Bachelor of Science (Hons) in Applied Computing

Dental Benefits Portal Mobile Application

IN COLLABORATION WITH

Michael Gerber (20093265)

24 April 2024

Table of Contents

Plagiarism Declaration

- I declare that all material in this report is entirely my own work except where duly acknowledged.
- I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams, or other material; including software and other electronic media in which intellectual property rights may reside.
- I have provided a complete bibliography of all works and sources used in preparation of this submission.
- I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: Michael Gerber
Student Number: 20093265

Introduction

Background & Motivation

This fourth-year project involves the redevelopment of an existing mobile application in collaboration with Sun Life. SETU reached out to students with an offer from Sun Life to undertake this assignment. This presented an opportunity for me to apply some of my skills in a real-world development scenario, so I applied and got the offer.

The existing app available on both iOS and Android devices, is named the Benefits tool app. The app is used to manage users' dental benefits. It allows users to view their dental coverage, access their digital dental ID cards, find a dentist, and access information about their claims and procedures covered by their policies. This app is showing signs of aging and needs a newer backend.

Sun Life's dental web portal offers the updated backend. The full proposed assignment is a redevelopment of the UI using a framework of my choice, while applying the newer backend from the web portal.

Project Goals

The main goal is to fulfil the feature requirements of this project. These requirements consist of a few tasks. Firstly, critically analysing the frameworks, and selecting a suitable option for this project. Then gathering information about how the two existing systems function namely the mobile benefits app and dental benefits web portal. Lastly, applying this knowledge of the existing systems and building a functional app with the selected framework.

While doing these things another personal goal for myself is expanding my existing skillset. Learning new technologies and learning from existing systems is a great way to improve my own knowledge. Collaborating with the Sun Life team will provide me with valuable experience in working with other people and will help improve my communication skills.

Requirement Analysis

Framework Selection

Starting development, a decision must be made on what cross platform mobile app framework to choose. The 2 options we had decided on was either Flutter or React Native.

Flutter promises great developer documentation and easy to use prebuilt components, while React Native offered flexibility and a huge community that offers thousands of 3rd party libraries.

The plan devised to choosing a suitable framework goes as follows:

1. Set up framework development environment.
2. Create the starter project.
3. Analyse development tools.
4. Add simple functionality:
    a. Create a text field.
    b. Create a button that moves to a new page displaying the text in the text field.

    c. Call a GET request from some API.
    d. Request some permission on the device.
5. Analyse ease of working with this framework and how easy it is to learn.

Functional Requirements

These requirements refer to specific features or functionality that the application requires. That includes the following:

1. Cross Platform
   Using a cross platform mobile development framework should allow for the app to be compiled for both iOS and Android mobile operating systems.

2. User Authentication
   Users require a secure and user-friendly authentication system. This should allow the user to login using a username and password with a simple user experience.

3. Dental Policy
   Viewing the user's dental policy and all the information that comes along with the it. Along with the policy information the following is included with the dental policy.

    a. Dental Claims
       View dental claims history for the logged in user. It should have the ability to filter by the claim's member, and search by claim numbers.

    b. Dental Procedures
       View procedures covered by the policy. Allow for searching procedure codes and names. Have the ability to view specific dental procedure's information.

    c. Dental Members
       View members covered on the policy. Allow the user to display specific member information, and the coverage for said member.

4. Dental ID Card
   The ID card PDF should be downloaded and displayed within the application. The user should be able to add either Apple Pay or Google Wallet depending on the device.

5. Dentist Search
   Sun Life has a service to search for dentists within their network using a specified location and radius. This dentist search functionality should be added into the application utilizing the Google Maps SDK to view the locations of the dentists.

6. View User Profile
   The user should have the ability to view their user profile. And edit basic information.

7. Permission Handling
   With mobile applications permission handling is very important. Location access will be required for this app. It should be requested and handled correctly if not provided.

8. Localisation
   International applications are sometimes required to include multiple language support. This app could theoretically have a US or Canadian audience where the French and Spanish languages are prevalent. That's why it should have a flexible and easy to use localisation framework set up where app labels can be translated.

9. Sun Life Themes
   Sun Life is proud of their brand and to stay true to the Sun Life brand the app requires a Sun Life theme set up to keep labels, colours, fonts, imagery, etc consistent throughout.

Non-functional Requirements

1. Security
   Security must be kept in mind with everything developed. User authentication and API endpoints need to be accessed securely.

2. Maintainability & Upgradability
   Keeping in mind the possibility of future API changes and updates. The API access system needs to be adapted to be upgraded in the future.

3. Performance
   Keeping an app performant and responsive is crucial for a good user experience.

Deciding on the UI framework

Before development started a decision had to be made on the UI framework, either Flutter or React Native. After applying the plan for framework selection. I had developed 2 simple apps utilizing the 2 frameworks discussed. The following are my findings.

Experience with React Native

I had created the starting project for React Native. Setting up the IDE was as easy as installing a Visual Studio Code extension. I am familiar with JavaScript and Typescript so working with the TypeScript language for this framework was not an issue.

React uses an application called Expo. Expo is a set of tools and services that React Native uses to help with development, debugging, and deployment of applications across multiple operating systems (Cloud Devs Inc 2024). This was a useful tool when building the UI and it helped with figuring out spacing and sizes of components.

Handling the API calls in React Native and state management of pages and components are handled the same way it would be in a normal React project using '`useState()`' methods etc. I found it hard and somewhat tedious to add new components and change the UI of the project.

Experience with Flutter

Flutter creates a Material3 design inspired application as it's starting project. The code for the starter app was surprisingly simple. Setting up Android Studio for Flutter was simple, as it only requires a plugin install. Flutter required me to learn the programming language Dart. This was not too big of a problem as it's similar to Kotlin. Dart offers null safety, and it is strongly typed, which I liked.

The development experience was very good. Documentation on components and features in Dart is great. Flutter took a very object-oriented approach to state management, async methods, such as API calls, are handled using something called a 'Future'. These Futures act like JavaScript promises where async results can be handled whenever they are complete, while letting the rest of the application run separately.

Pages and components can be defined as 'Stateless' or 'Stateful', where the one does not have any state and cannot change i.e. constant, and the other can change state. This allows the changing of state to only apply to stateful components. This is a form of optimization provided by Flutter that causes only stateful components to redraw when state changes.

Conclusion

Having done web development in the past for assignments I thought I would choose React Native as the framework for this project, but I found that the developer experience for me was better using Flutter. I found it very intuitive handling state changes and managing components. Documentation for Flutter components are very well done, and I could easily learn anything I needed about the vast number of native components provided. I ultimately decided on using Flutter.

Technologies

Front-End

The application is developed using Flutter, a cross platform framework created by Google. Single codebases can be used to create natively compiled applications for mobile, web, and desktop. The Dart programming language is used to develop apps in Flutter. IDEs that support Flutter include Android Studio and Visual Studio Code.

Google showcases all Google mobile apps being developed using Flutter. Chinese tech giants Tencent and Alibaba utilizes Flutter for their mobile applications. (Google LLC 2024)

Back-End

The back end for the application is provided by Sun Life. Sun Life has an existing back end for their own web portal and Mobile Benefits Tool mobile app. Two APIs have been provided for this project.

Mobile Benefits Tool API
This API was only used to handle user login and authentication.

