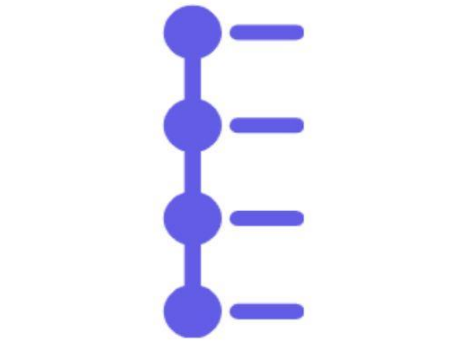
CS3216 Assignment 3



Group 10

Final Report

Link: https://cs3216-timeline.netlify.app/

Team:
Ang Chun Yang A0180311X
Balasubramaniam Praveen A0200541B
Chan Qin Liang A0208035N
Marcus A0194088R

Table of Contents

Milestone 0

Describe the problem that your application solves

For many, if they want to save their cherished moments through photos and videos, they would normally store them with their default gallery application or upload it to social media. However, pre-installed gallery applications tend to be limited in features, and are normally stored together with other photos and videos for other uses like screenshots and work. While social media like Instagram is an alternative solution, there are no features to organise your images and videos either, and your photos and videos are accessible to anyone that you allow to follow even if you make your account private, which is an inflexible approach if you have memories that you want to keep to yourself apart from those you share publicly. As such these applications do help to save photos and videos, but not in an ideal way for looking back at your memories in the future. We hope to create an app that solves this issue and allow users to have a unique way of visualizing their memories.

Milestone 1

Describe your application and explain how you intend to exploit the characteristics of mobile cloud computing to achieve your application's objectives, i.e. why does it make the most sense to implement your application as a mobile cloud application?

Timeline aims to provide a platform for users to collect and organise their significant memories in one place so that they can come back and reminisce about these memories in the future. It sets itself apart from a typical gallery application by storing more than just a photo or video for each memory, but also allows users to write an accompanying description for it and stores the location. These memories can then be displayed as part of a timeline, and users can organise their memories into multiple timelines. They can also view their memories as a line on the map, allowing them to see a geographic view of their timeline.

Since many people use their mobile phones to capture their photos and videos, it would make sense that our application targets mobile usage as it allows users to directly post to our application without first transferring to their computer. Similarly, social media users tend to scroll through prior posts by mobile rather than desktop. In 2019, mobile accounted for 79% of social media usage in the United States[1]. Also, implementing the application as a mobile cloud application means that we can make use of native features such as location tracking which is useful in our application. Since users do not have to download the application, users can easily try the application out to see if they prefer to use our application over their existing ways of collecting their memories, reducing the barriers to entry for users.

[1] https://www.statista.com/topics/4689/mobile-social-media-usage-in-the-united-states/

Milestone 2

Describe your target users. Explain how you plan to promote your application to attract your target users.

Target Users
Our target user base is quite broad because the problem that Timeline solves, keeping track of memories, is timeless. However, if we were to be a bit more specific, our target audience will be people that appreciate the sentimental value of photos and like looking back at their memories.

The user profile of the people using Timeline will be individuals that are already familiar with how to operate a smartphone and probably have experience using applications that allow them to share media with one another such as Instagram and Snapchat.

Our Plan
With our target users defined, the first part of the problem is how we are going to get them on board. For that, we have devised a multi-stage approach to attract users. These stages will correspond to the features that the application will offer.

Initial Stage
We will try to market this application as a phone gallery on steroids. In a typical phone gallery, you normally see all your photos in one album. Timeline offers users the ability to have different ‘lines’ for each group of memories. You can definitely do this with a phone gallery by manually moving images into different albums. However, the advantage timeline offers is that it lets you group memories together (e.g. a few pictures from one day) and add descriptions to that entire memory. Based on our understanding, phone galleries do not allow you to add additional information such as descriptions to images. This is another area where Timeline excels. Your memories can be tracked in a manner that’s much more flexible than your typical gallery application.

Mid Stage
By this stage, we would have ideally integrated features such as sharing memories with a group of friends. This means that we can start marketing this application to families, couples and groups of friends instead of individuals. We also hope to have a feature where the application will remind you about memories that happened near your current location. This is a nice way to remind users of positive memories and might be a decent way to continue to encourage them to use the application as they might want more pleasant surprises in the future.

Late Stage
By this stage, we aspire for our application (or company) to allow users to print out their memories. This is somewhat similar to features that some applications such as Day One offer. Day One is a journaling application that allows users to print out their e-journal entries into a physical diary. We feel that we can do something similar. Each line can be printed into a mini photo album/scrapbook with information such as the location and description of the memory.

Marketing Channels

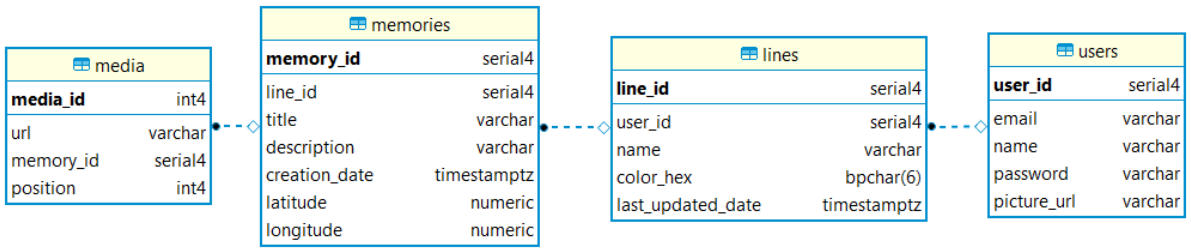
In the sections above, we talked about the different phases our application will have but did not elaborate on how we will be marketing the product. For an application that relies so much on the idea of memories, it is important to appeal to the viewer's emotions. To do this, the ideal platform would be to use instagram ads that feature videos that showcase the benefits our application offers.

In the initial stage of our application, we need to convince users why it's a good idea to start using the app. For this, it is important to highlight why our application is such a novel product. This can be done by showing them how they can try to do what our application does with existing applications. For example, it is possible to emulate some of the features our application offers with Instagram. Instagram has stories and highlights but they are not private and are visible to all of the user's followers. The user can create an account just for themselves but that does not solve all problems. To access a certain story in a particular highlight, users will have to keep tapping until they get to the story that they want to see. Furthermore, stories in Instagram are 100% chronological. We cannot backdate stories. However, we can add/edit memories in Timeline. For example, if we want to add new photos to a certain memory, we can. We can also edit descriptions but instagram does not allow us to edit stories that have already been posted.

