FindMyTradie

Title	Testing Specification / Documentation
Project Name	FindMyTradie
Names & Student ID	Christian McAuley - 19353261 Bryan Rohan - 19432556
Programme	Computer Applications (Sft Eng.)
School	School of Computing
Year	4th
Emails	christian.mcauley6@mail.dcu.ie bryan.rohan3@mail.dcu.ie
Module Code	CA400
Supervisor	Hossein Javidnia

Table of Contents

1 Type of Testing Covered

In our project, we implemented various testing strategies to guarantee the dependability and quality of our React Native app. These methodologies, including Unit Testing, System Testing, End User Testing, and Manual Testing, each played a vital role in the development process.

Unit Testing examined individual components or functions of the app separately, confirming that each code segment performed as anticipated. This testing is critical for a React Native app. It helps detect issues early in development and ensures all components function correctly before integrating into the larger system. We used Jest to cover the unit tests on nine application screens and ran 26 successful tests that passed.

System Testing assessed the entire application as a unified entity, ensuring all features and components interacted as expected. This testing level is essential for a React Native app, as it verifies its performance, usability, and general quality, meeting functional and non-functional requirements. To cover the system testing, we used Google Spreadsheets to keep track of scenarios but keeping track of input, expected out and ensuring the output is correct.

End User Testing gathered feedback from real users testing the app in realistic scenarios. This testing is crucial for a React Native app, offering valuable insights into the user experience and helping to identify any usability concerns, design defects, or other issues that could adversely affect the end user. We did this using Google Forms and got a sample of ten people, five tradesmen and five non-tradesmen from an 18+ age group.

Manual Testing enabled us to inspect the app from a human viewpoint, manually executing test cases to discover defects or problems that automated tests may not identify. Manual Testing is especially important for a React Native app, as it allows us to evaluate the application from the end user's perspective, ensuring it is both functionally robust and user-friendly.

By utilising these varied testing approaches throughout the development process, we effectively evaluated our React Native app, guaranteeing its quality, reliability, and user satisfaction, ultimately resulting in successful project completion.

2 End User Testing

End User Testing aimed to ensure that our application met the needs and expectations of the target users. It validated that the application is user-friendly, intuitive, and performed well in real-world scenarios. By involving end users in our testing process, we identified and addressed any issues or shortcomings in our application. This led to a better user experience and increased user satisfaction with our application.

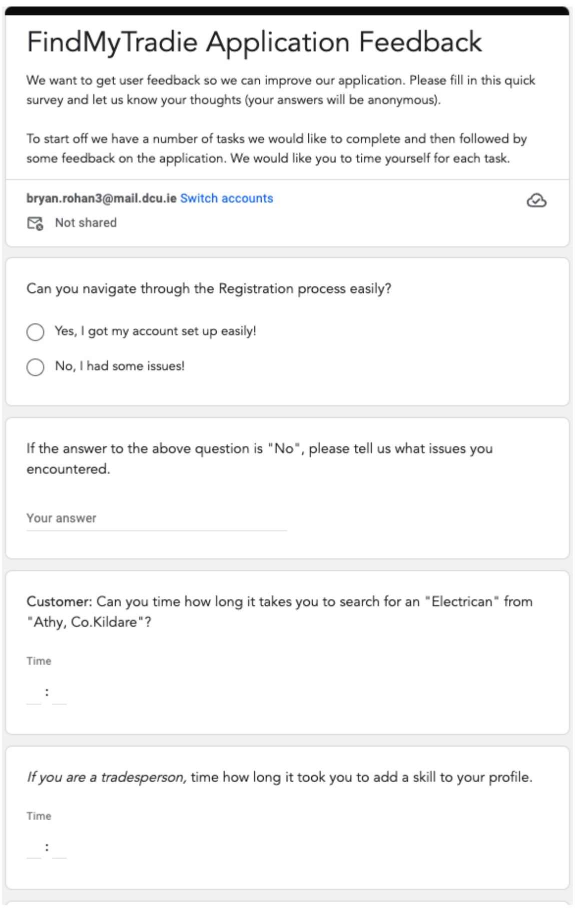
We got diverse participants for our End User Testing, including five tradesmen and five everyday people aged 18 to 70. We got a wide range of participants to ensure that our application was tested across different user groups with varying experience levels and familiarity with smartphones.