Web Portal API
This is a newer API used to fetch data for the Dental Benefits Web Portal. Most of the app's functionality is using this API. This API provides endpoints for fetching dental coverage, policies, claims, id cards, etc.

Database

The data is provided by Sun Life APIs and systems. After consideration the app is going to be treated as a web application that will pull the data from the API when needed. Therefor there will be no need for a local database.

Version Control

Git is used as the version control system used for this project.

Project Management

Sun Life uses Jira as their project management software. This project got its own Scrum board that was used to organize requirements and features. A view from the Scrum board can be found in the appendices.

Methodologies

Scrum is a software development methodology that uses small iterations of development to deliver a fully developed software product (Agile Scrum Methodology Explained n. d.). Sun Life uses a form of scrum called SAFe. SAFe is a larger framework of Scrum designed for large corporations to work on large projects in parallel. This project utilized a normal Scrum approach, but I got to learn about being part of a much larger SAFe framework.

Data Transfer Objects (DTO)

Data Models in Dart

Data Transfer Objects refers to the data models that are used to transfer data between services. These are the data models that are being received from the Sun Life API. Data models in Dart is handled as a simple class object. Serializing the responses from the API to a data model using Dart will be covered in a later section called 'JSON Serialization in Flutter'.

User Authentication

The user authentication model holds all the authentication data. When the user logs in an authentication response is received from the API call.

Most of the fields are self-explanatory, but there are some that stand out as more important. That being the "oauthToken", and "oauthRefreshToken". These 2 tokens are used in later requests from the Mobile Benefits API

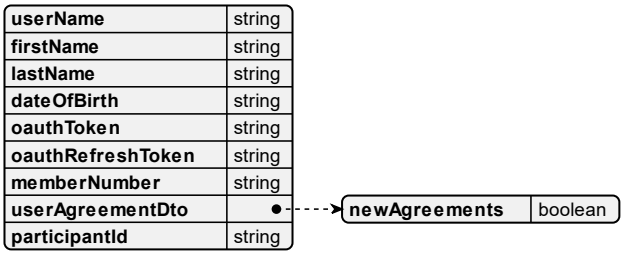

Diagram 1 User Authentication Model

Dental Coverage

The dental coverage model is complex and contains a lot of information. This Dental Coverage object is used for most of the information displayed to the user. It includes policy, covered procedures, benefit, and member/dependent information.

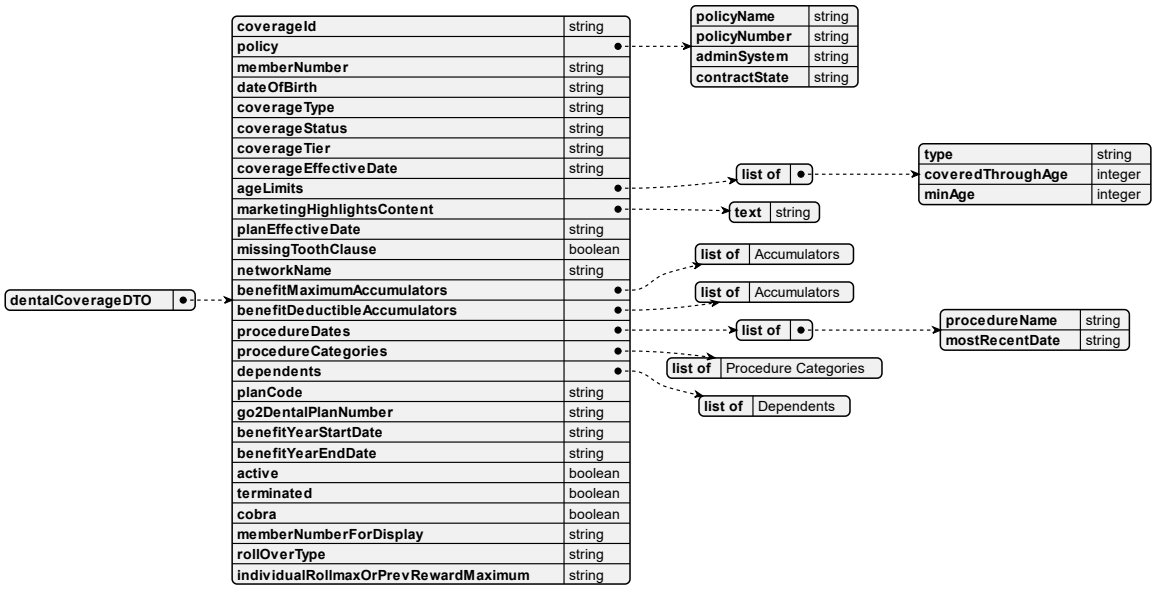

Diagram 2 Dental Coverage Model

Accumulators

Accumulators contain information about how much coverage the user's policy provides. The coverage is different based on the type of coverage, the types of coverage provided are 'In-Network' and 'Out-of-Network'.

Diagram 3 Accumulator Model

Procedures

Procedures come bundled in procedure categories. Each procedure category contains a list of Procedure objects. Procedures are also covered differently based on if it is 'In-Network' and 'Out-of-Network'.

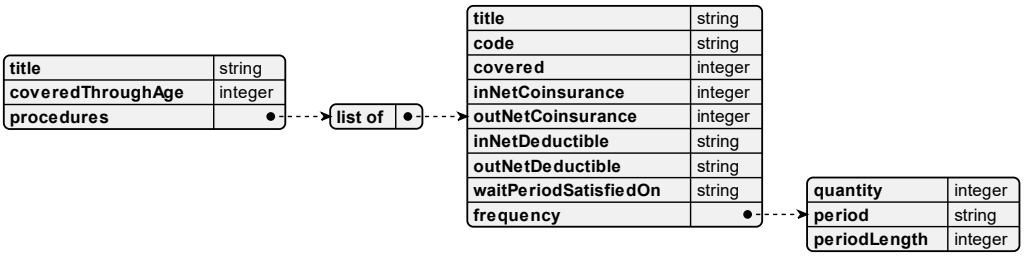

Diagram 4 Procedure Category Model

Dependents

The dental coverage object contains a list of dependents. These dependents are members also covered by the user's policy. The dependents have their own unique benefits and procedure coverage.

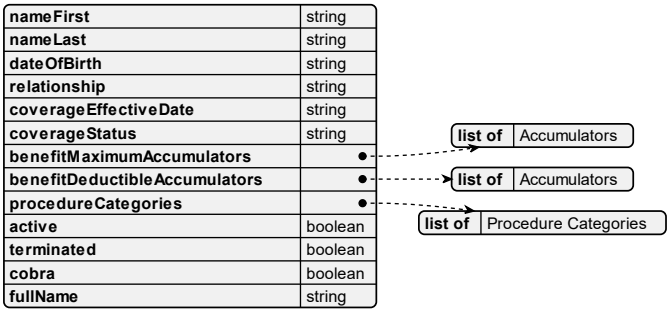

Diagram 5 Dependent Model

Dental Claims

A list of dental claims can be fetched for a user. This list of claims contains claims for all members covered by the policy. Displaying information for claims are simple as they contain the 'headers' and 'values' fields. These fields consist of a list of header strings that need to be displayed along with the corresponding value in the list of value strings.

