INFO 5440 App Design & Prototyping

Project Milestone 4

Team Hot Pot (H.P.)

Han Gao (hg359), Kehui Guo (kg357), Hang Jiang (hj438),
Hongxi Jin (hj439), Daisy Liu (hl2445)

Table of Content

Final Design Sketch

Our group's final design sketches for the mobile and desktop version of the Playful Plants App are shown below, which covered three main features of our app: ***Plant Catalog***, ***Journal***, and ***Profile***.

Mobile Version

Plant Catalog

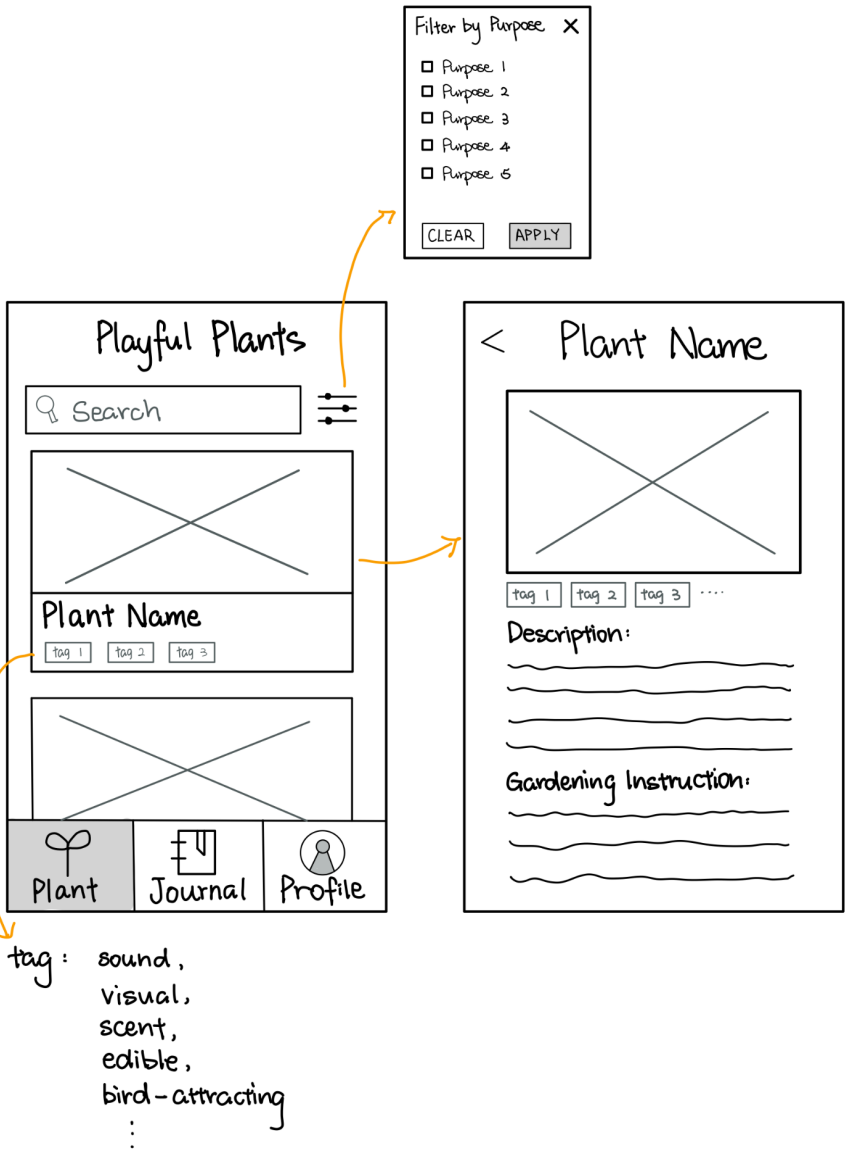


Image 1: *Plant Catalog* Page & *Plant Detail* Page

Journal

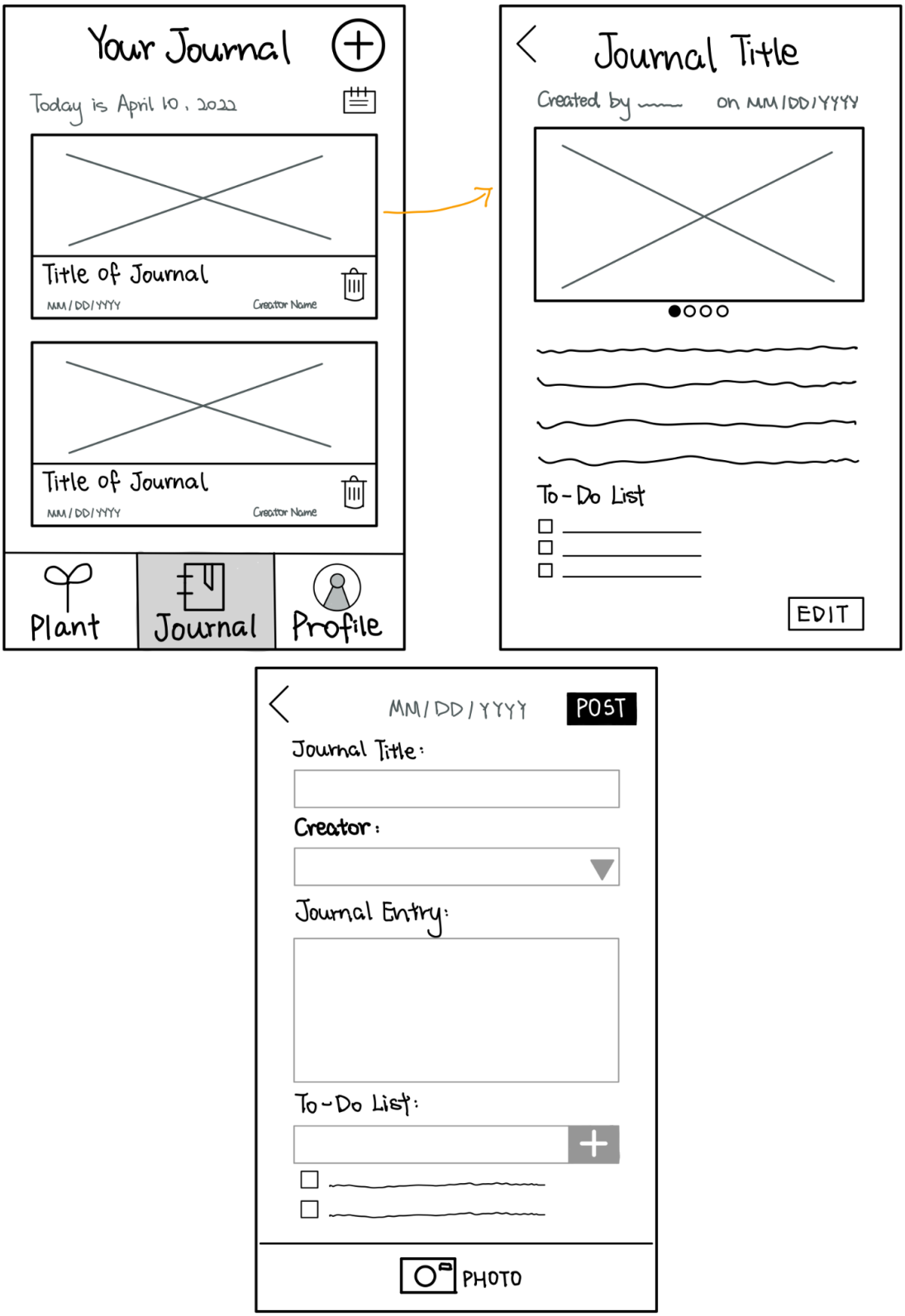


Image 2: *Journal Overview* Page, *Journal Detail* Page, and *New Journal* Page

Profile

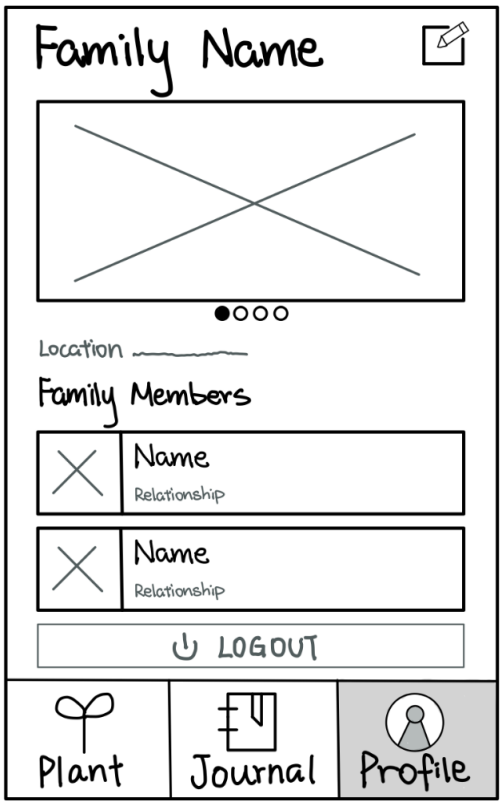


Image 3: *Profile* Page