We made a Google Form with some tasks for the users, and then we also asked questions on the application's design to get feedback on what the target audience thought of the application.

Linked Below are the Google Forms we asked users to fill out so that we could get feedback on the application:

https://docs.google.com/forms/d/e/1FAIpQLSc6WwS-3wPzxKTL4CVVnm8DYcuEwbdPI5jaG-cwDEYN_hdrQQ/viewform?usp=sf_link

As stated above, the target audience for our app was tradespeople of all ages and people interested in finding tradespeople to work for them.



FindMyTradie Application Feedback

We want to get user feedback so we can improve our application. Please fill in this quick survey and let us know your thoughts (your answers will be anonymous).

To start off we have a number of tasks we would like to complete and then followed by some feedback on the application. We would like you to time yourself for each task.

bryan.rohan3@mail.dcu.ie Switch accounts

Not shared

Can you navigate through the Registration process easily?

- Yes, I got my account set up easily!
- No, I had some issues!

If the answer to the above question is "No", please tell us what issues you encountered.

Your answer

Customer: Can you time how long it takes you to search for an "Electrican" from "Athy, Co.Kildare"?

Time

__ : __

If you are a tradesperson, time how long it took you to add a skill to your profile.

Time

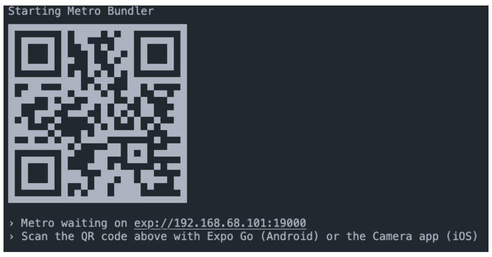
__ : __

Here is a snippet of the tasks and questions asked.

Below is the link to the results of the feedback we received from the participants:

https://docs.google.com/spreadsheets/d/1VEzXpsfQKNz0YUe1aQXXgWNy2oLFoARuUKLDVE-nA8U/edit?usp=sharing

The participants were mostly friends and family members because they needed to be on the same WiFi connection and get the QR code off us to use the application as we didnt deploy it on the Apple Developer Program. Christians father owns a building company so he went to work with him one day and got his fathers employees to use the application and other workers on site.



```
Starting Metro Bundler

› Metro waiting on exp://192.168.68.101:19000
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)
```

As shown above, we asked users to perform various tasks to evaluate various aspects of the application:

1) Customer Sign-Up Process: Assess the ease and efficiency of the registration process for customers.
2) Tradesman Sign-Up Process: Evaluate the registration process for tradesmen, ensuring it is straightforward and user-friendly.
3) Time to complete a search for an “Electrician in Athy”: This measured the efficiency of the search function, as well as the overall usability of the app for finding service providers.
4) Time for tradesmen to add skills to their profile: This assessed the ease and efficiency of updating tradesmen's profiles with their skills. It could tell us whether the Edit Profile feature on the application was straightforward to use.
5) App satisfaction rating (1 to 5): Gather a quantitative measure of users' overall satisfaction with the application.
6) Ease of use: Determine whether users find our app easy to navigate and interact with.
7) Feedback on difficulties encountered: This collected valuable insights into any issues or obstacles users face while using our app, which we used to make improvements.
8) Opinion on User Interface: We got user feedback on the app's visual design and layout.
9) Suggestions for additional features: We encouraged the users to share ideas for new features or improvements that could enhance the app's functionality and user experience.

Collecting this feedback and analysing the results helped us identify areas of improvement, and we made necessary changes to the application.

3 System Testing

The importance of System Testing in the software development process for our application can be insignificant. System Testing is crucial to ensure that the entire application functions seamlessly as a cohesive unit, validating that all features work together as intended and that the app meets its functional and non-functional requirements. By conducting thorough System Testing, we identified and rectified any defects or issues the application had. We were then able to fix these issues, ensuring a positive user experience and maintaining the app's overall reliability.

Throughout the development process, we executed over 120 test scenarios to evaluate various aspects of the app, such as checking whether features work correctly (e.g., login page, search function, database updates) and verifying that buttons were directed to the appropriate pages. This testing approach helped us ensure that all app components functioned as expected and that any issues were addressed correctly.