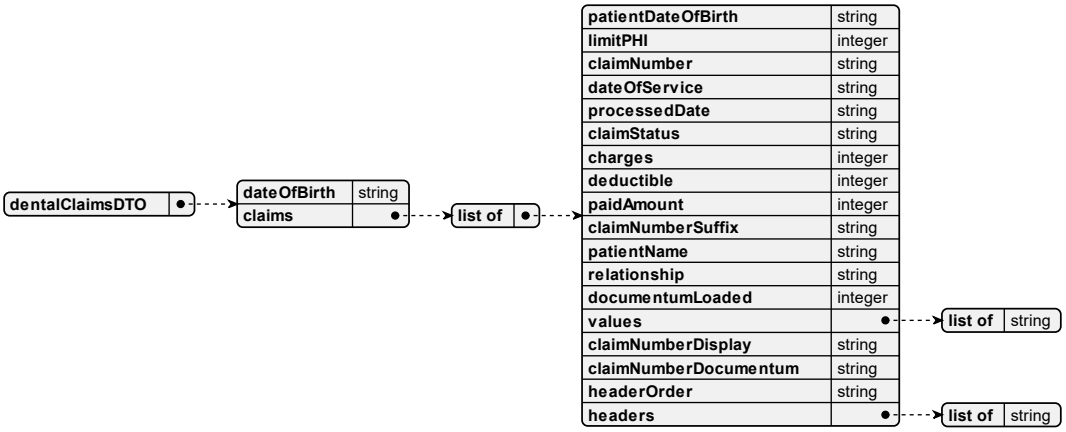

Diagram 6 Dental Claims Model

Dentist Location

The dentist locations are provided by the Google Places API. The API returns a list of search results. The most important part of these results is the location object within the geometry object. This location field provides a latitude and longitude value that can be used to display the location of the dentists.

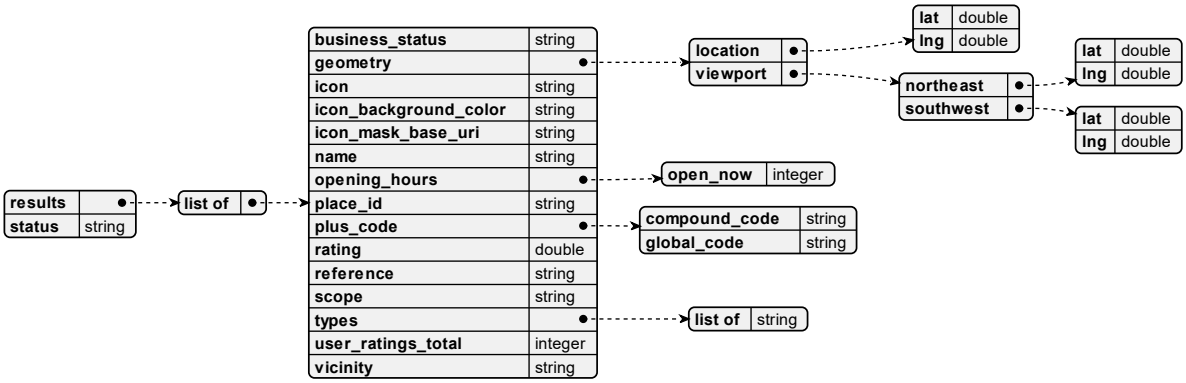

Diagram 7 Dentist Location Model

Implementation

Collaboration with Sun Life

Sun Life arranged meetings for every Monday. During these meetings we discussed issues from the previous week and features planned for development the next week. The team provided me with loads of resources, but most importantly, they provided me with their valuable time. Whenever I had issues, they were always there to help.

It is worth noting the only issue I had with the collaboration process was just the initial setup. I was provided with my own laptop for development and access to Sun Life services. The setup of the laptop took some time, and I had issues with access to certain resources that admittingly took too long to get sorted. This unfortunately caused a delay in development and planning in both the first and second semester. This is not entirely the team at Sun Life's fault. Some things take time, and I was also preoccupied with other assignments that led to things getting stretched out longer than it should have.

Development Environments

Before starting development environments needed to be setup. In Flutter environments are JSON files that get passed in as an argument when building the application. The variables declared in the environment can then be accessed within the application. Information such as API keys, and URLs can be stored within these files. For safety reasons this keeps the keys and sensitive information out of the source code as these files are ignored by Git. Sun Life's API is spread across different development, staging, and production regions. I set up different environments with the credentials for these regions.

Code Snippet 1 Environment File Example ./.env/dev.json.

```
{
  "ENV": "DEV",
  "WEB_PORTAL_API_BASE_URL": "URL_FOR_DENTAL_PORTAL_API",
  "BENEFIT_API_BASE_URL": "URL_FOR_BENEFITS_API",
  "GOOGLE_PLACES_API_KEY": "API_KEY_FOR_GOOGLE_PLACES"
}
```

Localization

According to the World Wide Web Consortium, localization refers to the adaptation of software to meet different languages and region requirements. The existing application and web portal is developed for a Northern American market therefor an implementation of a localization system was necessary to handle the multiple languages of America and Canada.

Flutter has its own library that handles localization. The only work needed from the developers are the translations. Translation strings are provided in separate JSON files each labelled as language codes. The library then generates its own code in the background when the application is compiled and creates a file that handles delegating the right string based on the locale of the application.

Here are some examples of the same page translated based on the locale of the application. I chose the 3 languages English, French, and Spanish.

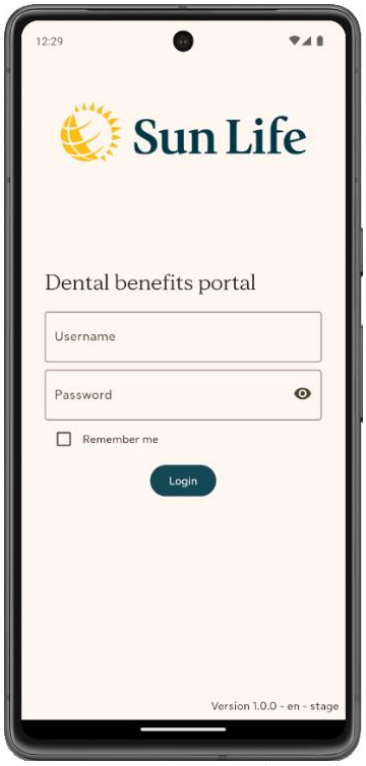

Figure 1 Login Screen (English Version)

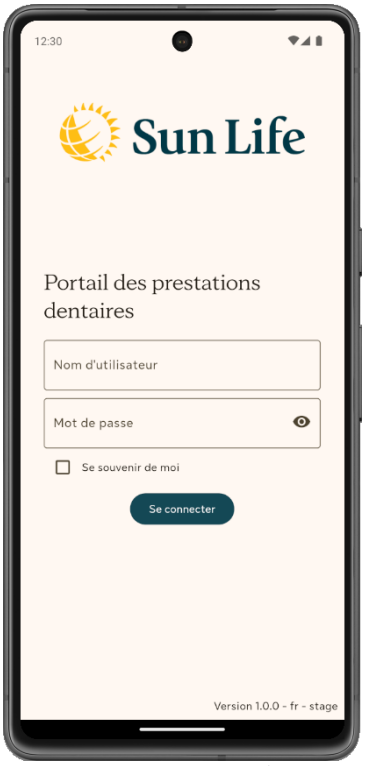

Figure 2 Login Screen (French Version)

Figure 3 Login Screen (Spanish Version)

User Interface

Flutter utilizes a design system known as Material3. Flutter therefor includes prebuilt components for building and designing UI layouts that utilizes this design system to keep a consistent look across the application. Material3 allows for global themes to be set that will translate a unified theme of colours and fonts across all the app's components.

