Brno University of Technology
Faculty of Information Technology

ITU - USER INTERFACE PROGRAMMING
Project documentation

Team captain - Naumenko Maksim (xnaume01)
Mikhailov Kirill (xmikha00)
Kovalets Vladyslav (xkoval21)

14. December 2022

Content

Individual project steps

Naumenko Maksim (xnaume01)

Project theme

Chosen theme

After interviewing people about applications they use, various examples from different fields were obtained. The plant care reminder application was chosen as the main project theme, because almost every person interviewed has plants at home.

Survey with user

When the project theme was picked, a small survey to take more information about the user's needs was made. The survey was mainly in the form of informal conversation.

User needs and key issues

After small research some main user’s needs and key issues were figured out:

- Most of the users stop using applications due to inconveniences.
- Some applications do not have notifications, therefore, users simply forget about them and about taking care of their plants.
- In general most of the users want the application to be easy to use, that does its job and helps users properly care for their plants.

Changeset

In order to solve the main key problems and provide users with what they need, our application should mainly aim at the following points:

- User friendly design, everything should be simple and intuitive.
- The application should attract and support users in caring for plants.

Current solution description

During team discussion, the plant care reminder app was chosen as the main topic of the project. After a small group study, one general application was selected, which will be used to survey user needs and key issues.

Survey with user

This time the survey was conducted not only in the form of an informal conversation, but also in the form of an analysis of how the user interacts with the application.

User needs and key issues

After survey and analyzing next user's needs and key issues were figured out:

- One of the main problems is the advertising that appears when the user uses the application.
- There is no way to add an image of a plant.
- Only one type of care (watering), different types of plants need different types of care.
- Frequency of care is limited to one week, some types of plants need less frequent care.

Survey of existing applications

After small research two applications with the same topic were found.

WaterPlants

Advantages:

- Has an option to add a picture of the plant.
- Has two types of care.
- Has a more flexible type of setting care frequency.
- Has tips for watering plants.

Disadvantages:

- Design of the app has a lot of colorful elements, so it's quite hard to notice needed functions.
- Creation page is not intuitive.
- Setting up the schedule of care is quite confusing.
- Tips are not intuitive.
- The information about when you should water your plant is hard to notice.

Plant Daddy

Advantages:

- Has an option to add a picture of the plant.
- Has a more flexible type of setting care frequency.

Disadvantages:

- Cared plant is marked as non cared plant after restarting the app.
- Only one type of care.
- No history of care.
- The application freezes from time to time.

User interface design

Application Mockup

After analyzing the topic with the user and analyzing similar applications, a custom application mockup was made.

Home page(Image??)

Contains a list of cards with general information about each plant(image, name, information about next care activity), navigation bar and plant creation button.

Plant creation page(Image??)

Pop-up window that contains input fields for plant information: image, name, description, care activity type.

Plant page(Image??)

Contains all information about the plant: image, name, description, care activity type. Also there is a plant edit button and care activity button.

Plant edit page(Image??)

Pop-up window that contains the same information as the plant creation page and button to delete the plant.

Settings page(Image??)

Contain button to turn on/turn off notification

Care tips page (Image??)

Contains a list of cards with the name of the care tip.

Care tip page(Image??)

Pop-up window that contains all information of the care tip: name, description.

Mock-up testing

When the mockup was done, it was tested on users to get some feedback for the future improvements. The feedback from the users was mostly positive but some of the users had suggested some improvements:

- Add the option to take care of the plant from the home page.
- Add care activity history.
- Dark theme.

Mikhailov Kirill (xmikha00)

Project theme

Chosen theme

I'm not really a "party person" but from my friends with different "party type" I've heard a lot about them wanting to go to party somewhere, to a club or something like that. When I asked them about various event trackers, they told me that in most cases, there's a little count of parties in these applications, mostly some events of another kind and then they find out there were a lot of parties they didn't know about. So that is why I decided to help the hanging out segments of the population with the search for the right events, and the party organizers and club owners with promotion.

Survey with user

Then I started to interview my friends in an attempt to find out what they want from the application. Requirements were relatively simple : all they wanted was an application with only a list of parties, relatively clear information about the location and time of the event and the "main theme"(halloween party, techno music party, old-school hip-hop music party and so on) of a party, if any.

Fortunately, I have a friend, who is working as a DJ at a lot of clubs in Brno. He told me that a lot of party organizers and club owners do not even try to attract as many people as possible to their event, since that may be quite difficult for them to pick a right way of promotion and that is why they often work with zero (or even negative) income. So that's the reason why I wanted to make my application useful for both sides of the party - for organizers and for participants.

User needs and key issues

- A lot of irrelevant events for party searchers
- Non-instinctive party search in existing applications
- Not all of available events are recorded in existing solutions
- Difficult party announcement for organizers

Changeset

In my solution there will be two options of user profile - searcher or organizer. As a searcher your main search tool is a map of your place(according to the user's geographical position) with pointers to the places where the party is planned in the near future or it's already going. If map is quite an inconvenient option, the user is able to switch it to the list with different options of sort (according to distance, time, genre and so on).

As an organizer you can create an event with its location, time, name, main theme, optionally DJs names, dress-code constraints, drink list description and so on.

Current solution description

During the first team meeting we was discussing, which of our three themes to choose and we decided to choose a plant care application since this case is much more common than my first suggestion and much more people need a good plant care reminder

Survey with user

After my research, I found that a lot of people need this kind of plant care application. But the problem is that most of the available solutions only serve as a reminder and nothing more. In most cases, the application does not attract the user and there's no immersive factor in them. That is why, after some time of using the application the user is more likely to get tired of constantly incoming "watering" reminders, and is more likely to turn off notifications from this application.
Also some part of users encountered non-intuitive application interface and "non-transparent" use of the application functionality.
Finally, I've suggested users download an application of a corresponding kind and I just followed how he uses an application. On the basis of this research I've made the following list of key issues.

User needs and key issues

- Low immersivity and attractivity.
- Usage of some application functionality is not clear.
- Some applications don't support an option to upload a user's plant's photo.
- A lot of applications support only watering as a type of care, no fertilizing nor repotting.
- Users need to close an application to learn how to properly perform one or another type of plant care.
- Most of applications functionality is in paid version

Survey of existing applications

I've found two applications with similar functionality, as has ours:

1. Seqvoia

    - Advantages
        - Option to add photo of a plant
        - Able to recognize plant with photo
        - Option to pick plant's type to choose correct caring plan

        - Ability to set up to 4 caring methods (watering, spraying, fertilizing and repotting)
        - Ability to set a potting day
        - Statistics about user's activity in application (days in app, number of plants, care activities)
        - Ranking system
        - "Care tips" cards
        - Feature with following actual season for better care analyze
        - Reminder functionality
        - Ability to contact support
        - Handy calendar with option to view planned care activities
        - Care suggestions when adding new plant according to its type
        - Option to follow your plant's growing process
    - Disadvantages
        - No achievements (rank updates according to what?)
        - Unable to undo action
        - Sometimes some inscriptions lag or display incorrectly
        - Able to add only 2 plants in free version
        - Plant recognition is limited in free version
2. Remindew
    - Advantages
        - Option to add photo of a plant
        - Notifications
        - Ability to edit text in notification
    - Disadvantages
        - Almost empty and a bit ugly main page
        - Unable to set multiple parameters in "Settings" menu at once, after setting one option application throws you to main page
        - Non-user-friendly interface, adding new plant menu, scheduling
        - Non-intuitive and non-obvious use of app functionality
        - No achievements, nor ranking system, nor activity statistics (=> low immersivity)
        - Unable to undo action
        - No care options except of watering, so the whole functionality of the applications is to remind user to water the plant at manually set day
        - No calendar, only shows today's date

User interface design

Application Mockup

After my research of similar applications and user's requirements I started to create my version of our future application. I really liked the Seqvoia application's design and logic, so I wanted to create something similar, but with some changes

Home page(Image??)
Home page contains a list of all the user's plants. At the top of this list will be the plant, which needs care more, then others (the closer the color of the drop to red, the more urgently the plant needs care). Also it contains an "Add plant" button at the top-left part of a screen.