The tests we conducted included usability tests to ensure that all buttons and functions on the app worked smoothly and security tests to verify that sensitive user data, such as email addresses and passwords, were properly handled. We also implemented error handling mechanisms to provide users with informative error messages if, for example, an incorrect password was entered.

To derive the expected output for each test, we used the following structure: scenario, input, expected output, output, tester, and date. For instance:

Example Scenario:

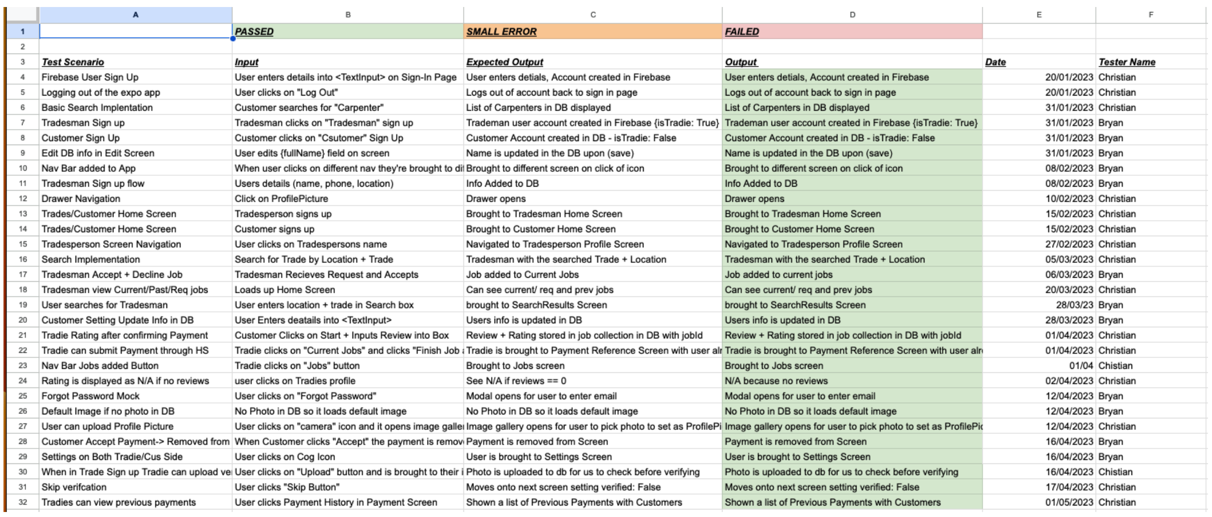
Scenario	Login Screen
Input	The user inputs email & password into the text field and clicks "Login"
Expected Output	1) The user is logged into their account. 2) The user is alerted if the password or email is wrong.
Output	1) User is logged in 2) The user receives a login alert indicating either their password or email is incorrect
Date	01/05/2023
Tester	Bryan

During System Testing, we identified several issues, including error handling problems with buttons on the sign-up screen, which allowed users to continue without entering data. We also encountered issues with the login feature, where users would receive random alerts. We promptly addressed these issues, ensuring a smoother user experience and improving the overall quality of the application.

Below is the link to the spreadsheet we covered the testing in:

https://docs.google.com/spreadsheets/d/1JDpcvcAL_yK-ttb3W6FUMMxXw6RCskc5NtyylK2Iy58/edit#gid=0

Here is a snippet of the spreadsheet:

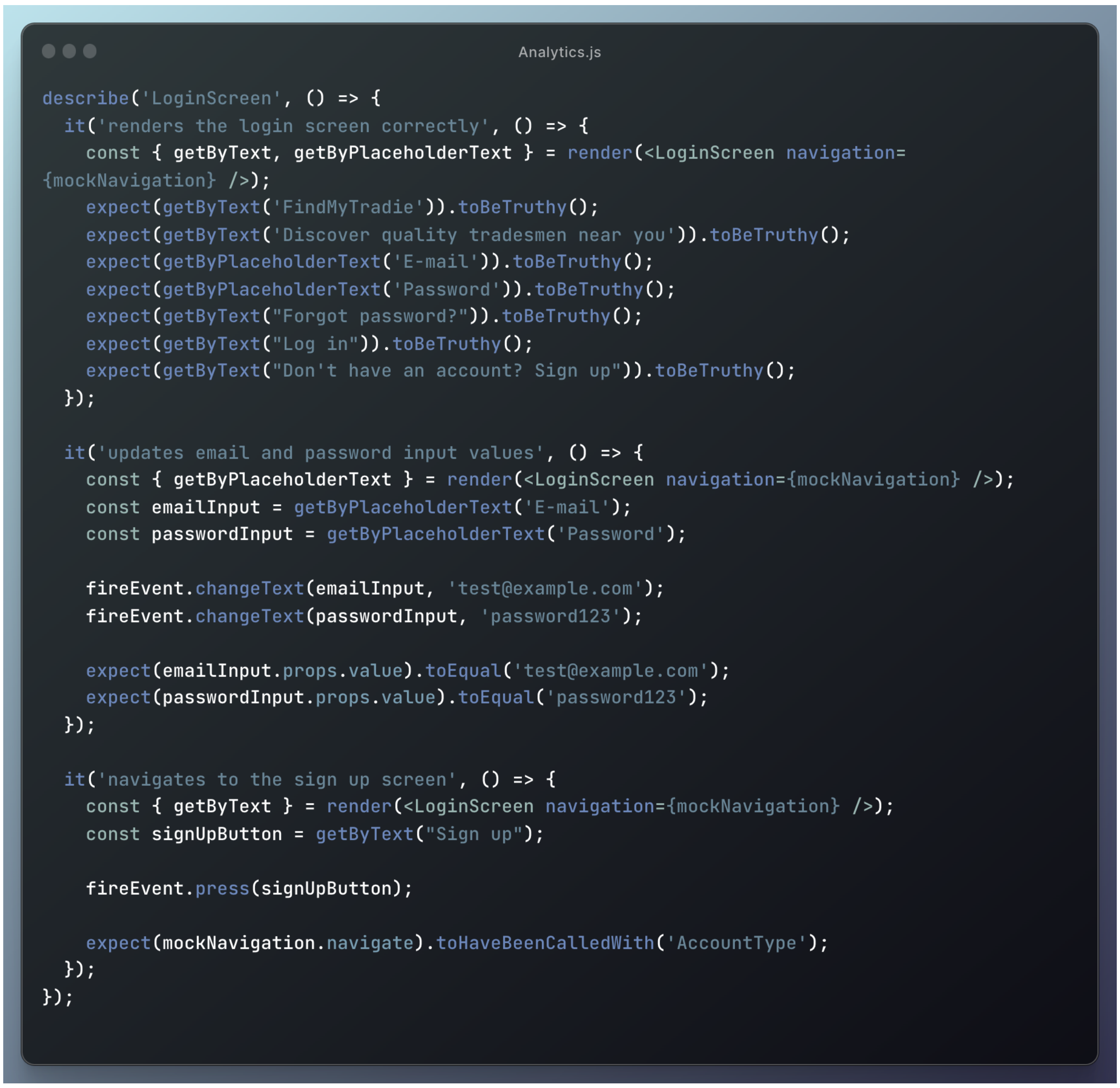


	A	B	C	D	E	F
1		***PASSED***	***SMALL ERROR***	***FAILED***		
2						
3	***Test Scenario***	***Input***	***Expected Output***	***Output***	***Date***	***Tester Name***
4	Firebase User Sign Up	User enters details into <TextInput> on Sign-In Page	User enters detials, Account created in Firebase	User enters detials, Account created in Firebase	20/01/2023	Christian
5	Logging out of the expo app	User clicks on "Log Out"	Logs out of account back to sign in page	Logs out of account back to sign in page	20/01/2023	Christian
6	Basic Search Implentation	Customer searches for "Carpenter"	List of Carpenters in DB displayed	List of Carpenters in DB displayed	31/01/2023	Christian
7	Tradesman Sign up	Tradesman clicks on "Tradesman" sign up	Trademan user account created in Firebase {isTradie: True}	Trademan user account created in Firebase {isTradie: True}	31/01/2023	Bryan
8	Customer Sign Up	Customer clicks on "Csutomer" Sign Up	Customer Account created in DB - isTradie: False	Customer Account created in DB - isTradie: False	31/01/2023	Bryan
9	Edit DB info in Edit Screen	User edits {fullName} field on screen	Name is updated in the DB upon (save)	Name is updated in the DB upon (save)	31/01/2023	Bryan
10	Nav Bar added to App	When user clicks on different nav they're brought to di	Brought to different screen on click of icon	Brought to different screen on click of icon	08/02/2023	Bryan
11	Tradesman Sign up flow	Users details (name, phone, location)	Info Added to DB	Info Added to DB	08/02/2023	Bryan
12	Drawer Navigation	Click on ProfilePicture	Drawer opens	Drawer opens	10/02/2023	Christian
13	Trades/Customer Home Screen	Tradesperson signs up	Brought to Tradesman Home Screen	Brought to Tradesman Home Screen	15/02/2023	Christian
14	Trades/Customer Home Screen	Customer signs up	Brought to Customer Home Screen	Brought to Customer Home Screen	15/02/2023	Christian
15	Tradesperson Screen Navigation	User clicks on Tradespersons name	Navigated to Tradesperson Profile Screen	Navigated to Tradesperson Profile Screen	27/02/2023	Christian
16	Search Implementation	Search for Trade by Location + Trade	Tradesman with the searched Trade + Location	Tradesman with the searched Trade + Location	05/03/2023	Christian
17	Tradesman Accept + Decline Job	Tradesman Recieves Request and Accepts	Job added to Current Jobs	Job added to current jobs	06/03/2023	Bryan
18	Tradesman view Current/Past/Req jobs	Loads up Home Screen	Can see current/ req and prev jobs	Can see current/ req and prev jobs	20/03/2023	Christian
19	User searches for Tradesman	User enters location + trade in Search box	brought to SearchResults Screen	brought to SearchResults Screen	28/03/23	Bryan
20	Customer Setting Update Info in DB	User Enters deatails into <TextInput>	Users info is updated in DB	Users info is updated in DB	28/03/2023	Bryan