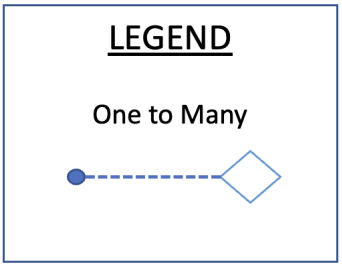
For the mid and late stages, we should focus on videos with stronger emotional appeal. These videos can be tailored based on the profile of the user that is viewing the ad. For example, parents with young children can be shown a video where they look back at their child's memories when their child has grown older. Couples can look back on their shared memories on their wedding day, and perhaps even print it out as a scrapbook. We feel that showing users advertisements like this will appeal to their emotions as they want to have similar experiences to those that were shown in the advertisements. This will hopefully convince them to get on board.

Milestone 3

Draw an Entity-Relationship diagram for your database schema.



The primary key of each table is labelled in bold at the top of each table. The lines between the tables represent a one-to-many relationship from right to left. For example, a user has 0 or more lines.



Milestone 4

Explore one alternative to REST API (may or may not be from the list above). Give a comparison of the chosen alternative against REST (pros and cons, the context of use, etc.). Between REST and your chosen alternative, identify which might be more appropriate for the application you are building for this project. Explain your choice.

We were trying to decide between REST and GraphQL for our API. We ultimately decided on using a REST. In this section, we will compare several aspects of REST and GraphQL before discussing why we settled on REST.

One benefit of GraphQL is that it gives the frontend the flexibility to query exactly what they need from the backend. This means that there are fewer redundant fields returned by the server. On the other hand, REST APIs are more 'chatty'. The frontend might need to send multiple API requests before consolidating all required data. It is also possible that certain endpoints return too much info. This means that REST APIs might use up more bandwidth as compared to GraphQL APIs.

However, REST APIs are significantly more cache-friendly than GraphQL APIs. This means that REST APIs can perform similarly (or perhaps even better than) GraphQL APIs. However, this is a difficult metric to measure and we cannot say with certainty that one performs better than the other.

REST APIs usually work very well on data that is organised in a rather 'flat' manner whereas GraphQL excels in use cases that require more complex data aggregation. In our case, our data is structured rather simply and can be cleanly divided into different sections (e.g. lines and memories). This structure is suitable for REST and we can model the API with endpoints representing the various collections.

One area where both approaches excel is the ability to decouple the client and server. This is an area where approaches like RPC perform poorly. RPC allows the client to call procedures (essentially functions) on the server. Each function can be tailored for a specific operation in the server. This has the potential to leak implementation details of the server. Now, if there's a change in the function name (or behaviour), both client and server will need to perform a non-trivial amount of refactoring. Another disadvantage is that client-side developers might also need to worry about issues like the potential side effects of various procedures. Furthermore, having one function for each feature will lead to an exponential growth in the number of functions and the documentation. For these reasons, we chose not to add RPC into the list of approaches we were considering.

After looking at the pros and cons of REST and GraphQL, and why we didn't want to consider some other approaches, we ultimately decided on REST as it was the best way to model the data we had in our application. Additionally, a minor point that helped us with our decision is that REST APIs require significantly less developer effort and tooling to implement. In a project with a rather tight deadline, REST seemed like the ideal choice.

Milestone 5

Design and document all your REST API. If you already use Apiary to collaborate within your team, you can simply submit an Apiary link. The documentation should describe the requests in terms of the triplet mentioned above. Do provide us with an explanation of the purpose of each request for reference. Also, explain how your API conforms to the REST principles and why you have chosen to ignore certain practices (if any). You will be penalised if your design violates principles for no good reason.

Apiary Link
You may view our API documentation at this link: https://timeline12.docs.apiary.io

REST Principles

Referenced from: https://www.ibm.com/sg-en/cloud/learn/rest-apis

1. **Uniform Interfaces**

The same URI standard is being used throughout our REST API schema to identify a resource.To retrieve a resource, the API HTTP method and endpoint will be of the form

```
GET /{collection_name}/{collection_id}
```

For example, to retrieve a memory with ID 300, the endpoint will be **`GET /memories/300`** and to retrieve a line with ID 2, the API will be **`GET /lines/2`**.

Similarly, the same standard is also applied in endpoints for manipulation of resources. To create a new resource, the API HTTP method and endpoint will be of the form

```
POST /{collection_name}
```

It does not include the ID as the ID is generated on the backend. The server will then return the newly created resource to the client, which comes with the newly generated resource ID. For example, to create a new line, the API will be **`POST /lines`**.

Finally, we also have endpoints to modify resources, which allows updating data fields of a resource. The API HTTP method and endpoint will be of the form

```
PATCH /{collection_name}/{collection_id}
```

This is similar to GET where the resource is identified by the same URI standard.

2. **Client-Server Architecture**

The Timeline app was designed using client-server architecture. Our client is the frontend React app, while our server is the backend Node app.

The only way the client and server can communicate with each other is through the specified REST API methods and endpoints, be it to create resources, retrieve resources, or modify

resources. The client and server can thus be modified individually and independently without fear of impacting the other. The API also allows for separation of concerns and decoupling of client from server. This allows the client to function without needing to know the internal implementation of the server, and likewise, the server can function without needing to know the internal implementation of the client. All they need to know is the API specifications provided by the server.

3. **Statelessness**

There are two key points in our discussion:
1. Response is created solely based on the request
2. Server does not store any information about the client's session

Based on the first requirement, our REST API is implemented such that the response is created solely based on the request. Each request sent to the server contains all necessary information for the server to be able to respond, without needing to rely on any session-related information inside the server.

To achieve the second requirement, our server should not keep track of session-related information. Thus, we decided to use JSON Web Token for our authentication implementation. The token contains all the necessary information to identify the user in the client session, which includes email and session expiration date. Only the server is able to decrypt the token. Based on the first requirement, the client has to attach the token to every request sent to the server, as the request should provide the server with all necessary information. This is done by attaching the token to the Authorization header of every request as follows:

```
Authorization: Bearer {token}
```

The server is able to identify client sessions without relying on any stored data, but only from data provided by the request. Thus, our server is thus free of application states.

