halcyon

(adj.) calm & peaceful

# CS3216 Assignment 3

Halcyon - A Self-Care Application

**Group 2**
Adam Chew Yong Soon
Gabriel Tan Chuan En
He Xinyue
Lim Yong Shen, Kevin

Link to Halcyon: [https://halcyon.caephler.com/](https://halcyon.caephler.com/)

## Table of Content

## Milestone 0 - Problem Description

People are leading increasingly fast-paced lives and need time to destress. The COVID-19 pandemic has only exacerbated the importance of mental well-being. The loss of work schedules and routines as a result of social distancing has led to significant physical and psychological impacts. Self-care has never been more crucial.

Self-care is a general term that describes one's deliberate actions towards bettering his/her mental, physical, and emotional well-being. According to Kelsey Patel[1], a Los Angeles–based wellness expert, [self-care] means listening to your body, taking moments to check-in, intentionally tuning in to the thoughts going on in your mind, and challenging your behaviours and belief systems if things feel out of alignment in your life.

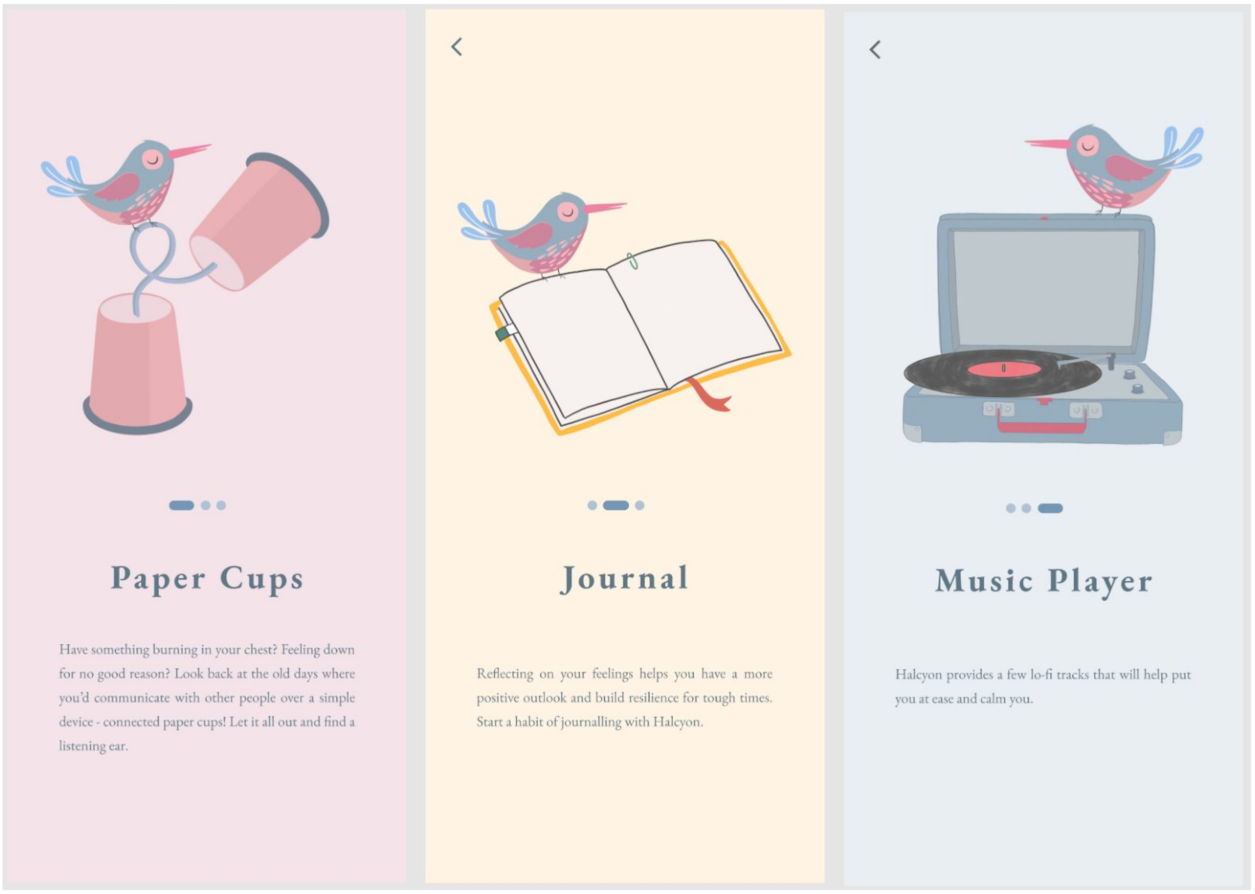
We recognise that self-care applications are an easy and effective means of enabling people to focus on their well-being and increase positive thinking. Such applications also reduce susceptibility to stress, anxiety, depression and other mental health issues.

Halcyon aims to help users start meaningful self-care routines. It provides a calm and peaceful space for users to take a pause from their hectic lives and check in on their inner feelings.

[1] Kelsey Patel: https://www.kelseyjpatel.com/

## Milestone 1 - Application Description & Mobile Cloud Computing

Gratitude is an essential part of self-care. Surrounding this theme, we wanted to cultivate self-care in our users. Halcyon provides a daily gratitude journal and an anonymous voice messaging service that allows users to voice their thoughts or reply with encouraging messages to other people around the world. Users can also listen to calming music through our vinyl player. We hope that it will help users to cherish the good things in life, increase their optimism, maintain healthy relationships with themselves, and spread positivity by caring for others.


Paper Cups

Have something burning in your chest? Feeling down for no good reason? Look back at the old days where you'd communicate with other people over a simple device - connected paper cups! Let it all out and find a listening ear.

Journal

Reflecting on your feelings helps you have a more positive outlook and build resilience for tough times. Start a habit of journalling with Halcyon.

Music Player

Halcyon provides a few lo-fi tracks that will help put you at ease and calm you.

Figure 1.1: Three key features of our application

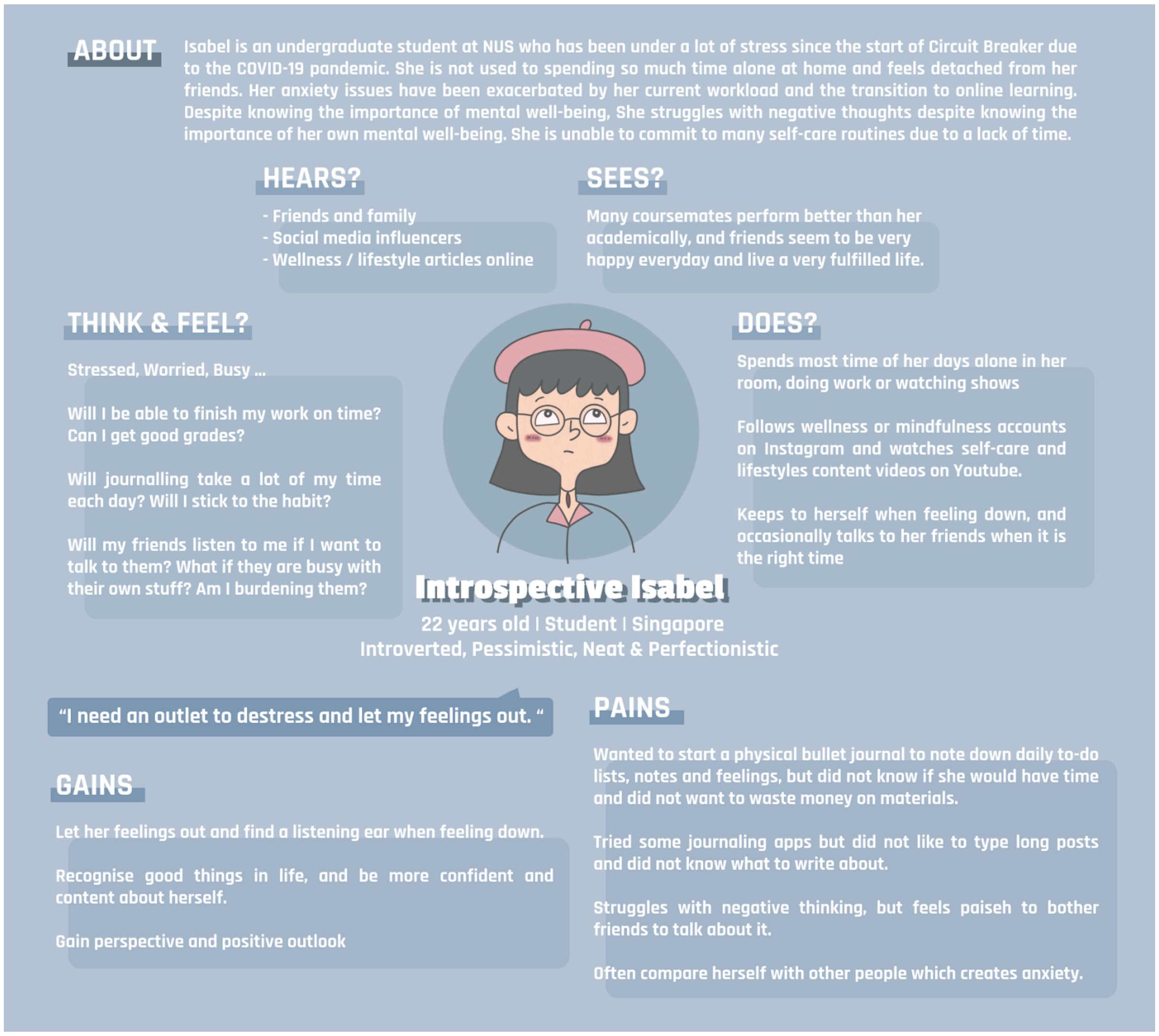
Halcyon's features do not require heavy computation. Hence, it makes sense to implement it as a mobile cloud application. The user interface is highly intuitive, and anyone with an internet connection will be able to use it. Self-care applies to everyone in the world. Thus, the application should be as accessible as possible. In some areas, people do not have access to the latest smartphones or might have an aversion to downloading mobile applications. The choice to implement a progressive web application was natural since users can get a native-like feel through a mobile browser. Targeting mobile phones significantly increases Halcyon's accessibility.

