learning materials and a supportive environment, we aim to empower individuals to learn and excel in programming effortlessly. Moreover, the platform's design emphasizes intuitive navigation and interactive tools, ensuring a smooth and engaging learning journey for users at all proficiency levels.

2 Overall Description

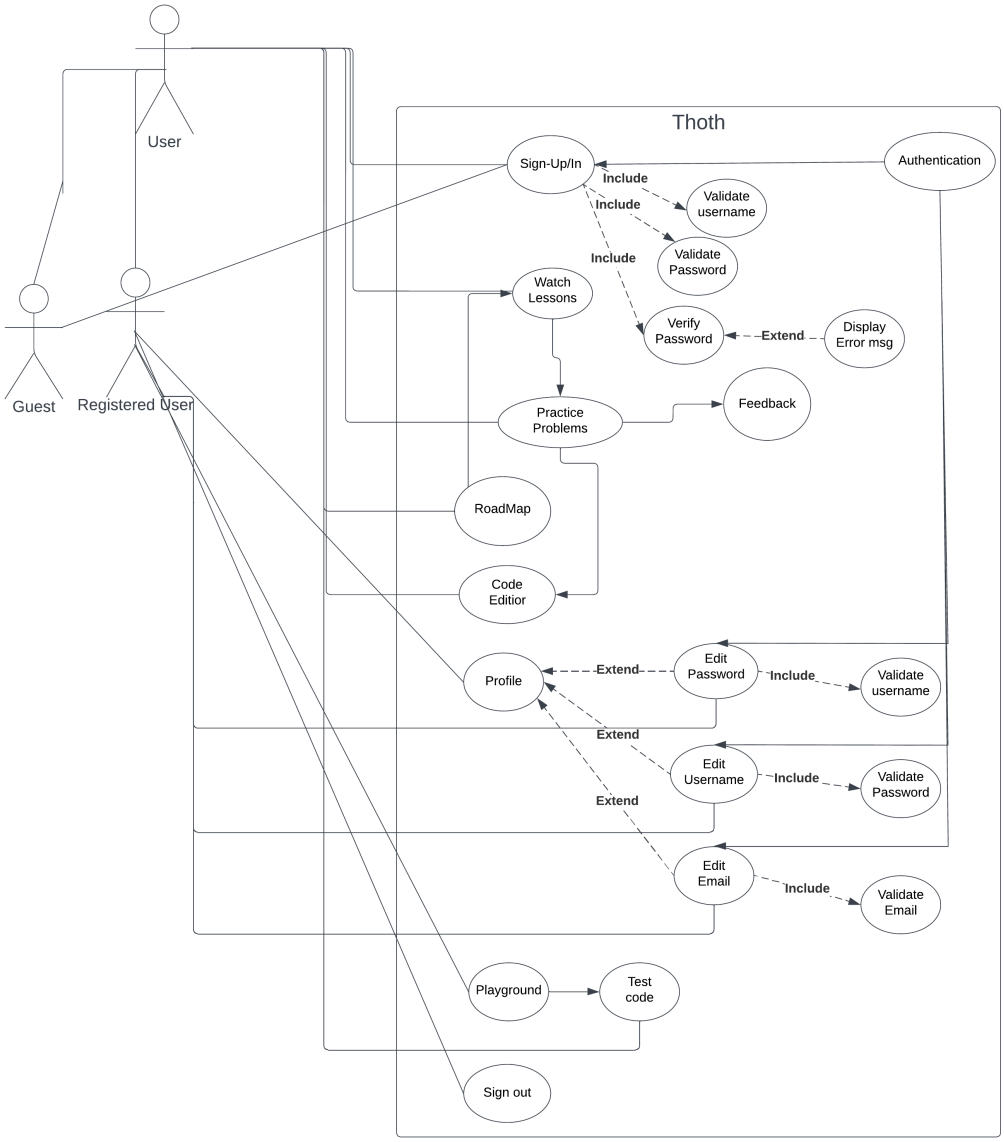
2.1 User Needs

The primary users of our product are individuals new to the programming domain, seeking to acquire foundational coding skills. They play a central role in the platform's usage, being the primary audience for learning programming concepts. These users typically come from diverse backgrounds, often without prior coding experience. For them, the product needs to fulfill the need of providing an accessible, user-friendly learning environment. This includes simplifying complex coding principles, offering comprehensive learning resources and practice problems, and guiding them through an engaging learning journey. Emphasizing intuitive navigation, gradual skill progression, and interactive learning tools becomes essential to ensure these primary users can comfortably grasp fundamental coding concepts.

2.2 Dependencies

The Website depends mainly on Firebase for user security to store user credentials. Moreover, it depends on Firebase as the problem sets and test cases are stored on a database hosted by it. It depends also on the used APIs for the assessment and problem set to function. It depends on Monaco text editor to enable the user to write code. It depends also on HackerRank as the problem sets use its links for the practice problems.

2.3 Product Functions



2.4 Operating Environment

The website can operate in any environment - Mac, Windows, Linux etc.

2.5 Design

The user has four activities:

1. Sign Up or sign in by entering email and password or click on Google or GitHub to link their accounts.
2. Watch Lessons by clicking on the lesson icon.
3. Access Practice Problems
4. Test code on the code editor.