4. **Cacheability**

Our REST API uses the default Cache Control headers (no-cache) which allows the client to use cached responses as long as they are always revalidated.

5. **Layered System Architecture**

The client and server do not know, and do not need to know, whether they are communicating with an intermediary server or directly to each other.

Milestone 6

Share with us some queries (at least 3) in your application that require database access. Provide the actual SQL queries you use (if you are using an ORM, find out the underlying query and provide both the ORM query and the underlying SQL query). Explain what the query is supposed to be doing.

1. **Query All Lines By User ID**

The following query returns all the lines owned by a user (with user_id = $1) with the details of the line, and also the details of the latest memory associated with each line if any, and the url of the first media of the said memory, if any. The lines are ordered by how recently they were updated, with the most recently updated returned as the first row.

```
SELECT L.*,
M.memory_id,
      M.title AS memory_title,
      M.description AS memory_description,
      M.creation_date AS memory_creation_date,
      M.latitude AS memory_latitude,
      M.longitude AS memory_longitude, (
            SELECT url
            FROM media
            WHERE memory_id = M.memory_id
            ORDER BY position
            LIMIT 1
      ) AS thumbnail_url
FROM lines L
LEFT JOIN memories M ON L.line_id = M.line_id
WHERE L.user_id = $1 AND (
      M.creation_date IS NULL
      OR
      M.creation_date = (
            SELECT MAX(creation_date)
            FROM memories
            WHERE line_id = M.line_id
      )
) ORDER BY L.last_updated_date DESC ;
```

2. **Query Memory by Memory ID**

The following query updates a specific memory by looking up the memory_id. Once updated, the updated row is returned. We used the “COALESCE” function so that we could use this same query for any combination of fields changed, where those fields that are not changed (passed a null value in the outer function) will just use the existing value in that specific field. This is safer than manually constructing a SQL query string by passing in a list of changes as a string, as our command allows us to make use of the variables (“$1”,”$2”, etc) to prevent SQL injection.

```
UPDATE memories
      SET line_id = COALESCE($1, line_id),
            title = COALESCE($2, title),
            description = COALESCE($3, description),
            latitude = COALESCE($4, latitude),
            longitude = COALESCE($5, longitude)
WHERE memory_id = $6
RETURNING * ;
```

3. **Query number of memories for a specific day**

This query returns the number of memories that a specific user has created for each day in the past, grouped by the creation date of the memories for a specified month and year. This can then be used for calendar features on the application.

```
SELECT date_part('day',creation_date) AS day, COUNT(memory_id) AS
number_of_memories
FROM memories M
JOIN lines L ON M.line_id = L.line_id
WHERE L.user_id = $1
      AND date_part('month',creation_date) = $2
      AND date_part('year',creation_date) = $3
GROUP BY date_part('day',creation_date) ;
```

Milestone 7

Create an attractive icon and splash screen for your application. Try adding your application to the home screen to make sure that they are working properly. Include an image of the icon and a screenshot of the splash screen in your write-up. If you did not implement a splash screen, justify your decision with a short paragraph. Add your application to the home screen to make sure that they are working properly. Make sure at least Safari on iOS and Chrome on Android are supported.

Icon

Splash Screen (Mobile)

Milestone 8

Style different UI components within your application using CSS in a structured way (i.e. marks will be deducted if you submit messy code). Explain why your UI design is the best possible UI for your application. Choose one of the CSS methodologies (or others if you know of them) and implement it in your application. Justify your choice of methodology.

Why our UI Design is the best

Our UI design fits our application because we wanted to have the users be able to perform actions using the app without much difficulty, keeping it easy for the user to navigate through the main user flows. Most importantly, we want to ensure that it is easy for them to view their memories and create a memory.

For most of our pages where buttons are required, we displayed them using the full width of the screen to make it clear and obvious for the user on its' purpose. This helps to captures the users' attention and know what tot

Furthermore, we wanted to provide the user different options to do the same task, giving them the freedom to choose whichever option is better for them. For instance, for logging in, they could do so by logging in through a Gmail/ Facebook account, or they can register through the conventional way. Furthermore, we also provided 2 different options of viewing the memories, in a timeline or in a map (This will be further elaborated in Milestone 12).

For creating a memory, the user needs to choose a location and there is a map which will show a marker on the chosen location. This allows the user not only search for a location but also use the map displayed to confirm that the selected location is right. Furthermore, for the map displayed when creating a memory, we disabled the map from moving so that the user will not accidentally move or zoom the map, since the map is just for displaying purposes. Hence, this prevents possible frustrations by the users when creating a memory.

In the Home Screen, besides including a search bar to search for the Lines, we also sorted the lines such that the most updated line will be at the top. Reason being users would probably most likely want to view their more recent changes. Also, we felt that photos were a good way to help users with identifying and remembering what memories were made, hence, we added a thumbnail image for each card.

Choice of CSS Methodology

Although we mainly made use of the Material UI Library to handle the styling of our UI (which will be further elaborated in Milestone 12), we still followed CSS methodologies when customizing the components to ensure that any styling related code is neat.

We followed Object Oriented CSS by abstracting out anything related to styles such as font size and color into a separate file. For instance, we have a colors.js file which contains all the common colors that we use for our design of our UI. Furthermore, we made use of a ThemeProvider to customize and ensure consistency of our font-sizes and primary color throughout our application.

Abstracting out the styling related code makes it easy for us to update them everywhere in our code. This is because if we want to use a purple color (#765dff) for our main color for our app design, we could simply do so by adding this color in the theme.js file, as the palette.primary.main property. This helped to make styling reusable and easily maintainable. If we wanted to change the color after getting more user feedback, we could simply update the color by just changing 1 line of code in the theme.js file, instead of updating every component that uses the color. We also abstracted out reusable components

Milestone 9

Set up HTTPS for your application, and also redirect users to the https:// version if the user tries to access your site via http://. HTTPS doesn't automatically make your end-to-end communication secure. List 3 best practices for adopting HTTPS for your application.

