Stack Underflow

SENG2021 - REQUIREMENTS AND DESIGN WORKSHOP 21T1

DELIVERABLE 4 & 5

Dimitri Tsardakas	z5259123
Reinier de Leon	z5257456
Jamie Clarisse Domingo	z5205571
Jiaqi Zhu	z5261703
Atharv Damle	z5232949

WorldPlay

Abstract

This document is our final report for SENG2021, following the completion of our project. It has been structured in a way that is easy to comprehend our project scope and our specifications. Firstly, in the “Project Scope + Spec” we go over the scope of our project and our motivation. We also introduce our problem statements, user stories and features that we set out to achieve. Next, we explain the Design and Architecture of the project before starting to code it. Here you can find our Interface/Software Designs and the Software Architecture. Following is the Implementation of our project, breaking down how we split up the work, how we coded, what changes we made and anything else related to completing the project. Lastly, we have included a summary of the project as a whole and our own reflection.

Table of Contents

Scope and Specification

Project Scope

WorldPlay directly focuses on providing users with a place to share personal experiences through music. The main feature of our application is an interactive world map where a user will be able to see tags on multiple locations. These tags are posts people have shared from all around the world. Each tag contains an attached spotify song, an image and a personal message written by each user.

Within our application, a user can also explore different cultures and find new music from all around the world, as each song posted will provide its meaning and interpretation alongside. People are redundant to music types from other cultures that they are not interested in, our application makes it easier for users to find understanding within these new genres.

Motivation

Ever since the pandemic made travel almost impossible (also people who never had the ability to travel) people across the world felt lonely locked up in their houses. As a result, people who love travelling and other people in general felt disconnected from society and the world.

In addition, the importance of music and its true meaning to an individual is often overlooked in modern web applications such as soundcloud, spotify, facebook. Music brings out emotions and is often intertwined with past experiences such as travel, culture, history, and enjoyment.

Our platform is purely focused on providing users with a place to share music and explain its meaning to them from all around the world. Being dedicated to sharing these important personal thoughts, our users will get the chance to be heard without judgement that is often found in more popular websites.

Problem Statements

For our project, we directly focused on these problem statements, basing our features and scope around them.

1. There is no platform solely dedicated to users sharing stories and personal experiences through music.
2. Individuals often find it hard to find new types of music from around the world.
3. Similar social media platforms are visually messy, cluttered, and distracting to the users.
4. There is no platform presenting users with a direct visualisation of the real locations of the posts through a map.
5. People are losing their connection to their own culture as they are not exposed to their traditional music enough.
6. People who lack understanding of a particular culture often find it difficult to connect with their particular genre of music.

User Stories

For convenience, we have attached a link to our Sequence Diagrams of these User stories.

When reading through our user stories, some things to note are that 1 Story Point (or SP) is equal to 1 hour of work and the priorities denote the order in which user stories will be completed, with lower value meaning higher priority.

Requirement 1. Authentication:

As a user, I want to have an authenticated account so that I can access all the features on the site.

Feature 1.1 Sign up for a new account

Story 1.1.1:

As a User
I want to Sign up to the platform
So that I can access all the features on the platform

Priority: 2
Story Points: 6

Scenario: User tries to sign up to the platform

Given: I am at the sign in portal
And: I am not logged in
When: I enter my newly created username and password
And: Click on the “Sign Up” button
Then: I should be signed up to the platform
And: Should be redirected to my new personal dashboard

Feature 1.2 Log in and log out my account

Story 1.2.1

As a User
I want to Be able to login with a user id and password
So That I can access content saved and personalized for me

Priority: 2
Story Points: 4

Scenario: User tries to sign into the platform

Given: I am at the Login portal
When: I enter the correct user id and password
And: Click the “Login” button

Then: I should be logged in to the platform
And: Redirected to my personalized dashboard

Story 1.2.2

As a User
I want to Be prevented from logging in with the incorrect credentials
So That I can make sure that my account cannot be accessed without my user id and password.

Priority: 2
Story Points: 1

Scenario: User fails to log in to the platform

Given: I am at the login portal
When: I enter the incorrect user id and password combination
And: Click the "Login" button
Then: An error pops up
And: does not allow me to log in.

Feature 1.3 Reset my password if I forget my password

Story 1.3.1

As a User
I want to be able to reset my password
So that I don't lose my data if I forget my password

Priority: 5
Story Points: 6

Scenario: User resets their password

GIVEN: I am on the sign in page
AND: I am not logged in
WHEN: I click on the "Forgot Password" button
THEN: I will be sent an email which I can then use to securely change my password

Feature 1.4 Link to my Spotify account

Story 1.4.1

As a User
I want to link my Spotify account to my account
So that I can access my Spotify playlists seamlessly

Priority: 3

Story Points: 5

Scenario: User links their account to the app

GIVEN: I am on the sign up page for the platform
WHEN: I click 'Next' button
THEN: I am redirected to the "Link to Spotify Account" page
WHEN: I click "Link to Spotify" button
THEN: A spotify OAUTH page will pop up
WHEN: I log in to my spotify account on the page
THEN: My spotify account would be linked to the platform

Requirement 2. Account Management & Preferences:
As a user, I can use settings so that I can change my profile details and other settings based on my habit.

Feature 2.1 Change my password
Story 2.1.1
As a User
I want to change my password
So that I can keep my account secure

Priority: 5
Story Points: 3

Scenario: User changes their password to "hunter2"

GIVEN: When I am on the Profile page
WHEN: I enter my new password (hunter2) in the password box
AND: I click on the "Change Password" button
THEN: My password will be updated to "hunter2"

Feature 2.2 Change my username
Story 2.2.1
As a User
I want to change my username
So that I can customize my appearance to other people on the platform

Priority: 6
Story Points: 3

Scenario: User changes their username to "AzureDiamond"

GIVEN: I am on the profile page
AND: Under the “My Profile” tab
WHEN: I enter my new username (AzureDiamond) in the username box
AND: I click “Change Username” button
THEN: My username will be updated to “AzureDiamond”

Feature 2.3 View notifications (comments, likes, titles)

Story 2.3.1

As a User
I want to be able to check notifications
So that I can be up-to-date on any comments, likes, or titles that I receive

Priority: 6
Story Points: 4

Scenario: User views their notifications

GIVEN: I am on the Profile page
WHEN: I click “Notifications” tab
THEN: I will see all the notifications in a list

Feature 2.4 View all my tags

Story 2.4.1

As a User
I want to be able to check all my tags
So that I review what I have shared in the past

Priority: 4
Story Points: 4

Scenario: User tries to view all his tags

GIVEN: I am on the profile page
WHEN: I click ‘My tags’ tab
THEN: I will see all my tags in a list

Feature 2.5 Set the website to light/dark mode

Story 2.5.1

As a User
I want to be able to change the view mode
So that I can switch to the mode that is suitable for the lightness of my environment

Priority: 6
Story Points: 8

Scenario: User tries to change from light mode to dark mode

GIVEN: I am on the settings page
WHEN: I see light mode is on
AND: I click on the switch
THEN: The mode will be changed to dark mode

Feature 2.6 Change default setting for showing the side menu

Story 2.6.1

As a User
I want to be able to change whether the side menu is displayed on default
So that I can always have a clear view on the map on the homepage

Priority: 6
Story Points: 6

Scenario: User tries to change the setting for side menu

GIVEN: I am on the settings page
WHEN: The side menu is set to be displayed on default
AND: I click on the switch
THEN: The setting will be turned off

Feature 2.7 Change the display language

Story 2.7.1

As a User
I want to be able to change the display language to my home language
So that I can better under the website and the posts

Priority: 6
Story Points: 6

Scenario: User tries to change the display language

GIVEN: I am on the settings page
WHEN: I select a language that I want to use
THEN: The website will be displayed in the language I selected

Requirement 3. Home page and dashboard:

As a user, I want to have a clear and visually satisfying homepage so that I can see every button clearly.

