**Requirement Analysis Document (CodeEd)**

Contents

**Introduction1**

Overview of the Project 1

Purpose of the Requirement Analysis Document2

Scope of the Document2

**User Requirements Definition3**

**Functional Requirements3**

Description of each functional requirement3

Prioritization of requirements3

**Non-functional Requirements3**

Performance requirements3

Security requirements4

Usability requirements4

Compatibility requirements, etc.4

**Constraints and Assumptions4**

Technical constraints4

Business constraints5

Assumptions made during the requirement gathering5

**Traceability Matrix5**

**Purpose of the Traceability Matrix5**

**Traceability Matrix6**

**References8**

Sources of information, frameworks, or methodologies used in the document8

**Introduction**

**Overview of the Project**

At Code Ed, our mission is to democratize programming education by making it accessible, engaging, and rewarding for everyone. We strive to eliminate the barriers to learning programming by providing a user-friendly platform that caters to learners of all levels, from beginners to advanced programmers.

We are committed to delivering a personalized learning experience that adapts to each user's pace and style of learning, fostering a sense of achievement and progress.

Our interactive, bite-sized lessons and immediate feedback system are designed to keep our users motivated and engaged, helping them see quick results and continuous improvement.

We believe that everyone should have the opportunity to learn programming and harness the power of technology. Through Code Ed, we aim to empower our users with the skills they need to succeed in the digital world, fostering creativity, problem-solving, and lifelong learning.

**Purpose of the Requirement Analysis Document**

The Purpose of the Requirement Analysis Document (RAD) for CodeEd is multifaceted and serves as a critical component in the app development process. Here's a detailed breakdown:

* **Defining Scope and Objectives**: The RAD clearly outlines the scope of the app, including its intended functionality and objectives. This sets clear boundaries and goals for the development process, ensuring that all stakeholders have a unified understanding of what the app is meant to achieve.
* **Gathering and Documenting Requirements**: It serves as a central repository for all requirements, both functional (like features and user interactions) and non-functional (like performance, security, and scalability). This comprehensive collection aids in ensuring that no critical aspect is overlooked during development.
* **Guiding Design and Development**: For the design and development teams, the RAD provides a detailed guide to what needs to be built. It helps in structuring the development process, deciding on technology stacks, and planning out the design elements.
* **Basis for Estimation and Planning:** It enables project managers to estimate resources, time, and budget more accurately. By understanding the complexity and extent of requirements, more realistic schedules and budgets can be formulated.
* **Quality Assurance and Testing:** The RAD is used by quality assurance teams to understand what needs to be tested. It provides a checklist against which the final app can be validated and verified, ensuring that all requirements are adequately met.
* **Future Enhancements and Scalability**: Post-launch, the RAD can serve as a reference for future enhancements and scalability efforts. It provides a historical context to the original requirements, which can be invaluable for long-term maintenance and updates.

**Scope of the Document**

The Scope of the Requirement Analysis Document (RAD) for an CodeEd development project is a critical section that defines and limits the extent of the project. This section typically includes the following elements:

* **Project Overview:** A brief description of the app, including its purpose, target audience, and high-level objectives. This provides a general context for the requirements and the overall vision for the app.
* **Functional Requirements:** Detailed descriptions of all the functionalities the app should offer. This includes user interactions, processes the app will support, data handling, and specific features that need to be developed.
* **Non-Functional Requirements:** Specifications that describe the operational aspects of the app. This includes performance criteria, security standards, usability, scalability, reliability, and compliance with legal and regulatory standards.
* **User Roles and Personas:** Definition of the different types of users who will interact with the app, such as end-users, administrators, and other stakeholders. This helps in understanding diverse user needs and expectations.
* **Technology and System Requirements:** Details about the technical environment in which the app will operate, including hardware, software, network, and other system specifications necessary for the app’s functionality.
* **Assumptions and Dependencies:** Listing of assumptions made during the requirement gathering phase and dependencies that might affect the project timeline or deliverables, such as third-party services or integration with existing systems.
* **Data Management and Security:** Requirements related to data handling, storage, protection, and privacy policies, especially if handling sensitive or personal user data.

**User Requirements Definition**

**Functional Requirements**

**Essential**

1. **Registration and authorisation:** The app should have registration and authorisation features to create individual user profiles.
2. **Creating personalised courses:** The app should be able to generate personalised courses based on user input and preferences.
3. **Creating Quizzes:** The app should offer a feature to create interactive quizzes related to the course material, allowing users to test their knowledge and track their progress.