Themes

Colours

Sun Life has a prominent brand that needs to be conveyed to users in this app. Using the following iconic Sun Life colours, I set up a Material3 theme for this application using a helpful tool provided on the Material3 tool called Material Theme Designer created by Google.

Figure 4 Theme Core Colours

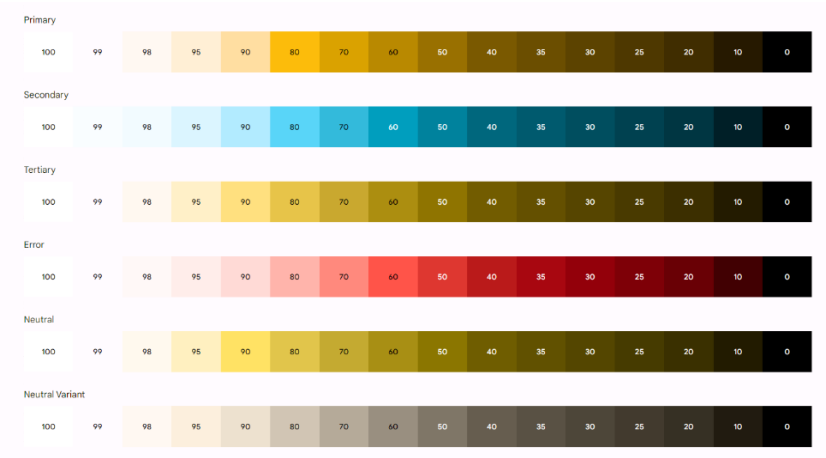

Figure 5 Theme Tonal Palette

Logos

Sun Life provided me with multiple variations of their logo that I got to use on the splash screen, app icon, and wherever it would fit throughout the app.

The app icon was generated using a tool created by Roman Nurik called Icon Kitchen. This would generate the icons in sizes and formats required for both Android and iOS.

Fonts

These are 2 fonts provided by Sun Life:

- Sun Life New Display
- Sun Life New Text

Screens & Features

Login Screen

This would be the first screen the user sees, and it has a simple interface to input a username and password. The “Remember me” checkbox will securely save the username for the next time the user needs to log in.

The login takes the username and password and sends a POST request to the old Mobile Benefit Tools API. If the response is successful, the payload is saved as a User Authentication model in memory. If the response is not successfully the corresponding error message will be displayed informing the user either something went wrong with the request, or that the username or password was incorrect.

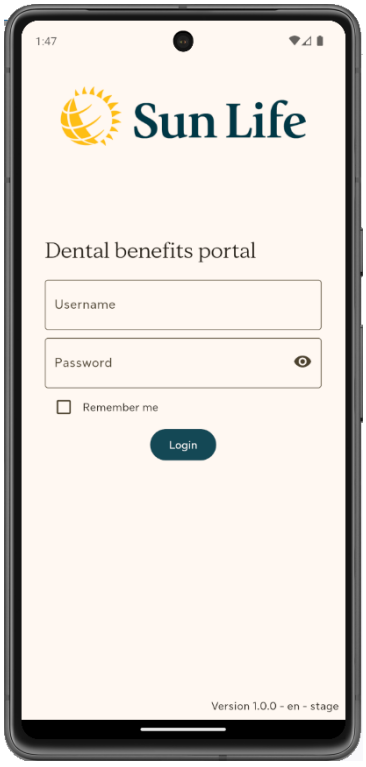

Figure 6 Empty Login Page

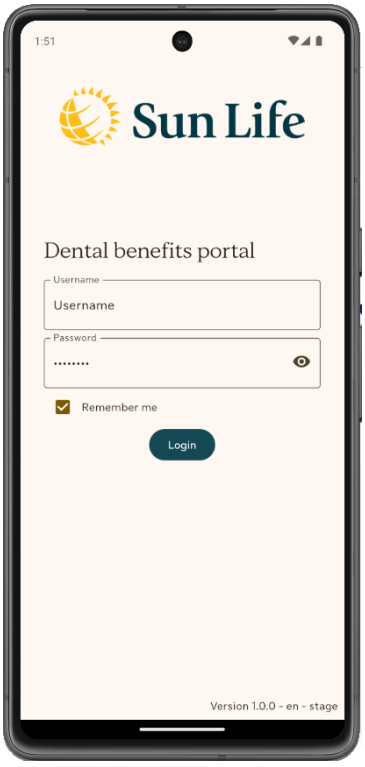

Figure 7 Filled Login Page

Home Screen

The home screen is the first screen the user will see after they login. The main goal for this screen is to display an overview of the user's dental policy. This page is broken down into the following sections.

Header Section

This displays the user's first and last name along with their member ID. There are two buttons on the top right that allows the user to view their profile or logout of their account.

Dental Coverage Overview Section

This section displays the bulk of the dental coverage information. It is broken down in its own separate section that will be covered in the Dental Coverage Overview Section.

Footer Section

The footer holds the Sun Life slogan "Life's brighter under the sun". Some links can also be found that leads the user to their corresponding webpage on Sun Life's website.

Navigation Section

The navigation bar at the bottom of the screen allows the user to navigate to other sections of the app. This bar is shown on the rest of the pages as well.

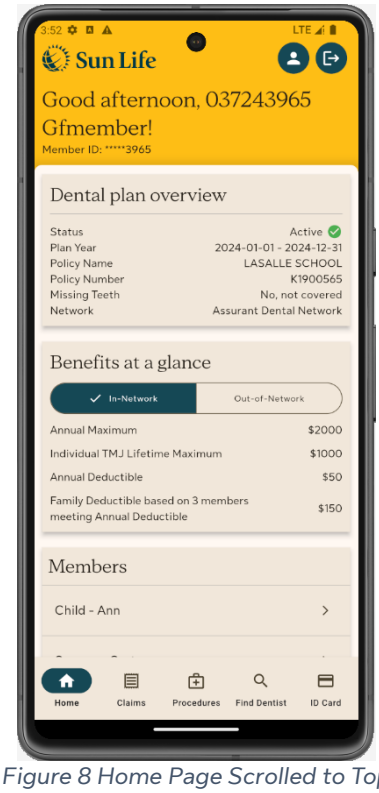

Figure 8 Home Page Scrolled to Top

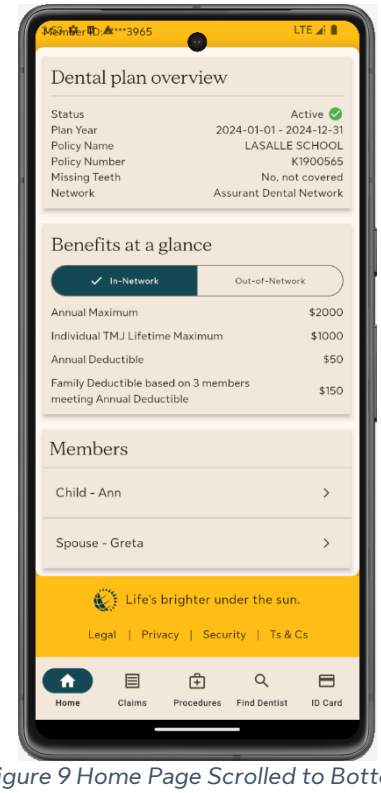

Figure 9 Home Page Scrolled to Bottom

Dental Plan Overview Section

All the data being displayed in these sections are retrieved from the newer dental benefits portal API. The data is retrieved from a single endpoint that returns the user's dental policy.

