SOFTWARE ENGINEERING FUNDAMENTALS

APRIL 24, 2024

Mrwan Alhandi

Table of Contents

Functional Requirements *2*

Non-Functional Requirements *3*

Architectural Design: MVC *4*

Components **5**

Relationships **6**

Architectural Decisions: Reward System & Framework *8*

Software Process: Agile *10*

Functional Requirements

Functional requirements define the specific behaviors, actions, and functionalities that a system must exhibit to fulfill its purpose. These requirements are crucial as they outline exactly what the system must do in various scenarios, providing a clear roadmap for development and testing. For a software platform like CodeQA, functional requirements might include enabling users to register and log in, post, and respond to questions, vote on the quality of answers, receive notifications about activities related to their posts or interests or others like the ones mentioned below. Additionally, the system should ensure these interactions are handled efficiently, securely, and in a user-friendly manner. Each requirement is not only a directive for building features but also serves as a criterion against which the final product is validated to ensure it meets user needs and operational demands.

Functional requirement 1: Achievements System

The CodeQA system must implement an achievement system to automatically track user activities, evaluate them against predefined criteria, and award badges when those criteria are met.

Functional Requirement 2: AI Help System

The CodeQA system shall integrate an AI help system with access to platform content to provide real-time assistance, understand and clarify context, and feature an interface accessible at any point during user interaction.

Functional requirement 3: Detection System

The CodeQA system shall automatically check for and suggest similar previously asked and answered questions when a user attempts to post a new question, using a content similarity detection algorithm.

Functional Requirement 4: Tags System

The CodeQA system shall include a tagging system, allowing users to categorize questions by topics and view all queries under a specific tag.

Functional Requirement 5: Users Profiles

The CodeQA system must offer a user profile interface comprising 'User Details' and 'Activity Level' sections, displaying a summary of personal achievements and a detailed log of the user's activities respectively.

Non-Functional Requirements

Non-functional requirements specify the quality attributes, system performance standards, and operational characteristics a software system must meet. Unlike functional requirements, which outline what the system should do, non-functional requirements describe how the system performs under various conditions and how it must behave regarding usability, reliability, performance, maintainability, and security. For instance, a platform like CodeQA might have non-functional requirements related to handling a high volume of concurrent users, ensuring data is encrypted for privacy, providing cross-platform compatibility to function seamlessly on various devices, and maintaining short response times even under peak loads. These requirements are essential for ensuring the system's sustainability, efficiency, and user satisfaction, forming the backbone of user experience and system robustness. They help ensure that the software is usable and effective in the real world and meets or exceeds industry standards and user expectations.

Non-functional Requirement 1: Portability

The CodeQA system shall be operable across various platforms including Windows, macOS, iOS, and Android operating systems, without the need for additional user-configured settings, ensuring full functionality on desktops, phones, and tablets.

Non-functional Requirement 2: Speed

The CodeQA system shall handle at least 1000 processed transactions per second, and the user/event response time shall not exceed 5 seconds during peak usage.

Non-functional Requirement 3: Security

The CodeQA system shall encrypt the personal information of customers using a public-key encryption algorithm.