Feature 3.1 Check the about page to learn about the website

Story 3.1.1

As a User
I want to check the about page of the website
So that I can get more information about the motivation and team behind the website

Priority: 4
Story Points: 2

Scenario: User looks at what the website is about

GIVEN: I am on the home page and not signed in
WHEN: I click on ‘More info’ button
THEN: I will be redirected to the about page

Feature 3.2 Explore tags by generating random tags

Story 3.2.1

As a User
I want to generate random tags
So that I can explore random posts from all over the world

Priority: 6
Story Points: 6

Scenario: User tries to generate random tags

GIVEN: I am on the home page
WHEN: I click on ‘Generate random tag’ button
THEN: I will be redirected to a page where a list of random post is shown

Feature 3.3 Explore tags by panning over the world map

Story 3.3.1

As a User
I want to be able to pan around the map
So that I can explore different parts of the world and view people’s tags

Priority: 2
Story Points: 8

Scenario: User tries to view a tag on the map

GIVEN: I am on the home page
WHEN: I click somewhere on the map
AND: hold onto the map
THEN: I will be able to move my view around on the map
WHEN: I click a tag on the map
THEN: A new window will pop out showing the post I clicked on

Feature 3.4 Hide the side menu for better view

Story 3.4.1

As a User
I want to hide the side menu (toggle list) to the side of the website
So that I can have a better view on the map

Priority: 4
Story Points: 4

Scenario: User tries to hide the side menu

GIVEN: I am on the home page
WHEN: I click on the 'Hide' button of the side menu
THEN: The side menu will be hidden to the side

Feature 3.5 Get support from the help page

Story 3.5.1

As a User
I want to check the help page
So that it can help me navigate the site in case I need support

Priority: 6
Story Points: 4

Scenario: User accesses the help page for support with sign up

Given: I am on the sign-up page
WHEN: I have trouble with entering details
AND: I click on "Help" button in the navigation bar
THEN: I will be redirected to the help page

Requirement 4. **Post creation:**

As a user, I want to be able to create a post so that I can share music and my thoughts with other people

Feature 4.1: Add a message to a new tag

Story 4.1.1

As a user
I want to be able to add a message to a new post I am creating
So that I can explain some experience or thought I have

Priority: 4
Story Points: 4

Scenario: User clicks on new post

GIVEN: I click “create new post” in the homepage
WHEN: I insert a message in the box and submit
THEN: The post will contain the message I just added

Feature 4.2: Attach a Spotify link to a new tag

Story 4.2.1

As a user
I want to be able to attach a Spotify song to my post
So that I can share it with others

Priority: 4
Story Points: 4

Scenario: Creating a new post

GIVEN: I click “create new post” in the homepage
WHEN: I click on “Add song”
THEN: I will be redirected to Spotify where I can pick the song I want to include in my post

Feature 4.3: Attach an image to a new tag

Story 4.3.1

As a user
I want to be able to attach an image to my post
So that I can share a particular memory or experience

Priority: 4

Story Points: 4

Scenario: Creating a new post

GIVEN: I click “create new post” in the homepage
WHEN: I click on “upload image” and choose an image from my computer
THEN: The image will be loaded to my post

Feature 4.4 Discard my tag

Story 4.4.1

As a User
I want to Be able to discard my tag
So that When I am not satisfied with my tag I do not have to keep it

Priority: 2
Story Points: 2

Scenario: User wants to discard a tag when making it

Given: I am on the tag page
WHEN: I click on ‘discard’ button
THEN: The tag will be discarded
AND: The window will be closed

Feature 4.5: Preview my tag

Story 4.5.1

As a user
I want to be able to preview my post
So that I can check if I have made mistake in my post

Priority: 4
Story Points: 4

Scenario: User previews his post

GIVEN: I have drafted my post
WHEN: I click on ‘Next’ button
THEN: I will be redirected to the preview page of my post

Feature 4.6 Publish my tag

Story 4.6.1

As a user
I want to be able to publish my post
So that I can share my post to people around the world on the platform

Priority: 4
Story Points: 4

Scenario: User publishes his post

GIVEN: I have previewed my post
WHEN: I click on ‘Publish’ button
THEN: I will be redirected to the page of my published post
WHEN: I click on ‘Back’ button
THEN: I will be redirected to the home page

Requirement 5. Interaction:
As a user, I want to be able to interact with posts so that I can engage with other people around the world.

Feature 5.1: Like a post
Story 5.1.1
As a user
I want to be able to ‘Like’ a post
So that I can engage with other people’s posts

Priority: 4
Story Points: 4

Scenario: User clicks the Like button when viewing a post

GIVEN: I am viewing a tag that I like
WHEN: I click on “Like” button
THEN: The tag will have my like added to it

Feature 5.2: Comment on a post
Story 5.2.1
As a user
I want to be able to comment on a post
So that I can interact with other people and share my thoughts

Priority: 3
Story Points: 6

Scenario: User comments “Nice” on a tag

GIVEN: I am viewing a tag that I like
WHEN: I click on the comment box

THEN: It gets highlighted
WHEN: I type in “Nice”
AND: Click “Comment”
THEN: My comment gets posted under the tag, with the content “Nice”

Feature 5.3: Play song attached to post

Story 5.3.1

As a user
I want to play the music attached in a tag
So that I can enjoy new music people have shared and learn more about it

Priority: 3
Story Points: 6

Scenario: User listens to the song “Lonely” in the tag

GIVEN: I am viewing a tag with the song “Lonely” attached
WHEN: I click on the “Play” button
THEN: The song “Lonely” starts playing
AND: The play button is replaced by a progress bar for the song

Note: *Song “Lonely” by Lauv feat. Anne Marie*

Requirement 6. Searching for posts:

As a user I want to be able to search for posts with filters I chose, so that I can find things that really interest me.

Feature 6.1: Filter for certain posts to appear on the map

Story 6.1.1

As a user
I want to filter out the tags that are presented on map
So that I can find the post that I’m interested in

Priority: 4
Story Points: 4

Scenario: Searching for specific posts

GIVEN: I am on the homepage
WHEN: I zoom into a specific country
THEN: I filter for a specific genre of music, culture or time of creation

AND: I click confirm
THEN: I will see music and posts based on what I searched for

Feature 6.2: Search for tags and music

Story 6.2.1

As a user
I want to be able to search for tags based on music genre, language, date etc
So that I can find explore music and tags based on these filters

Priority: 4
Story Points: 4

Scenario: Searching for tags and music

GIVEN: I am on the search page
WHEN: I select the filters I want
THEN: I will be shown posts related to my search only

Project Plan

Before starting to code our project, we spent a bit of time organizing our team. We chose Github to be our main repository where we would keep our code as well as all of our deliverables. We chose Notion to be our planner as we were all familiar with its simply to use design. Discord would be our main place where we discuss between each other, have meetings / standups and share anything project related. We also utilised a Google Drive to store any minutes and documents. More on this in the Implementation section.