Desktop Version

Plant Catalog

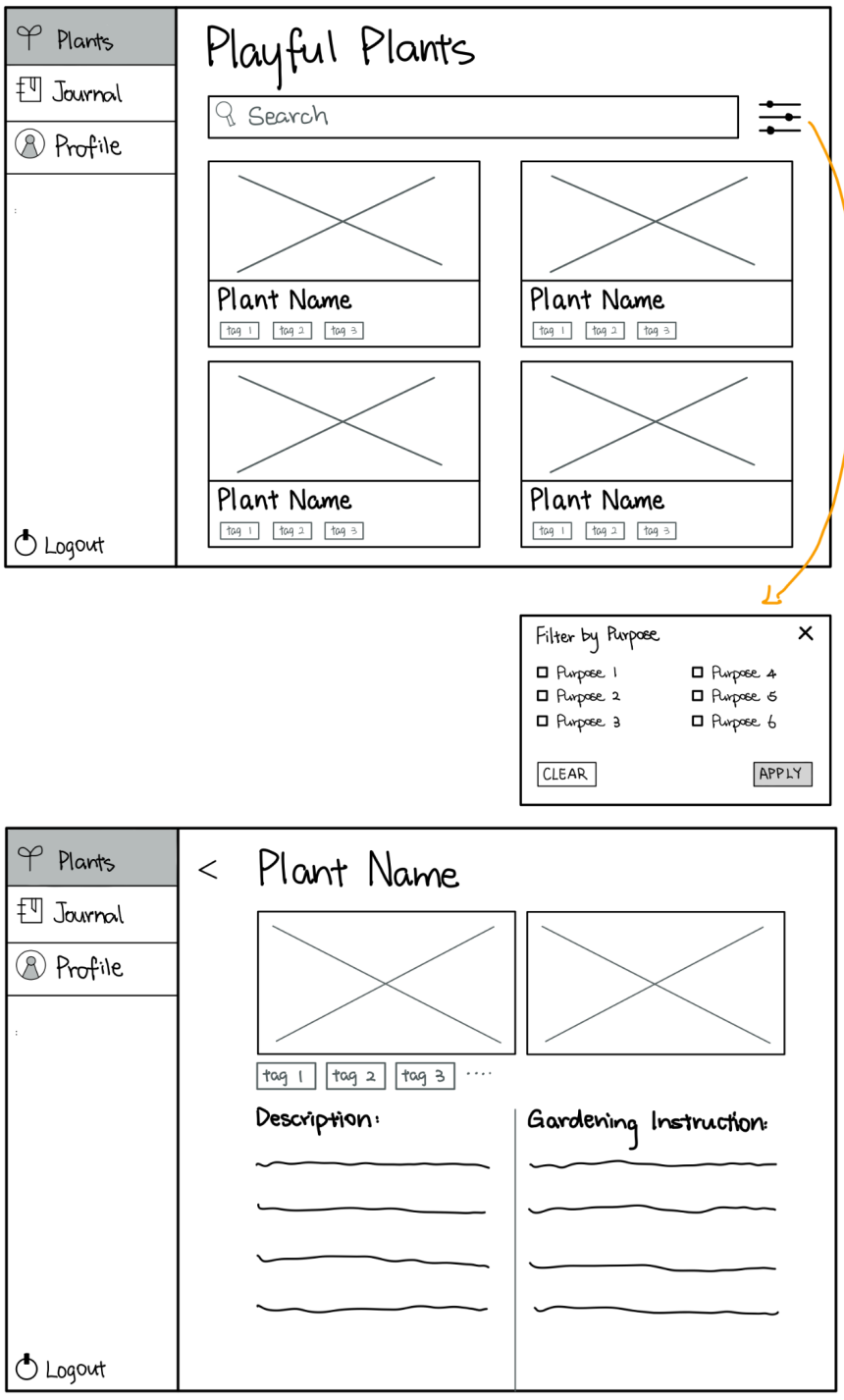


Image 4: *Plant Catalog* Page and *Plant Detail* Page

Journal

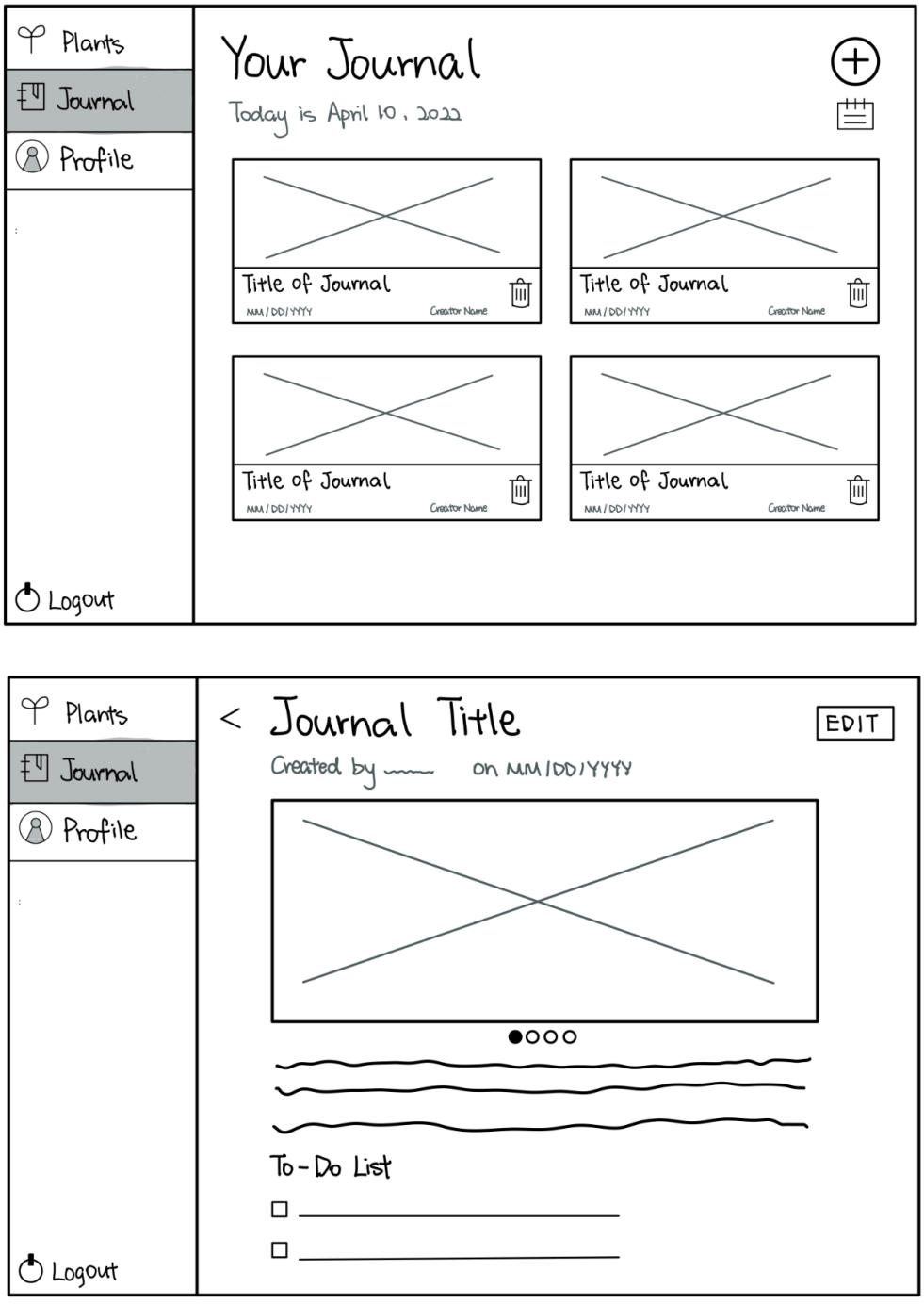


Image 5: *Journal Overview* Page and *Journal Detail* Page

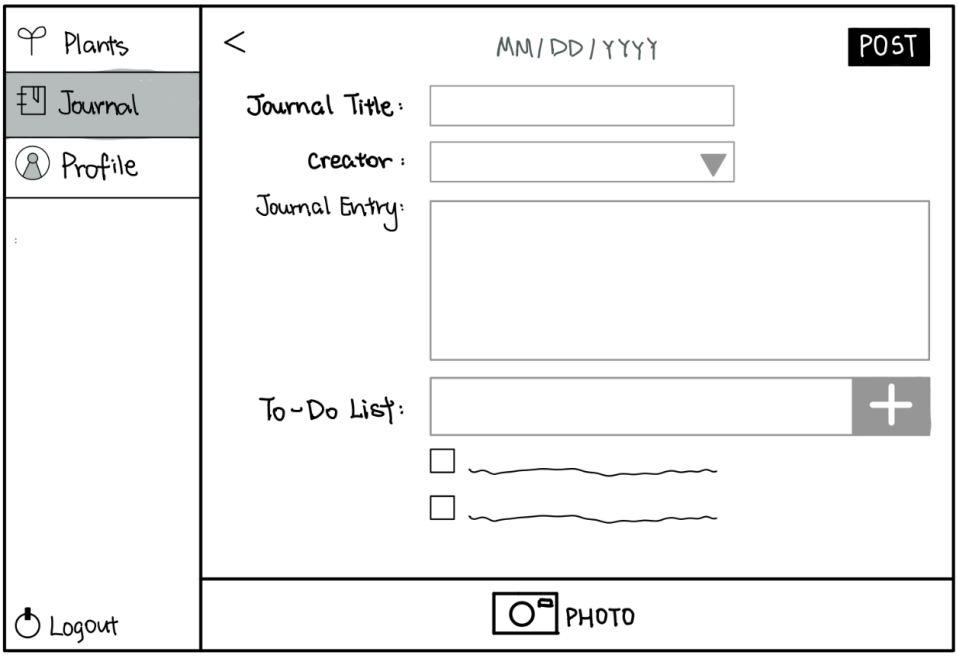


Image 6: *New Journal* Page

Profile

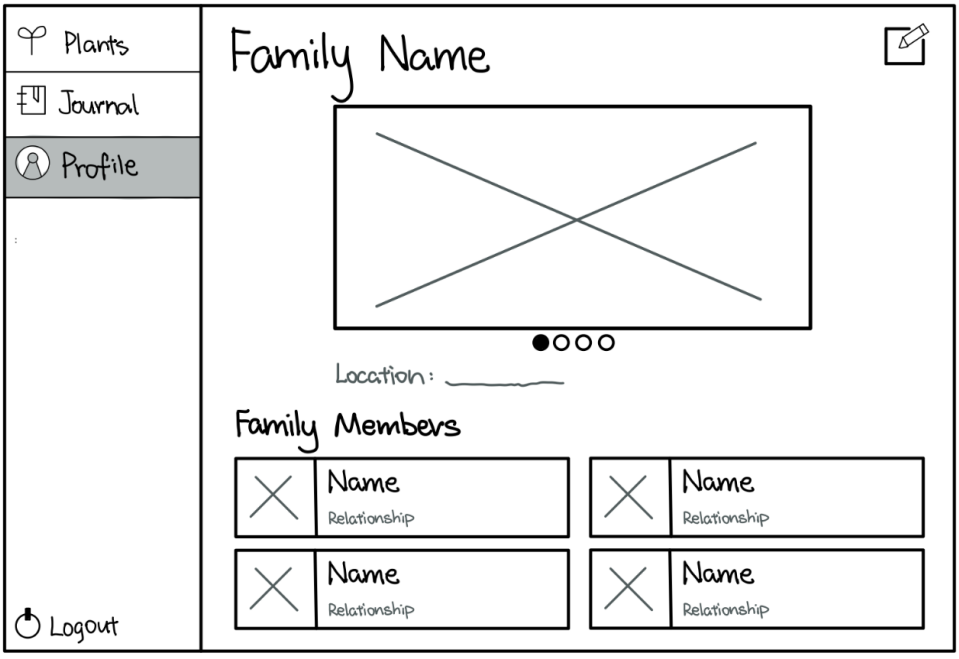


Image 7: *Profile* Page

Revisions

Based on the client's feedback collected from the Designing Phase, we made two major changes to our ***New Journal* Page** and ***Journal Detail* Page**: (1) we removed unnecessary advanced tools from the *New Journal* Page in both mobile and desktop versions, and (2) we slightly modified the layout of *Journal Detail* Page for the desktop version.