Do not mix HTTP and HTTPs

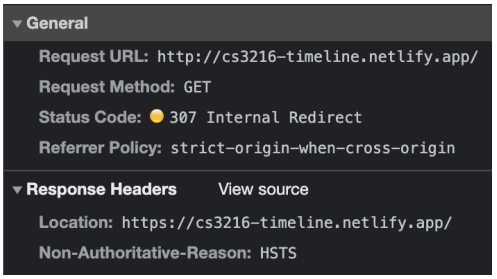
We should not mix HTTP and HTTPS. We should not be loading any content over HTTP on our site. In our case, all data retrieved by our endpoints and external services use HTTPS.

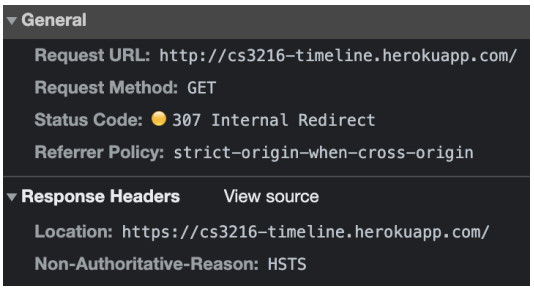
Certificates

We should use digital certificates from trusted Certificate Authorities (CAs). When a website has a certificate that is signed by a trusted CA, we know that we are visiting the real site. In our case, we hosted our backend on Heroku and frontend on Netlify. Both these providers have certificates signed by trusted CAs.

HTTP Strict Transport Security

We should enable HTTP Strict Transport Security (HSTS). HSTS ensures that browsers never connect to a site over HTTP and converts attempts to connect over HTTP to HTTPS. We have HSTS enabled on both our frontend and backend.





Milestone 10

Implement and briefly describe the offline functionality of your application. Explain why the offline functionality of your application fits users' expectations. Implement and explain how you will keep your client synchronised with the server if your application is being used offline. Elaborate on the cases you have taken into consideration and how they will be handled.

In order to implement the offline functionality of our application, we made use of a npm package called redux-persist so that the redux store is persisted. The package helps to cache certain data that we deem to be necessary in the web storage, so that the users will be able to view certain data even when the user is offline.

Below are some of the offline functionalities of our application:

1. **Viewing lines offline in the Home Page**

The lines that belong to the user are cached when the user first logs or registers, and they will get updated whenever the user goes to the home page. Thus, when the user is offline, he would still be able to view the content in the Home Page. It fits the user's expectations because they will be able to view their existing lines offline, and have a summary of what memories that were made in each line.

2. **Creating a draft line**

The user can create a draft line when he is offline, and it will be displayed on the top of the Home Screen once created. There will be an option for him to retry creating that line or delete that draft line. The retry creating function will work once he goes back online, and the line will be created. Having this functionality prevents the user from losing changes made when the user creates a new line.

3. **Viewing the calendar information and memories in each date even when offline**

The date tile of the calendar will be shaded in purple if the user has a memory created on that day. This data will be cached too, so when a user goes offline, he would still be able to see which are the dates where he has created memories. Furthermore, the memories that were made on the day can be seen offline too as long as the user selected the date before when he is online.

Other than the features mentioned above, we considered allowing users to create a draft for a memory when offline too. However, since we made use of the geolocation API to obtain location suggestions for the user when creating a memory, it was not possible to utilize the API when the user is offline. Hence, we decided not to implement this feature.

Milestone 11

Compare the advantages and disadvantages of token-based authentication against session-based authentication. Justify why your choice of authentication scheme is the best for your application

Scalability:

In session-based authentication, the server will have to store the session information for each user. On the other hand, token-based authentication by default does not require the storage of any additional information on the server side. Hence, for applications with a large and growing user base, token-based authentication may be more suitable.

Native platforms:

Session-based authentication usually makes use of cookies, which is not as suitable for native platforms as they may have to use technologies like webview to use cookies as a browser would. Furthermore, cookies may be disabled in browsers especially if the cookies are cross-domain, which might be the case when the API is served from a different domain from the application. This is not a problem for token-based authentication, as the authorization header can be used for any request to pass the token from the client to the server.

Size:

A typical JWT can be significantly bigger than the session id stored in a cookie. This means higher overhead when sending requests from the client to the server.

Flexibility:

A single token can be used to authenticate a user across many different services. It can also carry any data associated with the user. A session cookie just contains the session identifier, which means additional data has to be checked against a data store and sent separately if needed.

Our Decision:

We decided to use token-based authentication due to its scalability and its greater support for multiple platforms, as we felt these advantages are critical if our application were to expand in the future, both in terms of the number of users and the number of platforms supported. Also, we felt that neither method is more secure than the other and the advantages of token-based authentication are worth the additional size of the token.

Milestone 12

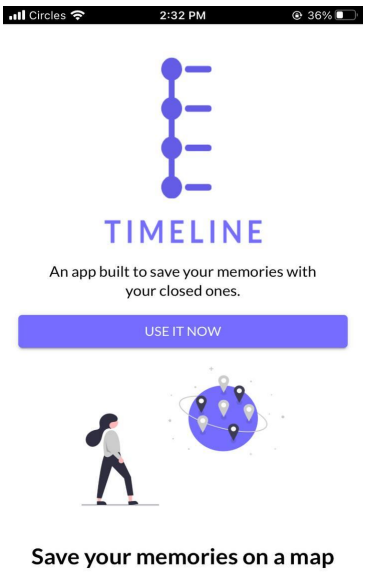
Justify your choice of framework/library by comparing it against others. Explain why the one you have chosen best fulfils your needs. Lastly, list down some (at least 5) of the mobile site design principles and which pages/screens demonstrate them.

We decided to use Material UI as the CSS framework for our application. Using the Material UI Library allowed our application to have a consistent native look and feel throughout all screens, especially since Material was built using a mobile-first approach. Furthermore, Material UI components are easily customizable and well documented, saving a lot of time on the developers side to create a user friendly interface. It also uses a grid system, making it easy for us to develop our app to be responsive in the future if we hope to do so.
We chose to use Material UI compared to other libraries such as Bootstrap, Tailwind and Semantic UI because we felt that the overall appearance of Material UI's components was more appealing. Furthermore, Material UI was a library built for React, which we are using for our frontend. Comparing Material UI to Tailwind, Tailwind is a framework meant for styling our own components, whereas Material UI is a framework of basic reusable components. Hence, this is also why we felt that it was better in terms of building the necessary features for our app's requirements within such a short time, while providing excellent designs for each component.