Monetisation

As our application gets more popular and the amount of users increases, we will eventually consider ways of monetising our application. There are a couple of ways we thought we can monetise. These include:

- In our world map, we can show festivals or concerts around the world in addition to our tags. These festivals/concerts will be clickable and the user will be able to see information about that event and click on a link to purchase tickets. This affiliation with such events will provide us some form of monetisation
- Artists sponsorships is another way of monetising our application. Music artists will be able to become part of our sponsorship program, where we will advertise their posts to our users.
- In addition, we can advertise different locations, events or products that are related to music through our application. These however, must align with the scope of our application which is sharing experience through music.

Scalability

Due to the time constraints and many other factors, some of the pre-planned features were not fully implemented in the final product. For resetting a user's password, we did not attempt to send a password-reset link to the user's email as this may require a ghost email account to send the emails. The way we have implemented for password reset is that the user can directly reset their password inside their profile, which is much more convenient and faster. What we can improve on this feature is to add a confirmation step, hash all the passwords, and hide the passwords at the frontend at all times.

For the backend, our web application uses SQLite. If higher level of security is required in the future, we can upgrade the database from SQLite to PostgreSQL which helps handling more complex functionalities.

Interface Design

When designing our interface, we first developed low-fidelity prototypes to illustrate the specifications and requirements we had decided upon. After these were approved by the team and finalised, we went on to develop high-fidelity prototypes to visualise how our application will look when finished. This considerably sped up development once we started coding, as we could refer to these images as we designed the frontend of the application.

Low-Fidelity Prototype

The Low-Fidelity Prototype document can be found within the Github page.

In order to ensure we had a strong foundation for the direction our application was going in for future sprints, we created multiple low-fidelity diagrams to explore and elaborate upon our ideas.

Inside the Low-Fidelity Prototype document, there are three diagrams:

1. Prototypes: The final low-fidelity prototypes depicting each screen on a separate page.
2. Lofi Storyboard Diagrams: A flowchart of the screens and the UI components that lead the user to different areas of the application.
3. Lofi Component Interaction Graph: A flowchart of the different components and how they are connected.

We used the following table below to help visualise which user stories are being met in each screen of the low-fidelity prototype. This helped us to track which user stories were completed and which stories required further attention, keeping us on top of the workload and improving our productivity.

#	Feature	Depicted in Prototype Screen	Page
1.1	Register for a new account	Register	4
1.2	Log in and log out my account	Log In	3
1.3	Reset my password when I forget my password	Help Page	22

1.4	Link to my Spotify account	Register (spotify)	5-6
2.1	Change my password	Profile Settings - My Profile	23
2.2	Change my username	Profile Settings - My Profile	23
2.3	View notifications (comments, likes, titles)	Profile Settings - Notifications	25
2.4	View all my tags	Profile Settings - View My Tags	25
2.5	Set the website to light/dark mode	General Settings	27
2.6	Change default setting for showing the side menu	General Settings	27
2.7	Change the display language	General Settings	27
3.1	Check About page to learn about the website	About Page	21
3.2	Explore tags by generating random tag	Generate Random Tag	2
3.3	Explore tags by panning over the world map	Homepage(signed in)	7
3.4	Hide the side menu for better view	Homepage(signed in)	7
3.5	Seek help from the help page	Help Page	22
3.5	Search tags by filtering by music genre	Homepage(signed in)	7
4.1	Add a message to a new tag	Add a New Tag	17
4.2	Attach a Spotify link to a new tag	Add New Tag (Select song from Spotify)	15-16
4.3	Attach image to a new tag	Add a New Tag	18
4.4	Discard my tag	Add a New Tag	17
4.5	Preview my tag	Add a New Tag (Review)	19
4.6	Publish my tag	Add a New Tag (Published)	20
5.1	Like a post	Add a New Tag (Published)	20
5.2	Comment on a post	Add a New Tag (Published)	20

5.3	Play song attached to post	Add a New Tag (Published)	20
6.1	Filter for certain posts to appear on the map	Homepage(signed in)	7
6.2	Search for tags and music	Homepage(signed in)	7

After conducting some rough sketches using a digital graphics application, Procreate, we used Balsamiq Wireframes to create our low-fidelity sketches. Using this tool gave us the simplicity and efficiency of paper prototyping with the added bonus of reusing, transforming and moving components and real-time team collaboration. The tool was extremely useful and considerably increased the productivity of our team.

We focused especially on usability to ensure our application was clear and easy to navigate. Usability considerations we made include:

- Taking a minimalistic approach for each screen and not over-populating it with unnecessary information. We gave the user the opportunity to hide the side menu if they no longer needed it, and our initial homepage contains only vital information such as a link to the 'More Information' screen, Sign Up and Log In options.
- Separating tasks into multiple steps to avoid overwhelming the user and help facilitate a productive workflow. Instead of a single page where users fill in all their details to register, we separated this into smaller, atomic steps.
- Keeping all similar operations in a single menu. We placed viewing and searching tags in the side menu on the right and added links to important pages visible in the fixed top navigation bar to reduce confusion and searching.

We were quite satisfied with the layout of our interface, however, before creating our high-fidelity prototypes, a few things we added to our task list was to add in our official logo, establish a colour scheme and make our components appear more accurate to the final product.

High-Fidelity Prototype

Within the Github page, the High-Fidelity Prototype files can be found in the links below.

High Fidelity Pages Link:
https://github.com/AD9000/Stack_Underflow/blob/main/Deliverables/Deliverable%201/HighFidelityPages.pdf
High Fidelity Chart Link:
https://github.com/AD9000/Stack_Underflow/blob/main/Deliverables/Deliverable%201/HighFidelityChart.pdf
In case the pdf is not clear, the link to the Figma directly is:
https://www.figma.com/file/IKdCm051jJTcLyzq1aAX9R/High-Fidelity-Chart

The documents:

1. High Fidelity Pages: The final high-fidelity prototypes depicting each screen on a separate page.
2. High Fidelity Chart: provides information in relation to the connections of each screen and how they interact with each other.

To help visualise which user stories are being met in each screen of the high-fidelity prototype, please refer to the table below.

