# Restroam

**Technical report – group J**

Idris Babay
Eya Chemangui
Aziz Chikh
Jan-Christoph Cramer
Ahmed Fourati
Manuel Scheurich

July 18, 2022

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Internet-Praktikum
Telekooperation
Sommersemester 2022
Fachbereich Informatik

# Contents

# 1 Motivation

The urge to go to the toilet is one of the natural needs of every human being. In an unfamiliar environment, quickly finding a toilet is often difficult and complicated, despite signage. That is why we have made it our task to programme **Restroam**, a location-based toilet finder app. This app allows the user to show all the toilets in their environment.
For this purpose, our app offers an interactive map that shows the toilets and their location in a user-friendly way. In addition, the user can also view the toilets in a comprehensible list that can be sorted by distance and price. The toilets themselves are added to the app by the users themselves. These can then be rated or reported by other users of the app. The basic function of the app, i.e. displaying the toilets, is also possible without logging into the app, so that you can find the nearest toilet as quickly as possible. For creating, rating and reporting the toilets, a registration is required. In addition, a user then has access to their profile, where he or she can view messages about their toilets, the ratings they have written and the toilets he or she has created.
Through these design ideas, we combine the main functionality of quickly locating toilets, with additional customisable features to enhance the experience of using the app.Since there are already several implementations of different toilet finder apps available in the popular app stores, we focused on a clear and user-friendly design during the implementation of this app. This was not always given in the other apps we found in the app stores. In addition, we made sure that we used a consistent design style in the entire app.

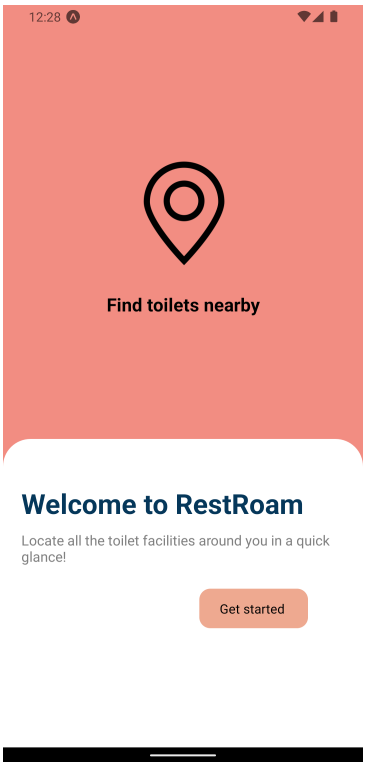


Figure 1.1: Welcome page

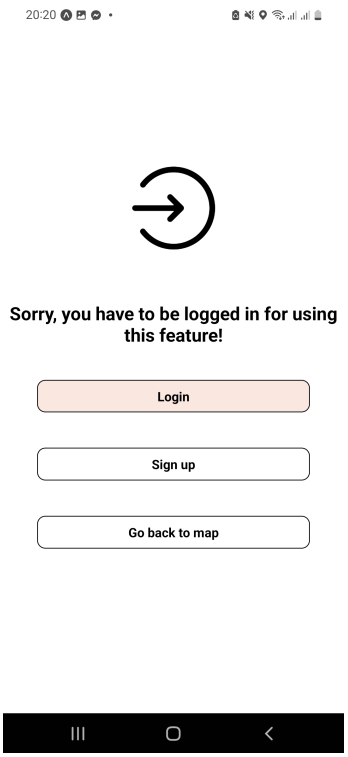


Figure 1.2: Outlogged screen

# 2 Technical overview

## 2.1 React Native

For the implemention of the frontend we decided to use **React Native**. This framework is an open-source UI framework that is widely used in the front-end development of applications. The big advantage of React Native is the platform independence of the developed apps. Thus, as a developer for Android and IOS, you do not have to develop a duplicate code base, but develop a code that is based on Javascript and can then later be displayed on all common platforms. The later user interface is not a web app, instead it is a native user interface. This means that the UI components of the respective operating system are used.
Another great advantage is the existence of already implemented components that can be quickly and easily integrated into the project. These components can then be individually designed and adapted to your own requirements via their properties, which enables a flexible field of application.

### 2.1.1 Used components

For our app, we used many pre-built React Native components that accelerated the front-end development of our app. In the following, we will take a look at the most important of these components to illustrate the fundamental elements of the app.

**react-native-maps** For our map we used the components from the library **react-native-maps**. This uses the native version of the existing operating system to display the map. So when integrating the component, Apple Maps or Google Maps for IOS devices are used and only Google Maps for Android devices are used. It provides an intuitive API for the declaratively controlled features of the map. The features are considered as child components of the general MapView. The MapView component can thus be used to control the properties of the displayed map, as well as to insert created toilet markers into the map. In addition, the component provides the interface for all event handlers, such as when a toilet icon is clicked. The current position of the mobile phone is also displayed on the map as a native marker if there is a GPS connection.

**react-native-maps-directions** The library **react-native-maps-directions** allows to plan a route between two coordinates based on the Google API. In our case we use the library to plan the route from the current location to the selected toilet. A MapViewDirections component is rendered and processed as a child of a MapView from the react-native-maps library.

