A computer network diagram with a globe and arrows

Description automatically generated with medium confidenceCairo University

Faculty of Computers and Artificial intelligence Department of Computer Sciences

Groofy Code

**Supervised by**

*Dr. Mohamad Abdelwahab*

TA: Nada Nasser

**Implemented by**

|  |  |
| --- | --- |
| *20200136* | *Hazem Adel Khalel Nabawy* |
| *20200359* | *Omar Mohmed Mostafa* |
| *20200268* | *Doha Abdelbasset Ahmed* |
| *20200514* | *Marwa Ahmed Mohamed Mubarak* |
| *20200501* | *Mahmoud Abdelrady Gad* |

Graduation Project

Academic Year 2023-2024

Final Year Documentation

## **Chapter 1: Introduction**

## 1.1 Motivation (Abstract)

In the current landscape of online coding platforms, coding enthusiasts often face a fragmented experience. Many platforms focus primarily on coding challenges, but they lack a cohesive ecosystem that caters to diverse coding interests. **Groofy Code** emerges as a solution to this problem by integrating challenges, problem-solving activities, and competitive 1 vs 1 matches within a comprehensive framework. Existing platforms often lack an inclusive social aspect, which hinders collaboration and community building among coding enthusiasts.

Our motivation to develop Groofy Code stems from a passion for fostering a collaborative and engaging environment for coding enthusiasts. We aim to provide a unified platform where users can seamlessly transition between challenges, problem-solving activities, and competitive matches while building connections with like-minded individuals through clans and an interactive chat system. Groofy Code is designed to be more than just a coding platform; it is a community where users can grow, learn, and compete together.

## 1.2 Problem Definition

The current landscape of online coding platforms presents significant challenges due to the lack of a cohesive and all-encompassing solution for coding enthusiasts. Existing platforms often specialize in isolated aspects, such as coding challenges, competitive matches, or collaborative problem-solving, creating a fragmented experience for users. This fragmentation hinders the development of a unified community where individuals can seamlessly transition between various coding activities, showcase their skills, and engage in social interactions.

The absence of an integrated platform poses several challenges for coding enthusiasts:

1. **Fragmented Experience**: Users must navigate multiple platforms to fulfill their coding interests, leading to a disjointed experience.
2. **Limited Social Interaction**: The lack of an inclusive social aspect on existing platforms prevents users from building meaningful connections with other coding enthusiasts.
3. **Inconsistent Skill Development**: With isolated functionalities, users may struggle to find a balanced approach to developing their coding skills across different activities.
4. **Lack of Community Engagement**: The absence of a unified platform makes it difficult to foster a sense of community and collaboration among users.

These challenges highlight the need for a comprehensive and dynamic environment that caters to diverse preferences and provides a seamless user experience.

## 1.3 Our Solution

**Groofy Code** redefines the online coding experience by offering a feature-rich platform that addresses the shortcomings of existing platforms. Our solution includes the following key features:

1. **Solver Profile**:
   * **Personalized Journey**: Each user has a detailed profile that showcases their achievements, progress, and social connections. This personalized journey helps users track their growth and stay motivated.
2. **Problem Solving**:
   * **Diverse Challenges**: Groofy Code offers a wide range of coding challenges that cater to different skill levels and interests, ensuring that users are constantly engaged and challenged.
3. **1 vs 1 Challenges**:
   * **Competitive Element**: Users can participate in 1 vs 1 matches, which add a competitive element to the platform. These matches are automatically timed and have a global rating impact, encouraging users to improve their skills and compete at higher levels.
4. **Clan System**:
   * **Collaboration and Community Building**: The clan system allows users to form or join clans, fostering collaboration and community building. Clans can participate in group challenges and compete against each other, enhancing the social aspect of the platform.
5. **Friends and Online Status**:
   * **Enhanced Communication**: Users can add friends and see their online status, making it easier to connect and communicate with other coding enthusiasts.
6. **Customizable Unrated Challenges**:
   * **Flexibility**: Users can create and customize their own unrated challenges, providing flexibility in the types of problems they wish to tackle and practice.
7. **Global Rating System**:
   * **Performance Evaluation**: The global rating system evaluates both individual and clan performance, encouraging healthy competition and providing users with a clear sense of their standing within the community.
8. **Velocity Matches**:
   * **Quick Challenges**: Groofy Code introduces quick 15-minute velocity matches designed to challenge users to solve problems rapidly, testing their coding speed and accuracy under tight time constraints.

**Groofy Code's** technology stack, featuring React, Java Spring Boot, and machine learning, underscores its commitment to providing a seamless, engaging, and technologically advanced coding environment. By integrating these cutting-edge technologies, Groofy Code ensures that users have a robust and enjoyable experience, making it the go-to platform for coding enthusiasts seeking a comprehensive and dynamic environment.

