

CODEKONNECT: COLLABORATIVE CODING AND MESSAGING PLATFORM

A Project Report

Submitted by

Shriya Kale	112103065
Himanshu Kamdi	112103067
Soham Kumthekar	112103076

of

TY (Computer Engineering)

Under the guidance of

Dr. Tanuja R. Pattanshetti

COEP Technological University



DEPARTMENT OF COMPUTER ENGINEERING
COEP Technological University

April, 2024

DEPARTMENT OF COMPUTER ENGINEERING

COEP Technological University

CERTIFICATE

Certified that this project, titled “CodeKonnnect: Collaborative Coding Messaging Platform” has been successfully completed by

Shriya Kale	112103065
Himanshu Kamdi	112103067
Soham Kumthekar	112103076

and is approved for the fulfilment of the requirements of “Software Engineering Mini Project- Stage II”.

SIGNATURE

Dr. Tanuja R. Pattanshetti
Project Guide

Department of Computer Engineering

COEP Technological University,

Shivajinagar, Pune - 5.

Abstract

The Collaborative Coding and Messaging Platform project aims to develop a versatile web application that facilitates seamless collaboration and efficient file management for distributed teams. In today's dynamic work environment, remote collaboration and effective communication are essential for team productivity and success. The project addresses these needs by providing users with a unified platform for real-time messaging, channel-based discussions, and comprehensive file collaboration.

The project methodology involves rigorous analysis of requirements, careful selection of technologies, systematic system design, meticulous implementation, and thorough testing. Key features of the application include real-time messaging, channel creation and management, file uploads, code editing with real-time collaboration, commit history tracking, and repository navigation.

Through the implementation of the Collaborative Team Communication and File Management System, users can streamline their communication, collaborate on files effectively, and track project progress in real-time. The system empowers teams to overcome geographical barriers, enhance coordination, and boost productivity.

The outcomes of the project demonstrate the efficacy of the Collaborative Team Communication and File Management System in addressing the challenges faced by distributed teams. By providing a centralized platform for communication and file collaboration, the application facilitates smoother workflow, fosters teamwork, and contributes to the overall success of collaborative projects.

Contents

1	Synopsis	5
1.1	Project Title	5
1.2	Internal Guide	5
1.3	Problem Statement	5
1.4	Plan of Project Execution	6
2	Problem Definition and scope	7
2.1	Problem Definition	7
2.1.1	Goals and objectives	7
2.1.2	Statement of Scope	8
2.2	Software context	8
2.3	Major Constraints	9
2.4	Outcome	10
2.5	Applications	11
2.6	Software Resources Required	11
3	Project Plan	13
3.1	Project Schedule	13
3.1.1	Gantt Chart	13

4	Software requirement specification	14
4.1	Introduction	14
4.1.1	Use-cases	14
4.1.2	Use Case View	15
4.2	Data Model and Description	17
4.2.1	Data objects and Relationships	17
4.3	Functional Model and Description	18
4.3.1	Data Flow Diagram	18
4.3.2	Non Functional Requirements:	22
4.3.3	Design Constraints	23
5	Testing	24
5.1	Testing Scenarios	24
5.2	Testcases with scenarios	26
5.3	Requirement Traceability Matrix	27
6	Detailed Design Document	28
6.1	Component Design	28
6.1.1	Class Diagram:	28
6.1.2	Sequence Diagram:	30
6.1.3	Component Diagram:	31
6.1.4	Swimlane Diagram:	32
6.1.5	Deployment Diagram:	33
6.2	Navigation Flow	34
7	Summary and Conclusion	38

List of Figures

4.1	Use case diagram	16
4.2	Entity Relationship diagram	18
4.3	DFD Level0	19
4.4	DFDLevel1	20
4.5	DFDLevel2	21
5.1	Testing scenarios	25
5.2	Test cases with scenarios	26
5.3	Tracebility matrix	27
6.1	Class diagram	29
6.2	Sequence diagram	30
6.3	Component diagram	31
6.4	Swimlane diagram	32
6.5	Deployment diagram	33
6.6	Login	34
6.7	Sign in with google	34
6.8	Add channel	35
6.9	Description	35
6.10	Repo contents	36

6.11	commits	36
6.12	code editor	37

Chapter 1

Synopsis

1.1 Project Title

CODEKONNECT: COLLABORATIVE CODING AND MESSAGING PLATFORM

1.2 Internal Guide

Dr. Tanuja R. Pattanshetti

1.3 Problem Statement

The Collaborative Team Communication and File Management System project endeavors to tackle the challenges faced by distributed teams in effectively collaborating on projects, managing files and messaging. In today's globalized and remote work landscape, teams often encounter difficulties in maintaining clear communication channels, coordinating file access and version control, and ensuring seamless collaboration among team members across different locations and time zones. The project seeks to provide a centralized platform for team communication and file collaboration, improve productivity, and foster a cohesive team environment.

1.4 Plan of Project Execution

Task	Start Date	End Date	Duration	Resources
Problem statement finalization	08-Jan-24	15-Jan-24	8	Shriya,Himanshu,Soham
Project Plan	16-Jan-24	31-Jan-24	16	Shriya,Himanshu
Requirement Analysis	01-Feb-24	05-Feb-24	5	Himanshu,Soham
Design Flow charts and ER diagrams	06-Feb-24	09-Feb-24	4	Shriya,Soham
Register and Login authentication using firebase	10-Feb-24	13-Feb-24	4	Soham
Messaging and Files upload	13-Feb-24	18-Feb-24	6	Shriya
Create repo and Github Integration	19-Feb-24	29-Feb-24	10	Soham
View Commit History, upload and download files	01-Mar-24	10-Mar-24	10	Himanshu,Soham
Code Editor Integration	11-Mar-24	20-Mar-24	10	Himanshu
Final UI changes	21-Mar-24	31-Mar-24	10	Shriya
Testing	01-Apr-24	10-Apr-24	10	Shriya,Himanshu,Soham
Documentation and project report	11-Apr-24	15-Apr-24	5	Shriya,Himanshu,Soham

Chapter 2

Problem Definition and scope

2.1 Problem Definition

2.1.1 Goals and objectives

Goal and Objectives:

- Goals
 - Develop a collaborative messaging and file management system to enhance team communication and collaboration in a professional setting.
- Objectives
 - **Real-time Messaging:** Implement real-time messaging functionality to facilitate instant communication between team members.
 - **Channel-based Discussions:** Organize discussions into channels to streamline communication based on topics or projects.
 - **Code Editor Integration:** Integrate a code editor into the platform to allow team members to collaborate on code files in real-time. ‘
 - **File Interface:** Accessing files from Github, making changes and committing to Github. Add and view commit history.

2.1.2 Statement of Scope

- **Target Features:**

- Real-time messaging functionality for instant communication.
- Channel-based discussions for organizing conversations based on topics or projects.
- Advanced file management capabilities, including file uploading, sharing, and editing.
- Integration of a code editor to facilitate real-time collaboration on code files.
- Version control features to track changes in files and ensure data integrity.

- **Target Audience:**

- This application is designed for professionals and teams who require efficient communication and collaboration tools in their workflow.
- It caters to individuals and groups across various industries, including software development, project management, marketing, and more.

2.2 Software context

The project operates within the context of web application development, emphasizing real-time messaging, channel-based discussions, and file management functionalities. It involves the integration of front-end technology React.js to build a responsive and intuitive user interface. On the back end, the application utilizes frameworks like Firebase and React-Redux for efficient state management, component-based architecture, real-time messaging functionality and file management features.

Additionally, the project employs version control system like Github for managing changes in code files and ensuring collaboration among team members. Database

management systems such as Firebase Realtime Database is utilized to store user data, channel information, and message history.

The software context of the project encompasses web development, real-time communication, database management, and version control technologies, all aimed at facilitating seamless team collaboration and file management.

2.3 Major Constraints

- **Data acquisition**

- **Integration with External Services:** Ensuring compatibility and seamless integration with third-party services for real-time messaging and file management.
- **API Rate Limits:** Adhering to API rate limits imposed by external services to avoid disruptions in data retrieval and processing.
- **Data Consistency:** Ensuring consistency and accuracy of data fetched from external sources to provide reliable information to users.

- **Client-side constraints**

- **Offline Functionality:** Implementing features that allow users to access and interact with the application even when offline, with data synchronization capabilities upon reconnection.
- **Cross-Platform Compatibility:** Ensuring the application functions seamlessly across different devices and operating systems to provide a consistent user experience.

- **Real-time Collaboration**

- **Concurrency Control:** Implementing mechanisms to manage concurrent edits and prevent conflicts when multiple users are editing the same file simultaneously.
- **Real-time Updates:** Ensuring timely propagation of changes made by one user to all other participants in the channel to maintain coherence and transparency.

2.4 Outcome

The primary outcome of the project is the successful development and deployment of a collaborative messaging and file management platform that enhances team communication and collaboration. Key components of the outcome include:

- **Robust Messaging System:** Implementation of a robust messaging system that supports real-time communication between team members. Users can exchange messages, share files, and engage in channel-based discussions efficiently.
- **Advanced File Management:** Integration of advanced file management features that enable users to upload, view, edit, and collaborate on files within the application. Version control and real-time synchronization ensure that all team members have access to the latest file versions.
- **User-Friendly Interface:** Development of a user-friendly interface that provides intuitive navigation and seamless interaction. The interface is designed to optimize user experience and productivity.
- **Real-time Collaboration:** Facilitation of real-time collaboration on files, allowing users to edit documents. Changes are synced instantly, enabling teams to work together effectively regardless of location.

2.5 Applications

The messaging and file management platform developed in this project offers various applications aimed at enhancing team communication, collaboration, and productivity:

- **Team Communication:** The platform serves as a centralized communication hub for teams, allowing members to exchange messages, share updates, and discuss project-related matters in real-time.
- **File Collaboration:** Users can collaborate on files within the platform, enabling multiple team members to view, edit, and comment on documents simultaneously. This fosters seamless collaboration and eliminates version control issues.
- **Knowledge Sharing:** Users can share knowledge and resources within the platform, promoting information exchange and continuous learning. This fosters a culture of knowledge sharing and enhances team members' skill development.
- **Client Collaboration:** The platform can be used for client collaboration, enabling teams to share project updates, deliverables, and feedback securely. Client access controls ensure confidentiality and privacy.
- **Training and Onboarding:** Organizations can utilize the platform for training and onboarding purposes, providing new employees with access to relevant documents, resources, and training materials. This accelerates the onboarding process and ensures consistent training delivery.

