

Capstone Stage 1

Description

Intended User

Features

User Interface Mocks

Screen 1

Screen 2

Screen 3

Screen 4

Screen 5

Screen 6

Screen 6

Screen 7

Screen 8

Screen 9

Screen 10

Screen 11

This screen allows the user to track their income on a yearly basis. This is used to calculate the government pension (Social Security in the US) that a person is entitled to. The FAB allows the user to enter salary data for a desired year.

Screen 12

Key Considerations

How will your app handle data persistence?

Describe any corner cases in the UX.

Describe any libraries you'll be using and share your reasoning for including them.

Describe how you will implement Google Play Services.

Next Steps: Required Tasks

Task 1: Project Setup

Task 2: Implement UI Login Activity

Task 3: Implement Firebase

Task 4: Implement ContentProvider

Task 5: Design Sqlite tables and corresponding JSON tables

Task 6: Create Navigation Drawer

Task 7: Actual Expenses

Task 8: Retirement Expenses

Task 9: Income Tracking

Task 10: Create Summary Activity

Task 11: Create Yearly Tax Table Activity

Task 12: Create Milestone Activity

GitHub Username: emuhlestein

Retirement Helper

Description

The app is intended to help people plan for their retirements. It will help them track their monthly expenses and the amount of income they can expect to receive in retirement. These will be tracked over time and the user can extrapolate out in the future to see when they can retire, that is when their projected retirement income will be more than their expenses. The user can set milestones. These are important ages. This will allow the user to see how much retire income they can receive at various ages. This will help them identify the age at which they can reasonably retire.

Intended User

This is for anyone who is interested in retiring some day.

Features

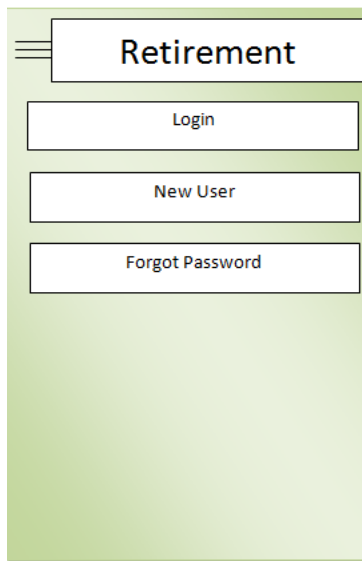
- Estimate Social Security benefits.
- Estimate spousal benefits.
- Estimate taxes during retirement.
- Track other sources of income: savings, 401(k), investments, ...
- Track monthly expenses, actual and retirement-only expenses.
- All income and expenses will be tracked in an Sqlite database.
- Extrapolate expenses and income to project a retirement date.
- Show mile stones: key mile stones in US: age 59 ½, 62, full retirement age, ...
- Track personal data like birthday so mile stones can be determined.

- Graphical show monthly expenses and income.
- Graphical show when a person can expect to retire.
- Firebase will be backend so information can be saved and shared.

User Interface Mocks

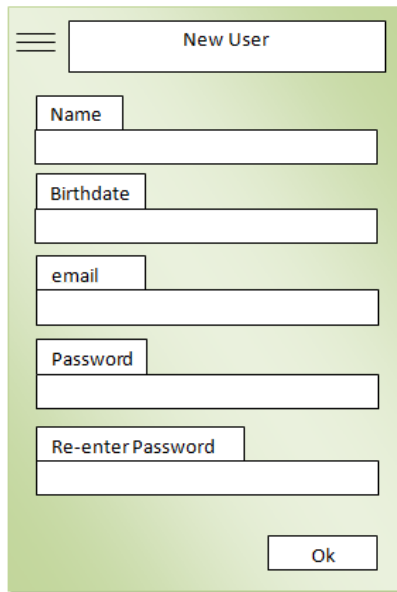
Each of the primary screens will have an associated navigation draw.

Screen 1



This is the login screen. This is where a new user will create a new login or where an existing user can login. Also, this screen allows an existing user to reset their password. The login will be their email address.