**expo-location** The library **expo-location** allows us to read the GPS data of the device. This allows us to determine the current location of the mobile phone in the app if the user allows the app to access the device's location data. To do this, the library provides the previously implemented methods *Location.requestBackgroundPersmissionAsync()* and *Location.requestForegroundPersmissionAsync()*, which ask for the user's native location access permission for the app. In combination with the react-native-maps library, we can thus display the location on the map in the app.

**react-native-modal-dropdown** For our list of toilets on the map, we decided to use the **react-native-modal-dropdown** library. This provides us with a customizable list that then displays the toilets. This list also allows filtering by selected criteria of the list.

**react-native-vector-icons** All the icons we used in our app are available in the library **react-native-vector-icons**. You can easily import the icons from the library into your project and then customize them to fit your needs.

**react-native-star-rating** For the ratings of the toilets we used the library **react-native-star-rating**. This provides clickable and customizable rating based on stars. The intuitive design should simplify the rating of a toilet. In addition, we have also presented the average rating of a toilet using this library.

**react-native-community/datetimepicker** For the selection of the opening hours of the toilet we decided to use the library **react-native-community/datetimepicker**. This provides an easy to use design that implements the input of the time using a disk. This is a useful tool especially for mobile applications, like our app, as keyboard input for times is often complicated. Of course, keyboard input is still possible with this library.

**expo-image-picker** For the selection of images from the local file browser of the device we have chosen the library **expo-image-picker**. This brings a UI through which the user can access his or her local file system if he or she has given the app the permissions. There, the user can then add the images to the toilet, rating or message. It is also possible to upload photos taken directly with the camera via the library.

**react-native-paper** The **react-native-paper** library has provided us with customizable components. Thus, the selectable badges of the account page consist of these components. The pre-built components can be customized to suit all application purposes via the properties. They follow the Google Material Design guidelines.

### 2.1.2 Expo

We used **Expo** in combination with React Native to test our app in development and later to build an executable app. This free framework takes a lot of work off your shoulders with its services, so you can use *start* to start the app on a local Expo server and then call it with the wrapper app Expo-Go from your own phone or from any emulator.
This framework also takes care of the final building of the app for free. The app is built on Expo servers via a pipeline adapted to the operating system.

## 2.2 Node.js

We chose **Node.js** as the runtime environment because it uses an event-driven and non-blocking I/O architecture. It is used to execute the JavaScript files outside the browser.

### 2.2.1 Node Package Manager

We used the **Node Package Manager (npm)** as package manager for our project. This offers a command line client (CLI) for Node.js as a runtime environment, which is equipped with over 1.3 million packages that

can then be installed by the developer via npm. To install packages that are located in the *package.json* file, a single command is sufficient. In addition, npm has the advantage that it intelligently resolves dependency conflicts of different packages.

### 2.2.2 Yarn

In addition to npm, we also used **Yarn** as a package management system. Yarn is another package manager for JavaScript. Yarn was developed as an alternative to npm while offering the nearly same features. The packages from Tensorflow that we need to recognize our Not-Safe-For-Work images are unfortunately not available in npm, so we had to use Yarn as an additional package manager.

## 2.3 MongoDB

For the implementation of the database we decided to use **MongoDB**. MongoDB is a document-oriented NoSQL database management system. For our app, we created a MongoDB Atlas cluster that manages our data as cloud storage. You get 5GB of cloud storage for your database application for free, which is totally sufficient for our application initially. The data we need in our database is stored in JSON-like documents. In addition, MongoDB enables dynamic schema design so that documents can be stored in different structures. This also allows changes to be made to the structure in an uncomplicated manner.
We used mongoose for Node.js as an Object Relational Mapper (ORM) for our MongoDB database. This allows us to store our various objects created in the app, such as toilets, accounts, etc. directly in the database and have access to them later.

# 3 User guide

When the user clicks on *Get started* located on the welcome screen, he or she will be redirected to the map even if he or she is not logged in. This contributes to a better usability and user experience of the app. In this case the user only can view nearby toilets. If he or she tries to add a toilet, review a toilet, make a complaint or view the account screen, he or she is redirected to the outlogged screen, where he or she will get the possibility to create a new account or to log in to an already existing account or to go back to the map.

## 3.1 Finding toilets

In order to find the toilets nearby, the user must first navigate to the start page of the app, i.e. the page where the map is displayed. When using the app for the first time, the user is initially asked which permissions for location access he or she would like to give the app. The selection *Only while using the app* is sufficient for the use of the app. The user must then ensure that the location services of their device are activated.
Once this has been done, the user can display the toilets in their facility on the map. The map can be moved using the standard control gestures. For example, the map moves by holding and dragging a finger on the map at the same time. The user can zoom in and out of the map with two fingers. The toilets are shown as clickable yellow toilet symbols on the map. The symbols are located exactly at the place on the map where they were registered by the owner and are then expected to be located there in the real world.
When the user clicks on a symbol on the map, a small window opens with information about the selected toilet. The name, the address, the average rating and the registered opening hours of the toilet are displayed in this window. In addition, the user has three interaction buttons there. By pressing the first button named *Directions*, the user is shown the fastest route from his or her current location to the selected toilet in his or her preferred navigation app.
Pressing the second button named *Reviews* will open a new window where the user can view the reviews of the selected toilet. On this page, the information about the toilet is displayed at the top. The button labeled *Add Review* allows the user to add a review of the toilet if he or she is logged in. Other users' reviews are displayed below the button. With the arrow at the top left, the user can navigate back to the map.