2.6 Software Resources Required

1. **Database Management System (DBMS):** Firebase Realtime Database or Firestore is required to store user data, channel information, messages, and file metadata.

Firebase offers NoSQL cloud databases that provide real-time synchronization and offline support, making it suitable for applications requiring instant updates and scalability.

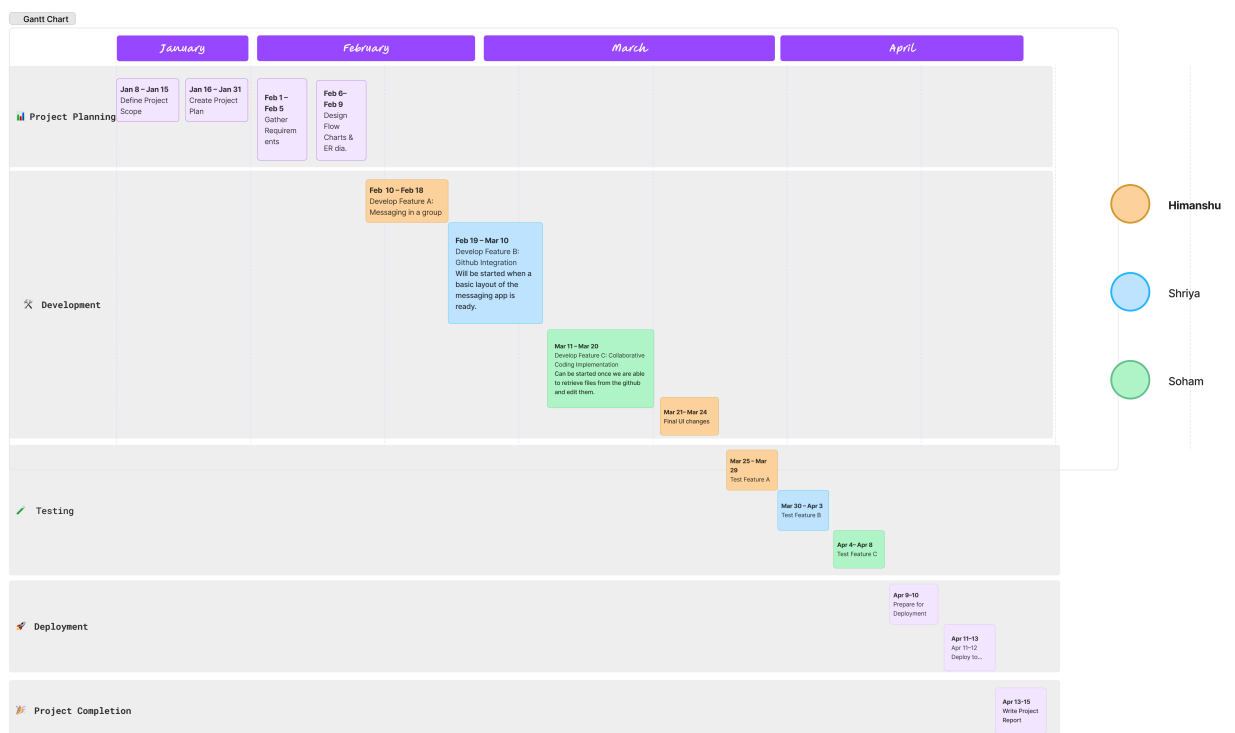
2. **Web Framework:** A web framework is necessary for building the backend logic and RESTful APIs of the platform. Frameworks like Node.js can be utilized to handle routing, request handling, and database interactions.
3. **Authentication Service:** Firebase Authentication provides ready-to-use authentication services including email/password authentication, social login with providers like Google. It offers secure authentication mechanisms simplifying the implementation of user authentication and authorization.
4. **Frontend Framework:** A frontend framework or library is required for building the user interface of the platform. Options include React.js which offer component-based development and efficient state management.
5. **File Interface:** Accessing files from Github, making changes and committing to those changes to Github and viewing commit history.

Chapter 3

Project Plan

3.1 Project Schedule

3.1.1 Gantt Chart



Chapter 4

Software requirement specification

4.1 Introduction

The software requirement specification (SRS) outlines the functional and non-functional requirements of the messaging and file management application. It describes the use-cases and features that the application must support to meet the needs of its users.

4.1.1 Use-cases

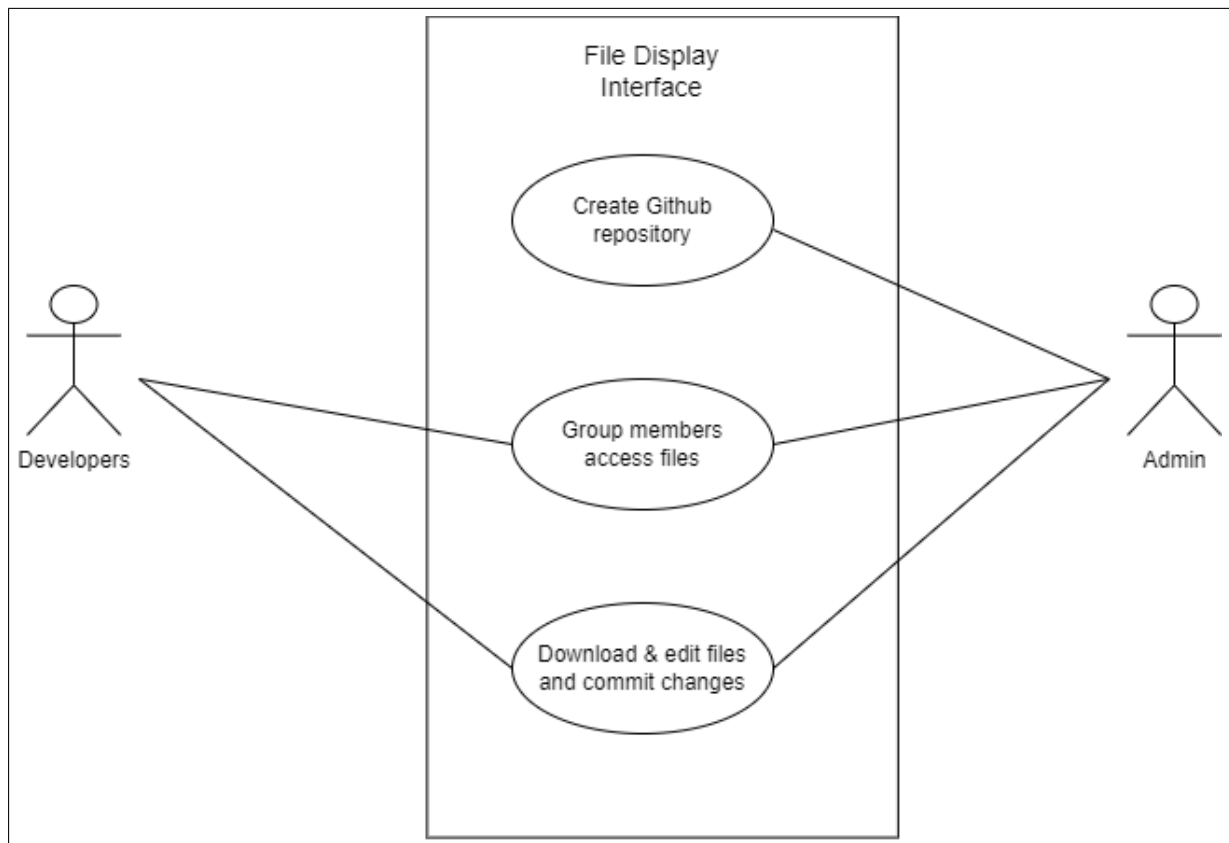
- **Real-Time Messaging:** Users can send and receive messages in real-time within channels or direct messages.
- **Channel-based Discussions:** Users can create channels for group discussions and collaborate with team members.
- **File Management:** Users can upload, view, and download files within channels, facilitating document sharing and collaboration.
- **Code Editor Integration:** Users can open files in a code editor within the application and make real-time changes visible to all users in the channel.
- **Commit History Viewing:** Users can view the commit history of files, tracking

changes made over time.

- **Repository File Access:** Users can access all files of a repository within the application, facilitating seamless navigation and file management.
- **Commit Addition:** Users can add commits to files directly within the application, contributing to version control and collaboration.

4.1.2 Use Case View

Use Case Diagram:



Use case diagram

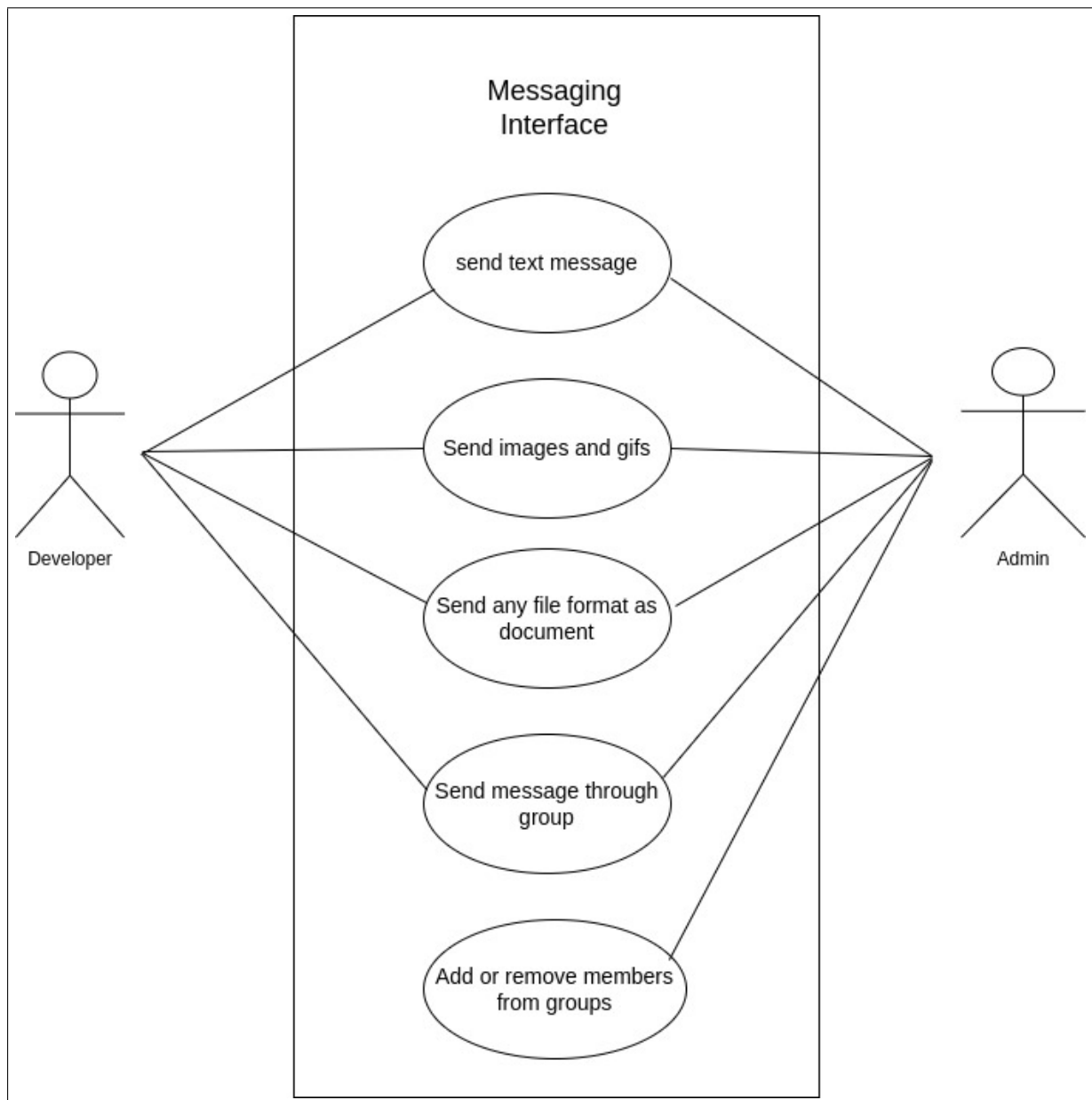
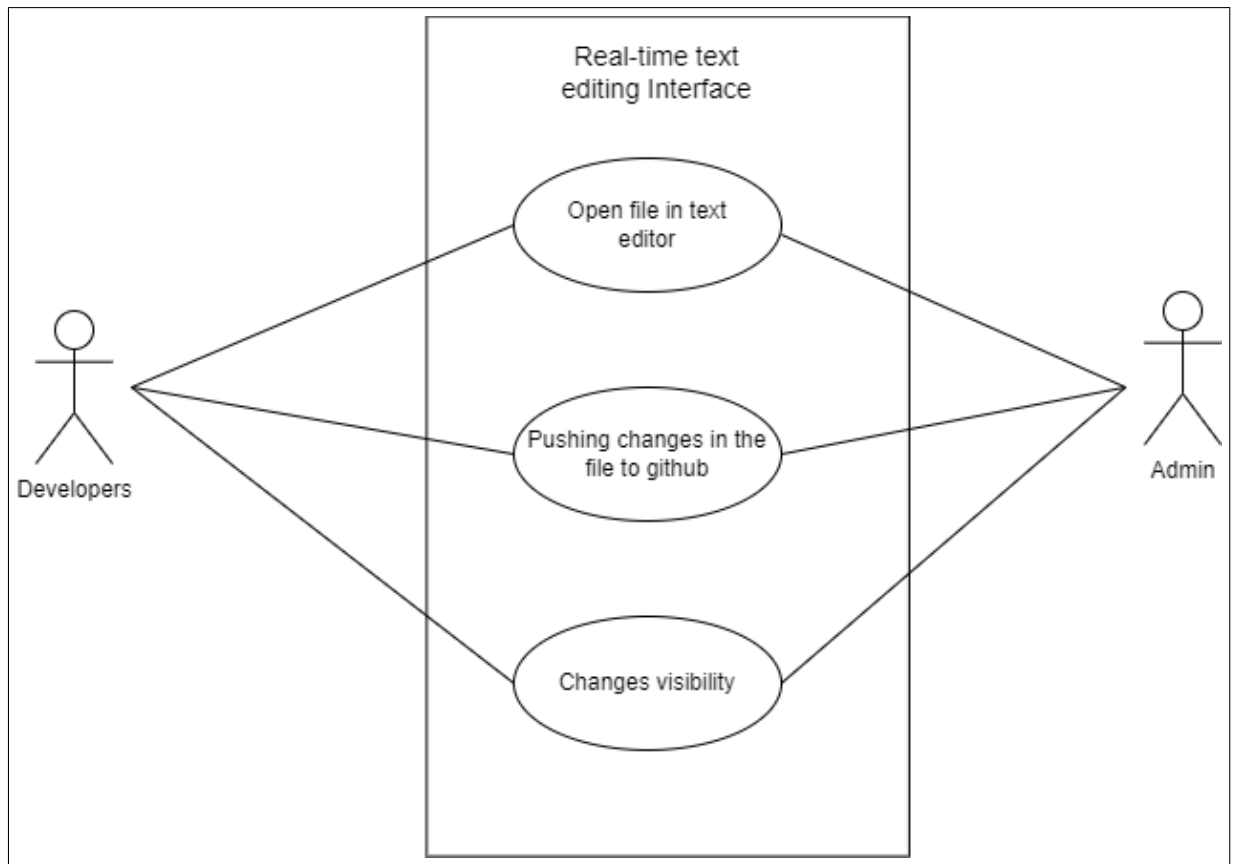


Figure 4.1: Use case diagram



Use case diagram

4.2 Data Model and Description

4.2.1 Data objects and Relationships

- **User:** Represents a registered user of the messaging and file management application. Users have accounts with unique identifiers and authentication credentials.
- **Channel:** Represents a communication channel within the application where users can participate in discussions, share files, and collaborate with team members. Channels have properties such as name, description, and list of members.
- **Message:** Represents a message sent within a channel or direct message conversation. Messages have attributes like sender, content, timestamp, and associated channel or conversation.

- **File:** Represents a file uploaded by a user within a channel. Files have properties such as name, size, type, and associated channel.
- **Commit:** Represents a commit made to a file within the application's integrated code editor. Commits have attributes like author, timestamp, message, and associated file.