Revised *New Journal* Page

In the previous design sketch, as shown in the below image, our *New Journal* Page contained three advanced tools: photo-taking, voiceover, and drawing. Given the time limit, we decided to keep the most relevant tool, **photo-taking**, to our target users' journaling action.

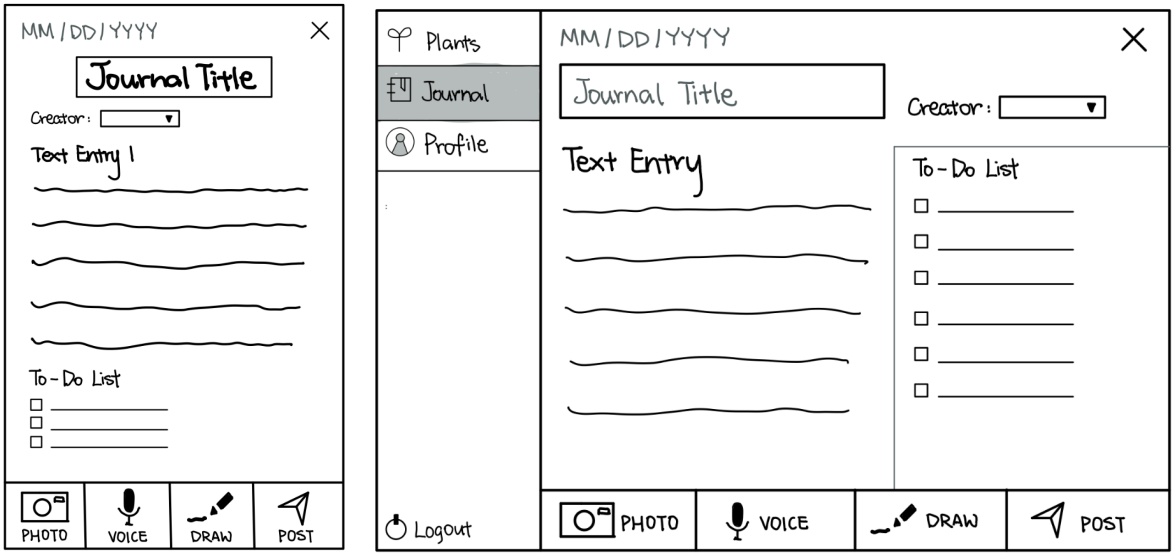


Image 8: Previous *New Journal* Page

Moreover, we previously grouped the Photo, Voice, Draw, and Post buttons at the bottom. Considering the Post button serves different purposes as other buttons do, we separated the Photo button from the Post button in two locations so that users can easily understand their different **affordances**. Our new design sketch is shown below:

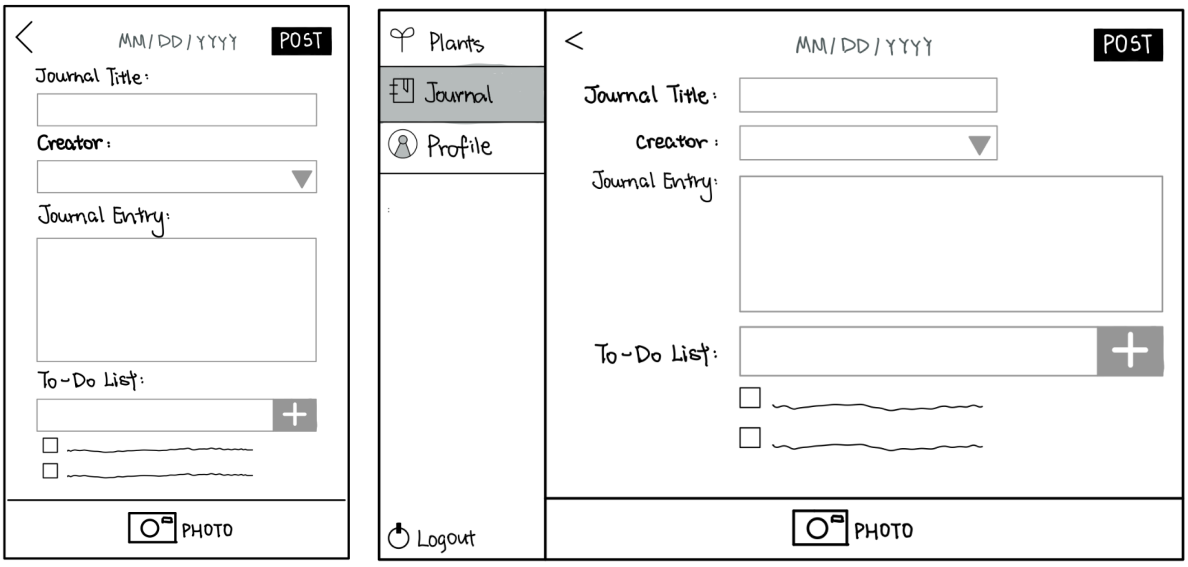


Image 9: Current *New Journal* Page

Revised *Journal Detail* Page

In our previous design, shown in the below image, the desktop version of the Journal Detail Page contains two columns. The left column includes the journal photos and text entries. The right column is for the to-do list.

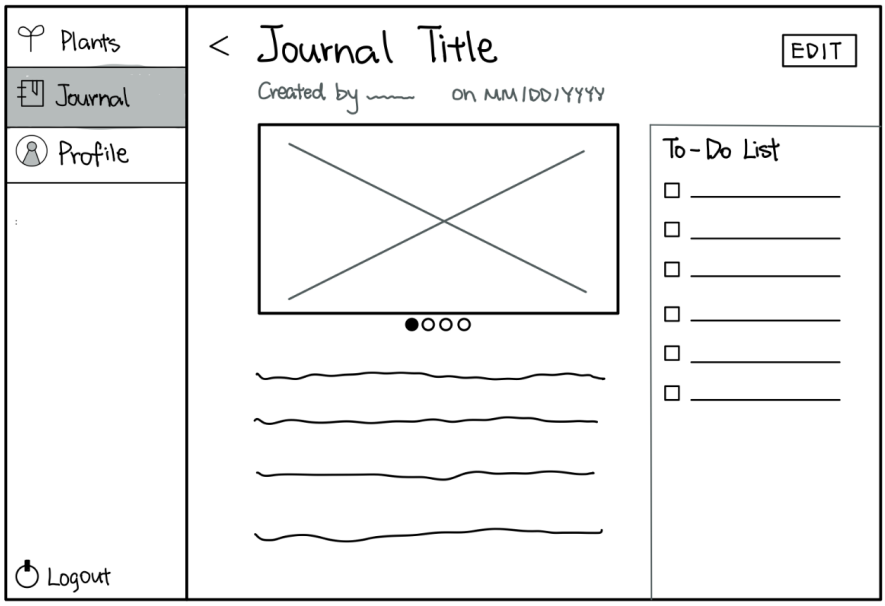


Image 10: Previous *Journal Detail* Page

However, this layout disturbs users' regular **reading** and **scanning patterns** from top to bottom. We decided to remove the second column and place the to-do list under the text entry, so users can quickly scan through the page and find the desired information.

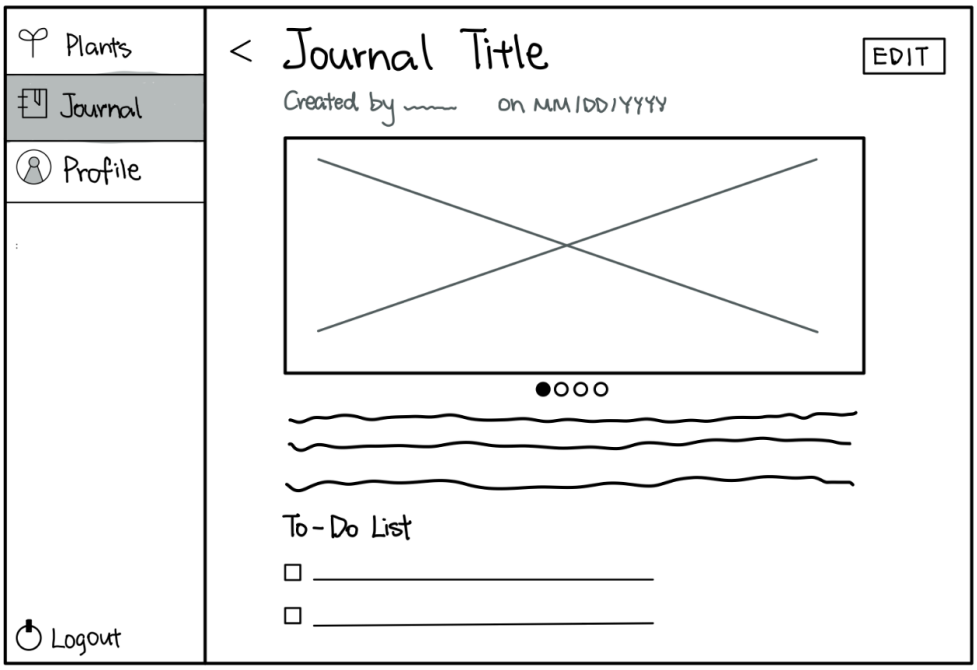


Image 11: Current *Journal Detail* Page

Implementation

In this milestone, our group implemented the above final sketches to prototype our most viable product (MVP) in Vue.js, which will be used later in usability tests.

Data

We created **three JSON data files** to store our data fixtures for plant information, fake journal entries, and fake user profiles.

