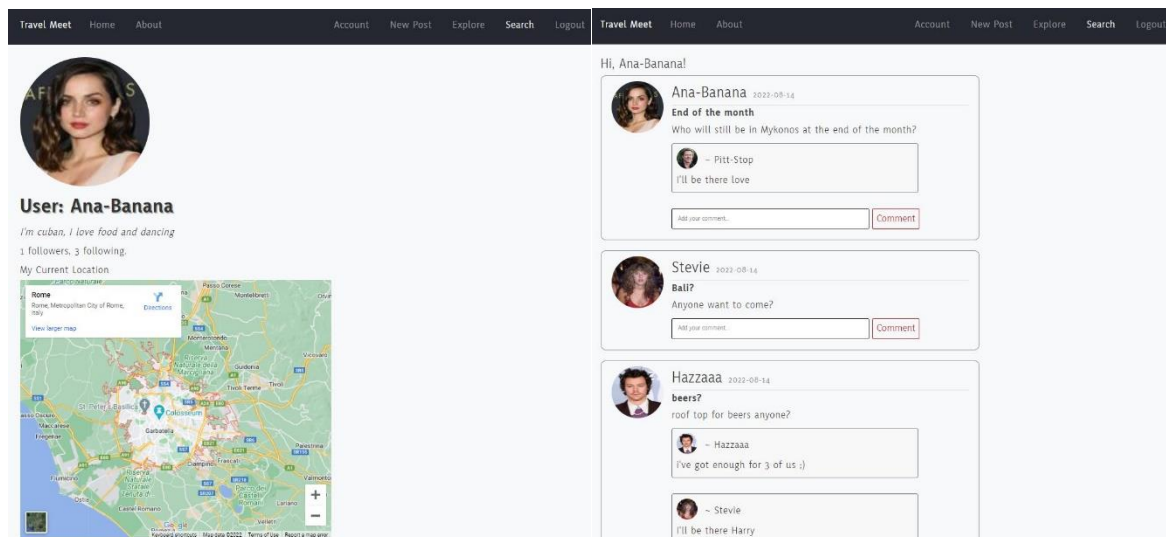


## INTRODUCTION:

Upon starting our project we discussed as a group the different struggles we come across in our daily lives that technology can alleviate. There were lots of big, daunting ideas floating around but we decided to find something fun we all had in common and that was travel. This was the point at which our ideas had turned towards creating a social media website. Travel and social media are a match made in heaven – we use it to show off where we are, where we are going, who we are with and what we are doing.

We decided to create a travel-centric application that allows you to follow like minded travellers, travellers that are in the same area as you are or have already been to the area you are destined for. We wanted this site to show you posts from people you follow, to create new posts, comment on posts, update posts, delete posts.



We wanted our users to be able to amend their profile with new pictures, the location shown as a pin on a google map and a little bio about themselves. There would also be the option to view all posts of everyone on the site and be able to pick and choose who you follow based on their location or post content. Another objective would be to filter the posts shown based on a search term that would return any matches e.g. breakfast options.

Outside of the ideas an aim for the group was to ensure we each participated and were comfortable with the work being done. We wanted to make sure we were writing clean code, following best practices and amending accordingly. We wanted to ensure we were testing the code and reviewing it in at least pairs to ensure what was being committed to the repository was necessary.

Finally we wanted to make sure that we were having fun, that it was collaborative and we were able to learn something new from each other.