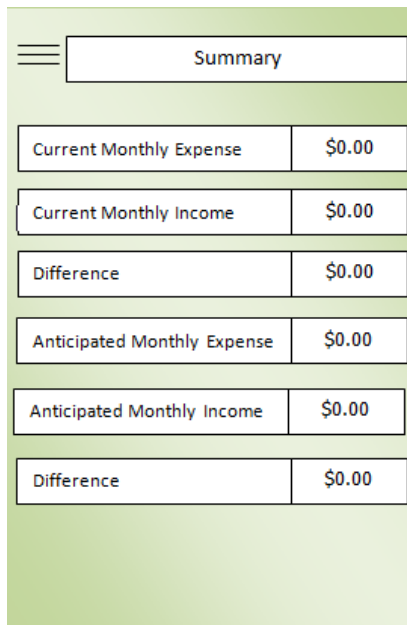
Screen 2



The 'New User' registration screen features a green header with a hamburger menu icon and a title bar. Below the header, there are five input fields: 'Name', 'Birthdate', 'email', 'Password', and 'Re-enter Password'. Each field has a label box on the left and a text input area on the right. At the bottom right, there is an 'Ok' button.

This is the new user registration screen. A user will need to enter their name, date of birth, their email and a password. The password is reentered to ensure the password is correct.

Screen 3

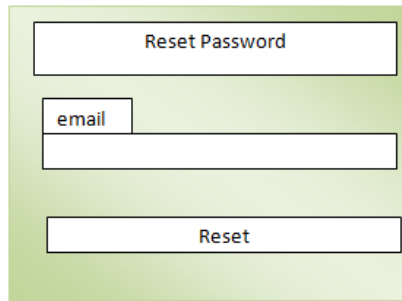


The 'Summary' screen has a green header with a hamburger menu icon and a title bar. Below the header, there are six rows of data, each consisting of a label box and a value box. The data is as follows:

Summary	
Current Monthly Expense	\$0.00
Current Monthly Income	\$0.00
Difference	\$0.00
Anticipated Monthly Expense	\$0.00
Anticipated Monthly Income	\$0.00
Difference	\$0.00

This is the summary screen, It shows the current monthly expenses, current monthly income, the anticipated (retirement) expenses and the anticipated (retirement) income. It also shows the difference for each of these.

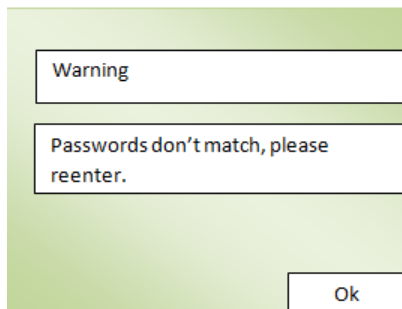
Screen 4



A screenshot of a 'Reset Password' screen. It features a light green background. At the top, there is a white rectangular button labeled 'Reset Password'. Below this, there is a label 'email' next to a white text input field. At the bottom, there is a white rectangular button labeled 'Reset'.

This is the reset password screen. This allows a user to reset their password. An email will be sent to them with instructions.

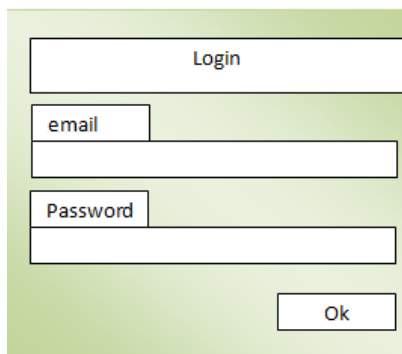
Screen 5



A screenshot of a warning dialog box. It has a light green background. At the top, there is a white rectangular box labeled 'Warning'. Below it, there is a larger white rectangular box containing the text 'Passwords don't match, please reenter.'. At the bottom right, there is a white rectangular button labeled 'Ok'.

Warning dialog that warns the user the passwords entered do not match.

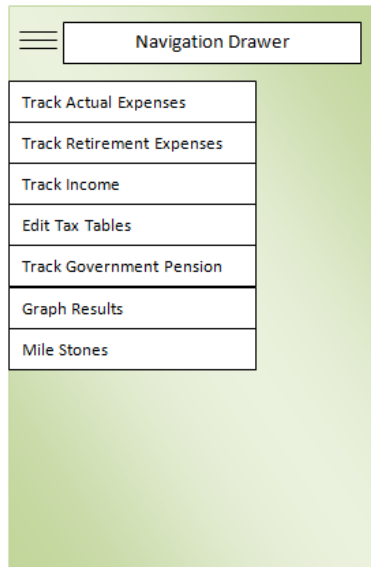
Screen 6



A screenshot of a 'Login' screen. It has a light green background. At the top, there is a white rectangular button labeled 'Login'. Below this, there is a label 'email' next to a white text input field. Below the email field, there is a label 'Password' next to a white text input field. At the bottom right, there is a white rectangular button labeled 'Ok'.

