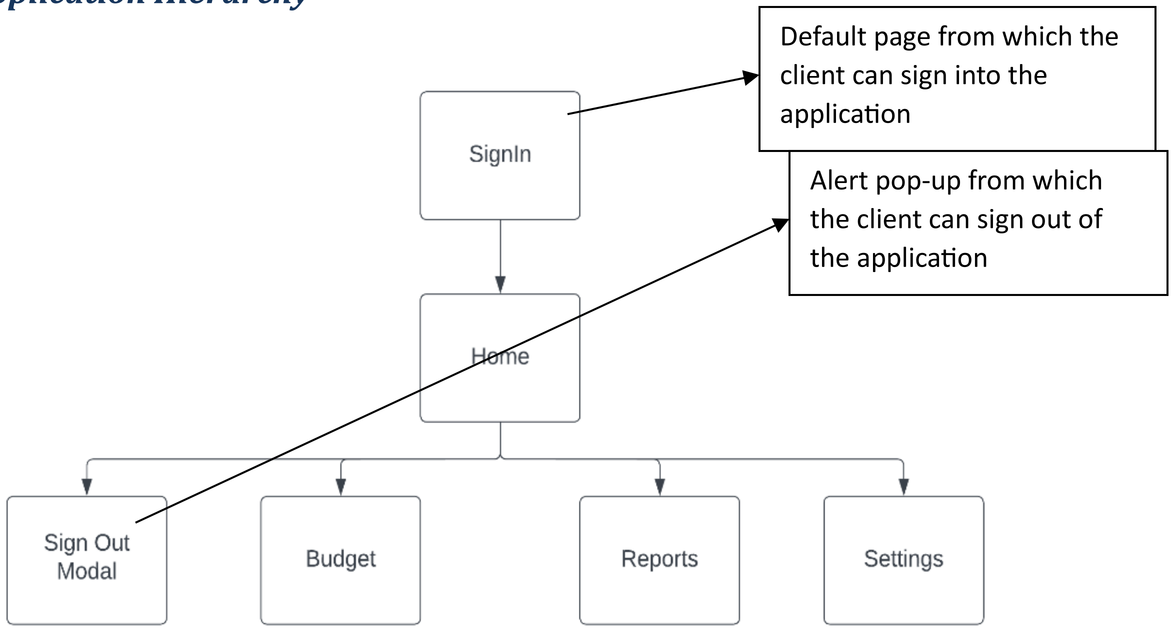
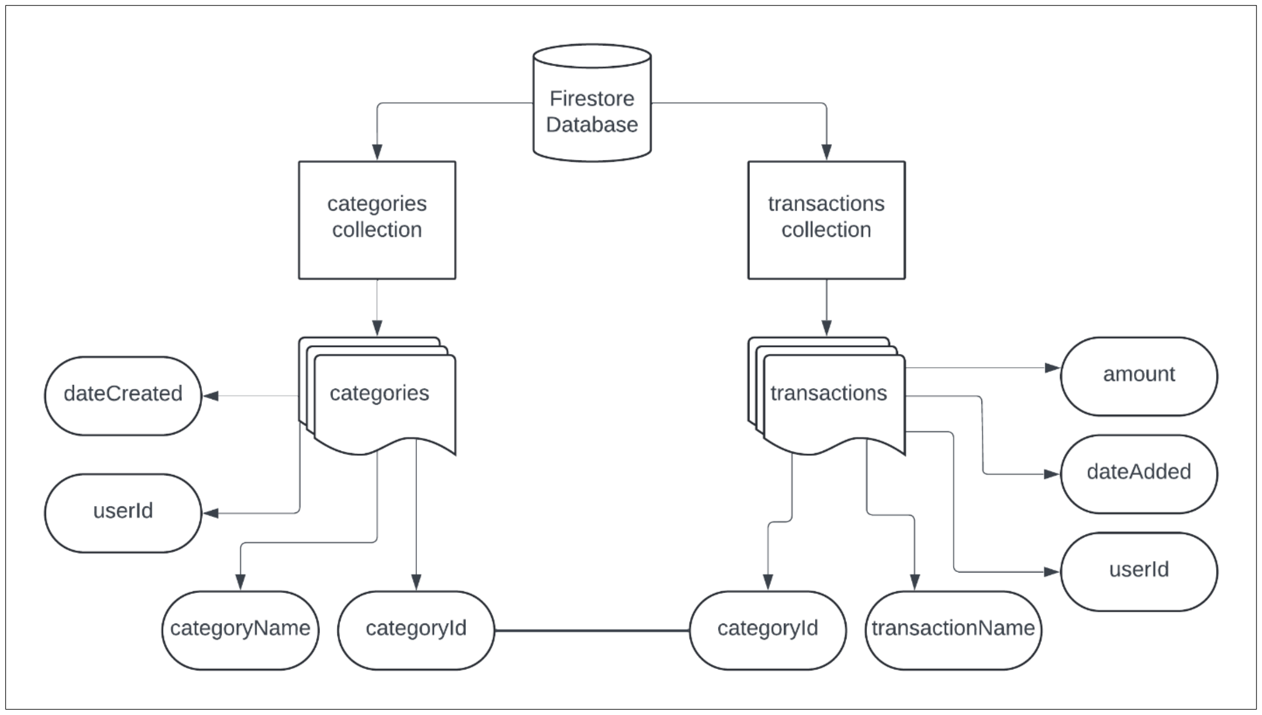
Criterion B: Design Overview

All user interface designs were made using the graphic design platform Canva *(Canva)*.

Application Hierarchy

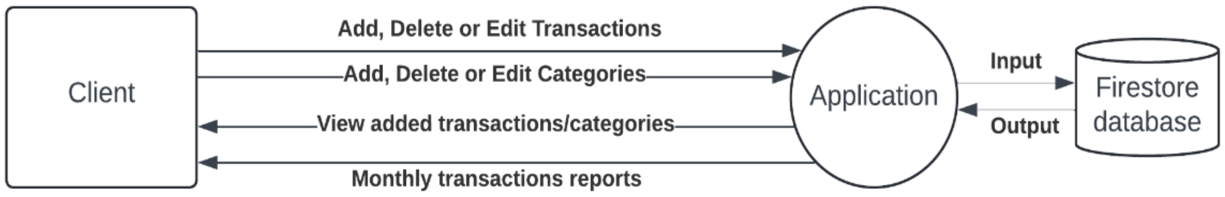

Default page from which the client can sign into the application
Alert pop-up from which the client can sign out of the application
SignIn
Home
Sign Out Modal
Budget
Reports
Settings

Firestore Database Model

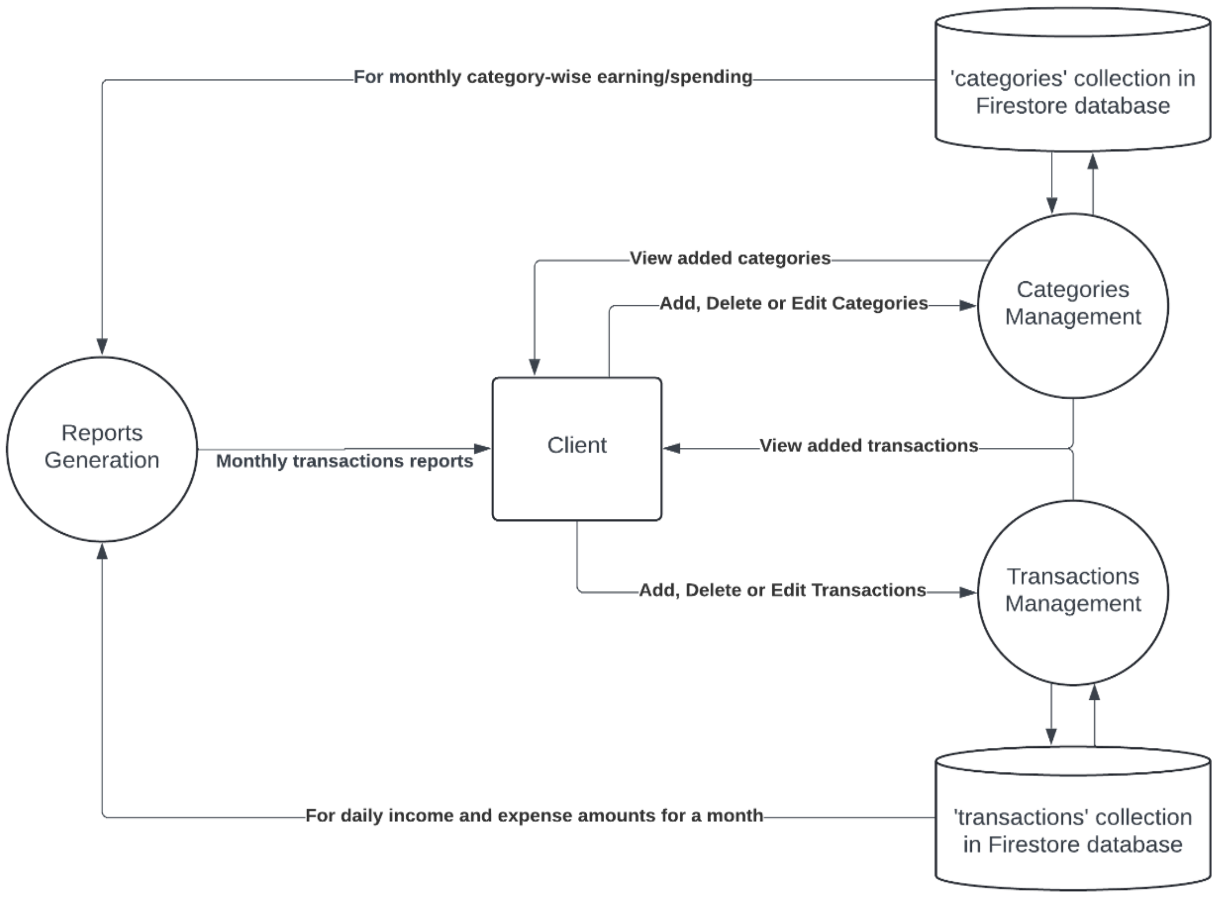


Application Data Flow Diagrams (DFDs)

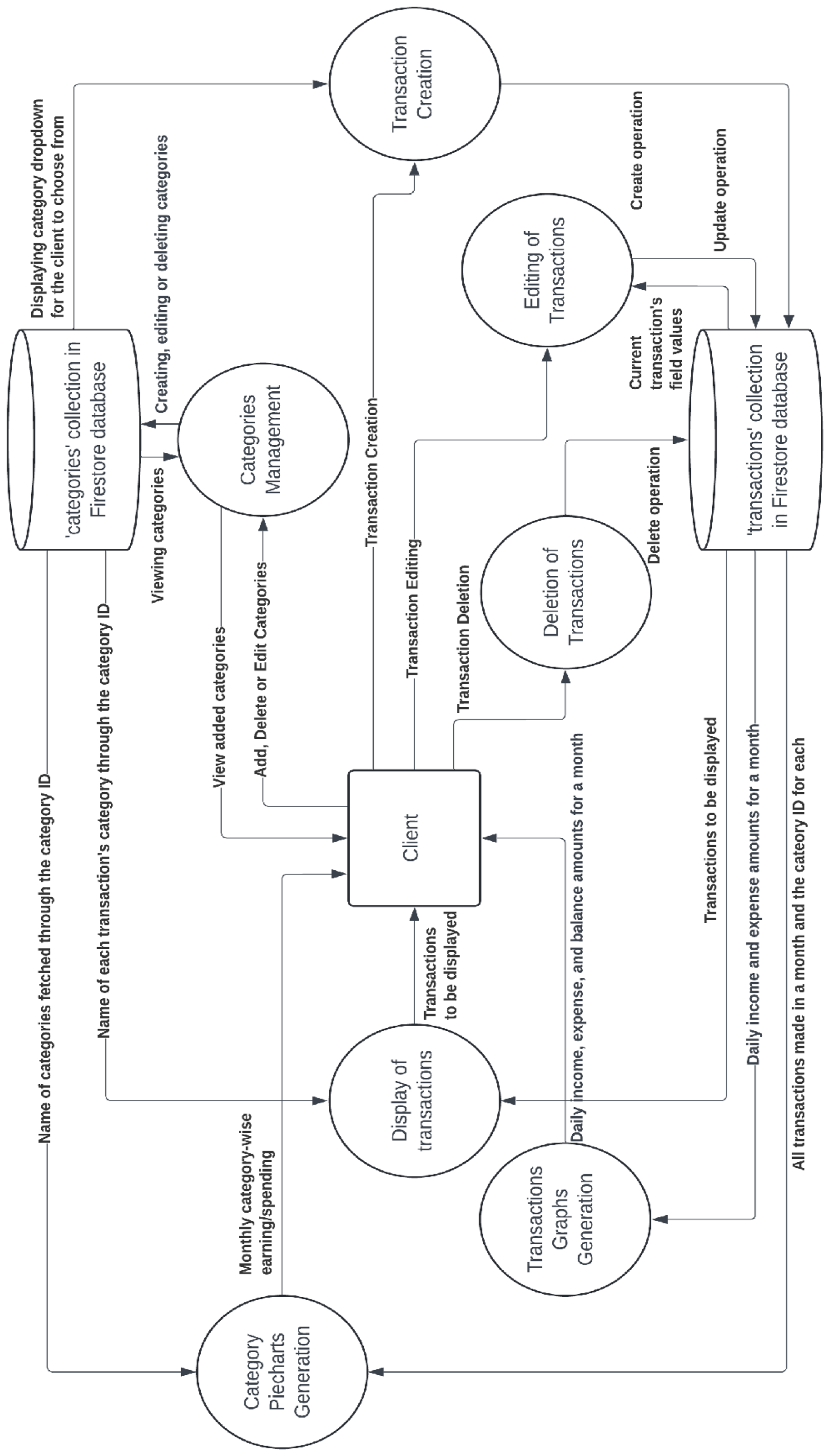
Level 0


Client
Add, Delete or Edit Transactions
Add, Delete or Edit Categories
View added transactions/categories
Monthly transactions reports
Application
Input
Output
Firestore database