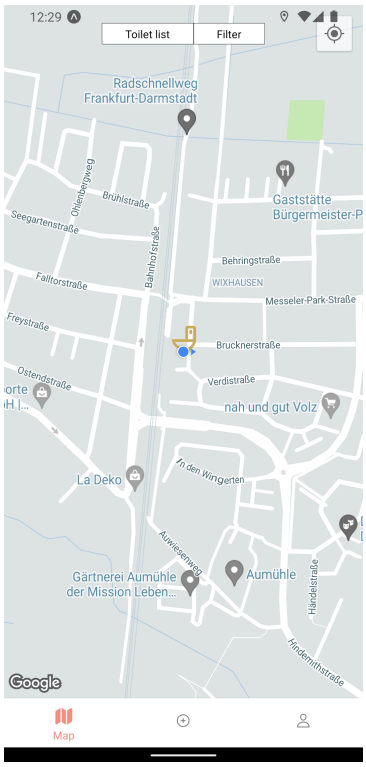


Figure 3.1: Map

When the third button with the name *Report* is selected, the user can report the toilet to the toilet owner for various reasons.
In addition to displaying the toilets on the map, the user can also view the toilets in a list. To show the list, the user must click on the button labelled *Toilet list*, which is located at the top of the map. The list can be sorted according to various criteria using the button *filter*. The criteria include the distance to the current

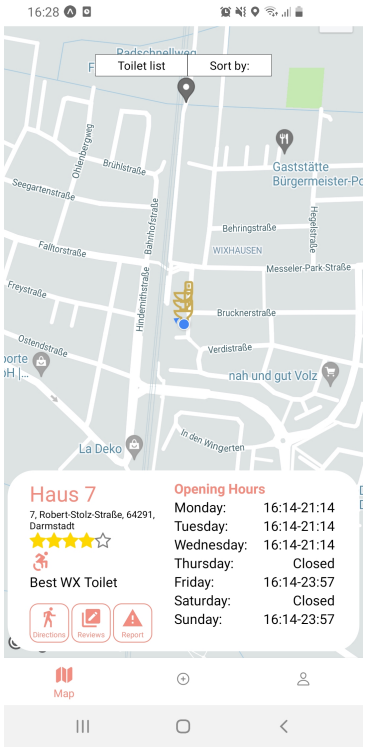


Figure 3.2: Map with selected toilet

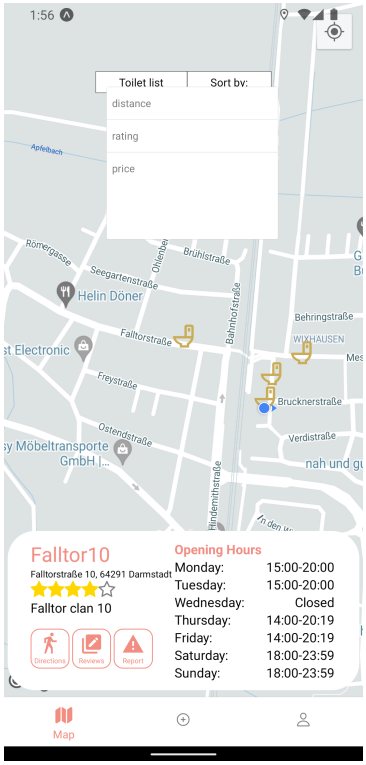


Figure 3.3: Filter options

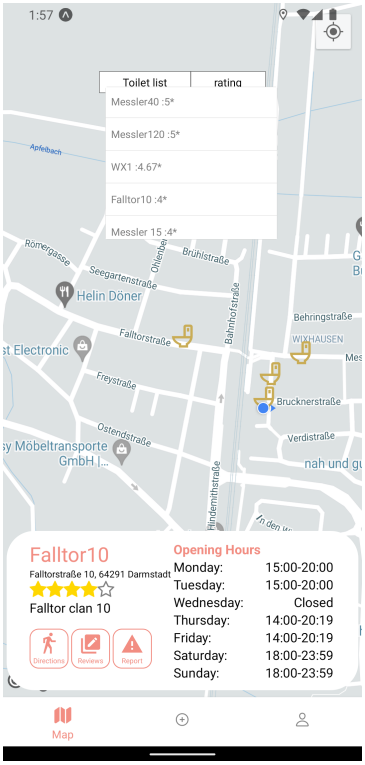


Figure 3.4: Filter applied

location, the rating or the price of the toilet.
The button in the top right corner allows the user to centre the map directly based on his or her current location.

### 3.1.1 Review a toilet

If the user of the app wants to give a review via the window of the toilet on the map, he or she is redirected to the review page. There, the user can then give their rating in the form of a star rating in different criteria. The criteria include *Cleanliness*, *Waiting time* and *Security*. The rating in the individual criteria can be directly specified by selecting the stars of the criteria. The stars can be assigned from one to five. Whereas one star is the worst and five stars the best rating for each criterion. The average rating of the three criteria is then calculated and displayed as an explicit rating. In addition to the star ratings that a user can give, he or she can also enter a text, which is limited to 250 characters. Furthermore, the author of the rating has the option of attaching images to the rating. The rating is then sent and saved via the *Submit* button.

