Application website:

https://findaloo.netlify.app

Milestone 0:

Need to find a toilet, but not sure where the nearest one is? Imagine going from building to building, circling around, being unsure of where to look for one. Or perhaps you're a parent, with twins in tow, desperately in need of a baby change station. Where *can* you go?

With findaloo, these questions will never go unanswered ever again! Whether you're a parent in need of nursing facilities, an individual in need of a toilet, or just want to find the nicest throne around to grace your rear ends, findaloo is the app for you!

Access us easily. Simply go to findaloo.netlify.app. No app download needed at all! No more wasting those precious gigabytes of data on downloading an app.

View a list of buildings and all the toilets in them, all in a handy map view. Find the ones nearest to you with just a tap. Check to see if the toilets have the features you need - and even browse through a list of photos to see what they're like. Baby change station? Hot water dispenser? Seat bidets? We'll tell you.

Or how about the basics? Is the toilet free? Are there handicap toilets? Or do the toilets alternate by gender on every floor? These might not seem like much, but in dire circumstances, they may be the factor that determines relief... or sheer embarrassment.

And that's not all! Had a bad experience with a particular toilet? Not cleaned frequently enough? Leave a review to help others know. Or maybe you're trying to wash your hands... when you found out all the automatic taps are broken. File a report in our app, and we'll notify those in charge.

So what are you waiting for? Take care of your *business* today, and save findaloo to your home screen!

Milestone 1:

The benefits of creating findaloo as a mobile cloud application are multifold. Firstly, the mobile part. findaloo's effectiveness relies heavily on the availability of location data from our users. Unlike computers, which usually rely on GeoIP to retrieve approximate location data that is highly inaccurate, a mobile cloud application is able to obtain a user's precise location using the GPS chip and other sensors located in the user's phone. This means that we are able to give our users the best experience possible by recommending the best toilets that are the nearest to them.

Additionally, because findaloo is a mobile app, it is always there for you, whenever you need it - because your smartphone is almost always with you, too. In this world where many of us cannot live without our devices, this means that our users will never miss a chance to use findaloo when they need it.

When it comes to the cloud, being a cloud application means we are able to provide features such as reviews and reports for our user remotely. Our application saves the review data on behalf of our users, and can process that as a rating for each toilet. Additionally, reports that are sent to our server can also possibly be viewed by building management, for them to respond quickly to any faults within their premises. All this means that findaloo can be the one-stop solution for both users and building managements alike to provide feedback for and manage their facilities.

Being on the cloud also means that our users have access to the latest and most accurate data anytime, whenever they want. Gone are the days of having to download a new version of the application containing the latest information, or having to update the data within the application on each launch. The latest information is always available on our servers. This also gives us the added advantage of being able to dynamically update and add new information, based on our user's needs and feedback.

Lastly, a progressive web application allows our users to easily access our application. Firstly, no download is required - the user simply has to visit the webpage in their browser. Afraid that you won't have enough data left after downloading an app? No more looking around in futility for free WiFi, or waiting till you get home before you download our app - just visit our webpage and that's it! No longer will we have to battle the inertia of having to download yet another application: our users can get the information they want, whenever they want. It's also so easy to share findaloo with your friends - simply copy the link and paste it into WhatsApp/Telegram, and that's it!

Afraid that you'll forget our website link, or just want a more convenient way to access findaloo? Save our application to your home screen, and it'll look almost like an application

you've downloaded - without the downloading part. With findaloo's progressive web app, finding the nearest toilet is always swift, seamless, and easy, like it should be.

A progressive web application also means that we can stay relevant with our user's feedback and demands. Unlike conventional mobile applications, which require updates in order to change the user interface or user experience, we can push new features, or do quick redesigns to optimise new features easily. We can even easily add new data to be displayed, to make it more useful. This ties in with our earlier point of data storage on the cloud, and means that it is easy for us to react and adapt to the varying needs of our users, no matter who they may be.

Milestone 2:

To promote findaloo, we have identified three primary groups of target users:

1. Drivers (especially full-time drivers such as taxi drivers, PHV drivers, bus drivers)

Full-time drivers spend long hours on the road each day fulfilling their various responsibilities in logistics and transportation. This means that opportunities to take a break and use the restroom are far and few between. While depots and petrol stations offer these facilities, they may be far and few between, or occur only before and after the task has been completed. Additionally, they are far from the nicest toilets to use, with many of them being wet, smelly or dirty.

With findaloo, drivers and their passengers can find the best toilet facilities nearby to stop at and take a clean and comfortable break at, and even check if facilities such as a nursing room are available should it be needed. This can also be useful for families taking road trips.

To target this audience, we plan to give out free windscreen decals with a QR code that links to our application on both sides of the decal. That way, it provides free advertising to those who may be walking past these vehicles, while also providing a handy way for the drivers and their passengers to access the website. Furthermore, such windscreen decals are very affordable, with prices as low as 5-10c each when mass produced.

2. Parents

It can often be a hassle to take care of infants outside home, however, it is also too risky to leave them at home unattended. Parents therefore require access to a variety of facilities when out and about with their children. Many shopping malls provide such facilities in the form of a nursing room. However, with their high frequency of usage, it is not uncommon to find soiled or even broken items within the nursing room.

With findaloo, parents can look for the most suitable nursing room near them, with the facilities they may require, such as hot water dispensers. If they find that the room is dirty or various facilities are in need of repair, they can also file a report within the application to inform building management to have it fixed soon.

To target this audience, we plan to place posters on the doors of and inside nursing rooms. This will ensure that parents who are using the nursing facilities will be able to see our posters and check it out at their convenience. Additionally, by placing a poster on the door as well, parents who are waiting for the nursing room to vacate can be led to scan the QR code and find an alternative nursing room with findaloo.

3. Anyone who just wants a clean, nice toilet.

No one can deny the pleasant experience of walking into a clean and dry toilet filled with pleasant fragrances. For all who simply want to have a better time when outside, whether it's before or after meals, when studying at a cafe, or even just going for a walk, findaloo can help them spend their time in the loo in a classier manner. No matter who you might be, dirty, smelly toilets are always a turn-off, and findaloo can help you keep your sanity by finding the nicest toilet around you.

To reach out to this audience, we can give out stickers that say 'Your world doesn't have to stink. Pamper yourself with a nice toilet today.' with a QR code to findaloo. This lets the user decide where he wants to keep the sticker, which serves as a reminder of our application. Additionally, stickers are very cheap to print, and can be distributed en masse.

