**CCGC – 5001**

**Virtualization**

# **Chat Application with flask and firebase**

Project Report



**Date – April 7, 2023**

**Submitted By:**

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Student Id** |
| 1 | Suraj Mandal | N01537188 |
| 2 | Venkata Narasimha Vedavyas Muppavarapu | N01583267 |
| 3 | Harsh Arvindbhai Tejani |  |

# INTRODUCTION

The Chat application is a web-based application developed using the React library and Firebase was used as database being flask used as backend for routing, initialize, state management. The app allows users to search for chat between rooms, personal messaging, channels. The goal of this project was to design and develop a chat app with a flexible Continuous Integration/Continuous Deployment (CI/CD) pipeline using GitHub, Kubernetes, Docker, GCP and lint for code quality.

The CI/CD pipeline was designed to automate the process of building, testing, and deploying the application to a production environment. Docker was used to create image, build and Kubernetes to containerize the app, make it easily deployable across different environments that is hosted on GCP. lint was used for code quality, providing feedback on the code's maintainability, reliability, and security.

The Firebase used as database for the app that was hosted on GCP, which provides scalable, universally available as cloud provided my google. The app is developed using React, which is a popular component-based architecture, easy to build and maintain large scale applications, each component can be developed independently.

This project report provides an overview of the chat application architecture and its different components. It also describes the CI/CD pipeline in detail, including the tools used and the various stages of the pipeline. The report also discusses the code quality aspects of the project and how lint was used to ensure high code quality.

# PROJECT OBJECTIVES

The main objectives of this project were to design and develop a chat app that provides the following functionalities:

* **Login**: The platform allows users to create their own profiles and log in using their Google credentials. Once logged in, users can access various chat rooms that are available on the platform. The platform offers users the ability to create personalized profiles with their own unique information and preferences. By using their Google login credentials, users can quickly and easily access the platform and all of its features. Within the platform, users can browse and join a variety of chat rooms, which may be organized by topics, interests, or specific communities. These chat rooms enable users to communicate and interact with others who share similar interests or have common goals.
* **Chat Rooms**: In addition to the available chat rooms on the platform, users can also create their own personalized chat rooms or channels to connect with others who share similar interests or hobbies.
* For instance, users can create private chat rooms to have one-on-one conversations with their friends or colleagues. On the other hand, channels dedicated to specific topics such as sports, music, dogs, food, general discussions, or news are also available.
* These channels provide a platform for users to share their thoughts, ideas, opinions, and experiences with others who have a similar passion or interest. By joining these channels, users can stay up-to-date with the latest news and trends in their preferred topics, as well as interact with like-minded individuals from around the world.
* **Announcements:** Announcements are an important feature on the platform as they enable administrators or moderators to communicate important information to all users. These announcements can be used to convey news, updates, changes in policies, or other relevant information. Typically, announcements are displayed prominently on the platform's homepage or within specific chat rooms or channels, making them easily accessible to all users. By using announcements, administrators or moderators can keep users informed and engaged with the platform, fostering a sense of community and promoting transparency. Users can also use announcements to share their own news, achievements, or announcements with the community, further enhancing the collaborative and interactive nature of the platform.
* **Logout:** When users are finished using the platform, it is important to log out to ensure the security of their account and information. Logging out of the platform terminates the session and removes any active connections between the user's account and the platform. This helps to prevent unauthorized access to the user's account and ensures the privacy of their personal information. Users can log out of the platform by clicking on the "logout" button, which is typically located in the settings or account menu. It is important to remember to log out of the platform after each use, especially when accessing the platform from a shared or public device, to prevent any potential security risks.

# IMPLEMENTATION AND DEVELOPMENT

The chat app was developed using the React library as frontend. Firebase was used as the database, where the application is hosted on GCP, a cloud-based service. Git and GitHub were used for version control, and Docker was used for building an image, push to docker hub and deploy to GCP. The lint is used to test syntax errors, test cases from flask and deployment of the app to production. Docker was used to containerize the app, allowing it to be deployed easily across different environments. Lint was used to ensure high code quality, providing feedback on code maintainability, reliability, and security. The app was containerize using Docker, and deployed on GCP, providing scalable, universally available, and secure hosting on Kubernetes.