Plant creation page(Image??)
At this page users can add the plant's name, its type and some description for it. Also there's an option to add a photo of your plant.

Achievements page(Image??)
This page contains the user's actual rank and bar of progression to the next one. Below there's a list of available achievements. Each achievement will have its own image and will be clickable. Each achievement will also have its own page with info about getting it and how many "points" of rank it gives and so on.

Plant page(Image??)
Plant page will allow you to edit your flower's data (name, type, description, photo). Also care settings edit may appear, since my first suggestion was to use some API (Trefle API for example), which will allow us to pull caring settings based on flower's type.

Mock-up testing

When I finished my small mockup, I gave it to users to have some feedback about it. They tested it for some time and now I have this list of needed fixes :

- If we use the suggested API (plant page section), we'll need to add an option to add caring time and type manually, switch between recognizing via API and manual approach with other words.
- Add settings page with some UI adapting options
- Add dark theme
- Little changes of fonts and whole design needed

Kovalets Vladyslav (xkoval21)

Project theme

Chosen theme

Crypto wallet. In recent years, the crypto world has been developing at a staggering pace, but it still has room to grow, and I would like to help it with this. I came to the conclusion that applications for crypto wallets clearly need to be improved.

Survey with user

In a personal meeting, I interviewed friends who use several similar applications. I wrote down all the disadvantages and advantages. And I realized that these applications have their own "entry threshold" and it is different for everyone.

I would like to lower this threshold to a minimum. After all, a person needs only a couple of seconds to understand how to use an alarm clock, so why should it be different with a crypto wallet?

Some even have to use a browser extension due to the lack of functionality in mobile apps. I think this needs to be fixed.

User needs and key issues

- No sorting of coins.
- No notification of price changes.
- No hints. For example: There is no explanation that in order to send a coin to an address, you need some amount of ether on your account.

Changeset

The application should be understandable for everyone from the first seconds. I believe that this can be achieved with a special lite and advanced version of the application, as well as with the help of hints.
To the basic functionality, we need to add coin sorting and the ability to create lists.

Current solution description

After talking with the team, it was decided to create an application that reminds the user to take care of the plants.

Survey with user

After the survey, I realized that people are divided into two categories: the first are those who didn’t even know about the existence of such applications and the second, who once used it in the past, but stopped for various reasons.

User needs and key issues

- Availability of a calendar
- Achievements
- Various plant care options
- Notifications
- Care Tips

Survey of existing applications

I analyzed two similar apps:

1. WaterMyPlant

Advantages:

- Achievements
- User friendly design
- The desired amount of milliliters of water can be indicated
- Has an option to add a picture of the plant

Disadvantages:

- Only one type of care
- Can't undo action
- No plant care tips
- No calendar

2. GreenS

Advantages:

- Plant care tips
- Lots of plant care options
- Various notifications
- Plant library
- User friendly design
- Has an option to add a picture of the plant

Disadvantages:

- No calendar
- Can't undo action

User interface design

Application Mockup

After studying similar applications, I chose the right colors and drew a preliminary mockup.

Garden page(Image1)
This page is the main one, which displays the user's garden. The user can see which plants need urgent care thanks to color indicators and which care actions should be taken. There are also buttons for adding a new plant, a calendar, favorite plants (from the plant library) and settings.

Plant creation page(Image2)
On this pop-up page, the user creates a plant to add to their garden. There are all the necessary settings for caring for the plant and you can also add the main photo.

Plant page 1(Image3)
Clicking on a plant in the garden will take the user to a page where they can see the name of the plant, photo, life expectancy, care information, and also an edit and back button.

Plant page 2(Image4)
You need to toggle the switch to see plant information: recommended light, temperature, humidity and description.

Calendar(Image4)
This page was created for the convenience of viewing the history of all plant events. On this page, you can change the actions in the past if the user forgot to mark it in the application or did it by accident.

Mockup testing

I have tested my mockup on a few friends who are planning to use this app in the future. It helped me look at my chosen design with different eyes. I have already corrected some of the shortcomings.
Issues that will be fixed in the next update:

- Enable light theme
- Adding custom notes to plants
- Achievements
- Plant growth history with photos

Team project steps

Project theme

After a team discussion of each proposed topic's advantages and disadvantages, we have chosen a plant care reminder app as the main project theme. In our opinion this topic will be very useful to most users, and also will help us to get acquainted with the basics of creating user interfaces in practice.

Survey with user

Our target audience is people that have house plants. Main activity of these people is to take care of their plants(watering, spraying, fertilizing and so on). Most people have more than one plant at home, which makes it difficult to properly care for each of them, since different types of plants need different types and frequency of care.

User needs

Due to the different types and frequency of plant care users need an application that will help them properly and timely take care of their plants.

Summary of all solutions

Our application should contain the following:

- Different types of plant care(watering, spraying, fertilizing).
- Plant care tips.
- Achievement system for instilling plant care in users.
- Changeable history of plant care.
- Flexible adjustment of the frequency of plant care.
- Option to add an image of the plant.
- No ads inside the app.

Design solution

Model

- **Achievements**
    - List<**Achievement**> achievements
    - int _wateringCounter
    - int _sprayingCounter
    - int fertilizingCounter

- **Achievement**
  - String name
  - String description
  - String svgPath
  - bool unlocked
- **CareType**
  - String name
  - int frequency
  - int frequencyIndex
  - int timeIndex
  - bool enabled
  - DateTime lastCare
  - DateTime nextCare
- **Plant**
  - String name
  - String description
  - String imagePath
  - **CareType** watering
  - **CareType** spraying
  - **CareType** fertilizing
- **Plants**
  - int plantsCounter
  - List<**Plant**> plants

View

After team discussion we extended our project mock-up to include all needed functionality.

Home page(Image1)

Home page of the application. It contains a list of owned plants cards with general information about the plant(image, name, information about next care activity) and button for care activity(if needed). Also there is a navigation bar at the bottom of the page and a new plant creation button at the top right corner.

Create plant page(Image2)

Plant creation page. It is a pop-up window that contains input fields for plant image, name, description and types of care.

Plant page(Image3,Image4)

Main window of the single plant. It contains all information about the plant(image, name, description, care types with information about next care activity and button for care activity). Also there is an edit button at the right top corner and calendar button.

Edit plant page(Image5)

Page for editing plant information. This page is similar to the plant creation page with one additional delete plant button.

Calendar page(Image6)

Calendar pop-up window. It contains history of all care activities of the plant. Green dots next to the date means that care is done, orange dots next to the date means that care is not done. On this page you can edit your plant care activity using buttons next to the care type.

Settings page(Image7)

Page for changing application settings. There are options to turn on/ turn off application notifications and switch application themes to dark/light.

Care tips page(Image8)

Page with some basic plant care tips. It contains cards with care tip title and brief descriptions.

Care tip page(Image9)

Care tip pop-up window. It contains the name of the care tip and its full description.

Achievements page(Image10, Image11)

This page contains application user rank and achievements. There are only achievements that the user got, but there is an option to show all available achievements. Achievements and rank concepts should stimulate users to be better owners of their plants.

Achievement page(Image12)

Achievement pop-up window that shows name of the achievement, its description and status.

Controller

Also during our team meeting we have discussed a draft version of the API. At this stage of the project description of the API is abstract, it should give a brief review of all activities with model data that will be done on the controller side of the project. API will be more specified and expanded during the implementation stage of the project.