#	Feature	Depicted in Prototype Screen	Page
1.1	Register for a new account	Register	4
1.2	Log in and log out my account	Log In	3
1.3	Reset my password when I forget my password	Help Page	18
1.4	Link to my Spotify account	Register (spotify)	5
2.1	Change my password	Profile Settings - My Profile	23
2.2	Change my username	Profile Settings - My Profile	23
2.3	View notifications (comments, likes, titles)	Profile Settings - Notifications	20
2.4	View all my tags	Profile Settings - View My Tags	21
2.5	Set the website to light/dark mode	General Settings	22
2.6	Change default setting for showing the side menu	General Settings	22
2.7	Change the display language	General Settings	22
3.1	Check About page to learn about the website	About Page	17
3.2	Explore tags by generating random tag	Generate Random Tag	2
3.3	Explore tags by panning over the world map	Homepage (popped out side menu)	6
3.4	Hide the side menu for better view	Homepage	9
3.5	Seek help from the help page	Help Page	18

3.5	Search tags by filtering by music genre	Homepage(signed in)	7
4.1	Add a message to a new tag	Add a New Tag	13
4.2	Attach a Spotify link to a new tag	Add New Tag (Select song from Spotify)	15
4.3	Attach image to a new tag	Add a New Tag	13
4.4	Discard my tag	Add a New Tag	13
4.5	Preview my tag	Add a New Tag (Review)	14
4.6	Publish my tag	Add a New Tag (Review)	14
5.1	Like a post	Add a New Tag (Published)	16
5.2	Comment on a post	Add a New Tag (Published)	16
5.3	Play song attached to post	Add a New Tag (Published)	16
6.1	Filter for certain posts to appear on the map	Homepage (popped out side menu)	6
6.2	Search for tags and music	Homepage (popped out side menu)	6

The application we used to create our high-fidelity prototypes is Figma, a collaborative interface design tool. Due to our familiarity with this tool and the high quality of our low-fidelity prototypes, we were able to complete the high-fidelity prototypes swiftly.

Since we focused on usability for our low-fidelity prototype, we decided to pay attention to visual and aesthetic qualities during our next iteration.
Design considerations we made include:

- Introducing a simple and visually-pleasing colour scheme. We used a mixture of black, white and blue shades with the occasional red colours to liven up our website. When incorporating these into our prototypes, we made sure we used them consistently to produce a harmonious interface.
- Using real photos instead of sketches. We included an image of a globe to represent the ‘Travel’ theme of our website on the home page, and a real map to represent the map API we will use.
- Instead of creating separate pages for our register and log in operations, we used pop up modals that open on our home page instead. This reduced routing complexity for our website and assisted functionality by letting the user know they can return to the home page at any time by clicking outside of the modal.

We were very happy with our high-fidelity prototype overall, apart from the use of 'Lorem Ipsum' text in the About and Help pages of our website. This was done due to time restraints, however, slowed down development in future iterations as we had to write up this text from scratch.

Final Product Interface

Within the Github page, screenshots of the Final Product can be found in the link:
https://github.com/AD9000/Stack_Underflow/blob/main/Deliverables/Deliverable%205/FinalProductInterface.pdf

During the development of our application's frontend, we referred to the high-fidelity prototypes heavily to ensure consistency, since different team members coded different screens. The prototypes proved very helpful, and through the use of React, Material UI and Leaflet, we were able to produce a finished product that was very close to the high-fidelity prototype.

Some changes that were made to the interface during development include:

- When we implemented the Spotify API, the Spotify account authorisation changed from a screen within our application to an external screen for the user to log in to their Spotify account. This change occurred naturally and did not impact the project considerably. Once the user was verified, they were provided with a successful registration page to take them back to our website.
- We wrote all the About and Help information and incorporated these into their respective pages.
- Instead of using a horizontal tab to display the user's profile settings, we found it was easier to implement separate routes for each page: 'My Tags', and 'My Profile'. We also removed the 'Notifications' page as we were not able to finish this due to time restraints.
- We removed the draggable pin button that was used to create tags and added a 'Create A Tag' button to the navigation bar instead. We decided that the draggable pin was difficult to see and had poor affordance, only vaguely implying to the user what it's purpose was. Instead, a bold 'Create A Tag' button is much easier to see and provides more instruction to the user, reducing confusion.
- Changes were made to the 'Create A Tag' workflow. When the user clicks on the 'Create A Tag' button and then clicks on the map, a pin is placed in that location, which then opens a modal for creating a new tag. This was moved from the side menu to improve the user's efficiency and prevent the user from losing their location in the side menu. To further facilitate efficiency, we included a single page to enter in all the details instead of separating this into multiple steps. This saves the user from clicking through the modal and thus saves precious time.

Overall, our team was very content with the user interface design of our application. In the beginning of the project we wanted to create a sleek yet engaging website, and we were able to achieve that throughout the duration of the term.

Final Software Architecture

We took careful consideration when deciding on the software architecture of our application and the stack we used to achieve our requirements and implement our prototypes.

Software Architecture Diagram
To help visualise the software architecture of our application, we have attached a diagram to this link:
https://github.com/AD9000/Stack_Underflow/blob/2a369cdc28ab51731460f8c4676f0c5df45bc6f3/Deliverables/Deliverable%202/architecture_diagram.svg
However, if any issues loading the images are encountered, kindly take a look at:
https://drive.google.com/file/d/1Ne0_6oBHXMm2xVSRbQJIYiKuwC-4gzKr/view?usp=sharing

We used the questions provided by the SENG2021 Project Deliverable 2 Specification as a guide to describe and elaborate upon our software architecture choices. Using these questions allowed for a detailed and thorough explanation of our chosen external data sources, software components, languages, libraries and machine requirements, as well as justifications for why we have chosen them. However, since initially writing these, we made a few changes to the architecture during the implementation phase and thus have incorporated them below.

External Data Sources:

The external data sources our application accesses are the following application programming interfaces (API): Spotify API and Mapbox API.

Spotify API

The Spotify API is a free web API that provides access to JSON metadata about music, artists, albums and a user's Spotify data. It also provides a JavaScript library that allows developers to create a music player and play Spotify audio tracks straight from the browser.

Mapbox API

The Mapbox API is a multi-functional web service API that provides four distinct services: Maps, Navigation, Search, and Accounts.

Software Components:

Our application consists of the following software components: Browser, Frontend, Backend Database, and Web Hosting.