Halcyon leverages on the cloud to store data. Most computations can be done locally, and many features are still available when the application is offline. The application does this by using the mobile phone's own local storage cache data. Users are still able to add journal entries and listen to music while offline.

## Milestone 2: Target Audience & Marketing Plan

With Halcyon, we aim to target teenagers and young adults who are conscious about self-care and mental health, and to help them start meaningful self-care routines. Having said that, we encourage users from any demographic to use our app.

We needed to empathise with our users and consider their reasons for using the app. We concluded that there are two types of potential users. Those who need to let their feelings out and find a listening ear (primary target audience), and those who are empathetic and love to help others (secondary target audience). The figure below shows the user persona card for the primary target audience of our application.



**ABOUT**

Isabel is an undergraduate student at NUS who has been under a lot of stress since the start of Circuit Breaker due to the COVID-19 pandemic. She is not used to spending so much time alone at home and feels detached from her friends. Her anxiety issues have been exacerbated by her current workload and the transition to online learning. Despite knowing the importance of mental well-being, She struggles with negative thoughts despite knowing the importance of her own mental well-being. She is unable to commit to many self-care routines due to a lack of time.

**HEARS?**

- Friends and family
- Social media influencers
- Wellness / lifestyle articles online

**SEES?**

Many coursemates perform better than her academically, and friends seem to be very happy everyday and live a very fulfilled life.

**THINK & FEEL?**

Stressed, Worried, Busy ...

Will I be able to finish my work on time? Can I get good grades?

Will journalling take a lot of my time each day? Will I stick to the habit?

Will my friends listen to me if I want to talk to them? What if they are busy with their own stuff? Am I burdening them?

**DOES?**

Spends most time of her days alone in her room, doing work or watching shows

Follows wellness or mindfulness accounts on Instagram and watches self-care and lifestyles content videos on Youtube.

Keeps to herself when feeling down, and occasionally talks to her friends when it is the right time

**Introspective Isabel**

22 years old | Student | Singapore
Introverted, Pessimistic, Neat & Perfectionistic

"I need an outlet to destress and let my feelings out."

**GAINS**

Let her feelings out and find a listening ear when feeling down.

Recognise good things in life, and be more confident and content about herself.

Gain perspective and positive outlook

**PAINS**

Wanted to start a physical bullet journal to note down daily to-do lists, notes and feelings, but did not know if she would have time and did not want to waste money on materials.

Tried some journaling apps but did not like to type long posts and did not know what to write about.

Struggles with negative thinking, but feels paiseh to bother friends to talk about it.

Often compare herself with other people which creates anxiety.

Figure 2.1: User persona card for primary target audience

Please refer here for a higher resolution of the user persona card.

Social Media Marketing:

Since our target users are teenagers and young adults, we plan to promote our application through various leading social media platforms.

- YouTube

  YouTube is the leading video platform nowadays and it is becoming the go-to entertainment platform for many millennials. Including advertisements at the start or in the middle of the video tends not to lose the audience's attention, as people are incentivised by the video content that follows the ads.

  Moreover, we plan to collaborate with well-known lifestyle YouTubers with an audience base interested in mental well-being and self-care content (our target audience). Since the connection between a YouTuber and his/her audience is established over time, users are likely to try out our application if their favourite YouTuber recommends it.

- Telegram

  Telegram is one of the most-used messaging platforms among teenagers and young adults, and is loved for its wide variety of stickers. These stickers have become a huge part of Telegram conversations. We believe that our mascot, together with many other hand-drawn elements of our application, has the attributes required to become a popular sticker pack - fun, cute and expressive. Hence, we created a Telegram sticker pack to put ‘Halcyon’ into the minds of our target audience. Upon receiving these stickers from friends, people are enticed to find out more about the application behind these cute designs. We have also included stickers that allow people to share/recommend our application to friends in a fun and easy way. The QR code directs people to our web application, thus promoting our application to a wide range of potential users.

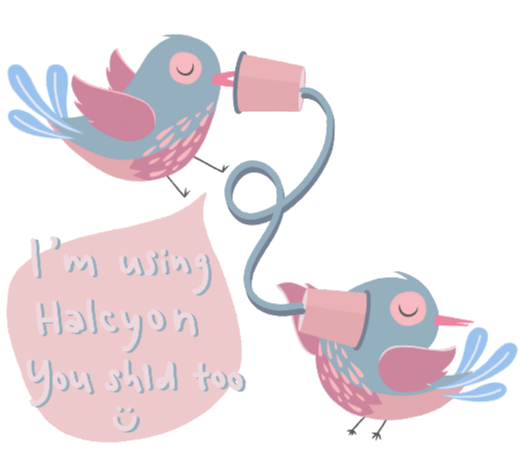


Figure 2.2: Sample stickers that allow easy sharing of Halcyon with friends

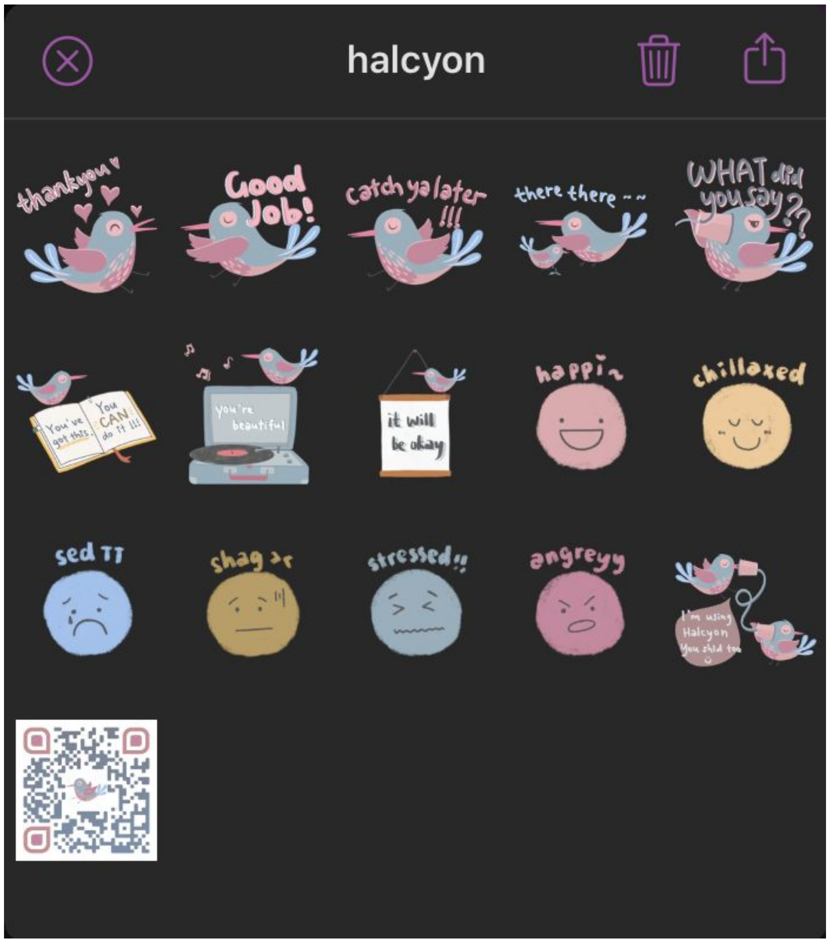


Figure 2.3: Telegram Sticker Pack

For a higher resolution version of the telegram sticker designs, please refer here.

## Zoom Conference

Video-conferencing platforms like Zoom have become more prominent than ever in light of the pandemic. We designed a Zoom background to complement the release of our app. It features the main elements of our app's interface and has a QR code with our mascot engraved into its centre. This background provides a warm and comforting atmosphere that aligns with our goal of focusing on self-care. It also entices users to explore the app by scanning the interesting looking QR code that is shown prominently at the top left corner.

We would also like to partner with counselling services to host self-care and mental wellness talks over Zoom, which ties in very nicely with our custom background. This partnership lends credibility to our app while simultaneously helping counselling services to raise awareness so that more people become interested in starting self-care routines.

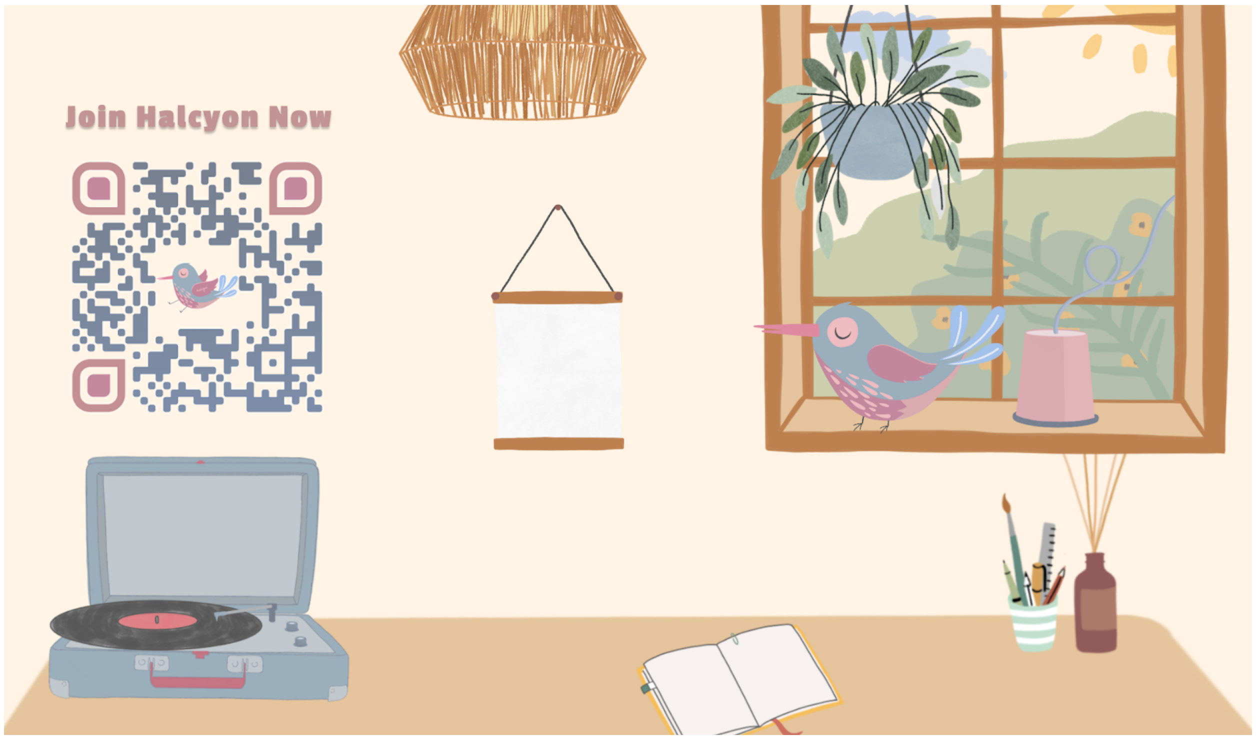


Figure 2.4: Zoom background

## Merchandising

Last but not least, since our application stands out from the competitors with its unique design, we plan to leverage this by merchandising our mascot. The bird makes for a cute soft toy, and many hand-drawn elements in our app can also be easily transformed into nice T-shirt designs. We intend to publicise our merchandise by uploading the designs to RedBubble. If there are customers indicating interests in our design, RedBubble will then proceed with production. Due to our lack of budget and our primary intention being promoting our application, we think RedBubble is the perfect platform for us to merchandise our designs and market our application at zero cost.