### 3.1.2 Report a toilet

The user of the app has the possibility to report a toilet to the owner of the toilet. To do this, he or she must click on the toilet that he or she wants to report and then select the button labelled *Report* in the window. A new page will then open where the user can choose between five different reasons for reporting. The first


Figure 3.5: Inspect ratings

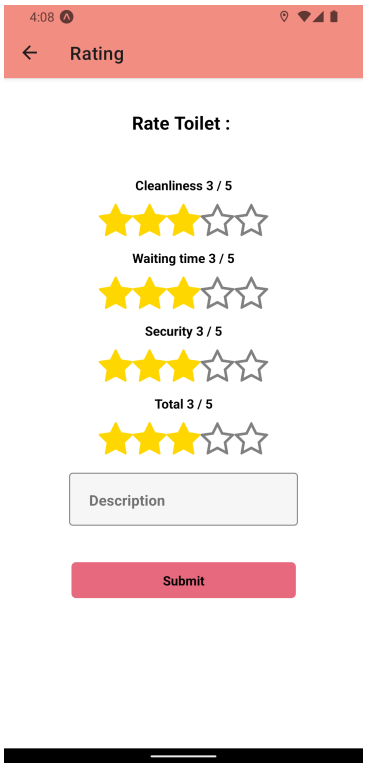


Figure 3.6: Review a toilet

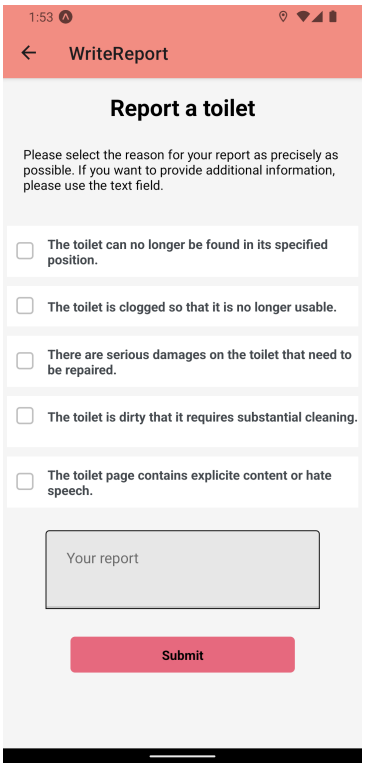


Figure 3.7: Report a toilet

reason should be selected if the toilet was not found by the user at the position indicated by the map. In this case, a closed toilet outside the specified opening hours does not justify a message with this reason. The second reason should be selected if the toilet drain is clogged so that the water and excrement can no longer flow out. The third reason should be selected if there is severe damage to the toilet so that the owner of the toilet can repair it. The fourth option should be selected if the toilet is heavily soiled and it needs cleaning. The last reason should be selected if the description of the toilet contains hate speech or explicit content is found on the page of the toilet.
In addition to the five given choices, a user can still write a text about the report. This should be done if either the reason for his or her message was not covered by one of the five options or he wants to attach additional information to his report. The text input is limited to 250 characters. The user finishes his reporting process with the button labeled *Submit*.

## 3.2 Creating toilets

To create a toilet, the user of the app must initially select the middle box labelled *Add* via the bottom navigator. There, the user first specifies the weekly opening hours of his or her toilet. To do this, he selects the time when the toilet opens via the predefined button *PICK START TIME* and the time when the toilet closes via the button *PICK END TIME*. Both buttons open a time setting interface where the user can set the hours by clicking on the hours and the minutes by clicking on the minutes through the selection dial. An entry with the keyboard is possible by selecting the keyboard symbol at the bottom left. The user can then use the checkboxes to determine the days on which the specified times apply. If the user wants to specify different opening times for other days, he can use the button *[+] Add more Slots* to create more slots, in which he or