FIG:1 Project Implementation How each implementation works in project:

## Web-application with React:

We utilized the React library to create the front-end of the application, including the components, templates. For the back-end, we used Flask to handle messages responses a responsive. We also implemented CSS to enhance the design and functionality of the application.

Here are some ways in which React is used:

* 1. Views: React is to create views for different parts of the chat application, such as the user list, chat history, and message composer. These views can be updated in real-time using React's state management capabilities and can be customized based on the user's preferences or settings. Additionally, views can be optimized for mobile devices or different screen sizes using React's responsive design features
  2. Templates: React can be used to create templates for various parts of the chat application such as the chat room layout, user profiles, and settings. These templates can be reused across the application and updated dynamically using React's state management capabilities.
  3. Components: React components are used to create various UI elements such as buttons, text inputs, drop-down menus, and message displays. These components can be customized and reused throughout the chat application, making it easier to maintain a consistent UI and UX.
  4. Authentication: React is used to create authentication components that enable users to log in to the chat application securely. These components are integrated with third-party authentication services such as Google to suit the application's specific requirements.

## Firebase:

We chose to use Firebase for the database. Firebase is a real-time database that is built on a NoSQL architecture, which means it is flexible and can handle a large amount of unstructured data. Our application uses firebase to create a mapping between the react code and automatically creates columns on its own. This allows us to work with the data in the database using message objects, rather than writing complex SQL queries. Firebase also handles the creation of database tables, and the mapping between the database schema and the flask code. This makes it an ideal database for chat applications, as chat data can be highly unstructured and requires real-time updates.

With Firebase, developers can easily set up a real-time database for their chat application, which can be accessed and updated in real-time from both the client and server side. Firebase also provides a wide range of tools and features, including data synchronization, authentication, cloud storage, and hosting, which can help to simplify the development process.

One of the key benefits of using Firebase as a database for a chat application is its scalability. Firebase can handle large amounts of data and traffic, making it a great choice for chat applications that may experience sudden spikes in traffic. Additionally, Firebase's real-time capabilities ensure that data is always up-to-date and can be accessed quickly, which can improve the overall user experience of the chat application.

Firebase is a powerful and flexible database solution that can help to simplify the development process and improve the scalability and real-time capabilities of a chat application.

Some of the specific ways that Firebase is used in the React and Flask chat application include:

* 1. Creating the database schema: Firebase's NoSQL database is schema less, which means developers don't have to worry about creating a rigid database schema before storing data. Instead, data is stored as JSON objects, making it easy to add, update, or remove data as needed. This flexibility can be particularly beneficial for chat applications, as chat data can be highly unstructured and require real-time updates.
  2. Storing and retrieving data: With Firebase, storing and retrieving data is simple and can be done using its real-time database. Developers can use Firebase's client-side SDKs to store and retrieve data in real-time, making it easy to update the UI as data changes. Additionally, Firebase's server-side SDKs can be used to perform complex server-side operations, such as processing data and triggering actions based on user actions.
  3. Performing queries: Firebase's real-time database allows for simple and efficient querying of data. Developers can use Firebase's query API to filter and sort data based on specific criteria, such as timestamps, user IDs, or keywords. These queries can be performed in real-time, making it easy to keep the chat application up-to-date with the latest data. Additionally, Firebase supports indexing, which can improve query performance for large datasets.