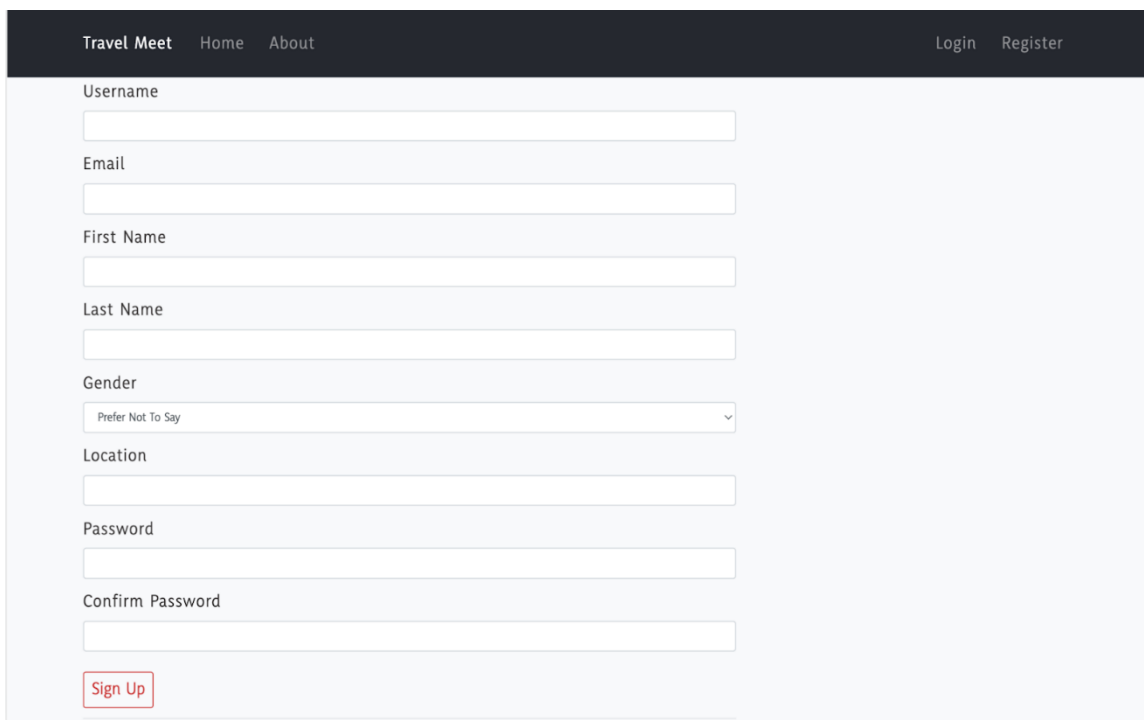
## BACKGROUND

This project is called Travel Meet - a social media app for those interested in travelling.

We are trying to help solo travellers find other people interested in exploring a new area and doing activities together. All of us love to travel and travel independently, and we found that things were much more expensive when attempting to book for a single person (for example, a taxi). Moreover, there are many 'secrets' that other travellers know better and could suggest to others through their blogs.

We thought it would be great if we could find other like-minded people who would also want to share in those experiences.

Firstly, the user is able to register and fill in 7 fields: username, email, first name and last name, gender, current location, and password. The location field is connected to Google Maps API and displays the input location on the map on the user profile page. Once the user is registered, they can also log in next time to use the web application or log out to terminate the connection.

A screenshot of a web application's registration page. At the top, there is a dark navigation bar with the text 'Travel Meet' on the left, and 'Home' and 'About' in the center. On the right side of the bar are the links 'Login' and 'Register'. Below the navigation bar is a light blue registration form. The form contains the following fields: 'Username' (text input), 'Email' (text input), 'First Name' (text input), 'Last Name' (text input), 'Gender' (a dropdown menu currently showing 'Prefer Not To Say'), 'Location' (text input), 'Password' (text input), and 'Confirm Password' (text input). At the bottom of the form is a red 'Sign Up' button.

This app will allow users to post their location and travel plans, which other users can search and comment to connect. It is also possible to search by the post title. For example, if you are curious about pizza in Naples, you could search for such posts and get the best pizza suggestions! If you enjoyed reading the posts by a specific user, you can follow them and drop a comment. If you are still looking for your favourite content creators - you can visit the Explore page and have a look at posts by all users in the network. Once you have selected your favourite bloggers, you can read and interact only with them in your feed.

You can also write your own posts, modify or delete them, as well as edit your profile page. There are features to add a profile picture, update your location and write a bio about yourself.

[Travel Meet](#) [Home](#) [About](#) [Login](#) [Register](#)

Log In

Username

Password

☐ Remember Me

Sign In

Forgot Password?

Need An Account? Sign Up Now

CATEGORIES

Day trips

Road trips

Weekend trips

Pet-friendly trips

Nature

Sightseeing

USEFUL LINKS

Home

Account

About

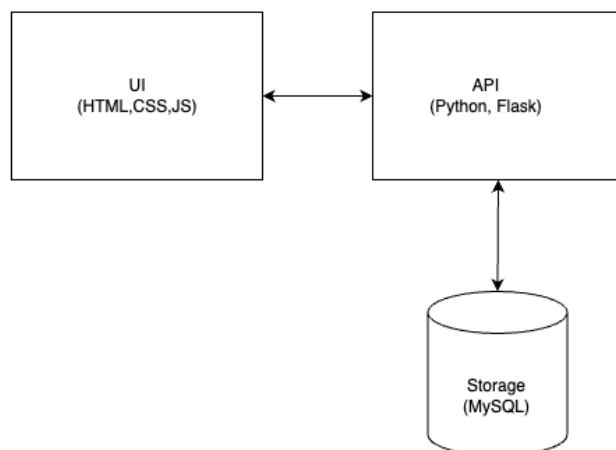
COMPANY

Email us

F.A.Q

Careers

## SPECIFICATIONS AND DESIGN



We have used three layered architecture, frontend, backend and database. We have used HTML, CSS and Python for frontend, Python and Flask for backend and MySQL as database.