21	Tradie Rating after confirming Payment	Customer Clicks on Start + Inputs Review into Box	Review + Rating stored in job collection in DB with jobId	Review + Rating stored in job collection in DB with jobId	01/04/2023	Christian
22	Tradie can submit Payment through HS	Tradie clicks on "Current Jobs" and clicks "Finish Job	Tradie is brought to Payment Reference Screen with user alr	Tradie is brought to Payment Reference Screen with user alr	01/04/2023	Christian
23	Nav Bar Jobs added Button	Tradie clicks on "Jobs" button	Brought to Jobs screen	Brought to Jobs screen	01/04	Chistian
24	Rating is displayed as N/A if no reviews	user clicks on Tradies profile	See N/A if reviews == 0	N/A because no reviews	02/04/2023	Christian
25	Forgot Password Mock	User clicks on "Forgot Password"	Modal opens for user to enter email	Modal opens for user to enter email	12/04/2023	Bryan
26	Default Image if no photo in DB	No Photo in DB so it loads default image	No Photo in DB so it loads default image	No Photo in DB so it loads default image	12/04/2023	Bryan
27	User can upload Profile Picture	User clicks on "camera" icon and it opens image galle	Image gallery opens for user to pick photo to set as ProfilePi	Image gallery opens for user to pick photo to set as ProfilePic	12/04/2023	Christian
28	Customer Accept Payment-> Removed from	When Customer clicks "Accept" the payment is remov	Payment is removed from Screen	Payment is removed from Screen	16/04/2023	Bryan
29	Settings on Both Tradie/Cus Side	User clicks on Cog Icon	User is brought to Settings Screen	User is brought to Settings Screen	16/04/2023	Bryan
30	When in Trade Sign up Tradie can upload ve	User clicks on "Upload" button and is brought to their i	Photo is uploaded to db for us to check before verifying	Photo is uploaded to db for us to check before verifying	16/04/2023	Chistian
31	Skip verifcation	User clicks "Skip Button"	Moves onto next screen setting verified: False	Moves onto next screen setting verified: False	17/04/2023	Christian
32	Tradies can view previous payments	User clicks Payment History in Payment Screen	Shown a list of Previous Payments with Customers	Shown a list of Previous Payments with Customers	01/05/2023	Christian

4 Unit Testing

We decided to complete our unit testing using Jest because React Native and Expo officially recommend Jest for unit testing. Unit testing is an important part of application development. It ensures that we write modular and well-structured code, which helps us to maintain and update our code much faster. Due to time constraints and the amount of time spent using other testing methods, we only managed to set up basic unit tests for nine screens, and it totalled to 26 small tests. These tests mainly checked whether or not the screens rendered correctly and whether certain buttons function correctly.