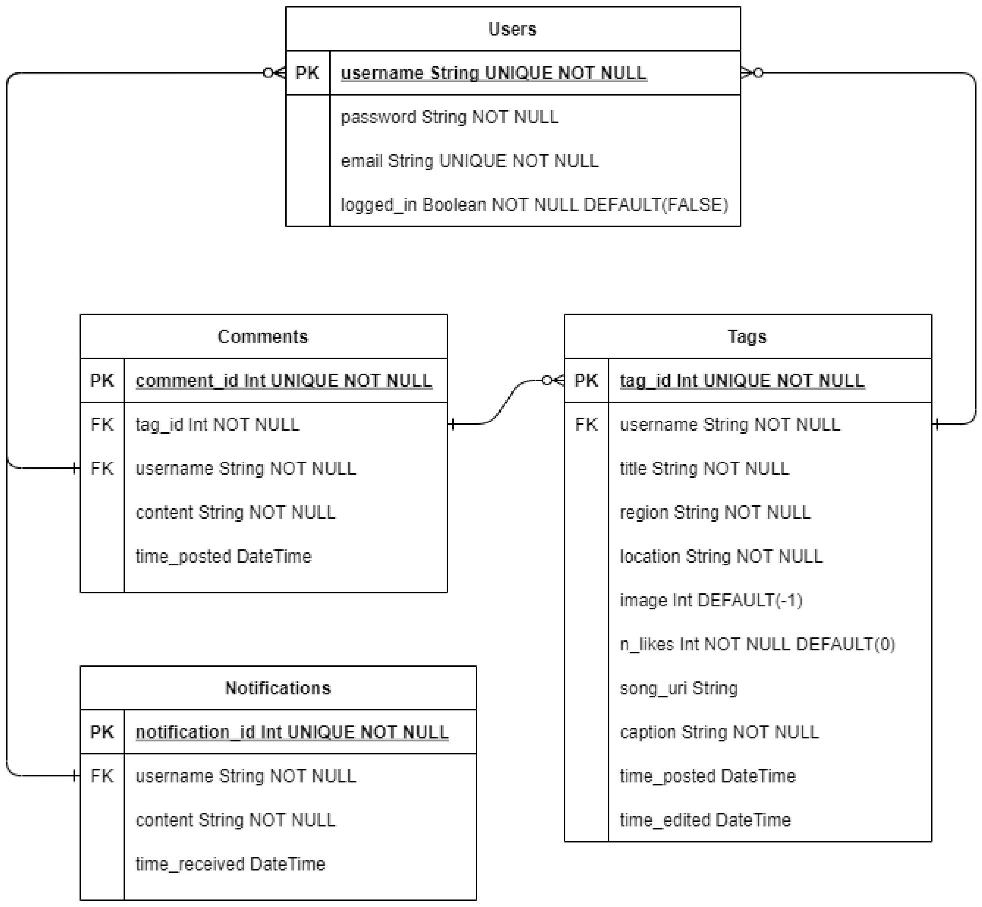
Browser

Users interact with the app using their browser of choice (Safari, Firefox, Google Chrome). Since the frontend (discussed later) is packaged as a web application, the UI is rendered in the user's browser, from where they can access the different features provided by our application.

Frontend
The frontend consists of the elements and technologies used to render the user interface in their browser. This is the visual component of the application, and is developed with user accessibility and aesthetics in mind for the best user experience possible.

Backend
The backend acts as a bridge between the frontend that renders the information and the data sources (the APIs) that provide that information. The backend itself is modelled as a RESTful API, and the frontend can access the features by simply making an appropriate request.

Database
The database is used to store the data that powers the website. This includes information about tags, likes, and data important for maintaining user accounts. This allows the user to customize their experience on the website and lets them access their data whenever they log into their account. Below is an entity relationship diagram depicting how our database is set out.



Web Hosting
This is the place (or the hardware) where the application itself is hosted. It is the place where the frontend code is served to the user and where the backend runs and accesses the required information from the database.
Note**:** The deployment for the frontend and the backend is done on *separate* servers to reduce coupling between them.

Language & Library Choices

Frontend

- *Typescript (https://www.typescriptlang.org/)*
  Typescript can be described as a superset of JavaScript that extends its functionality by providing static type checking.

- *React (https://reactjs.org/)*
  React is a JavaScript library that is primarily used for building UI components and developing user interfaces. In our application, we used React to implement reusable and professional front end components.

- *Material UI (https://material-ui.com/)*
  This simple and customizable component library was used together with React to develop an accessible and aesthetically pleasing interface for our users.

- *Leaflet (https://leafletjs.com/)*
  Leaflet is an open-source JavaScript library for creating interactive maps. This was used in our application to pin locations on the map.

Backend

- *Python3 (https://www.python.org/)*
  Python3 is an interpreted, high-level and general-purpose programming language that everyone in our team is very experienced with. It was used to implement the majority of our backend.

- *FastAPI (https://fastapi.tiangolo.com/)*
  FastAPI is a fast, high-performance web framework for building API's with Python 3.6+. It enabled us to use a REST interface to implement the main building blocks of our application.

Database

- *SQL*
  SQL (Structured Query Language) allowed us to access and manipulate the data in our database.

- *SQLite (https://docs.python.org/3/library/sqlite3.html)*
  SQLite is a fast, light relational database management system.

- *SQLAlchemy (https://www.sqlalchemy.org/)*
  SQLAlchemy is an object-relational mapping tool that allowed us to easily access and interact with our database as objects rather than relational database tables, abstracting away the SQL code itself, and making it more maintainable.

Web Hosting

- *Docker (https://www.docker.com)*
  Docker is an open source containerization platform. Docker enables developers to package applications into containers—standardized executable components that combine application source code with all the operating system (OS) libraries and dependencies required to run the code in any environment.

- *Google Cloud (https://cloud.google.com/)*
  Google cloud is a cloud computing platform, which in simple words, means it provides the hardware for the application to run on, but with zero maintenance costs. We use the compute engine in particular, which is then used to run the Docker containers.

Platform Choices

Client Platform

Our application supports Windows, Mac and Linux operating systems. Though the latest version of Google Chrome was the primary web browser we tested and demonstrated our application on, we will also provide support for the latest version of Safari, Mozilla Firefox, and Microsoft Edge 13.

Website Hosting

All the components of the website will run on a lightweight Linux virtual machine. The applications themselves will be containerized using Docker which means there are no fixed requirements for the processing power and RAM, and can be scaled up or down easily using a container management system such as Kubernetes if needed. We will use Google Cloud to set up and deploy these containers as it makes the process of deployment much easier.

Key Benefits and Achievements of Architectural Choices

External Data Sources

We chose to use the Spotify API as Spotify is one of the most used music streaming services in the world, thus allowing us to reach a multitude of users who already own a Spotify account. Furthermore, the Spotify API is free to use and it has detailed, comprehensive documentation with many easily-accessible tutorials, thus making it easy to learn and implement.

Successfully implementing this data source in our application allowed us to provide our users with access to Spotify's vast range of audio content from songs, audio books to podcasts once they have registered with their Spotify account. When creating tags, users can search for their favourite songs and add them to their posts. The Spotify API's playback functionality also allows the user to play a small snippet of the song inside their web browser.

We chose the Mapbox API as it provides all the functionality our application requires, providing services such as: Maps, Navigation, Search, and Accounts. We implemented Mapbox's Maps Service in our application to render map tiles on the browser. Mapbox's extensive functionality gave us the option to customise the maps for our purposes. We were able to implement a simple, colourful map in our interface with country names written in English. We were also able to give our users the opportunity to pan around the map and zoom in and out as they please. Thus, we were able to achieve multiple user stories including allowing users to view the posts of others, and visualise their respective locations.
Other similar API's were considered, such as the Google Maps API and OpenStreetMaps API, however, we found Mapbox to be the cheapest to use, with readable and easy to understand documentation. Due to the limited time restraints of this project, we needed to prioritise efficiency, and thus employing a simple, manageable API such as Mapbox helped us achieve more in a short amount of time.

Frontend

When deciding on our web stack for the frontend, both Typescript and JavaScript were initially considered. However, although we are more familiar with JavaScript, we eventually settled on Typescript since it is an overall ideal choice for large software projects due to its advanced type system and extra features. Implementing our system in Typescript improved the readability and maintainability of our code, helped us to debug our code efficiently, and minimised errors that may be caused due to type ambiguity during compile time. Furthermore, due to it's many similarities to JavaScript, it was easy to pick up for those in the team who have not used it before.