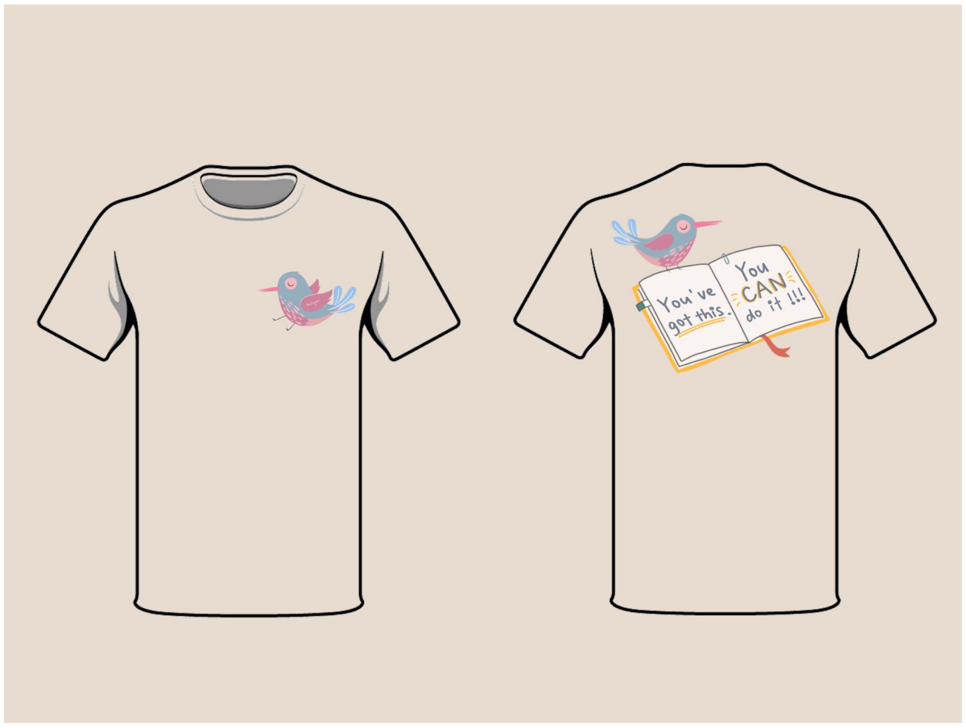


Figure 2.5: T-shirt design 1

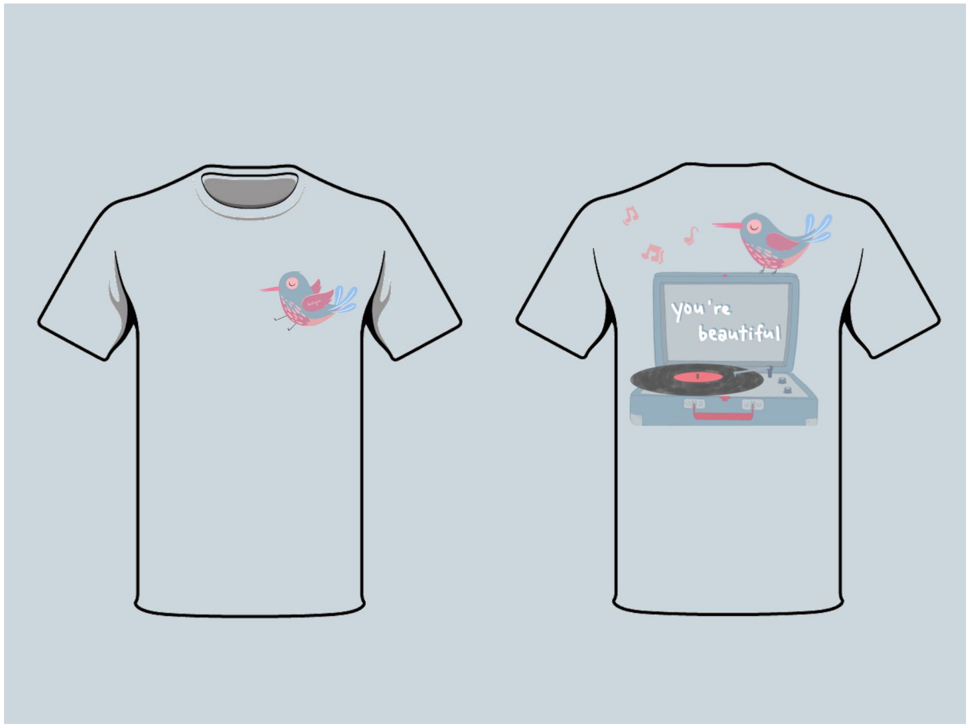


Figure 2.6: T-shirt design 2

For a higher resolution of the T-shirt designs, please refer here.

## Milestone 3: ER Diagram

Figure below shows the Entity-Relationship diagram for our database schema. Bold attributes make up the primary key of each table.

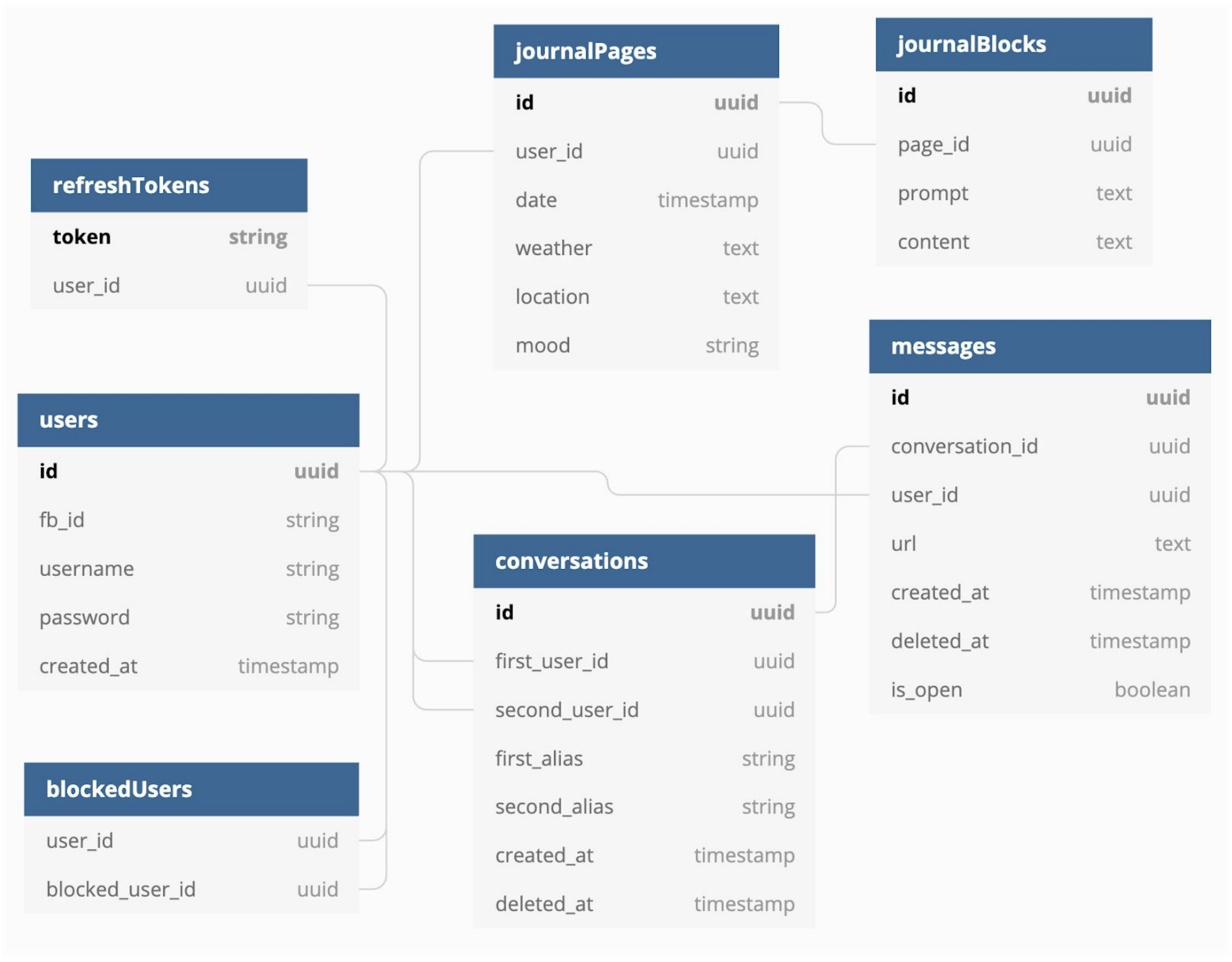


Figure 3.1: Entity-Relationship diagram for the database schema of Halcyon

The **users** table uniquely identifies all users. A user can sign up for a new account (username and password), use his/her Facebook account (fb_id), or use the app as a guest user. Each user is assigned a unique ID regardless of which sign up option is chosen. Guest users have their own user IDs and the users table will be updated if the guest user decides to register an account or log in with Facebook in future.

The **refreshTokens** table stores refresh tokens to allow users to remain logged in even after their access tokens expire since we use JSON web tokens for authentication. Refresh tokens are invalidated by deleting them from this table, such as when the user logs out.