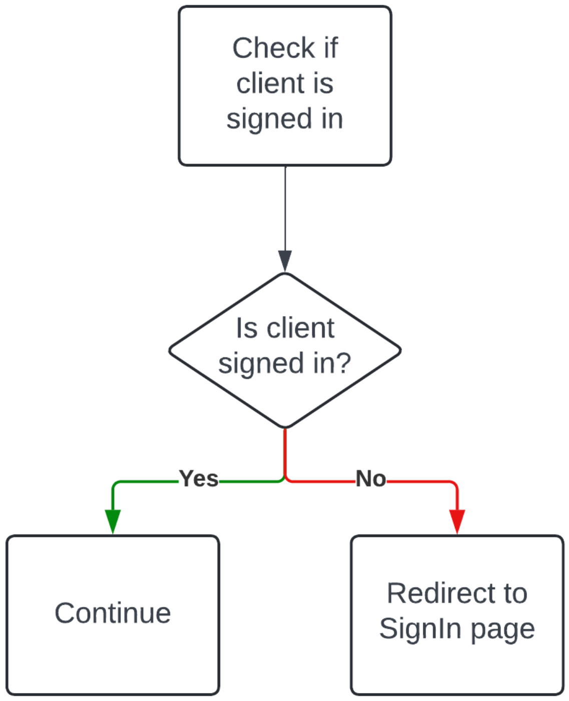
Level 1


For monthly category-wise earning/spending
'categories' collection in Firestore database
View added categories
Add, Delete or Edit Categories
Categories Management
Reports Generation
Monthly transactions reports
Client
View added transactions
Add, Delete or Edit Transactions
Transactions Management
For daily income and expense amounts for a month
'transactions' collection in Firestore database

Level 2


'categories' collection in Firestore database
Categories Management
Transaction Creation
Editing of Transactions
'transactions' collection in Firestore database
Deletion of Transactions
Client
Display of transactions
Transactions Graphs Generation
Category Piecharts Generation
Displaying category dropdown for the client to choose from
Creating, editing or deleting categories
Viewing categories
View added categories
Add, Delete or Edit Categories
Transaction Creation
Transaction Editing
Transaction Deletion
Create operation
Update operation
Current transaction's field values
Delete operation
Name of categories fetched through the category ID
Name of each transaction's category through the category ID
Monthly category-wise earning/spending
Transactions to be displayed
Daily income, expense, and balance amounts for a month
Transactions to be displayed
Daily income and expense amounts for a month
All transactions made in a month and the cateory ID for each

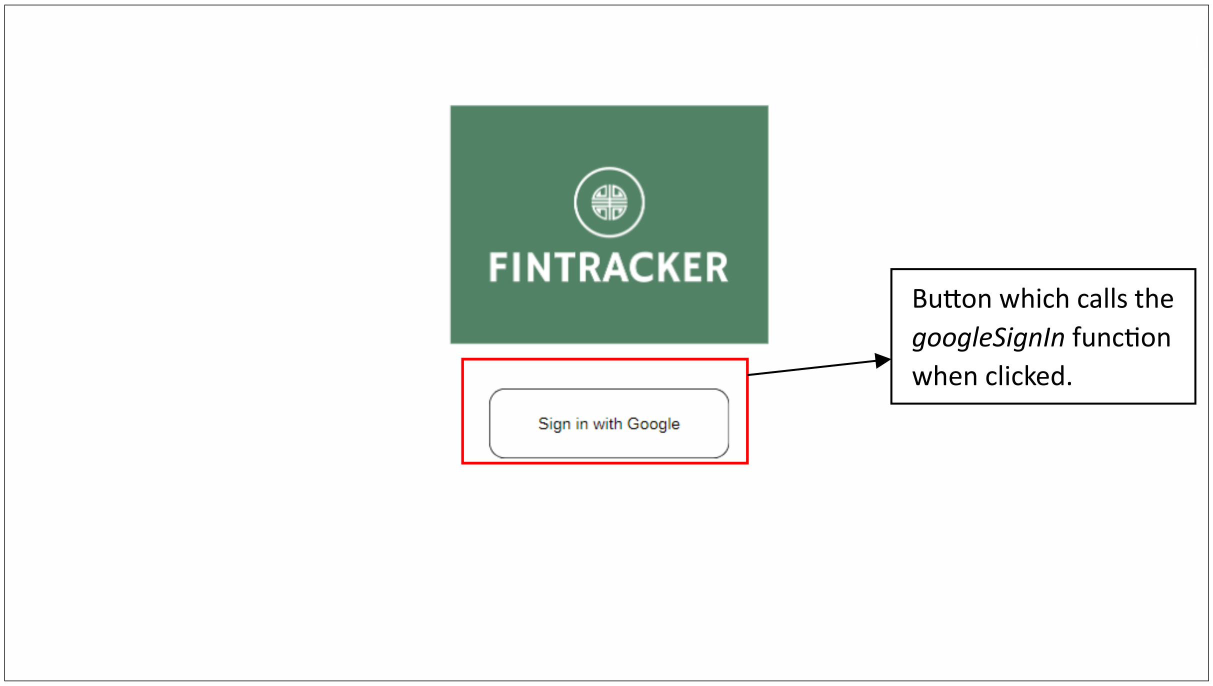
Authentication Check Flowchart and Pseudocode for All Pages

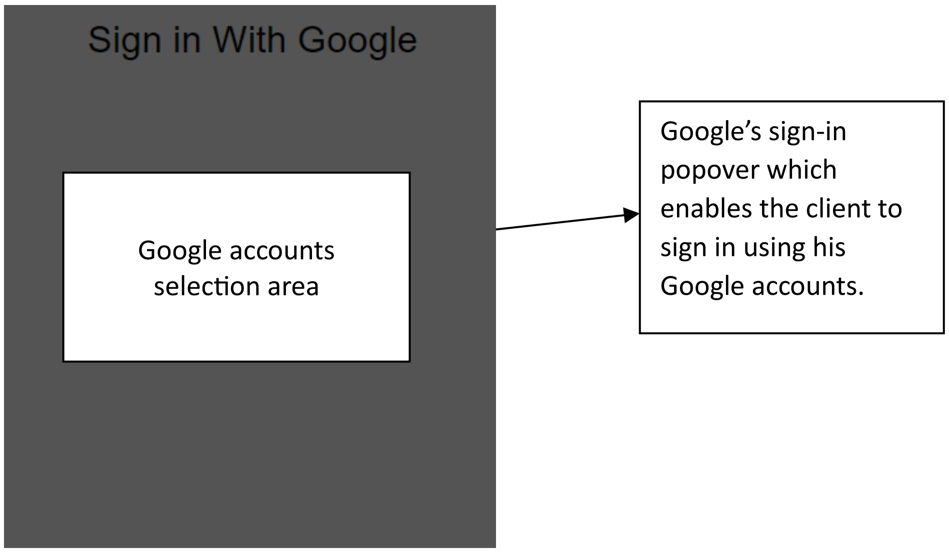


```
signedInStatus = checkSignInStatus()  // Check if the client is signed in


if signedInStatus = false then  // If true, then the page loads as normal
   redirectToSignInPage()  // Redirects client to SignIn page if he is not signed in with a Gmail account
end if
```

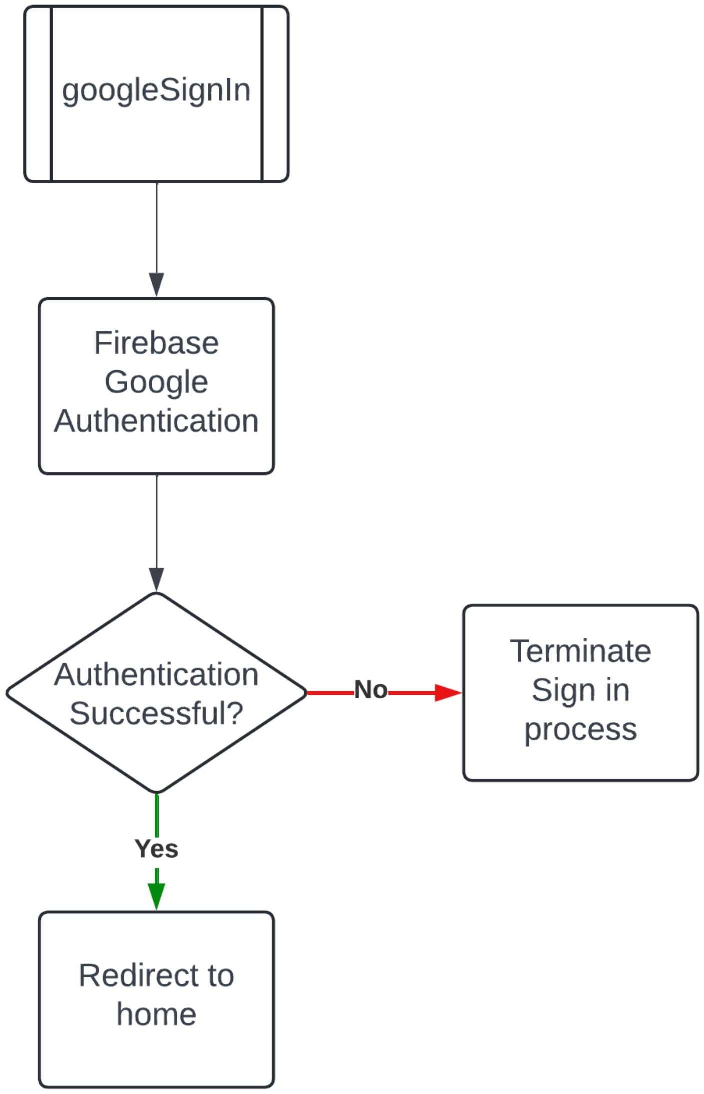
SignIn Page – Design, Structure, and Flowcharts

Page Design and Sign in with Google screen popover


FINTRACKER
Sign in with Google
Button which calls the *googleSignIn* function when clicked.

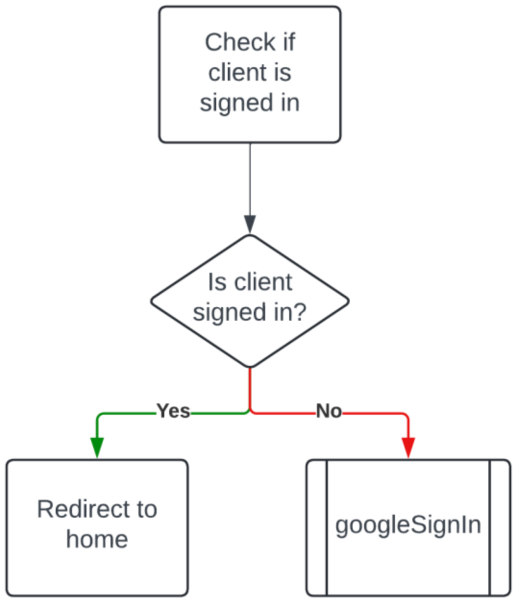

Sign in With Google
Google accounts selection area
Google's sign-in popover which enables the client to sign in using his Google accounts.

googleSignIn Flowchart and Pseudocode


googleSignIn
Firebase Google Authentication
Authentication Successful?
No
Terminate Sign in process
Yes
Redirect to home

```
subprogram googleSignIn (
  authenticationStatus = firebaseGoogleAuth()  // Perform Firebase Google Authentication


  if authenticationStatus = true then  // Check if authentication was successful
    redirectToHomePage()  // Redirect to the Home page if client signs in successfully
  else
    cancel()  // Terminate the sign in process if a failure occurs
    alert("Sign in unsuccessful, try again.")  // Inform client sign in process was unsuccessful.
  end if
)
```

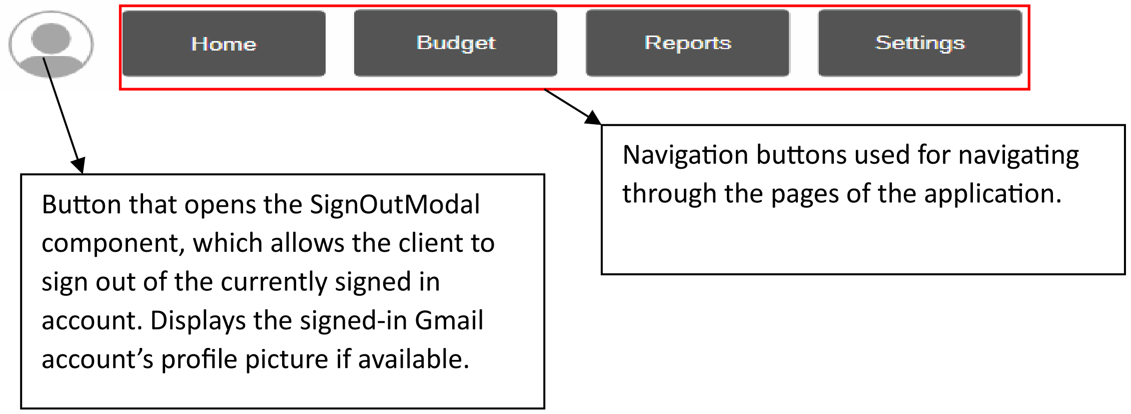
Authentication Check Flowchart and Pseudocode (for SignIn page only)



```
signedInStatus = checkSignInStatus()  // Check if the client is signed in

if signedInStatus = true then  // If true, then the client gets redirected to the Home page
  redirectToHomePage()  // Redirects client to SignIn page if he is not signed in with a Gmail account
else
  googleSignIn()
end if
```

Header Component

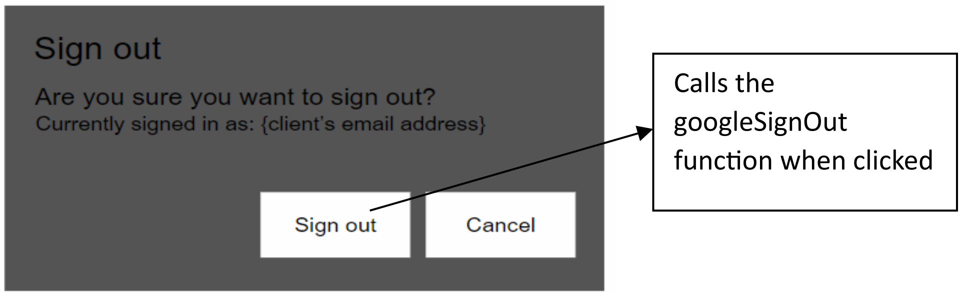
Design



Button that opens the SignOutModal component, which allows the client to sign out of the currently signed in account. Displays the signed-in Gmail account's profile picture if available.