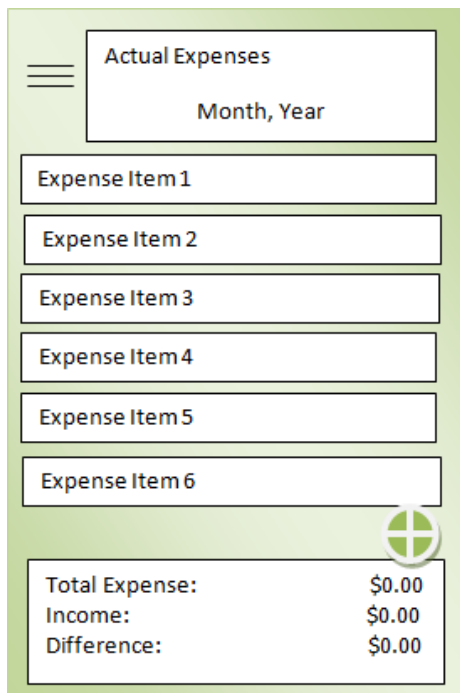
The normal login screen.

Screen 6



This is the navigation drawer.

Screen 7



This screen is where the user tracks their actual expenses. This is a list of the user's expenses. It shows the category and the amount. These are monthly expenses, not daily expenses. The FAB will allow the user to enter new expense categories. The user will be able to edit amounts.

Screen 8

A dialog box with a light green background. It contains three input fields: 'New Category' at the top, 'Category' below it, and 'Amount' below that. The 'Amount' field is pre-filled with '\$0.00'. At the bottom right, there are two buttons: 'Cancel' and 'Ok'.

This dialog allows the user to enter in an amount of the selected expense category.

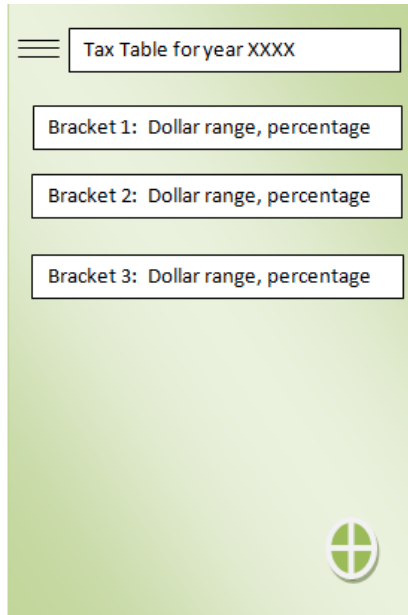
Screen 9

A screen with a light green background. At the top left is a hamburger menu icon. To its right is a box containing the text 'Retirement Expenses' and 'Month, Year'. Below this is a list of six input fields labeled 'Expense Item 1' through 'Expense Item 6'. At the bottom, there is a box containing a summary table:

Total Expense:	\$0.00
Income:	\$0.00
Difference:	\$0.00

This screen is where the user tracks their retirement expenses. This is a list of the user's expenses that would be incurred in retirement. It shows the category and the amount. These are monthly expenses, not daily expenses. The user will be able to edit amounts. The reason for this is that retirement expenses probably will be different than our normal expenses. For example, if there are repairs made on a second vehicle and you only plan on having one vehicle in retirement, this expense would be recorded in retirement expenses. That is this amount would be subtracted from what is in the actual expenses for this category.

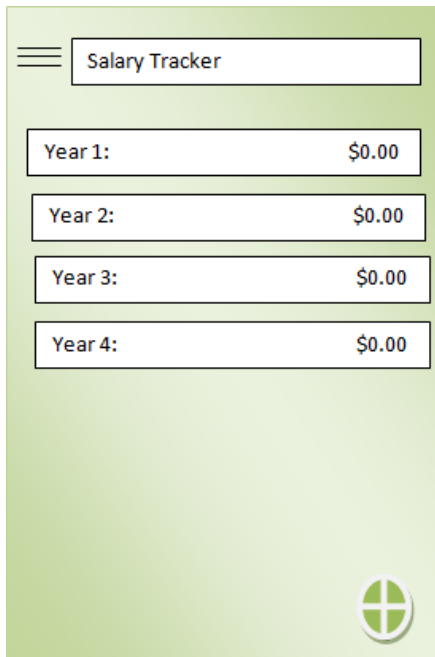
Screen 10



Screen 10 is a mobile application interface for a tax table. It features a light green background. At the top left, there is a hamburger menu icon (three horizontal lines) next to a title bar that says "Tax Table for year XXXX". Below the title bar, there are three input fields, each labeled "Bracket 1: Dollar range, percentage", "Bracket 2: Dollar range, percentage", and "Bracket 3: Dollar range, percentage". At the bottom right, there is a green circular button with a white plus sign inside.