To support our journal-related functionalities, we have two tables, **journalPages** and **journalBlocks**. There is a one-to-one relationship between journalPages and journalBlocks. They help to store the journal entries that users enter daily.

When a paper cup message is sent out by a user, a new entry will be created in the **messages**. The message will be marked as “opened” only when another user sends a reply. In the meanwhile, a new conversation will also be created with the two users’ IDs. Two aliases for the two users will also be generated to protect their privacy. We use the soft delete method for the deletion of both messages and conversations. If a user blocks another user, an entry will be added in the **blockedUsers** table. We implement a two-way blocking; both users will not receive messages from each other if one party blocks the other.

## Milestone 4: API Choice

REST vs GraphQL

| | REST | GraphQL |
|---|---|---|
| Network requests | Constrained by the API endpoints, so if data needed spans across multiple endpoints, multiple requests might need to be made. | Only one request needs to be made to one endpoint. |
| Over/under fetching | It's easy to fetch more data than needed, because each endpoint has a fixed data structure that it returns upon a request. Similarly, it's easy to fetch less data than needed for the same reason. | Declarative query language allows the client to fetch only what is needed, nothing more and nothing less. |
| Error handling | Easy to tell from HTTP status code and how to go about resolving it. | Response is always 200 OK, even if there is an error. Error messages are sent to the client as well inside the response. |
| Caching | HTTP already implements caching, and REST is implemented using HTTP. Thus the client can rely on HTTP caching to avoid refetching resources. | GraphQL has no caching mechanism (although this can be implemented with Apollo in-memory cache.) |

From the above pros and cons comparison, it would seem that GraphQL does indeed have many advantages over REST. However, we decided to go with REST for our app as not all of us were familiar with GraphQL and we needed to get the app up and running on a short timeline. In the context of our app, the benefits were not significant enough for us to switch to GraphQL. For example, our client does not need many endpoints to retrieve all the data it needs from the server, and it would have taken us a lot more time to study GraphQL and implement it correctly than to write a few more endpoints to serve the client's needs.

## Milestone 5: API Documentation

Link to Apiary: https://halcyon3.docs.apiary.io

The following paragraphs detail the ways in which our API conforms to the REST principles.

Uniform Interface: Resources are made available with intuitive URI names for ease of access. For instance, messages can be accessed via the “/message” route, while conversations can be accessed via the “/conversation” route. Links are provided to the associated URIs to fetch related information. For example, a random unopened message can be retrieved from the “message/randomUnopened” route. A common resource representation (JSON) is used for the response information. HTTP methods such as GET, POST, PUT, and DELETE offer the desired functionality when interacting with the server.

Client-Server: Our REST API conforms to a client-server architecture. The client displays information to the user and makes requests to the server, while the server stores data and manipulates information. API calls can be made as long as the request routes remain the same, which enables the client and server to evolve independently. Our front end only makes calls to the relevant routes of the back end server.

Stateless: The architecture which we have implemented is stateless and treats every request as new. It does not store information about prior requests. JSON web tokens are used to provide authentication information to the server. Refresh tokens are also available to enable users to remain signed in. Hence, all API calls are entirely stateless with authentication information being provided as part of a request.

Cacheable: Halcyon does caching on the client side to improve performance. This requirement is ever more important since Halcyon is a PWA. Caching at the client side provides the added benefit of enabling some degree of offline functionality, which is one of the core characteristics of a PWA. Furthermore, we do not handle complex data or make resource heavy computations on the server, which was another reason why we cached data on the client side.

Layered System: The API also makes use of a layered system. In particular, the server side code (NodeJS) and database (PostgreSQL) are run on separate Docker containers. The client has no knowledge of this and simply sends requests to the server via the

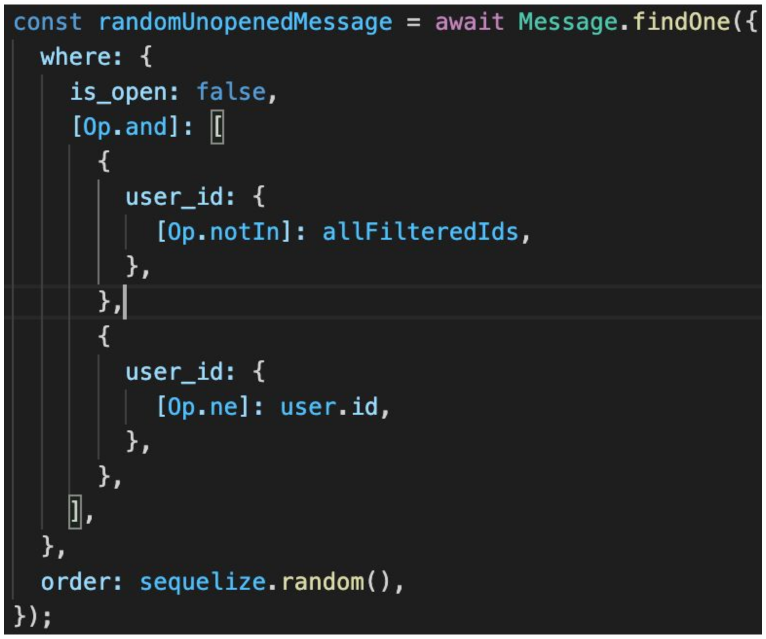
API. For example, when creating a new user, there is no way for the client to tell whether it is interacting directly with the database.

Code on Demand (optional): Not implemented since static resource representations are enough for Halcyon’s purposes. This constraint is also optional and is not needed for an architecture to conform to RESTful principles.

## Milestone 6: SQL Queries

1. When a user goes to "Paper Cup" and clicks on "I want to listen", the following query gets a random unopened message from the database. The message does not belong to any blocked user or the current user himself/herself.

   a. ORM query:



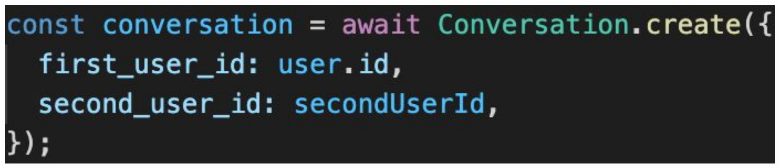
```
const randomUnopenedMessage = await Message.findOne({
  where: {
    is_open: false,
    [Op.and]: [
      {
        user_id: {
          [Op.notIn]: allFilteredIds,
        },
      },
      {
        user_id: {
          [Op.ne]: user.id,
        },
      },
    ],
  },
  order: sequelize.random(),
});
```

   b. SQL query:

```
SELECT "id",
       "is_open",
       "url",
       "createdat",
       "deletedat",
       "user_id",
       "conversation_id"
FROM   "messages" AS "messages"
WHERE  ( "messages"."deletedat" IS NULL
         AND ( ( "messages"."user_id" !=
'0fec09f4-be23-40e5-982c-24c5e4d75b1d'
               )
               AND "messages"."is_open" = false ) )
ORDER  BY Random()
LIMIT  1;
```

2. After a person listened to a paper cup message and sent out a reply, a new conversation involving two specified users will be created.

    a. ORM query:



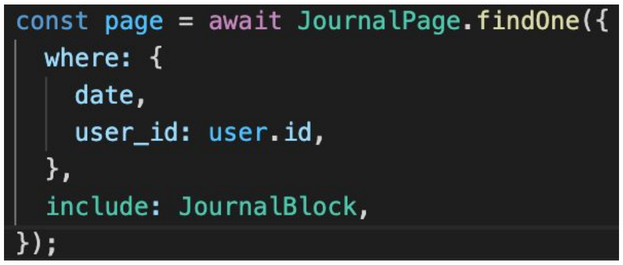
```
const conversation = await Conversation.create({
  first_user_id: user.id,
  second_user_id: secondUserId,
});
```

    b. SQL query (first_alias and second_alias are generated as part of the model instead of the route):

```
INSERT INTO "conversations"
            (
                        "id",
                        "first_alias",
                        "second_alias",
                        "createdAt",
                        "first_user_id",
                        "second_user_id"
            )
            VALUES
            (
                        $1,$2,$3,$4,$5,$6
            )
            returning "id",
            "first_alias",
            "second_alias",
            "createdAt",
            "deletedAt",
            "first_user_id",
            "second_user_id"
```

3. When a user views his/her past journal entries and clicks on a specific date. Information on the journal entry which was created on the specified date will be queried.

   a. ORM query:



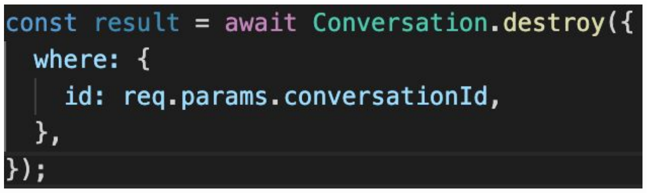
```
const page = await JournalPage.findOne({
  where: {
    date,
    user_id: user.id,
  },
  include: JournalBlock,
});
```

   b. SQL query:

```
SELECT "journalpages"."id",
       "journalpages"."weather",
       "journalpages"."location",
       "journalpages"."date",
       "journalpages"."mood",
       "journalpages"."user_id",
       "journalBlock"."id"      AS "journalBlock.id",
       "journalBlock"."prompt"  AS "journalBlock.prompt",
       "journalBlock"."content" AS "journalBlock.content",
       "journalBlock"."page_id" AS "journalBlock.page_id"
FROM   "journalpages" AS "journalPages"
       LEFT OUTER JOIN "journalblocks" AS "journalBlock"
                    ON "journalpages"."id" =
"journalBlock"."page_id"
WHERE  "journalpages"."date" = '2020-09-23'
       AND "journalpages"."user_id" =
'0fec09f4-be23-40e5-982c-24c5e4d75b1d'
LIMIT  1;
```

4. When a user decides to delete a conversation, the conversation will be marked as “deleted” in the database (soft delete method is used).

    a. ORM query:



```
const result = await Conversation.destroy({
  where: {
    id: req.params.conversationId,
  },
});
```

    b. SQL query:

```
UPDATE "conversations"
SET    "deletedat" = $1
WHERE  "deletedat" IS NULL
       AND "id" = $2
```

## Milestone 7: Icon & Splash Screen

For the icon of our application, we designed a bird with our application name on its wing. The bird also serves as the mascot for our application. This is because the word halcyon can either mean “a mythical bird” or means “calm and peaceful” as an adjective. For our splash screen design, we explicitly provided the definition of the word for users to understand the goals of our application.