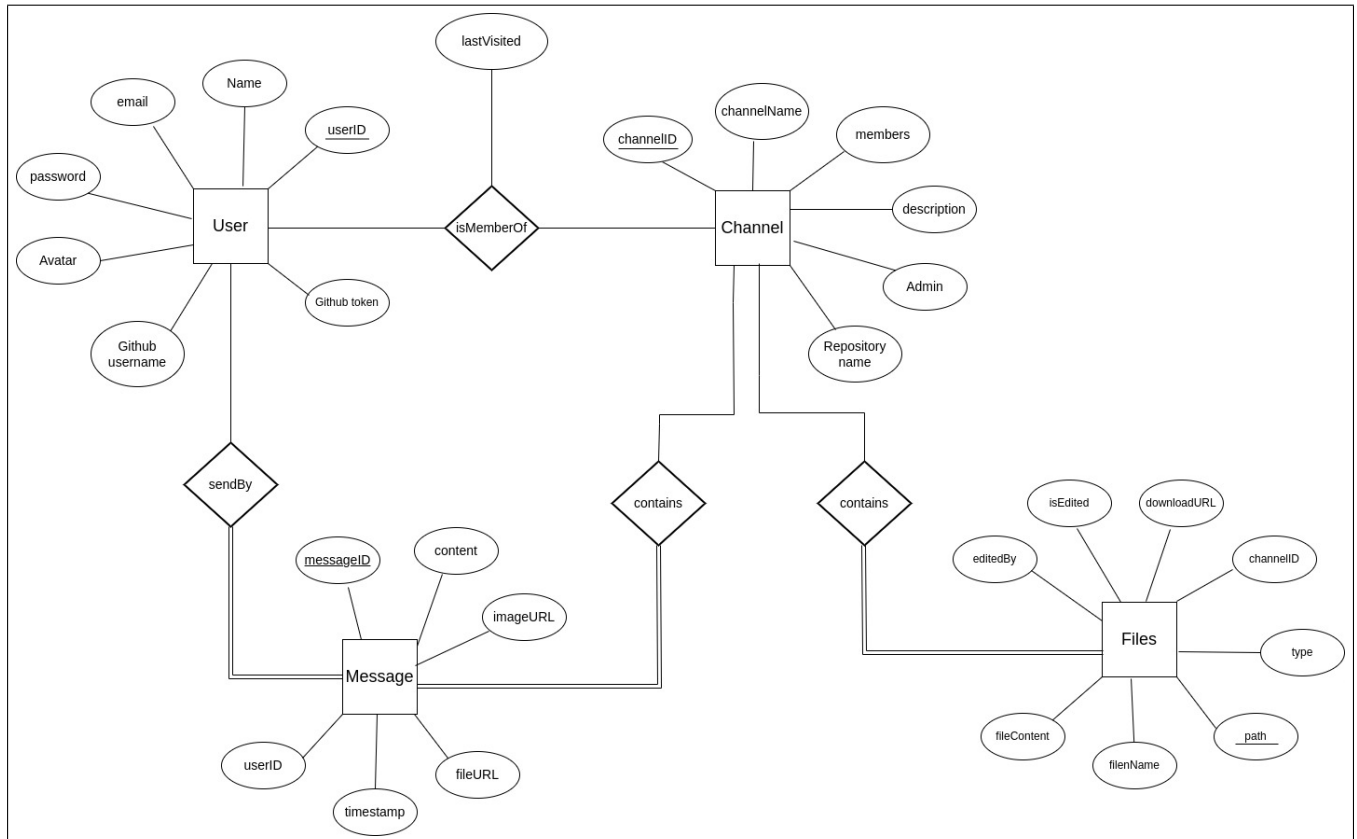


Figure 4.2: Entity Relationship diagram

4.3 Functional Model and Description

4.3.1 Data Flow Diagram

Level 0 Data Flow Diagram

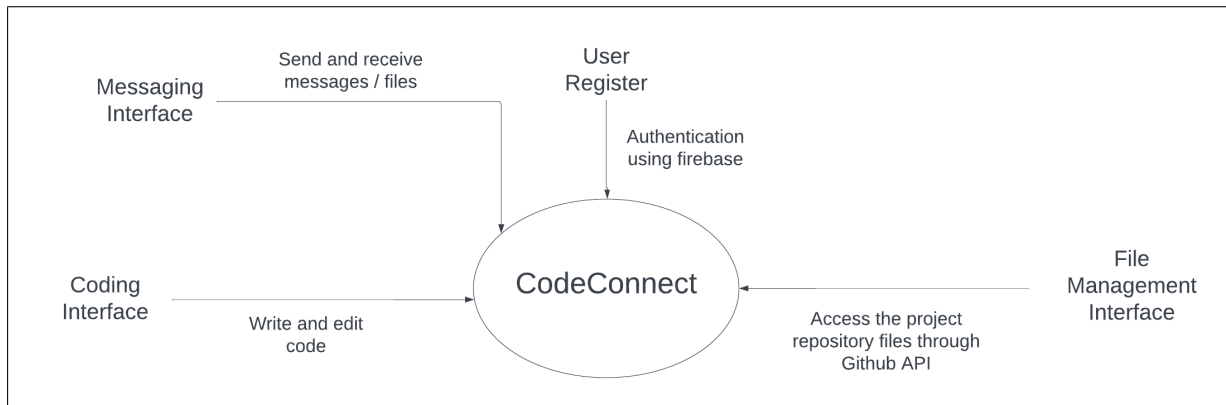


Figure 4.3: DFD Level0

Level 1 Data Flow Diagram

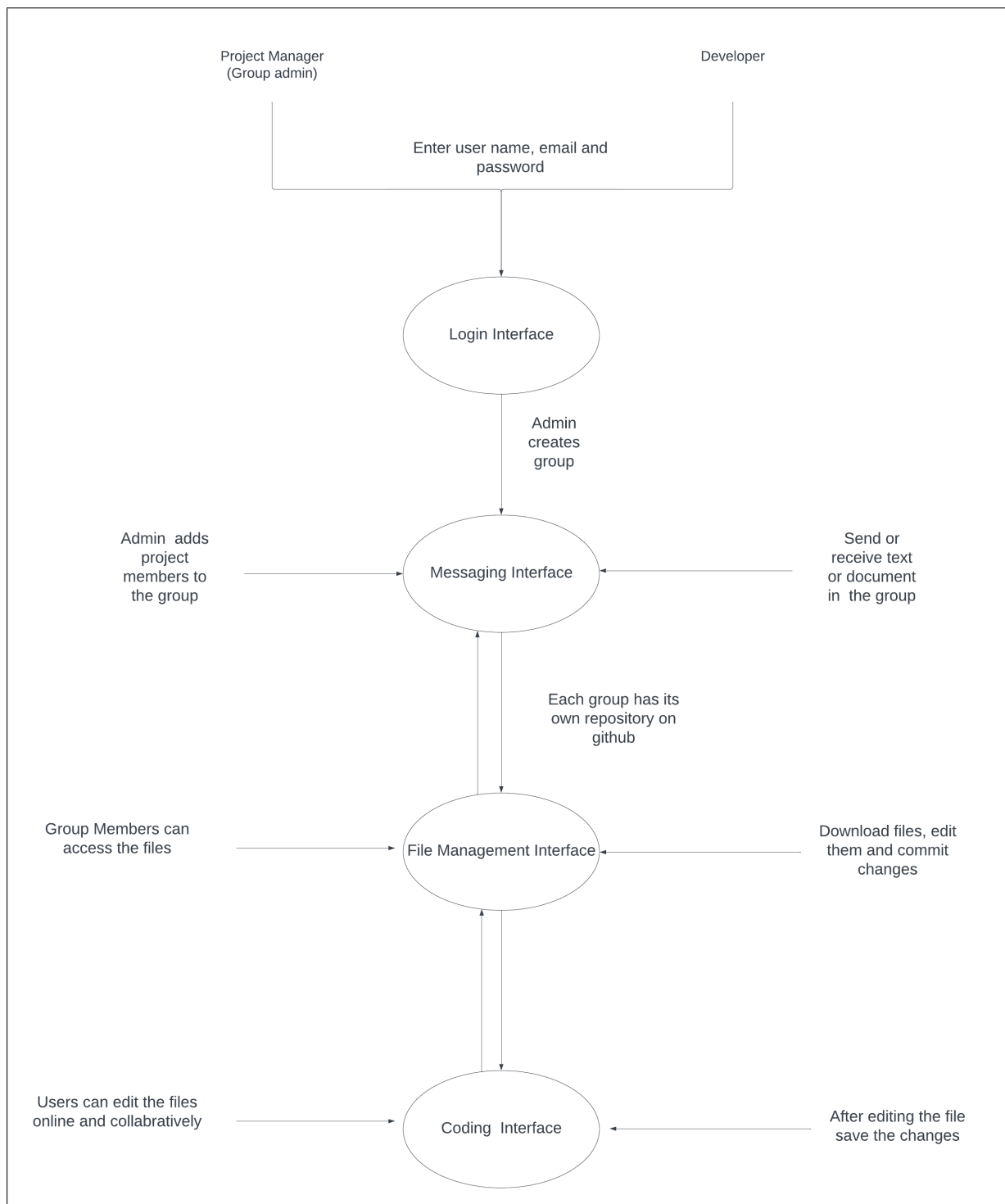


Figure 4.4: DFDLevel1

Level 2 Data Flow Diagram

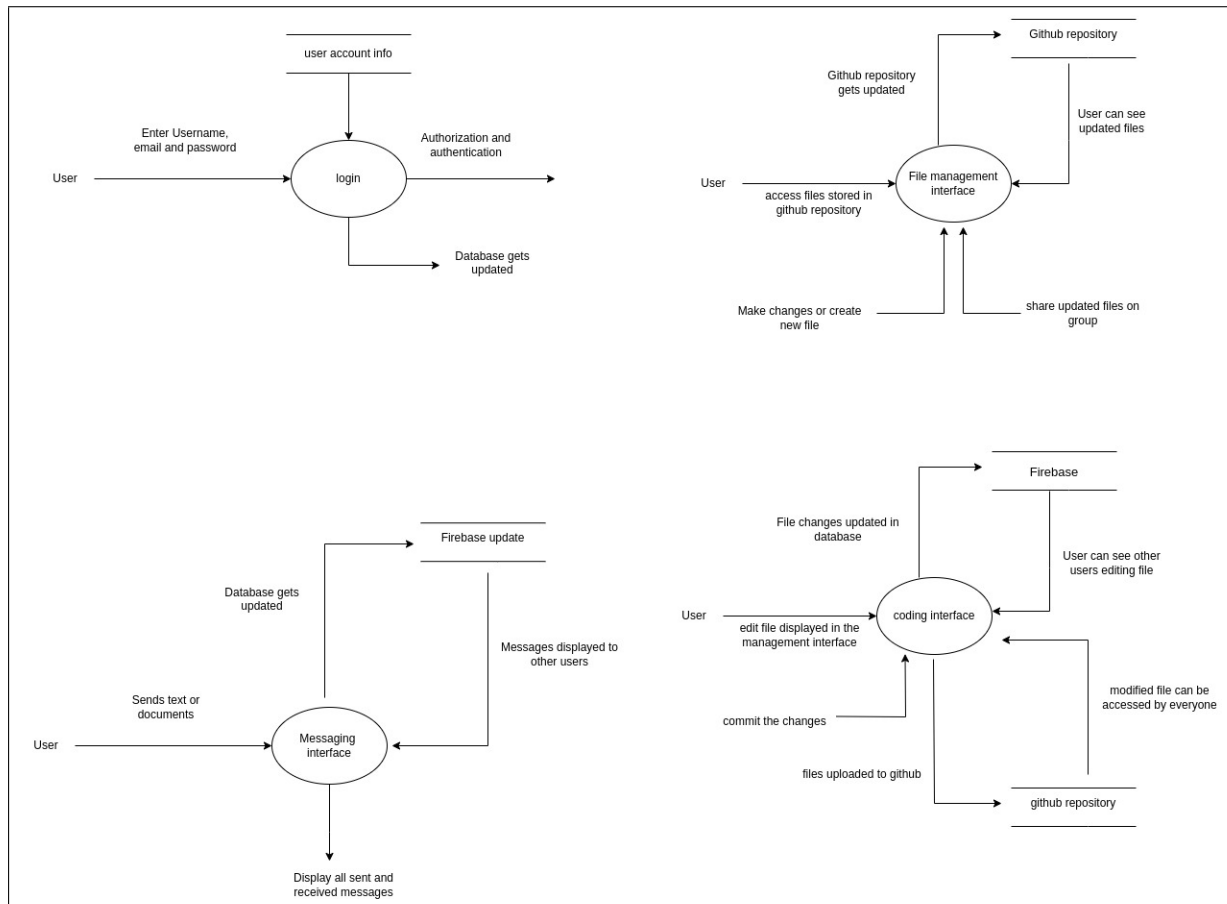


Figure 4.5: DFDLevel2

4.3.2 Non Functional Requirements:

Performance Requirements

- **Response Time:** The system should respond promptly to user actions such as sending messages, uploading files, and viewing channel content.
- **Scalability:** The system should be able to handle a growing number of users and channels without significant degradation in performance. This may involve optimizing server resources and implementing scalable architecture.
- **Concurrency:** The system should support concurrent user interactions within channels, ensuring smooth communication and collaboration among team members.
- **Error Handling:** Errors should be handled gracefully, with informative error messages displayed to users and detailed logs recorded for system administrators to diagnose and resolve issues.
- **Database Performance:** Ensure efficient performance of Firebase Firestore for storing and retrieving user data, channel information, messages, and file metadata. Optimize database queries and indexing for faster data access.

Safety and Security Requirements:

- **Data Privacy:** Ensure that user data, including messages and files, is securely stored and transmitted using encryption protocols to protect against unauthorized access or data breaches.
- **Input Validation:** Implement input validation mechanisms to sanitize user inputs and prevent injection attacks or malicious activities.
- **Content Integrity:** Verify the integrity of user-uploaded files and messages to prevent the dissemination of harmful content or malware within the application.

4.3.3 Design Constraints

- **Data acquisition**

- **Firestore Integration:** Adapting to the Firestore platform's structure and limitations for data storage and retrieval.
- **Real-time Updates:** Ensuring that real-time updates are synchronized across all clients without significant latency.

- **Client-side constraints**

- **Offline Functionality:** Providing basic functionality for users when offline, with automatic synchronization upon reconnection.
- **Cross-Platform Compatibility:** Ensuring compatibility across various devices and web browsers to facilitate seamless user experience.