Next, we will share about some of the mobile design principles that we have utilized:

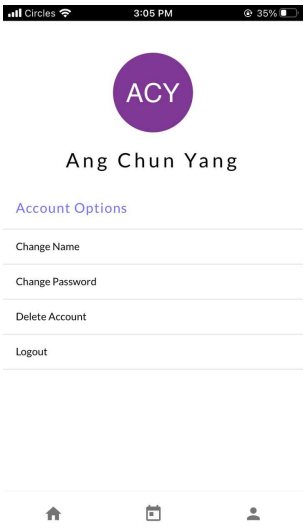
Keeping call to action buttons mobile friendly

We made our call-to-action buttons big, capitalizing them, and taking up the whole width of the screen so that it is clear to the user on mobile. For instance, in our landing page, we have a call-to-action called "USE IT NOW". We will have more examples of such call-to-action buttons being used in the user flows shown in the other images in this Milestone and Milestone 13.



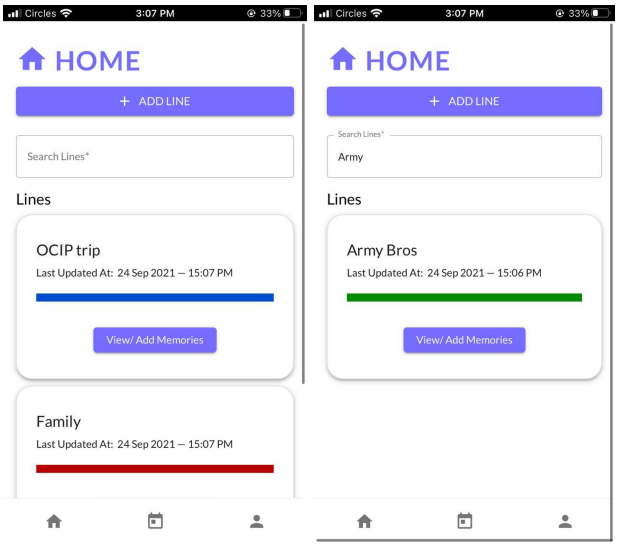
Making menus short and sweet

In every screen, we placed a bottom navigation bar that allows the user to go to the Home Page, Calendar Page and Profile Page, as we felt that these are the 3 main components required for the user to start finding, editing or creating their memories. Another example of how we followed this principle is in our profile screen. We only showed the necessary actions that a user can do such as changing password, deleting account, changing his name and logging out.



Implementing search feature to narrow down view

It would be difficult for a user to find a specific line if they created many of them. Hence, we utilized the principle of narrowing down views for the user by implementing a search feature in the home screen so that the users are able to find their specific line easily.

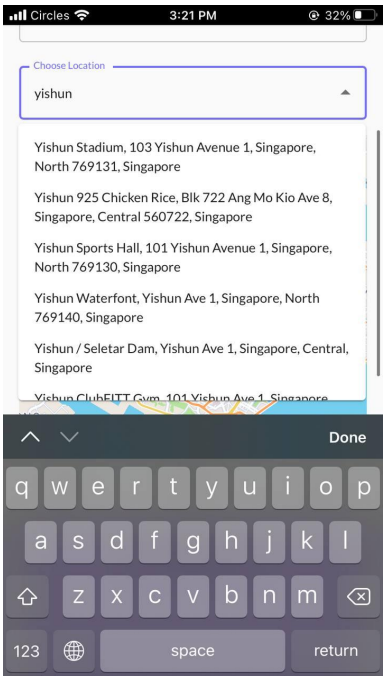


Using only vertical scrolling and not using horizontal scrolling

For all of our screens, we only allow vertical scrolling, which is becoming a standard navigation pattern.

Minimize user effort to input data

To improve the user experience, we allowed the user to login through google/facebook so they do not need to register another account. Also, for choosing a location when a user is creating a memory, we implemented a search feature that provides suggestions for the user so that the user does not need to type in the specific location even if it is very long, hence reducing their effort to input data.



Keeping forms short and sweet

As we know that users would prefer forms to be short, for all of our forms, we thought through and decided the necessary information that they need to fill in. For example, when creating a memory, they are only required to fill in the title of the memory, location, and photos. Description was made optional and some details such as createdDate of the memory are applied in our backend instead of leaving it to be filled by the user.

Keeping user in a single browser window

We designed Timeline such that the user is not redirected to another browser when using the application. The only time where they need to do so is when they login through google or facebook as they will be redirected to verify their gmail/ facebook. However, this was something unavoidable to improve the onboarding experience for the user.

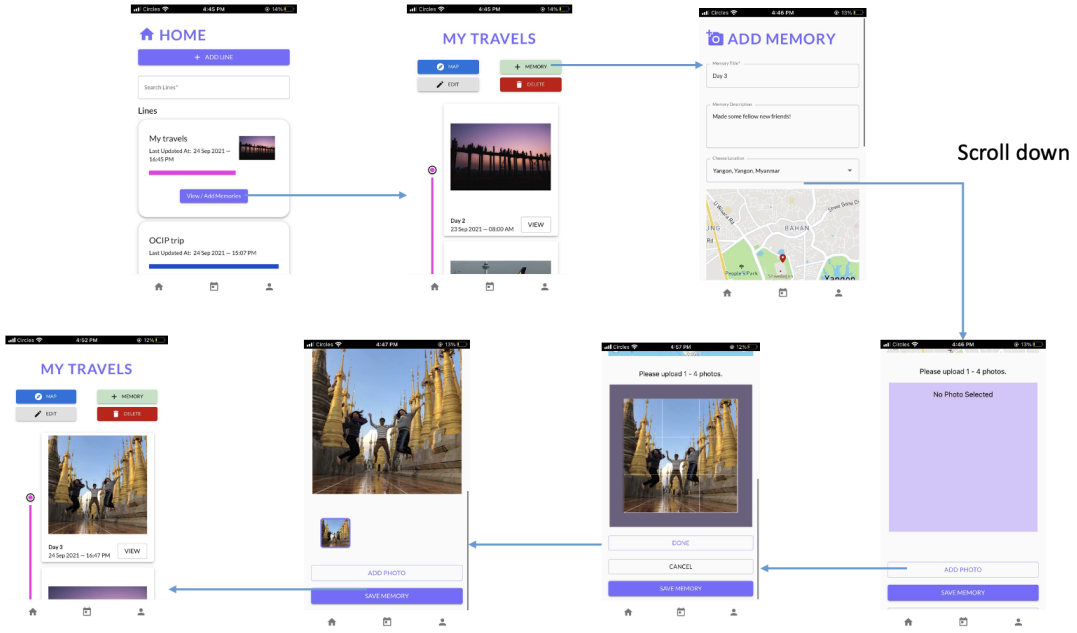
Making it easy to go back to Home Screen

We added the Bottom Navigation Bar as seen in most of the screenshots too so that the user can easily navigate back to the Home Screen, an important function to allow the user to go back to the starting point of the application if he wants to start creating a line/memory from scratch.