### **1.4 Gantt chart of project time plan**

### **1.5 Development Methodology: Agile**

In the software development life cycle, we use the agile methodology, which promotes iterative development, collaboration, and flexibility to adapt to changing requirements. Agile is an appropriate method because it aims to achieve these goals by enabling continuous feedback, early and frequent delivery of functional components, and a focus on customer satisfaction. The entire software development process is divided into iterative cycles called sprints, each resulting in a potentially shippable product increment. The phases in our agile methodology are:

#### Requirements Gathering

In this phase, we collaborate with stakeholders to gather all the functional and non-functional requirements for the project. The requirements include the main functionality of the platform, such as user profiles, problem-solving interfaces, 1 vs 1 challenges, clan system, and chat system.

#### Sprint Planning

In each sprint planning session, we prioritize the requirements and create a sprint backlog, which includes the tasks and features to be developed in the upcoming sprint. This planning ensures that the team has a clear understanding of the goals and deliverables for each sprint.

#### Development

The development phase involves the actual coding and implementation of the features planned for the sprint. We follow best practices, coding standards, and agile principles to ensure high-quality and maintainable code. Development is done in small, manageable increments, allowing for continuous integration and frequent delivery of working software.

#### Testing

During the testing phase, we conduct thorough testing of the features developed in the sprint to ensure they meet the requirements and function as intended. Automated testing tools and quality assurance techniques are used to verify the accuracy and reliability of the code. Testing is an integral part of each sprint, ensuring that any issues or bugs are identified and addressed promptly.

To ensure the reliability of our API endpoints, we will use **Swagger** for API documentation and testing. Swagger allows us to design, build, document, and test our APIs with ease. This tool will help us ensure that our APIs function correctly and meet the specified requirements by providing a user-friendly interface for testing each endpoint. It also facilitates automated testing and validation of API responses, contributing to a robust and reliable system.

#### Review and Retrospective

At the end of each sprint, we hold a sprint review meeting to demonstrate the completed features to stakeholders and gather feedback. This feedback is crucial for making any necessary adjustments and planning future sprints. We also conduct a sprint retrospective to reflect on the sprint process, identify areas for improvement, and implement changes to enhance the team's efficiency and effectiveness.

#### Maintenance and Continuous Improvement

Once the platform is deployed, it will require ongoing maintenance and updates to ensure it continues to function correctly and meet user needs. We develop a maintenance schedule and plan to address any issues or bugs that arise, as well as implement new features and enhancements based on user feedback and evolving requirements.

By following the agile methodology, we ensure that Groofy Code is developed in a flexible, efficient, and customer-focused manner, delivering a high-quality platform that meets the needs of coding enthusiasts.

## 1.6 Tools and Technologies Used in the Project

### 1.6.1 Back-end

* **Java**
* **Java Spring Boot**
* **MySQL**
* **Spring Security**
* **Spring JPA**
* **Spring WebSocket**
* **Spring REST APIs**
* **JWT (JSON Web Token)**
* **Swagger** (for API documentation and testing)
* **Integration with Codeforces** (Third Party)
* **Integration with Firebase** (Third Party)
* **Postman** (for API testing)

### 1.6.2 Front-end

* **React.js**
* **TypeScript**
* **Redux**
* **SCSS**
* **MathJax**
* **PrimeReact**
* **WebSocket**
* **CodeMirror**
* **Formik**

## 2.1 Related Work

In the landscape of online coding platforms, several existing platforms provide coding challenges and competitive programming, much like Groofy Code. This section highlights the similarities and differences between Groofy Code and some of the most notable platforms in this space.

### Similar Features

#### Codeforces

**Codeforces** is a popular platform known for its regular programming contests and a vast collection of coding problems. It offers features such as:

* **Solving Problems**: Users can solve problems independently to improve their coding skills.
* **User Interaction**: Users can search for other users, send messages, and view profiles.
* **Group Competitions**: Codeforces supports the creation of groups and allows users to compete as teams in contests.

#### AtCoder

**AtCoder** is another competitive programming platform that provides regular contests and a large repository of problems. Key features include:

* **Contests**: AtCoder organizes various contests, from beginner to advanced levels, to challenge participants.
* **Problem Practice**: Users can practice problems from past contests.
* **Rating System**: Participants are rated based on their performance in contests.

#### V-Judge (Virtual Judge)

**V-Judge** aggregates problems and contests from various online judges, allowing users to practice and compete across different platforms. It offers:

* **Problem Aggregation**: Users can solve problems sourced from multiple online judges.
* **User and Group Interaction**: Similar to other platforms, V-Judge allows users to search for and interact with other users and join groups.