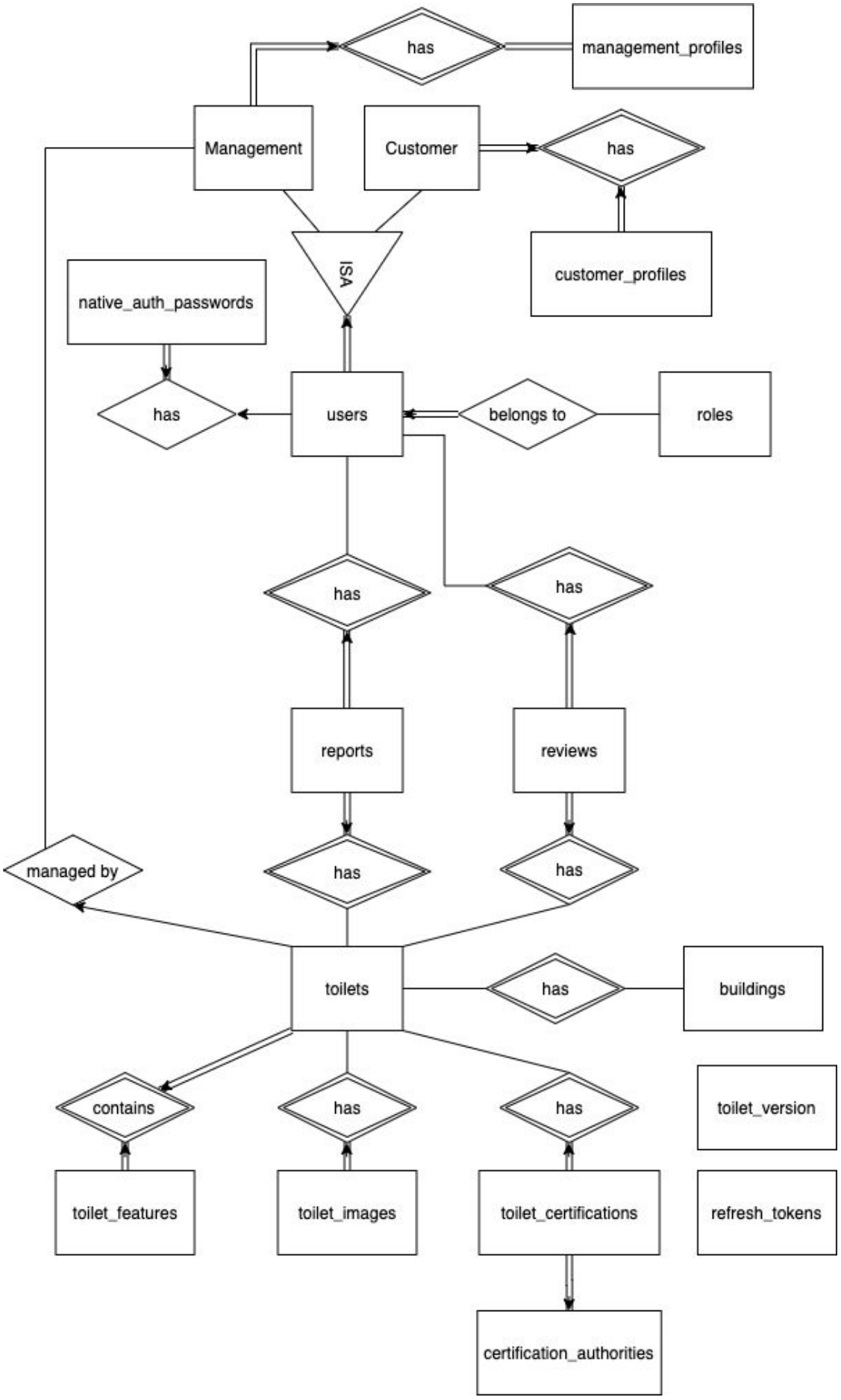
Milestone 3:

Our ER diagrams were designed on draw.io. Due to the limitations of a PDF file, should you be unable to see the diagrams clearly, please visit the following link to view the original diagram.

https://drive.google.com/drive/folders/1DBCq0xKPYfR2_znsiTieMi3OYKvGWh8s?usp=sharing

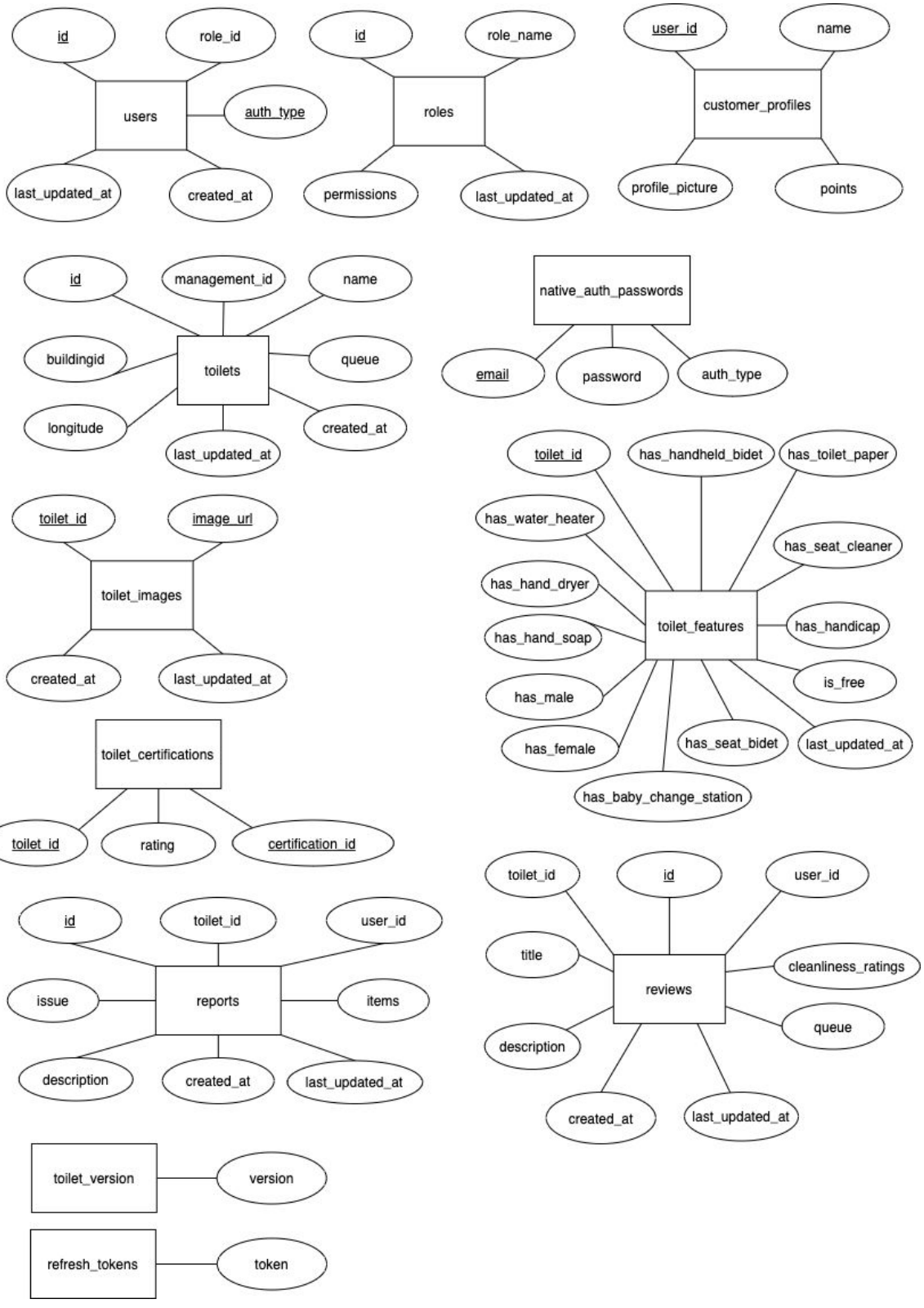
High-Level ER Diagram

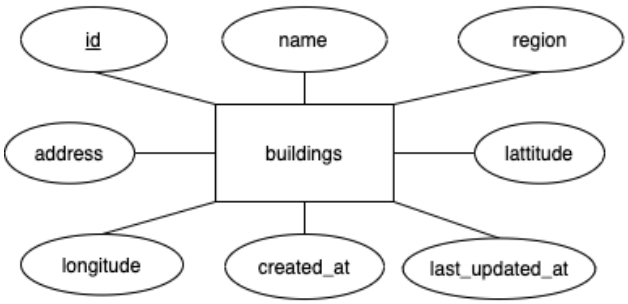
The following ER diagram depicts the high-level database design of our application, omitting the individual attributes.