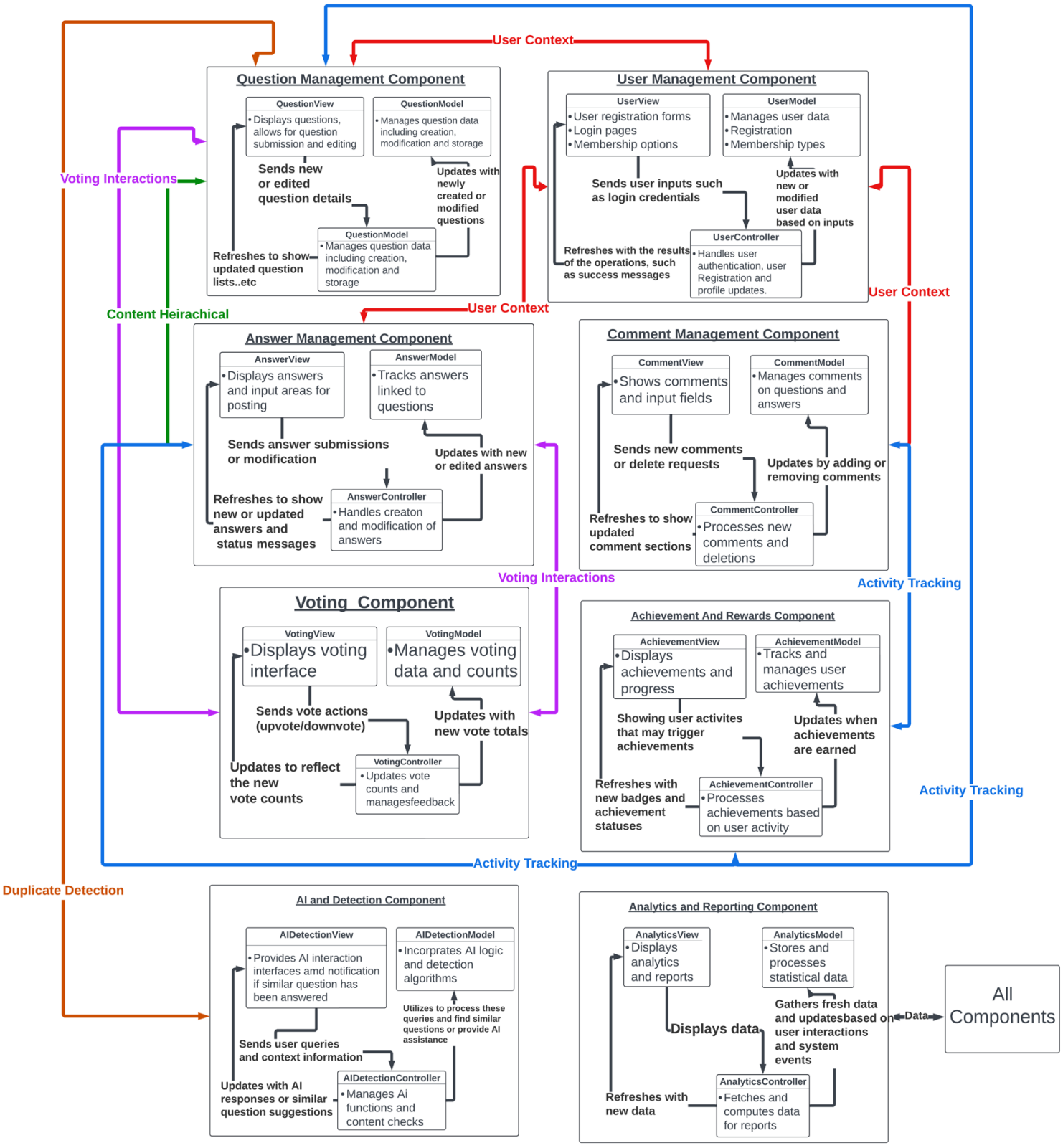
Non-functional Requirement 4: Training

Users shall be able to perform basic functions within the CodeQA system after no more than one hour of training, and post-training, the average number of errors per hour of system use shall not exceed two.

Non-functional Requirement 5: Ease of use

The CodeQA system shall provide an interface that enables users to complete any common task with no more than three clicks. Additionally, users shall be able to access any help frame within the system in no more than two actions from any point in the application.

Architectural Design: MVC



Architecture design in software development refers to the high-level structuring of a system, defining the components or modules of the system, and their interactions to meet the technical and operational requirements of the project. Effective architecture design ensures that the software is scalable, maintainable, and meets the business goals and user needs.

A popular architectural pattern often employed in web and desktop applications is the Model-View-Controller (MVC) architecture. This pattern separates the application into three interconnected components, each responsible for different aspects of the application's functionality:

- **Model:** This layer manages the data and business logic of the application. It responds to requests for information about its state (usually from the view), and instructions to change state (usually from the controller). In essence, it represents the application's dynamic data structure, independent of the user interface. It directly manages the data, logic, and rules of the application.
- **View:** This component displays the data that the model contains and sends user commands (e.g., button clicks, menu selections) to the controller. The view is responsible for rendering the data from the model in a manner that is suitable for interaction, typically through a user interface.
- **Controller**: Acting as an intermediary between Model and View, the controller receives user input and initiates a response by making calls to model objects and updating the view as necessary. This separation helps manage complexity when building or maintaining a large application, as it helps separate input logic from business logic and UI.

The following text outlines the components of our system, as represented in the accompanying diagram:

Components

Component 1: User Management

Manages user data, authentication, and membership types, ensuring secure access and personalized user experience.

Component 2: Question Management

Handles the creation, modification, and storage of questions, providing a central hub for user inquiries.

Component 3: Answer Management

Manages user-submitted answers linked to specific questions, facilitating knowledge sharing and community interaction.

Component 4: Comment Management

Oversees the addition and moderation of comments on both questions and answers, enhancing discussion and feedback mechanisms.

Component 5: Voting System

Allows users to vote questions and answers up or down, determining the visibility and ranking of content based on community approval.

Component 6: Achievement and Rewards System

Tracks user activities and awards badges and other achievements based on predefined criteria, motivating user engagement.

