AtBay Pest Control

Anuja Mehta, Nolan Raghu, Kobin Kempe, Kahero Harriott

Project Description

Our project this semester is an app that we've designed for a subscription based pesticide service. This app will provide an interface for any pest control company to consolidate all their different prevention and infestation products and coordinate every users' treatment plans. At any time, Users can update their current plan, find instructional videos for applying products, and change their personal info.

Subsystem: UI and Navigation

The subsystem we've selected to share for this code review is our UI and Navigation components of our overall code. This subsystem includes the three main components of our app: product selection, current plan description, and personal profile information. To access more information about specific plans, a user can tap on a specific plan and navigate to a new screen. From here, a user can add the product to their plan and update their plan, confirm payment, and order the new products. The current plan description details all the products a user currently has, and they can tap on specific ones to see more information about it. The last screen has personal information about the user including address and payment details.

Source Code Files to Test

These are the files you will be testing, with their line counts. They can all be found in the "navigation" and "screen" folders. They are organized the way they logically fit together.

- **navigation/BottomTabNavigator.tsx** - *56 loc* - This is the navigation wrapper, which organizes the screens into tabs at the bottom of the screen, and puts them in a stack so the 'back' button goes to your previous screen.
  - **navigation/BugsTabNavigator.tsx** - *29 loc* - This implements the navigation within the bugs tab, keeping it logically together so you can transition within the tab
    - **screens/BugsTabScreen.tsx** - *285 loc* - This screen shows the prevention plan and possible infestation plans you can add
    - **screens/BugInfoPopup.tsx** - *240 loc* - This screen currently shows an example screen of information about a bug infestation (before you add it to your plan)
    - **screens/PlanUpdatePopup.tsx** - *83 loc* - This screen will eventually be a confirmation screen for you to update your plan

navigation/PlanTabNavigator.tsx - *17 loc* - This implements navigation within the plan tab.

screens/PlanTabScreen.tsx - *155 loc* - This is the main screen where you can view the products that are part of the plan that you have signed up for

screens/PlanProductPopup.tsx - *0 loc* - This screen is not yet implemented

navigation/ProfileTabNavigator.tsx - *16 loc* - This screen controls navigation within the profile tab.

screens/ProfileTabScreen.tsx - *138 loc* - This screen currently shows example profile information

Total loc: 1019

*Note: some of our style sheets are duplicated to allow for both light and dark mode, so you will only have to test one of these, which will significantly cut down on code to test.

Upcoming Changes

- Pull product info and client info from an external source (database or .json file) and have that displayed for the different pages and popups
- Ability to add plans and products to your personal plan
- Include instructional videos on the Plan Descriptions tab for applying different products
- Make the design for each page more uniform throughout each screen
- A sign in page where users can create an account
- Ability for users to update their profile information

Testing the Code

For a thorough test complete all testing steps that follow.

1. **Loading screen and app compilation**
    a. In the command line, run 'expo start' and scan the QR code that shows up using the Expo app (or attempt to run the app through an emulator). This should run the current version of the app on your phone.
    b. Ensure that the loading screen with the logo of the company shows up properly on the app's initial start up screen.
    c. If all tests have passed move on to step 2.
2. **Bottom Tab Navigator**
    a. Ensure that the bottom tab navigation buttons function properly.

b. The three buttons take you to different screens of the app. In order from left to right, ensure that the buttons bring you to the Bugs Tab Screen ("Packages"), the Plan Tab Screen ("Your Plan"), and the Profile Tab Screen ("Profile").
c. Return to the Bugs Tab Screen using the bottom tab navigator and test the functionality of swiping to move from screen to screen. use a swiping motion to the left to move to the Plan Tab Screen. Do this once more to move to the Profile Tab Screen. Make a swiping motion to the right to move back to the Plan Tab Screen and again to move back to the Bugs Tab Screen.
d. From the Bugs Tab Screen, select the third tab to move to the Profile Tab Screen. From here, test the functionality of using the back button (android UI or app UI for iphone) to return to the previous screen. In this case, the back button should return the user to the Bugs Tab Screen.
e. If all tests have passed move on to step 3.

3. Bugs Tab Screen

a. Move to the Bugs Tab screen using the bottom tab navigator or swiping.
b. Ensure that the screen looks as it should, with the price of the current plan and a button to add bugs to the plan in the top, and a scrollable view of the prevention plan and different bugs taking up the rest.
c. Clicking on either the prevention plan or the bugs buttons should bring up a detailed description of what products the plan or specific insect cover. Ensure that the popup screen comes up when each button is clicked and that the transition between popup and Bugs Tab Screen functions well (without any jittering or lag)
d. Inside the popup for the prevention plan and individual bugs check that the 'Add To Plan' button works to bring the user back to the Bugs Tab Screen after it is pressed.
e. Press the back button in the top left corner (or the Android UI back) to test the buttons ability to bring the user back to the previous screen.
f. If all tests have passed move on to step 4.

4. Plan Tab Screen

a. Move to the Plan Tab Screen in the same way as before.
b. Ensure that the screen looks as it should, with Your Plan as the top text and a description of what is on the screen underneath, as well as a scrolling view of the different products in the plan taking up the rest of the screen.
c. Click on each individual Product. This should reveal a description of the product. Make sure this is the case.
d. If all tests pass move on to step 5.

5. Profile Tab Screen

a. Move to the Profile Tab Screen.

b. Ensure that the screen looks as it should. There should be a title that reads 'Profile' along with a list of user details in a scrollable view.
c. Make sure that all user information is displayed in the view. This includes username, address, current plan, and a list of payment options.
d. If all tests pass, testing is complete and the app is functioning properly.

Installation:

- (Optional) Download WebStorm (the JetBrains IDE for JavaScript) by logging into your JetBrains Vanderbilt account and activating the license.
- Download the project and extract it. The root directory is 'AtBayPestControl'.
- Download Yarn at least in the root directory. (https://classic.yarnpkg.com/en/docs/install). It's a very convenient package manager that we recommend you install globally.
- Open the root with WebStorm as an existing React Native project. A little popup in the bottom right corner should ask you to download some dependencies. Do this.
- If you do not have WebStorm or were not quick enough to catch the popup, run `yarn install`.
- The above should take a while, so in the meantime download the Expo Client app from the Google Play/AppStore onto your device.
- Run `npm install -g expo-cli` .
- To run the app, run `yarn start` in the root folder.
- A QR code should now display both in your web browser and terminal. Scan either to run the app on your personal smartphone.
    - If you have Android/IOS simulators, you can use them instead by running `yarn android` or `yarn ios` respectively

FAQ for common bugs:

- *Do NOT run any commands with the word 'init' in it. These commands create a new project, which is not necessary and will only bloat the code.*
- Issues with 'package.json' or 'package-lock.json'
    - Try running the following in the root directory
        - yarn upgrade; npm-update; npm audit fix
- Issues with Expo (framework)
    - Run the following in root:
        - `npm install -g expo-cli`
        - *npm install –g react-native-cli*
- Issues with Expo (mobile app)
    - Typing 'r' in the terminal while expo is running will restart the project, and after force-quitting + starting Expo again, all changes should be registered.

If you run into any problems, please feel free to email us below:

Nolan Raghu	*nolan.raghu@vanderbilt.edu*
Kobin Kempe	*kobin.g.kempe@vanderbilt.edu*
Kahero Harriott	*kahero.t.harriott@vanderbilt.edu*
Anuja Mehta	*anuja.mehta@vanderbilt.edu*