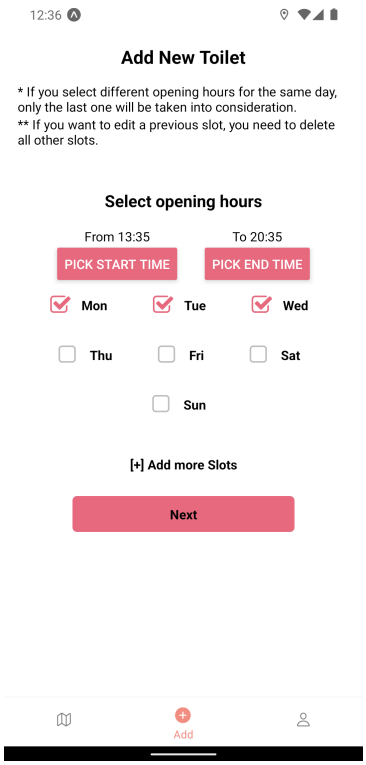


Figure 3.8: Select opening hours

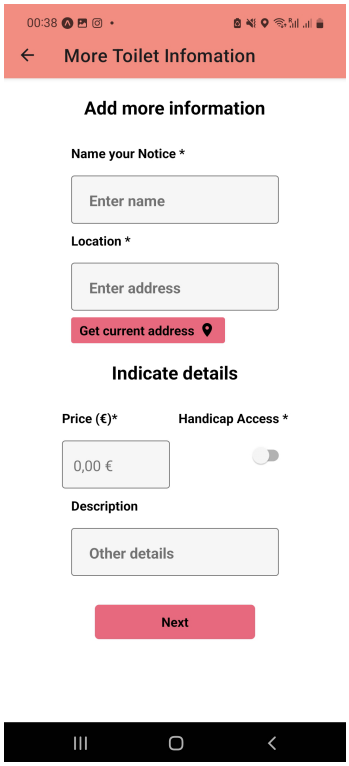


Figure 3.9: Add more details

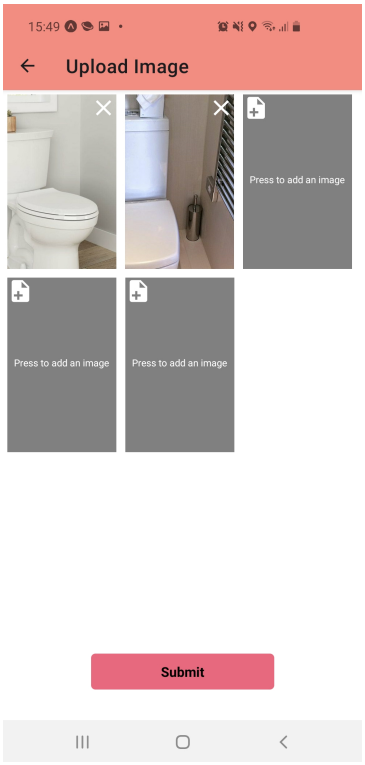


Figure 3.10: Upload images

she enters the times for the selected days as in the first step. A maximum of 7 time slots are possible for a toilet at the same time. The button *[-] Delete Previous Slot* deletes the last slot added with its entries. If the days in the different slots overlap, the last slot added is always taken into consideration. If the user wants to edit a previous slot, he or she must first delete the subsequent slots. The button labelled *Next* takes the user to the next page of the creation process. At the same time as the button is pressed, the entered opening hours are checked. In this way, it is checked if any data has been entered at all. Then it is checked if all opening hours are valid. If this is not the case during one of the checks, an error window appears and the user can check his or her entries again. If the opening hours are entered successfully, the user is redirected to the next page.

On the next page of the toilet creation process, the user can add more information about the toilet.The user can enter the name, address of the toilet and the price of the toilet on this page. In addition, the user indicates on this page whether the toilet has access for handicapped people. When the user presses the button *Get current address*, the address line is automatically filled with their current address. This requires that the device's location services must be turned on and the app must have the access permissions to retrieve the location data. In addition, the creator of the toilet can enter a description of the toilet, which is limited to a maximum of 250 characters. With the button *Next* the user navigates to the last page of the creation process. On the last page, the user can add up to 5 pictures to the toilet. To do this, the user must click on one of the 5 dummy photos and can then select the photo he wants to upload via his file browser. The user finishes his toilet creation process with the button labeled *Submit*.