**Desirable**

1. **Feedback and support:** The application should provide a means to receive feedback from users and provide support in case of problems.
2. **Sending Notifications:** The app should be able to send timely notifications to users about new courses, quiz results, and other relevant updates based on their preferences and activities within the app.

**Optional**

1. **Adding Widgets:** The app should include customizable widgets that users can add to their home screen or app interface for quick access to key features like course progress, upcoming quizzes, or daily learning goals.
2. **Profile Changing:** The app should allow users to easily update or modify their profile information, including their name, photo, learning preferences, and contact details.

**Non-functional Requirements**

Performance Requirements:

* The app should load its main interface within 2 seconds of launch.
* Response time for generating personalized courses after receiving user input should not exceed 5 seconds.
* The app should handle at least [specific number] of concurrent users without significant performance degradation.

Security Requirements:

* User data, including personal information and course progress, should be encrypted in transit and at rest.
* Implement secure user authentication methods, like OAuth or two-factor authentication, for registration and login.
* Regular security audits and updates should be conducted to ensure ongoing protection against vulnerabilities.

Usability Requirements:

* The app should have an intuitive, user-friendly interface, ensuring easy navigation for users of all ages and technical abilities.
* Accessibility features should be included, such as text-to-speech, adjustable text sizes, and high-contrast modes for users with disabilities.
* The app should provide clear, concise instructions and feedback for user actions, enhancing the overall user experience.

Compatibility Requirements:

* The app should be compatible with a range of devices, including smartphones and tablets, across different operating systems (iOS, Android).
* Ensure the app is optimized for different screen sizes and resolutions to accommodate various devices.
* The app should be tested and optimized for the latest two versions of major browsers (like Chrome, Firefox, Safari) for any web-based components.

Scalability Requirements:

* The backend infrastructure should be designed to scale seamlessly with increasing user numbers or data volumes.
* The app should maintain consistent performance and user experience under scaling conditions.

Maintainability and Support Requirements:

* The app’s code should be well-documented and adhere to coding standards to facilitate maintenance.
* A system for user feedback and bug reporting should be in place, with a dedicated support team to address user concerns.

**Constraints and Assumptions**

Technical Constraints:

* Device Compatibility: The app must be compatible with a wide range of mobile devices and operating systems, limiting the use of features or technologies specific to certain platforms.
* Internet Dependency: The app's functionality, especially personalized courses and quizzes, may rely heavily on internet connectivity, limiting offline capabilities.
* Integration Limitations: Challenges in integrating with third-party services or APIs, which could affect features like notifications or data import/export.
* Resource Limitation: Limited memory and processing power of some user devices may constrain the app's performance and feature complexity.

Business Constraints:

* Budget Constraints: Limited funding may restrict the scope of development, affecting the number of features or the depth of customization in the app.
* Time to Market: A tight timeline for launching the app could limit the extent of testing and refinement, impacting overall quality and feature set.
* Market Competition: The need to differentiate from competitors may influence the prioritization of unique features or innovative solutions.

Assumptions Made During Requirement Gathering:

* Stable Internet Access: Assuming users will have consistent and stable internet access, as the app's core functionalities like course updates and quiz participation rely on online connectivity.
* Device Availability: Assuming that users will access the app through smartphones or tablets with certain minimum specifications (like OS version, screen size, RAM).
* Content Creation: Assuming that the content for personalized courses and quizzes will be regularly updated and maintained, ensuring relevance and engagement.
* User Feedback: Assuming that users will be proactive in providing feedback and participating in support queries, which is crucial for iterative improvements and user satisfaction.