- **AchievementController**
  - switchView
  - getViewSetting
  - getListToView
  - getRank
  - getProgressBarCoeff
  - leftToNextRank
  - nextRank
  - platinumEarned
  - updatePlantsAchievements
  - updateWatering
  - updateSpraying
  - updateFertilizng
  - unlockDescriptionAchievement
  - unlockRenameAchievement
  - unlockDeletedAchievement

    - unlockPlatinumTrophy
- **CreationController**
    - getSumbit
    - careChanged
    - setImagePath
    - getImagePath
    - validateName
    - getPlantNameController
    - validateDescription
    - validateCare
    - getPlantDescriptionController
    - getCareFrequencyText
    - updateCareFrequencyText
    - getFrequencyIndex
    - setFrequencyIndex
    - getTimeIndex
    - setTimeIndex
    - getTimeListLength
    - getTimeItem
    - UpdateFlag
    - getFlag
    - cancel
    - _transformCareFrequency
    - createPlant
- **EditController**
    - getSumbit
    - getChanged
    - careChanged
    - getCareFrequencyText
    - validateName
    - validateDescription
    - validateCare
    - updateCareFrequencyText
    - getFrequencyIndex
    - setFrequencyIndex
    - getTimeIndex
    - setTimeIndex
    - getTimeListLength
    - getTimeItem
    - UpdateFlag
    - getFlag
    - setImagePath
    - getImagePath
    - setPlantName

    - getPlantNameController
    - setPlantDescription
    - getPlantDescriptionController
    - _transformCareFrequency
    - initializeController
    - saveChanges
    - deletePlant
- **HomeController**
    - careTodayPlantNeeded
    - careTodayPlantsNeeded
    - counterPlantsToCareToday
    - whatNeedToDoToday
    - nextCare
    - nextCareDays
    - doCare
    - getPlantsSortedByNextCare
- **NavigationController**
    - getCurrentPage
    - getBucket
    - updateCurrentPage
    - getCurrentPageIndex
- **PlantController**
    - getChanged
    - getHeight
    - wateringEnabled
    - sprayingEnabled
    - fertilizingEnabled
    - wateringNeeded
    - sprayingNeeded
    - fertilizingNeeded
    - wateringNextCareDays
    - sprayingNextCareDays
    - fertilizingNextCareDays
    - water
    - spray
    - fertilize
- **SettingsController**
    - isDarkTheme
    - setDarkTheme
    - notificationsEnabled
    - setEnableNotifications

Responsibilities

All team members work on the same app, but on different parts of the app:

- Naumenko Maksim (xnaume01)
  - Plant create/edit page
  - Plant page
  - Navigation between pages

- Mikhailov Kirill (xmikha00)
  - Achievements page
  - Settings page
  - Notifications

- Kovalets Vladyslav (xkoval21)
  - Home page
  - Care tips page

Used tools

Application is implemented in **Flutter** (check [1]) framework which allows developers to create multi-platform applications in C-like Dart language. In terms of our application this framework lets us handle programs' back-end and implement it as an MVC model as well.
We chose Git as a version control system, which is a classic choice in applications as big as ours' is. Each team member had his own branch, where he was implementing his part of a project and then we was creating pull requests and after avoiding merge conflicts whole project was in the **main** branch
Also we chose a VSCode as an IDE, since this is the more common choice for each of us. The corresponding extension for **Flutter** framework left a good impression of working with framework in the chosen IDE, despite the fact that creators of **Flutter** implied the use of Apple's XCode IDE.

Implementation details

Most important from application's root directory :

- *assets* directory - contains all icons required for buttons and achievements.
- *ios* directory - contains specific definitions so programs can be able to build and run on **iOS**-driven systems. Root may also contain an *android* repo for the same reasons with **Android** devices, but our **iOS**-oriented application allows us to ignore it.
- *pubspec.yaml* - file, which defines project dependencies with their specific versions, assets source path and used **Flutter** framework version
- *lib* directory - directory, which contains the whole application itself, including ***main.dart*** and *models, views, controllers, services, constants, get-in-mixins and utils* directories in our case.
    - *get-it-mixins* (check [2]) contains a ***locator.dart*** file with declarations of "global" singletons so we can easily access controllers or models from different parts of code.

Below you can find description of every specific part of our application:

- Home page
    - Home page can be divided into conditional 2 parts:

      1) The top bar of the page, which contains up-to-date information about the state of planned care for today, as well as an add plant button that calls the *bottom modal sheet [3]*.

      2) The body of the page, which displays the entire "garden" of the user. The main part is implemented using the *ListView.builder* widget [4] that generates a scrollable linear array of widgets without "duplicating" code.

      The controller for this page is the **home_controller.dart** file, which allows you to organize plant cards according to the need for planned care, as well as the implementation of plant care itself.

- Plant page
    - This page contains all information about a plant (name, description, image and care type activities). Moreover we are able to perform many actions with the plant from this page. There are buttons on the bottom side of the Plant page to register a care of it. At the top right corner there is a button for the plant editing. When we have done everything we need - we can return back to the home page by clicking on the button in the

top left corner. All information and actions of this page are controlled by the **PageController** that connects View with Models information.

- Plant create/edit page
    - These two pages have almost the same user interface and functionality. The same items are: text fields for the name and description, 3 widgets for configuring types of care and image picker widget. Two different controllers are used for these pages to make them active. **CreationController** provides an API to validate and add a new plant to the **Plants** model. **EditController** provides an API to validate and save changes that were made with the plant. Furthermore this controller allows us to delete plant.

      For image widget two Flutter plugins are used (image_cropper and image_picker). **ImageHelper** provides an API for the image management and **ImagePicker** represents the View of the image widget.

      CareConfiguration is used for the care type widget. It has parameters that help us to configure different types of care with only one widget, that is a great opportunity for the further extensions of the application.

- Achievements page
    - Achievement page meets the user with a rectangle with rounded angles, which contains information about the user's actual rank and how many trophies are left to get the next one. Every string with actual rank or string with a number of left trophies is counted and returned from the **AchievementController**.

      Also there's a progress bar below, which is implemented via built-in **Stack** widget, which allows us to create two objects superimposed on each other, so there's a nearly transparent green rectangle and then green progress bar. Coefficient for changing the length of a bar is also returned from a controller reminded above.
      Every new rank requires **3 new achievements** to be reached

      Achievement list and achievements themselves are described in corresponding **achievement** and **achievement_list** models.

      Each trophy contains a flag, which signals if the trophy is achieved with provided getters and setters.
      Following achievement list is implemented with the help of **GridView** built-in widget. Every achievement plate is pushable and after push some information about a trophy will appear (via **showModalBottomSheet()** function). Color of a plate depends on achievement status - if it is achieved it becomes green, otherwise it's gray.

      By default, you see only achieved trophies on this page, but you can view all available achievements via the "**Show All >**" button. Correct viewing of is also solved in **AchievementController**. By pressing the

button reminded above, the user will switch the private **view** flag. Then a function from the same controller sends a list of required achievements to be printed back to the view.

- Settings page
    - In this page users can turn on or turn off **Push Notifications**. This is a classic Apple-styled looking settings page. An Apple-style slider was implemented with help of Cupertino Library (check [5]). I set a switching value to the **CupertinoSwitch** widget "*value:*" attribute and then when switch state changes, the required flag from **SettingsController** will be changed by function from the same controller. Unfortunately, we've encountered some language and library issues while implementing turning notifications On/Off and sometimes functionality is not confirmed.
- Notifications
    - This part of the program is implemented in path ***/lib/services/local_notification_service.dart*** and implemented with the **flutter_local_notifications** package (check [7]). I followed the examples on sources [7] and [6], but I've done a few changes according to our requirements. Our **local_notification_service** allows you to create two types of notifications : instant and scheduled (delayed with use of [8]). Both of those functions are implemented in the path from above of this subsection.
      All I needed was to implement my custom class **LocalNotificationService** and all following functions with a ***setup()*** method and then I was able to call any function of ***showNotification()*** or ***showScheduledNotification()*** and create **async** notifications.

- Care tips page
    - The tips page contains three widgets that, when clicked, will bring up a *bottom modal sheet* with relevant information. The implementation of these widgets uses conditions to replace the corresponding text to avoid unnecessary code repetition.

Screenshots of the resulting application

Check the Images section.

Test report

On whom was it tested

Testing was conducted on friends and relatives whose age ranged from 15 to 55 years. We made this choice to understand what is missing in our application for each age group. Some of them were interviewed at the time of creating the mockups and they stated that they plan to use this application. The rest saw and tried this application for the first time.

How was the testing and what exactly was tested

The application was created primarily for **iOS**, and therefore, if the tested user has a device with this system, then we installed our application on his device and asked him to use it for a couple of days. In this case, we managed to get long-term feedback, which shows whether the user has enough functionality of the application and whether he finds it useful.