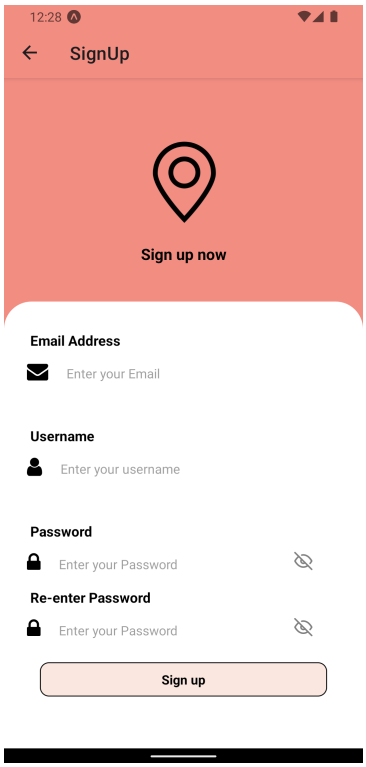


Figure 3.11: Sign up

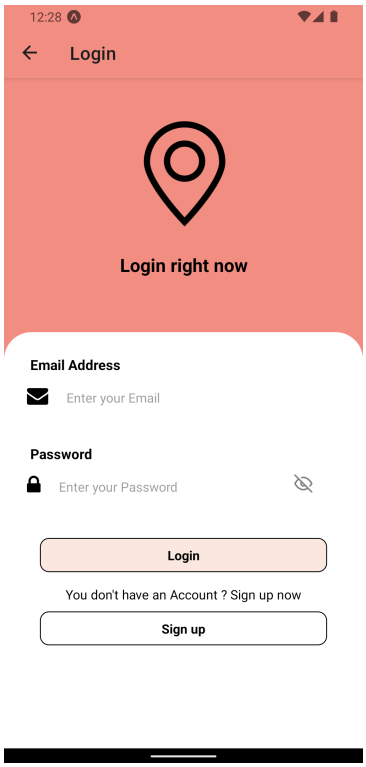


Figure 3.12: Log in

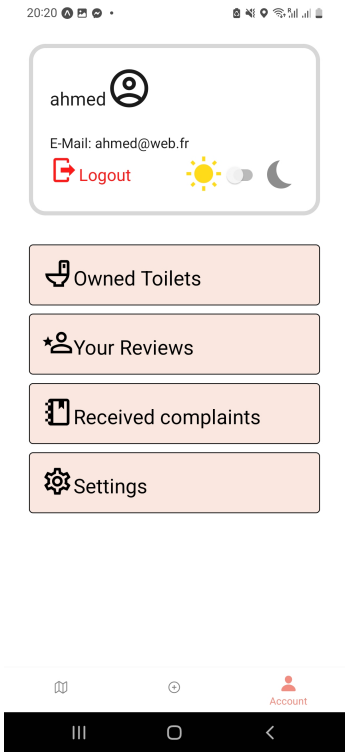


Figure 3.13: Account screen

## 3.3 Account

### 3.3.1 Sign up

To register a user in the app, the user must first navigate to the page labelled *Account* via the bottom navigation bar. There the user can then select the button labelled *Sign up*. The user is redirected to the registration page. There the user enters his or her e-mail address in the first line. If the e-mail address is valid, a green box is displayed on the right side of the line. If this box is not present, the entry should be checked again. In the next line, the user selects his or her user name for the app. Then the user must enter his or her desired password twice. If the user wants to see the inserted password when entering his or her password, he or she must click on the corresponding eye. To hide the password again, the button has to be pressed again. The user completes his or her registration by clicking on the button labelled *Sign up*.

### 3.3.2 Log in

To register a user in the app, the user must first navigate to the page labelled *Account* via the bottom navigation bar. There the user can then select the button labelled *Log In*. The user is now redirected to the login page. In the first line, the user now writes the e-mail address that belongs to his or her account. In the second line, the user must now enter the corresponding password. By clicking on the eye, the user can make the password visible. Clicking on the eye again hides it again. The user completes the login process by clicking the button labelled *Login*. The data entered is checked for correctness when the button is pressed. If the entered data is correct, the user will be redirected to the map. If the input is incorrect, the user can check his or her input data again.

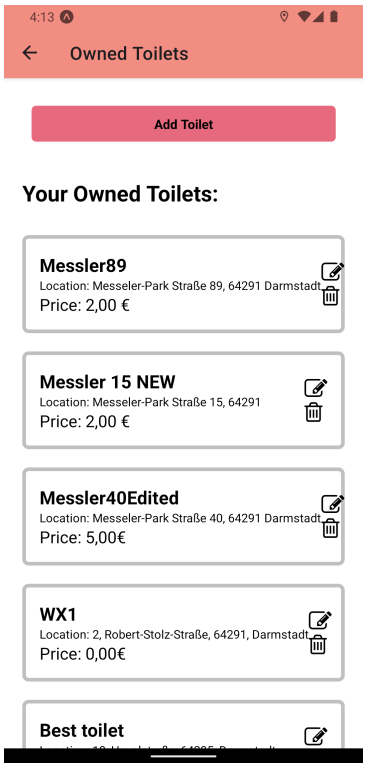


Figure 3.14: Owned toilets

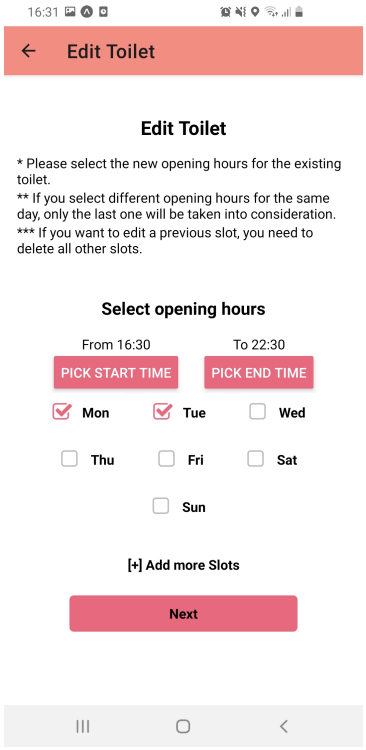


Figure 3.15: Set new opening hours

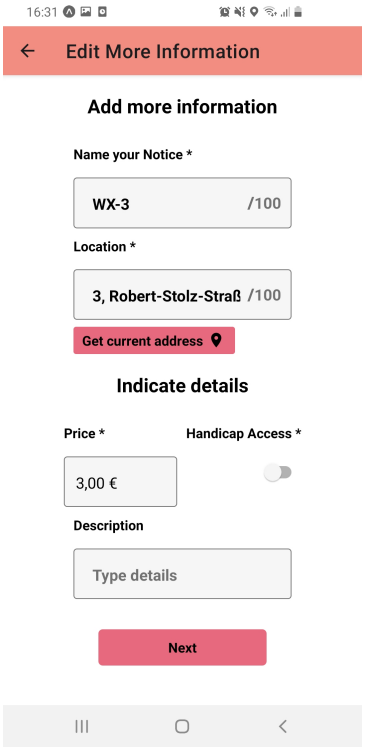


Figure 3.16: Edit more information

### 3.3.3 Log out

If the user has successfully logged into his or her account, he or she can now log out again via the page *Account*. To do this, the user must click on the field *Logout*.