Figure 10 shows the dental plan section. The goal of this section is to display basic information about the user's policy.

Figure 12 shows a section that will list all the members that a user has on their dental policy. These members can be clicked on to open up the member details screen for the selected member.

Figure 11 and Figure 13 shows the benefits section. This section has a "in network" or "out network" toggle. The benefits provided by the user's policy is different depending on if it is within the Sun Life network or outside of it. This toggle shows the difference in coverage based on it being in or out of network.

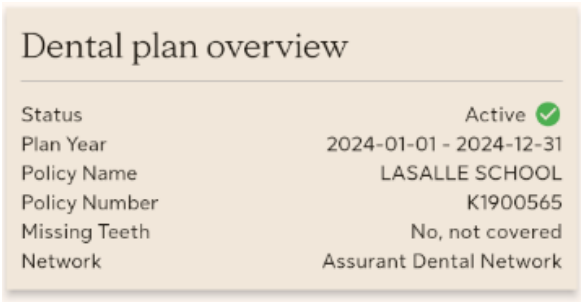

Figure 10 Dental Plan Overview Section

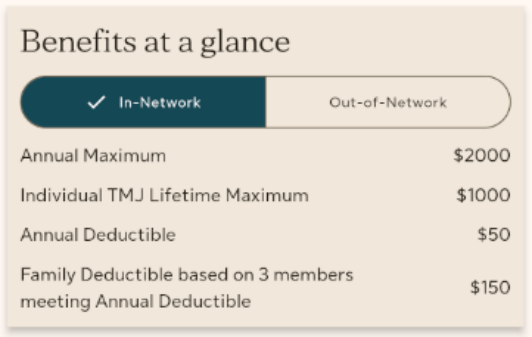

Figure 11 Benefits at a glance (in network)

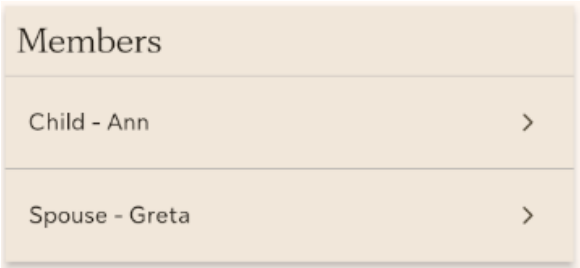

Figure 12 Policy Members Section

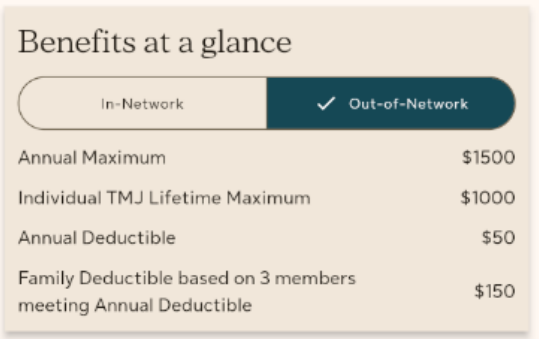

Figure 13 Benefits at a glance (out of network)

Dental Claims Screen

Dental claims are fetched from the dental benefits portal API. This page by default displays all the claims ordered by date of service. Each claim has a section that can be dropped down to display more information about the claim. More detailed information about a claim is provided by a document called an EOB. This is unfortunately not available as I had some small issue with the API call to retrieve the EOB. But the intended functionality of the button is to open up a page displaying the EOB in the same PDF viewer used to display the dental ID card.

The filter section on the top of the page originally starts with showing a dropdown to filter the claims by members. The search button on the right opens up the 'Search claim number' input box that can be used to search for a specific claim number.

The figures below show the different states of the claim and filter sections being dropped down or collapsed.

Figure 16 shows a filter applied to the list of claims.

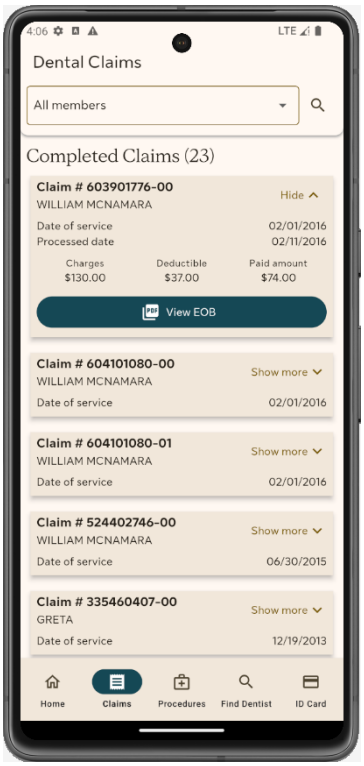

Figure 14 Claims Screen

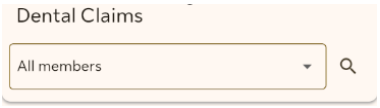

Figure 15 Claim Filter Without Search

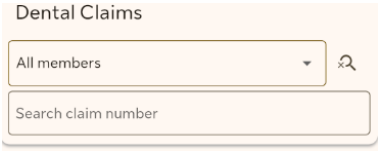

Figure 17 Claim Filter with Search

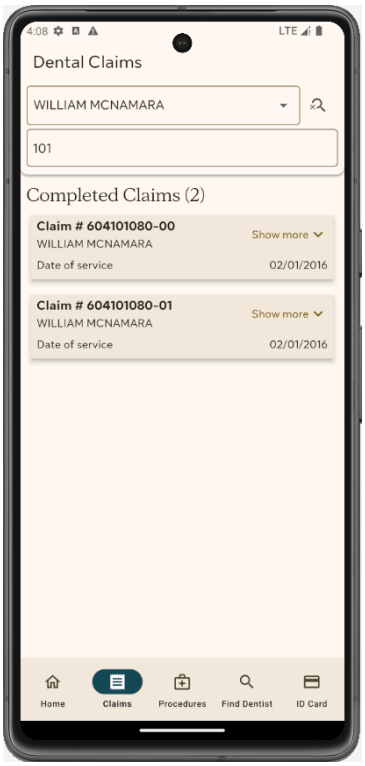

Figure 16 Claims Screen with Search Applied

Procedures Screen

These procedures are covered by the user's policy. They are provided by the dental portal API as a list of categories filled with procedures. Categories are displayed as an expandable section that reveals a list of procedures when expanded. Each of the procedures can be tapped on to open up a page that will view more detailed information about the procedure.

There is a search button on the top right of the screen that can be tapped to reveal a search input field. This search will filter the procedures based on their title and code. If the title of a category matches the search string, the entire category will be filtered out.