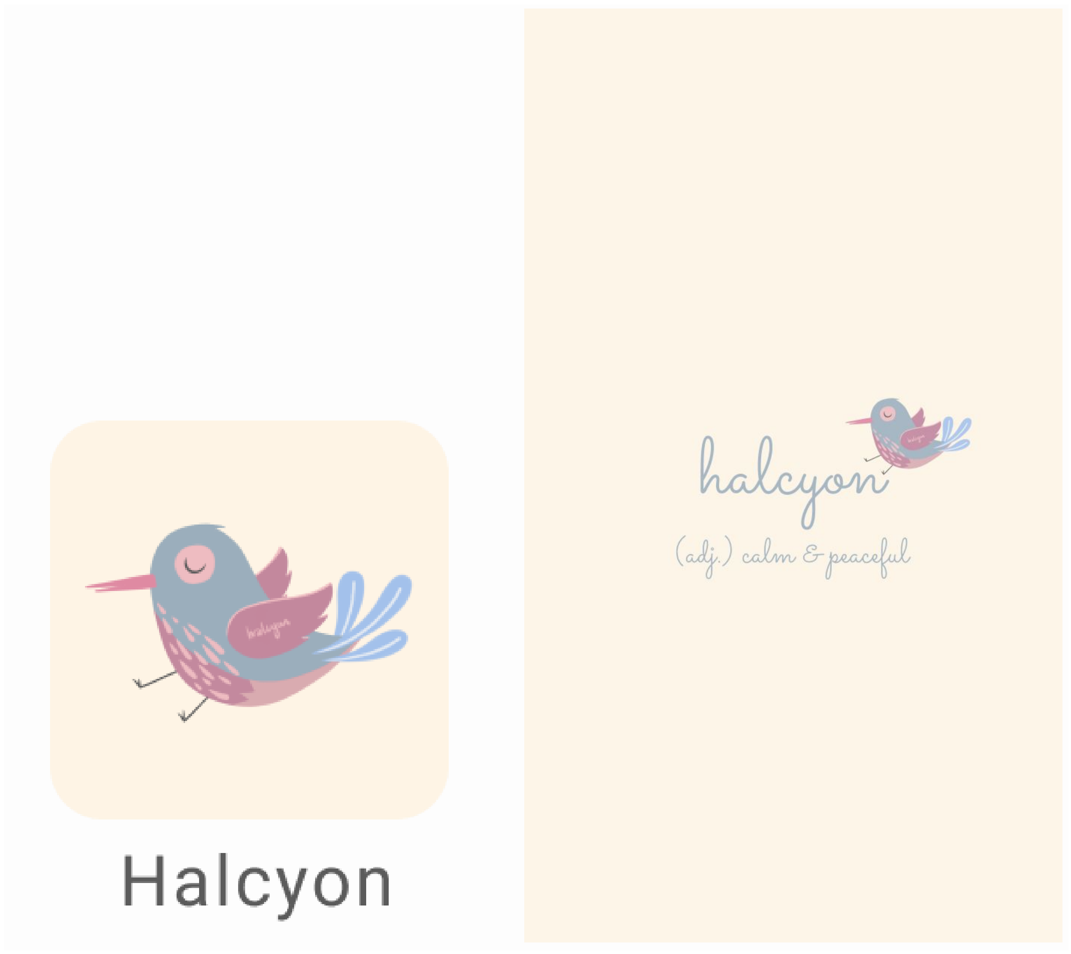


Figure 7.1: Home Screen Icon & Splash Screen Design

## Milestone 8: UI Design & CSS Methodology

We styled our UI components within the application by using a combination of global utility CSS helpers, CSS-in-JS styling via Emotion, and inline styles whenever necessary. The global utility CSS helpers help to easily adjust layouts so that we get standardized spacing with ease using predefined values for margin and padding. Using CSS-in-JS and inline styling works best for our application because it is highly stylized and many sets of CSS properties are only used in one component and never again. When we further consider animations that rely on state and are specific to only one component, it becomes much harder to handle it via pure CSS only.

We utilize Emotion, a library designed for writing CSS styles with Javascript for most of our custom container components, as it provides powerful and predictable style composition. On top of that, it allows us to develop rapidly, being able to create components that are for the most part standalone. When Emotion does not give us enough flexibility to create custom animations, we resort to writing inline styles. The benefits of separating CSS, markup, and logic are largely overshadowed by the greatly lowered cognitive workload required to create and update components. When we further consider that components are designed to be atomic in nature, i.e. being small enough and fulfilling only one purpose, the possibility of ending up with messy and unreadable code is minimized.

CSS frameworks like BEM and Object-Oriented CSS worked for an era before JSX - but it results in huge CSS files, redundant declarations, and unwieldy names over time. With JSX, instead of separating styling from function, they are commonly mixed together, which allows us to entirely package components as a standalone unit. With individual component styling, we can enforce an organization style such that styles from one component are isolated and cannot be referenced by another. Frameworks like BEM have mostly lost favour amongst the frontend crowd for good reasons - the isolated component CSS scopes are small enough such that having long CSS names does not make economical sense any more.

## Milestone 9: HTTPS

We deploy our client application on Netlify, which helps generate the SSL certificate via LetsEncrypt. Similarly, we deploy our backend server on Heroku, which does the same via DigiCert. The alternative to this is to generate our own SSL certificates, but since we are not an authorized Certificate Authority, our SSL certificates might be rejected by any browser worth their salt.

The 3 best practices for adopting HTTPS are:

1) Ensure the SSL certificate is obtained from a trusted Certificate Authority
2) Keep an eye on the expiry date of the SSL certificate and replace them in a timely manner
3) Use strong private key, such as a 2048-bit RSA key or 256-bit ECDSA key

Heroku and Netlify are trusted hosting providers which adopt all of the above best practices for generating the SSL certificates on their platforms.

## Milestone 10: Offline Functionalities

We utilized IndexedDB to synchronize the client with the server. Most of our data is stored in IndexedDB, and calls to the server simply update the entries locally. We operate via a cache-first strategy - first showing the user all cached entries while asynchronously performing a sync with the server. If the server returns any new data, the local cache is updated and the new data is repopulated on the client's screen. Using this method, we ensure that the client will always see something on the screen at the fastest time possible. Furthermore, using this method, we are able to queue actions from the user that only activate once the user is online.

We have taken careful care to ensure that the user is able to play with the application as a guest even if he is offline, provided that he has already pre-cached the entire application before signing in. The creation of the guest account will be delayed until he comes online. This fits the users expectations because the user might be offline during account creation, but still wants to try the application out.

On top of that, even when offline, the user has access to all features of journal writing and the vinyl player, as well as being able to re-listen to past conversations. The user should not feel any difference when using the application other than when trying to communicate with other users, since we have taken care to ensure that the offline experience is as close as possible to the online experience.

## Milestone 11: Choice of Authentication Scheme

| | Token-based | Session-based |
|---|---|---|
| Statelessness | Stateless server. Tokens are stored client-side and sent to the server upon every request for authentication. | Stateful server. The session data per user needs to be stored either 1) in memory on the backend or 2) in the database. |
| Scalability | No issues with scaling as the token is stored client side. | If storing sessions in memory, horizontal scaling will potentially be limited due to the need for session persistence across multiple domains. To ensure session persistence, there will be a need for load balancers configured to create sticky sessions by tracking the user (by IP or some other identification) to route all the user's requests to a specific server throughout the session. In short, there is a lot more work to be done for scaling session-based authentication. In addition, making use of the server memory means there will be issues when there are a large number of users using the system at once.<br><br>If storing sessions in a database, then there are issues with performance, which are described below. |
| Performance | The server only needs to verify the token signature, and is able to store information about the user (e.g. User ID, permissions, etc.) in the token, and thus retrieve it when verifying it upon a request. | If storing sessions in memory, then there will be issues with scalability, described above.<br><br>If storing sessions in a database, an additional look-up of the database has to be done to verify that the session ID sent by the client is valid, thus creating  a bottleneck. In addition, it adds an additional entity to maintain. |

| | | |
|---|---|---|
| Size | Tokens are very large. The amount of additional data stored on the token is thus limited as tokens have to be sent on every request (for protected routes). | Session is identified by its ID, which makes it very small. |
| Mobile-friendliness | Tokens are much easier to implement on both iOS and Android. Tokens are also easier to implement for Internet of Things applications and services that do not have a concept of a cookie store. | Sessions are not suited for mobile apps because mobile apps do not automatically maintain and send session cookies. |

In light of the many pros that tokens have over session, we decided to use token-based authentication. An especially important consideration is the fact that our app is a PWA, which essentially necessitated the use of token-based authentication due to the problems with cookies on mobile browsers, as described above. Even though tokens may be large in comparison to session IDs, the accompanying data in the token payload, as well as the size of our API requests parameters/query/body, is small, so there's no concern for being limited in that regard. Lastly, we were in favour of using tokens to store user information to avoid doing a one round trip to retrieve the user from the database using the session ID.

## Milestone 12: Choice of Framework & Mobile Site Design Principles

We wanted to build our application on top of React, so we shortlisted a few frameworks which provided React bindings for their components. We shortlisted the following frameworks: Framework7, Onsen UI, and Ionic.

Framework 7 gives us native-looking UI components out of the box with minimal configuration. It provided a large range of mobile-friendly components out of the box with great documentation all around. Their documentation site had plenty of demos, which would be very important during development. They also seemed to have all the features that we desired, like platform-specific components and easily overridable theming.

On the other hand, Onsen UI's documentation was largely lacking and seemed outdated. Its last major version update was 2 years ago, and their examples looked outdated and unpolished. It seemed like they only provided a few mobile friendly components out of the box, requiring the developer to handcraft most of the components. As such, we ruled out Onsen UI quickly.

Ionic framework looked beautiful and had many pre-built components, but it seemed to be overkill for our project. It aimed to build for web, native desktop, as well as native iOS and Android applications. Due to how much it promised, it would also require us to learn new tools that were specific to Ionic, as well as learn all its intricacies which would not be transferable to other frameworks. As we were only aiming to develop a PWA, we ruled out Ionic framework as well.