Navigation buttons used for navigating through the pages of the application.

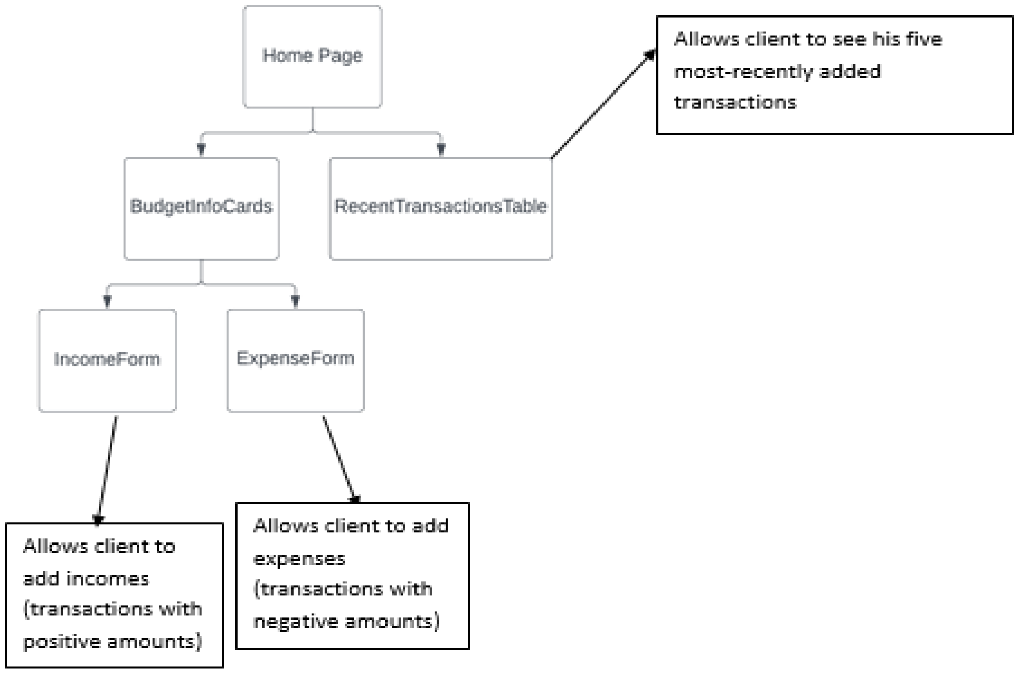
Header – Sign Out Modal


Sign out
Are you sure you want to sign out?
Currently signed in as: {client's email address}
Sign out
Cancel
Calls the googleSignOut function when clicked

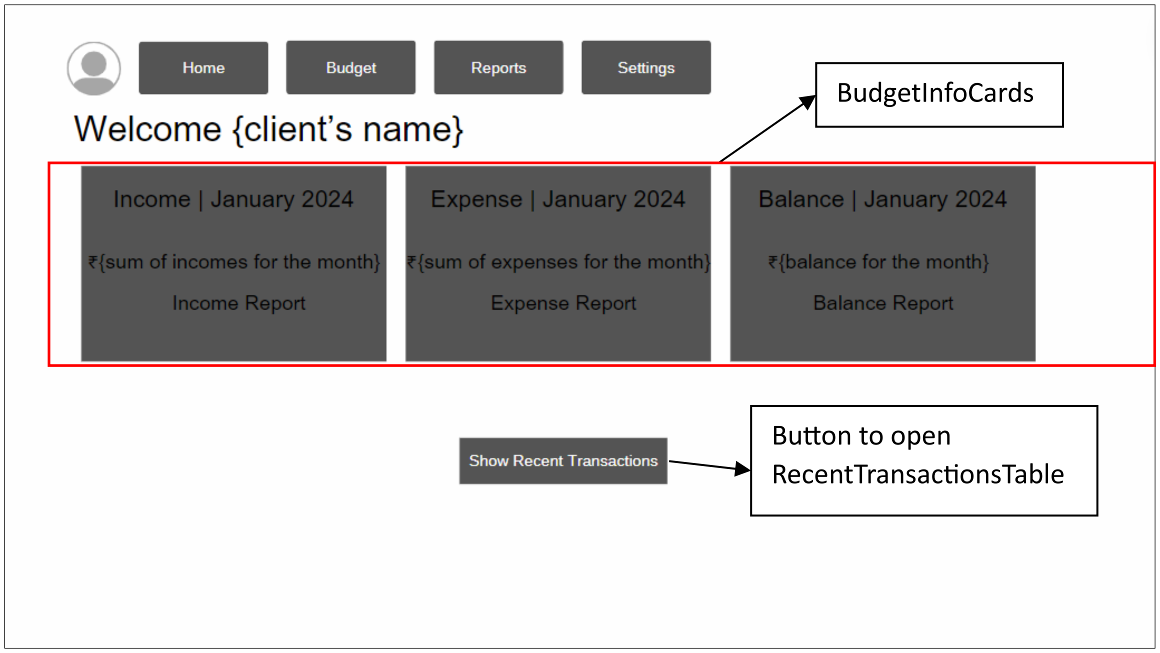
```
subprogram googleSignOut (
  signOut = firebaseGoogleSignOut()  // Sign out function from Firebase

  if signOut = true then  // Check if signOut was successful
    redirectToSignInPage()  // Redirect to the Home page if client signs in successfully
  else
    cancel()  // Terminate the sign out process if a failure occurs
    alert("Sign out unsuccessful, try again.")  // Inform client sign out process was unsuccessful.
  end if
)
```

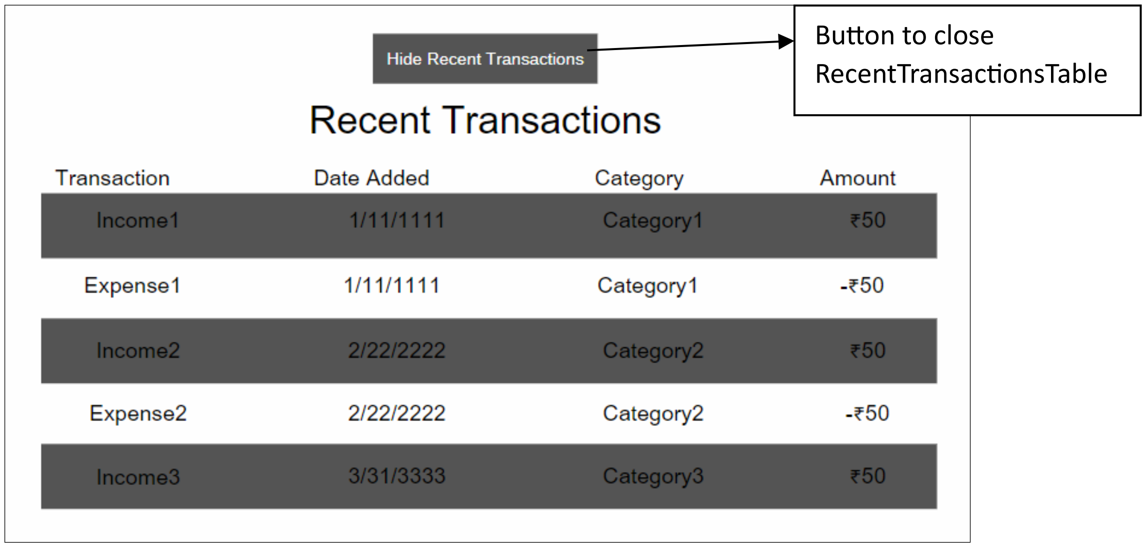
Home Page – Design, Structure, and Flowcharts

Structure


Home Page
BudgetInfoCards
RecentTransactionsTable
IncomeForm
ExpenseForm
Allows client to see his five most-recently added transactions
Allows client to add incomes (transactions with positive amounts)
Allows client to add expenses (transactions with negative amounts)

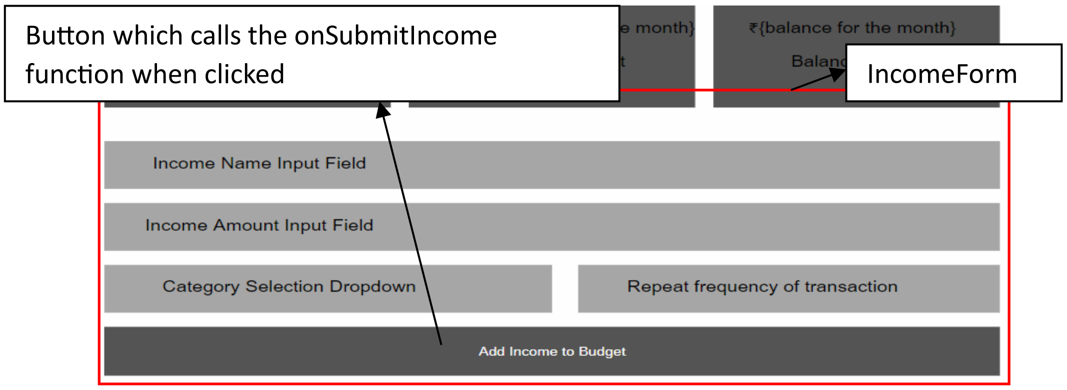
Page Design and RecentTransactionsTable when Opened



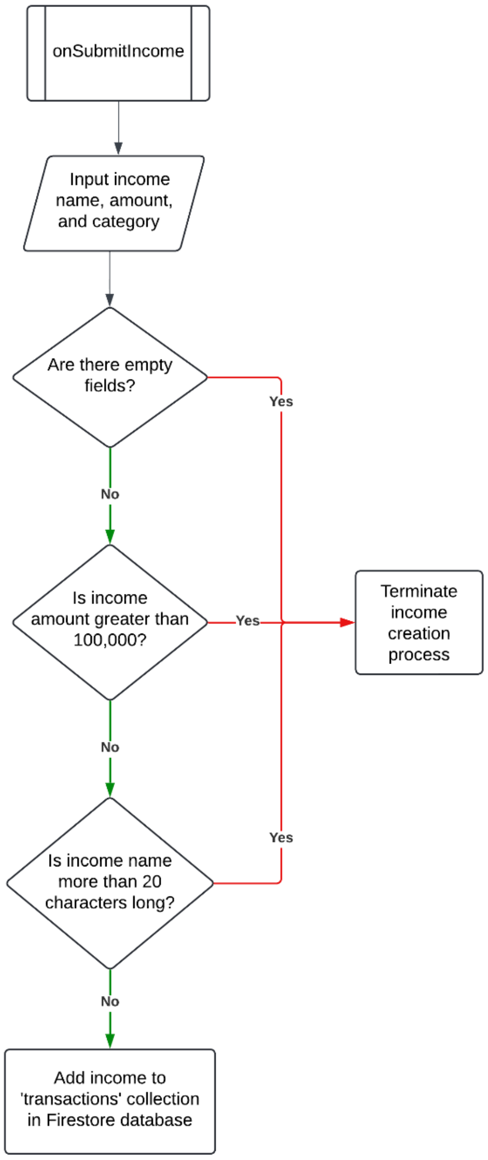


Transaction	Date Added	Category	Amount
Income1	1/11/1111	Category1	₹50
Expense1	1/11/1111	Category1	-₹50
Income2	2/22/2222	Category2	₹50
Expense2	2/22/2222	Category2	-₹50
Income3	3/31/3333	Category3	₹50