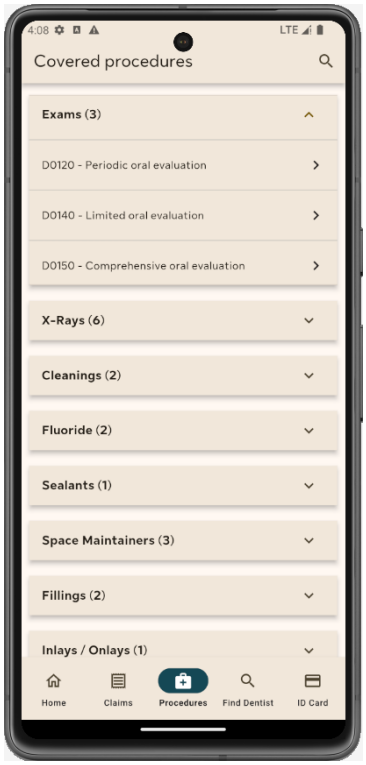

Figure 18 Covered Procedures Screen

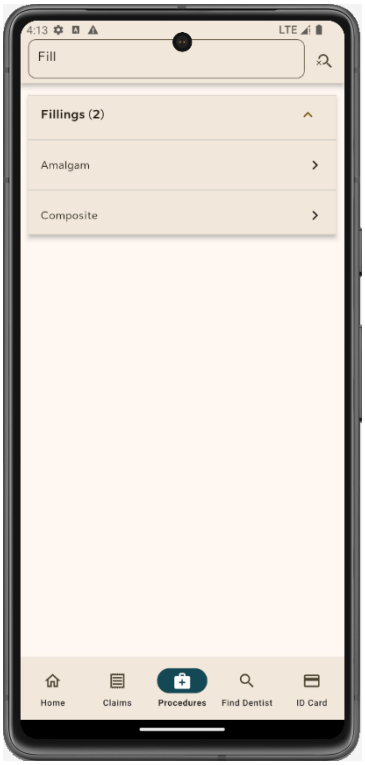

Figure 19 Covered Procedures Screen with Search Applied

Find Dentist Screen

This screen is probably the most complex. There are 3 different Google services working together form the full dentist search feature.

Figure 20 shows the default state this page will show when opened. This is the phone's current position retrieved from the GPS. The radius of the boundary is 1 mile, using miles for the North American target audience.

Figure 21 is the same as above, but with a selected location and the radius is set to 2 miles.

Figure 22 shows the location search feature displaying location suggestions.

Display Dentist Locations

The most obvious is the big Google map being displayed. This is done by using the Flutter Google map library.

Nearby Dentists

Originally the plan was to integrate Sun Life's own solution for searching for nearby dentists, but after complications I ended up using the Google nearby search service. Using this nearby search service, I retrieved the locations of all dentists within a certain radius of a position. This is done without libraries, and the request is sent and parsed manually.

Search Locations

The search bar uses a Google places library to find recommendations to a search prompt. When the user selects a location the location changes on the map and the dentist locations refresh. These search suggestions can be narrowed down to only specified countries, I set it to only include suggestions from the USA, Canada, and Ireland for testing purposes.

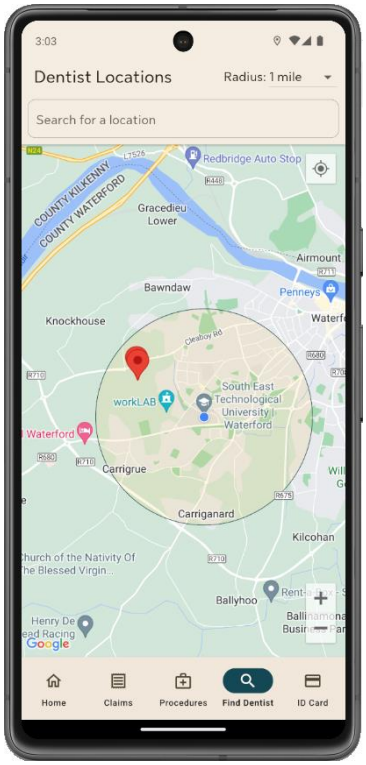

Figure 20 Find Dentist Default Location

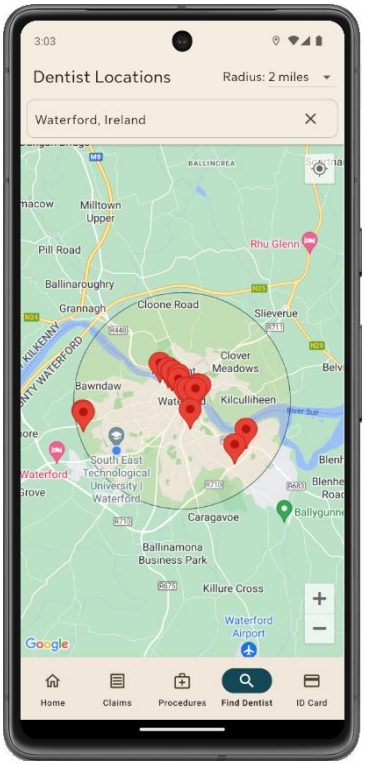

Figure 21 Find Dentist Provided Location

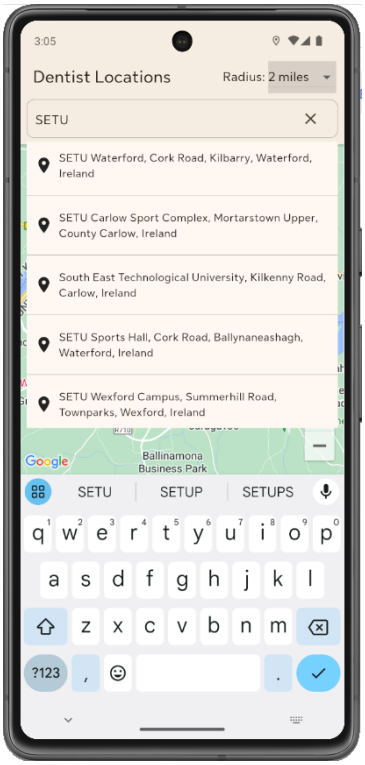

Figure 22 Find Dentist Location Suggestion

Dental ID Card Screen

The dental id card is provided through the API as a string of bytes. This page takes the bytes, processes them, and displays the PDF in a PDF viewer component.

The wallet button on the top right was intended to be used to add the PDF card to the user's Google Pay or Apple Pay wallet. This process turned out to be much more complicated than I had initially thought it would be. Looking for documentation on adding documents to the wallet I was required to create a business account in order to access the Google Pay Console, and I was not interested in getting into business accounts etc. In the end I set the feature aside and moved on with other requirements.

Figure 23 Dental Card Screen

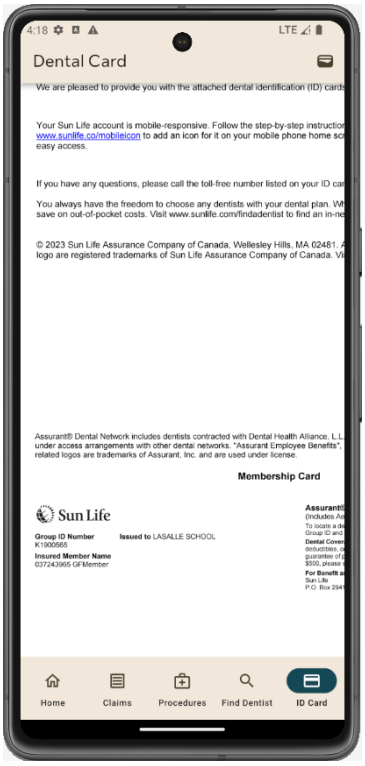

Figure 24 Dental Card Screen with Zoom

Member Screen

When a member is clicked on in the home screen, this screen is shown. Basic information such as name, relationship to the policy holder, and date of birth is displayed first.