As such, we ended up using Framework7 as the base for our application.

Some of the mobile site design principles that have been demonstrated by our application are:

1. Keep calls to action front and center
   As seen from Figure 12.1, the call to action buttons, such as "Sign up" and "Log in" are put in prominent positions of the page.

2. Ensure enough space between buttons for finger tapping
   We have intentionally made the button big enough and left space in between for easy tapping.

3. Let users explore before they commit
   We allow users to proceed as guest users so that they can explore our application and use most of our basic functionalities before signing up.

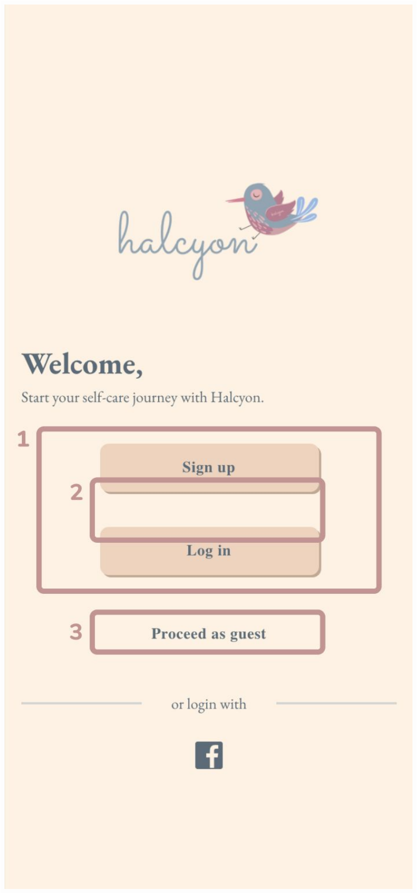
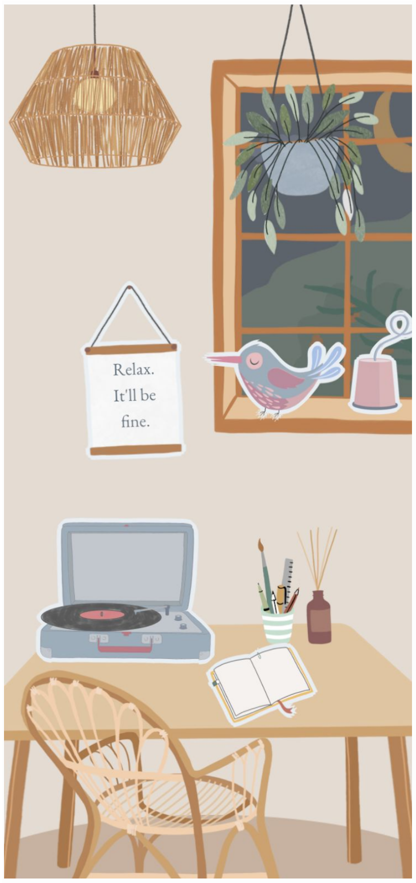


Figure 12.1: Screenshot demonstrating principles 1, 2 and 3

4. Optimise the entire site for mobile (no horizontal scrolling)
   We used a responsive layout for different screen sizes. Figure 12.2 shows our application on phone (iPhone X) and figure 12.3 shows our application on tablet (iPad Air).

5. Don't make users pinch-to-zoom (appropriately sized UI elements)
   All UI elements are appropriately sized and no zooming in and out are required for the entire user experience with our application.



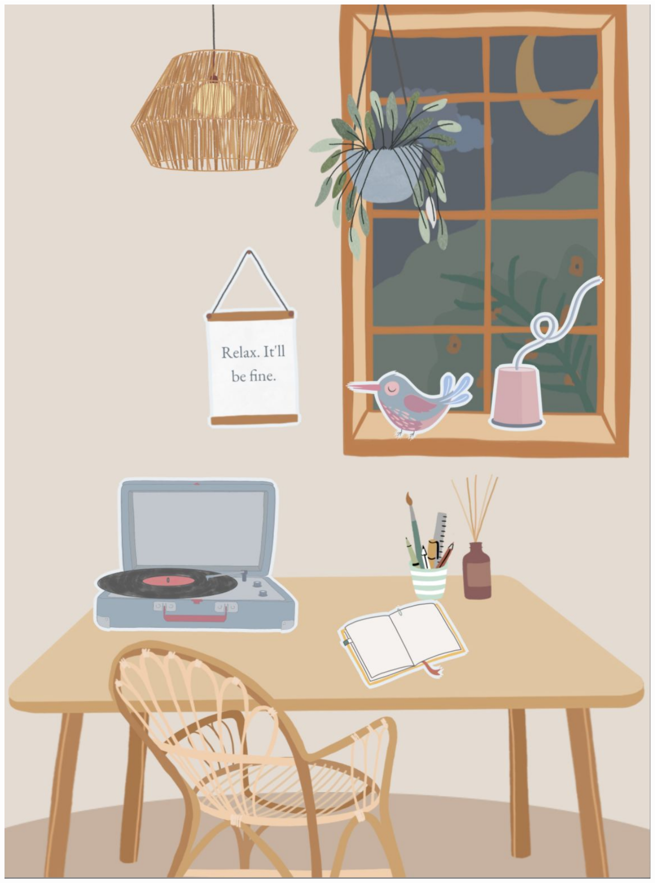


Figure 12.2 (Left) and 12.3 (Right): Screenshots demonstrating principles 4 and 5

6. Keep users in a single browser window
   Our application does not launch any new windows. Everything is rendered in one single browser window.

7. Provide visual calendar for date selection
   We provide a calendar view for past journal entries. Users can easily navigate between months to find his/her past journal on a particular day. No reference to another calendar app is needed.

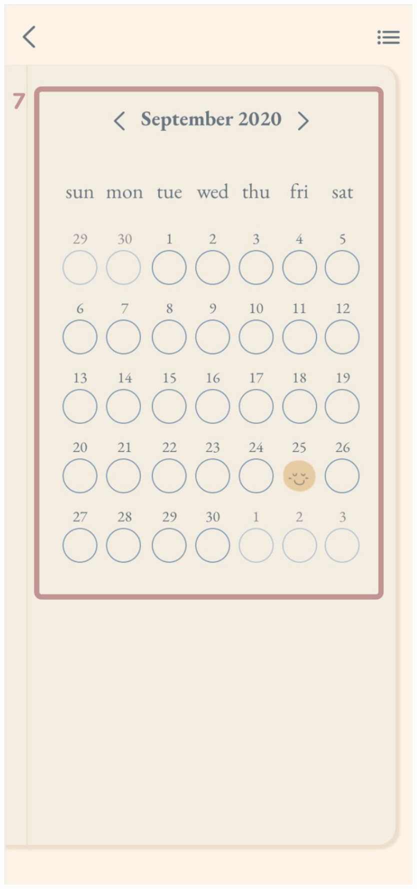


Figure 12.4: Screenshot demonstrating principle 7

## Milestone 13: Main User Workflows

Workflow 1: Complete a Journal Entry

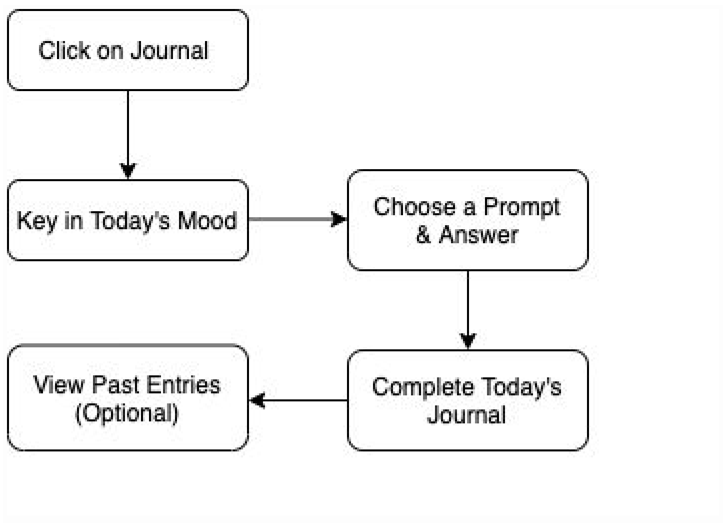
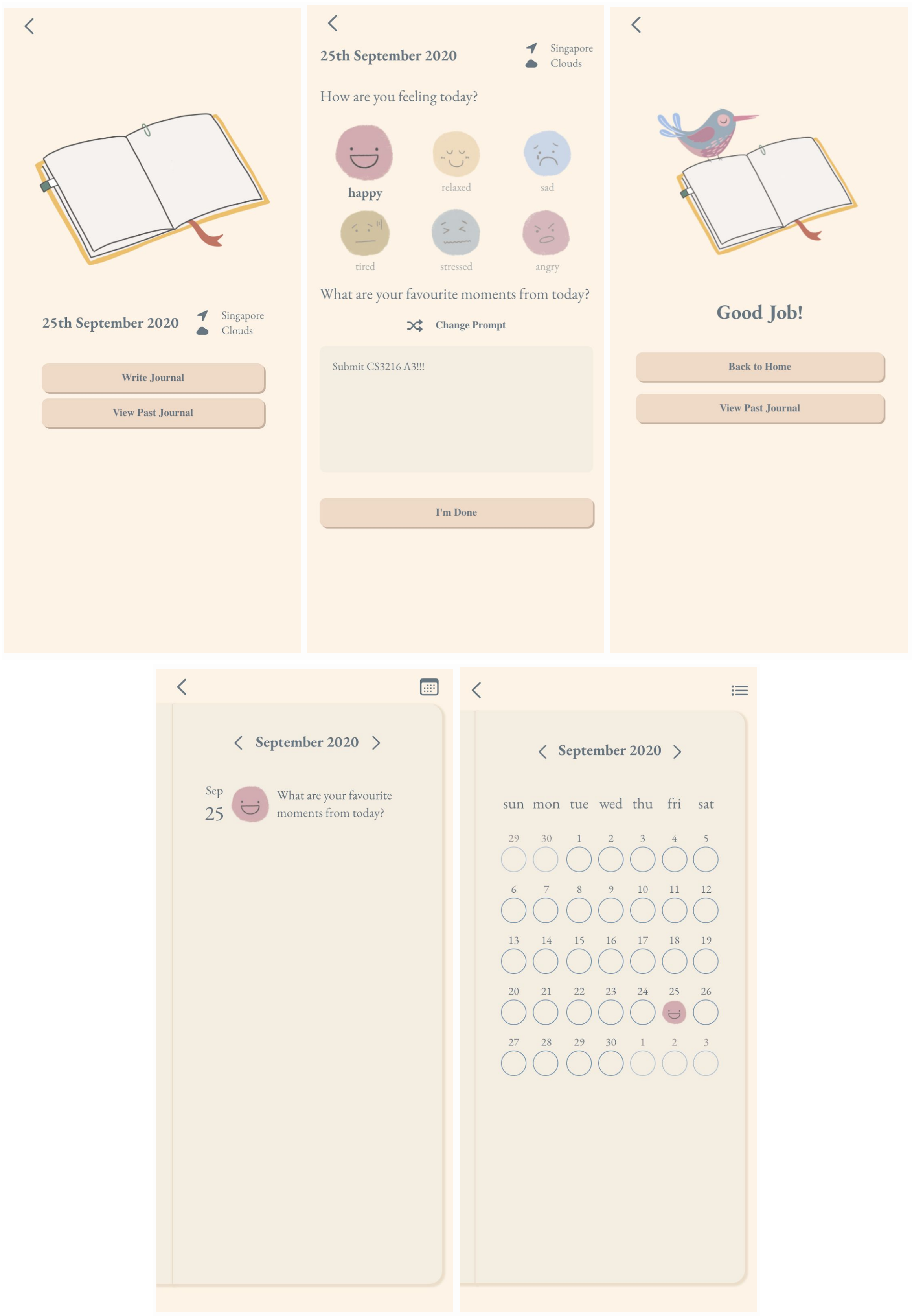