We chose to use the React JavaScript library as most of our team have used this language this past and are quite familiar with it. Using this language helped us to build reusable components quickly and manage their states effectively, thus increasing our productivity and speeding up development time. We considered other JavaScript frontend frameworks such as Vue and Angular, however, we were not confident in using them for our project and decided against learning a new framework due to time restraints.

As mentioned previously, fast development is a priority of ours. Using the components in the Material UI library gave our frontend a sleek, professional look with very minimal effort. Though we initially wanted to prioritise functionality over aesthetics, using Material UI allowed us to achieve both within time restraints. Our group briefly discussed using Bootstrap, however, we found that since Material UI is pure CSS and does not require a library, such as jQuery, Material UI is a more convenient library to employ in our project. We took advantage of Material UI to quickly implement website responsiveness, consistent button and modal designs, and incorporate aesthetic icons.

Initially, we did not plan to use the Leaflet JavaScript library, however, once we had implemented a map in our application we realised we needed to implement map interaction. This is where we introduced Leaflet, utilising it's map markers to pin different locations. We decided on Leaflet as it is free to use and the leading open-source library for making maps interactive.

Backend

Python was our first choice of language to employ in our backend as every member of the team is experienced with it. It's clear and straightforward syntax helped with readability and maintainability. Employed in our application, Python exposed a RESTful API to the frontend, which then enabled us to abstract away all the complexity, while providing a simple, and well-documented interface for the frontend to interact with. This also allowed us to seamlessly add more features to the APIs, such as user account management, by simply adding another endpoint for the frontend to call.

FastAPI was chosen for our application due to its high performance, as it is said to be one of the fastest Python frameworks available. Unlike Flask, it can detect invalid data types at runtime, thus reducing the possibility for bugs, and compared to Django, takes minimal time and effort to implement. The readily available documentation helped members of our group to quickly understand how FastAPI works as they were straight to the point and easy to follow. FastAPI also also provided a testing page for quicker development. Overall, it was an efficient, productive choice for the implementation of our backend.

We decided that it was necessary to use an object-relational mapping (ORM) to help us access the database, especially since Python is an object-oriented language. For an ORM, we settled on SQLAlchemy as it operates smoothly with Python and has a similar style, thus reducing the complexity and learning curve of our project. Using this ORM simplified our database queries, helped us to keep our database organised, and made our code more readable and maintainable.

Database

We chose to use SQL for our database as it is a familiar language to our team and we have previous experience developing SQL databases. It is an effective and efficient language, enabling us to query and manipulate the data as we see fit.

SQLite was our chosen database management system as it is simple and uncomplicated to use, robust as it encapsulates our data into a single file, and is simple to employ since there is no need for setup or configuration. Since we were only required to implement a minimum viable product for this project, SQLite was a suitable choice for our needs. However, if we were to scale our application up and deploy it, we may then consider PostgreSQL. In the end, SQLite proved to be a satisfactory choice. We were able to implement a simple database, delete this file if we needed to start over or make any changes, and insert dummy data for demonstration purposes.

Web Hosting

Although we only hosted our application locally, we chose Docker as our containerization solution to decrease coupling with the operating system itself. Docker installs what is essentially a lighter version of a linux virtual machine on the server it is running before it runs the application, making it platform independent. The frontend, backend and database are also deployed in separate containers for decoupling purposes. If by any chance any one of these three containers go down, it would not affect the functioning of the other two. Furthermore, if we use Kubernetes for container management, the traffic from the user can be simply redirected to another container running the frontend, which allows for even lower downtime.

Each of the docker containers will then be hosted in Google cloud using the cloud compute engine by Google. Cloud computing is the best hosting solution for the app as there are no hardware maintenance costs, along with high scalability combined with the docker containers, since the hardware can also be easily scaled up using the google cloud console. It also allows us to easily add in kubernetes management for the docker containers in the future using the Google Kubernetes Engine (GKE).

Client Platform

The choices for providing support to Mac, Linux and Windows, and browser support for Safari, Mozilla Firefox, and Microsoft Edge 13 were decided based on the operating systems and browsers supported by our external data sources, Spotify API and Mapbox API. We wanted to ensure that our application can accommodate and fulfil the user's needs without any issues.

Project Implementation

Team Structure and Responsibilities

For the actual implementation of our project we decided to split up into two teams according to everyone's coding skills and preferences. The two teams were the backend and the frontend team. Not only this made things easier when making changes to the code, but also allowed us to work faster and complete tasks quicker. We decided to separate the group into two separate teams: backend and frontend. The two group members with the least frontend experience, Jiaqi and Reinier, were responsible for the functionality of the backend, whereas the rest of the group members, Jamie, Dimitri and Atharv, were delegated to work on the frontend.

Project Responsibilities:

- Project Manager (Atharv)-
    - Responsible for maintaining the GitHub repository
    - Responsible for delegating the tasks equally between team members
- Business Analyst (Reinier)-
    - Responsible for analysing purpose of application

        - Responsible for researching competition and how our application is better
- Scrum Master (Dimitri)-
    - Responsible for breaking down user stories and features into tasks and epics in Notion
    - Maintaining Notion and reminding team of anything that is late or of high priority
- Dev Tester & Developer
    - Backend Manager (Jiaqi)- Responsible for keeping the backend tidy and functioning
        - Backend Member(s)- Reinier
    - Frontend Manager (Jamie)- Responsible for keeping the frontend tidy and functioning
        - Frontend Member(s)- Dimitri and Alan

Backend Responsibilities:

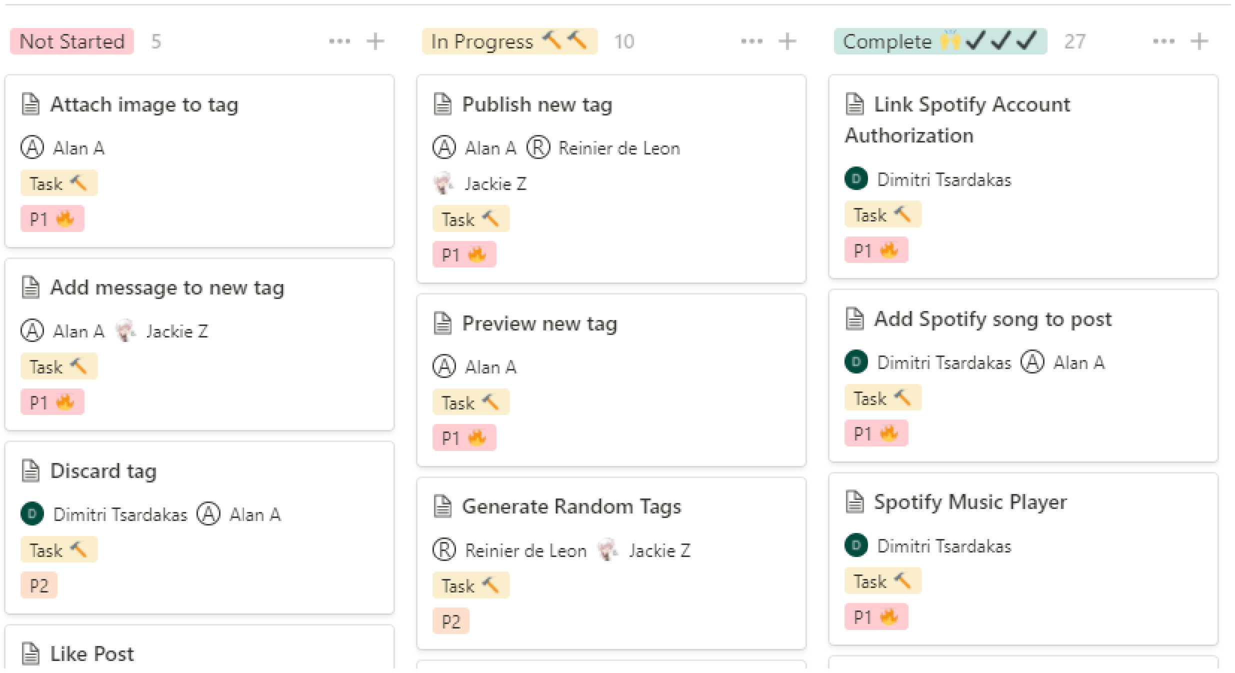
- Create ER diagram for the database
- Implement all tag functionalities and connectivity to database
- Ensure that the adding, removing and changing of a user's details works
- Have a functioning database that follows the ER diagram
- Connect API endpoints with the frontend