In the case when the tested user didn't have such an opportunity, we gave him our device, but for a shorter period, in order to immediately observe his reaction. We tested how the user works with the application from the first minutes, whether there are cases when he doesn't understand how to produce any application script and whether he likes the developed design.

Metrics used to measure the quality / usability of the resulting solution compared to the reference solution

First of all, the satisfaction of the user from working with our application was important to us. All interviewed users stated that they were satisfied with the developed design. Two out of six users said that the app lacks the ability to turn on night mode.
We were inspired mainly by the best of the reference solution, but of course we were also looking at other noteworthy applications and tried to take the best parts of them to our project.
We and our users think that we have succeeded in this task and also brought something new to applications of this kind - system of ranking and achievements, which captivates the user and immerses him in the plant caring process on a different level than just a duty.

Specific outputs from testing and specific suggestions for modifications and further development.

Generally, our team is satisfied with the result of our work. We have adopted the best of the reference application and added more immersivity and gamification to the plant care process.
Main suggestion for modification for our application is the dark mode, of course. We've planned to implement it, but made a little mistake with **style** structure prediction and to add this feature, we would have to rewrite a lot of code manually, which was unacceptable for us due to project deadline.

Literature and Sources

[1] Google and community developers. October, 2022 : Flutter - Build apps for any screen. [seen 2022-12-14]
Available on: https://flutter.dev/

[2] Flutter Community. November, 2022 : get-it-mixin package. [seen 2022-12-14]
Available on: https://pub.dev/packages/get_it_mixin

[3] Flutter Community. September, 2022 : modal_bottom_sheet package.
[seen 2022-12-14]
Available on: https://pub.dev/packages/modal_bottom_sheet

[4] GeeksforGeeks Community. June, 2022 : Listview.builder in Flutter.
[seen 2022-12-14]
Available on: https://www.geeksforgeeks.org/listview-builder-in-flutter/

[5] Google and community developers.November, 2022: Cupertino Library - Dart API.
[seen 2022-12-14]
Available on: https://api.flutter.dev/flutter/cupertino/cupertino-library.html

[6] GeeksforGeeks Community. March, 2022 : Flutter – Schedule Local Notification using Timezone
[seen 2022-12-14]
Available on: https://www.geeksforgeeks.org/flutter-schedule-local-notification-using-timezone/

[7] Google and community developers.November, 2022: flutter_local_notifications.

[seen 2022-12-14]
Available on: https://pub.dev/packages/flutter_local_notifications

[8] Google and community developers.September, 2022: timezone | Dart Package.
[seen 2022-12-14]
Available on: https://pub.dev/packages/timezone