### Requirements technical and non-technical

1. MySQL: database system to store users, blog posts, comments, and followers.

The schema 'travel\_meet' is created to store four tables: comment (stores comments), followers (relationship table for linking followed users with followers), post (stores published blog posts), user (stores registered users).

Table user includes such columns: id, username, email, first\_name, last\_name, gender, location, image\_file (used to display profile picture), about\_me (profile bio), password (stored hashed).

Table post: id (post id), title, content, timestamp (date posted), user\_id (who posted it).

Table followers: follower\_id (which user id follows), followed\_id (which user id is being followed).

Table comment: id (comment id), date\_posted, content, user\_id (which user commented), post\_id (under which post the comment was left).

2. Python: used extra libraries

3. Flask: micro web framework in Python, that provides third-party extensions.

Extensions used:

- flask\_sqlalchemy: allows interacting with the database through Python, without manually creating SQL tables
- pymysql: connects Python to SQL database
- flask\_bcrypt: used for password hashing
- flask\_login: handles user authentication for the web application
- Pillow: processes images through which it is possible to display profile pictures, upload, delete them and save them in the database
- flask\_wtf and jinja: used for easier templating in routes.py file and HTML files; allowed avoiding duplicate code
- email\_validator: validates email addresses
- flask-unittest: extension for unit testing the application

4. Frontend: HTML, CSS, Bootstrap, jQuery

Non-technical:

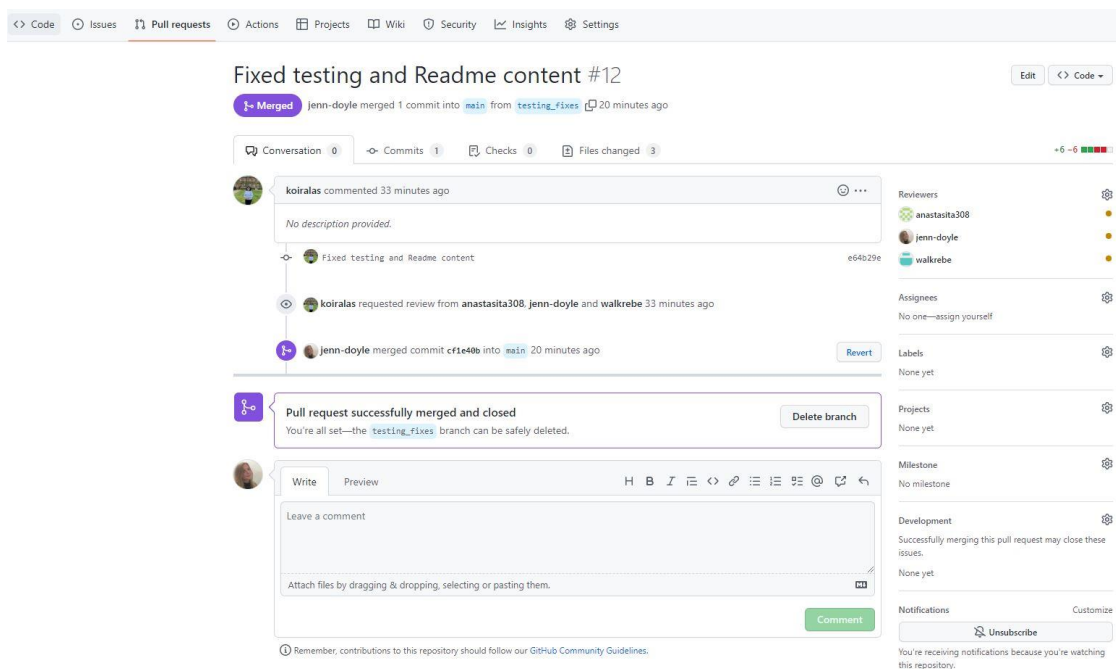
1. Figma: wireframing and web design
2. Trello: planning activities and tasks
3. Communication: Zoom meetings and chat on Slack
4. Canva: logo design

## IMPLEMENTATION AND EXECUTION

Name	Usage
SQLAlchemy	SQLAlchemy is an Object relational mapper (ORM) which allows us to manage databases as classes objects and methods instead of tables and SQL. This essentially translates the operations into database commands and represents objects as rows in the table, with their attributes as columns – useful for OOP.
PILL	Imaging library for python. Image class allows us to open and save images to the database