Frontend Responsibilities:

- Produce low-fidelity and high-fidelity prototypes
- Ensure the frontend meets all the project requirements
- Make sure frontend interface follows prototypes closely
- Make sure the frontend sends and receives the API requests properly, ensuring the correct values and files are sent to the correct places
- Ensure Spotify player is integrated well within the Tag

Meetings and Organization

To keep track of work and all tasks, we used Notion as our planner. Before starting to code, we broke down all of our user stories and features into tasks and sprints which made coding a lot easier. Here is an example of what our Notion planner looked like.



Not Started 5

Attach image to tag
Alan A
Task 🔨
P1 🔥

Add message to new tag
Alan A Jackie Z
Task 🔨
P1 🔥

Discard tag
Dimitri Tsardakas Alan A
Task 🔨
P2

Like Post

In Progress 🔨🔨 10

Publish new tag
Alan A Reinier de Leon
Jackie Z
Task 🔨
P1 🔥

Preview new tag
Alan A
Task 🔨
P1 🔥

Generate Random Tags
Reinier de Leon Jackie Z
Task 🔨
P2

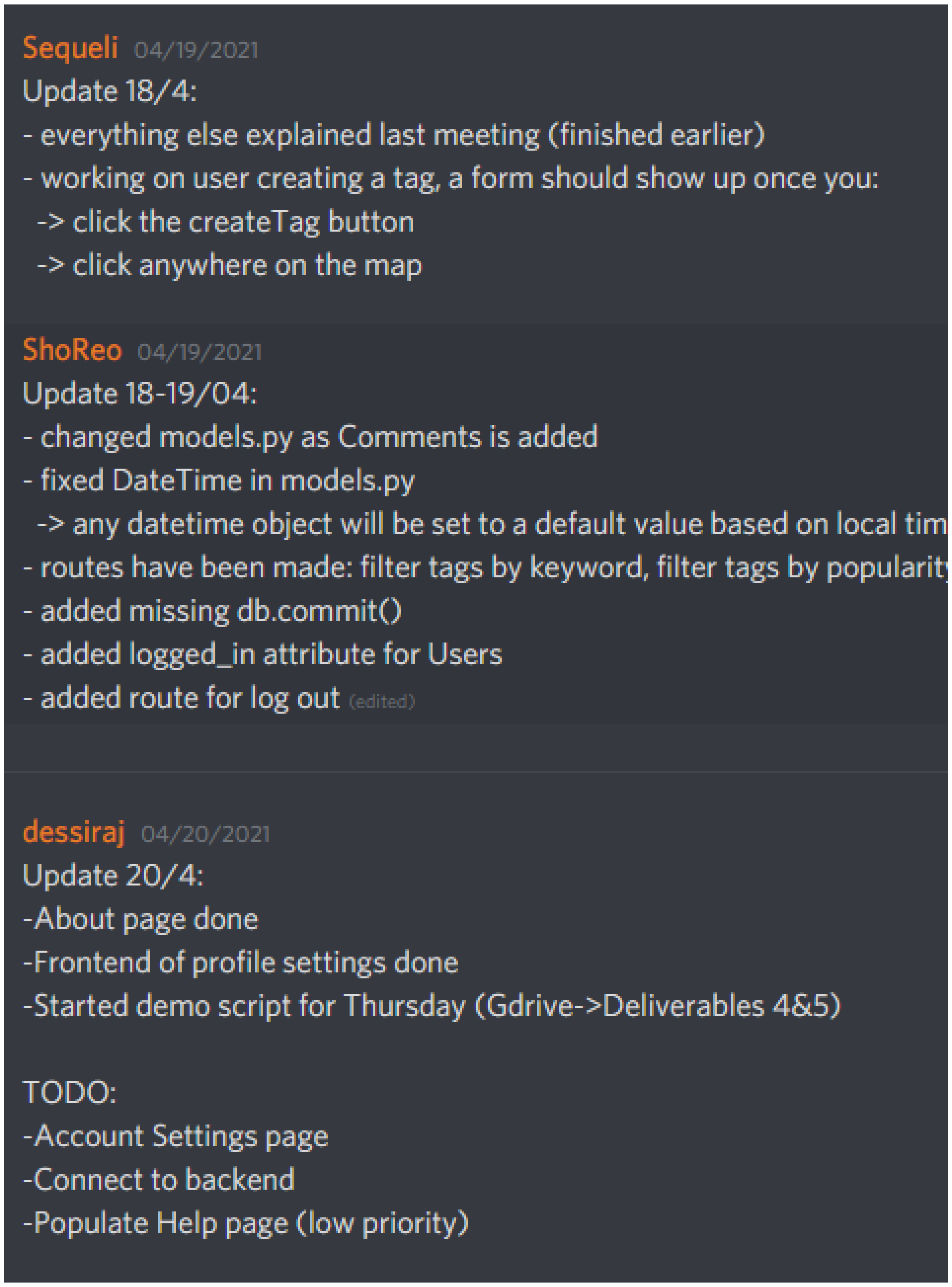
Complete 🙌✔✔✔ 27

Link Spotify Account Authorization
Dimitri Tsardakas
Task 🔨
P1 🔥

Add Spotify song to post
Dimitri Tsardakas Alan A
Task 🔨
P1 🔥

Spotify Music Player
Dimitri Tsardakas
Task 🔨
P1 🔥

For team meetings and organization we used Discord. We had around 2-3 quick group meetings a week and often we had discord calls to help each other out with any deliverables or with the code. On Discord we also added a channel for our Git where we could see all changes. We also updated each other there which turned out to be very good in terms of managing our team and the work.