### Distinguishing Features of Groofy Code

While the aforementioned platforms share similarities with Groofy Code, there are several key differences that set Groofy Code apart:

#### Competitive Programming

* **1 vs 1 Matches**: Unlike traditional platforms, Groofy Code emphasizes 1 vs 1 competitive matches, offering a more personalized and direct competition experience.
* **Team vs Team Matches**: In addition to individual matches, Groofy Code supports team-based competitions, enhancing collaborative problem-solving skills.
* **Velocity Matches**: Groofy Code introduces quick 15-minute matches designed to challenge users to solve problems rapidly, testing their coding speed and accuracy under tight time constraints.

#### Personalized Problem Recommendations

* **Machine Learning**: Groofy Code employs a machine learning model to recommend problems tailored to the user's skill level and learning pace. This ensures that users are consistently challenged but not overwhelmed, providing a customized learning experience.

#### Dynamic Environment

* **Badges and Trophies**: To foster a sense of achievement and recognition, Groofy Code incorporates badges, rankings, and trophies, motivating users to engage more actively.

#### Comprehensive Practice

* **Solo Matches**: In addition to competitive matches, Groofy Code offers solo practice sessions, allowing users to improve their skills in a non-competitive environment.
* **Customizable Challenges**: Users can create and customize unrated challenges, providing flexibility in the types of problems they wish to tackle.

#### Collaborative Aspect

* **Clan System**: Groofy Code enhances the social aspect by allowing users to form or join clans. This fosters a sense of community and collaboration, encouraging teamwork and shared learning.

### Summary

While existing platforms like Codeforces, AtCoder, and V-Judge provide robust features for coding challenges and competitive programming, Groofy Code distinguishes itself with its unique focus on personalized competitive matches, a dynamic and engaging environment, and enhanced collaborative features. These elements create a comprehensive and enriching experience for coding enthusiasts, setting Groofy Code apart from its counterparts.

### References

* [Codeforces](https://codeforces.com/)
* [AtCoder](https://atcoder.jp/)
* [V-Judge](https://vjudge.net/)

## 3. System Analysis

### 3.1.1 Functional Requirements

1. **User Registration and Authentication**
   * Users should be able to register with a unique username and password.
   * Users must be able to log in securely using their credentials.
   * Password recovery/reset functionality should be available.
2. **Solver Profile Management**
   * Users should be able to create and edit their solver profiles.
   * Profile information should include a display name and optional bio.
   * Users can link their solver profiles to their social media accounts.
3. **Global Score and Badges**
   * The platform should calculate and display a global score for each solver.
   * Badges should be awarded based on achievements and milestones.
4. **Problem Solving**
   * Solvers can access a diverse range of coding challenges.
   * Each challenge should specify the allowed programming languages.
   * The platform should support the submission and execution of code.
   * A submission history and a code editor with syntax highlighting should be available.
   * Use a machine learning model to match users with appropriate problems.
5. **Clan System**
   * Solvers can create, join, and manage clans.
   * Each clan should have a logo, achievements, and bio.
   * Clans should have a ranking based on collective performance.
6. **Friends System**
   * Solvers can search for friends and send friend requests.
   * Users can manage their list of friends.
7. **Chat System**
   * Clan members can communicate through an interactive chat system.
   * The chat system should support reactions, replies, and emojis.
8. **Rating System**
   * The platform should maintain a global rating system for individual solvers and clans.
   * Ratings should be updated based on performance in rated challenges and challenge difficulty.
9. **Matchmaking System Using Machine Learning**
   * Utilize historical user data, including user rate, types of problems solved, and problems rate, to build a recommendation model that ensures fairness in matches between users.
   * Implement collaborative filtering techniques to suggest potential opponents with similar skill levels and problem-solving strategies.
   * Recommend problems to the user for practicing by predicting the rate of the problem to be solved and its tags based on the historical data of the user.

### 3.1.2 Non-Functional Requirements

1. **Scalability**
   * The system should scale horizontally to accommodate an increasing number of users.
2. **Performance**
   * Response time for key actions, such as code submission and challenge initiation, should be within 2 seconds.
3. **Reliability**
   * The platform should reliably track and update user scores and ratings in real-time.
4. **Usability**
   * The user interface should be intuitive, with clear navigation and a responsive design for various devices and screen sizes. Aim for an average user satisfaction rating of at least 4 out of 5 in user feedback surveys.
5. **Portability**
   * The platform should be accessible and fully functional across major web browsers, including Chrome, Firefox, Safari, and Edge.
6. **Security**
   * Utilize secure coding practices to protect user data and prevent unauthorized access. Implement encryption for data transmission and storage, including user passwords.