### 3.3.4 View owned toilets

If the user has successfully logged into his or her account, he or she can now view and edit his or her owned toilets via the *Account* page. To do this, the user must click on the *Owned Toilets* button. The user will then get a list with all his created toilets. The most important information about each toilet is displayed there, i.e. name, location and price. On the right side of each toilet, the user can edit the toilet by clicking on the pencil. This will redirect the user to the edit page, where he must first enter the new opening hours. By clicking *Submit*, the user will be redirected to the next page where they can edit more information. This page is structured in the same way as the functionality to add toilets. After that, the user can still add photos to his or her toilet.
Clicking on the trash can will completely delete the toilet after the user confirms. The button at the top left takes the user back to his or her account page.
The button labeled *Add Toilet* will redirect the user to the functionality to add toilets.

### 3.3.5 View your written reviews

If the user has successfully logged into his or her account, he or she can now view, edit and delete his or her written reviews of other toilets via the *Account* page. To do this, the user must click on the button *Your*

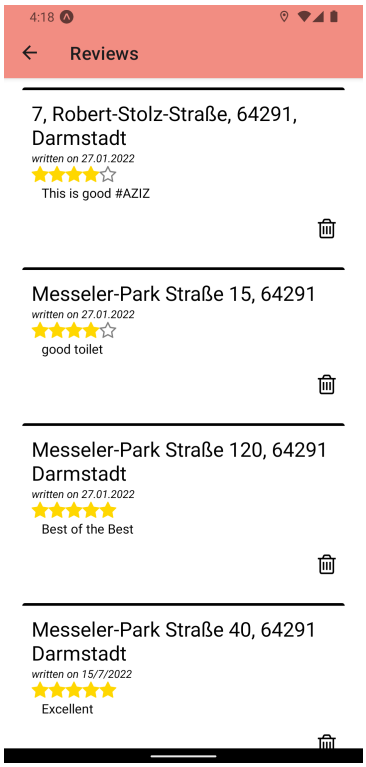


Figure 3.17: Written reviews


Figure 3.18: Received complaints

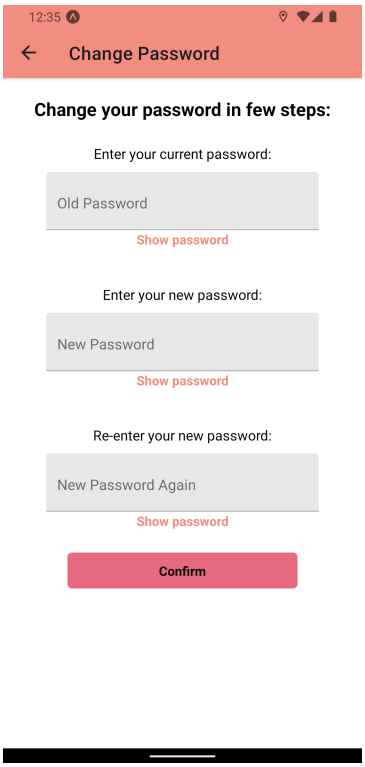


Figure 3.19: Change password

*Reviews*. The user can view all his or her reviews here. By clicking on the bin, the user deletes the review completely. The button at the top left takes the user back to the account page.

### 3.3.6 View complaints from your toilets

If the user has successfully logged into his or her account, he or she can now view the complaints about his or her owned toilets via the *Account* page. To do this, the user must click on the button *Received complaints*. There, all complaints are displayed in a list. The name of the affected toilet is stated in the headline of the complaint. The button at the top left takes the user back to the account page.

### 3.3.7 Change password

If the user has successfully logged into his or her account, he or she can now change his or her password via the *Account* page. To do this, the user must click on the *Settings* button. There the user clicks on the button *Change Password*. On the next page, the user must first enter his or her old password. Then he or she enters the new password in the following two entries. With the buttons *Show password* the user can make the corresponding password visible. By clicking on it again, the user hides the password. With the button *Confirm* the user confirms his or her new password and the entries are checked.