Plant Information

Considering we need to display a plant catalog in our app, we made up information for **six plants** based on the actual Playful Plants database and stored them in the *plants.json* file. The basic data structure for a plant is shown below:


```
   "Autumn Brilliance Serviceberry": {
       "tags": {
           "tag1": "edible",
           "tag2": "colorful",
           "tag3": "nature attracting",
           "tag4": "medicinal"
       },
       "images": {
           "image1": "SH_02",
           "image2": "SH_02_1"
       },
       "description": "The berries are native to North America, northen Europe and
Asia. The berries are very nutritious and have long been harvested in the spring to
begin the summer berry season. Their shapes vary by seasons. In Spring, the berries
have white flowers, which gradually mature into a deep purple color. In summer, they
are edible. They taste like a combination of blueberries and apples, and ripen at same
time as cherries. In fall, the berries are blue-green foliage, orange-red. As a
durable plant, Brilliance Serviceberry is friendly for children to play with.",
       "instruction": {
           "Space": "Plant at least 20-30 ft apart in a loose triangular pattern",
           "Sunshine": "Provide enough sunshine with partial shade",
           "Maintenance Needs": "Prune after flowering",
           "Watering": "Provide a medium level of water"
       }
```

Its name, tags, image names, general description, and gardening instructions are stored as a JavaScript Object. To access all plant data, we declared a global variable **$plant_fixtures** in the *main.js*.

Journal Entry

Our app aims to enhance interactive family learning experiences by allowing users to build a family gardening journal. Therefore, the main feature of our app is journaling. Our group created **six fake journal entries** written in the tone of **our persona**, Emma, and her families, including journal titles, entry dates, creators' names, text entries, to-do lists, and gardening photos. These entries are stored in the journals.json file, and its basic data structure is shown below:

```
"journal 1": {
          "title": "Ready for the Garden Project",
          "date": "03/12/2021",
          "creator": "Lucas",
          "image": "journal 1",
          "text": "The construction of our garden is completely down today! Can't
wait for starting the gardening project with my daughter, Emma. This is my first time
of taking care of the garden. Hopefully, we can success and Emma can enjoy. By the
way, the wether is getting warmer these days. Probably, it's a good sign!",
          "to-do": {
              "task 1": "Create a shopping list for gardening tools",
              "task 2": "Search fo local farms to purchase seeds",
              "task 3": "Go shopping with Emma"
          },
          "checked": []
      },
```

Moreover, we created an empty array, "**checked**," to store completed to-do items. We also created an empty JavaScript Object, "**update**," to temporarily catch journal data from users' new journal entries and display them on our *Main Journal Page*. We declare a global variable **$journal_fixtures** in *main.js* to access all journaling data throughout the app.

User Profile

Last but not least, the one distinct feature of our app is the user account, which all family members share. To help users and later participants of our usability tests understand how they

and their families can use the app to build a journal with each other, we made up a fake user profile for our **persona**, Emma, and stored it in the profiles.json file. She can view her family name, family photos, family location, and all family members' basic information in her user profile. The basic data structure is shown below:

```
{
   "family name": "Emma's Family",
   "location": "New York",
   "family photos": {
       "photo 1": "family 1",
       "photo 2": "family 2",
       "photo 3": "family 3",
       "photo 4": "family 4",
       "photo 5": "family 5",
       "photo 6": "family 6"
   },
   "members": {
       "member 1": {
           "name": "Emma",
           "role": "mother",
           "photo": "Emma"
       },
       "member 2": {
           "name": "Lucas",
           "role": "son",
           "photo": "Lucas"
       },
       "member 3": {
           "name": "Richard",
           "role": "father",
           "photo": "Richard"
       }
   }
}
```

Similarly, we declare a global variable **$profile_fixtures** in *main.js* to access the profile data throughout the app.

Library

To build our app efficiently, we implemented libraries to support us building the application.

BootstrapVue

We leveraged the components and icons from the BootstrapVue library. We used the navigation bar and sidebar component to create our app navigation in mobile and desktop versions, card component for journals view and plants view, form component for journal input, checkbox component for the to-do list and filter list, etc. As for the styling, we used standard styling from the components and added additional styling using CSS. All icons are from the BootstrapVue library for styling consistency.

Navigation

We utilized the sidebar component for the desktop version for the navigation, showing vertical navigation on the left side of the website. As for the mobile app, following the convention for app navigation, we created horizontal navigation that stays at the bottom of the app using the navbar component.

```
<b-sidebar
  width="200px"
  class="side-nav"
  :visible="true"
  no-header-close
  :no-close-on-route-change="true"
  :no-close-on-esc="true"
  :no-close-on-backdrop="true"
  bg-variant="success"
  text-variant="light"
  shadow
>
```

Sidebar for Desktop

```
<b-navbar class="navBar" type="dark" variant="success" fixed="bottom">
  <!-- <b-navbar-toggle target="nav-collapse"></b-navbar-toggle>
b-collapse id="nav-collapse" is-nav> -->
  <b-navbar-nav class="w-100 ml-auto" align="center" justified>
    <!-- to Plant Catalog Page -->
    <b-nav-item class="navItem" to="/" exact>
      <b-icon icon="flower1" class="mt-3"></b-icon>
      <b-nav-text class="hnavbarp"> Plant </b-nav-text>
    </b-nav-item>
    <!-- to Journal Page -->
    <b-nav-item class="navItem" to="/journal" exact>
      <b-icon icon="journal-richtext" class="mt-3"></b-icon>
      <b-nav-text class="hnavbarp"> Journal </b-nav-text>
    </b-nav-item>
    <!-- to Profile Page -->
    <b-nav-item class="navItem" to="/profile" exact>
      <b-icon icon="person-fill" class="mt-3"></b-icon>
      <b-nav-text class="hnavbarp"> Profile </b-nav-text>
    </b-nav-item>
  </b-navbar-nav>
```

Navbar for Mobile

Plant/Journal Cards

Our design wants to show users a list of plants and journals in the main view. We used the card component (b-card) to create cards that include images, descriptions, and tags to achieve the goal.

```
<b-card
        :img-src="getImgUrl()"
        img-alt="Journal Image"
        img-top
        v-on:click.stop="ShowDetail(journal_id)"
>
  <b-card-title v-on:click.stop="ShowDetail(journal_id)"> {{ journal_name }} </b-card-title>
  <b-card-text id="date" v-on:click.stop="ShowDetail(journal_id)"> {{ journal_date }} </b-card-text>
  <b-card-text id="creatorName" v-on:click.stop="ShowDetail(journal_id)"> {{ journal_creator }}</b-card-text>
  <b-button id="delete" variant="outline-dark" size="sm" @click.stop="delClick">
    <b-icon icon="trash"></b-icon></b-button>
</b-card>
```

Filter Pop-Up

```
<b-modal
  id="filter-options"
  class="filter-popup"
  size="sm"
  centered
>
  <template #modal-header="{ close }">
    <h4>Filter by Purpose</h4>
    <b-icon-x-lg size="sm" @click="close()">
    </b-icon-x-lg>
  </template>

   <template #default>
    <b-form-group v-slot="{ ariaDescribedby }">
      <b-form-checkbox class="options"
        v-for="option in options"
        v-model="selected"
        :key="option.value"
        :value="option.value"
        :aria-describedby="ariaDescribedby"
        name="filterlist"
      >
        <p class="option_text"> {{ option.text }} </p>
      </b-form-checkbox>
    </b-form-group>
   </template>
```

We created a pop-up window for the filter feature using the modal component and listed down all of the filter options using the checkbox components. At the bottom of the filter pop-up, we used button components to show the actions that the users can do within the filter feature.

Add New Journal Form

On the *New Journal* Page, we utilized the form components to request users' journal inputs and create a new journal.

```
<b-form-input
  id="title"
  class="titlebox"
  v-model="title"
  v-on:change="changeGlobalTitle(title)"
  placeholder="Enter Journal Title"
></b-form-input>
```

```
<b-form-select
  id="creators"
  v-model="creator"
  class="w-100"
  required
  v-on:change="changeGlobalCreator(creator)"
>
```

Datepicker

```
<b-form-datepicker
  v-model="startDate"
  class="startDatePicker"
  size="sm"
  :date-format-options="{
    year: 'numeric',
    month: '2-digit',
    day: '2-digit',
  }"
  locale="en"
  :max="setMax()"
  placeholder="start date"
  reset-button
></b-form-datepicker>
```

In our journal view, we supported a date picker feature using the form datepicker component so that users can set the date range and find the journals within that date range quickly.

Router

Based on the result of our design phase, we decided to divide our prototype into three main parts: **plant** part, **journal** part, and **profile** page. The users can switch three pages by clicking the buttons on the navigation bar. The whole router is shown in the figure below.

Home (Plant List)

In *index.js*, we define the three routes for the plant part.

```
{
  path: '/',
  name: 'plant',
  component: PlantView
},
```

```
{
  path: '/plant/:plant_id',
  component: () => import('../views/PlantDetail.vue')
},
```

```
{
  path: '/plant/catalog/filter',
  name:'viewFilter',
  component: () => import('../views/FilterView.vue')
},
```

In the Home (*Plant Catalog*) Page, when users click a certain plant card, the *ShowDetail()* function would be called, and the certain plant id would be passed to the plant detail page.

```
v-on:click.native="ShowDetail(value)"
```

```
ShowDetail(plantId) {
  this.$router.push(`/plant/${plantId}`);
},
```

In the *Plant Detail* Page, we get the current plant id through the global variable *$route.params*. Then, we can get all information about the plant from the global variable *$plant_fixtures* by the key *curPlant*.

```
curPlant: this.$route.params.plant_id,
```

We can return back to the home page by clicking the "←" icon.

```
<router-link class="returnlink" to="/">
```

We also complete the plant tag filter functionality in a plant filter view. The following figures show the implementation detail.

```
v-on:click="ShowFilter"
```

```
ShowFilter() {
  this.$router.push("/plant/catalog/filter");
},
```

In the plant filter view, users can return back to the home page by clicking the "*close*" button.

```
returnback() {
  this.$router.push("/");
}
```

Journal List

In *index.js*, we define the four routes for the journal part.

```
{
  path: '/journal',
  name: 'journal',
  component: () => import('../views/JournalView.vue')
},
{
  path: '/journal/new',
  name: 'addNewJournal',
  component: () => import('../views/AddNewJournal.vue')
},
{
  path: '/journal/new/camera',
  name: 'openCamera',
  component: () => import('../views/cameraView.vue')
},
{
  path: '/journal/:journal_id',
  component: () => import('../views/JournalDetail.vue')
},
```

The journal list view demonstrates the list of journals created by family members before. In the journal list view, the users can perform two kinds of operations: (1) create a new journal and (2) review old journals.

