Cloud App Group Report

Ash Azhagesh (aa9g22)
Cheuk Him Tai (cht1g22)
Anna Santafianou Monogiou (asm3g21)
Muhammad Izi Arman (maia1g22)
Ruichong Peng (rp10g22)
Jincheng Guo (jg16n22)
Yifei Zhu (yz44u22)

September 2, 2025

1 Description of prototype functionality

Ode is a social networking app meant to help people find music events to attend and meet people with similar interests. The prototype includes these features:

1. Register User or Organiser:
When visiting for the first time, users can register an account.
Organisers of events, or their team can also register with a new organiser or an existing organisation.

2. User and Organiser Login and Logout:
After creating an account, the User/organiser can log in to the application.
When a user or organiser wants to leave the app, they can log in.

3. Post and Attend Events:
Organisers can post events, including the information and location of the event at the home page of the application. Users can then choose which events to attend by clicking the "interested" button.

4. Map feature:
The users can see their locations as well as their friends' recent locations on an interactive map.

5. Search Engine:
The user can type a search string in the search field and it will perform searching in the user index and the organiser index, which gives different profile links to the respective profile pages.

6. Friend Request:
Users can browse other users' profiles, and then send friend requests to each other.

7. Direct Messages and Group Chats for Events/Users:
After a friend request is accepted, the two users can message each other privately.
Users can create group chats with other users if they are "friends," and organisers can make group chats for events, such that every one that is going to the event is added to the group chat.

8. Notifications:
Once a user has subscribed to an event, they will have a notification in the app regarding the newly joined event's detail, generated by open AI.

9. Create Event:
The organiser can create a new event in a form while entering the name, location, start time, end time, and price as well as uploading an image for the event that will be used for display on the main page.

10. Calendar:
Once a user has joined an event, the event details such as location, start time and end time will be automatically added to the user's google calendar.

11. Create and update Profile:
Users and Organisers can create a profile and display it on the profile page, including images, videos, events they've organised/attended and interests of genres.
We were possibly considering verification of profiles, and organisers, but subsequently dropped the idea, as it is difficult to implement.

12. Song Playlist from Spotify:
Using the Spotify API, we build a player in the profile page in which when different users access this page they can know the music taste of the profile owner.

2 List of tools and techniques used

2.1 Development Tools:

For the front end, we used React, html, Javascript and SCSS, to design and create the pages. Furthermore, react-dom was used for the routing of the web pages. React icons were used for the icons on the website.

For the backend, the language used was mainly Python, and this handled all of the API calls and functionality. The backend was powered by Firebase for database storage and retrieval of information. Google Cloud was also used as cloud infrastructure for database management and authentication.

As a reference, this UI template was used to develop some of the pages.
https://github.com/safak/youtube2022/tree/react-social-ui

2.2 Communication Tools:

We communicated through a variety of platforms, such as WhatsApp, discord, and Google Meets. Used GitHub for collaborative coding, and overleaf for writing the latex report. Outlook was used as a form of more formal communication to contact professors and ask for advice. Trello was used as a task-based management tool, to assign and monitor tasks.

2.3 Editors and Operating systems:

VSCode was mainly used by most of the team. However, Neovim and Vim were used extensively in the development of the react frontend. The front end was developed in Linux, and was designed to work seamlessly in a Linux operating system, there is also support for other operating systems.

2.4 Cloud technologies used:

We have used a variety of Cloud technologies and API in this project. Below is a summary of the used examples:

- Algolia Search as our Search engine for indexing and searching.
- Geolocation API for retrieval of the user location.
- MapBox for the interactive map.
- Cloud Translation Api for the translation of pages.
- stream-chat-react for the chat features.
- OpenAI for notification generation.
- Google Calender API for adding events to user's calender.
- Spotify API for playist in Profile.
- Firebase as our backend for storing images and using firestore for storing data.

3 Relevant statistics

3.1 Lines of Code (LOC):

The total number of lines of code in this project is about X, including the backend (such as server, database interaction, etc.) and frontend (user interface, client logic, etc.) code. The approximate number of lines of code is distributed as follows:
Backend: 1550 lines

Front end: 221429 total

3.2 Code Sources and External References:

During the development process, we referenced and used some external resources and open source codes. The following are the external codes we used and their sources:

1.Flask Framework
Source: Flask official documentation
As a Python web framework, Flask is used for backend development.

2.Firebase Admin SDK:
Source: Firebase official documentation
The Firebase Admin SDK is used in the backend to perform database operations (such as user registration, event storage, etc.).

3.React and Create React App
Source: React official documentation
The front-end uses Create React App to initialize the project.

4. Github for the UI template
https://github.com/safak/youtube2022/tree/react-social-ui
This was used as inspiration for some elements of the fornt end.

5. Documentation of Recat Dom:
https://legacy.reactjs.org/docs/react-dom.html
This was used for the routing protocol.

6. SCSS Documentation
https://sass-lang.com/documentation/
This was used for styling the webpages.

4 Brief overview of design and implementation

4.1 Design Overview:

Architecture Design:

This project is based on a front-end and back-end separation architecture. The back-end uses Flask to provide a RESTful API, and the front-end uses React, where we use (.jsx) files for the basic HTML structure and utilise SCSS for the styling, to provide a user interface.
The front-end interacts with the back-end through HTTP requests to obtain data and display it to the user.
The back-end uses Firebase to store data, including information, text and images, such as users, events, friend requests, etc.