We managed to pass all the tests that were written. Here is a snippet of the tests for the Login Screen:



```
describe('LoginScreen', () => {
  it('renders the login screen correctly', () => {
    const { getByText, getByPlaceholderText } = render(<LoginScreen navigation=
{mockNavigation} />);
    expect(getByText('FindMyTradie')).toBeTruthy();
    expect(getByText('Discover quality tradesmen near you')).toBeTruthy();
    expect(getByPlaceholderText('E-mail')).toBeTruthy();
    expect(getByPlaceholderText('Password')).toBeTruthy();
    expect(getByText("Forgot password?")).toBeTruthy();
    expect(getByText("Log in")).toBeTruthy();
    expect(getByText("Don't have an account? Sign up")).toBeTruthy();
  });

  it('updates email and password input values', () => {
    const { getByPlaceholderText } = render(<LoginScreen navigation={mockNavigation} />);
    const emailInput = getByPlaceholderText('E-mail');
    const passwordInput = getByPlaceholderText('Password');

    fireEvent.changeText(emailInput, 'test@example.com');
    fireEvent.changeText(passwordInput, 'password123');

    expect(emailInput.props.value).toEqual('test@example.com');
    expect(passwordInput.props.value).toEqual('password123');
  });

  it('navigates to the sign up screen', () => {
    const { getByText } = render(<LoginScreen navigation={mockNavigation} />);
    const signUpButton = getByText("Sign up");

    fireEvent.press(signUpButton);

    expect(mockNavigation.navigate).toHaveBeenCalledWith('AccountType');
  });
});
```

We wanted to automate the basic tests, like making sure users can enter their details correctly and buttons work as expected.

We Had 9 test suites, and all 26 tests passed.

Throughout the unit testing process, we found ourselves refactoring our code slightly based on the results of the tests; it made it easier for us to see where we could improve our code.

5 Manual Testing

Manual Testing's significance in our application's software development process is primarily due to its capacity to reveal defects and usability issues that automated tests may overlook. Manual Testing offers a human perspective, enabling us to evaluate the application from the end user's standpoint and pinpoint any issues that could adversely impact the user experience. Integrating Manual Testing throughout development ensured our application was functionally robust and user-friendly.