This screen allows the user to track income taxes. The user can add as many tax brackets as needed. This is important for calculating net income in retirement. (It would be nice if this data could be downloaded from the internet. I have not been able to find out how to do that, so the user will have to enter it manually).

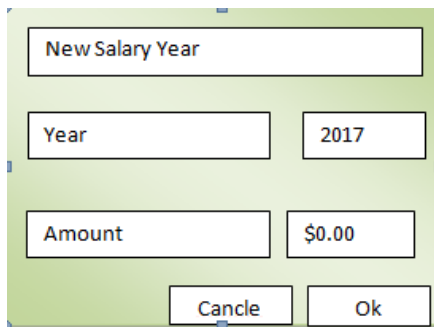
Screen 11



Screen 11 is a mobile application interface for a salary tracker. It features a light green background. At the top left, there is a hamburger menu icon (three horizontal lines) next to a title bar that says "Salary Tracker". Below the title bar, there are four input fields, each labeled "Year 1:", "Year 2:", "Year 3:", and "Year 4:". Each input field has a value of "\$0.00" displayed to its right. At the bottom right, there is a green circular button with a white plus sign inside.

This screen allows the user to track their income on a yearly basis. This is used to calculate the government pension (Social Security in the US) that a person is entitled to. The FAB allows the user to enter salary data for a desired year.

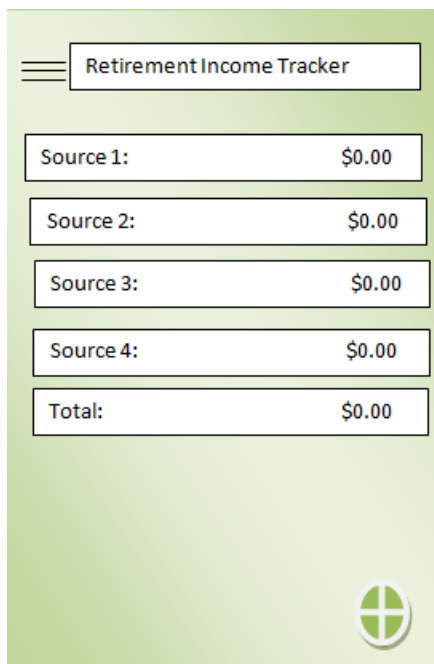
Screen 12



The screenshot shows a dialog box titled "New Salary Year". It contains two input fields: "Year" with the value "2017" and "Amount" with the value "\$0.00". At the bottom, there are two buttons: "Candle" and "Ok".

This dialog will allow the user to enter salary for the desired year. The year field will be type in.

Screen 13



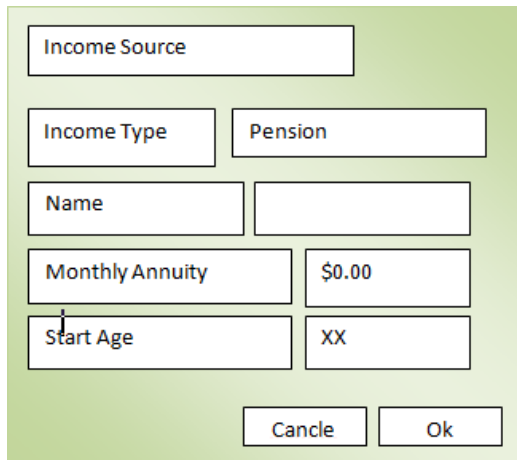
The screenshot shows a screen titled "Retirement Income Tracker". It features a list of four sources, each with a label and a corresponding amount:

Source	Amount
Source 1:	\$0.00
Source 2:	\$0.00
Source 3:	\$0.00
Source 4:	\$0.00
Total:	\$0.00

At the bottom right corner, there is a green circular icon with a white plus sign inside.

This screen is used to track retirement income from any source. This screen shows the source and the expected monthly amount from the source.