Milestone 13

Describe 3 common workflows within your application. Explain why those workflows were chosen over alternatives with regards to improving the user's overall experience with your application.

Creating a Memory



To create a memory, the user will go to an existing timeline and select the *Add Memory* button. This will lead the user to a form, where the user should fill in the following details:

1. Title of the memory (required)
2. Description of the memory (optional)
3. Location of the memory (required)
4. Photos of the memory (required, up to 4 photos)
    a. Upon upload, a cropper will be displayed.
    b. The user will have to crop the photo into the square cropping window.
    c. The user will click *Done* to add the cropped photo

Once the user is done filling the form, the user then clicks *Save Memory* which will redirect the user to the newly created memory.

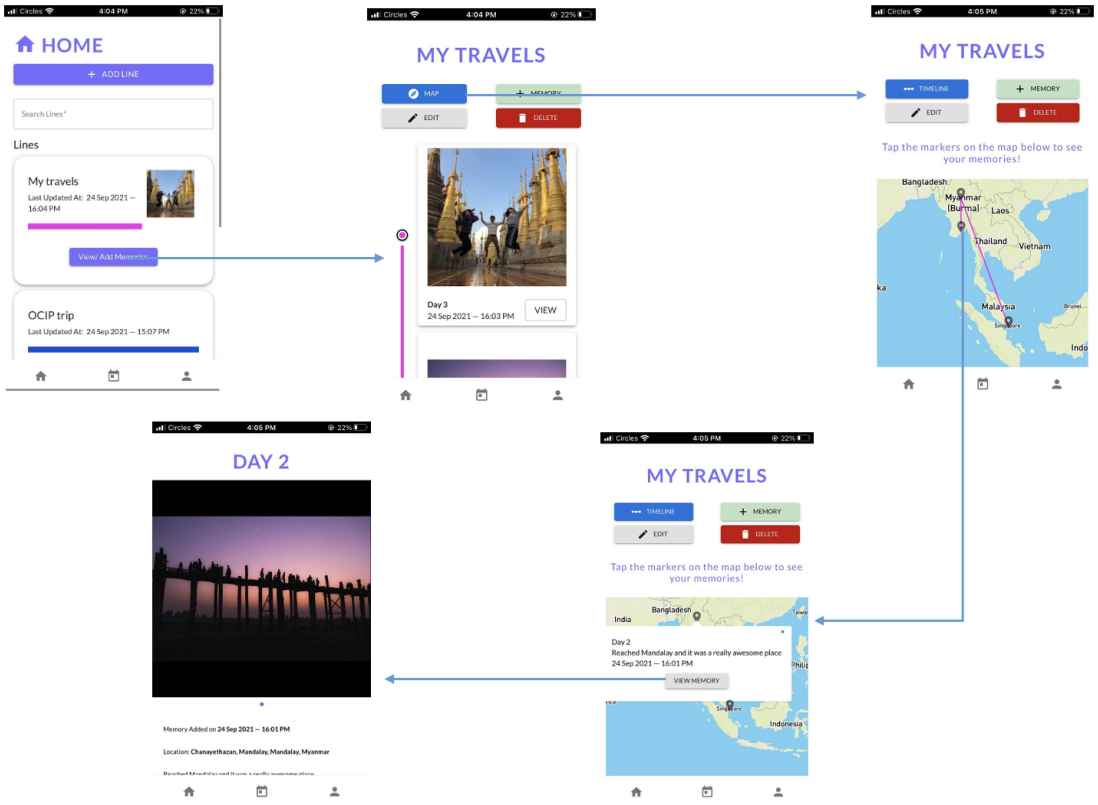
There were several considerations we made for the flow, specifically:

1. Should description be required or optional? **Optional**
    a. We felt that there would be a good number of users who would not want to fill in so many details when creating a memory, hence we allowed description to be an optional field. This also follows the mobile design principles of keeping forms short and sweet.
2. Should photos be required or optional? **Required**

a. A memory page's main content consists of its photos and description. The only reason for users to view memory content would be to view its main content, but we realised that one of these two is sufficient.
b. Between photo and description, we felt that photos gave our app much more uniqueness, as description only would be no different from a blog. Photos are also much more meaningful and relevant to our app's theme which revolves around keeping memories. Memory with only description and without photos feels empty as compared to memory with photos and without description.
c. Thus, photos are made a requirement while description is not enforced.

3. Should location be required or optional? **Required**
   a. One of the features we included in this app is a Map Display of memories, which is further described below as our second main flow. This feature allows users to view the places associated with their memories being connected as a line, inspired by MRT lines. Points are connected in sequential, chronological order. Thus, having one memory without location will mess up this Map Display, as we felt it may not give a good user experience if lines skip some memory due to missing location.
   b. Furthermore, referring back to our theme of storing memories, we felt that like the date and time, location is also an important part of a memory. Thus, this rule is enforced.