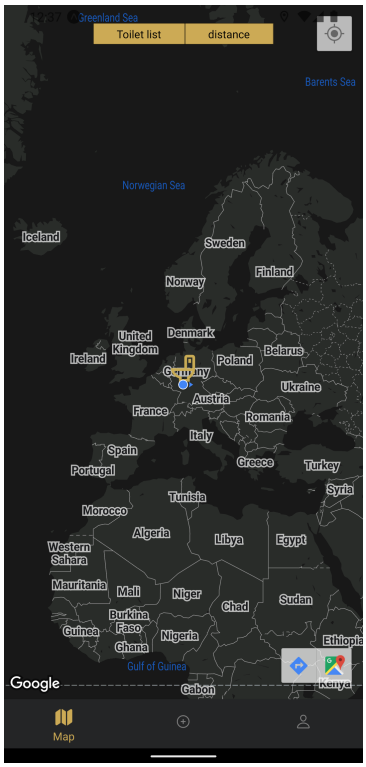


Figure 3.20: Map in dark mode

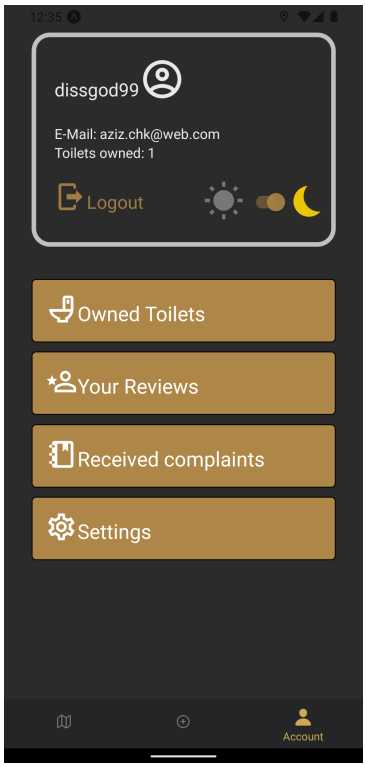


Figure 3.21: Account in dark mode

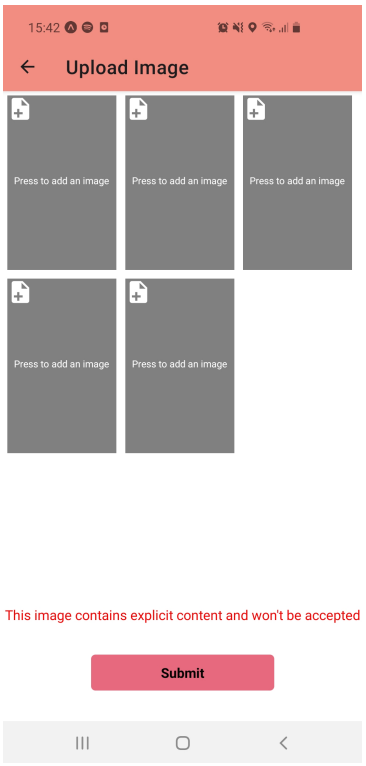


Figure 3.22: Detect a not-safe-for-work-image

## 3.4 Additional features

### 3.4.1 Dark mode

As an additional feature we have decided to provide the user with a **Dark mode**. To activate the dark mode, a user has to be logged in. The login process is explained in the **User guide** . Once the user has successfully logged in, he or she can enable dark mode through the page labeled *Account*. There, the user must move the slider from the sun to the moon. This is done by simply clicking on the slider. After that, the app's appearance will change to a design that uses darker colors. This is mainly to ensure more comfortable use at night and for brightness-sensitive eyes. Beyond that, the dark mode is of course also recommended for users who like the dark design more.
To implement the dark mode, we have outsourced a theme where different colors are used for different references depending on the user's selection. These can be changed or extended later. They can be found in the file */frontend/src/darkMode/Theme.js*.

### 3.4.2 Removing not-safe-for-work pictures

Another feature of our app is the detection and removal of images that are dangerous for the community. For this purpose, we use the Tensorflow model **MobileNet**. This is a pre-trained image classification model. This model is specially adapted to devices which have only a limited number of resources available, i.e. optimal for smartphones. The model then assigns to the image the most probable classes out of 1000 predefined classes. We look at all the assigned possible classes for the image and then check if any of the assigned classes are

contained in the predefined blacklist in the file *forbidden.json*. If this is the case, an image is classified as not-safe-for-work. If an image is classified as not-safe-for-work, the user will see an error message for the image and will not be able to upload it. Thus, for all upload functionalities, i.e. when creating, rating and reporting a toilet, we check whether the uploaded images are dangerous for the community.

# 4 Implemented features

- Find toilets nearby
  - Get an overview on toilet information
  - Display the toilets nearby in a list
  - Sort the nearest toilets by different criteria: distance, price and rating
  - View reviews from others to one toilet
  - Display the driving way
  - Center the map to current location
- Add a toilet
  - Determine the toilet's name
  - Select opening hours
  - Determine price
  - Choose location
  - Select handicap access
  - Add description
  - upload images
- Delete a created toilet
- Edit a created toilet
- Write a review to an existing toilet
  - Rate toilet based on three criteria: Cleanliness, waiting time and security
  - Add description
  - Add images to the review
- Write a complaint to a toilet
- Sign up
  - Enter email address
  - Determine username
  - Determine password
  - Hide password
- View self-written reviews

  - Delete review
  - View toilet name
  - View date of creating the review
  - View written description
- View received complaints
  - Show complaints that the user selected
  - Show description
  - Respect anonymity of the user, by hiding the personal data
- Log in the account
  - User remains logged in until he or she clicks logout
- Logout
- Change the password
- Enable dark mode
- Remove not-safe-for-work pictures

# 5 Conclusion and outlook

With our app Restroam, we have developed a working prototype for a location-based toilet finder app. This app fulfills our most important goal, which is to quickly find a toilet in the local area. This is accelerated by the fact that this functionality is also possible without logging in. All additional features of the app, such as creating, rating and reporting a toilet, as well as applying the dark mode are then possible with a login to the app. With this design we have tried to satisfy all the requirements of this app in a user-friendly way.
This app can of course be extended in addition to the implemented features. For this we have put special attention to a clear and understandable coding style. Additionally, our project is organized in an intuitive folder structure. Together with this technical documentation it should be easy for a developer to gain an understanding of the implementation of our app. For the future, we could implement additional settings for the account. For example, personalizable notification settings for received reports or ratings would be imaginable. In addition, privacy settings could be used to specify whether the username should be displayed when reviews are written. An optional profile picture could also be added to the account. There are also organizational tools for the report list for the created toilets of a user possible, which allows for example a confirmation of the processed reports.
As you can see, there are still many possibilities for expansion of our app. We would also be happy if this app is also further developed with own ideas.