Key Design Decisions:

1.Choosing Flask and React:

Flask is a concise and easy-to-extend backend framework suitable for rapid development of RESTful APIs.
React provides a responsive interface and enhances maintainability through component-based design.
SCSS had nesting and uses component-based support, therefore, allowing the CSS code to be easily separateable and maintainable.

The overall file strcture was designed so that the components that will be reused by the front end code was in a folder called compoenets, we had a seperate folder for assets used (images and gif). There was a seperate folder for the pages, and the main.jsx file was importing an overall app.jsx. This is the main file where all of the oruting was encapsulated and handled.

2.Using Firebase as a database and storage solution:

Firebase provides real-time database support, which simplifies backend development and is suitable for processing real-time messages, events, and other information.

4.2 Implementation Overview:

In the implementation phase, the design of the project's infrastructure and API was first completed to ensure smooth data exchange between the front and back ends.
Then the backend implements functions such as user registration, login, and activity management.
The Firebase database is used to store and read user information and event data.
Apart from the Firebase, for special features such as search engine and location with map, we have introduced the use of external API calls apart from the one we have developed.
Then afterwards, the front end pages, were made along with the dummy values for the rendered posts, events, stories, profile pictures and various other assets. Then by utlising iterative development, we integrated the backfront and frontend step by step, and throughly tested the application at every stage to ensure quality.

5 Critical evaluation of the prototype submitted

5.1 The Advantage of Prototype

Backend:Functions were implemented in a very simple yet effective way. Because of this we were able to complete the backend quickly and connecting it to the frontend was easy.

Modular design: Clear modular design, which makes the code easier to maintain and expand.

Real-time data: By integrating Firebase, the application supports real-time data updates, which makes it particularly suitable for dynamic environments such as event management.

Scalability: Using Firebase ensures that the application can scale well.

Frontend:

Styling: We experimented with multiple styles and colour schemes. At the end we decided to use a simple but classy style.

React framework: The front end is built with React, ensuring the updatability and maintainability of the UI.

5.2 Need to be Improved

User authentication: The specific implementation of user authentication (such as using Firebase Authentication or a custom authentication system) is unclear and user data may be compromised.

The events posts, can be further improved by adding a like, comment and share functionality. Furthermore we could have better privacy setting for the map, such as showing a general location rather than the exact location of the user.

Functions such as blocking the user and reporting the user, while partially implemented in the backend, did not end up being implemented due to time constraints.

There are some parts of the scss styling that are not maintainable due to the constants being imported into the jsx files that handle the html structure, this was originally due to poor design implementation of the backend, this makes the overall application less maintainable and readable.

5.3 Conclusion

Overall, the prototype is well designed, with clear backend and frontend structures and good scalability. It effectively utilizes modern technology stacks such as React and Firebase. However, there is still some area for improvement.

The lack of familiarity with good software engineering principals, affected the overall maintainability of the codebase, and so in the future this codebase should be refactored to streaminline the encapsulation and reduce data duplication.

Furthermore theere is a lack of dynamic updating for certain elements of the front end which backend did not support.

6 List of contributions

Ash Azhagesh (aa9g22)

Did some of the front end pages, styling and routing, using React and Scss, the routing is handled by React-Dom. Did the first few pages of the initial draft for the report, and led the elevator-pitch presentation. Organised some team meetings and presentations throughout the project, planning the timeline for various aspects of the project that need to be completed, and ensured a certain level of quality was maintained.

Cheuk Tai (cht1g22)

Worked mostly on the front end and the connection to the backend and varies APIs, I worked on the design of register, login and the display of users' dynamic contents in the front-end, while creating an update profile page and its functionalities, create events for organisers and mapping users by friend request. Additionally, worked on developing a search engine using Algolia Search and a map displaying user location using Geolocation for precise positioning as well as Mapbox for the interactive map, and also making the final presentation structure.

Anna Santafianou Monogiou (asm3g21)

Handled friend requests and various event related functions in the backend, added admin logic to the project and helped with the implementation of the frontend. Worked on pages 1, 2 and 6 of the report, organised meetings and prepared the presentation for the elevator pitch.

Muhammad Izi Arman (maia1g22)

For the backend, i created files for user login and logout, organisor login and logout, create user profile, delete user, add audio to chat and translator. For the frontend, i help with the profile update handling, implement webpage translator from the backend and use spotify api for the user profile. prepared the presentation for the elevator pitch.

Ruichong Peng (rp10g22)

Completed most of the first iteration of the backend, and used firebase to accomplish this. Handled registration and editing of users and organizers. Modified notification generation and event related functions in order to link the notification page with the backend. And implemented Google Calendar in the backend.

Jincheng Guo (jg16n22)

Completed the development of the entire real-time chat module, including private chat, group chat, sending and receiving files, etc. Handled the backend for the Open A.I. notifiactions and worked on of pages 3 to 5 for the report.

Yifei Zhu (yz44u22)

Worked pages 3 to 5 for the report. Handled the user sign up functions and various backend functions. Summarised results from everyone to produce a PowerPoint for subsequent presentations.