Figure 13.1: Completing a journal entry workflow diagram

Journaling is one of the key features of our application. Through this app, we aim to encourage the habit of daily journaling in users and to practice gratitude in day-to-day life. Therefore, "entering a journal entry" is expected to be a common workflow that users are likely to perform on a daily basis.

Mood tracking is a positive psychology technique for improving mental health and it helps people identify patterns in how their mood varies. Therefore, we prompt the users to enter their mood of the day before moving on to the journal entry.

One problem which many people faced while trying to start journaling is that they did not know what to write about or they did not have the time to write long paragraphs on a daily basis. Therefore, we used a prompt journaling method in our application. We asked the users one question a day and users can change the question if they do not like the question. The question prompts are all related to themes of gratitude, which encourage the users to think of the good things in their lives, to gain perspective and have a positive outlook.



How are you feeling today?

What are your favourite moments from today?

Submit CS3216 A3!!!

What are your favourite moments from today?

Figure 13.2: Screenshots showing workflow 1

Workflow 2: Send a Paper Cup Message

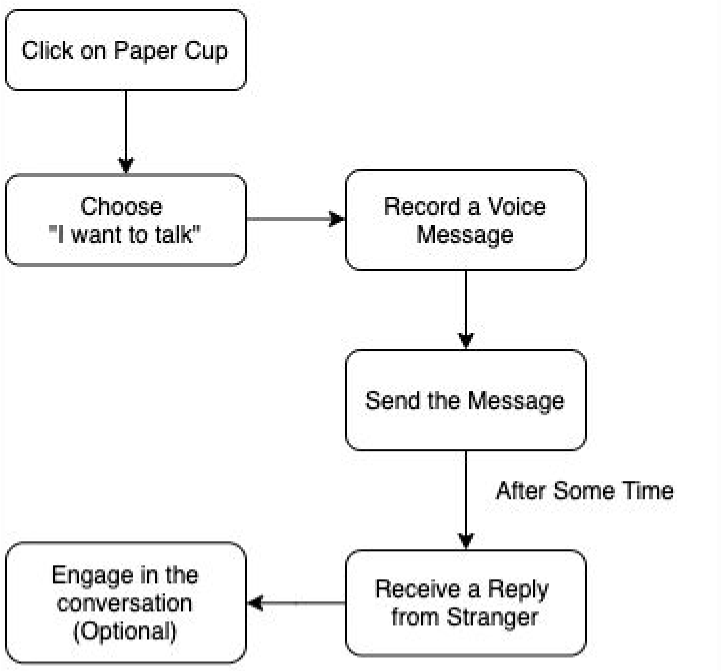


Figure 13.3: Sending a paper cup message workflow diagram

Our application provides a channel for users to let their feelings out on days when they feel down or lonely. They can send anonymous voice messages and may receive a reply from someone elsewhere in the world. We believe that anonymity is of utmost importance with regards to this form of communication in order to create a safe space for users to vent freely.

A more conventional workflow would be to let users converse with their friends or people they know, much like a social network. We decided against this because then our app would not be any different from other social media platforms. We would instead like users to remain anonymous so that they can speak their minds freely.

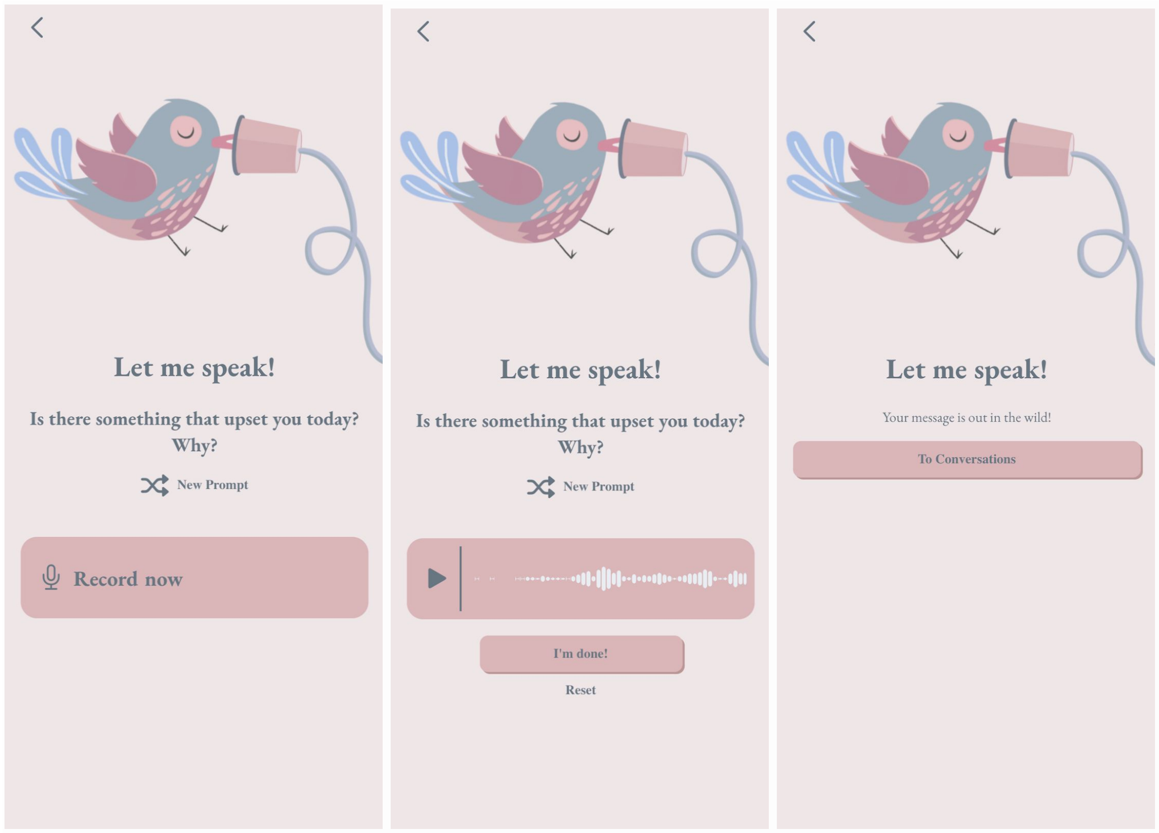

Let me speak!
Is there something that upset you today? Why?
New Prompt
Record now
Let me speak!
Is there something that upset you today? Why?
New Prompt
I'm done!
Reset
Let me speak!
Your message is out in the wild!
To Conversations

Figure 13.4: Screenshots showing workflow 2

Workflow 3: Listen & Reply to Random Paper Cup Message

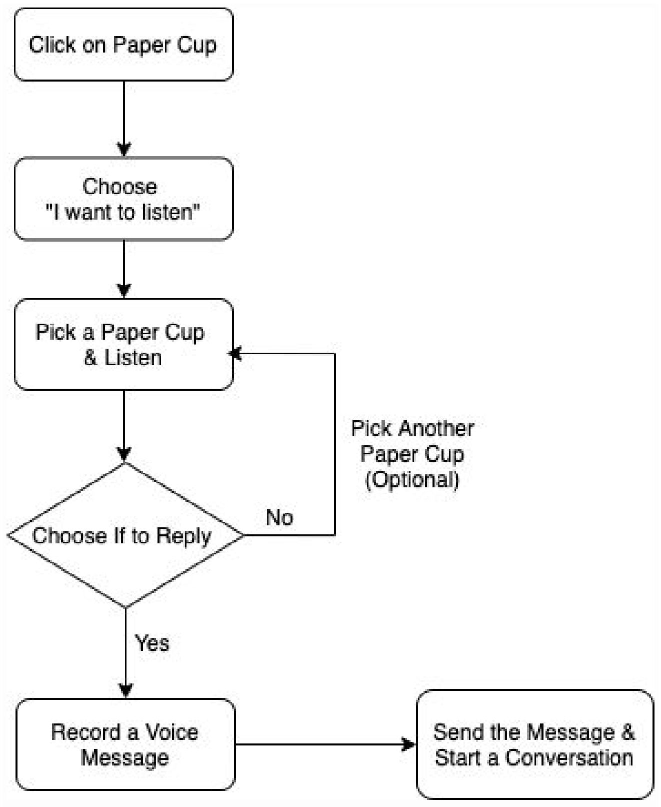

Click on Paper Cup
Choose "I want to listen"
Pick a Paper Cup & Listen
Pick Another Paper Cup (Optional)
No
Choose If to Reply
Yes
Record a Voice Message
Send the Message & Start a Conversation

Figure 13.5: Listening & replying to a random paper cup message workflow diagram

Just as users have the freedom to upload any content of their choice within voice messages, listeners can also choose whether or not to engage in a conversation based on the random paper cup message which they receive. An alternative workflow would be to automatically start a conversation upon listening to a random message from the cloud. We did not go with this because both parties should give their consent before engaging in a conversation. Of course, the user who sends the initial voice message has implicitly consented to let others listen to it. The user who receives the message may decide that the content is inappropriate or may not know how to respond to it. It is only right to let users decide whether to initiate a conversation. Hence, this strengthens our position on going with the workflow that we chose.