When users click the "+" button, the page will route to the *addNewJournal* view.

```
<b-button v-on:click="AddJournal" size="sm">+</b-button>
```

```
AddJournal() {
  this.$router.push("/journal/new");
},
```

In the *New Journal* Page, users can call the camera on their device by opening the *openCamera* view.

```
<b-button class="cameraBtn" v-on:click="openCamera()"
```

```
openCamera() {
  this.$router.push("/journal/new/camera");
},
```

After taking photos, they can click the "*upload*" button and return back to the *addNewJournal* view.

```
v-on:click="returnAndSave()"
```

```
returnAndSave() {
  this.stopCameraStream();
  Vue.prototype.$photo = this.items[0].src;
  this.$router.push("/journal/new");
},
```

Users can "*post*" the new journal and then route back to the journal list view.

```
this.$router.push("/journal");
```

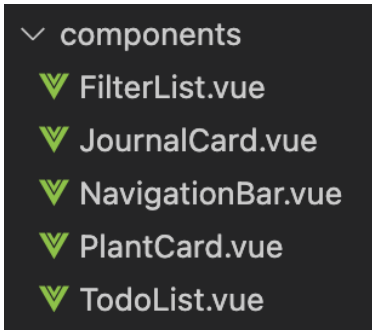
When users open the old journals, the *ShowDetail()* function would be called, and the certain journal id would be passed to the *journalDetail* page.

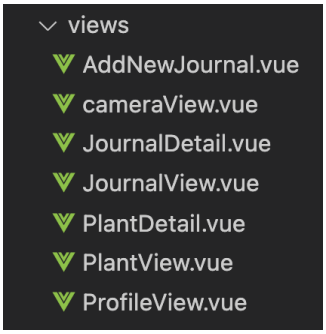
```
ShowDetail(journalId) {
  this.$router.push(`/journal/${journalId}`);
},
```

In *journalDetail* view, users can return back to the home page by clicking the "←" icon.

```
<router-link class="returnlink" to="/journal">
  <b-icon icon="arrow-left"></b-icon>
</router-link>
```

Features





In our App, we have **7 views** and **5 reusable components**. The navigation bar is consistent throughout the entire App but has two modes for desktop and mobile versions, and it is used in App.vue. The App can be broken up into three main parts: Plant, Journal, and Profile. The App is responsive on mobile, tablet, and desktop to provide accessibility for users like Emma and her family using media queries. We used **BootstrapVue components** for all the buttons, icons, forms, and grid systems.

Because this is only a high-fidelity prototype, not all functions are implemented. Here is a list of features that are ***usable*** on this prototype:

- The main *Plant* page (**PlantView.vue**) allows: 1) view all plants with PlantCard.vue, 2) filter plant information based on options on FilterList.vue, 3) click to see details by routing to PlantDetail.vue.
- The *Plant Detail* page **(PlantDetail.vue)** allows: 1) view detailed information about the specific plant, 2) return back to the main Plant page.
- The main *Journal* page (**JournalView.vue**) allows: 1) view all journal entries with JournalCard.vue, 2) filter journal entries based on datepicker, 3) delete existing journal entries, 4) click to see journal details by routing to JournalDetail.vue, 5) click to add a new journal by routing to AddNewJournal.vue.
- The *Journal Detail* page **(JournalDetail.vue)** allows: 1) view detailed content about the specific journal, 2) click on the to-do list with the TodoList.vue by marking a task as completed, 3) return back to the main Journal page.
- The *New Journal* page **(AddNewJournal.vue)** allows: 1) make a new journal entry, 2) add a to-do list, 3) add a photo using cameraApi.vue, 4) return back to the main Journal page.

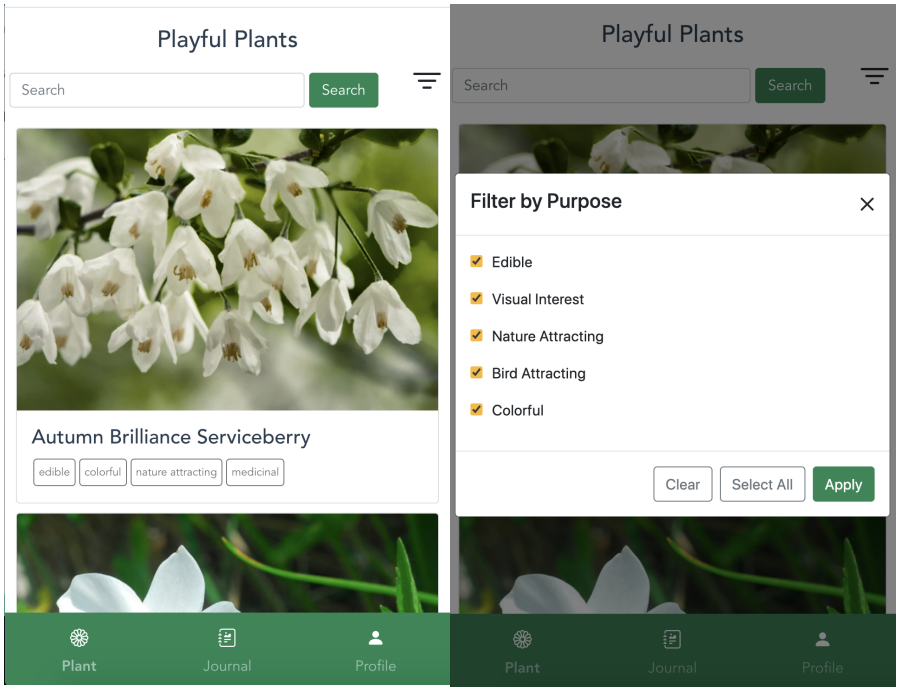
Because of time constraints and priority decisions, here is a list of features that are not functional on this MVP version:

- The search function on the main *Plant* page does not work, but users should be able to filter out plants based on certain criteria.
- Users should be able to edit their family profiles and log out of the account. However, because they are not the key features in the user scenarios and for testing the product goals, the Profile page is only viewable with no actual functionalities.

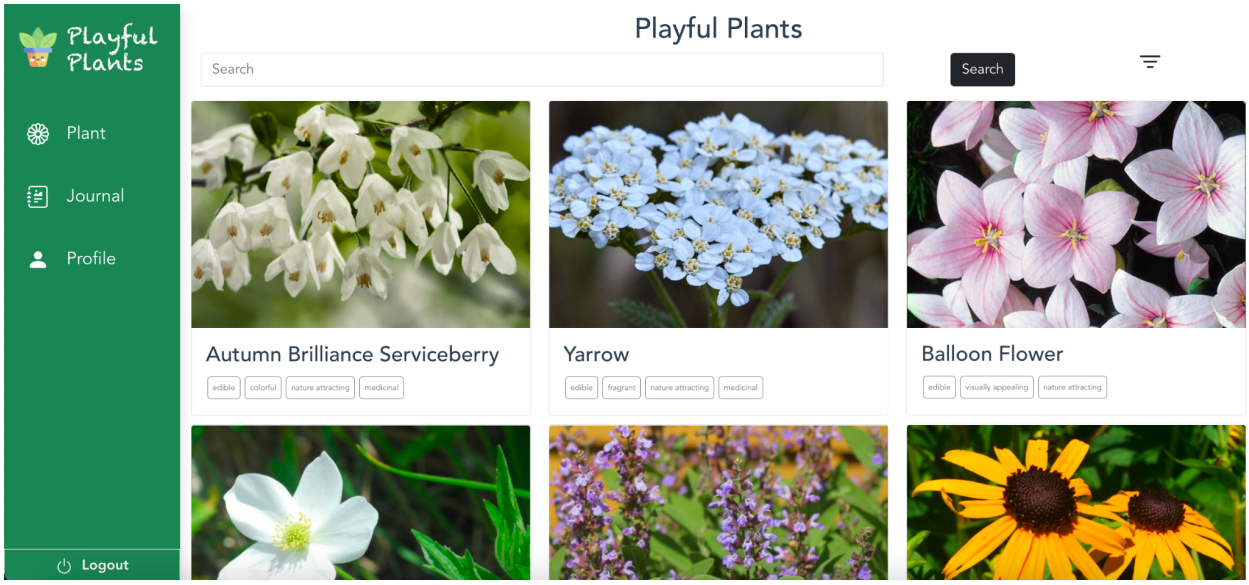
Plant

Main *Plant* Page

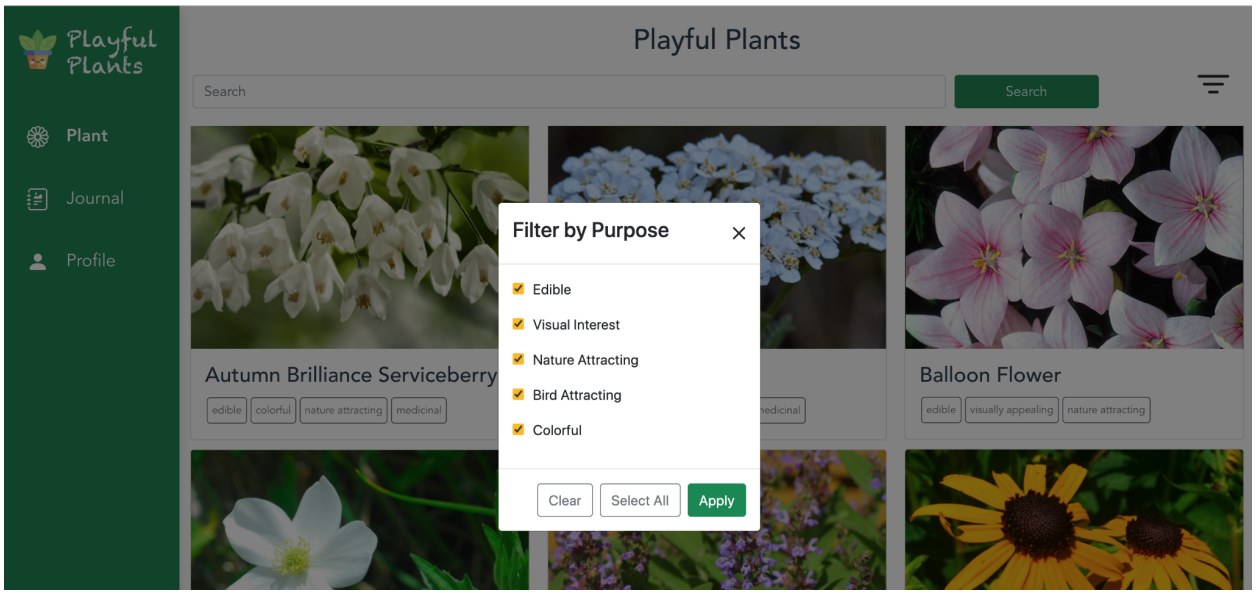
For the Plant part of this App, there are two main pages. The first one is the main Plant page **(PlantView.vue)**, where Emma and her family members can browse through the plant catalog and learn more about them. This corresponds to Scenario 1, where Lucas and Emma can learn about plants together with free access to the basic information about various playful plants. Besides browsing through them, users can also use the filter icon to filter out specific plants based on criteria such as edible, bird-attracting and so forth.