Screen 14

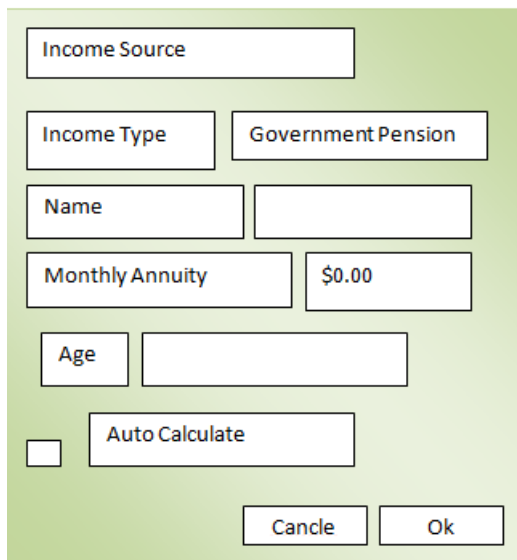


A screenshot of a software interface for defining an income source. The form has a light green background and contains several input fields and buttons. At the top is a single-line text field labeled 'Income Source'. Below it are two side-by-side fields: 'Income Type' containing the text 'Pension' and an empty 'Name' field. Further down are two more side-by-side fields: 'Monthly Annuity' containing '\$0.00' and 'Start Age' containing 'XX'. At the bottom right are two buttons labeled 'Cancel' and 'Ok'.

This screen is one of three types of income sources. This income source is a pension. It has a name, a monthly annuity and a the age at which the pension begins.

The currently selected income type will determine which views are displayed below. This will be handled via fragments.

Screen 15



A screenshot of a software interface for defining a government pension income source. The form has a light green background. It features a 'Income Source' field at the top. Below it, the 'Income Type' field is set to 'Government Pension'. There is an empty 'Name' field, a 'Monthly Annuity' field set to '\$0.00', and an 'Age' field. At the bottom left, there is a small square checkbox followed by an 'Auto Calculate' button. At the bottom right are 'Cancel' and 'Ok' buttons.

The next type of income is a government pension like Social Security. There is a name and a monthly annuity expected, and a the age at which the annuity begins. This can also be auto calculated based on salary history.

Screen 16

Income Source	
Income Type	Savings
Name	
Current Balance	\$0.00
Monthly Contribution	\$0.00
Expected APR	%
Amount to withdraw	\$0.00
<div>CancelOk</div>	

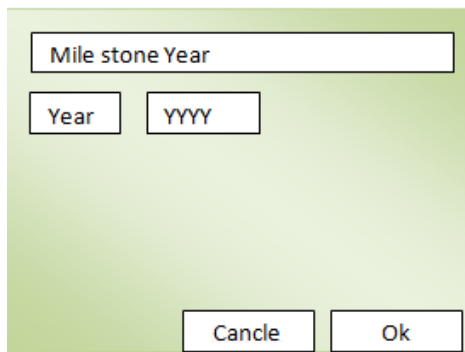
This income source is savings. The name of the institution, the current balance, monthly contributions and expected APR. In this release, this screen will also have to do for other types of investments like brokerage accounts, 401(k)s and IRAs.

Screen 15

☰	Mile Stones
Age A: Amount of retirement income	
Age B: Amount of retirement income	
Age C: Amount of retirement income	
<div></div>	

This screen will allow the user to set mile stones. For example, in the US 59 ½ is when 401(k) contributions can be withdrawn without penalty. 62 is the minimum age when Social Security can be collected. At each mile stone (age), the app will determine how much retirement income the user can expect to receive. The FAB allows the user to add milestones.

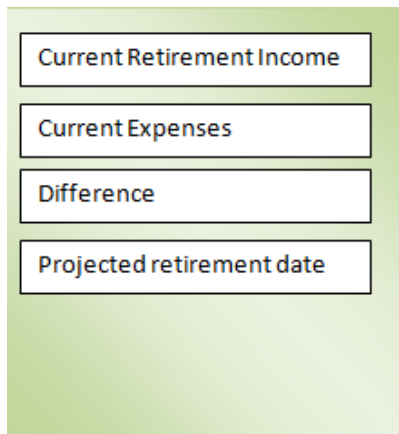
Screen 16



A dialog box with a light green background. At the top is a text input field labeled "Mile stone Year". Below it are two smaller input fields: "Year" and "YYYY". At the bottom right are two buttons: "Candle" and "Ok".

Dialog for adding milestone year.