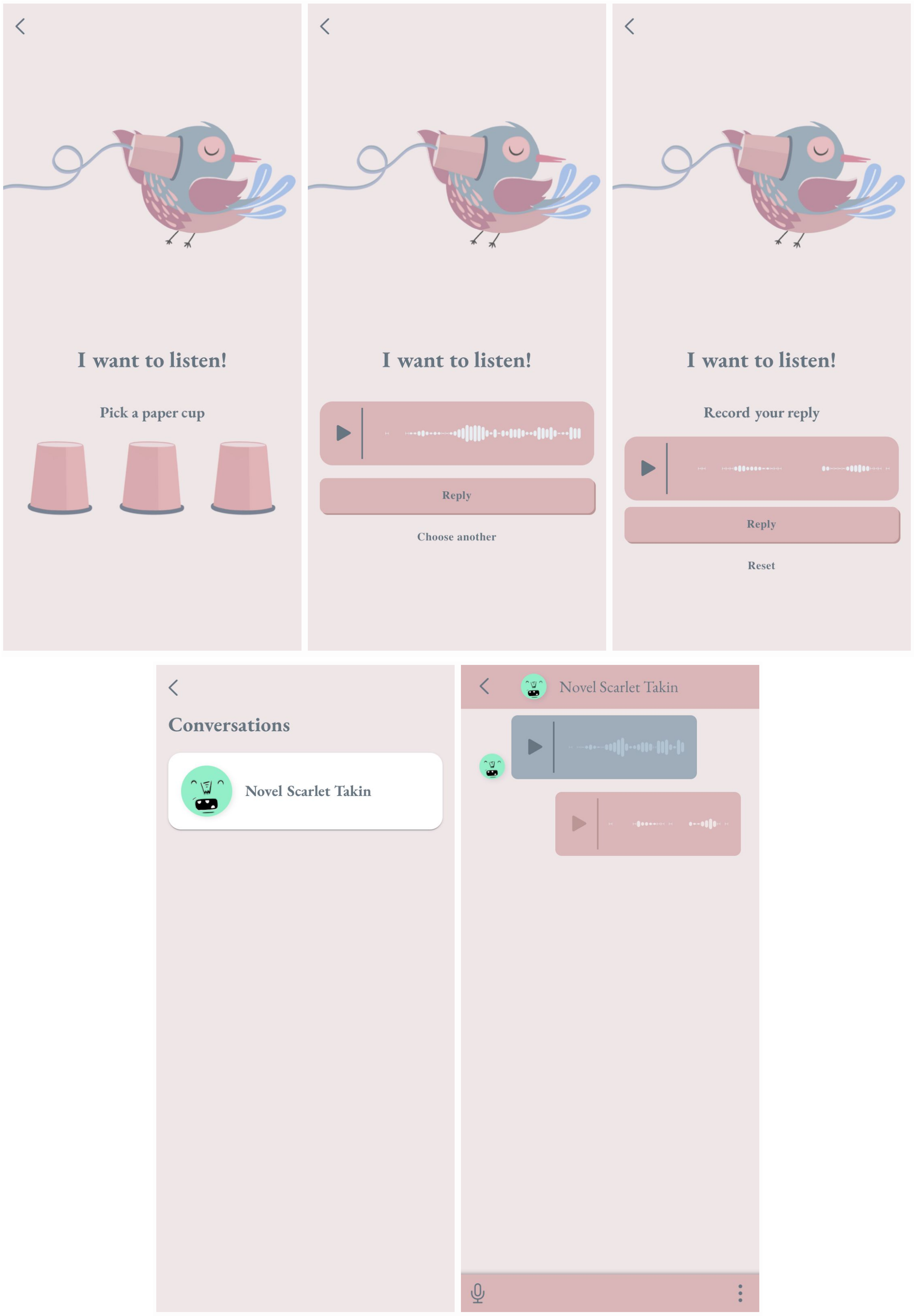
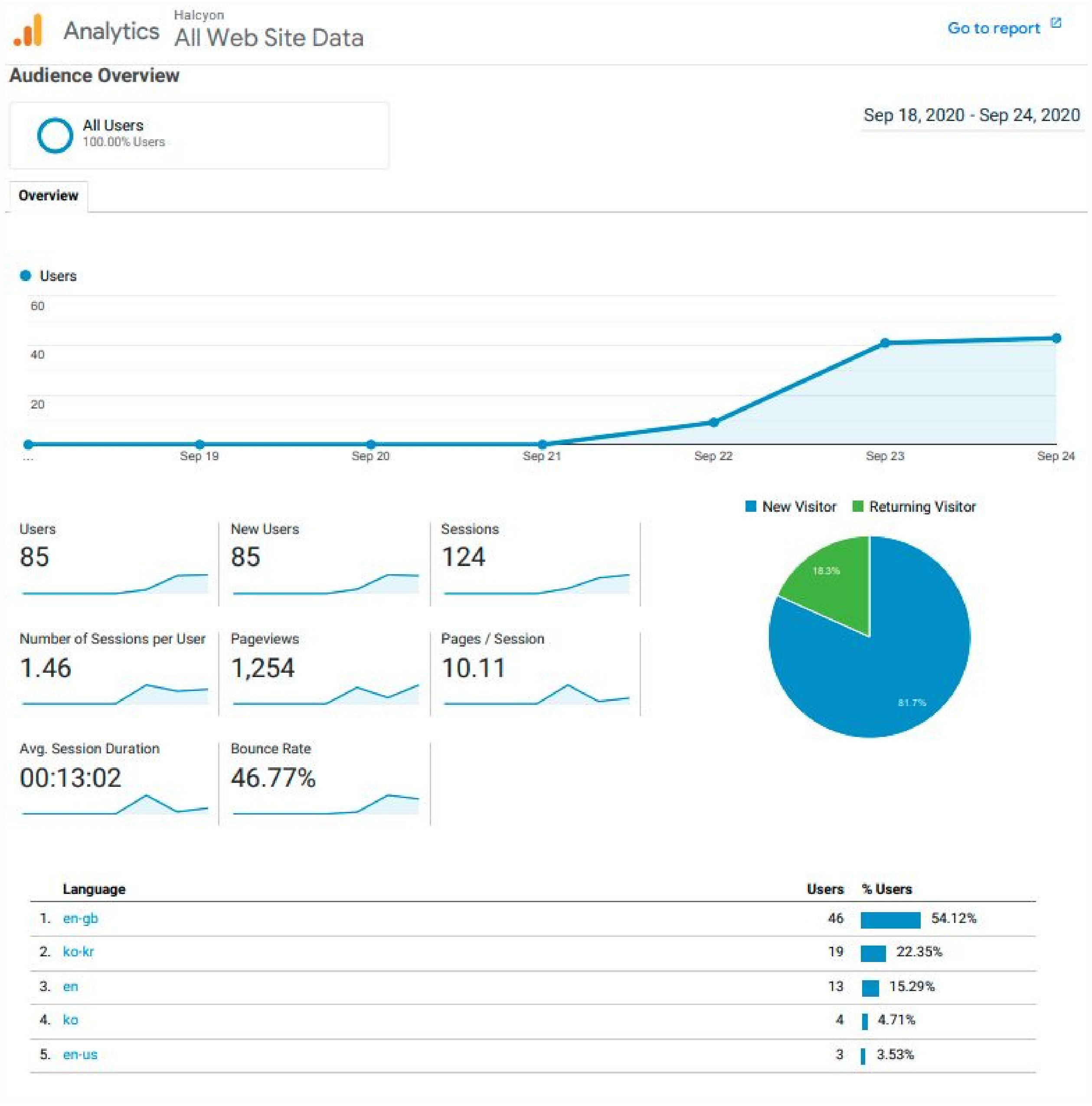


Figure 13.6: Screenshots showing workflow 3

## Milestone 14: Google Analytics



| Users | New Users | Sessions |
|---|---|---|
| 85 | 85 | 124 |

| Number of Sessions per User | Pageviews | Pages / Session |
|---|---|---|
| 1.46 | 1,254 | 10.11 |

| Avg. Session Duration | Bounce Rate |
|---|---|
| 00:13:02 | 46.77% |

| | Language | Users | % Users |
|---|---|---|---|
| 1. | en-gb | 46 | 54.12% |
| 2. | ko-kr | 19 | 22.35% |
| 3. | en | 13 | 15.29% |
| 4. | ko | 4 | 4.71% |
| 5. | en-us | 3 | 3.53% |

Figure 14: Google Analytics Report

For a higher resolution version of the Google Analytics Report, see here.

## Milestone 15: Lighthouse Report

Lighthouse html report can be found at our GitHub repository.

## Milestone 16: Social Plugins

We use Facebook login to provide users an alternative login method to expedite the login process. By doing so, we hope to improve the onboarding user experience for users who may not be willing or patient enough to create a new account on our app.

Why did we choose not to include social media sharing features?

We decided to not make use of Facebook sharing in our app as we believe it does not add value to our target users, who come to relieve their stress and care for their own mental wellbeing in a separate space. Research has shown that social media is a contributing factor to the stress that people face[2], which is a disincentive for us to incorporate such sharing features of social media into our app.

## Milestone 17: Geolocation API

We use the Geolocation API in our application to find out where the user is as an additional parameter to a journal entry. It does not have much other use in our application at the moment, but a possible feature for expansion would be a world map with pins on the places that the user has visited and wrote a diary entry at.

[2] The Facebook Effect: How Is Social Media Impacting Your Stress Levels? https://www.chestercountyhospital.org/news/health-eliving-blog/2020/march/how-is-social-media-impacting-your-stress-levels