Mobile Frame: Main Plant page + Plant Filter



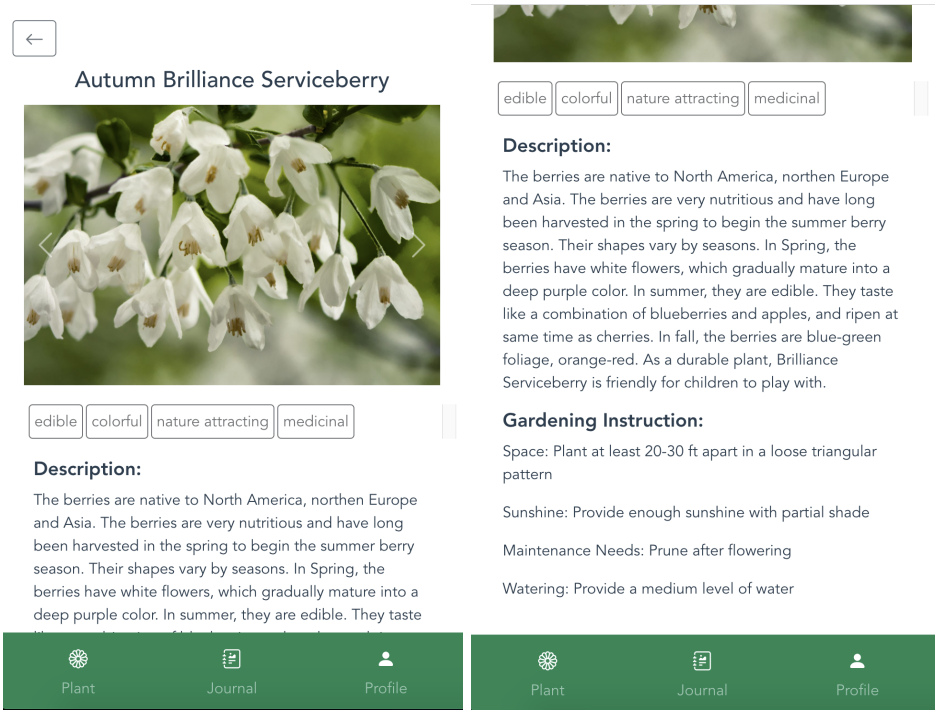
Desktop Frame: Main Plant page



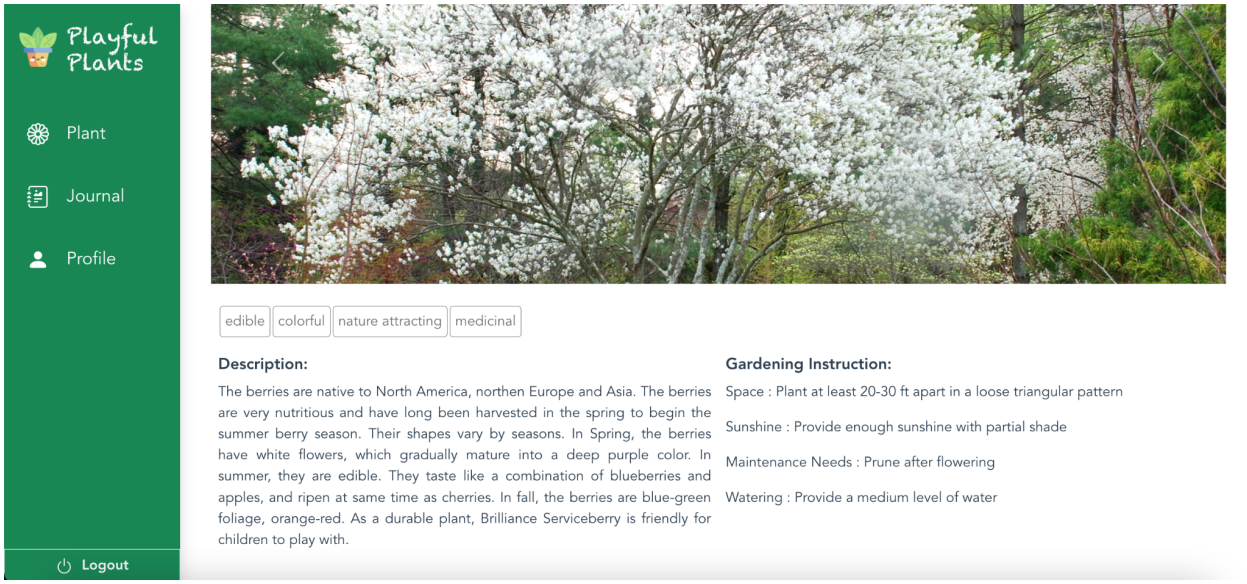
Desktop Frame: Plant Filter

Plant Detail **Page**

The second one is the Plant Detail page **(PlantDetail.vue)**, where Emma and her family can learn more detailed information about each plant individually. It contains the plant name, picture, tags, description, and gardening instructions.



Mobile Frame: Plant Detail Page

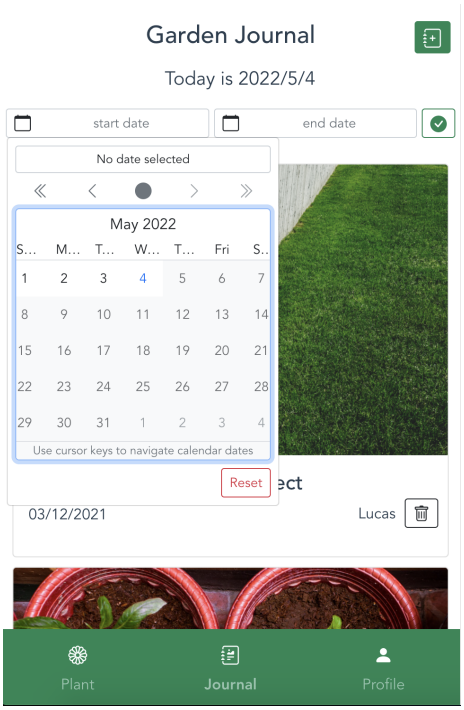


Desktop Frame: Plant Detail Page

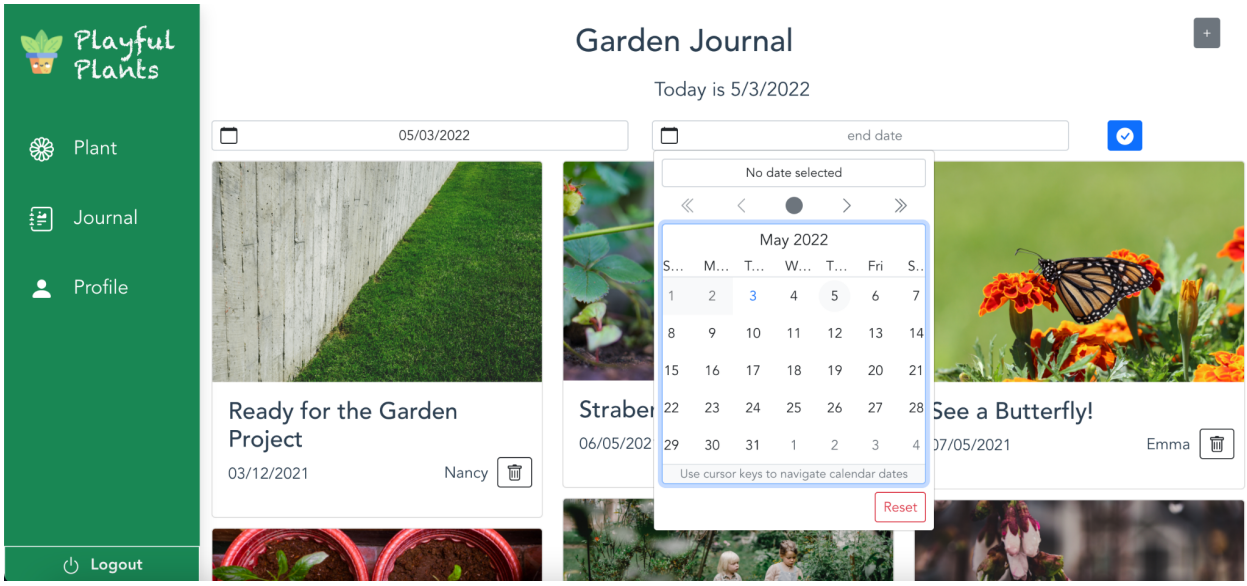
Journal

Main *Journal* Page

For the Journal part of this App, there are three main pages. The first one is the **main Journal page (JournalView.vue)**, where Emma and all her family members can browse through the past journals uploaded by everyone. Emma can also select a specific date or date range to view the journals that fall on those dates. The journal cards use the component of **JournalCard.vue**. We also included a delete button for every journal entry for error recovery. If Lucas accidentally creates a mistaken journal, this function will allow him to delete it. It is helpful for Lucas, who is relatively young, so he does not post a mistaken journal entry accidentally.