- **Data processing**

- **Optimization for Real-time Updates:** Optimizing data processing algorithms to handle real-time updates and notifications efficiently.

Chapter 5

Testing

5.1 Testing Scenarios

Case	Test Input	Expected Results
TS-1 Register	Leave one input blank	Display message to fill all the fields
	Password less than 8 characters long	Password length should be greater than 8
	Confirm password different from entered password	Password and Confirm Password does not match
	Register with same email	Email address is already in use by another account
	All inputs correct	Successfully registered
TS-2 Login	Not Registered	Prevent login
	Incorrect username or password	Wrong Username OR Password
	Forgot Password	Reset password pin
	Correct email and password	Redirect to messaging interface
TS-3 Logout	Sign out	Redirect to login page
TS-4 Messaging	Not provided Github access token	Display message to add your git details
	Search particular message	All messages containing that string are displayed
	Search empty message	No message is displayed
	Add channel with all empty fields	No channel is created
	Clicked on star button	Channel is displayed at top of channels list
	Clicked on channel name	Channel description and members are displayed
	Multiple user signed in to the system	Online users are shown with green circle
	Unread personal messages	Message count displayed in notification
TS-5 File Interface	Clicked on file interface button	Contents and commit history of the repository are displayed
	Clicked on the file name	File is opened in text editor
	Download button clicked	File is downloaded
	Upload file button clicked	Interface to upload file/image is displayed
	Add commit button clicked	Interface to add commit after making changes in the repository

Figure 5.1: Testing scenarios

5.2 Testcases with scenarios

TS#	Test Scenario	TC#	Test Case
TS-1	Register	TC-1.1	One input blank
		TC-1.2	Confirmed password not equal to entered password
		TC-1.3	All inputs correct
TS-2	Login	TC-2.1	Not registered
		TC-2.2	Incorrect password
		TC-2.3	Incorrect Email
		TC-2.4	Empty Fields
		TC-2.5	Correct email and password
TS-3	Logout	TC-3.1	Click Logout button
TS-4	Messaging	TC-4.1	Send Text Messages
		TC-4.2	Send images and gifs
		TC-4.3	Send any file format as document
		TC-4.4	Add or remove members from groups
TS-5	File Interface	TC-5.1	View repository content
		TC-5.2	View commit history
		TC-5.3	Add commit
		TC-5.4	Upload file
		TC_5.5	Open file in text editor
		TC_5.6	Changes visible to other members
		TC-5.7	Changes in file pushed to github

Figure 5.2: Test cases with scenarios

5.3 Requirement Traceability Matrix

Project Name: Collaborative Coding & Messaging Platform(CodeKconnect)					
Business Requirement		Functional Requirement		Test Case Doc	
ID #	Use Case	ID #	Use Case	Priority	Test Case ID#
BR_1	User Registration & Login	FR_1	Efficient registration for new users	High	TC_1.3
		FR_2	Successful Login	High	TC_2.5
BR_2	Messaging	FR_3	Send Text Messages	High	TC_4.1
		FR_4	Send images and gifs	Medium	TC_4.2
		FR_5	Send any file format as document		TC_4.3
		FR_6	Send message through group	High	TC_4.1
		FR_7	Add or remove members from groups		TC_4.4
BR_3	File Display	FR_8	View repository content	Medium	TC_5.1
		FR_9	View commit history		TC_5.2
		FR_9	Add commit	High	TC_5.3
		FR_9	Upload file		TC_5.4
BR_4	Realtime text editing	FR_10	User open file in text editor	High	TC_5.5
		FR_11	Changes visible to other members		TC_5.6
		FR_12	Changes in file pushed to github	Medium	TC_5.7