IncomeForm

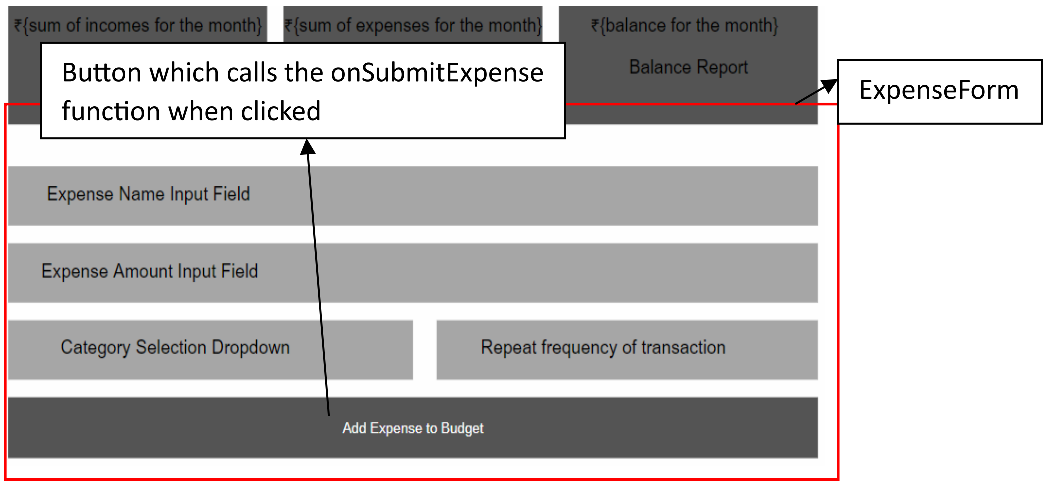

Button which calls the onSubmitIncome function when clicked
IncomeForm
e month}
t
₹{balance for the month}
Balanc
Income Name Input Field
Income Amount Input Field
Category Selection Dropdown
Repeat frequency of transaction
Add Income to Budget

onSubmitIncome Flowchart and Pseudocode

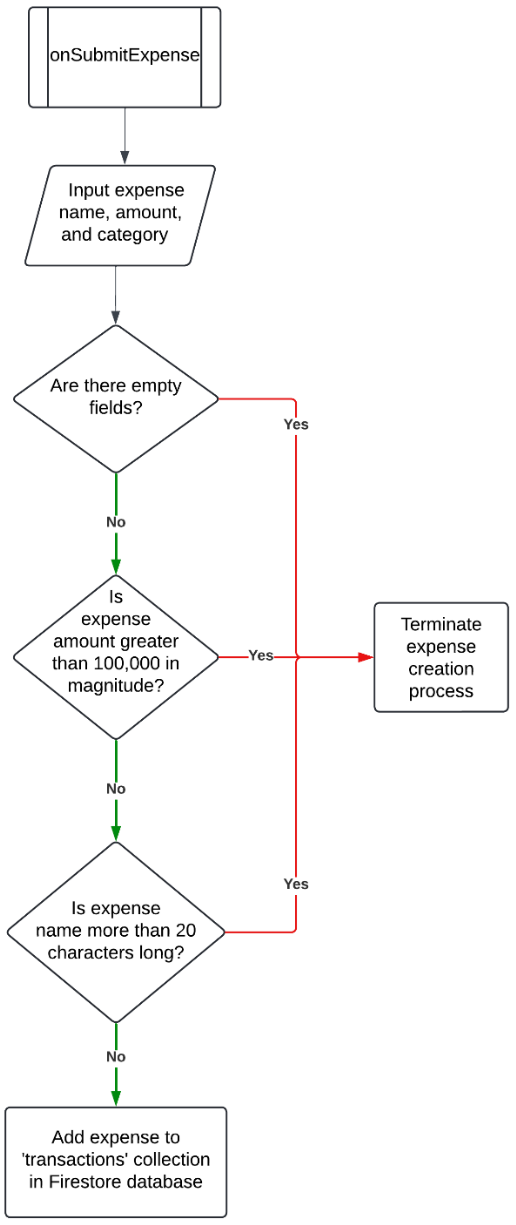

onSubmitIncome
Input income name, amount, and category
Are there empty fields?
Yes
No
Is income amount greater than 100,000?
Yes
No
Terminate income creation process
Is income name more than 20 characters long?
Yes
No
Add income to 'transactions' collection in Firestore database

```
subprogram onSubmitIncome(
  incomeDocument = input(incomeName, incomeAmount, transactionCategory)

  if anyFieldIsEmpty(incomeDocument) = true then  // Check if any of the input fields are empty
    cancel()  // Terminate income creation process
    alert("Please fill out all the fields.")  // Inform client that all the fields need to be filled out.

  else if incomeAmount > 100,000 then  // Check if the income amount is greater than 100,000
    cancel()
    alert("Income amount should be less than or equal to 100,000")
    // Inform client that incomeAmount should  be less than or equal to 100,000.

  else if incomeName.length  > 20 then
  // Check if incomeName has more than 20 characters.
    cancel()
    alert("Income name should be less than 20 characters long")
    // Inform client that incomeName should be less than 20 characters long.

  else
    addToFirestore(incomeDocument)
    // Add income to the 'transactions' collection in the Firestore database.
  end if
)
```

ExpenseForm


₹{sum of incomes for the month}
₹{sum of expenses for the month}
₹{balance for the month}
Balance Report
Button which calls the onSubmitExpense function when clicked
ExpenseForm
Expense Name Input Field
Expense Amount Input Field
Category Selection Dropdown
Repeat frequency of transaction
Add Expense to Budget

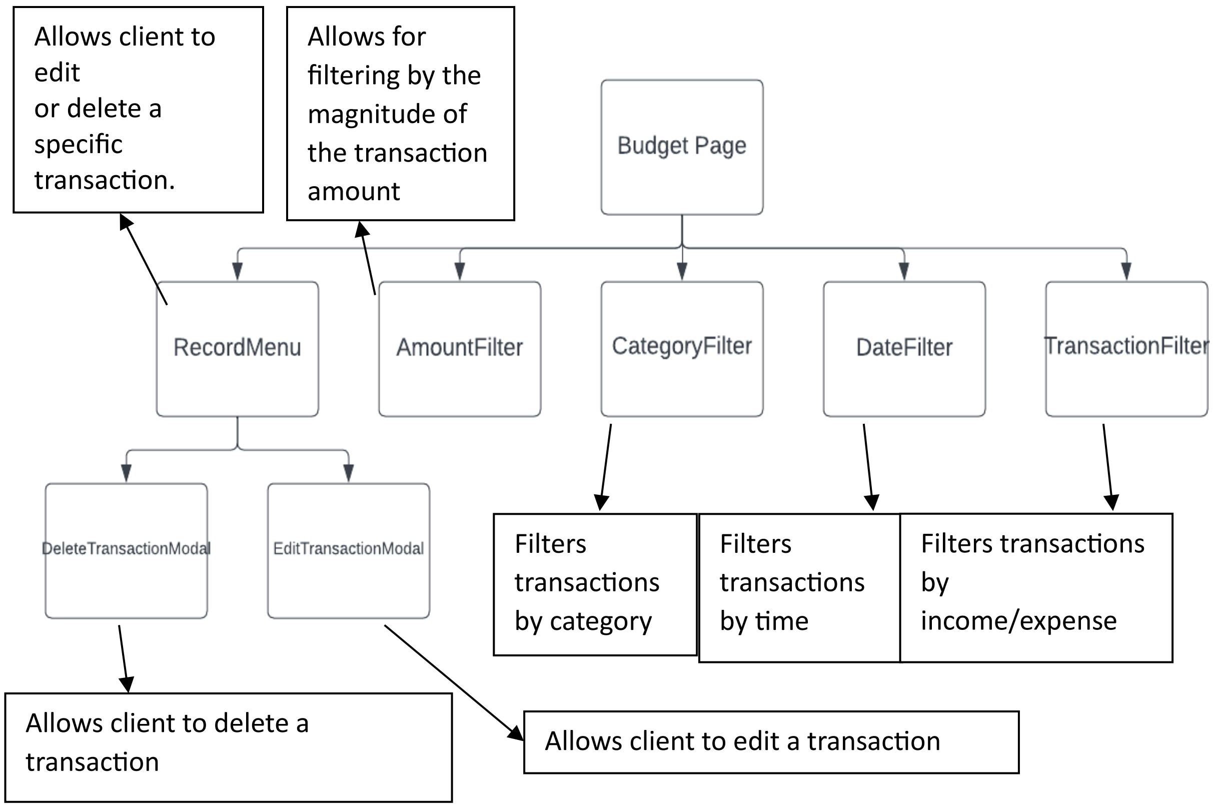
onSubmitExpense Flowchart and Pseudocode


onSubmitExpense
Input expense name, amount, and category
Are there empty fields?
Yes
No
Is expense amount greater than 100,000 in magnitude?
Yes
Terminate expense creation process
No
Is expense name more than 20 characters long?
Yes
No
Add expense to 'transactions' collection in Firestore database

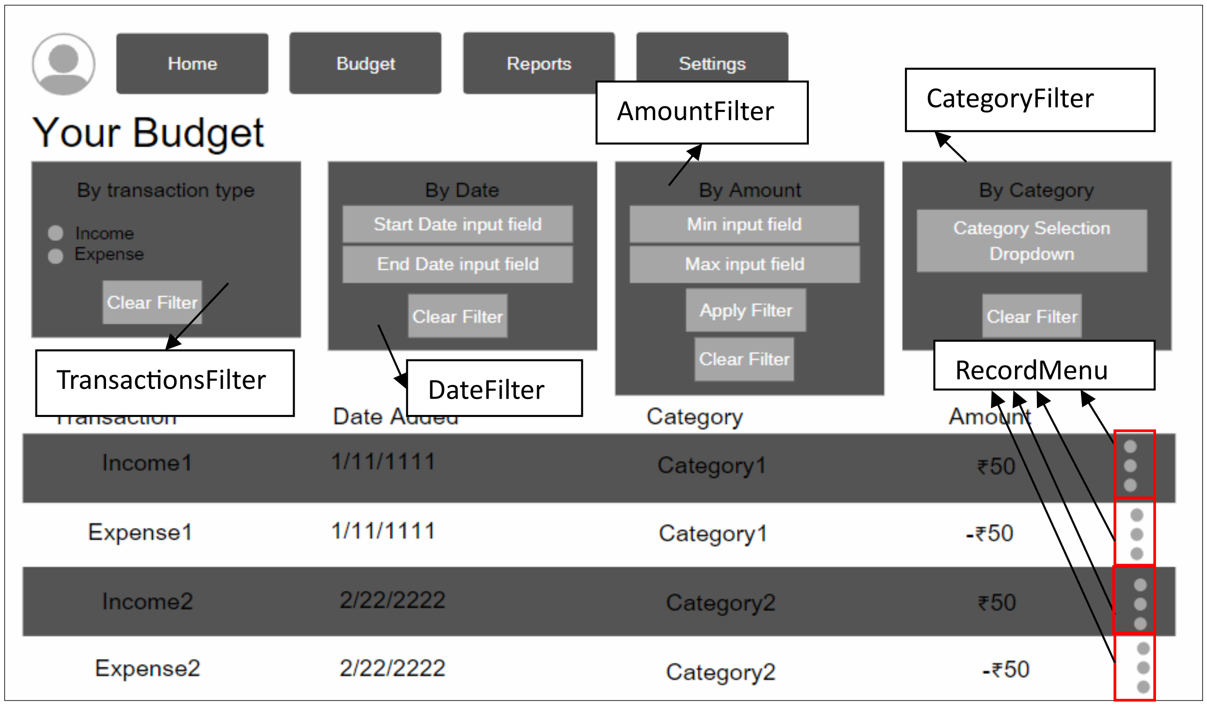
```
subprogram onSubmitExpense(
  expenseDocument = input(expenseName, expenseAmount, transactionCategory)


  if anyFieldIsEmpty(expenseDocument) = true then  // Check if any of the input fields are empty
    cancel()  // Terminate expense creation process
    alert("Please fill out all the fields.")  // Inform client that all the fields need to be filled out.
  else if expenseAmount > 100,000 then  // Check if the expense amount is greater than 100,000
    cancel()
    alert("Expense amount should be less than or equal to 100,000")
    // Inform client that expenseAmount should  be less than or equal to 100,000.


  else if expenseName.length  > 20 then
  // Check if expenseName has more than 20 characters.
    cancel()
    alert("Expense name should less than 20 characters long")
    // Inform client that expenseName should be less than 20characters long.


  else
    addToFirestore(expenseDocument)
    // Add expense to the 'transactions' collection in the Firestore database.
  end if
)
```

Budget Page – Design, Structure, and Flowcharts

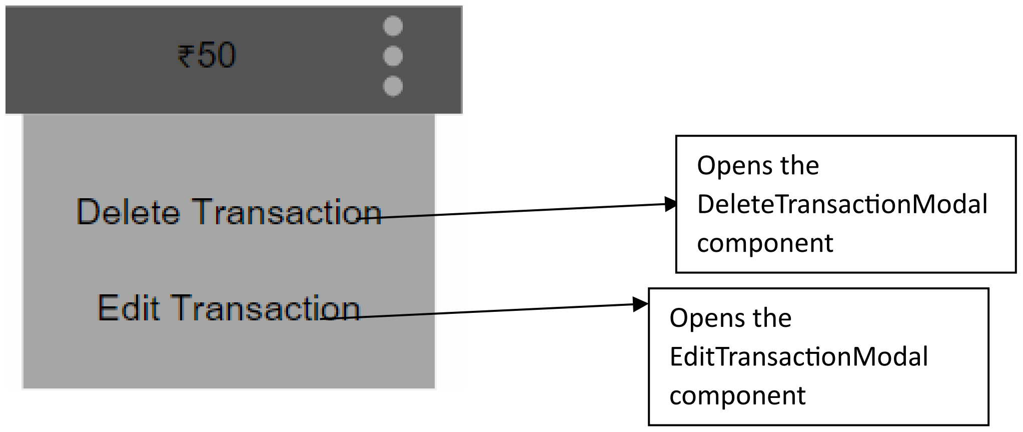
Structure


Allows client to edit or delete a specific transaction.
Allows for filtering by the magnitude of the transaction amount
Budget Page
RecordMenu
AmountFilter
CategoryFilter
DateFilter
TransactionFilter
DeleteTransactionModal
EditTransactionModal
Filters transactions by category
Filters transactions by time
Filters transactions by income/expense
Allows client to delete a transaction
Allows client to edit a transaction