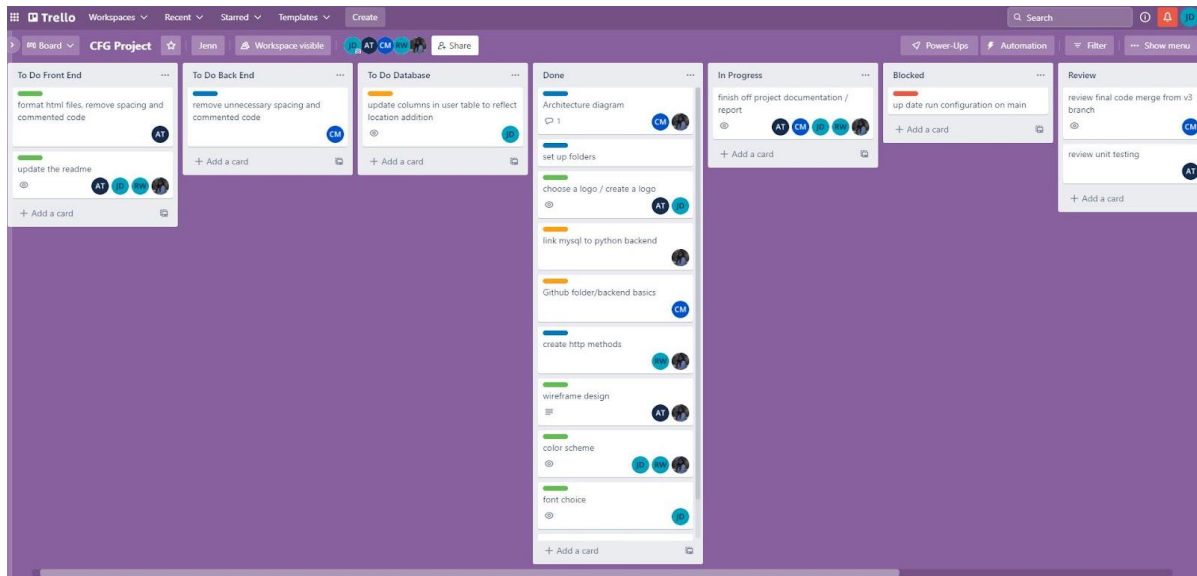
Flask	Micro web framework in Python, that provides third-party extensions. Extensions described in requirements sections
MySQL	Relational database management system used to store application data
WTForms	Flexible forms validation and rendering library for Python web development. Supports data validation, CSRF protection. Used to validate data

When it came to this project there was a very collaborative approach, we each shared ideas on things we wanted to change or what we wanted to add to elevate the idea. We kept the lines of communication very open so if anybody wanted to challenge something that had been done they could and we followed a majority rules process to ensure things were kept fair. We made sure to review the code in pairs or more to make sure the code was good enough to be pushed to main and was a feature, function, styling decision that we were all happy with.



We came across a few of the usual issues when it came to implementing changes to the code and making sure we were all able to access the project. There were the usual issues with connecting to the repository, creating branches, switching branches and making sure we always did a Git Pull before adding new code and trying to Git Push. There was a very simple problem which cropped up that prevented the code from running at all for one user and that was cloning the code into a pre-made project folder, we all learnt from that mistake! There were also issues encountered with PyCharm and making sure any dependencies were installed correctly and the files were configured to run on localhost without issue.

This was able to be resolved by keeping track of our installation in the requirements.txt file.



We created a Trello board to track the progression of the project and assigned tasks to ourselves that we felt would show off our strengths. We used slack to communicate daily on our progress and any issues we faced. This was incredibly useful as most of the group are in full time employment and unable to work on anything during the day whereas others were. We weren't able to have daily stand ups or retros as it wasn't applicable to this situation but we made sure we kept in constant contact. We also tried to have regular zoom sessions and take advantage of the time given in the sessions to catch each other up on where we were at. Zoom was great to allow us to pair programme an issue, refactor the code or to all come together and mob code a problem. We also used zoom when reviewing the code to merge our branches on GitHub.

We tried to keep the Trello board as up to date as possible by moving tickets into the correct columns whether we were currently working on something, completed it or were blocked by something else. We updated the tickets with who was or had been working on it so we knew who to talk to if there was an issue with any of the code.

## TESTING AND EVALUATION

We have done some basic unit testing, using the unittest framework. We have used unittest framework. The areas we have covered with testing are, registration, login, unauthorised access, creation of new posts. All of these tests are done at the API level.

The API testing cover would ensure the proper function at the API levels, however the automated test that covers user interacting with the application is missing. This will be covered following the manual testing.

The system has following limitations:

- It does not support languages other than English (text in navigation bar, buttons, form labels etc.)
- It is missing robust automation testing.
- The performance of the application might be impacted by the growing number of users. This can be overcome by creating and using proper indexes.

## CONCLUSION

The goal of the project was to apply our classroom learnings that we have acquired in the last 13 weeks. Encourage students to collaborate with one another, work in a team and present some deliverables on certain time, which we have tried and managed to do.