**Traceability Matrix**

**Purpose of the Traceability Matrix**

The Purpose of the Traceability Matrix in a project, particularly in software development or system engineering, is multifaceted and plays a critical role in ensuring the project's success and quality.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Req. ID** | **Requirement Description** | **Justification** | **Data Structures** | **Dependencies for Swift/Android Development** | **Test Result** |
| RQ1 | User can create a new account | Enable users to access personalized content | User database | Swift: Authentication Framework  Android: Authentication Library | Pass |
| RQ2 | User can log in to the system | Secure access to user-specific information | Authentication | Swift: Authentication Framework  Android: Authentication Library | Pass |
| RQ6 | User receives notifications | Stay informed about interactions and updates | Notification system | Swift/Android: Push Notification Service | Pass |
| RQ7 | User can edit their profile | Personalize and update user information | User profile data | Swift/Android: User Interface Library | Pass |
| RQ8 | User can delete their account | Provide an option to permanently remove account | User database | Swift/Android: Account Management Library | Pass |
| RQ9 | User can start a course | Begin educational programs | Course information | Swift/Android: Educational Content Library | Pass |
| RQ10 | User can complete a programming level | Demonstrate proficiency in programming skills | Progress tracking | Swift/Android: Progress Tracking Library | Pass |
| RQ11 | User can complete an assignment | Mark assignment as completed or not completed | Assignment data | Swift/Android: Assignment Management Library | Pass |
| RQ12 | User can choose another course | Explore different educational options | Course selection data | Swift/Android: Course Selection Library | Pass |
| RQ13 | User can access coding exercises | Practice coding skills through exercises | Coding exercise data | Swift/Android: Coding Exercise Library | Pass |
| RQ14 | User can view code solutions | Learn from example solutions to coding exercises | Solution database | Swift/Android: Coding Exercise Library | Pass |
| RQ16 | User can view progress analytics | Track and analyze personal progress in programming | Analytics data | Swift/Android: Analytics Library | Pass |
| RQ17 | User can access video lectures | Learn through video-based educational content | Video lecture data | Swift/Android: Video Streaming Library | Pass |
| RQ18 | User can download course materials | Access educational materials offline | Downloadable files | Swift/Android: Download Manager Library | Pass |
| RQ19 | User can receive real-time coding feedback | Get immediate feedback on coding exercises | Feedback engine | Swift/Android: Real-time Feedback Library | Pass |
| RQ20 | User can share achievements on social media | Share progress and accomplishments on social platforms | Social media integration | Swift/Android: Social Media API | Pass |
| RQ22 | User can bookmark favorite coding challenges | Save and revisit preferred coding challenges | Bookmarking system | Swift/Android: Bookmarking Library | Pass |
| RQ23 | User can access coding tutorials | Learn programming concepts through step-by-step guides | Tutorial data | Swift/Android: Tutorial Presentation Library | Pass |
| RQ24 | User can customize coding environment | Adapt the coding environment to personal preferences | Customization settings | Swift/Android: User Interface Library | Pass |
| RQ25 | User can earn virtual badges | Receive badges for completing challenges and milestones | Badge system | Swift/Android: Badge Management Library | Pass |
| RQ28 | User can export progress data | Save and analyze personal progress data | Export feature | Swift/Android: Data Export Library | Pass |
| RQ29 | User can set coding goals | Establish and track individual coding objectives | Goal-setting features | Swift/Android: Goal-setting Library | Pass |
| RQ30 | User can provide feedback on course content | Contribute input to improve the quality of course material | Feedback system | Swift/Android: Feedback Management Library | Pass |
| RQ31 | Admin can create a new course | Add new courses to the platform | Course management | Swift/Android: Course Management Library | Pass |
| RQ32 | Admin can edit existing course details | Modify information about existing courses | Course management | Swift/Android: Course Management Library | Pass |
| RQ33 | Admin can delete a course | Remove outdated or irrelevant courses | Course management | Swift/Android: Course Management Library | Pass |
| RQ34 | Admin can create a new exam | Add new exams to the platform | Exam management | Swift/Android: Exam Management Library | Pass |
| RQ35 | Admin can edit existing exam details | Modify information about existing exams | Exam management | Swift/Android: Exam Management Library | Pass |
| RQ36 | Admin can delete an exam | Remove outdated or irrelevant exams | Exam management | Swift/Android: Exam Management Library | Pass |

**References**

**Sources of information, frameworks, or methodologies used in the document**

* **Industry Best Practices:** Leveraged standard practices in software development for requirement analysis and documentation.
* **Agile Methodology:** Adopted principles from Agile methodologies for iterative development and stakeholder involvement.
* **IEEE Standards:** Referred to IEEE standards for software requirements specifications.
* **Project Management Body of Knowledge (PMBOK):** Utilized PMBOK guidelines for managing and documenting project requirements and processes.
* **User-Centered Design Principles:** Incorporated user-centered design principles for usability and user experience requirements.
* **Security Protocols and Compliance Regulations:** Considered industry-standard security protocols and relevant compliance regulations (like GDPR) for security and privacy requirements.