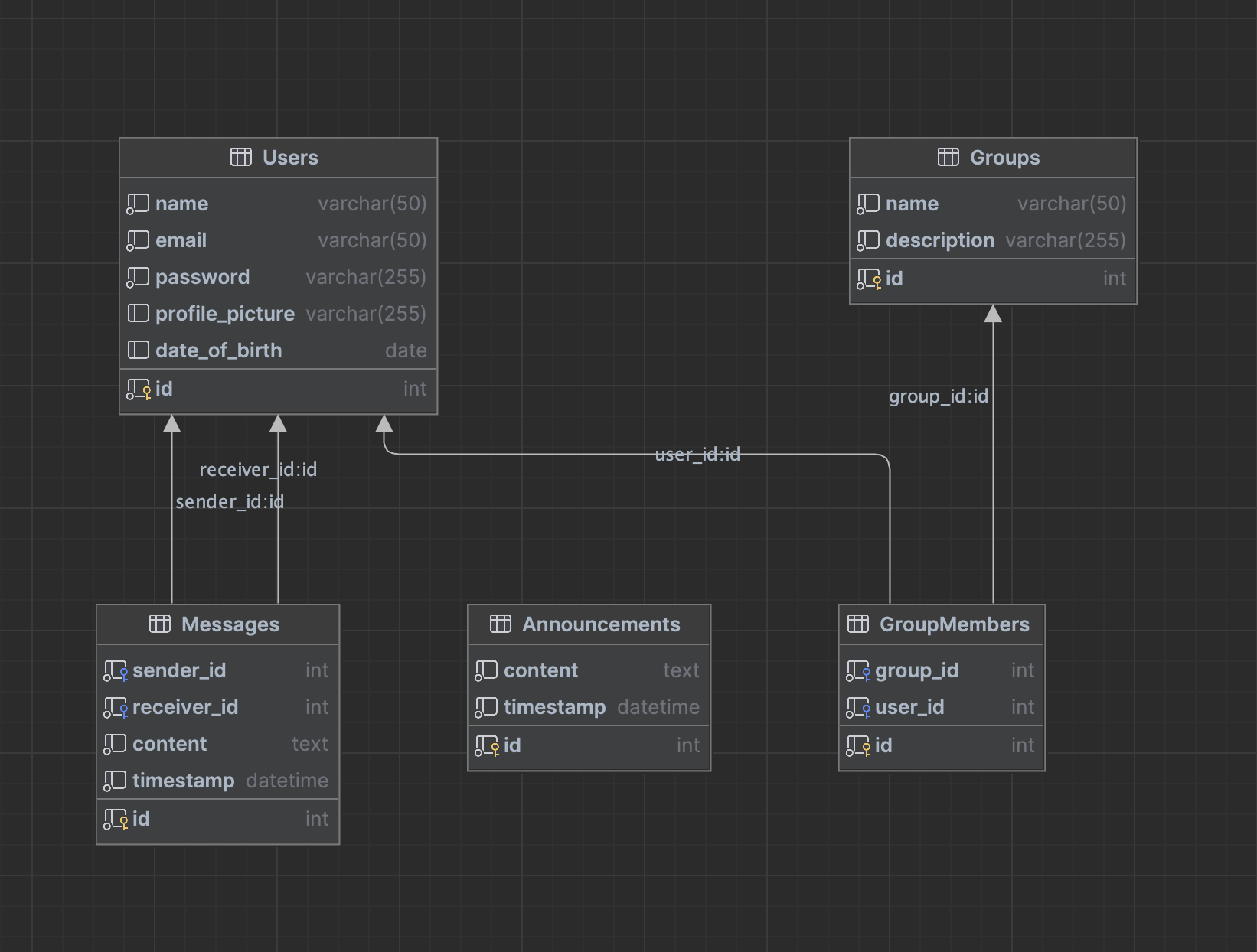


FIG:2 Database

## Git and GitHub:

Git and GitHub are essential tools in the development of the book-review app. Here's how they were used:

* 1. Version Control: Git was used for version control of the application's source code. This allowed us to keep track of changes made to the codebase over time, including who made the changes and when. It also provided the ability to roll back changes if necessary.
  2. Collaboration: GitHub was used as a remote repository for the Git repository, which allowed multiple developers to collaborate on the codebase simultaneously. We could push other group member’s changes to the remote repository and pull the changes made by others. This ensured that all of us were working with the latest version of the codebase.
  3. Continuous Integration: GitHub was also used for continuous integration (CI). A CI pipeline was set up in GitHub that automatically built and tested the codebase every time a change was pushed to the remote repository. This ensured that any issues or errors were caught early in the development process.