Figure 5.3: Traceability matrix

Chapter 6

Detailed Design Document

6.1 Component Design

6.1.1 Class Diagram:

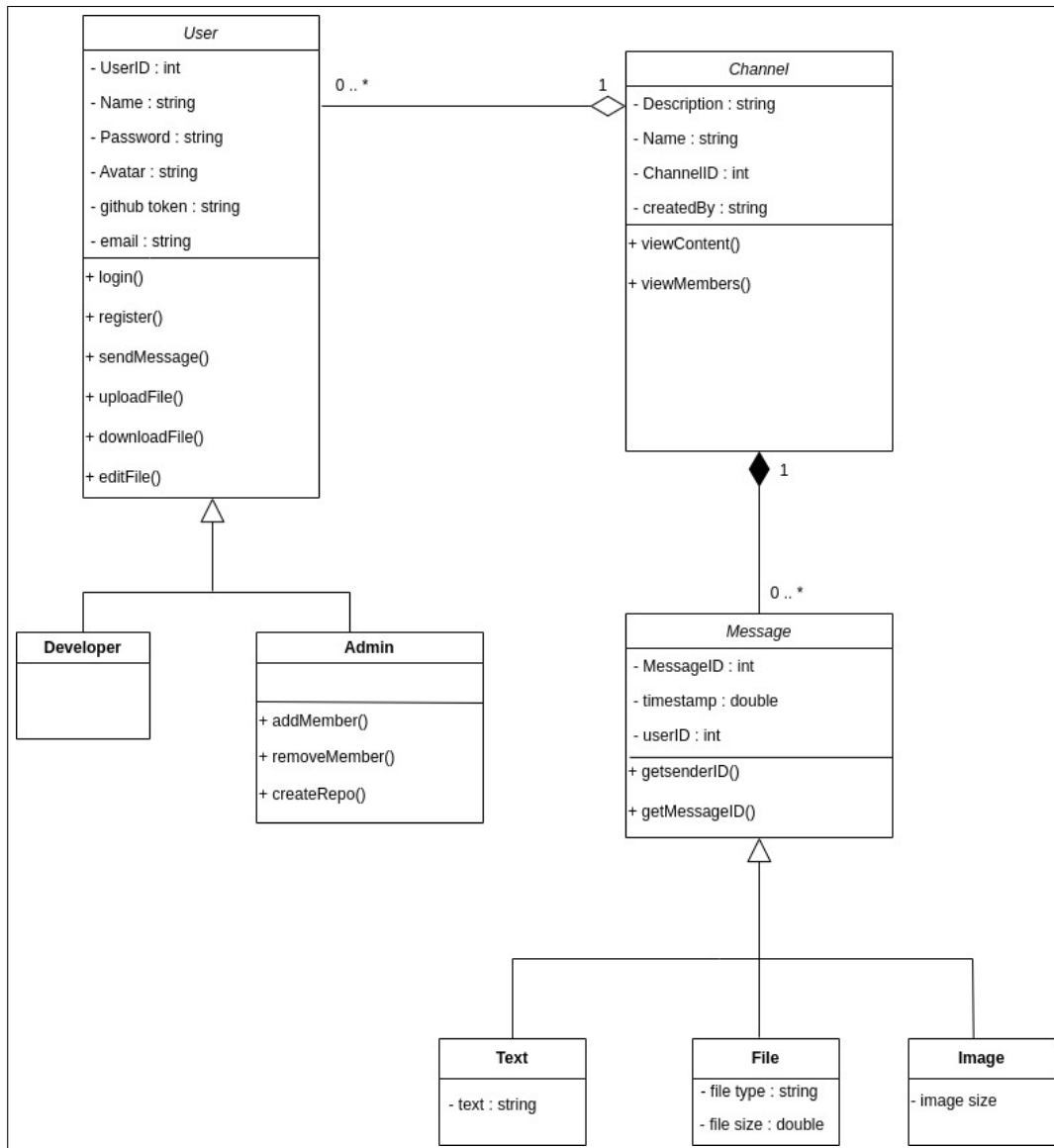


Figure 6.1: Class diagram

6.1.2 Sequence Diagram:

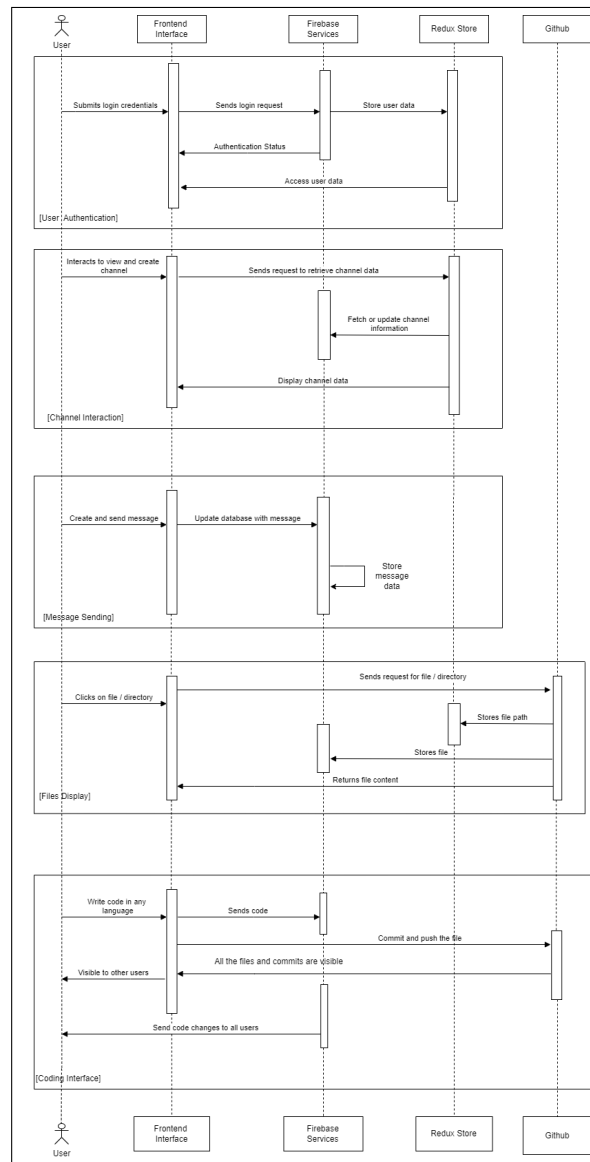


Figure 6.2: Sequence diagram

6.1.3 Component Diagram:

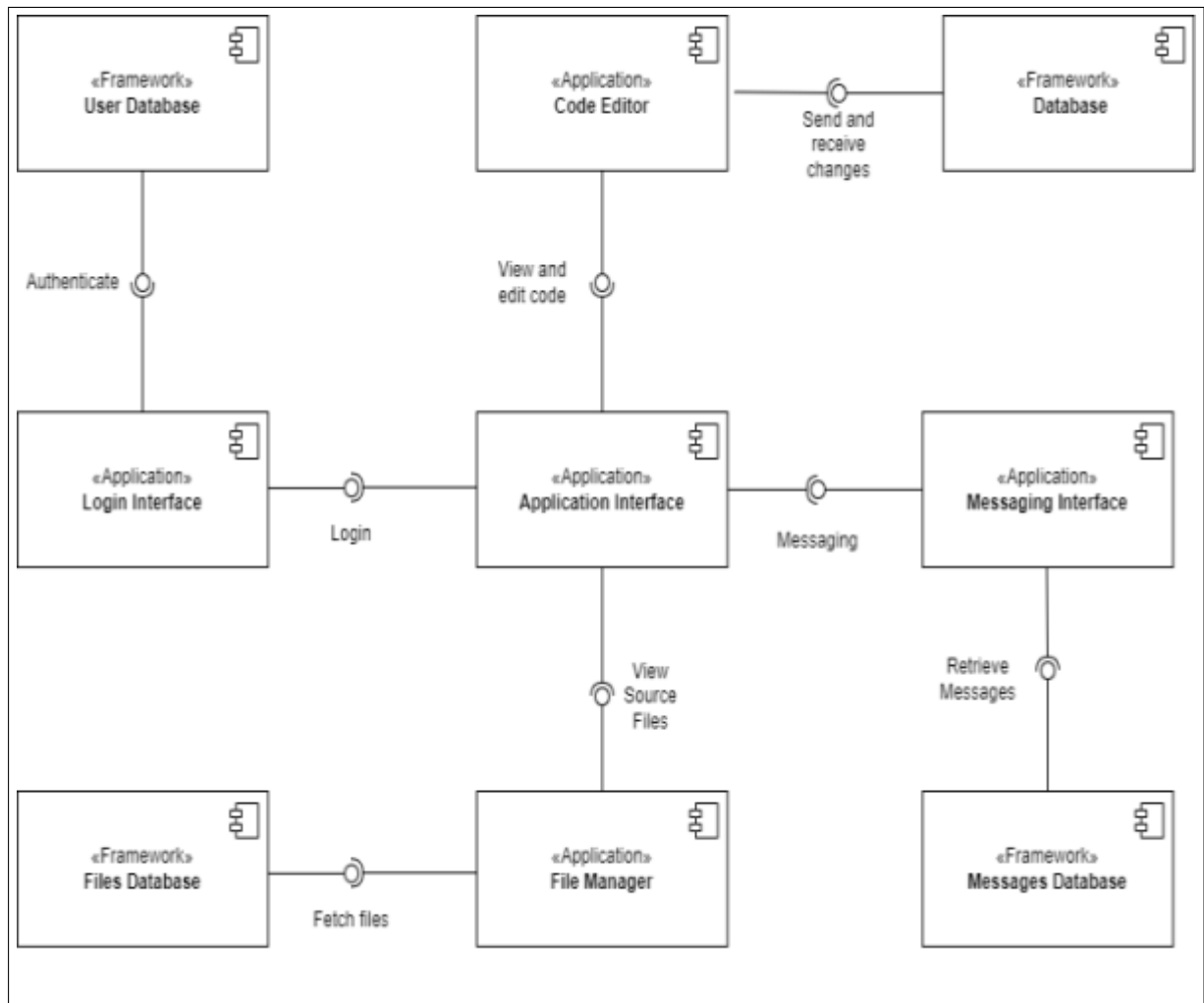


Figure 6.3: Component diagram

6.1.4 Swimlane Diagram:

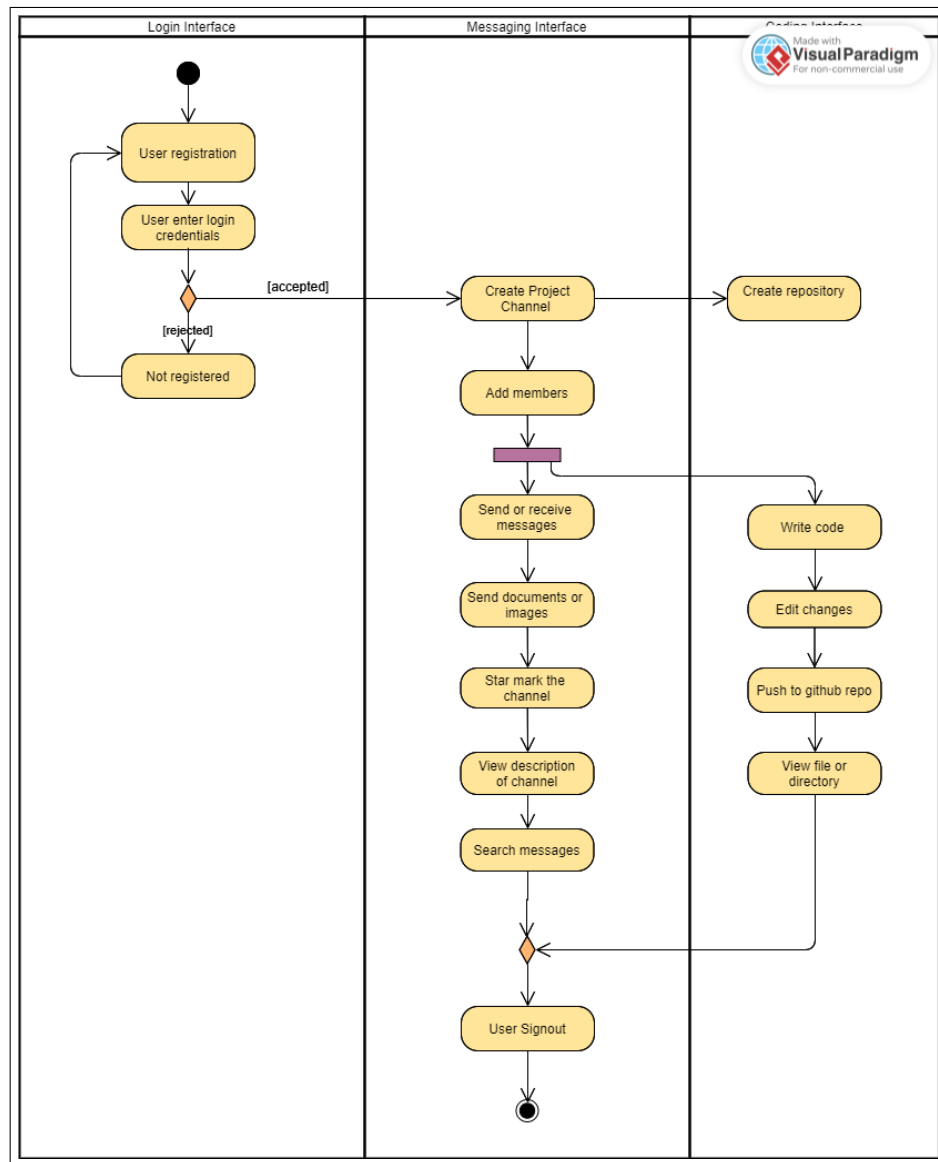


Figure 6.4: Swimlane diagram

6.1.5 Deployment Diagram:

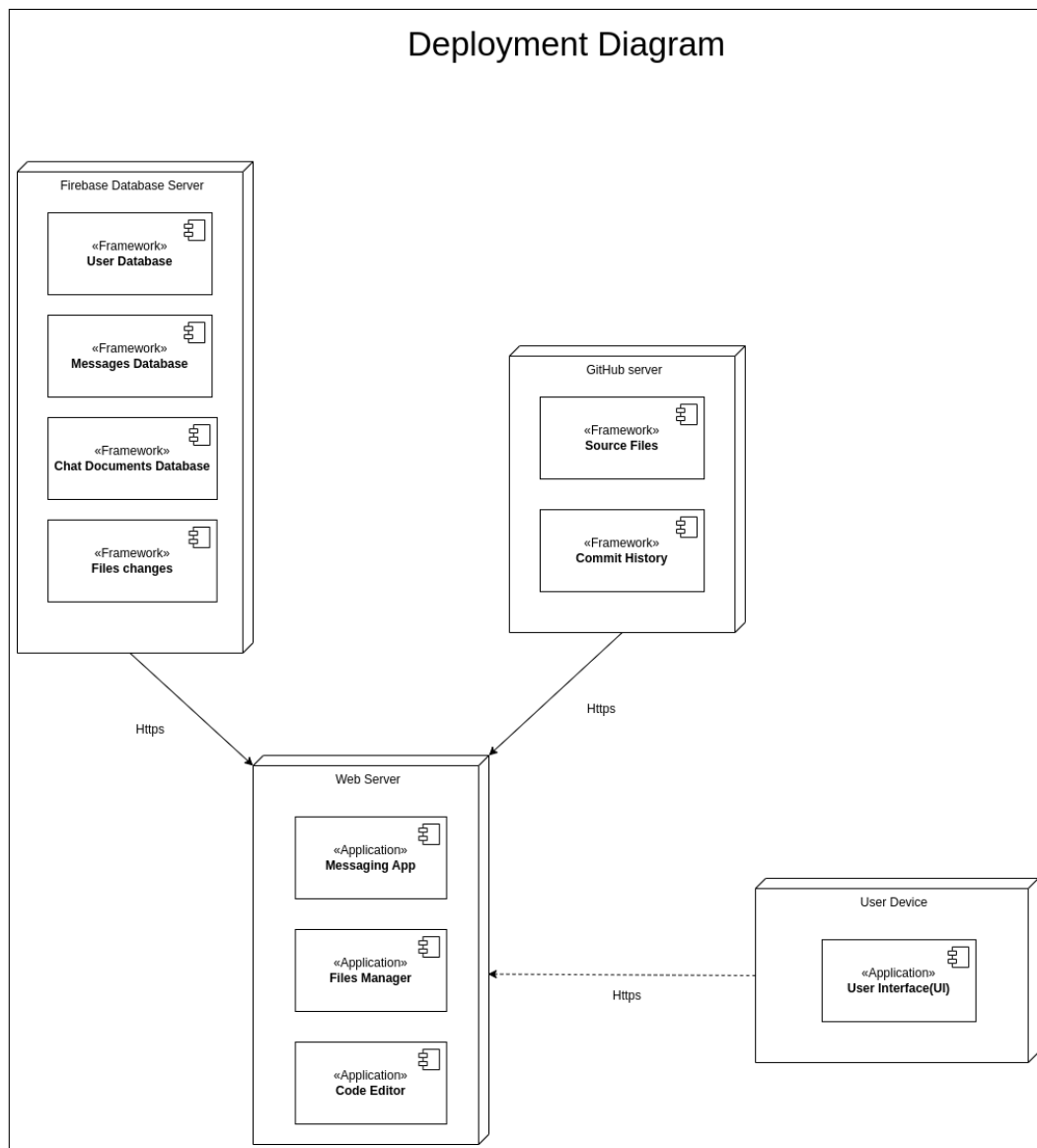


Figure 6.5: Deployment diagram

6.2 Navigation Flow

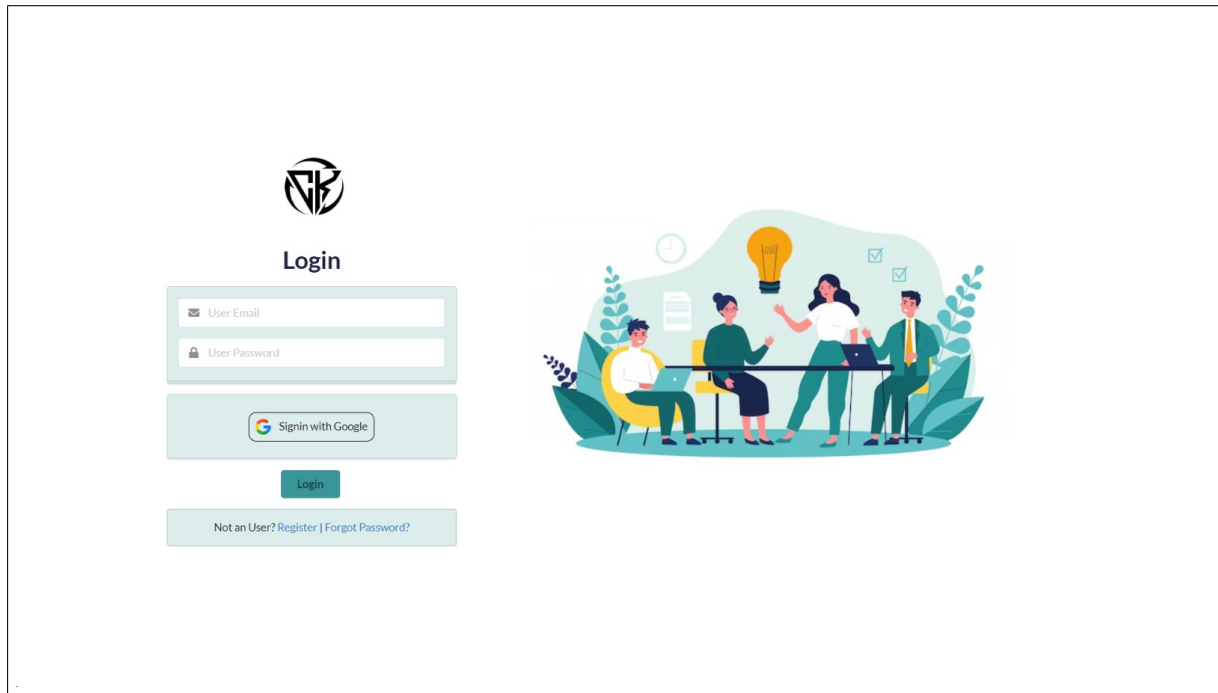


Figure 6.6: Login

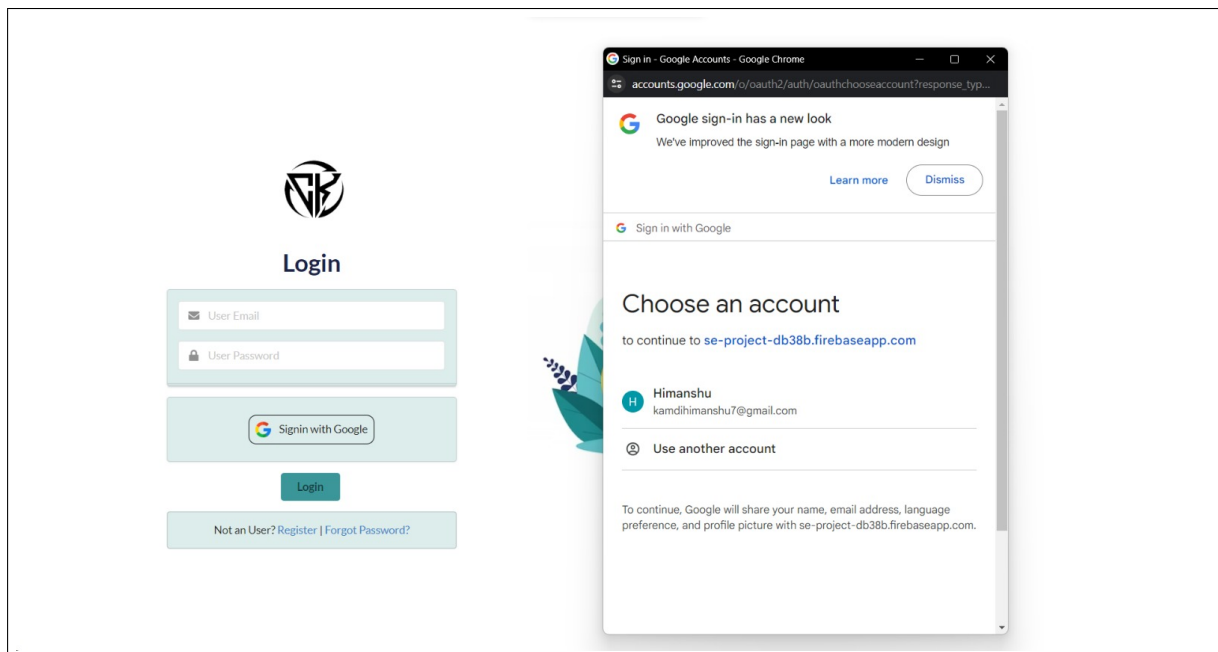


Figure 6.7: Sign in with google

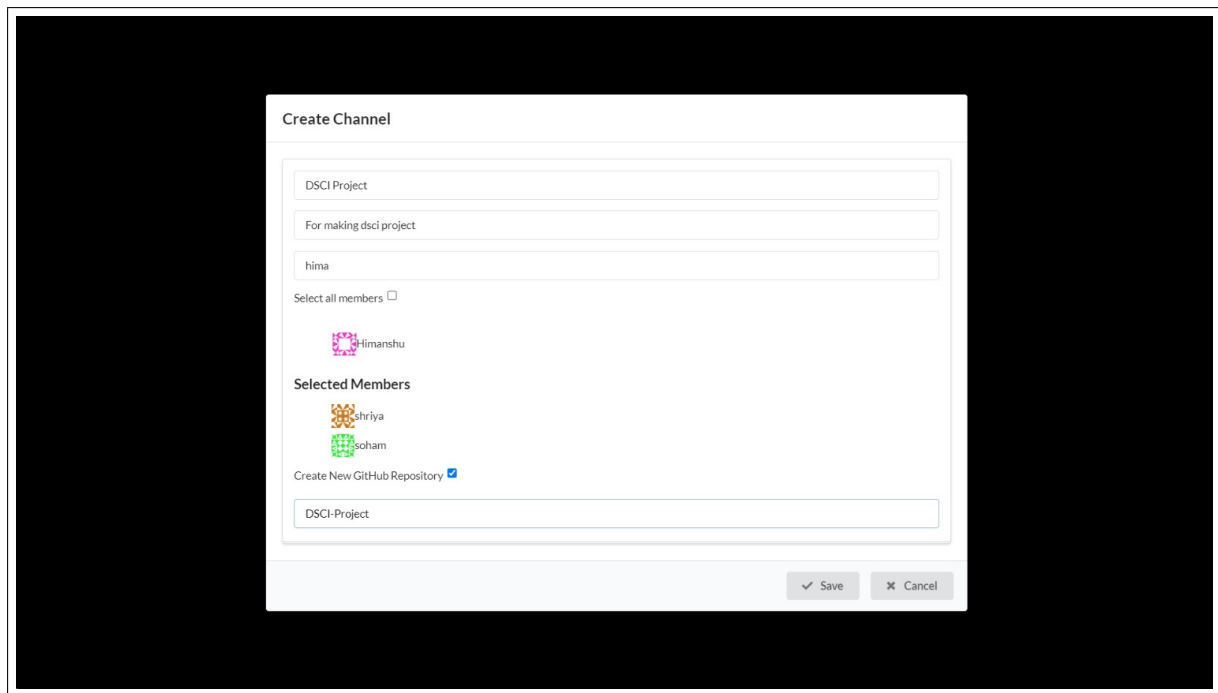


Figure 6.8: Add channel

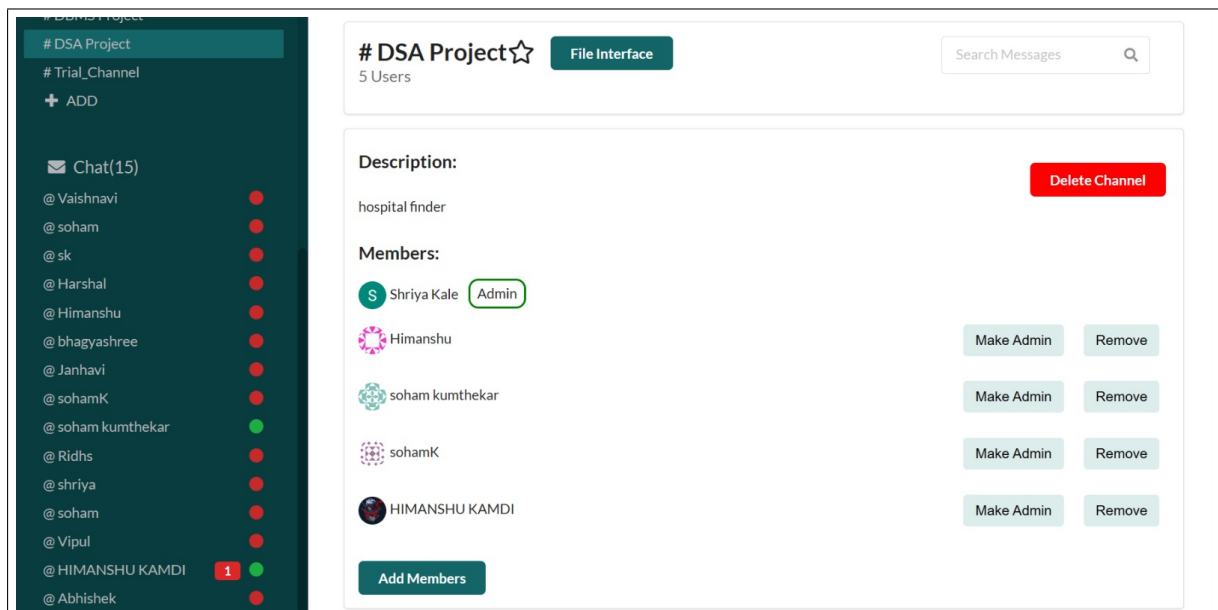


Figure 6.9: Description

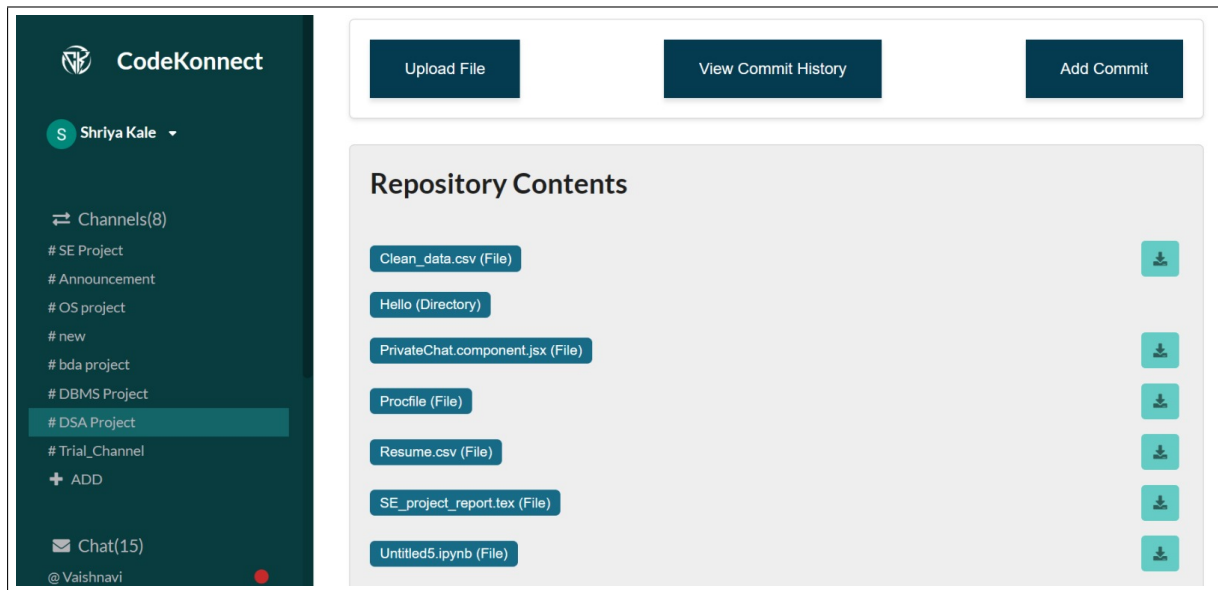


Figure 6.10: Repo contents

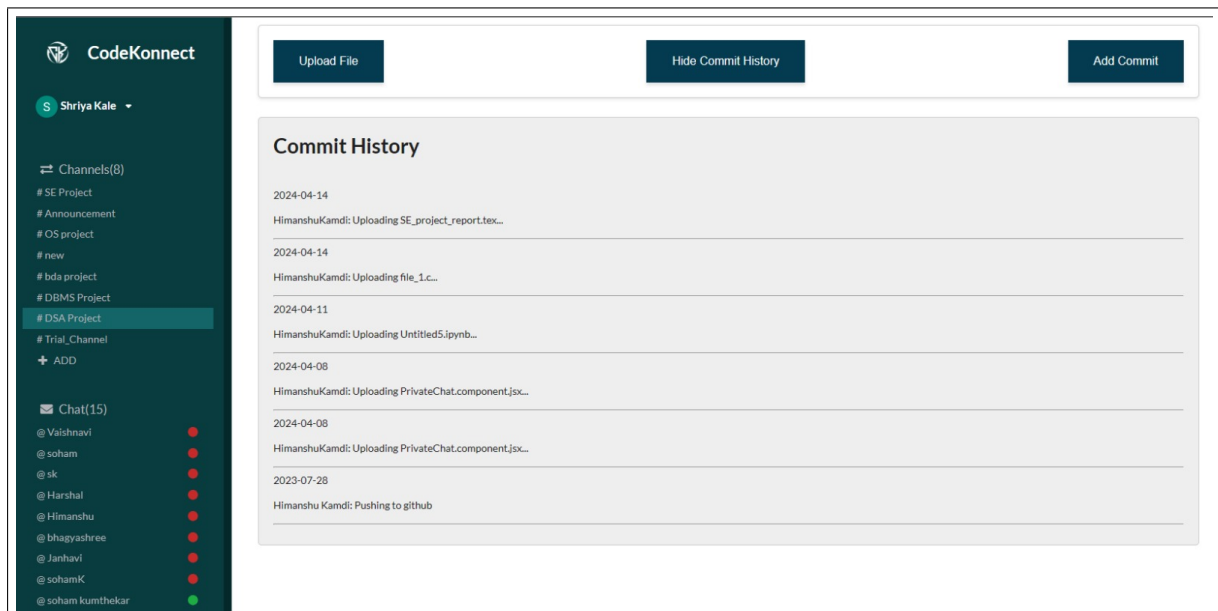
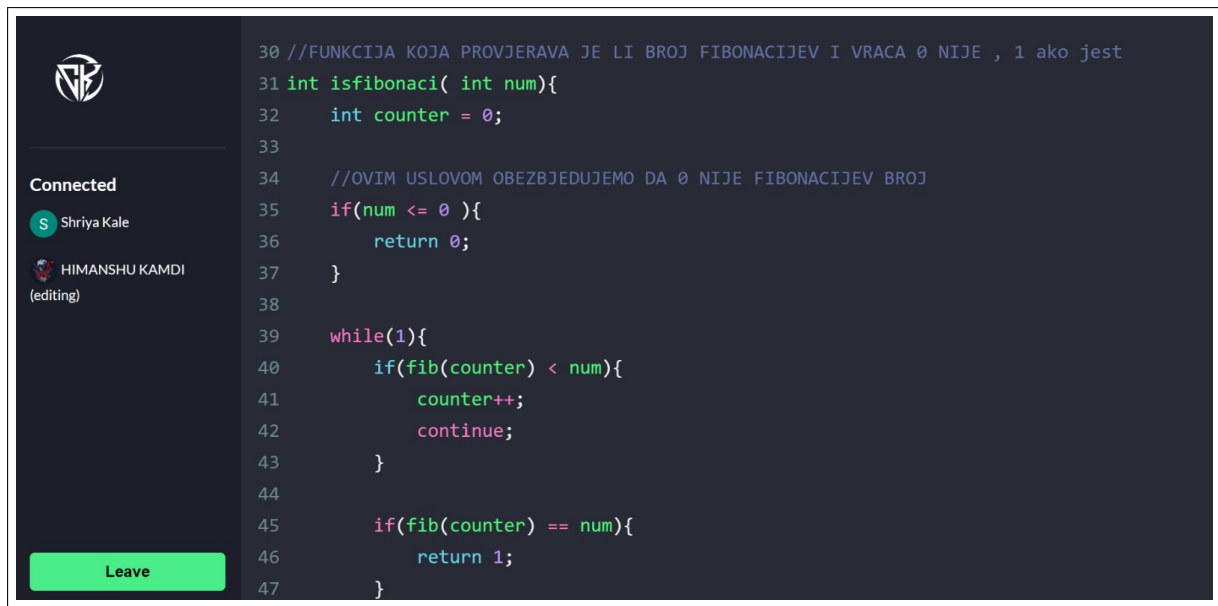


Figure 6.11: commits



The image shows a code editor interface with a dark theme. On the left side, there is a sidebar with a logo at the top, a 'Connected' section listing two users (Shriya Kale and HIMANSHU KAMDI), and a green 'Leave' button at the bottom. The main area displays C++ code for a function 'isfibonaci' that checks if a number is a Fibonacci number. The code is line-numbered from 30 to 47.