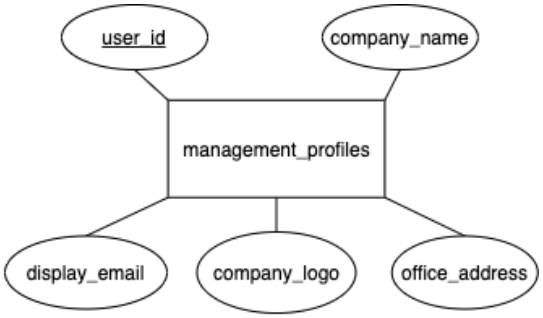

has
management_profiles
Management
Customer
has
ISA
customer_profiles
native_auth_passwords
has
users
belongs to
roles
has
has
reports
reviews
managed by
has
has
toilets
has
buildings
contains
has
has
toilet_version
toilet_features
toilet_images
toilet_certifications
refresh_tokens
certification_authorities

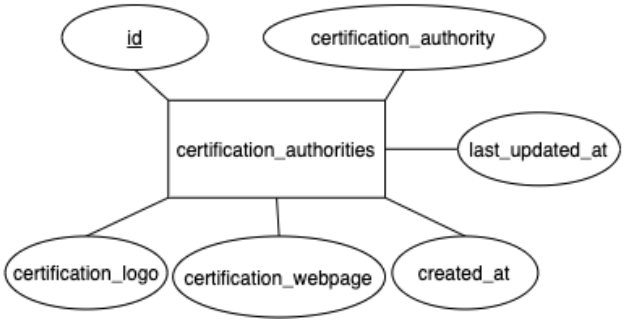
Entity ER Diagram

The following diagrams depict the ER diagram for each entity.




id
name
region
address
buildings
lattitude
longitude
created_at
last_updated_at


user_id
company_name
management_profiles
display_email
company_logo
office_address


id
certification_authority
certification_authorities
last_updated_at
certification_logo
certification_webpage
created_at

Milestone 4:

One alternative to the REST API that we explored was GraphQL. Both are similar in their approach, in that they both supply a route to retrieve a resource. They also allow for IDs to be supplied to dynamically select the resource to be returned, along with a request body, and both data to be sent in JSON format.

The primary difference between GraphQL and REST is the use of multiple endpoints, as compared to a single endpoint. What this means is that the client must specify what data to retrieve in a GraphQL query, as compared to a REST query, where the REST endpoint itself defines the data to be returned.

This has both advantages and disadvantages. In a typical REST query, the structure of the data to be returned is fixed. This can often lead to the oversupply or undersupply of data from an endpoint. GraphQL's ability to specify the data to be sent mitigates this issue.

However, in exchange for this, GraphQL can be much more difficult to implement due to the use of resolvers to map each data request to a corresponding function. This is compared to REST, where each endpoint is fixed to a function.

Additionally, while it allows for queries to be more 'involved', it also means that more complexity is involved in each query as the front-end must know what to query from the back-end.

REST's use of multiple endpoints also means that caching is done automatically, as compared to the single endpoint in GraphQL. This can result in significant performance gains, especially in large data sets. Additionally, error handling in REST is much simpler as specific errors can be sent for each route.

For our application, we chose REST for a few reasons. While GraphQL's data-specific requests would help us mitigate data oversupply issues in our app, the performance difference between querying all and querying part of the data was negligible when used in person, even with over 1000+ records being queried from our database at a single time. To further mitigate this issue, we provided different endpoints for querying all the available data, and querying one record of data. This meant that the application could decide which query was more suitable for use at a certain time, thus minimizing the possibility of oversupply when not needed.

Additionally, the majority of our team was already familiar with REST. In such a project where time is of the essence, having to learn an alternative to REST, and being unfamiliar

with its potential pitfalls could make or break the product. We thus chose to stay on the safe side with REST.

Milestone 5:

Our REST API is separated into two servers: one for authentication related tasks, and one providing the main API services for the application.

The authentication API server can be found at https://a3.dawo.me:4000/ and is documented on Apiary at https://findalooauthentication.docs.apiary.io/

The application API server can be found at https://a3.dawo.me:3000 and is documented on Apiary at https://findalooapi.docs.apiary.io/

With regards to REST principles, there are 5 main principles to follow (taken from https://restfulapi.net/). We have added an elaboration on how we fulfill these principles.

1. Client-server

Our design fulfills the client-server model. The client and the server are independent of each other and the interaction between the client and the server happens when the client calls the server and expects a result from the server. The server will return data back to the client only as a response to every request from the client. In addition, our server returns data in JSON format. This means that it is interchangeable with different client types, and is not specific to a particular client or framework.

2. Stateless

This principle is fulfilled as all routes are stateless. Request parameters are supplied either in the form of query parameters or in the request body. Additionally, users are identified and authenticated by decoding their BEARER token. This means that a route does not need a state, or previous data, to be able to fulfill a request, and is able to treat each request as an independent and complete transaction, thus fulfilling the stateless property.

3. Cacheable

Under the REST principles, each route is supposed to indicate its cacheability. However, in our application, we designed it such that all data retrieved from the backend is suitable for caching. We therefore skipped the implementation of the cache indicator in favour of caching everything.

To mitigate the potential for desynchronisation from the server, a versioning system was introduced for the client to quickly check if the data held locally is out of date. If it is, it will immediately retrieve the updated data on the server.

4. Uniform interface
   This principale mention that REST is defined by four interfaces, including:
   a. **Resource-based identification**
      Our server routes include resource identifiers as it is in the form of Uniform Resource Identifiers (URI). It follows the syntax of URI which consists of
      - Scheme or protocol, which is **https** in our case. The client and the server communicate using the HTTP protocol with added TLS security layer on top of it to increase the security of data transferred between the client and the server
      - Authority, which is **a3.dawo.me** in our case. It is an elastic IP address allocated to our server hosted in AWS.
      - Ports; there are 2 ports 3000, and 4000 that are hosted on our server. The port 4000 handles all authentication routing, including sign-up and login, while port 3000 hosts all of the routing except the authentication routes.
      - Paths; we have 18 paths in total, 11 paths are hosted in port 3000 and 7 paths are hosted in port 4000, these paths uniquely identifies the resources accessed.
      - Query parameters are used in some of the paths in our server, which functions to further specify the resource accessed. For example, in our '/toilets/search' path, a query parameter of 'keyword' is needed, as this keyword identifies the keyword to be searched. It filters the toilets according to the query parameter 'keyword'.
   b. **Management of resources through representations**
      The resources, which are the data are exchanged between client and server through representations which in this case is JSON format.

      The response from the server has enough information for the client to modify their resources. For example, our '/toilets' route returns information about all toilets. We return the data in a complete manner, including the building that a toilet belongs to and all of its features. This way, the client is able to display all buildings information, as well as displaying all toilets in a building. Also, the client is able to deduce the top 2 or 3 features that are present in a toilet from the response.
   c. **Self-descriptive messages**
      Each request-response from the client and the server are full and complete. Information provided in the request is enough and complete for the server and information provided in the response is enough and complete for the client.
   d. **Hypermedia as the Engine of Application State**
      The client and the server represent forms of media in hypermedia. For example, our '/toilets/{toiletId}' route returns a JSON representation. One of the keys of the JSON is 'imageUrls' which is an array of string URLs,

representing the link of the images of the toilet which is located in the server itself. This way, the client can retrieve the images of the toilet by accessing these URLs given by the server.