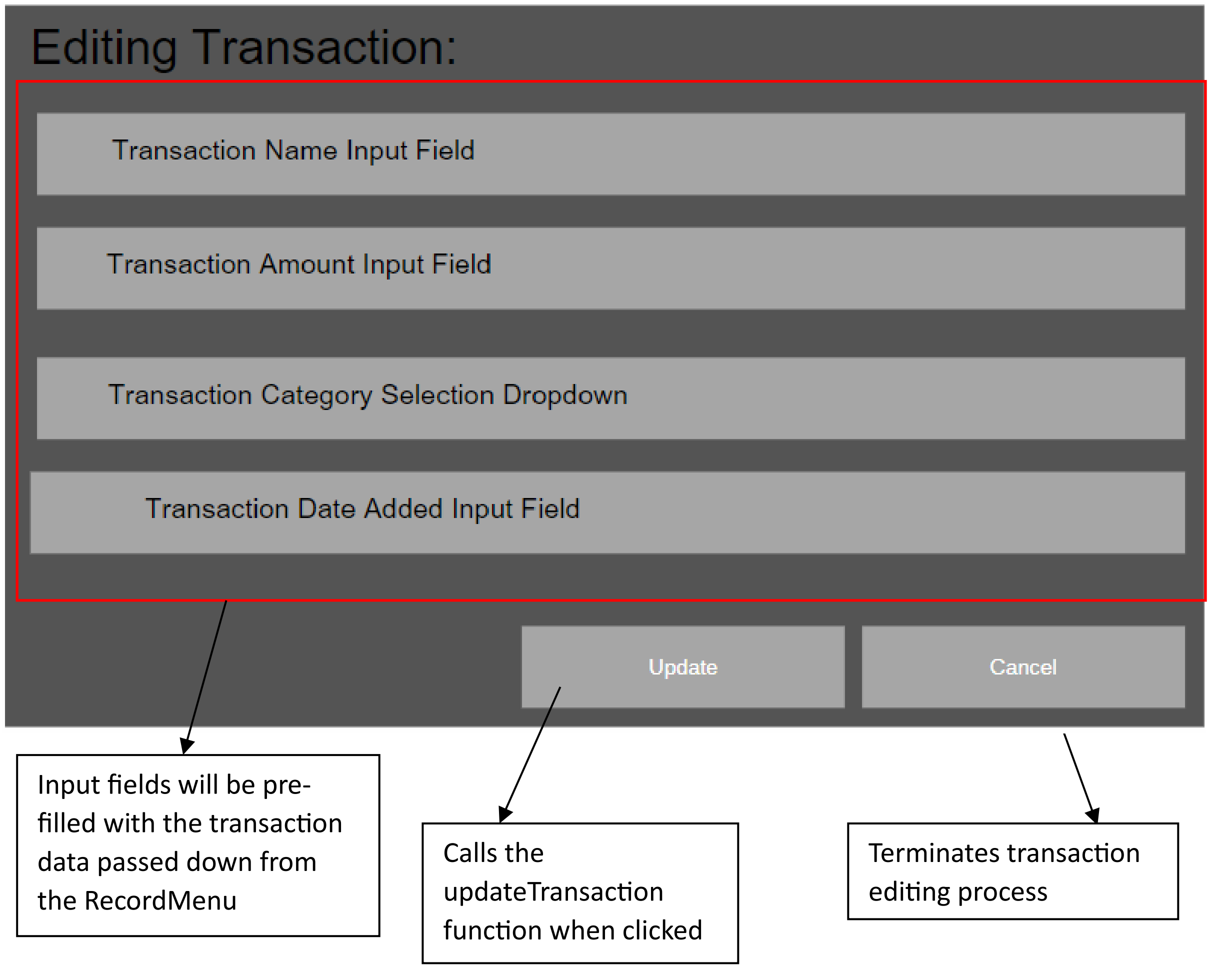
Page Design



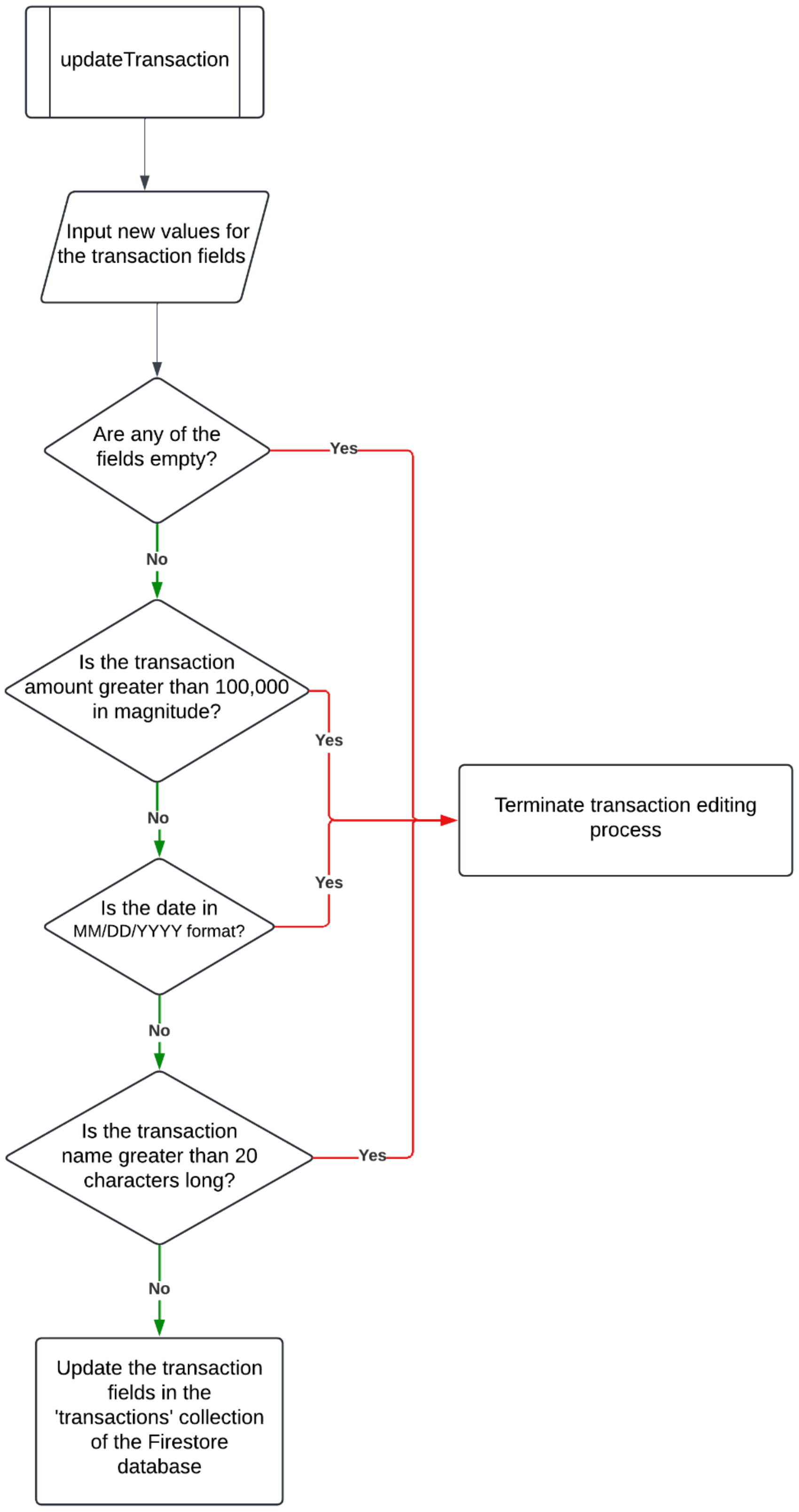
RecordMenu when Opened


₹50
Delete Transaction
Edit Transaction
Opens the DeleteTransactionModal component
Opens the EditTransactionModal component

EditTransactionsModal

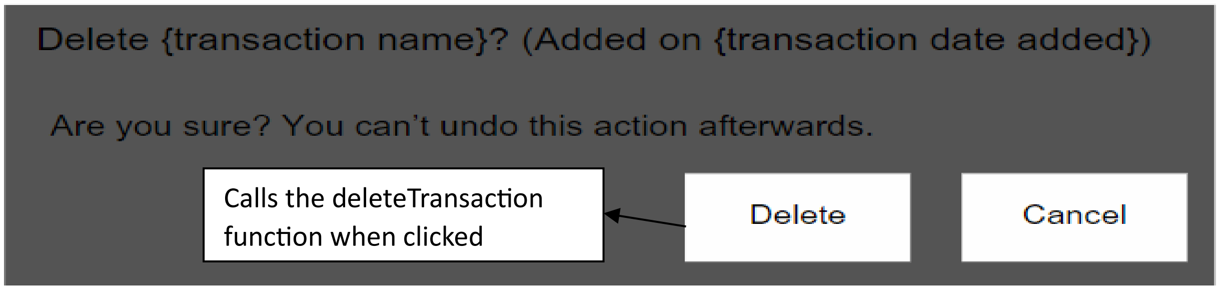

Editing Transaction:
Transaction Name Input Field
Transaction Amount Input Field
Transaction Category Selection Dropdown
Transaction Date Added Input Field
Update
Cancel
Input fields will be pre-filled with the transaction data passed down from the RecordMenu
Calls the updateTransaction function when clicked
Terminates transaction editing process

updateTransaction Flowchart and Pseudocode


updateTransaction
Input new values for the transaction fields
Are any of the fields empty?
Yes
No
Is the transaction amount greater than 100,000 in magnitude?
Yes
No
Terminate transaction editing process
Yes
Is the date in MM/DD/YYYY format?
No
Is the transaction name greater than 20 characters long?
Yes
No
Update the transaction fields in the 'transactions' collection of the Firestore database

```
subprogram updateTransaction(
  transactionDocument = input(transactionName, transactionAmount, transactionCategory, transaction Date)

  if anyFieldIsEmpty(transactionDocument) = true then  // Check if any of the input fields are empty
    cancel()  // Terminate transaction update process
    alert("Please fill out all the fields.")  // Inform client that all the fields need to be filled out.

  else if transactionAmount > 100,000 then
  // Check if the transaction amount is greater than 100,000.
    cancel()
    alert("Transaction amount should be less than 100,000")
    // Inform client that transactionAmount should be less than 100,000.

  else if transactionDate ≠ "MM/DD/YYYY" then // Check if date is in MM/DD/YYYY format
    cancel()
    alert("Date must be in MM/DD/YYYY format.")
    // Inform client that transactionDate must be in MM/DD/YYYY format.

  else if transactionName.length > 20 then // Check if name is more than 20 characters long
    cancel()
    alert("Transaction name should be less than 20 characters long.")
    // Inform client that transactionName must be less than 20 characters long.

  else
    updateInFirestore(transactionDocument)
    // Update the transaction document in the 'transactions' collection in the Firestore database.
  end if
)
```

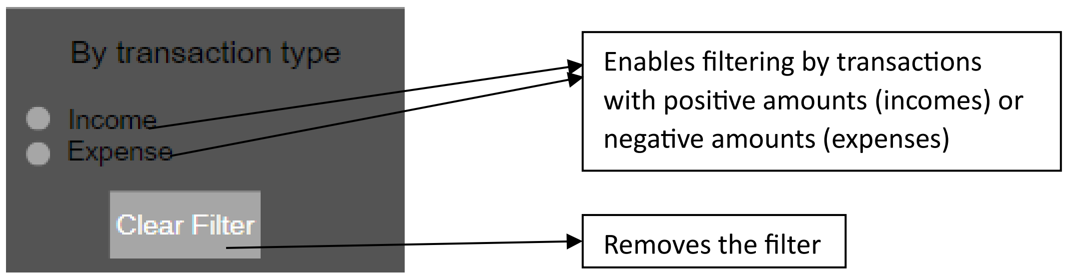
DeleteTransactionsModal


Delete {transaction name}? (Added on {transaction date added})

Are you sure? You can't undo this action afterwards.

Calls the deleteTransaction function when clicked

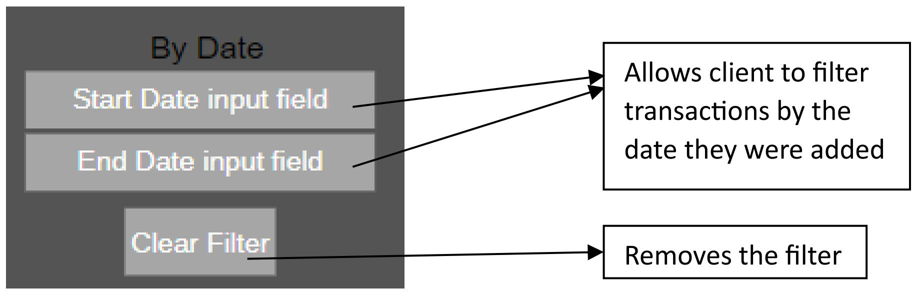
TransactionFilter



Enables filtering by transactions with positive amounts (incomes) or negative amounts (expenses)

Removes the filter

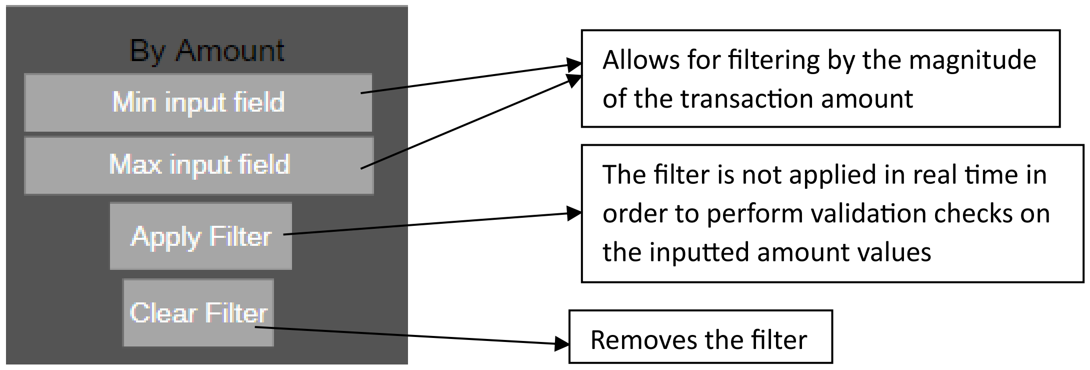
DateFilter



Allows client to filter transactions by the date they were added

Removes the filter

AmountFilter

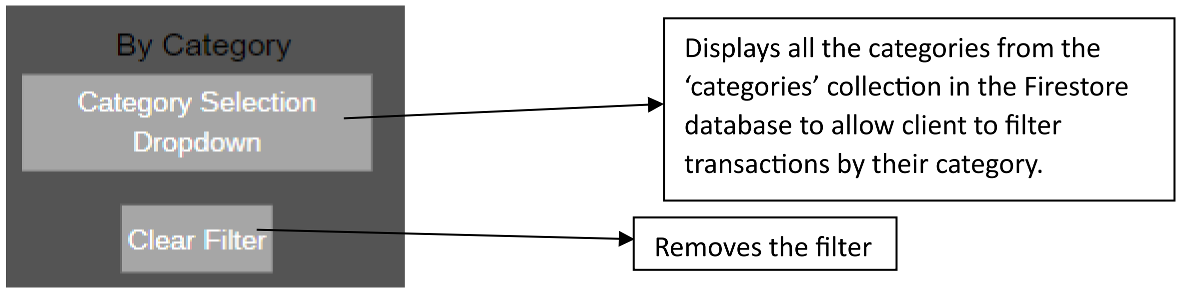


Allows for filtering by the magnitude of the transaction amount

The filter is not applied in real time in order to perform validation checks on the inputted amount values

Removes the filter

CategoryFilter



Displays all the categories from the 'categories' collection in the Firestore database to allow client to filter transactions by their category.