Images

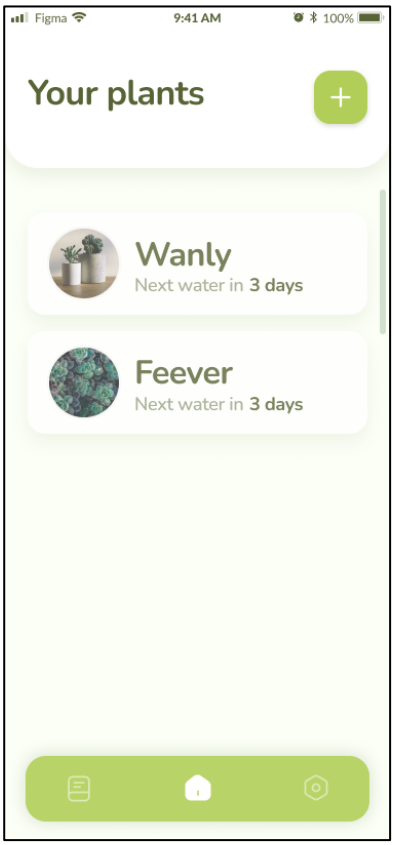

Figure 1 Home page

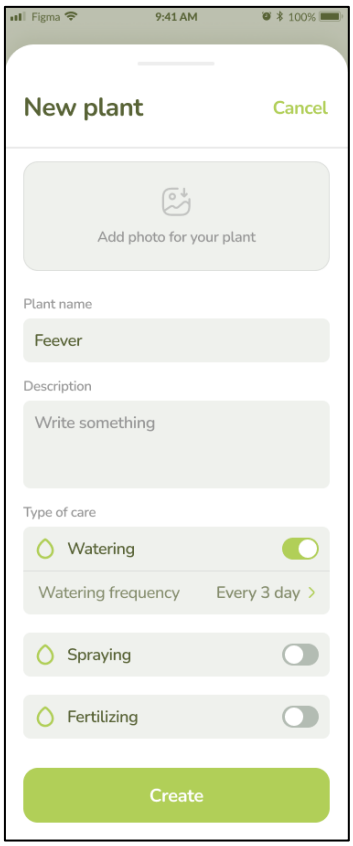

Figure 2 Plant creation page

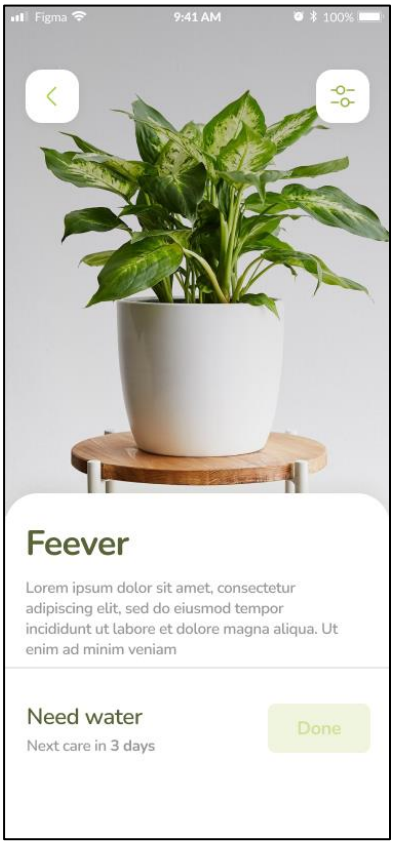

Figure 3 Plant page

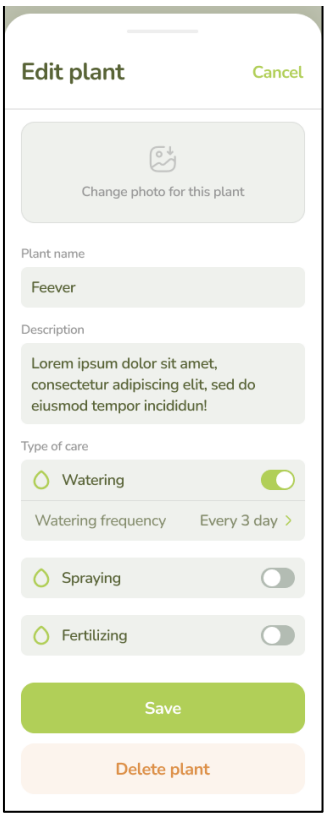

Figure 4 Plant edit page

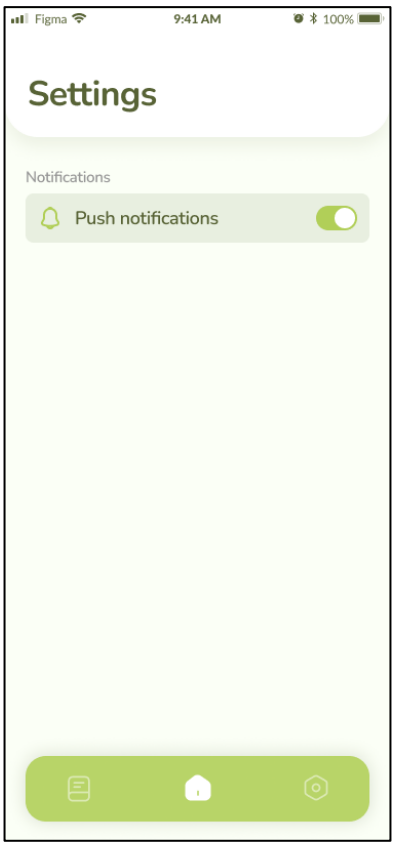

Figure 5 Settings page

Figure 6 Care tips page

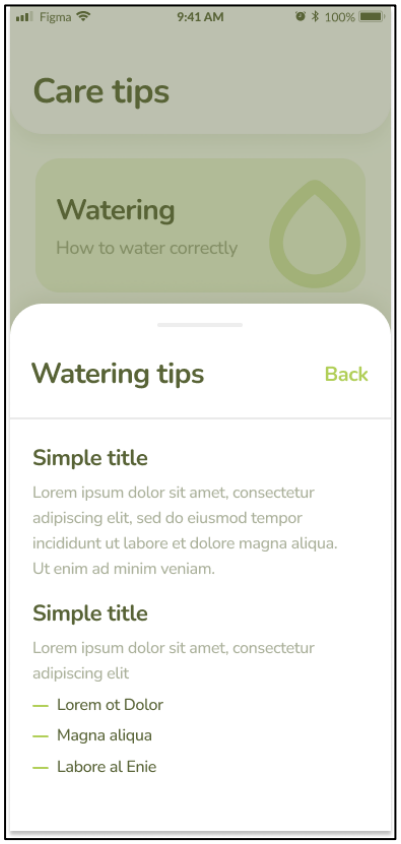

Figure 7 Care tip page

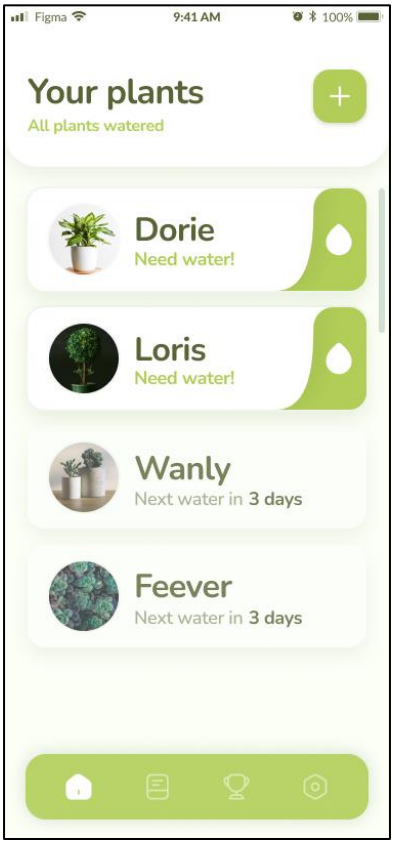

Figure 8 Home page

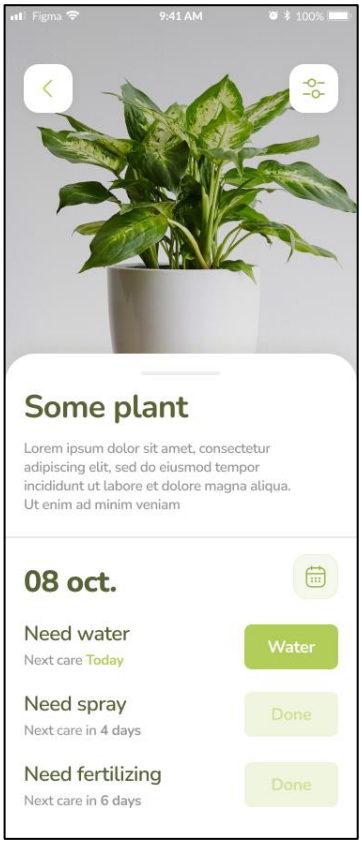

Figure 9 Plant page