5. Layered system
   The client side is only aware of the immediate layer that it talks to, in this case, **a3.dawo.me**. This is an elastic IP address that can be allocated to different infrastructure, including proxy or load balancers, which can later then be connected to the actual server, without the client knowing.

   The client does not know whether it is talking to an actual server computer or proxy or load balancers because of the elastic IP. However, in our case, there is no layer in between the client and the server. The elastic IP is currently attached to the actual server because we assess that there is no need of implementing a proxy or load balancer in our system as the traffic to our system is still manageable by a single server.

Milestone 6:

This section will describe 3 example SQL queries used within our application.

1. /customer/profile - Retrieves the currently logged in user's profile

To reduce the complexity and frequency of querying within our application, SQL views are used to join relevant data together.

For example, the /customer/profile route, which returns the current logged-in user's profile details, uses the following SQL query:

```
SELECT *
    FROM CustomerSummary
    WHERE id = (${userId})
```

As can be seen from the above data, querying is made simple due to the underlying CustomerSummary view. The CustomerSummary view is as follows:

```
DROP VIEW IF EXISTS CustomerSummary;

CREATE VIEW CustomerSummary(id, name, email, auth_type,
profile_picture, points) AS (
    SELECT CP.user_id, CP.name, U.email, U.auth_type,
CP.profile_picture, CP.points
    FROM customer_profiles CP
    JOIN users U
    ON CP.user_id = U.id
)
```

2. Account creation

To ensure that a user's account is created properly, it is created by using a transaction. This ensures that all the necessary information is inserted before a confirmation is sent, and if it fails for any reason, the database will revert to a consistent state.

The following is the SQL query used to insert a new user.

```
BEGIN;

INSERT
    INTO users("role_id", "email", "auth_type")
    VALUES (${user.roleId}, ${user.email}, ${user.authType});

SELECT id
    FROM users
    ORDER BY id DESC
    LIMIT 1;

INSERT
    INTO customer_profiles("user_id", "name", "profile_picture")
    VALUES (${customerProfile.userId}, ${customerProfile.name},
${customerProfile.profilePicture})

INSERT
    INTO native_auth_passwords
    VALUES (${credentials.email}, ${credentials.authType},
${credentials.password})

COMMIT;
```

3. Toilet versioning

To allow the frontend to cache toilet data and avoid re-downloading it with every page load, the server stores a version number in toilet_version, which increments with each insert or update on any of the toilets table. The route to query the version can be found at /toilets/version.

This approach was used to save database queries, as a hash based upon actual data would result in a lot of computational power being used to calculate the hash on each query. This method enforces a version number that is inherently linked to the state of the database.

The query used to obtain the version number is shown below:

```
SELECT *
  FROM toilet_version;
```

To ensure that the version increments with each insert or update action, a PostgreSQL trigger is used. The trigger to enforce this is shown below:

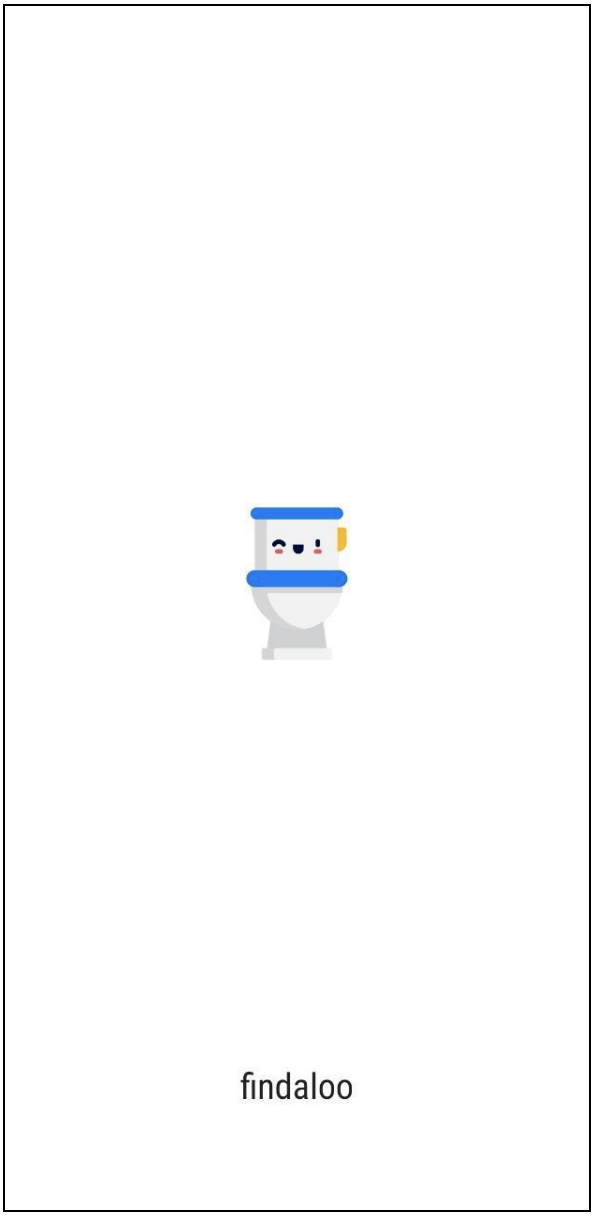
```
CREATE OR REPLACE FUNCTION increment_version() RETURNS TRIGGER AS $$

BEGIN
    UPDATE toilet_version
    SET version = version + 1;
    RETURN NULL;
END;

$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS increment_version_toilets ON toilets CASCADE;
CREATE TRIGGER increment_version_toilets
    AFTER INSERT OR UPDATE ON toilets
    EXECUTE FUNCTION increment_version();

DROP TRIGGER IF EXISTS increment_version_certifications ON
toilet_certifications CASCADE;
CREATE TRIGGER increment_version_certifications
    AFTER INSERT OR UPDATE ON toilet_certifications
    EXECUTE FUNCTION increment_version();

DROP TRIGGER IF EXISTS increment_version_featuress ON toilet_features
CASCADE;
CREATE TRIGGER increment_version_features
    AFTER INSERT OR UPDATE ON toilet_features
    EXECUTE FUNCTION increment_version();

DROP TRIGGER IF EXISTS increment_version_images ON toilet_images
CASCADE;
CREATE TRIGGER increment_version_images
    AFTER INSERT OR UPDATE ON toilet_images
    EXECUTE FUNCTION increment_version();
```

Milestone 7:

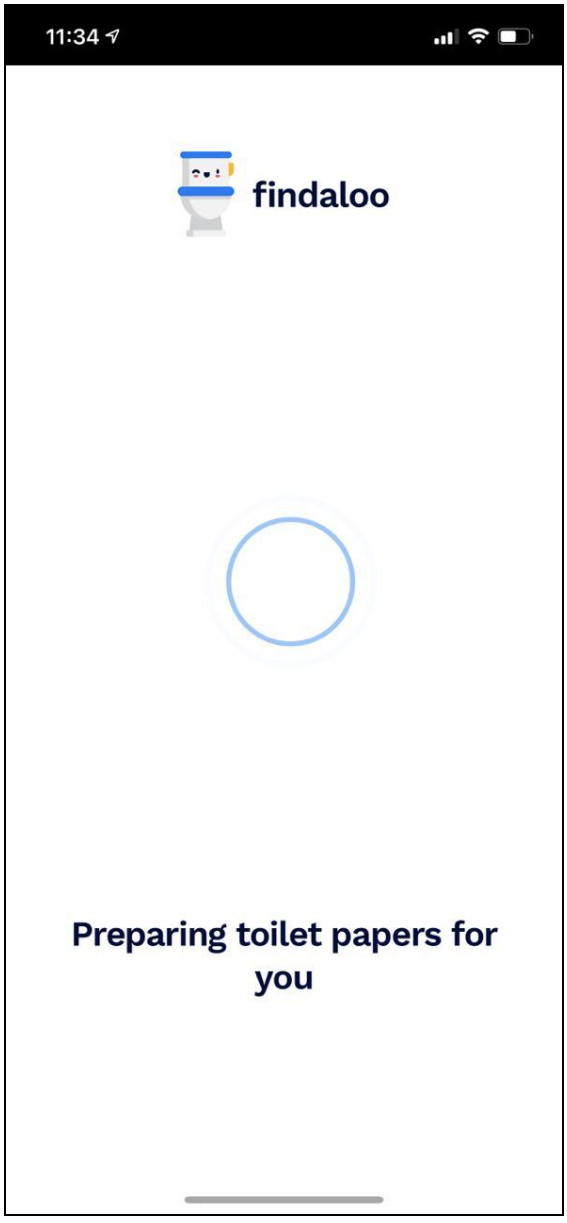
Icon design:

findaloo

Splash screen:



Loading screen:



Milestone 8:

UI Design

Our UX design aims to help users find toilets nearby that fit their needs in a short time. The explore page, which is the home page of our app, provides easy access to different functions of our app and thus saves users' time. The bottom sheet focuses the user's attention on what is relevant to them: places with toilets. The full-view persistent map allows information to be accessible from any part of the screen. The UI was also compacted, with no unnecessary bottom bar or hamburger menu to save screen real estate and provide as much information as possible.

The toilet markers allow users to search for toilets nearby fast and easily. In particular, we chose the light overall UI theme, the green toilet markers, the bright colours, and the spacious breathable UI give a "clean" and "simple" impression, which are expected factors when it comes to lavatories.

Clicking the search bar, a grid view of toilets overlaps the map view. Stateful interactions like this in the explore screen allows the user to follow the flow of information easily. Each card in the grid view displays essential information of a toilet to help users find the toilet that fits their needs. In particular, the compact card design also allows for more suggestions to be displayed. Users can easily go back to the map view by sliding down the grid view. At the bottom part of the map, users can see buildings or toilets nearby without any searching.

After logging in, the user can also go to the profile page by clicking on the profile picture in the search bar. The explore page includes all essential functions and saves users' time for searching toilets nearby.

The user can navigate to a toilet detail page by clicking on a toilet card. The toilet detail page displays photos and toilet information. Thus, the user can report any errors they found by clicking on the report button after seeing the details. Under the review tab, users can give reviews by clicking on the stars or scroll down to view other users' reviews. This allows the user to have a comprehensive understanding of the toilet status.

Object-Oriented CSS (OOCSS)

We adopted the OOCSS principle and separated the layout and skins attributes for reusable components. This separation allows for a consistent styling across the app and a maintainable and extensible code base. Furthermore, this abstraction reduces the possibility of conflicting or overriding CSS attributes designed by different team members, thus increasing the development's efficiency.

Milestone 9:

Three best practices for adopting HTTPS include:

1. **Eliminating mixed content**

This means that all application data should be retrieved by HTTPS, and not a mix of both HTTP and HTTPS. In findaloo's case, both the front- and back-end endpoints were deployed with HTTPS.

2. **Using secure cookies**

All cookies sent between the application and the server should be sent via HTTPS. This can be done by setting the `Secure` flag in the cookie. However, we did not use cookies in our application at all, instead opting for LocalStorage whenever needed.

3. **Evaluate third-party code**

Third-party code, such as Google Analytics, can be a vector for attack by malicious parties. In our app, we only use Google Fonts, Google Analytics, and Google Maps Platform APIs. While this may be a potential risk, these services' extensive development and adoption within various different types of applications render it unlikely to be insecure.

Milestone 10:

Because findaloo is reliant on the Google Maps Platform APIs, it is subject to the Google Maps Terms of Services (ToS), which prohibits us from saving any part of the map. This unfortunately means that the user will not be able to see the map view when offline, and will simply see a grey background in place of the map. Though, the toilets and past location markers are still available on the grey map, allowing the user to at least make approximations.

However, despite this limitation, we have managed to implement quite a number of functionality offline:

1. **Browse list of nearby buildings**

All buildings and their associated toilets are cached on the first run of the application. This means that while the user is offline, they are still able to swipe up and see the list of toilets on the explore page, sorted by distance from last saved location.
Additionally, the list of buildings nearby will still show, under 'Is nature calling now?' tagline. The user is still able to tap on the buildings and see the toilets within each building.

2. **Show details of toilets**

If a toilet's detail page has been accessed before, data in it will be cached and displayed to the user, even if offline. This means that the user can lose his internet connection and still view the various details about the toilet they are currently looking at.

3. **Posting review for a toilet**

In the case that the user is offline, the review to be posted will be kept in the persistence management store. Upon reconnection, the review will then be uploaded, ensuring a seamless experience for the user.

4. **Post reports for faulty items**

Reports for faulty items are kept in the persistence management store until a network connection is established, upon which it will be uploaded to the server. This means that the user will not have to deal with the frustration of failed reports.

5. **Change password**

Changing passwords can be done offline, also stored by the persistence management store. When the user reconnects, the current access token, along with the password, is sent to the backend to be updated.

6. **Location services**

Even if the user is offline, the user is still able to click the 'my location' button, which is the circular floating button on the bottom left with a navigational arrow within it. While the map will not load, the markers already on it will be displayed. This means that the user can still roughly gauge which toilet is nearby them, and still be able to browse the list of toilets by building.

We believe that the above fits our user's expectations as we do not expect them to use the app primarily offline. Instead, what is likely to happen is that bouts of disconnection will be experienced, owing to the unreliability of mobile networks. Upon reconnecting to a working mobile network, the map will automatically load, and any pending user action executed over the network. This ensures a seamless and uninterrupted experience for our users.

To maintain data synchronization between our client and server, upon each reconnection to a network, the client calls the /toilets/version route to check if there has been an update to the toilet data on the server. If there is an update on the toilet version on the server, the client will re-fetch the list of buildings and their corresponding toilets from the /toilets route. This keeps the data on the client-side in sync.

As for the toilet's details page, the data is cached for offline viewing as well. However, as soon as the client detects a working network connection, it will query from the server instead and override the locally cached copy.

Milestone 11:

Session-based authentication and token-based authentication both share similarities in the way they are implemented. At their core, both send a request to the server to authenticate, which will return a 'token' to be supplied in subsequent requests to authorise and identify the user. In the case of session-based authentication, this comes in the form of a cookie with a sessionId, and in the case of token-based authentication, a token is used instead. The most common form of this is the JSON Web Token (JWT).