Then the 'Member Benefits' section displays the benefits the policy provides to this member. The benefits are off course different based on if it is being covered in the Sun Life network, or outside the network.

The 'View member claims' is a handy shortcut button that opens up a separate dental claims page. This page will automatically be filtered to the member that was loaded onto the member screen.

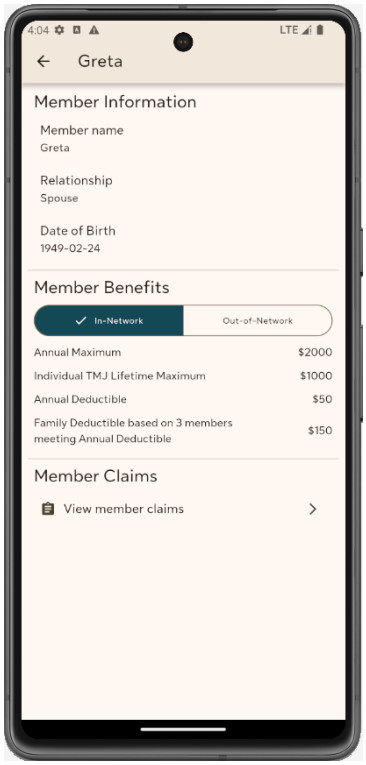

Figure 25 Policy Member Details Screen

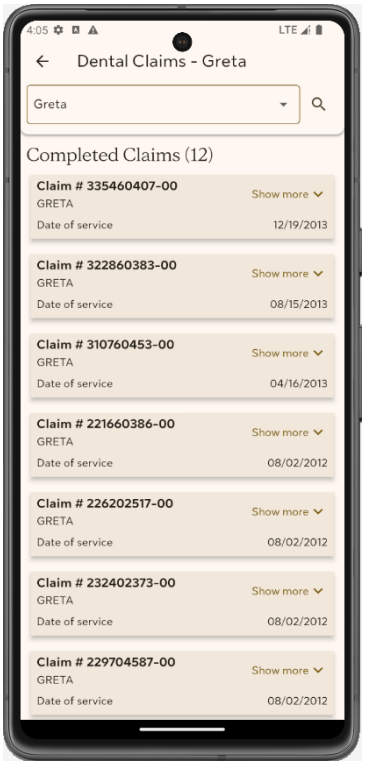

Figure 26 Dental Member Claims Screen

Selected Procedure Screen

When a procedure is clicked on in the covered procedures screen, this page will be opened up to display more detailed information about the procedure.

Once again there is an ‘in or out of network’ toggle present. Coverage for procedures is different based on if the procedure is done in the Sun Life network or outside.

User Profile Screen

This is another simple page. There is no complex functionality here. The main goal of this page is to display the basic user profile information.

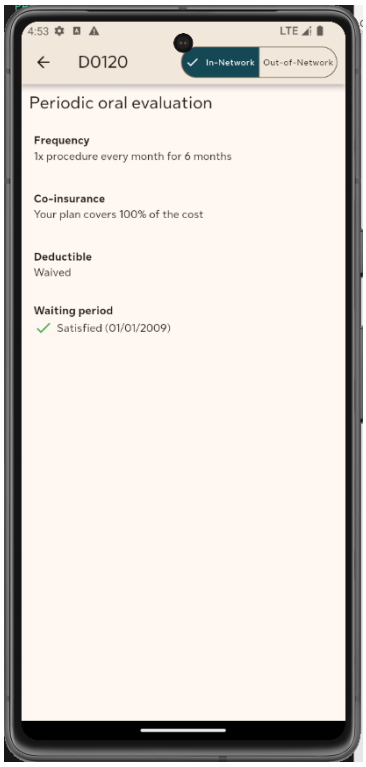

Figure 27 Selected Procedure Screen

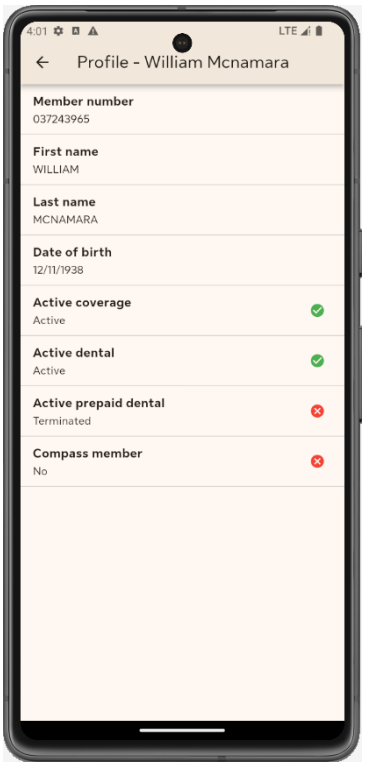

Figure 28 User Profile Screen

Network Access

API Calls

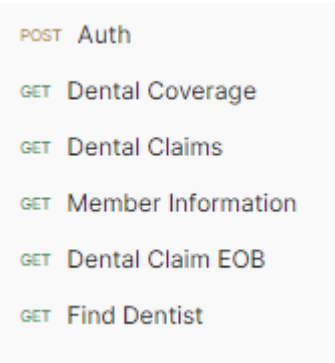

Figure 29 API Calls Overview

Figure 29 shows an overview of the endpoints used in this application. Due to the nature of the dental web portal API most of the provided endpoints were GET requests. The web portal is mostly used to provide information to the users. Most of the information is provided by the GET Dental Coverage endpoint. This reduced the number of unique requests needed.

More information about the endpoints used in this project is available on Postman by following the link in the **Error! Reference source not found.** or clicking here.

JSON Serialization in Flutter

Flutter has no access to libraries such as GSON that utilizes reflection to take care of JSON serialization. Instead, it has a library that generates files with the appropriate code to serialize from and to JSON. Instead of the library approach, I opted for the JSON to Dart tool developed by Javier Lecuona. Using this I could throw the response JSON into the generator and get a model class created for me instead of having to build the models myself.

Testing Architecture

Originally testing was planned to be done using a tool called Appium. This is the current solution for testing the existing Sun Life mobile application. After doing research on testing using Flutter, I found that Flutter's built in testing toolkit is more than capable for our needs and handles all levels of testing including unit, widget, and full integration testing. Mock data can be inserted into the testing environment using a library called Mockito, but this was not necessary for this project's testing.

