COMP3207 Cloud Application Development

Team L

https://linex-374511.nw.r.appspot.com/

Team Members:
Alexandre Pires | 31733905 | abp1g20@soton.ac.uk
Alex Tonev | 30984394 | amt1u19@soton.ac.uk
Cristian-Sebastian Csete | 31870163 | csc1u20@soton.ac.uk
Denitsa Radeva | 31693849 | dr5g20@soton.ac.uk
Ivan Entchev | 32083416 | ie1g20@soton.ac.uk
Kristian Ivanov | 32121474 | kii1u20@soton.ac.uk
Lyubomira Kaynakchieva | 32288654| lvk1u20@soton.ac.uk

1. Description of prototype functionality

1.1 Problem

Coordinating schedules with groups of people can be a challenging and time-consuming task. There are often many conflicting schedules and commitments to consider, and it can be difficult to find a time that works for everyone. Additionally, as the group size increases, it becomes more challenging to communicate and keep track of everyone's availability. This can lead to confusion and frustration for all parties involved and ultimately hinder the ability to make adequate plans and get things done.

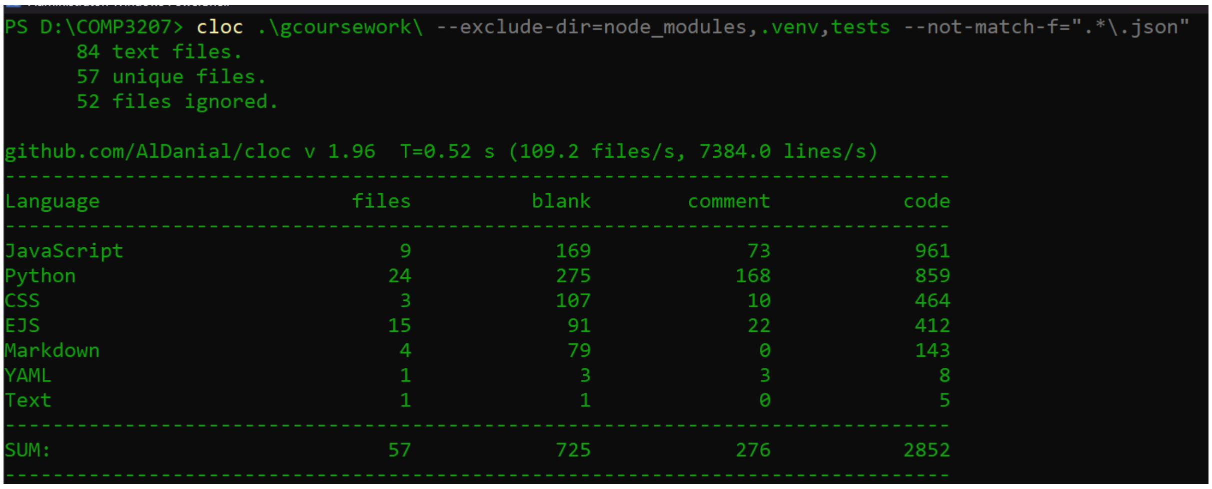
1.2 Solution

Our app is designed to alleviate these challenges by providing an easy-to-use platform for coordinating schedules with groups of people. By allowing users to share their personal calendar and see how it aligns with others, our app makes it easier to find mutually convenient times for events and meetings. Additionally, by providing tools for communication, such as friend lists and event invites, our app makes it easier to keep track of everyone's availability and coordinate plans. With these features, our app aims to streamline the scheduling process and reduce the confusion and frustration often associated with coordinating schedules with large groups of people.

2. List of tools and techniques used

Tool/Technique	Justification
Azure (CosmosDB and Functions)	We used Azure to build and deploy our database.
Python 3.8	We used Python to build our backend and create a connection to the Azure database.
JavaScript	We used javascript to build client and server-side.
Express.js	We used Express.js to handle the GET and POST requests.
Socket.IO	We used socket.io because of its low latency and ability to implement bidirectional connections. It allowed to dynamically change different events without the need for refresh.
EJS	We used ejs instead of HTML because we wanted to have the dynamic functionality of ejs.
VIS-Timeline	We used vis-timeline to achieve our primary goal of displaying time as a timeline instead of the standard calendar view.
Vue	We used Vue tool to keep everything in sync, given the different events our application needs to handle.
Balsamiq	Balsamiq is a prototype-building tool. We used it to build a prototype of our application so everyone could have a better overview of how the frontend would look like.
Figma	Figma is another prototype-building tool. We used it to create a modern-looking design.
SotonGit	We used the university version control system for collaboration.
Agile & Scrum	We used agile principles while working on the project. We worked as pairs during three sprints. Additionally, we created a scrum section in our discord server where we would inform the others what we have done, what we are working on and what we need help with.
Trello	We used Trello as our product management tool. We created a board and classified all tasks according to the MoSCoW method. We used labels to show who was working on which task.