Mobile Frame: Journal Datepicker

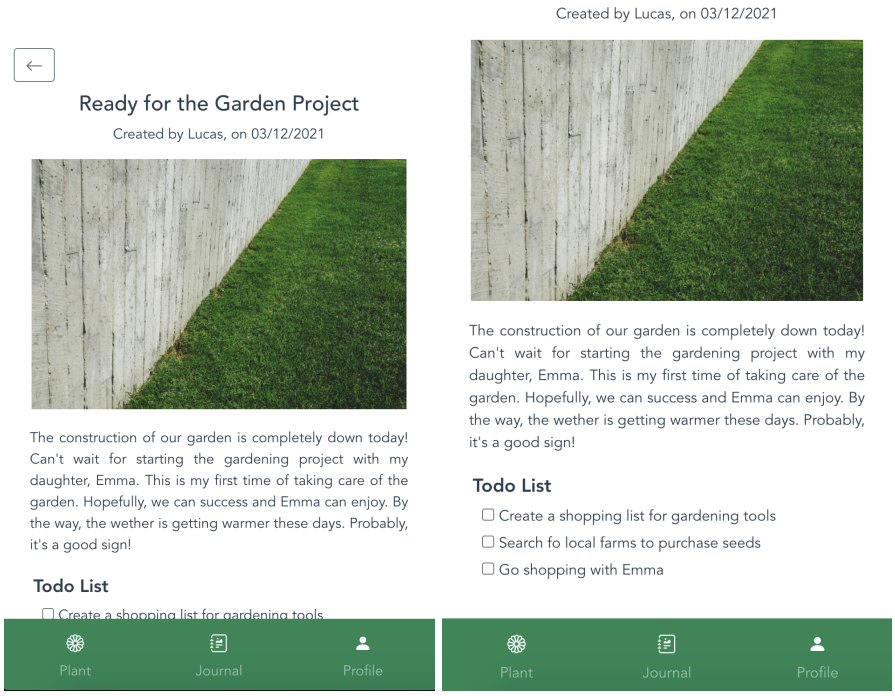


Desktop Frame: Journal Datepicker

***Journal Detail* Page**

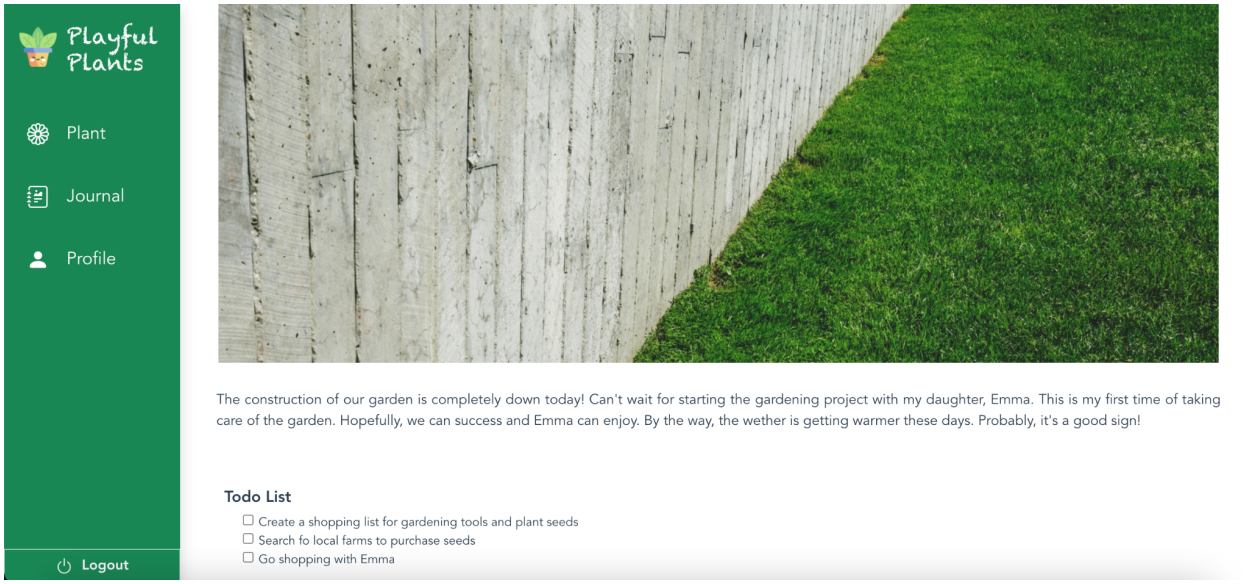
The second page is the **Journal Detail page (JournalDetail.vue)**, where Emma and Lucas can click on past journal entries and review their gardening experiences. This page includes the journal title, creator, date, picture of that particular gardening experience, description, and a to-do list. For instance, after Nancy creates this journal entry, Lucas can view it and see the to-do list

waiting to be completed. After Lucas finishes the tasks, he can click and check on them, and the progress will be recorded. The to-do list is reusing the component of **TodoList.vue**.



Mobile Frame: Journal Details + to-do list

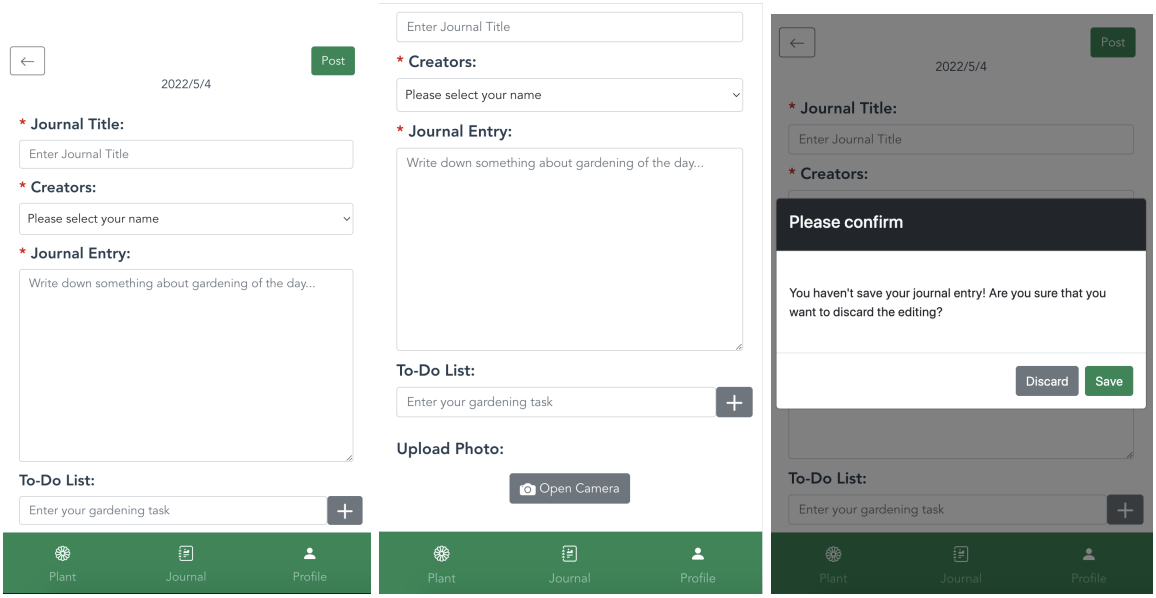
Desktop Frame: Journal Details



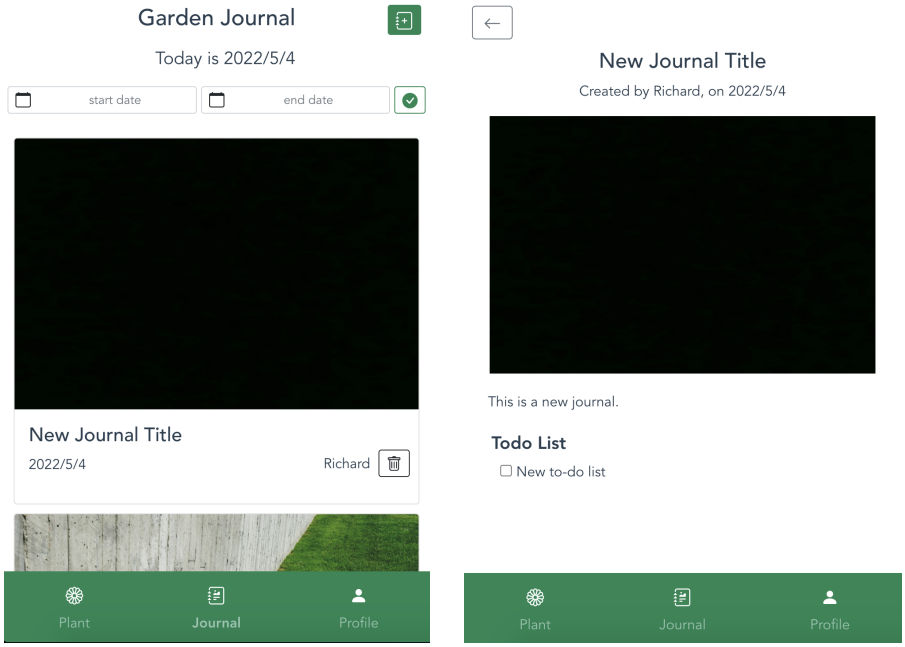
Desktop Frame: Journal Details + to-do list