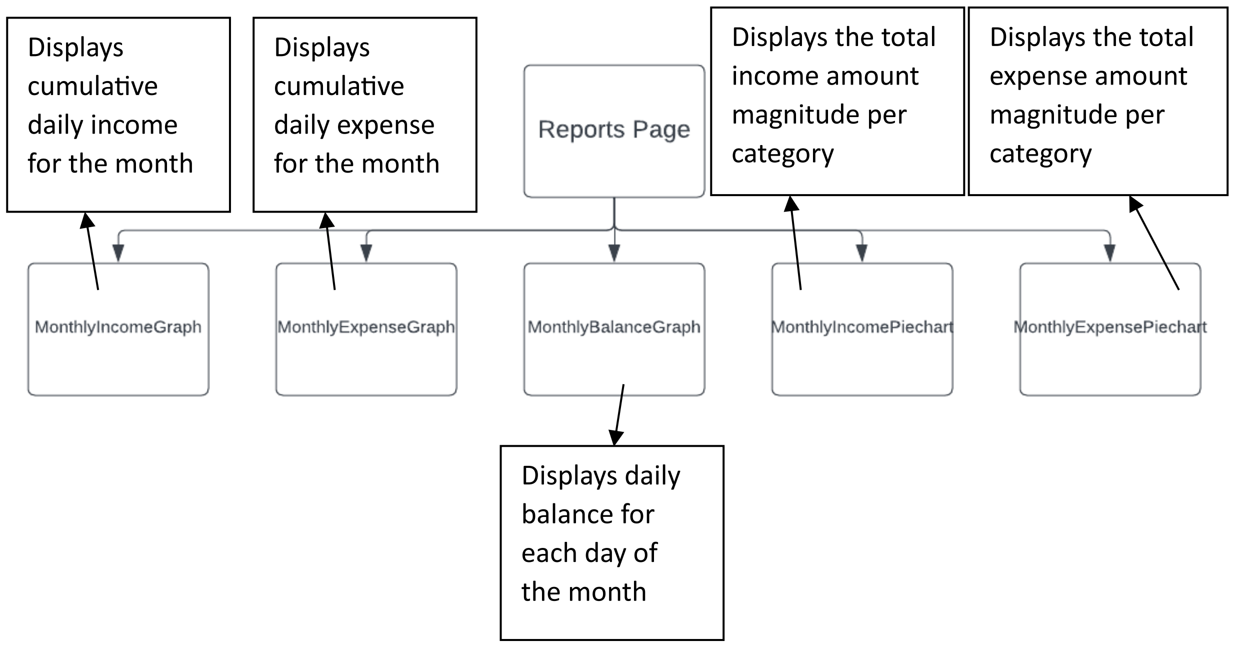
Removes the filter

Transactions Filtering Pseudocode

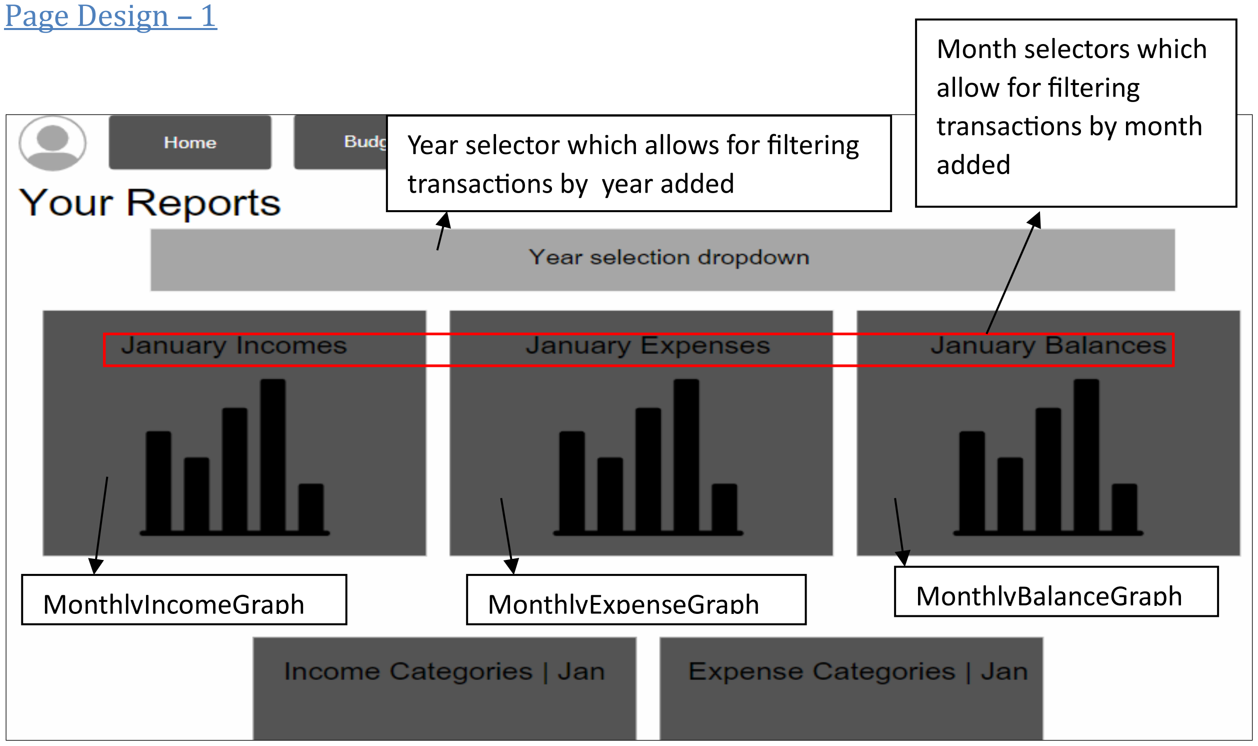
```
const query = transactionsQuery(clientAccountID, transactionType, startDate, endDate, minAmount,
maxAmount, category)


firestoreDatabaseQuery(query) // Applies the client's transaction filters on the Firestore database.


subprogram transactionsQuery(clientAccountID, transactionType, startDate, endDate, minAmount,
maxAmount, category) (
   TransactionFilter(transactionType)  // Apply the transaction type filter (income/expenses)
   dateFilters(startDate, endDate)  // Apply date filters
   amountFilters(minAmount, maxAmount)  // Apply amount filter
   categoryFilter(category)  // Apply category filter
)
```

Reports Page – Design, Structure, and Flowcharts

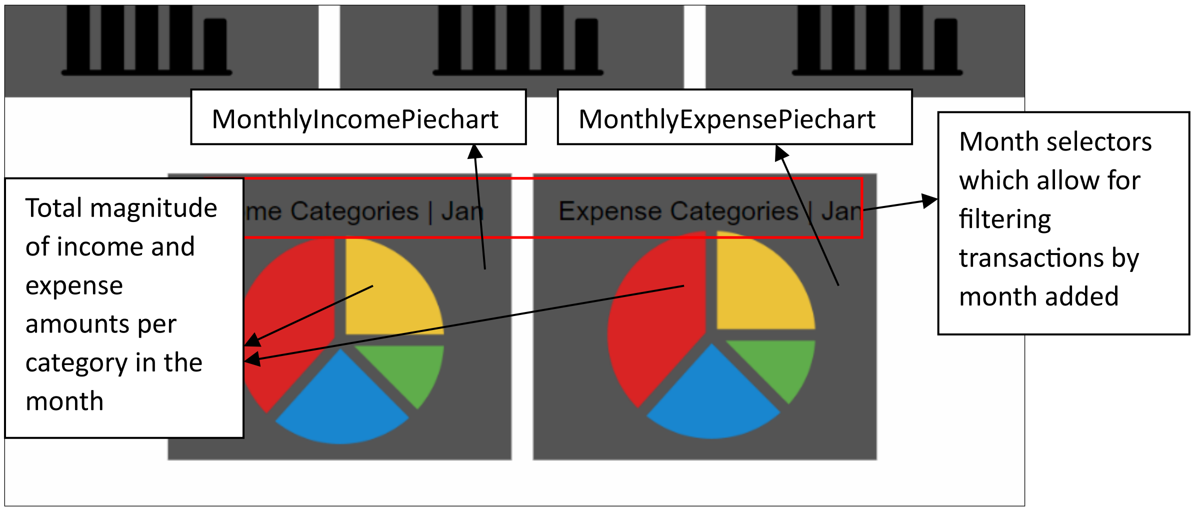
Structure


Displays cumulative daily income for the month
Displays cumulative daily expense for the month
Reports Page
Displays the total income amount magnitude per category
Displays the total expense amount magnitude per category
MonthlyIncomeGraph
MonthlyExpenseGraph
MonthlyBalanceGraph
MonthlyIncomePiechart
MonthlyExpensePiechart
Displays daily balance for each day of the month

Page Design – 1

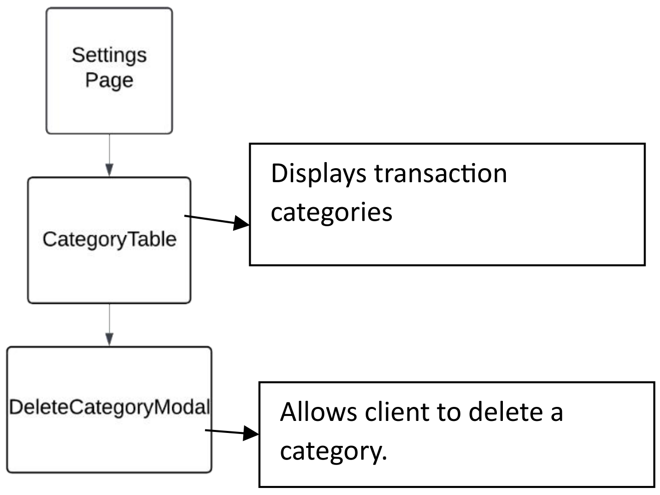


Page Design – 2

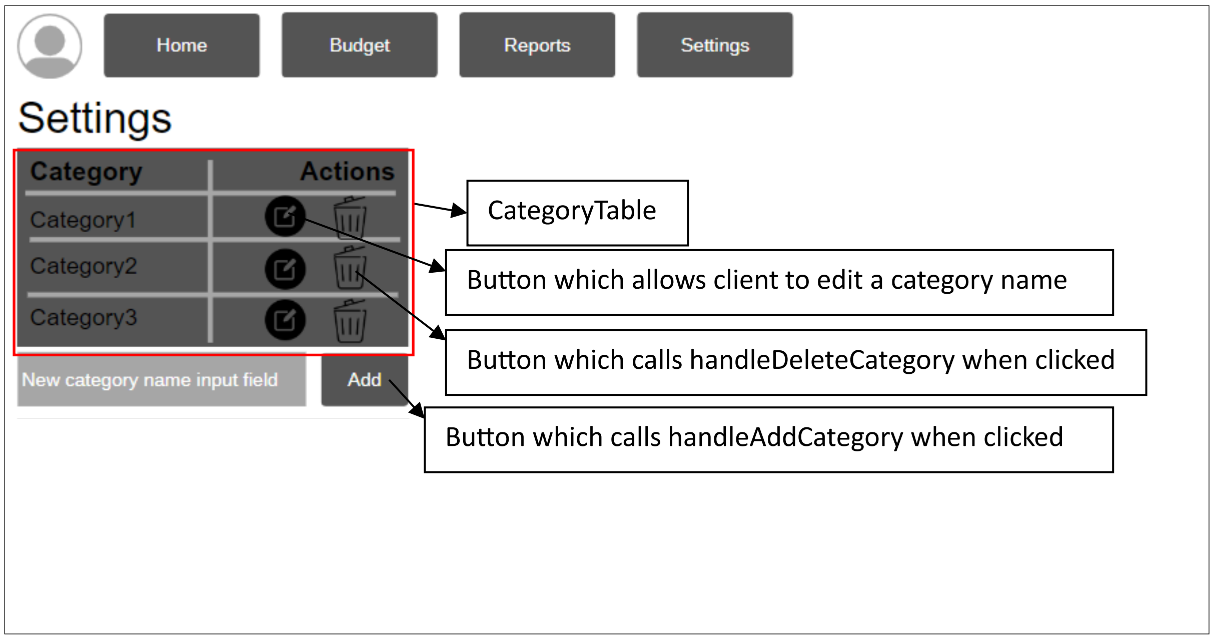

MonthlyIncomePiechart
MonthlyExpensePiechart
Month selectors which allow for filtering transactions by month added
Total magnitude of income and expense amounts per category in the month
me Categories | Jan
Expense Categories | Jan

Settings Page – Design, Structure, and Flowcharts

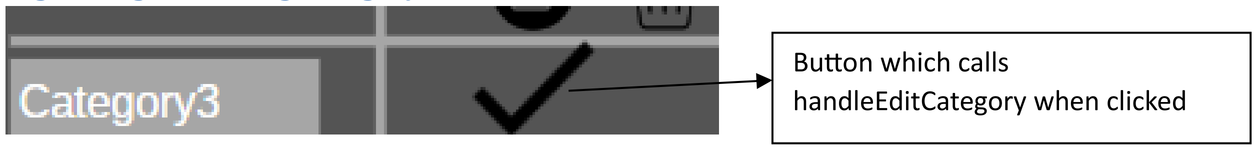
Structure


Settings Page
CategoryTable
DeleteCategoryModal
Displays transaction categories
Allows client to delete a category.