3. Relevant statistics



```
PS D:\COMP3207> cloc .\gcoursework\ --exclude-dir=node_modules,.venv,tests --not-match-f=".*\.json"
      84 text files.
      57 unique files.
      52 files ignored.

github.com/AlDanial/cloc v 1.96  T=0.52 s (109.2 files/s, 7384.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JavaScript                       9            169             73            961
Python                          24            275            168            859
CSS                              3            107             10            464
EJS                             15             91             22            412
Markdown                         4             79              0            143
YAML                             1              3              3              8
Text                             1              1              0              5
-------------------------------------------------------------------------------
SUM:                            57            725            276           2852
-------------------------------------------------------------------------------
```

4. Brief overview of design and implementation

4.1 Backend

We have used Python 3.8 as our primary programming language to develop our backend system. To implement the database aspect of our system, we used Azure CosmosDB, and we utilised Azure Functions to handle the read and write operations within the database. This combination of technologies allows us to efficiently and effectively store and retrieve data while ensuring scalability and reliability.
Our backend consists of three tables - users, events, and personal calendar. The users have a list of active events, which we store by their unique id. They have a corresponding accepted/pending status. Similarly, we store a list of friends for an event with a friend/pending status.

4.2 Frontend

The frontend of our app is built using EJS and Javascript. EJS is responsible for the visual implementation of the interface, while Javascript handles the communication between the server and the client. We decided to use cookies, which play an essential part in that. They are used to keep track of the state between different pages and allow users to stay logged in for 2 hours before needing to log in again. We use JavaScript and routing through Express.js to handle POST and GET requests on the server side. Different requests are sent with information about the user's personal calendar, friends list, current and pending friends, and a list of events (both accepted and pending). To handle the communication between the server and the client, we use Socket.IO. We use EJS instead of just HTML for the interface to have dynamically generated parts such as friend lists and events lists. Vue handles the dynamic nature of events, such as users adding events to their calendar or accepting/declining events. To achieve our goal of displaying time in a line view instead of a calendar view, we use a vis-timeline, which provides the functionality.

5. Critical evaluation of the prototype

5.1 Strengths

Our app aims to strike a balance between providing helpful information for scheduling events and protecting personal privacy. While the app displays schedules for all participants to help coordinate event timing, it does not disclose specific details about what a person is busy with. Instead, it simply indicates that the user is busy. Additionally, participants are given the flexibility to suggest options outside the initial time frame specified by the event creator, to accommodate busy schedules or preferred timing. Additionally, the app implements cookies to ensure that even if the user accidentally closes the app, they will not need to log in again, and their session will be saved.

5.2 Limitations

The system has several limitations, including when a user receives an invite, everyone in the group can see that user's timeline before they accept the invite. This can potentially compromise privacy. Additionally, when an event remains without any users, it does not get deleted, leading to clutter and confusion. The system also has inadequate security measures for cookies, leaving user data vulnerable to attack. Moreover, when a final time is decided for an event, the decided time could be presented in a more user-friendly form. Finally, some features need to update dynamically for a better user experience.

5.3 Future work

In the feature, the most crucial goal would be to improve the security measures. Also, improving user experience by making everything update dynamically. Other extensions that will attract more people would be to integrate the functionality of syncing their personal calendar in the app with another calendar, such as Google Calendar or Apple Calendar. Adding a profile picture would also be a great feature so users can personalise their profile more. Additionally, adding a group photo album would be a great feature so everyone can upload pictures from when they attended an event instead of people having to send others pictures separately. Finally, adding a suggestions feature would give suggestions for events so people would have ideas of what to do in cases where they want to see each other but do not know what to do.

6. List of contributions

- Alexandre Pires
  - Backend: friends functionality
  - Frontend: personal Calendar functionality and UI, personal events display, add personal event functionality, styling contributions
- Alex Tonev
  - Presentation of our idea
  - Front-End: styling of pages with forms on them, research for the timeline to use
- Cristian-Sebastian Csete
  - Product management: setting up version control
  - Backend: friends update status functionality
  - Frontend: friends functionality, dynamic update of events and friends components, routing of the pages, general refactoring, styling of pages (login & register)
- Denitsa Radeva
  - Product management: setting up the repository
  - Backend: Register, GetCalendars, getEvent, getting the active user events, all suggestion, and all votes functions + their testing
  - Frontend: Events functionality and UI, Event timeline view functionality and UI, Event suggestions functionality and UI, Event votes functionality and UI, minor contributions to the overall css
- Ivan Entchev
  - Product management: Balsamiq prototype, Figma prototype(for styling), class diagram, use cases
  - Frontend: designing of overall views, styling of pages (home & profile)
  - Work on report
- Kristian Ivanov
  - Product Management: setting up Trello, scrum master
  - Backend: Setting up Azure containers, All event functions(creating event, accepting/rejecting/quitting event, updating user's active events, adding an event to the personal calendar), login function
  - Frontend: Create events, Save final event, Google cloud deployment
- Lyubomira Kaynakchieva
  - Frontend: Sending requests to the Azure functions, Login functionality, Register functionality
  - Work on the report