```
30 //FUNKCIJA KOJA PROVJERAVA JE LI BROJ FIBONACIJEV I VRACA 0 NIJE , 1 ako jest
31 int isfibonaci( int num){
32     int counter = 0;
33
34     //OVIM USLOVOM OBEZBJEDUJEMO DA 0 NIJE FIBONACIJEV BROJ
35     if(num <= 0 ){
36         return 0;
37     }
38
39     while(1){
40         if(fib(counter) < num){
41             counter++;
42             continue;
43         }
44
45         if(fib(counter) == num){
46             return 1;
47         }
48     }
```

Figure 6.12: code editor

Chapter 7

Summary and Conclusion

The Collaborative Platform for Coding and Messaging project aims to develop a comprehensive web application that facilitates seamless communication and file collaboration among team members. By providing features such as real-time messaging, channel-based discussions, and advanced file management capabilities, the platform empowers teams to streamline workflows, improve productivity, and foster a cohesive working environment, regardless of geographical constraints.h, budget planning, and market analysis.

Throughout the development process, careful attention is given to factors such as user experience, data security, system performance, and scalability. Responsibilities of developers include requirements analysis, system design, implementation, testing, documentation, and collaboration with stakeholders. The project estimates encompass timeframes, resource allocation, risk management, and contingency planning to ensure successful project execution.

Use cases illustrate the diverse applications of the Real-Time Collaboration Platform for teams in various industries and domains. The platform serves as a centralized hub for team communication, file sharing, project management, and collaboration on tasks

and initiatives. It enables teams to work efficiently and effectively, fostering creativity, innovation, and synergy among team members.

In conclusion, the Real-Time Collaboration Platform project aims to address the challenges faced by teams in remote work settings and distributed environments. By providing a robust and feature-rich platform for communication and collaboration, the application enhances teamwork, productivity, and success in achieving organizational goals. Through continuous iteration and refinement based on user feedback, the platform strives to remain a valuable asset for teams in navigating the complexities of modern work dynamics.