Sequeli 04/19/2021
Update 18/4:
- everything else explained last meeting (finished earlier)
- working on user creating a tag, a form should show up once you:
  -> click the createTag button
  -> click anywhere on the map

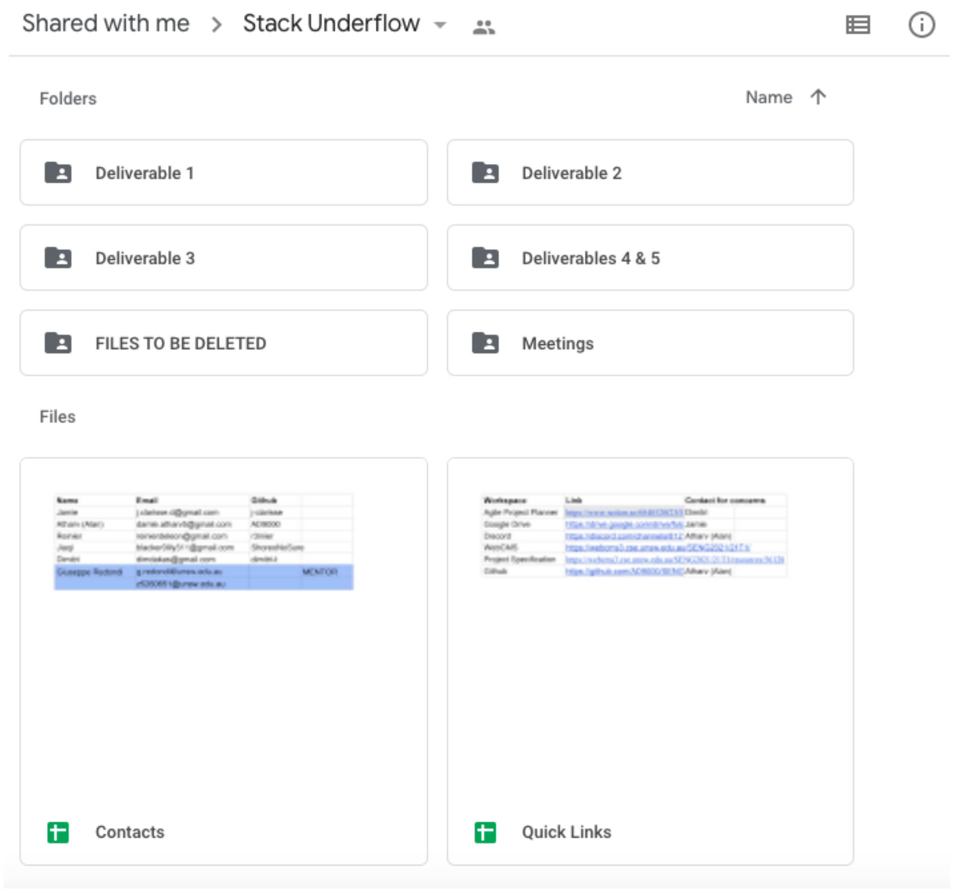
ShoReo 04/19/2021
Update 18-19/04:
- changed models.py as Comments is added
- fixed DateTime in models.py
  -> any datetime object will be set to a default value based on local tim
- routes have been made: filter tags by keyword, filter tags by popularit
- added missing db.commit()
- added logged_in attribute for Users
- added route for log out (edited)

dessiraj 04/20/2021
Update 20/4:
-About page done
-Frontend of profile settings done
-Started demo script for Thursday (Gdrive->Deliverables 4&5)

TODO:
-Account Settings page
-Connect to backend
-Populate Help page (low priority)

We also had a Google Drive folder for our project. We had folders for each team member's information on forms of contact as well as important links that we might need (ie. GitHub, WebCMS, Discord). We created a folder for each deliverable as well, which we used religiously to make collaborating much easier as it allowed us to make live changes and updates to certain documents for everyone to see. A "Meetings" folder was also created, where we created a document for all the discussed topics during our meetings.

Google Drive Link:
https://drive.google.com/drive/folders/1DheWt_TJsCZFiN0QuJPxYUmk-af_-Mn4?usp=sharing



Name	Email	Github	
Jamie	[illegible]	[illegible]	
Atharv (Alan)	[illegible]	AD8000	
Romer	[illegible]	[illegible]	
Jiaqi	[illegible]	[illegible]	
Dimitri	[illegible]	[illegible]	
Giuseppe Rodondi	[illegible]		MENTOR

Workspace	Link	Contact for concerns
Agile Project Planner	[illegible]	Dimitri
Google Drive	[illegible]	Jamie
Discord	[illegible]	Atharv (Alan)
WebCMS	[illegible]	
Project Specification	[illegible]	
Github	[illegible]	Atharv (Alan)

Project Summary and Team Reflection

In summary, our project has been successfully completed with almost all the features we set out to include. We have a fully functional frontend connected to our backend and our application feels elegant to use. Due to the time constraints of the course, we were not able to implement all of our user stories. Here are the stories we did not manage to complete:

- Resetting a password does not send an email to the user
- View Notifications (comments, likes)
- Set the website to light/dark mode
- Change the default setting for showing the side menu in the homepage
- Change the display language
- Liking a post
- Comment on a post
- Searching for posts

We were planning on completing all of the above but we did not have the time for it. However, we managed to implement all of our other functionality based on user stories and our web application looks great. Some of our favourites have been the world map integration because of how good it looks panning around the map and adding tags. Additionally, being able to search for a song when creating a tag then seeing it played by the Spotify player in the tag is also a big feature we completed.

We encountered a few problems with our implementation which was expected. At the start we thought Spotipy would've been a good choice for our Spotify API. Spotipy is a lightweight Python library for the Spotify Web API. However, there were some issues whilst using it and we couldn't get the Spotify Authentication working properly so we switched over to the actual Spotify Web API. Another issue that we faced was the rendering of files from the backend to the frontend, where React was having issues receiving image files.

This project has been intense, but has also allowed us to learn so much in terms of time management, group management, software developing and engineering. Like always, after completing a project there are things that could have been done better. Here are some things we believe that could have been done better:

- Start coding earlier to give us time to complete all of our features
- Didn't stick to plan in the early weeks as much as we should've and worked a bit later than what we should have
- We didn't try to get feedback on the usability of our application in terms of UI/UX.

There are lots of things we did very well during this project. One of them has been our Notion planner as shown previously. We broke down all of our user stories into tasks and our features into epics which worked really well. In addition, our Discord channel was a great place to share any updates, notes, ideas and have meetings. We also pushed ourselves within the coding aspect of the project and did really well.

There are definitely a few aspects of our implementation that could've been done differently or simply improved on. Adding a search bar with filters to search through tags in the front end would be very beneficial to our users. Additionally, we have thought about handling the situation where multiple tags are posted at the same location, and our solution is, instead of having only one tag in each location, each 'tag' on the map will open up a list of tags in that region. Furthermore, to facilitate more interaction and connection among the users, we can implement comments, likes and potentially messaging or group features.

This project pushed us to learn more about designing a software solution and implementing it. We learnt a lot about the business side of things, such as monetization, scalability, thinking about UI/UX which directly relates to the overall user experience. Being able to choose our own development stack was also something new but exciting since we all learned and developed our coding skills. Another exciting part of the project was designing lofi-hifi diagrams, then implementing them in the front end. Having complete control over our project and its requirements proved to be quite daunting and hard at the start, but highly rewarding once we saw our finished web application.