Unit Testing

Unit testing refers to testing small parts of logic in the code. This includes a function, method, or the functionality of a single class. The goal is to test logic under a set of conditions. These tests tend to be small and fast to run. (Google LLC 2024)

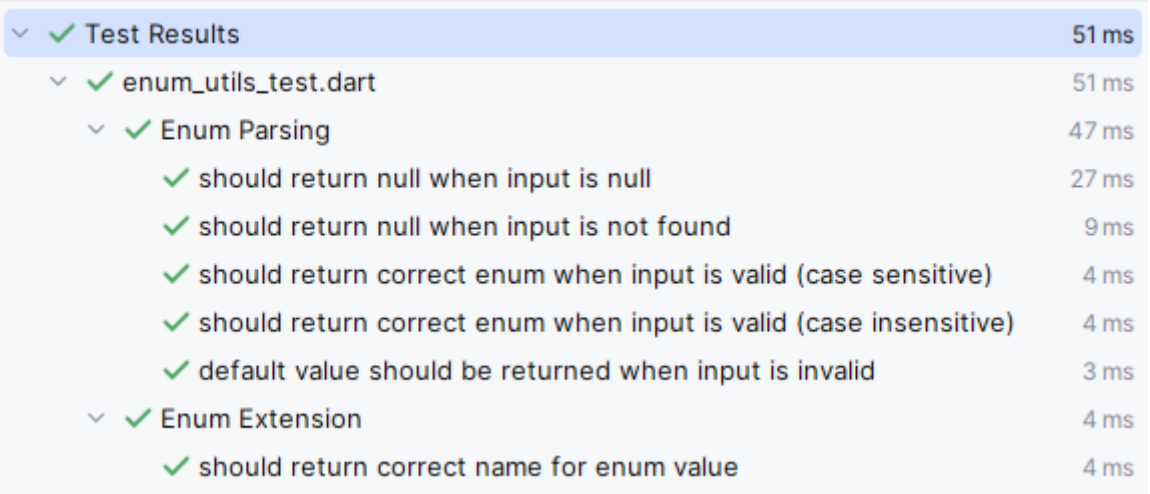

Figure 30 Unit Test Results

Widget Testing

Widget testing or also known as component testing is tests written to test the behaviour and responses of UI widgets. These tests can check if certain UI components are visible when required, and that they respond appropriately to input. These tests are usually more complex than unit tests, but they should be kept simple enough to test a single widget instead of an entire user interface. (Google LLC 2024)

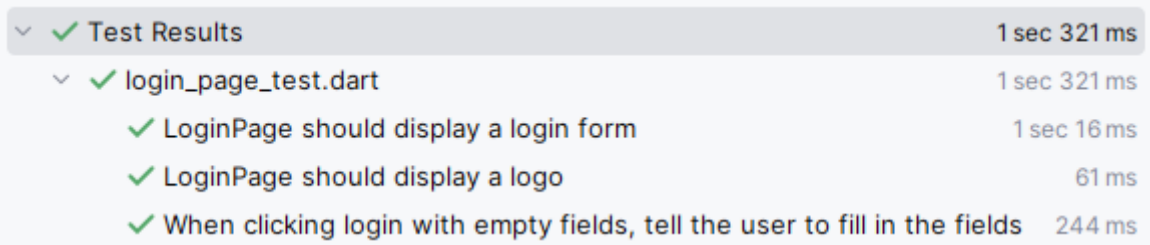

Figure 31 Widget Test Results

Integration Testing

Integration testing is the wider scope where the entirety of the application or certain systems is tested. These tests are usually very complex and can take some time to run. They are not very important but can be useful for big applications in production. Another use is to test the performance of the entire system using integration tests. Real devices or OS emulators are used when doing full integration tests. (Google LLC 2024)

Unfortunately, this project does not have any integration tests.

Future Work

Some plausible future work would be finishing the unfinished features. Some of these features include, fixing the 'Fetch Claim EOB' button and looking into adding the Dental ID card to the Google Pay and/or Apple Pay wallets.

Swapping the current Google Places API for a Sun Life specific searching service is a possible upgrade in the future.

Adding more tests can be useful to test the stability of the system and keep the code robust.

Unfortunately, due to the nature of the dental web portal, there are not a lot of data manipulation from the API in this application. The app mostly fetches and displays data. Using other Sun Life APIs and services, the functionality to change data, or even submit claims can be added into the app.

Reflection

The first big decision I had to make for development in this project was to decide on an appropriate framework. I learned a lot about both Flutter and React Native during this process. Ultimately, I'm satisfied with the choice I had made. I learned so much about mobile development using Flutter and Dart. I am happy to have expanded my skillset and I now have another tool to utilize for mobile app projects like this.

Collaborating with Sun Life was a great learning experience. The team took their own precious time to have meetings with me and teach me all about their systems. I had a meeting where I was taught about SAFe Scrum, which is like Scrum we learned about in class, but scaled up to corporate levels. This was great to see class work being applied in real life scenarios.

This topic of class work being applied to aspects of this project came up a few times. Without the knowledge of my earlier mobile app development classes, learning Flutter and Dart would have been a lot more difficult. Developing UI in Flutter is similar to Kotlin which I had just learned last year. Accessing the endpoints of the APIs would have been a lot more complicated if I had not learned about how APIs worked and how to access them in earlier web development classes.

I also learned a lot of new development strategies while working on this project. I had learned techniques on how to handle states, how to manage localizations on apps, and much more.

Conclusion

Since applying for the project from SunLife almost 7 months ago, I had learned an incredible amount. Simple things about writing reports, to learning entire new programming languages. I feel I have grown as a Software Engineer. I felt like my past 4 years of study had finally paid off with the development on this entire project. I am extremely grateful to have had the opportunity to do this project with the people that helped me along the way.

Special Thanks

Special thanks to my supervisors for this project. Both Sonya Hogan and Siobhan Roche gave me excellent feedback and advice.

Thanks to the team in Sun Life, who sacrificed their own time to help me with issues and give me presentations on how systems work in Sun Life. They provided me with resources about the existing projects, Sun Life brand, how SAFe works and much more.

I am truly grateful to the support provided by my supervisors and the Sun Life theme.

Acknowledgement of Third-Party Assets

The icons, colours, fonts, and other design elements utilized in this application and report are owned and copyrighted by Sun Life. Their usage within this project is acknowledged and appreciated.

References

Cloud Devs Inc (2024) *React Native and Expo: Simplifying Development and Deployment*, available: https://clouddevs.com/react-native/expo/ [Accessed 16 Apr. 2024]

Google LLC (2024) *Flutter*, available: https://flutter.dev/ [Accessed 15 Apr. 2024]

Google LLC (2024) *Material Theme Builder*, available: https://m3.material.io/theme-builder [Accessed 18 Apr 2024]

Google LLC (2024) *Testing Flutter apps*, available: https://docs.flutter.dev/testing/overview [Accessed 21 Apr. 2024]

Inflectra Corporation (2024) *Agile Scrum Methodology Explained*, available: https://www.inflectra.com/Solutions/Methodologies/Scrum.aspx

Lecuona J. (2024) *JSON to Dart*, available: https://javiercbk.github.io/json_to_dart/ [Accessed 18 Apr. 2024]

Nurik R. (2024) *Icon Kitchen App Icon Generator*, available: https://icon.kitchen/ [Accessed 18 Apr.2024]

World Wide Web Consortium (2024) *WHAT IS INTERNATIONALIZATION*, available: https://www.w3.org/International/i18n-drafts/nav/about [Accessed 18 Apr. 2024]

Appendices

Benefits tools app webpage

https://www.sunlife.com/us/en/mobile-apps/benefits-tools-app/

API Reference

Dental Benefits Mobile Application API Reference
https://documenter.getpostman.com/view/32058537/2sA3BrX9hA