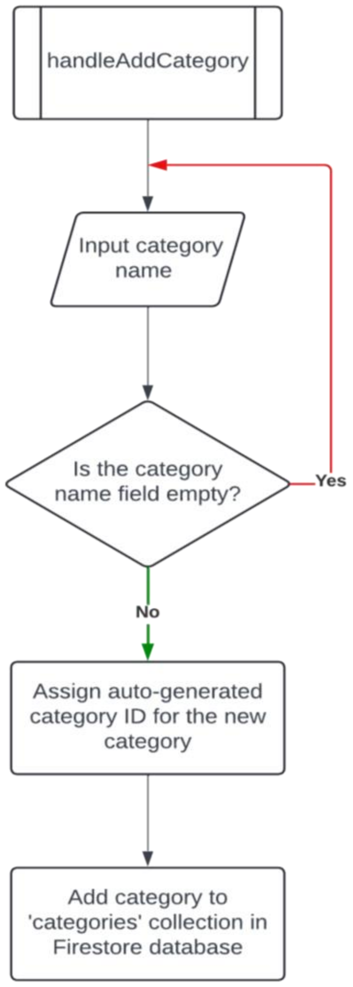
Page Design


Home
Budget
Reports
Settings
Settings
Category
Actions
Category1
Category2
Category3
New category name input field
Add
CategoryTable
Button which allows client to edit a category name
Button which calls handleDeleteCategory when clicked
Button which calls handleAddCategory when clicked

Page Design – Editing Category

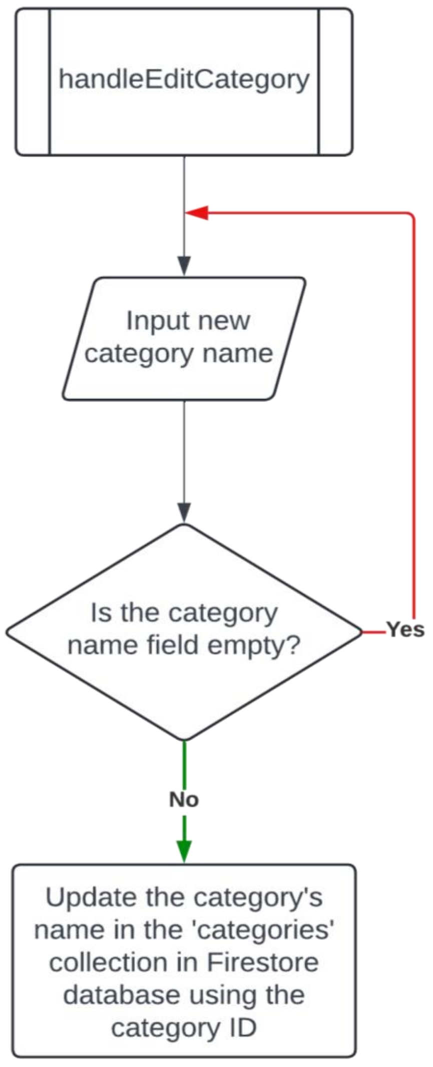


handleAddCategory Flowchart and Pseudocode


handleAddCategory
Input category name
Is the category name field empty?
Yes
No
Assign auto-generated category ID for the new category
Add category to 'categories' collection in Firestore database

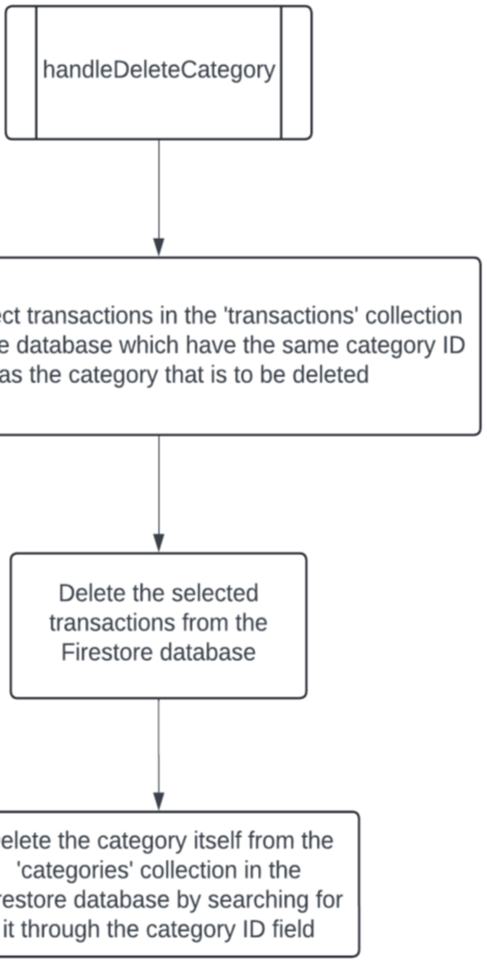
```
subprogram handleAddCategory(
  categoryName = input("Enter category name")  // Client inputs category name

  if categoryName = "" then  // Check if the category name field is empty
    cancel()  // Terminate category creation process
    alert("Category name cannot be empty.")  // Inform client that the category name is required.

  else
    categoryID = generateCategoryID()  // Assign auto-generated category ID for the new category
    addToFirestore(categoryID, categoryName)
   // Add category to 'categories' collection in Firestore database.
  end if
)
```

handleEditCategory Flowchart and Pseudocode


handleEditCategory
Input new category name
Is the category name field empty?
Yes
No
Update the category's name in the 'categories' collection in Firestore database using the category ID

```
subprogram handleEditCategory(
  newCategoryName = input()  // Input new category name


  if newCategoryName = "" then  // Check if the category name field is empty
    cancel()  // Terminate category editing process
    alert("Category name cannot be empty.")  // Inform client that a category name is required.


  else
    updateCategoryInFirestore(categoryID, newCategoryName)
    // Update category name in the 'categories' collection in Firestore using the category ID.
  end if
)
```

handleDeleteCategory Flowchart and Pseudocode


handleDeleteCategory
Filter and select transactions in the 'transactions' collection of the Firestore database which have the same category ID field as the category that is to be deleted
Delete the selected transactions from the Firestore database
Delete the category itself from the 'categories' collection in the Firestore database by searching for it through the category ID field

```
subprogram handleDeleteCategory(
  transactionsToBeDeleted = filterTransactionsByCategory(categoryID)
  // Filter and select transactions in the 'transactions' collection of the Firestore database which have the same category ID field as the category that is to be deleted.

  deleteInFirestore(transactionsToBeDeleted)
  // Delete the selected transactions from the Firestore database.

  deleteCategoryInFirestore(categoryID)
  // Delete the category itself from the 'categories' collection in the Firestore database by searching for it through the category ID field.
)
```

Bibliography

Canva. *Canva Homepage*. n.d. 25 September 2023. <https://www.canva.com/>.

Test Plan

Success Criteria Tested	Action to be tested	Test method	Expected Result
Success Criterion 1: The client must be able to switch between his personal and work Gmail accounts without losing any of his financial data. Each account's data should be isolated, ensuring that only the transactions and categories related to the signed-in account are accessible.	Ability to sign in and out using different Gmail accounts	Signing into the application using the SignIn page using different Gmail accounts and then signing out	Clicking on the 'Sign in with Google' button opens up the Google sign-in popup. After signing in successfully, the profile picture of the signed-in Gmail account is shown in the left side of the Header component at the top of the application. Clicking on this opens the SignOutModal component which displays their current Gmail address. Clicking on the 'Sign Out' button in the aforementioned component signs the user out and redirects them to the SignIn page
	Effect of closing Gmail sign-in popup before singing in	Clicking on the "Sign in With Google" button and closing the popup that appear afterwards before signing in	Application does redirect client to the Home page. An error message is displayed to the client, which should prompt him to try again
	Effect of inputting an incorrect password and then closing the Gmail sign-in popup	Clicking on the "Sign in With Google" button and then selecting an account to sign in with. After inputting an incorrect password, the popup is then closed	Application does redirect client to the Home page. An error message is displayed to the client, which should prompt him to try again
	Testing if the transactions and transaction categories made in different Gmail accounts are visible in	Signing into the application using a new Gmail account (other than the test Gmail account which contains some transactions and	When the user is signed into a new Gmail account, they should not be able to see or access the transactions and categories that

	other Gmail accounts as well	transaction categories which are visible in the Firestore database)	were made in different Gmail accounts in any part of the application whatsoever. This should be achieved through filtering the Firestore data using the Firebase-provided userId value before showing the data to the user
Success Criterion 2: The application must ensure that sensitive financial data is only accessible when the client is logged in using one of their Gmail accounts. Attempts to access the application without logging in must redirect the user to the sign in page.	Accessing the application without signing into a Gmail account	Navigating to the different pages of the application by changing the URL after signing out	The application redirects the user to the SignIn page, thus preventing malicious actors from being able to access the financial data
	Trying to access transactions and categories created by other Gmail accounts	Signing into the application using a new Gmail account (other than the test Gmail account which contains some transactions and transaction categories which are visible in the Firestore database)	When the user is signed into a new Gmail account, they should not be able to see or access the transactions and categories that were made in different Gmail accounts in any part of the application whatsoever. This ensures that the client's kids cannot view his office finances and vice versa for his co-workers
Success Criterion 3: The client should be able to add new incomes and expenses on the Home page. The transactions must also be validated – transaction amounts cannot exceed 100,000 and transaction names cannot be longer than 20 characters long.	Addition of an income to the database with amount that falls under the 100,000 limit (normal value)	Creating a new income via the IncomeForm component and typing in 50,000 in the amount field	Able to see success/error message regarding the addition of the income. If addition was successful, the newly added income should be visible in the 'transactions' collection in the database as a Firestore document with the correct values in each field
	Addition of an expense to the database with	Creating a new expense via the ExpenseForm	Able to see success/error message

amount that falls under the 100,000 limit (normal value)	component and typing in 50,000 in the amount field	regarding the addition of the expense. If addition was successful, the newly added expense should be visible in the ‘transactions’ collection in the database as a Firestore document with the correct values in each field
Addition of an income to the database with amount 100,000 (extreme value)	Creating a new income via the IncomeForm component and typing in 100,000 in the amount field	Able to see success/error message regarding the addition of the income. If addition was successful, the newly added income should be visible in the ‘transactions’ collection in the database as a Firestore document with the correct values in each field
Addition of an expense to the database with amount 100,000 (extreme value)	Creating a new expense via the ExpenseForm component and typing in 100,000 in the amount field	Able to see success/error message regarding the addition of the expense. If addition was successful, the newly added expense should be visible in the ‘transactions’ collection in the database as a Firestore document with the correct values in each field
Addition of an income to the database with amount greater than 100,000 (abnormal value)	Creating a new income via the IncomeForm component and typing in 2,00,000 in the amount field	Able to see error message informing the user that the amount specified must be under 100,000
Addition of an expense to the database with amount greater than 100,000 (abnormal value)	Creating a new expense via the ExpenseForm component and typing in 2,00,000 in the amount field	Able to see error message informing the user that the amount specified must be under 100,000