The Registered user has Three activities:

1. Access Playground from the AppBar.
2. Sign Out
3. Access Profile page and edit personal information.

The Guest has One activity:

1. Sign Up by entering email and password or click on Google or GitHub to link their accounts.

3 System Features and Requirements

3.1 External Interfaces

3.1.1 User Interfaces

1. Login Page in which the user can enter username and password or click on Google or GitHub to link their accounts to sign in or sign up.
2. Roadmap in which the user can access any lesson by clicking on this lesson icon
3. Lesson Page in which the user can read the lesson content, take an assessment by using the code editor and receive feedback on their solution.
4. Problem set page in which the user can access the problem link or access a solution video link for the problem.

3.1.2 Software Interfaces

1. The problem set and the test cases for each problem shall be stored on an a database hosted by **Firebase.**
2. The platform shall connect to **Firebase** to process and handle sign-in, sign-up, and authentication.
3. Monaco text editor software shall be integrated into the system to enable code typing. The system will utilize **Node.js** to establish communication with this embedded software interface.
4. The system shall connect to an external judge API to be able to run the submitted codes against the test cases.
5. The system should connect to an external algorithm graphical visualiser API to view some of complicated complicated algorithms. The exact algorithm visualizer is not yet specified.
6. The practice problems section shall have links to introductory problems on HackerRank.
7. The lesson section and the playground section shall have the InterviewBit code editor embedded in them and will compile the code via InterviewBit's external compiler.

3.2 Functional Requirements

1. The user shall sign-up using one of the provided methods.
2. The user shall sign-in using one of the provided methods.
3. The user can sign-up using their email address, password, and password confirmation.
4. The user can sign-up using their Google or GitHub accounts.
5. The user can sign-in using their email address and password.
6. The user can sign-in using their Google or GitHub accounts.
7. The user can request a password reset link via a "Forget Password" button.
8. The user shall have an app bar featuring buttons for the roadmap, playground, practice problems, and App Bar menu with extra features.
9. The user shall be able to click the "Roadmap" button for direct navigation to the roadmap page.
10. The user shall be able to click the "Practice Problems" button for direct navigation to the practice problems page
11. The user shall be able to click the "Playground" button for direct navigation to the playground page.
12. The user shall be able to click the App Bar menu which provides access to the following features: settings, profile, and logout.
13. The user shall be able to click on the "Profile" button and update their personal information which includes their email address, username, password, and city.
14. The user shall be able to view a roadmap labeled with visually distinct buttons to represent the unique lessons.
15. The user shall be able to click on a lesson on the roadmap and be directed to that lesson without requiring additional navigation steps.
16. The user shall be able to access comprehensive lessons covering explanations of C++ programming topics as well as relevant C++ code snippets.
17. The user shall have a clear presentation of the lesson content with a distinct separation between the textual content and the code snippets.
18. The user should be able to see on the roadmap which lessons they have completed.
19. The user shall access an integrated code editor sourced from InterviewBit within the lesson interface.

20. The user shall compile C++ code entered within the integrated editor, enabling verification of code syntax and structure.

21. The user shall execute compiled C++ code directly within the lesson environment to observe its functionality and output.

22. The user shall receive immediate feedback on any encountered errors or compilation issues within their entered C++ code in the editor.

23. The user shall have access to the code editor within each lesson, allowing interactive engagement with C++ code examples presented in the lesson content.

24. The user shall be able to access the Practice Problems page through the "Practice Problems" button in the app bar.

25. The user shall be presented with a collection of lessons, each containing individual coding problems.

26. The user shall be able to view the difficulty level (easy, medium, or hard) associated with each problem within a lesson.

27. The user shall be provided with links to video solutions for each problem, available on YouTube.

28. The user shall be able to access and view C++ solutions for each problem as a reference.

29. The playground section shall have the InterviewBit code editor embedded in it.

30. The system shall provide a designated script, allowing the user to initiate the code upload process to GitHub easily.

31. The user shall be required to authenticate their GitHub account securely using their credentials before enabling the code upload feature.

32. The system shall organize and label the uploaded code within the user's GitHub account based on the problem's name and lesson number.

33. The system shall verify the successful upload of the code and provide error messages to the user in case of any issues during the upload process.

34. The system shall respect the user's privacy and not access or modify any repositories on GitHub without permission.

35. The user shall have the option to enable or disable the GitHub code upload feature from their account settings.

36. The user shall be able to solve 5 MCQ questions for each lesson.

37. The user shall be provided with the correct answer along with the explanation upon solving the MCQ.

38. For coding problems, the user shall have AI-generated feedback that provides hints for the problem, detect logical errors, and suggest potential optimizations.

39. The user shall be able to opt for email notifications from the settings in the App Bar menu.

40. The user shall be notified upon completing a lesson.

41. The user shall be able to customize the platform theme from the settings in the App Bar menu.

42. The user shall be able to choose between light, dark, and colorblind themes.

3.3 Nonfunctional Requirements

1. **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.

2. **Availability:** Any updates should be done during non-peak hours to minimize user disruption.

3. **Customization:** Theme changes should apply to the entire platform ensuring consistency.

4. **Notification Delivery:** Notifications should be delivered promptly in no more than **two minutes**.