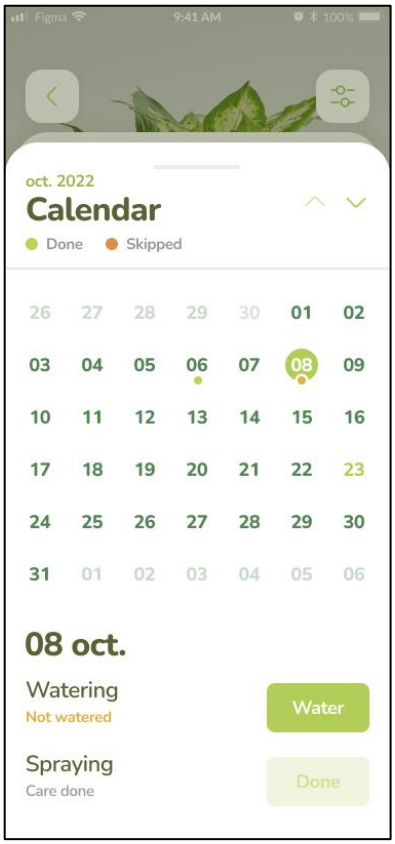

Figure 10 Calendar page

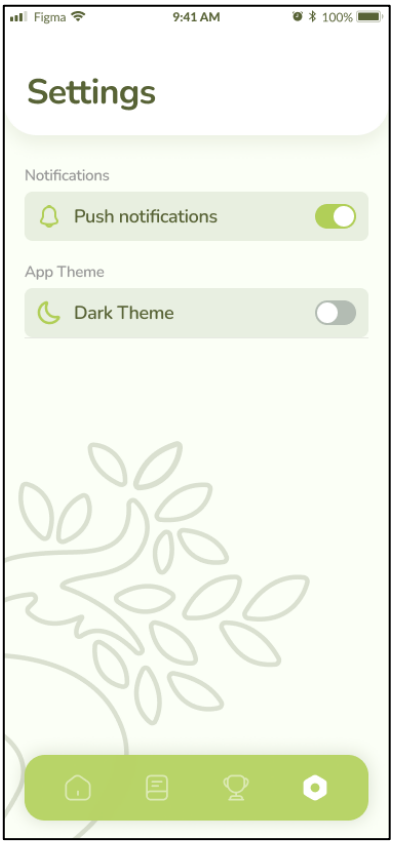

Figure 11 Settings page

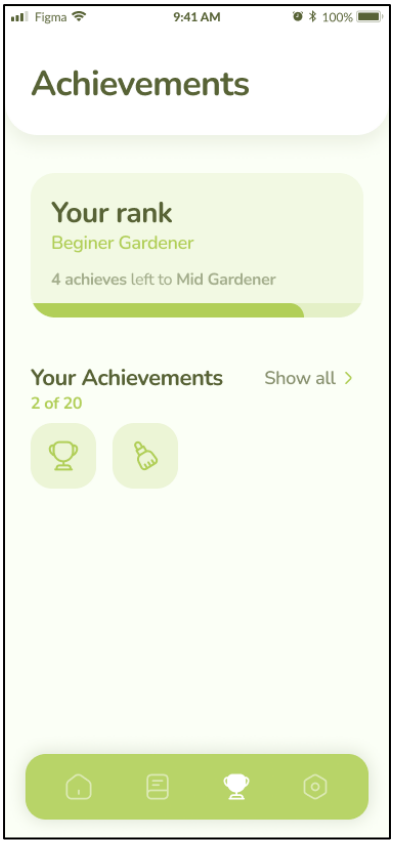

Figure 12 Achievements page 1

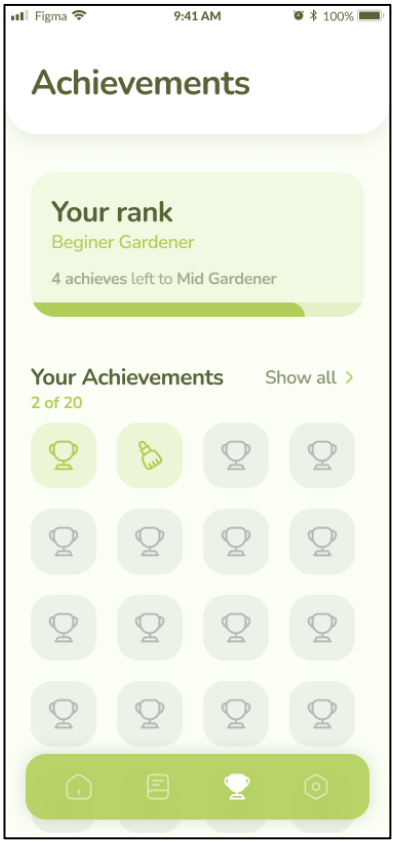

Figure 13 Achievements page 2

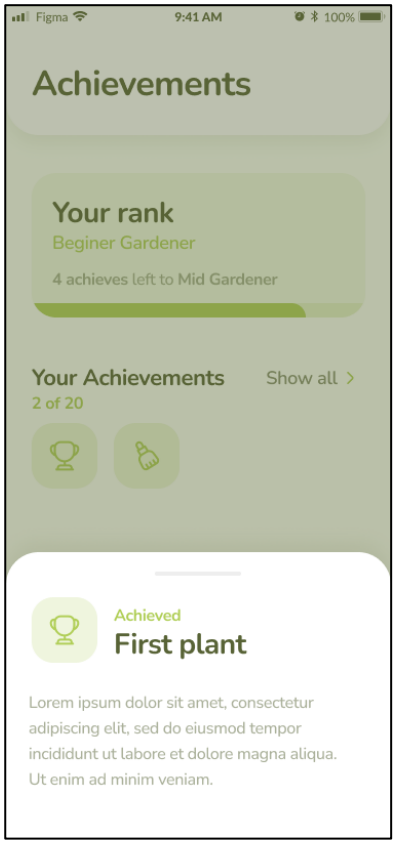

Figure 14 Achievement page

Figure 15 Components

Figure 16 Home page

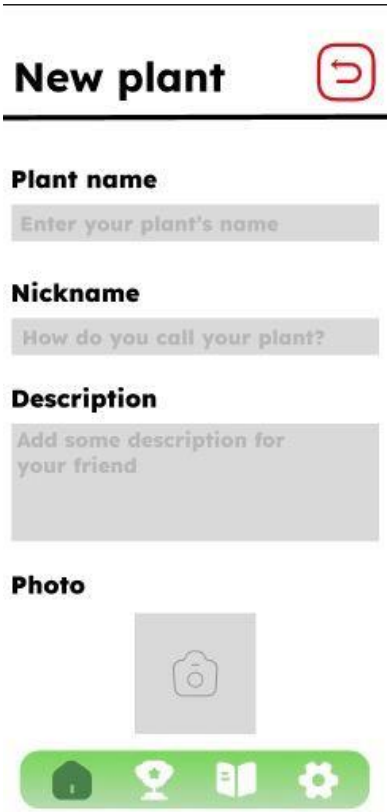

Figure 17 Plant creation page

Figure 18 Plant page

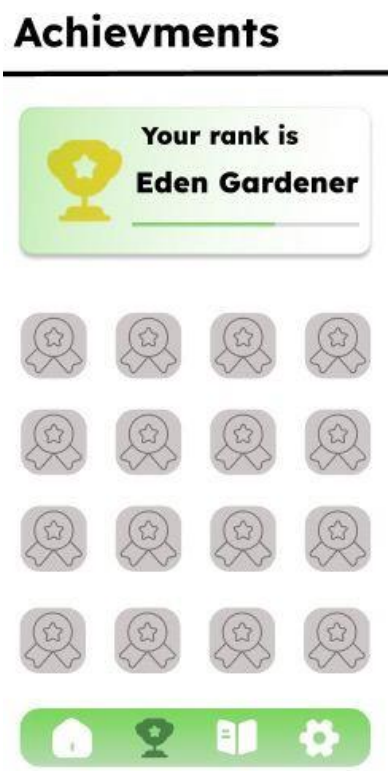

Figure 19 Achievements page

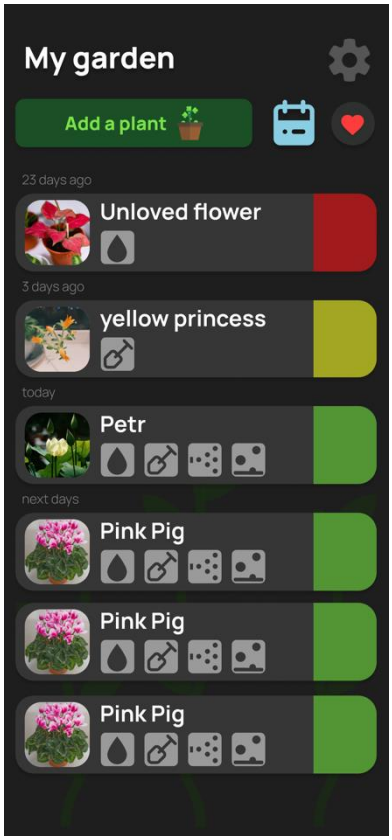