## Code Quality with ESLint:

For a React-based chat application, using a linting tool like ESLint can be particularly beneficial. ESLint can be configured to enforce best practices and coding standards for the React framework, such as ensuring that components follow a consistent naming convention, that props are passed correctly, and that there are no unused variables or functions.

By using a linting tool like ESLint, developers can catch potential errors early in the development process, ensuring that the codebase remains clean and maintainable. This can also help to prevent bugs and other issues that could impact the performance and user experience of the chat application. Additionally, linting can help to enforce consistency and improve code readability, making it easier for developers to understand and maintain the codebase over time.

## Docker, Kubernetes and Containerization:

In the chat app, Docker and containerization were used to package and deploy the application and its dependencies in a consistent and reproducible manner. Docker allowed us to create lightweight, portable containers that could be easily deployed to various environments, including development, testing, and production.

By containerizing the application and its dependencies, we were able to isolate them from the host system, thereby reducing the risk of conflicts and improving security. Containerization also made it easy to scale the application horizontally, by spinning up multiple containers to handle increased traffic.

As soon as the docker image is built, the image is deployed in container, as the new release come in, image is pushed to docker hub, for continuous integration, a make file has been deployed to deploy it in GCP, load balance the application with Kubernetes as the users grows by with reliability and indefinite scalability.

## GCP Deployment:

In the chat app, Docker and containerization were used to package and deploy the application and its dependencies in a consistent and reproducible manner. Docker allowed us to create lightweight, portable containers that could be easily deployed to various environments, including development, testing, and production.

By containerizing the application and its dependencies, we were able to isolate them from the host system, thereby reducing the risk of conflicts and improving security. Containerization

also made it easy to scale the application horizontally in Kubernetes, by spinning up multiple containers to handle increased traffic.

# EXECUTIVE SUMMARY

The Chat App is an excellent example of the power of modern web development tools and techniques. The application provides users with a platform to share their love of collaborating with channels, groups and engage with likeminded community in a meaningful way. The project showcases the capabilities of modern web development tools and techniques and serves as a testament to the power of collaboration and teamwork in software development.

# CHALLENGES

Some of the challenges that may have been encountered during the development and deployment of the ChatApp include:

* 1. Data management: As the application deals with user-generated data, managing and securing the data can be a complex and challenging task.
  2. Scalability: As the user base grows, the application must be able to handle the increased load and traffic. Scaling the application infrastructure to meet these demands can be a significant challenge.
  3. User experience: The application must provide a user-friendly and intuitive interface that encourages users to engage with the platform. Designing an optimal user experience can be a complex and challenging process.
  4. Performance: The application must perform optimally to provide a seamless user experience. Ensuring that the application meets performance requirements can be a significant challenge, especially when dealing with large amounts of user-generated data.

# CONCLUSION AND FUTURE WORK

In conclusion, the Chat Application is a successful project that has addressed the challenges of connecting to beloved channels, groups and community engagement. Through the use of React, Firebase, GCP, Kubernetes, Docker and ESLint we were able to create a scalable and user-friendly platform for sharing book reviews and recommendations.

In the future, there is room for additional features and enhancements to be added to the application. One potential area of improvement is the incorporation of machine learning algorithms for personalized book recommendations. Additionally, expanding the application to include social media integration and gamification elements could further increase user engagement and retention.

Overall, the Book Review Application represents a successful collaboration between various technologies and programming languages to create a functional and valuable tool for book lovers. We are proud of our accomplishments and look forward to seeing the application continue to evolve and improve in the future.

# CITATIONS/REFERENCES:

* *Amazon RDS FAQs | Cloud Relational Database | Amazon Web Services, https://aws.amazon.com/rds/faqs/.*
* *Docker Docks: How to Build, share and run applications*