New Journal Page

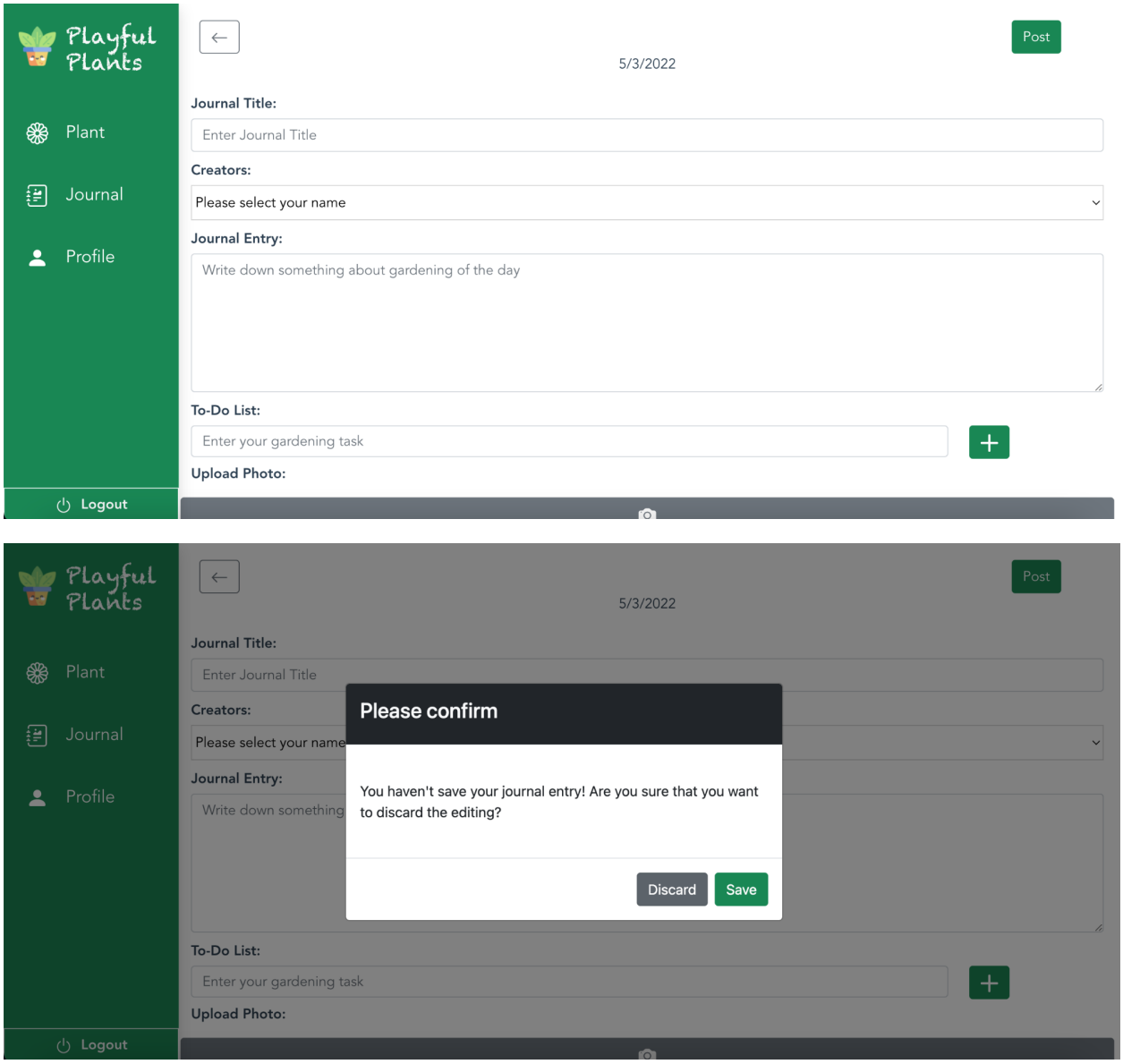
The third page is the page for adding a **new journal entry (AddNewJournal.vue)**. It allows all family members to create a journal entry, either documenting their gardening progress or their gardening experiences. For instance, in Scenario 2, where Lucas writes a journal tracking plant growth, he can use this page to write down what happened that day and take a picture of the plant. In Scenario 3, Emma can add a new journal entry documenting Luca's gardening progress with photos. It serves as an interactive family learning platform to promote parent-child relationships and regular documentation of gardening experiences. We also included a pop-up modal for error recovery. If Lucas accidentally leaves the page by clicking on the return icon, this modal will show up asking him whether he wants to discard the editing. It is helpful for Lucas, who is relatively young, so he does not delete an entry accidentally.



Mobile Frame: Add new journal + Discard Confirmation



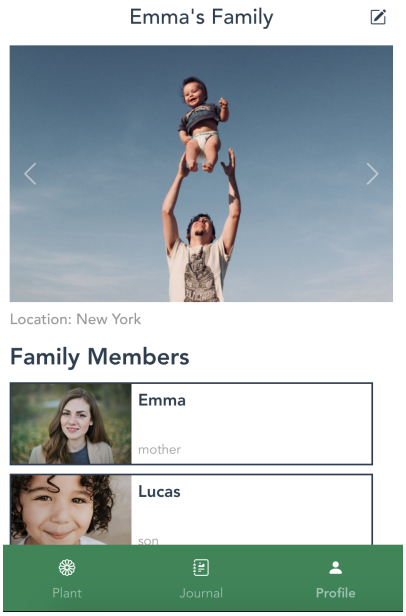
Mobile Frame: New Journal in list + New Journal details



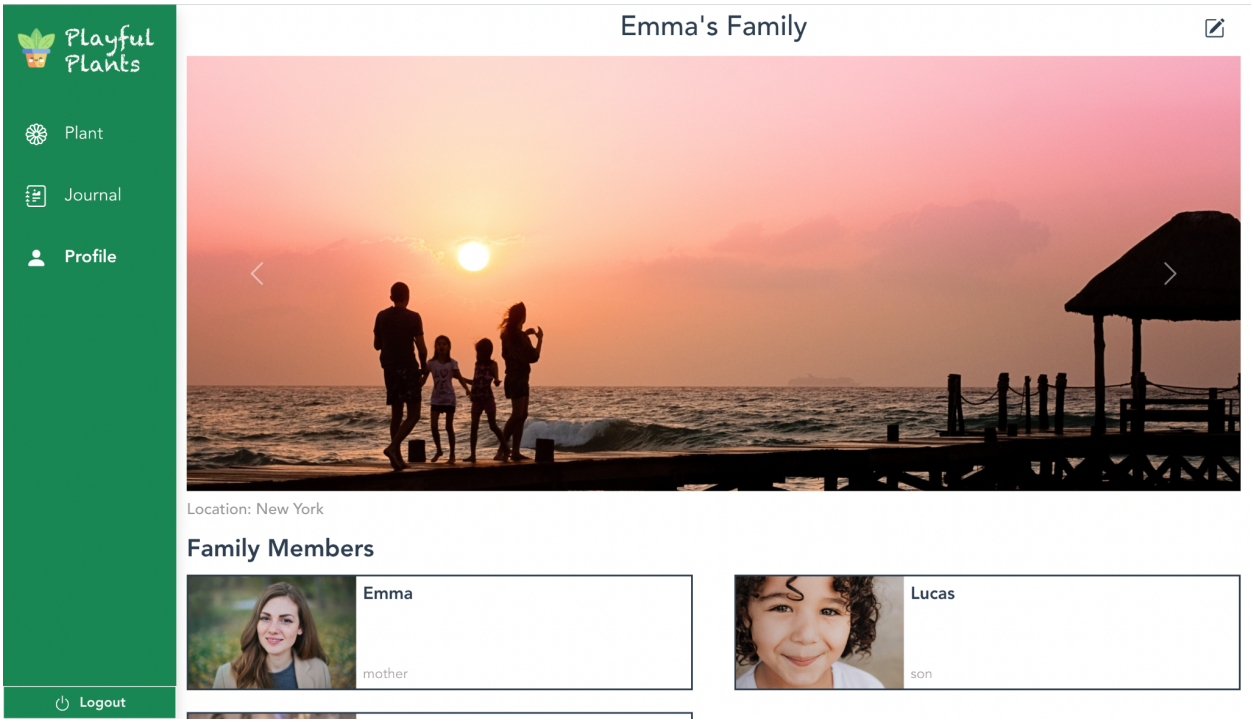
Desktop Frame: Add new journal + Discard Confirmation

Profile

The ***Profile*** **Page (ProfileView.vue)** shows the family information, including family photos, family members who are using this App, and their location. Because editing the family profile and logging out of the account are not the key features in the user scenarios and for testing the product goals, the Profile page is only viewable with no actual functionalities.



Mobile Frame: Profile Page

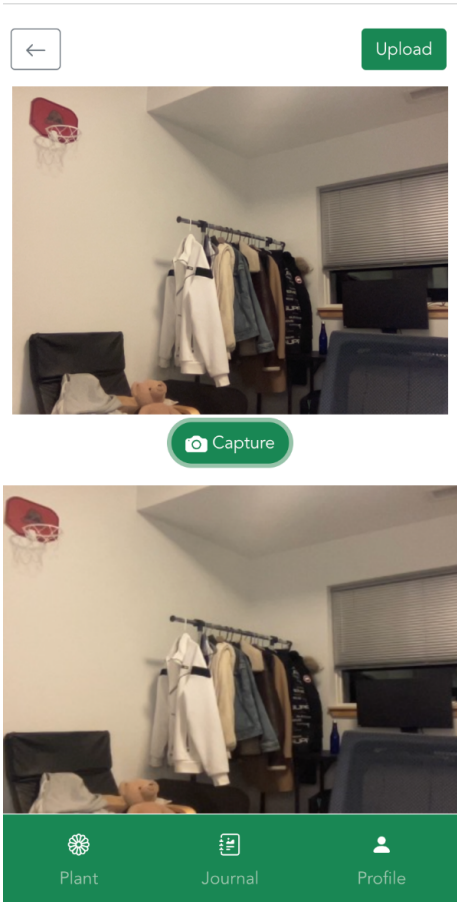


Desktop Frame: Profile Page

Advanced Feature

We implement the web camera in our project. Our implementation steps are as follows.

1. Create a video stream which can show the real time camera view.
2. Click the "*Capture*" button to capture one frame of the streaming video
3. Get the image URL from the video canvas, and the captured photo would display under the video screen.
4. Click the "*Upload*" button to return back to the *AddNewJournal* page, and save the image source to local storage.



Team Member Contribution

Han Gao

- Revised the design sketches based on feedback from the Design Milestone
- Coded the Profile View, Navigation components, and Datepicker filter
- Helped responsive styling and app layout
- Wrote the Final Design Sketch, Revision, and Data parts of the report

Kehui Guo

- Coded the layout of the app, made the app responsive, and supported app styling
- Wrote the Features part of Milestone 4 report

Hang Jiang

- Coded the Plant Card and Journal Card components, and supported app styling and layout
- Supplemented the Features part of Milestone 4 report

Hongxi Jin

- Coded the router and realized the functionalities of the app.
- Implemented the camera api for the app.
- Documented the Router part and the Advanced Feature part of the Milestone 4 report.

Daisy Liu

- Coded to-do list and filter components and functionality, made the app responsive, supported app styling.
- Documented the implementation of using bootstrap vue library for Milestone 4 report.