Figure 20 Home page

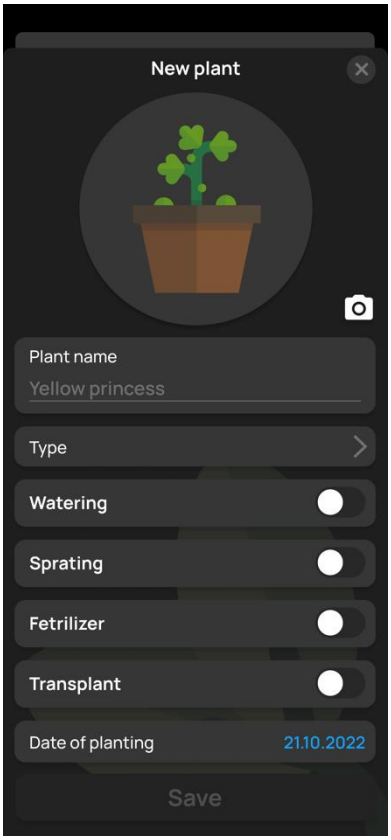

Figure 21 Plant creation page

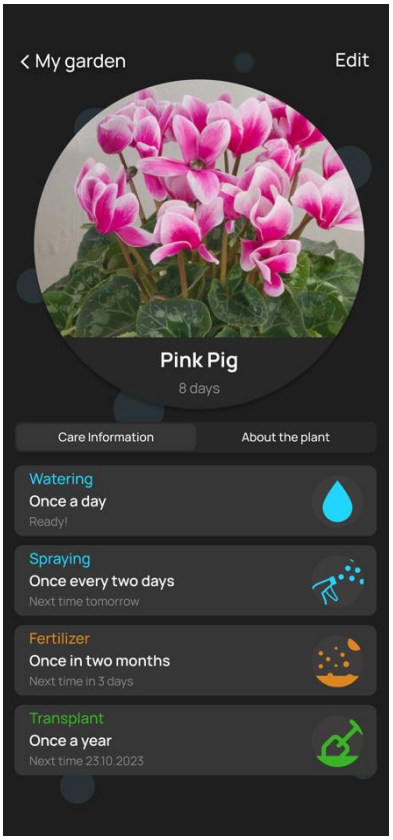

Figure 22 Plant page 1

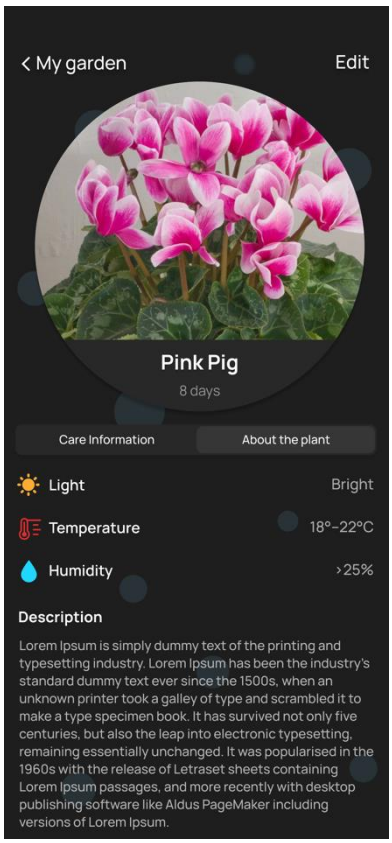

Figure 23 Plant page 2

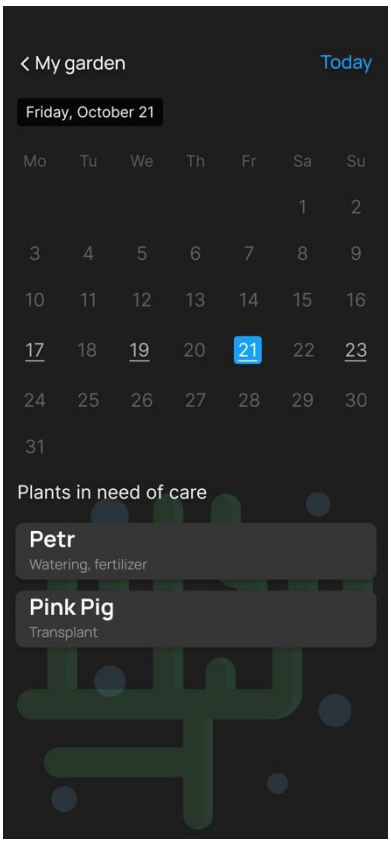

Figure 24 Calendar page

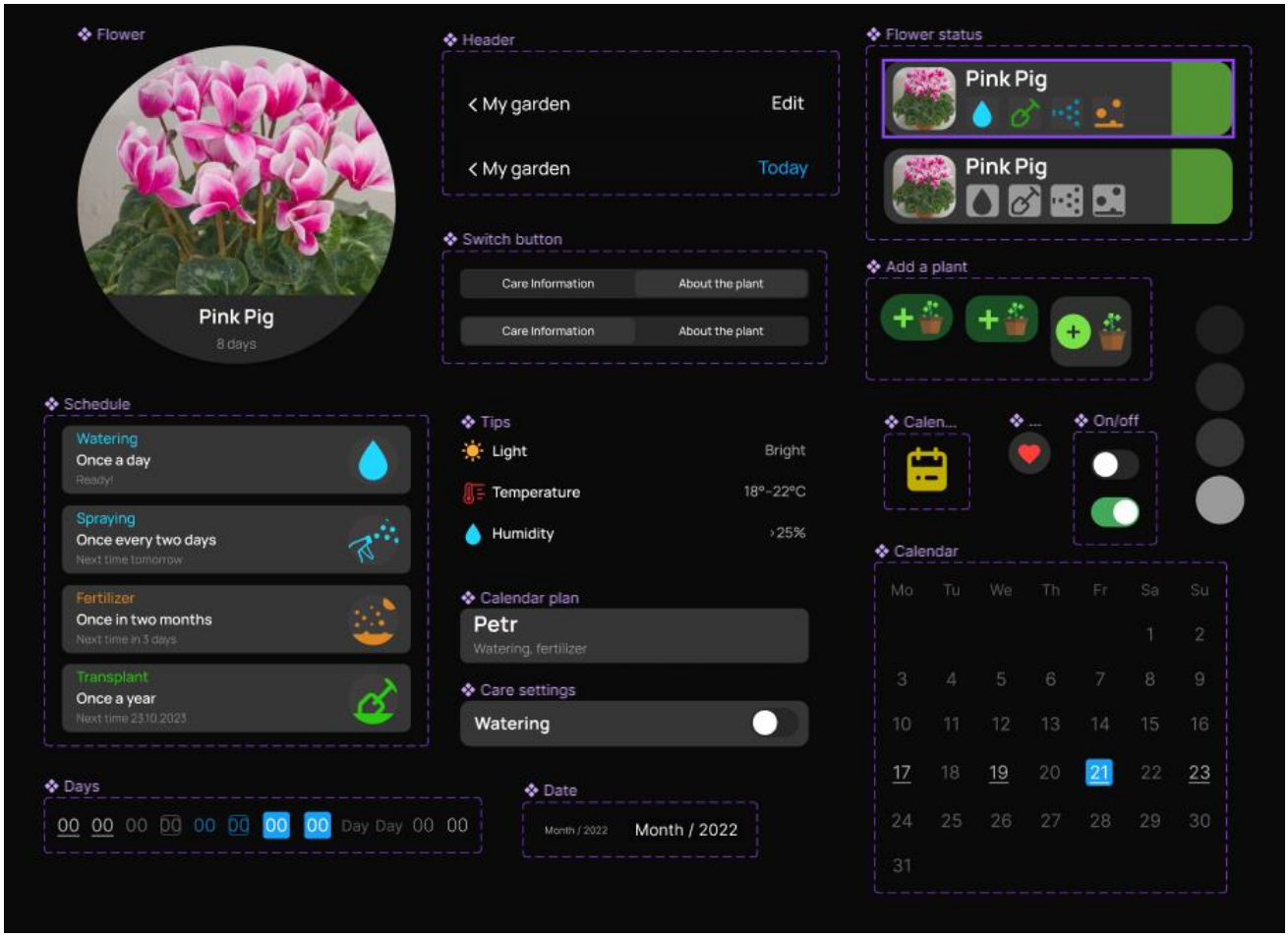

Figure 25 Components

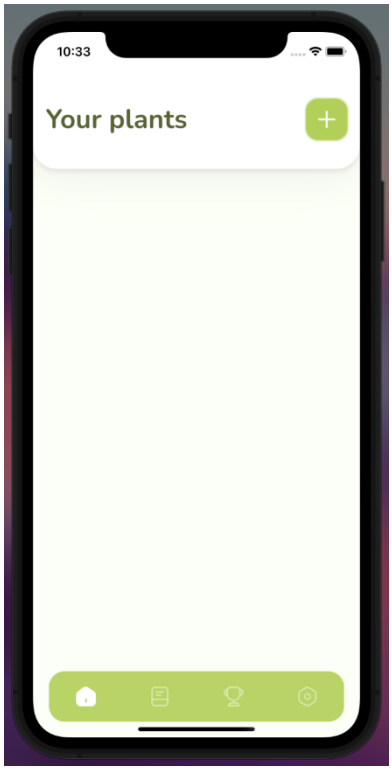

Figure 26 Home page

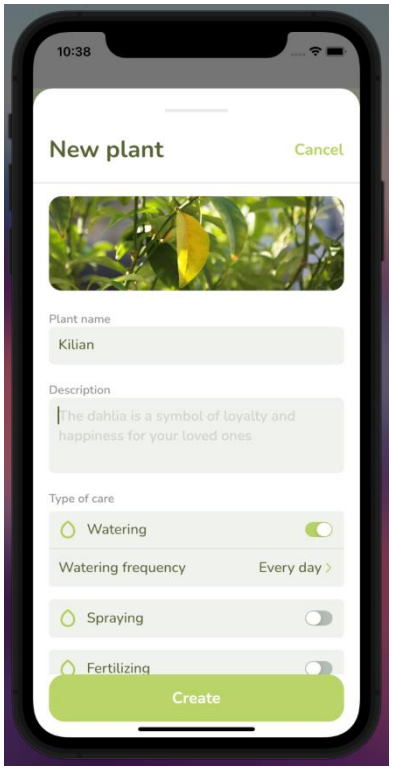

Figure 27 Create page 1

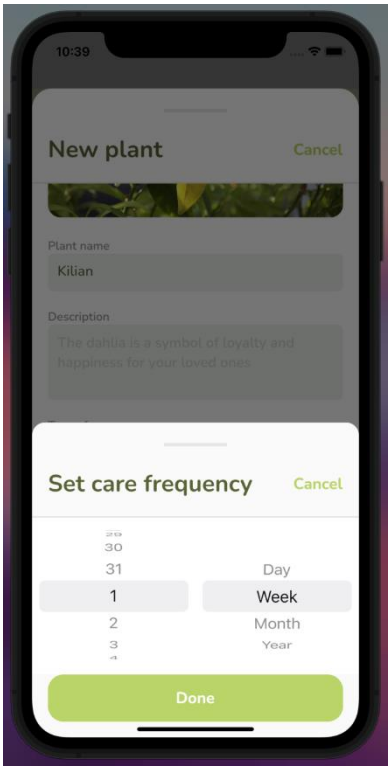

Figure 28 Create page 2