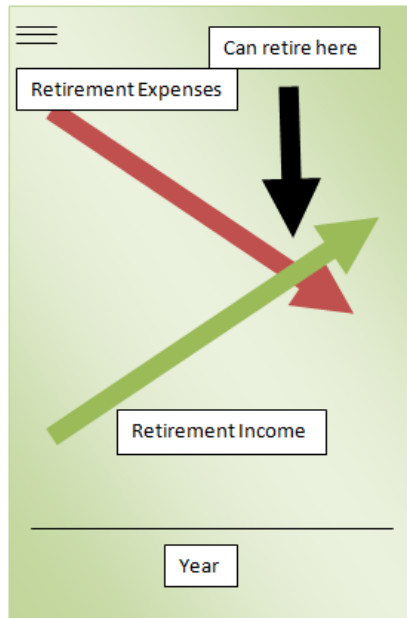
Screen 17



A screen with a light green background. It contains four stacked rectangular boxes, each with a label: "Current Retirement Income", "Current Expenses", "Difference", and "Projected retirement date".

The App Widget will display the user's current expense and the current retirement income that would be generated from all sources. Of course, certain retirement income will not be available at any time. Certain age criteria must be met before the income becomes available. Also, it will display the different in the income and expenses. And it will display the projected retirement date.

Screen 18



This screen will display a graph of the user's expenses versus projected retirement income. When the lines cross, it's time to retire!

For later releases, there could be additional income source types added, like, as was already mentioned, like brokerage accounts, 401(k)s and IRAs.

Key Considerations

Accuracy

The values calculated for a government pension are estimates. The user needs to be aware that all the amounts are estimates. This app is only intended to help the user estimate what their retirement income will be.

How will your app handle data persistence?

All the data will be persisted locally in an Sqlite data base. A content provider will be used. The data will also be persisted in the cloud using Firebase. This will allow multiple people you review the data.

Describe any corner cases in the UX.

I don't anticipate any corner cases. There will be a navigation drawer that will quickly and easily take the user to where they want to go.

Describe any libraries you'll be using and share your reasoning for including them.

Firebase for writing data to cloud and maintaining user accounts. This allows for the sharing of data. Butterknife makes UI code more manageable and easier to implement. MPAndroidChart for doing graphs.

Describe how you will implement Google Play Services.

Firebase has already been mentioned. Sign-In to allow users to authenticate using their Google account. Analytics to track how the user interacts with the app.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

- Install and configure Firebase in Android studio
- Install and configure Google Play Services.
- Install and configure ButterKnife.
- Install and configure MPAndroidChart.

Task 2: Implement UI Login Activity

- Build the Login Activity.
- Build the New User fragment.
- Build the Login dialog.
- Build the Reset Password.
- Build the Warning dialog.

Task 3: Implement Firebase

- Hook up UI to Firebase.
- Design JSON User structure to write to Firebase.
- Create an IntentService for handling the Firebase calls.

Task 4: Implement ContentProvider

- Implement contract
- Implement query method.

- Implement add method.
- Implement delete method
- Implement update method

Task 5: Design Sqlite tables and corresponding JSON tables

- Create User table.
- Create Actual Expense table.
- Create Retirement Expense table.
- Create Expense Category table.
- Create Tax Table table.
- Create Mile Stone table.
- Create Salary Table.
- Create Income Source table.
- Create all associated JSON table structures.

Task 6: Create Navigation Drawer

Task 7: Actual Expenses

- Create activity for actual expenses.
- Create RecyclerView for list. List will be loaded with CursorLoader.
- Create dialog for adding expenses categories.
- Create dialog for adding and editing expenses.
- Hook up activity to Navigation Drawer.
- Write expenses to cloud using firebase.
- Read expenses from cloud.

Task 8: Retirement Expenses

- Create activity for retirement expenses.
- Create RecyclerView for list. List will be loaded with CursorLoader.
- Reuse dialog for adding expenses categories.
- Reuse dialog for adding and editing expenses.
- Hook up activity to Navigation Drawer.
- Write expenses to cloud.
- Read expenses from cloud.

Task 9: Income Tracking

- Create Income Tracker Activity
- Create RecyclerView for list. List will be loaded with CursorLoader.
- Create fragment for pension.
- Create fragment for government pension.
- Create fragment for savings.

- Save income data to cloud.
- Read income data from cloud.

Task 10: Create Summary Activity

Task 11: Create Yearly Tax Table Activity

- Create activity.
- Create RecyclerView for list. List will be loaded with CursorLoader.

Task 12: Create Milestone Activity

- Create activity.
- Create RecyclerView for list. List will be loaded with CursorLoader.