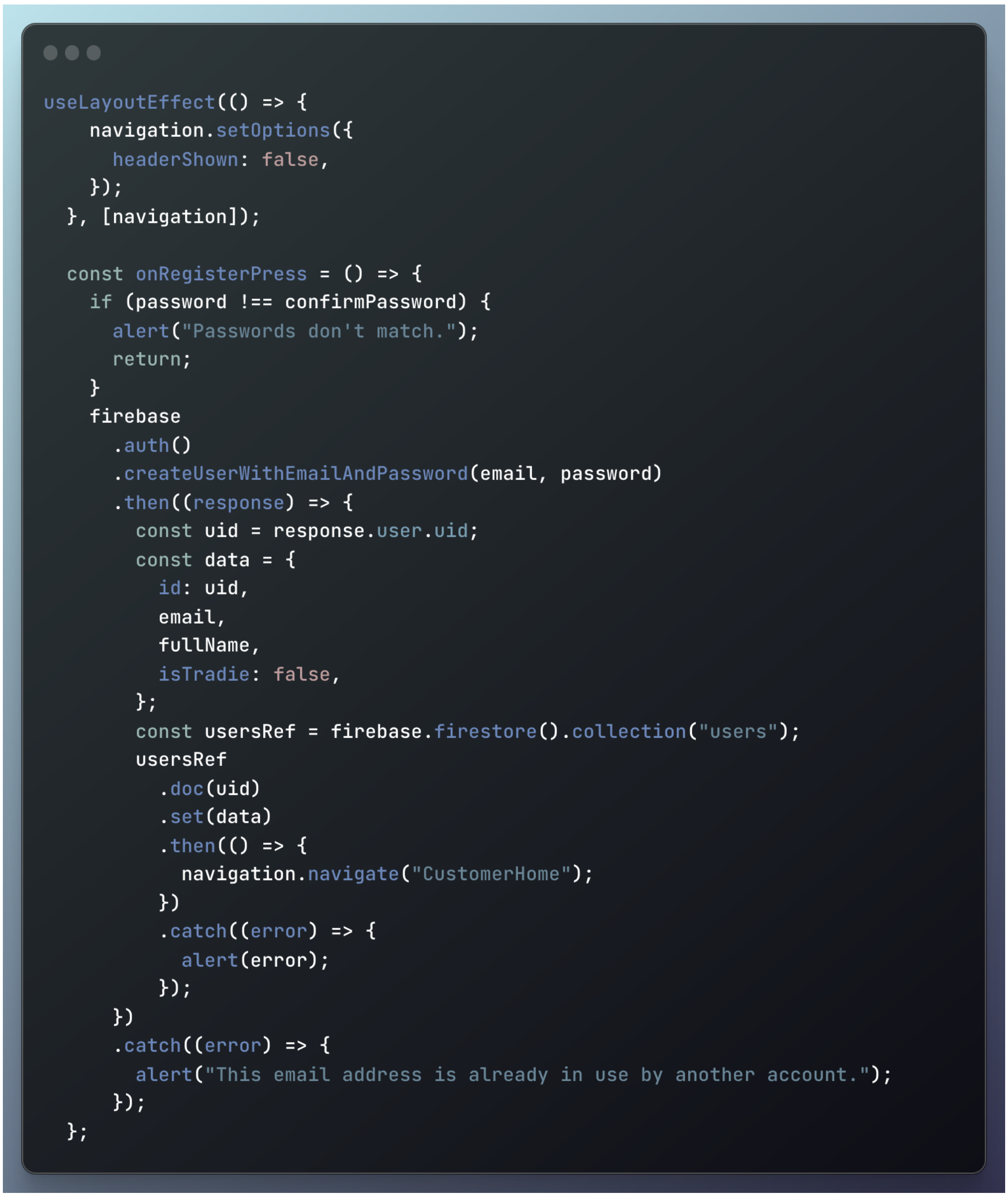
We performed Manual Testing at different stages of the application's development, such as when adding new features (e.g., Search Feature, Around You Feature) or fixing error handling bugs (e.g., incorrect email or password handling). We also conducted Manual Testing after making UI changes to verify that the alterations did not affect the app's functionality. This comprehensive and ongoing testing approach allowed us to preserve the app's quality and usability throughout development.

During Manual Testing, we discovered several issues, including error handling problems with buttons on the sign-up screen that let users proceed without entering data and issues with the login feature that caused users to receive random alerts. Addressing these issues enhanced the overall user experience and ensured the app operated seamlessly.

One of the primary lessons we learned from Manual Testing was the necessity for thorough error handling and user input validation. We improved the application by identifying issues in these areas, such as stopping users from continuing with incomplete sign-up data and providing informative error messages when users input incorrect email addresses or passwords. This improved the app's functionality and contributed to a more intuitive and user-friendly experience. By learning from these issues and addressing them, we were able to develop a more refined and dependable application, ultimately leading to increased user satisfaction.

Here is an example of code where we had to add alerts while doing manual testing because there was a bug in the system if the passwords didn't match or the email address was already in the database.

Error Handling for Login Screen:



```
useLayoutEffect(() => {
    navigation.setOptions({
      headerShown: false,
    });
  }, [navigation]);

  const onRegisterPress = () => {
    if (password !== confirmPassword) {
      alert("Passwords don't match.");
      return;
    }
    firebase
      .auth()
      .createUserWithEmailAndPassword(email, password)
      .then((response) => {
        const uid = response.user.uid;
        const data = {
          id: uid,
          email,
          fullName,
          isTradie: false,
        };
        const usersRef = firebase.firestore().collection("users");
        usersRef
          .doc(uid)
          .set(data)
          .then(() => {
            navigation.navigate("CustomerHome");
          })
          .catch((error) => {
            alert(error);
          });
      })
      .catch((error) => {
        alert("This email address is already in use by another account.");
      });
  };
```

We added the alert for Passwords not matching and if another account already uses an Email address.