An advantage of the token-based approach is that it is much more scalable than the session-based approach as no server-side processing is needed. Session-based authentication relies on the server to store a mapping between sessionId and the associated user, querying the relevant database to determine which user the sessionId belongs to. This can become an issue when a large number of users are all logged in at once.

Token-based authentication, on the other hand, does not have this problem, since the token itself stores the user's data, and can simply be decoded at the backend using the token secret. No querying is required to identify the user.

Additionally, session-based authentication is often limited only to one website. This is due to browsers disabling cross-domain cookies by default. Token-based authentication does not have this issue, as the token is simply included in the request header, regardless of the domain.

We decided to use the token-based approach as it would save us having to store a mapping of sessionIds to the respective users, and having to query each sessionId. Additionally, Express had built in support for JWT with express-jwt, making authentication a breeze to implement.

To ensure that we had a way to invalidate tokens, we implemented both access tokens, with a 1 hour validity, and refresh tokens, which are stored in the database under the refresh_tokens table. This lets us log users out properly, as their access_tokens will eventually expire.

We were also familiar with JWT, and hence, it was also easier and faster for us to implement token-based authentication.

Milestone 12:

First we listed the components we need on our application. Our whole aim was to get a UI library that gives us most of the components we need so that we can cut down on our code on implementing them. Of course, the UI library needs to give us a good overall look and a native feel. Below are some of the components that we needed in our platform and the framework that provide those components:

Components	**Framework7**	**Ionic**	**Onsen**
Slider / Swiper	✔️		
Input Field	✔️		✔️
Card	✔️	✔️	✔️
Infinite Scroll	✔️	✔️	
Tabs	✔️		✔️
Bottom Drawer	✔️	✔️	

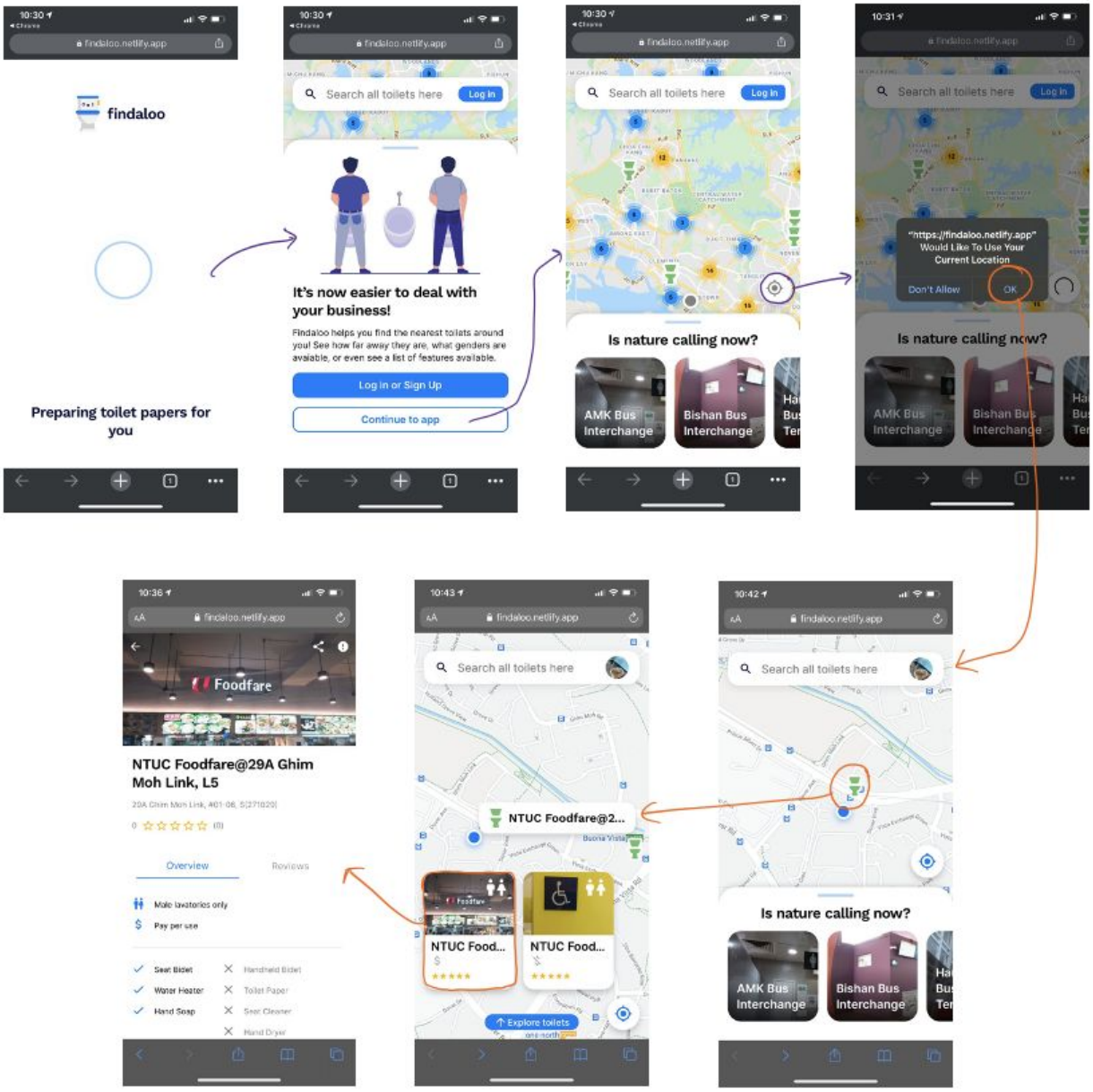
From the table above, it is clear to us that Framework7 provides most of the components we need. In addition, Framework7 also provides routing capabilities, which helps us with the heavy lifting on routing between pages.