Component 7: AI and Detection Systems

Integrates AI to assist users in real-time and employs algorithms to detect and suggest similar questions, improving efficiency and user experience.

Component 8: Analytics and Reporting

Collects and analyzes data from all aspects of the platform to provide insights into user behavior and content effectiveness.

The following text outlines the relationships between the components, as represented in the accompanying diagram. Note: The color of each sentence corresponds to the relationship color used in the diagram:

Relationships

Relationship 1: User Management ↔ Question, Answer, Comment Management (User Context)

Provides user context to all posts for content ownership and user-specific management. It ensures that every user interaction is authenticated and personalized, effectively linking content creation and engagement activities to the individual user's account.

Relationship 2: Question ↔ Answer Management (Content hierarchical)

Establishes a hierarchical link where answers are directly associated with their respective questions.

Relationship 3: Voting System ↔ Question, Answer (Voting Interactions)

Enables the voting functionality for questions and answers, influencing content prominence based on user votes.

Relationship 4: Achievement and Rewards System ↔ Question, Answer, Comment Management

Monitors and records user interactions with questions, answers, and comments to drive the achievements system.

Relationship 5: AI and Detection Systems ↔ Question Management
Enhances question posting with AI-driven suggestions for existing similar content, preventing duplicate entries.

Relationship 6: Analytics and Reporting ↔ All Components

Feeds data from each component into the analytics system to track and report on platform activity and trends.

Architectural Decisions: Reward System & Framework

Design issue		Choosing the best approach to implement an achievement system that accurately tracks user activities, evaluates them against predefined criteria, and awards badges.
Context		The CodeQA platform seeks to motivate user engagement through a sophisticated achievement system. The chosen solution must seamlessly integrate with the existing platform and handle the complexity of various user activities.
Quality attributes		Maintainability, Integration Complexity, Implementation Time.
Architectural option 1	Identifier	Custom-Built Achievement System
	Description	Develop a custom-made achievement system integrated directly into the CodeQA platform's backend.
	Status	Rejected.
	Evaluation	Developing a custom system offers high customization but at the cost of increased complexity and longer development time, potentially causing significant delays.
	Rationale	Rejected due to the high complexity and extended development time required, which pose significant risks to the project's schedule and resource allocation.
Architectural option 2	Identifier	Third-Party Gamification Service
	Description	Implement an achievement system using a third-party service like Badgeville or Gamify, which offers pre-built solutions adaptable to various platforms.
	Status	Accepted.
	Evaluation	Third-party services streamline integration and accelerate development, offering scalable and flexible solutions that adapt well to the evolving needs of the CodeQA platform.
	Rationale	Chosen for its straightforward implementation process, lower complexity, and ability to quickly integrate into existing systems. This option minimizes development time and allows the project to focus on other critical areas of development.

Design issue		Choosing a web application framework that supports rapid development, simplicity, and flexibility for the CodeQA platform.
Context		The CodeQA platform requires a framework that allows for quick prototyping and is lightweight enough to not impose significant overhead on development.
Quality attributes		Development Speed, Simplicity, Flexibility, Scalability.
Architectural option 1	Identifier	Comprehensive Framework (e.g., Django)
	Description	Using a full-stack web framework like Django which comes with a lot of built-in functionalities.
	Status	Rejected.
	Evaluation	Although Django offers an extensive set of features, which can accelerate development, its monolithic structure might be more than is necessary for the CodeQA platform's requirements and could potentially slow down the development due to its complexity.
	Rationale	Rejected because its structured nature and bundled components might limit flexibility and add unnecessary complexity for the CodeQA platform's needs.
Architectural option 2	Identifier	Micro Framework (e.g., Flask)
	Description	Implementing Flask, a micro-framework that provides the basics of web application development while remaining lightweight and extensible.
	Status	Accepted.
	Evaluation	Flask offers enough flexibility for developers to pick and choose components and libraries as needed, which can facilitate rapid development and easier maintenance.
	Rationale	Chosen for its simplicity, ease of use, and the modular approach that aligns with the need for a quick development cycle and the potential for scaling up the application as it grows.

Software Process: Agile

Given the nature of the CodeQA platform with potentially ambiguous requirements, I recommend using the Agile software development process. This approach supports flexibility, allowing for adjustments in design and development as students gain a clearer understanding of project expectations through ongoing interactions with instructors. Agile facilitates frequent check-ins and iterative refinement, ensuring that each iteration reflects a more accurate interpretation of the project requirements. This process not only enhances practical learning but also ensures the final product closely aligns with university expectations, promoting effective collaboration and communication within the team.