Viewing the memories as a timeline/ map



The user can view the memories that they created as a timeline. The timeline will be ordered such that the most recent memory will be shown at the top and the oldest memory will be

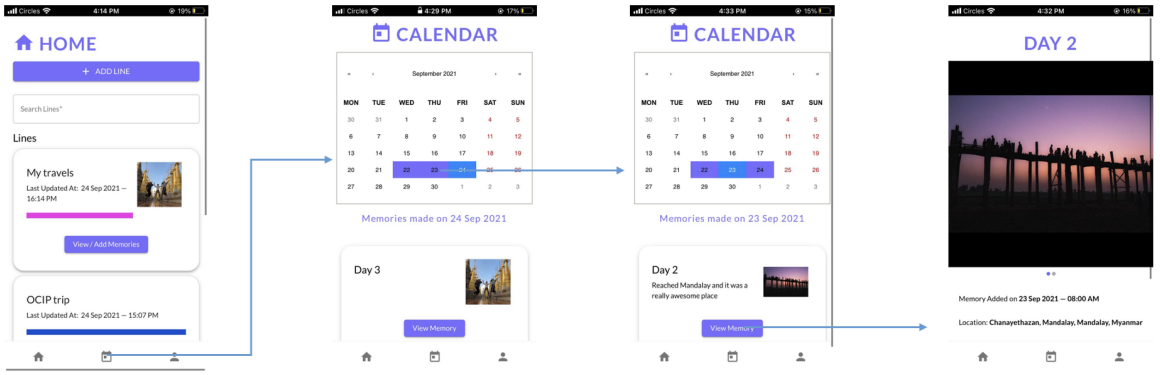
shown at the bottom. The user can also click the map button to toggle the view such that the memories will be marked as markers on a map, and these markers are clickable too, allowing the user to click on it and navigate to a specific memory. The markers that represent each memory on a map will be connected to each other in chronological order too, providing a unique experience for the users to see all their memories.

Initially, we only wanted to display the memories in a timeline. Our other option was to display the memories on a map. We decided on displaying them on a timeline as it felt that this would better fit mobile design principles of only allowing vertical scrolling.

After implementing the feature of displaying the memories as a timeline, we got our friends to do some user testing a week before the submission of the project. Many of them liked the idea of showing the memories using a timeline. However, a good number of them felt that it would have been nice to have another way to visualize how the memories are connected to each other too. We then brought up our other idea of connecting the memories and displaying them on a map, essentially a timeline represented in a map. Many of them felt that this idea was really cool as this was unique and they have not heard of such an idea before. Hence, we felt that it was worth spending some time to implement two different views of showing the memories.

This way, they are able to have a clearer visualisation of where all the memories from a line are made too, instead of going to each individual memory to find out, improving user experience. Furthermore, this idea really appealed to users who like to travel, as they are able to create their memory each day and plot it on a map, tracking where they have gone and show their itinerary on a map easily.

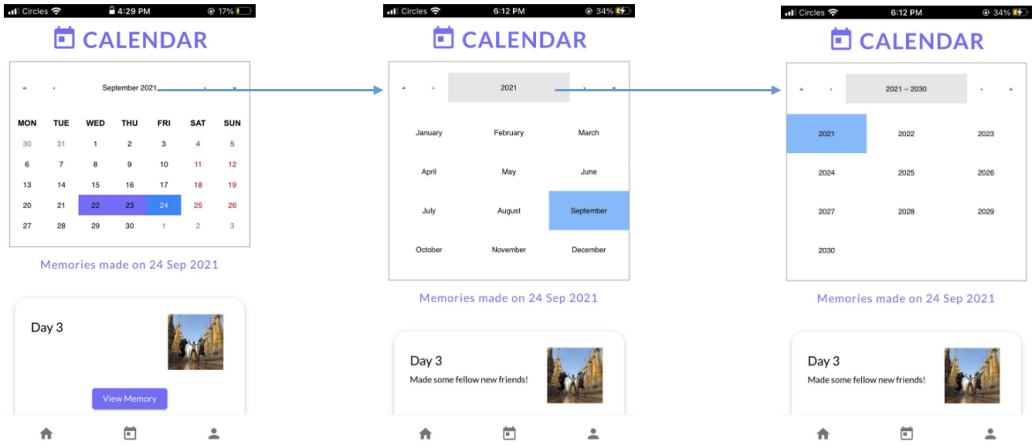
Viewing a memory through the calendar feature



Users can also view the memories that were made on each day through the calendar feature. In the bottom navigation bar, the user can tap on the calendar icon and he will be directed to a calendar. The date tiles that are shaded in purple indicates that the date has memories made on that day. Once he selects a date, the memories that were made on the day will be displayed below the calendar, and he can click the “View Memory” button in the card to be directed to the memory page.

The main idea for the Calendar feature was to create a different way for the user to access each memory, as we were concerned about the case where the user might forget which line one of the memories that he created belongs to. The other alternative was to use the search bar in the Home Page to search for that specific memory. However, there could be a case where the user forgot the title/description of that specific memory. After obtaining some feedback and conducting user interviews, we found that most people felt that one of the most important factors on how they would remember such a memory is through a photo. However, photos are not searchable through a search bar.