Mobile Design Principles

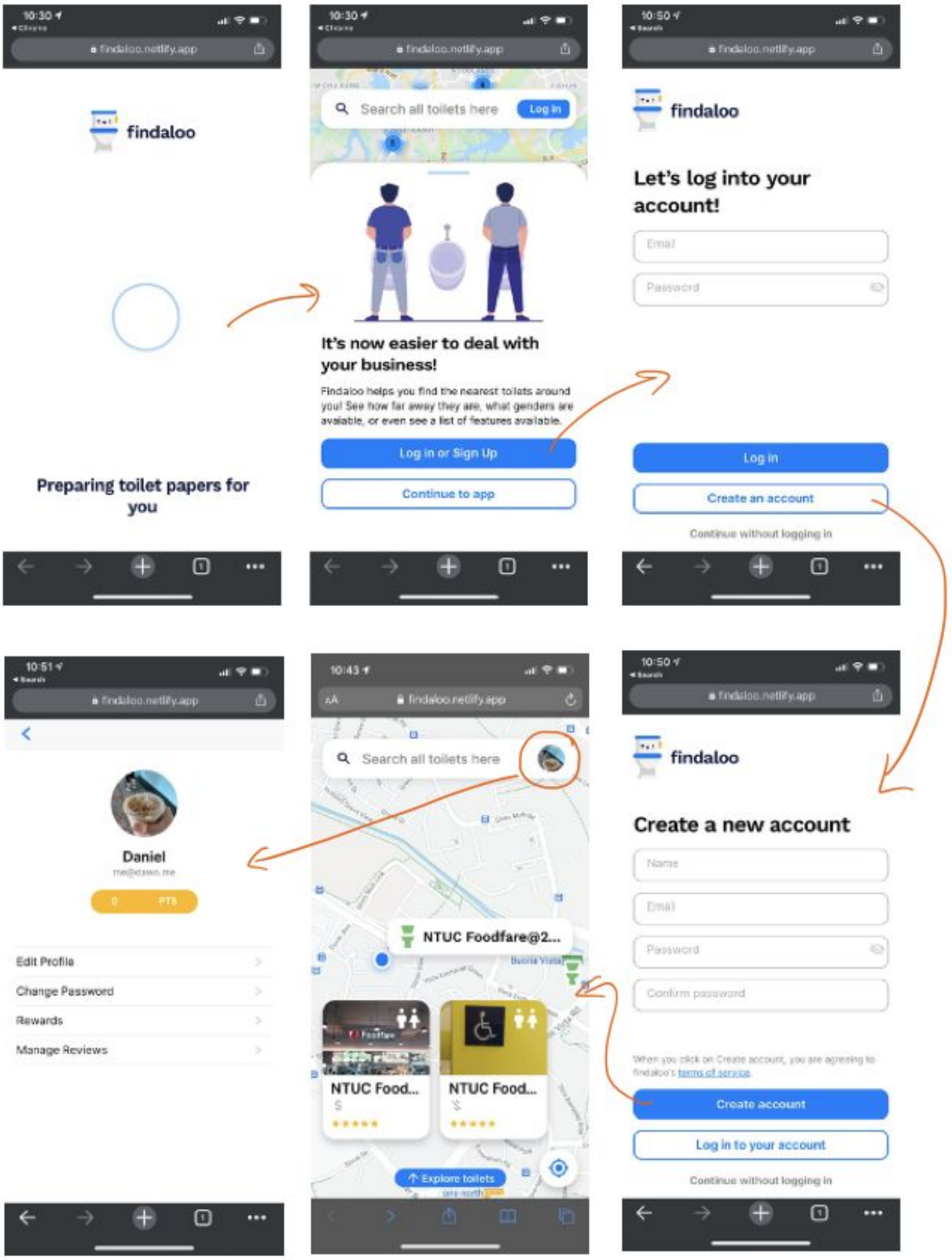
Principles	**Pages / Screens**
Bottom Drawer	• Explore Screen (Google Maps Screen)
Ability to go to previous page	• Toilet Details & sub screens • Profile & sub screens
Mobile Layout Design	• All screens
Swipe Gesture	• Explore Screen (Google Maps Screen) - Bottom Drawer • Toilet Details - Images
Intuitive Navigation	• Explore Screen

Milestone 13:

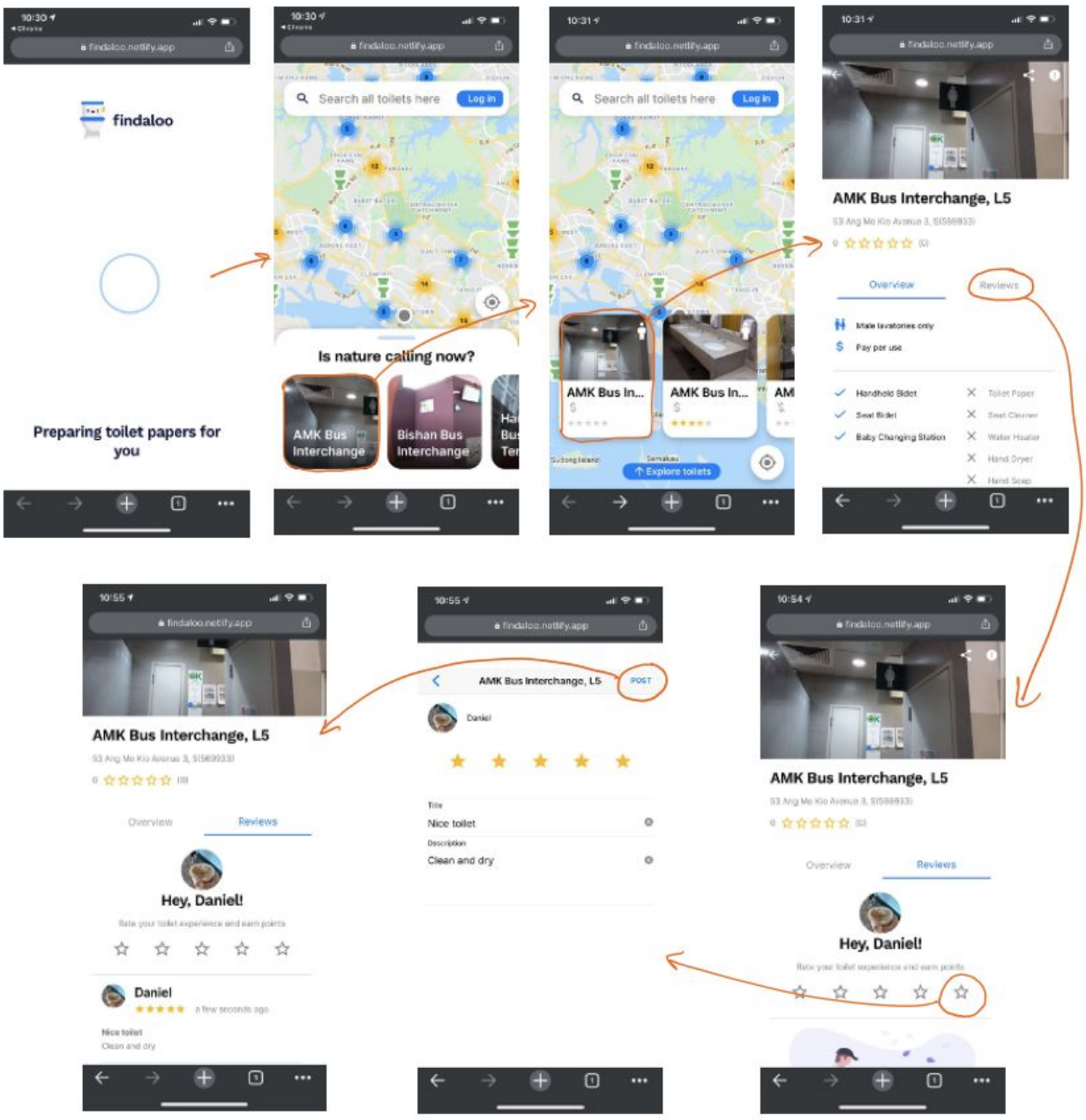
1. Find a nearby toilet



2. Signup for the application



3. Create a review



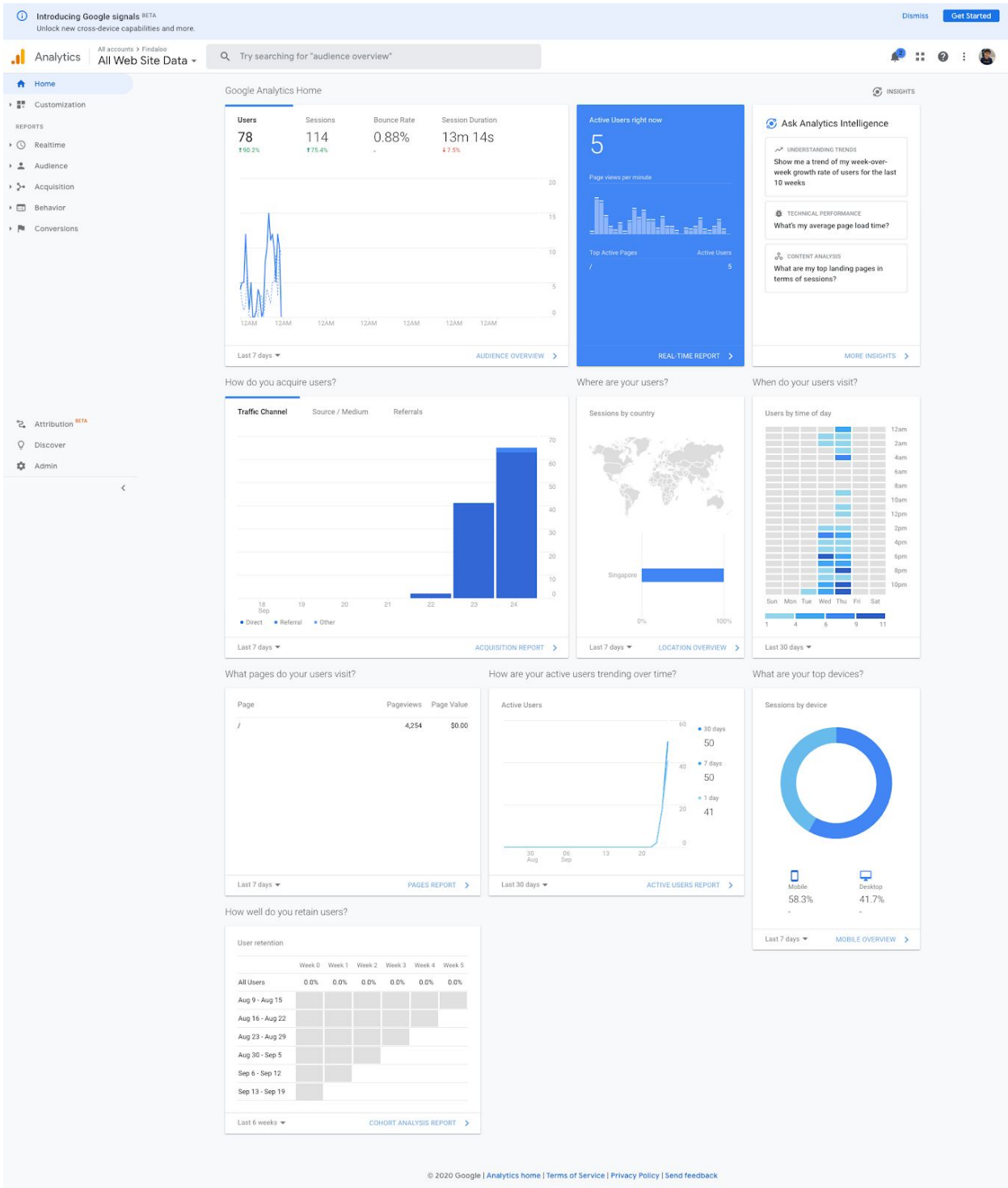
The three workflows shown above are the 3 main workflows of our application.

Find a nearby toilet:
The purpose of this application is to allow
The purpose of the application is to allow users to find a toilet fast, we highlighted this workflow to show how simple a user can find a toilet that is nearest to him and also see some important information related to the toilet

Create a review: As reviews could help to boost the overall experience on how users can know the experience of users in the toilet using the same toilet. This give all users a sense of how the toilets has been recently
Signup: Signing in allows the user to do more on the application. If they are not logged in, there are several pages which are not accessible to them. Thus by logging in, it further improve the whole experience on the application.

Milestone 14:

Due to the limitations of a PDF file, the screenshot below may not be readable. To view a higher resolutions document, please open the following link: https://drive.google.com/drive/folders/1_UD8lpCJ0pemWf_eYRGqfI3m-XL-W4Bp?usp=sharing


Introducing Google signals BETA
Unlock new cross-device capabilities and more.
Dismiss
Get Started
Analytics
All accounts > Findaloo
All Web Site Data
Try searching for "audience overview"
Home
Customization
REPORTS
Realtime
Audience
Acquisition
Behavior
Conversions
Attribution BETA
Discover
Admin
Google Analytics Home
INSIGHTS
Users
78
Sessions
114
Bounce Rate
0.88%
Session Duration
13m 14s
Last 7 days
AUDIENCE OVERVIEW
Active Users right now
5
Page views per minute
Top Active Pages
Active Users
/
5
REAL-TIME REPORT
Ask Analytics Intelligence
UNDERSTANDING TRENDS
Show me a trend of my week-over-week growth rate of users for the last 10 weeks
TECHNICAL PERFORMANCE
What's my average page load time?
CONTENT ANALYSIS
What are my top landing pages in terms of sessions?
MORE INSIGHTS
How do you acquire users?
Traffic Channel
Source / Medium
Referrals
Direct
Referral
Other
ACQUISITION REPORT
Where are your users?
Sessions by country
Singapore
LOCATION OVERVIEW
When do your users visit?
Users by time of day
Sun Mon Tue Wed Thu Fri Sat
Last 30 days
What pages do your users visit?
Page
Pageviews
Page Value
/
4,254
$0.00
PAGES REPORT
How are your active users trending over time?
Active Users
30 days
50
7 days
50
1 day
41
ACTIVE USERS REPORT
What are your top devices?
Sessions by device
Mobile
58.3%
Desktop
41.7%
MOBILE OVERVIEW
How well do you retain users?
User retention
Week 0
Week 1
Week 2
Week 3
Week 4
Week 5
All Users
0.0%
Aug 9 - Aug 15
Aug 16 - Aug 22
Aug 23 - Aug 29
Aug 30 - Sep 5
Sep 6 - Sep 12
Sep 13 - Sep 19
Last 6 weeks
COHORT ANALYSIS REPORT
© 2020 Google | Analytics home | Terms of Service | Privacy Policy | Send feedback

Milestone 15:

The Lighthouse HTML report can be found in our repository in the root directory as group-3-lighthouse.html

Alternatively, it can be accessed here: https://github.com/cs3216/2020-a3-mobile-cloud-2020-group-3/blob/master/group-3-lighthouse.html

We have fulfilled 14/14 of the criteria required for a progressive web application.

Milestone 16:

Given additional time, the social network we would integrate with would be Google. While not exactly a social network in the traditional sense (considering Google+ has shut down), it nevertheless provides a ubiquitous OAuth2 solution to integrate and authenticate users with.

We chose Google because almost everyone has a Google account. It is highly likely that you are already logged into a Google account on your mobile device, whether you want to or not. Additionally, on Android, Chrome automatically logs into any Google website based on the account currently on the device.

The benefits of Google not being a social network also help us reduce potential friction in the sign-up process: users are more likely to sign-in with Google, knowing that no one will be able to see that they used findaloo, as compared to Facebook, which may reveal the applications that you have authorised, depending on your privacy settings.

Additionally, sign in with Google is used only to identify the user. In the sign-up phase, only their name, profile picture and email are retrieved, and these can be changed any time. This means that users that do not want to be publicly associated with their findaloo accounts can maintain their privacy while opting to use their Google accounts as a convenient sign-up option.

Milestone 17:

As findaloo is primarily a map-based application, geolocation is not just an optional feature that merits inclusion, but instead, a necessary feature to improve the user experience. Our application supports geolocation with the help of the location button, represented by the circular floating icon with a navigational arrow inside it.

By clicking on it, the map will automatically focus on the location the user is currently at. This location data can also be used to retrieve relevant information for the user, such as the nearest toilets to them, and also sort results by relevancy (distance).