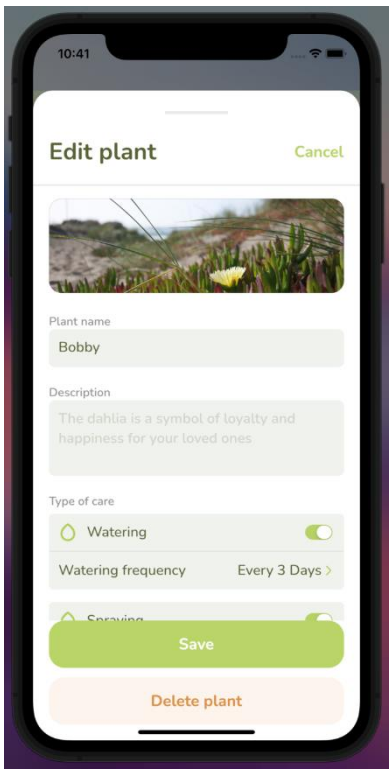

Figure 29 Edit page

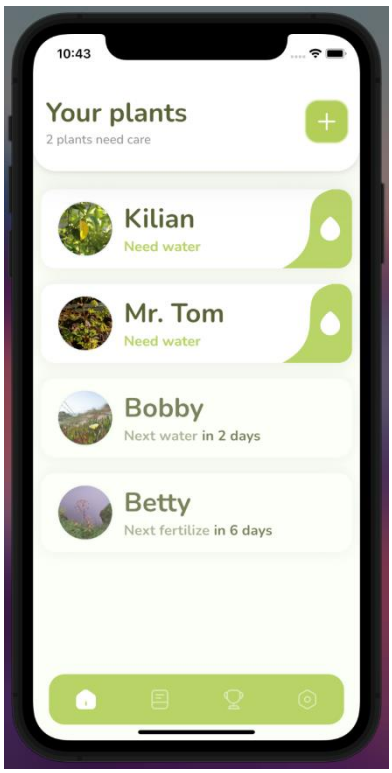

Figure 30 Home page 2

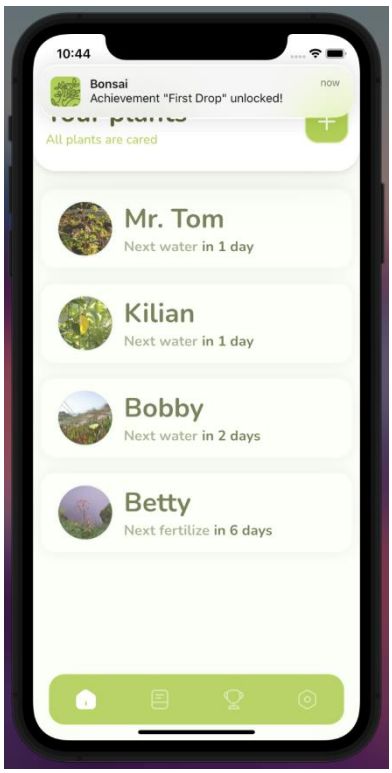

Figure 31 Notification

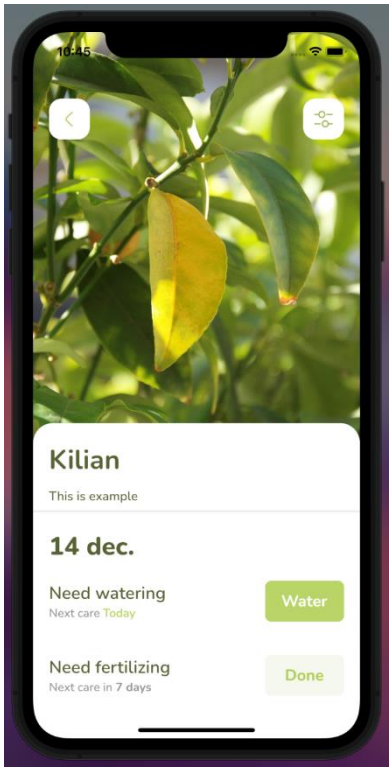

Figure 32 Plang page

Figure 33 Care Tips page

Figure 34 Watering tips

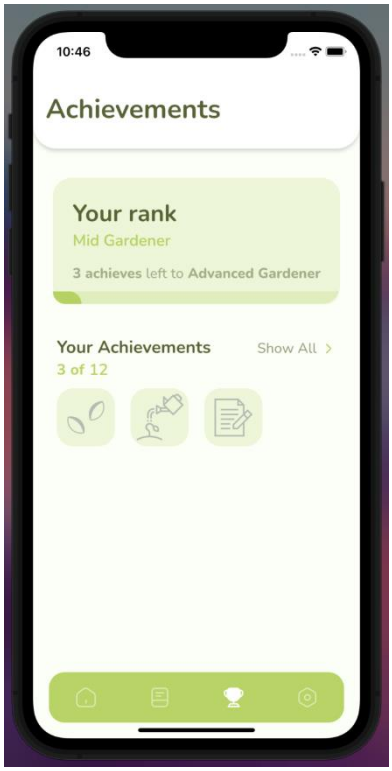

Figure 35 Achievements 1

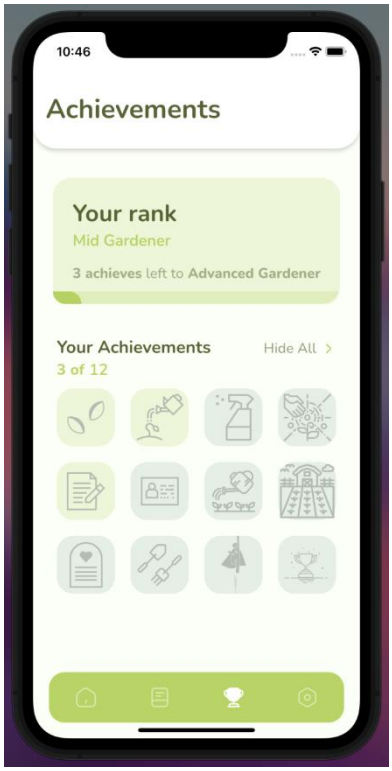

Figure 36 Achievements 2

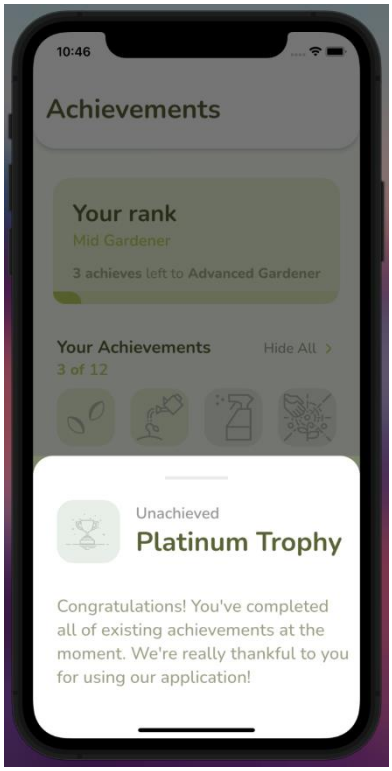

Figure 37 Platinum Trophy

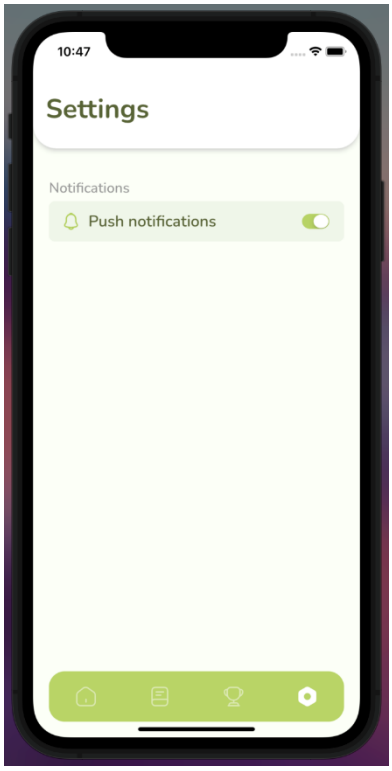

Figure 38 Settings page