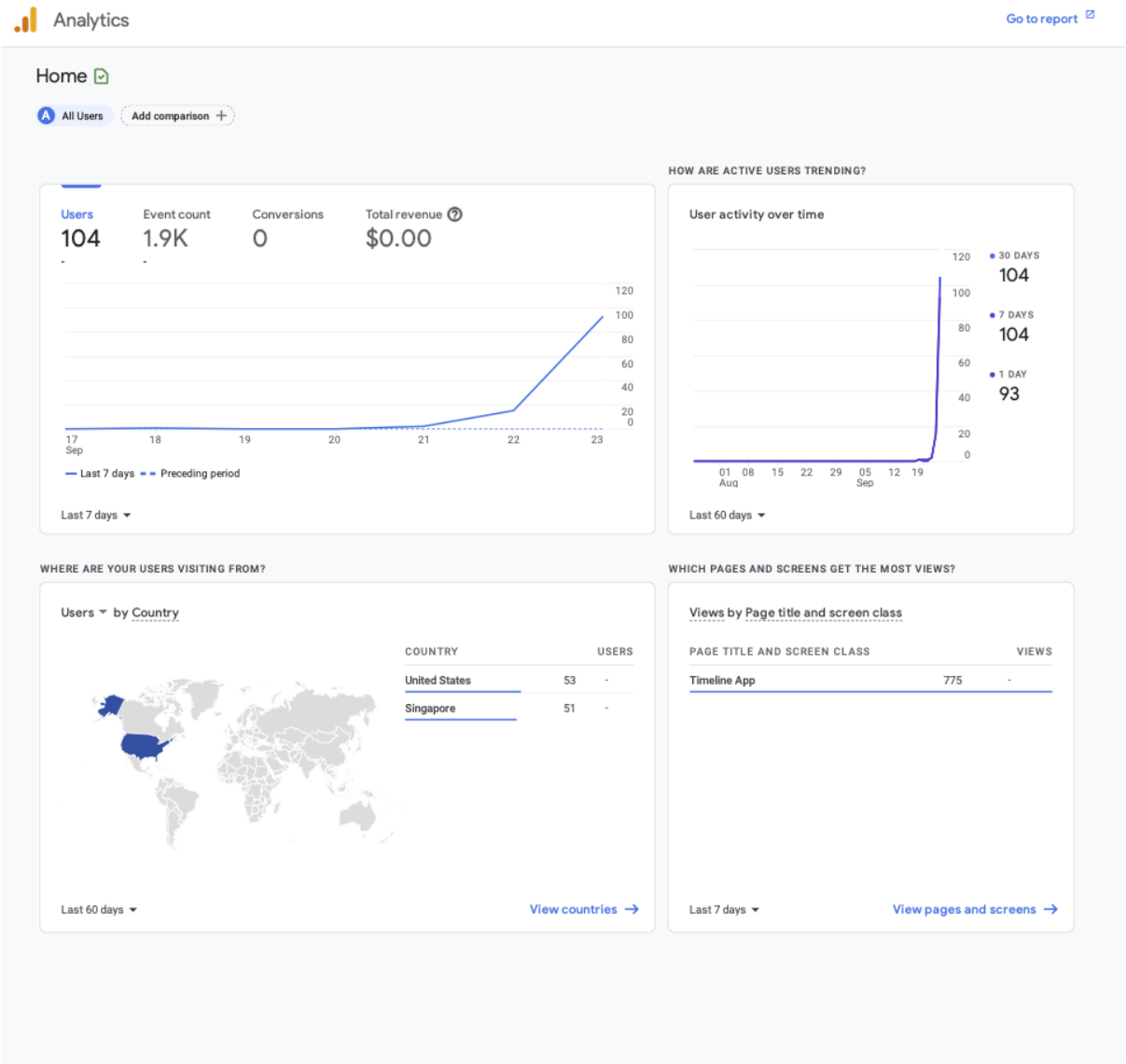
Hence, we decided to use a calendar for the user to find the memories where they forgot which line it belongs to. We found out from those whom we interviewed that many people would also be able to remember when that memory happened. Hence, we felt that it would be a good idea to allow the users to search for that specific memory using a calendar. The UI of the calendar also allows easy navigation between months and years, hence making it easy and quick for the users to search for a memory.



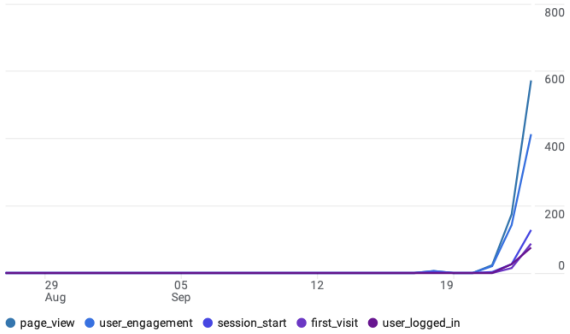
Furthermore, by indicating dates with memories with our app's theme color (purple), it helps to save time for the user as they do not need to click each and every date to find out if a memory was created on that day. This helps to significantly reduce the time taken for a user to search for a memory that they forgot which line it belongs to.

Milestone 14

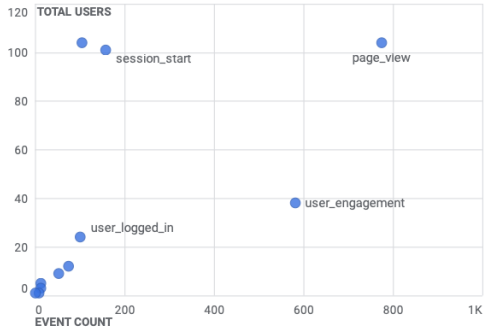
Embed Google Analytics or equivalent alternatives in your application and give us a screenshot of the report. Make sure you embed the tracker at least 48 hours before the submission deadline as updates for Google Analytics are reported once per day



Event count by Event name over time



Event count and Total users by Event name



Search...

Rows per page: 10 Go to: 1 1-10 of 11

Event name	Event count	Total users	Event count per user	Total revenue
Totals	1,885 100% of total	104 100% of total	18.13 Avg 0%	$0.00
1 page_view	775	104	7.45	$0.00
2 user_engagement	582	38	15.32	$0.00
3 session_start	158	101	1.56	$0.00
4 first_visit	105	104	1.01	$0.00
5 user_logged_in	101	24	4.21	$0.00
6 visit_landing_page	75	12	6.25	$0.00
7 create_line	53	9	5.89	$0.00
8 create_memory	13	5	2.60	$0.00
9 user_deleted_account	13	3	4.33	$0.00
10 notification_received	9	1	9.00	$0.00

Milestone 15

*Achieve a score of at least 8/9 for the Progressive Web App category on **mobile** (automated checks only) and include the Lighthouse HTML report in your repository.*

Refer to the HTML report in the repository root.

Milestone 16

Identify and integrate with social network(s) containing users in your target audience. State the social plugins you have used. Explain your choice of social network(s) and plugins. (Optional)

We have integrated Google and Facebook social login to our application. This allows users to use their existing accounts with Google and Facebook to use our application, without having to register separately with us, making it more convenient for the user. We chose Google and Facebook because of the sheer size of their user base compared to other platforms. Gmail itself has 1.5 billion active users in 2019[2], while Facebook has 2.89 billion active users in 2021[3]. Hence, many users should be able to use these accounts to login and benefit from this integration.

Milestone 17

Make use of the Geolocation API in your application. (Optional)

We have made use of the Geolocation API to get the location of the user, search locations and retrieve location given the latitude and longitude. By knowing the location of the user, we will be able to provide better suggestions when they search for a name of a place when creating a memory, as we can provide suggestions of locations closer to where he is currently.

[2] https://www.cnbc.com/2019/10/26/gmail-dominates-consumer-email-with-1point5-billion-users.html
[3] https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/