	Adding a transaction name that is less than 20 characters in length (normal value)	Creating a new income and expense via the IncomeForm and ExpenseForm components and typing in a 15-character name for the transaction	Transaction name should be permitted and, if successfully added to the ‘transactions’ collection of the database, should be visible in the ‘transactionName’ field of the transaction’s Firestore document
	Adding a transaction name that is 20 characters in length (extreme value)	Creating a new income and expense via the IncomeForm and ExpenseForm components and typing in a 20-character name for the transaction	Transaction name should be permitted and, if successfully added to the ‘transactions’ collection of the database, should be visible in the ‘transactionName’ field of the transaction’s Firestore document
	Adding a transaction name that is longer than 20 characters in length (abnormal value)	Creating a new income and expense via the IncomeForm and ExpenseForm components and typing in a 25-character name for the transaction	Error message shown to client alerting that transaction names should be less than or equal to 20 characters long
	Effect of leaving a field’s value as blank when creating a new income (abnormal value)	Creating a new income but leaving the transaction name field blank. Clicking on the “Add Income” button. Steps to be repeated for each field in the IncomeForm component	Able to see warning message informing the user that all transaction fields must have a value
	Effect of leaving a field’s value as blank when creating a new expense (abnormal value)	Creating a new expense but leaving the transaction name field blank. Clicking on the “Add Expense” button. Steps to be repeated for each field in the ExpenseForm component	Able to see warning message informing the user that all transaction fields must have a value
	Effect of non-numerical characters in the	Inputting non-numerical characters in the	Non-numerical text not being added in the

	transaction amount field in the IncomeForm and ExpenseForm components (abnormal value)	transaction amount field in the IncomeForm and ExpenseForm components	transaction amount field
Success Criterion 4: The client must be able to view five of his recently made transactions in the Home page.	Effect of clicking on the ‘Show Transactions’ button in the Home page	Clicking on the ‘Show Transactions’ button at the bottom of the Home page	The RecentTransactionsTable component should come into view, the ‘Show Transactions’ button should now display ‘Hide Transactions’, and the user’s view window should automatically scroll downwards to keep the RecentTransactionsTable closer to the center of the screen
	Effect of clicking on the ‘Hide Transactions’ button in the Home page when the RecentTransactionsTable is visible	Clicking on the ‘Hide Transactions’ button at the top of the RecentTransactionsTable	RecentTransactionsTable goes out of view, the ‘Hide Transactions’ button should display ‘Show Transactions’, and the user’s view window should automatically scroll upwards to keep the rest of the components in the Home page at the center of the screen
	Testing the transaction limiting filter of the database query function in RecentTransactionsTable	Clicking on the ‘Show Transactions’ button at the bottom of the Home page when there are 4, 5, and 6 transactions in the database	The component must display all 4 transactions in the first case, then all 5 transactions in the second case, then only the 5 most recent transactions in the third case
	Effect of changing the dataAdded field in an old transaction so that it becomes the recent-most transaction and vice versa	Making 6 transactions in the database, opening the RecentTransactionsTable component, then changing the dateAdded fields of the oldest and	In the first opening of the component, the oldest transaction should not be visible while the newest transaction should be on the top. In the

		newest transactions so that they become the newest and oldest respectively then re-opening the RecentTransactionsTable	second opening, the previously oldest transaction should be on top and the previously newest transaction should not be displayed
Success Criterion 5: The client must have the ability to edit or delete existing transactions. Editing transactions will carry the same validation limitations (maximum amount being 100,000 and maximum transaction name length being 20 characters).	Deletion of transactions	Deleting a transaction through the DeleteTransactionsModal component	Able to see success/error message. If deletion was successful, the deleted transaction document should no longer be visible in the 'transactions' collection of the database. The deleted transaction must not appear in any components of the application
	Ability to edit the fields of existing transactions	Editing all the fields for a transaction with different normal data values through the EditTransactionsModal component	Able to see success/error message. If successful, the edited transaction document in the 'transactions' collection of the database should show the updated values for all the fields. The changes in the field values must be reflected in all components of the application
	Effect of leaving a field's value as blank when trying to edit a transaction (abnormal value)	Deleting the pre-filled value of one of a transaction's fields in EditTransactionsModal and clicking on the update button. Steps to be repeated for each field in the EditTransactionsModal component	Able to see warning message informing the user that all transaction fields must have a value and the transaction editing process not be executed
	Effect of inputting a date that doesn't follow the	Editing the date added field of a transaction via the EditTransactionsModel	Able to see warning message informing the user that the date added field of the

MM/DD/YYYY format (abnormal value)	to be in the DD/MM/YYYY format (where the number of the month is above 12 and/or the date is above 31) . Other transaction fields left unchanged	transaction must be in the MM/DD/YYYY format and the transaction editing process not be executed
Editing transaction amount to be less than 100,000 (normal value)	Editing an existing transaction's amount field to be 50,000 (if it is not already). Other transaction fields left unchanged	Able to see success/error message regarding the edit of the transaction amount. If the update was successful, the new transaction amount should be visible in the 'transactions' collection in the database as a Firestore document with the correct updated values in each field
Editing transaction amount to be equal to 100,000 (extreme value)	Editing an existing transaction's amount field to be 100,000 (if it is not already). Other transaction fields left unchanged	Able to see success/error message regarding the edit of the transaction amount. If the update was successful, the new transaction amount should be visible in the 'transactions' collection in the database as a Firestore document with the correct updated values in each field
Editing transaction amount to be greater than 100,000 (abnormal value)	Editing an existing transaction's amount field to be 2,00,000. Other transaction fields left unchanged	Able to see error message informing the user that the amount specified must be under 100,000 and the transaction editing process not be executed
Adding a transaction name that is less than 20 characters in length in	Editing an existing transaction's name to be 15-characters long. Other transaction fields left unchanged	The new transaction name should be permitted and, if successfully added to the 'transactions'

	EditTransactionsModal (normal value)		collection of the database, should be visible in the ‘transactionName’ field of the transaction’s Firestore document
	Adding a transaction name that is 20 characters in length in EditTransactionsModal (extreme value)	Editing an existing transaction’s name to be 20-characters long. Other transaction fields left unchanged	The new transaction name should be permitted and, if successfully added to the ‘transactions’ collection of the database, should be visible in the ‘transactionName’ field of the transaction’s Firestore document
	Adding a transaction name that is longer than 20 characters in length in EditTransactionsModal (abnormal value)	Editing an existing transaction’s name to be 25-characters long. Other transaction fields left unchanged	Error message shown to client alerting that transaction names should be less than or equal to 20 characters long and the transaction editing process not be executed
	Effect of non-numerical characters in the transaction amount field in the EditTransactionModal component (abnormal value)	Inputting non-numerical characters in the amount field in the EditTransactionsModal component	Non-numerical text not being displayed in the transaction amount field
	Effect of non-numerical characters in the date added field in the EditTransactionModal component (abnormal value)	Inputting non-numerical characters in the date added field in the EditTransactionsModal component	Non-numerical text not being displayed in the date added field
Success Criterion 6: In order to segregate transactions, the solution should have the functionality to create, edit, and delete transaction categories, and they	Creating a category with a name that less than 25 characters in length (normal value)	Creating a new transaction category via the CategoryTable component and typing in a 20-character name for the category	Category name should be permitted and, if successfully added to the ‘categories’ collection of the database, should be visible in the ‘categoryName’ field of

should be less than or equal to 25 characters in length.			the category’s Firestore document
	Creating a category with a name that is 25 characters in length (extreme value)	Creating a new transaction category via the CategoryTable component and typing in a 25-character name for the category	Category name should be permitted and, if successfully added to the ‘categories’ collection of the database, should be visible in the ‘categoryName’ field of the category’s Firestore document
	Creating a category with a name that longer than 25 characters in length (abnormal value)	Creating a new transaction category via the CategoryTable component and typing in a 30-character name for the category	An error message should be shown which alerts the client that the category name cannot be longer than 25 characters in length
	Effect of leaving the category name field blank when trying to add a category (abnormal value)	Not inputting any value in the category name input field in the CategoryTable component and clicking on the add button	Able to see warning message informing the user that the category name field must have a value and not be left blank
	Editing the name of an existing transaction category with a name that is less than 25 characters long (normal value)	Clicking on the edit button next to a category’s name in the CategoryTable component and inputting a new name that is 20-characters long	Able to see success/error message. If successful, the edited category document in the ‘categories’ collection of the database should show the updated value for the categoryName field. The change in the field’s value must be reflected in all components of the application
	Editing the name of an existing transaction category with a name that is 25 characters long (extreme value)	Clicking on the edit button next to a category’s name in the CategoryTable component and inputting a new name that is 25-characters long	Able to see success/error message. If successful, the edited category document in the ‘categories’ collection of the database should show the updated value for the categoryName

			field. The change in the field’s value must be reflected in all components of the application
	Editing the name of an existing transaction category with a name that longer than 25 characters long (abnormal value)	Clicking on the edit button next to a category’s name in the CategoryTable component and inputting a new name that is 30-characters long	An error message should be shown to the client which alerts him that category names cannot be longer than 25 characters in length
	Effect of leaving the category name field blank when trying to edit a category’s name (abnormal value)	Clicking on the edit button next to a category’s name in the CategoryTable component and then not inputting any value in the category name input field and clicking on the tick button	Able to see warning message informing the user that the category name field must have a value and not be left blank
	Deleting transaction categories	Clicking on the trash-bin button next to a category’s name in the CategoryTable component	Able to see the DeleteCategoriesModal component. Clicking on ‘Delete’ button in the aforementioned component should result in the user being able to see success/error message. If deletion was successful, the deleted category document should no longer be visible in the ‘categories’ collection of the database. The deleted category must not appear in any components of the application. Moreover, the transactions that were within the deleted category must not exist anywhere in the application or in the database as documents

			in the ‘transactions’ collection
Success Criterion 7: The client must be able to filter transactions by their type (income/expense), category, date added and the magnitude of the amount.	Ability to filter transactions by their type (income/expense)	Selecting either "Income" or "Expense" from the TransactionFilter component’s dropdown menu. No filters specified for the other filtering components	Only the transactions of the selected type (income/expense) are displayed in the Budget page
	Ability to filter transactions by category	Selecting a category from the CategoryFilter component’s dropdown menu. No filters specified for the other filtering components	Only the transactions belonging to the selected category should be displayed
	Ability to filter transactions by the date they were added	Selecting a start and end date within the current year using the DateFilter component. The end date should be on or after the start date. No filters specified for the other filtering components	Only the transactions within the selected date range should be displayed
	Ability to filter transactions by the transaction amount	Entering a minimum and maximum amount (normal values only) in the AmountFilter component and clicking on the “Apply Filter” button. Repeated by inserting a hyphen sign (-) in the input fields. No filters specified for the other filtering components	For positive values, incomes with an amount that falls within the specified range are displayed. For negative values, expenses with an amount that falls within the specified range are displayed
	Ability to remove filters and display all transactions	Clicking on the "Remove Filter" button in the CategoryFilter, DateFilter, and AmountFilter components. No filters specified for the other filtering components	All transactions should be displayed
	Ability to apply multiple filters simultaneously	Selecting valid values and options in multiple	Only the transactions that match all selected

	(type, category, date, amount)	filters: TransactionFilter, CategoryFilter, DateFilter, and AmountFilter components. No filters specified for the other filtering components	filters should be displayed
	Validation of the amount input in the AmountFilter fields using normal data	Inputting 50,000 and 70,000 in the AmountFilter fields for the minimum and maximum amount values respectively. No filters specified for the other filtering components	Transactions with an amount that falls within the specified amount range are displayed
	Validation of the amount input in the AmountFilter fields using extreme data	Inputting 0 and 100,000 in the AmountFilter fields for the minimum and maximum amount values respectively. No filters specified for the other filtering components	All transactions are shown
	Validation of the amount input in the AmountFilter fields using abnormal data	Attempting to input non-numerical characters and then 200,000 in either of the AmountFilter fields. No filters specified for the other filtering components	Non-numerical values not displayed in the input fields. Alert shown to client if either of the amount fields exceed 100,000 in magnitude. All transactions are shown
	Validation of the amount input in the AmountFilter fields when the maximum amount is lower than the minimum amount	Inputting 10,000 and 5,000 in the AmountFilter fields for the minimum and maximum amount values respectively. No filters specified for the other filtering components	Alert shown to client regarding this data validation error. All transactions are shown
	Selecting an “End Date” date that is before the “Start Date”	Selecting a “Start Date” as the current date and attempting to select yesterday’s date in the “End Date” field	This action is not permitted as the dates before the current date are “grayed out” (meaning they cannot be selected by the

			client). All transactions are shown
Success Criterion 8: The implemented solution should display monthly transaction data in graphs (daily incomes/expenses, daily balance, and monthly categorical earnings/spendings). This data must be filtered by month and year.	Effect of changing the selected month in each graph in the Reports page	Selecting various months through the dropdown located in the top of each graph component in the Reports page	Each graph must update with the new transactions to reflect the changes in the selected month
	Effect of changing the selected year in the Reports page	Selecting various years through the dropdown located at the top of the Reports page	All graphs must update with the new transactions to reflect the changes in the selected year
	Effect of hovering over each bar and section in the bar graphs and pie charts respectively	Hovering the mouse cursor over each bar and section in the bar graphs and pie charts respectively	The graphs must show a hover-box below the cursor to show more information regarding each bar/section (hovering over bars should show the magnitude of the daily income/expense or the daily balance) (hovering over sections should show the name of the category and the sum of its incomes/expenses)
Success Criterion 9: The application must be fully accessible and functional on both the client's MacBook and Windows desktops. The interface should maintain consistency across devices, ensuring that there is no loss of functionality between platforms.	Testing if and how the application appears in the client's Macbook laptop and Windows desktop	Opening the application in both the client's Macbook at home as well as his Windows desktop at the office	The application must function as expected in both operating systems. The app's components should be placed in similar positions/areas in both systems after accounting for the varying screen dimensions
	Adding, deleting and editing transactions in both of the client's computer systems	Adding, deleting and editing transactions using normal, extreme, and abnormal data in the client's Macbook and then performing the same transaction	The transaction operations must process as expected without any errors in both systems

		operations in his Windows desktop	
Success Criterion 10: The product should be able to handle data validation errors and provide feedback regarding the completion of database operations with clear success/warning/error messages.	Checking if application prevents submission when any transaction form fields are left empty	Attempting to submit the IncomeForm and ExpenseForm components with empty fields and clicking on the submit button	Client receives an error message ("Please fill out all the fields.") and the form submission is cancelled
	Ensuring that transactions with amounts greater than 100,000 are rejected	Entering 200,000 (abnormal value) as the amount value in the IncomeForm and ExpenseForm components then clicking on the submit button	Error message "Transaction Amount should be less than or equal to 100,000" appears, and form submission is cancelled
	Verifying that the transaction name cannot exceed 20 characters	Entering transaction name 30 characters in length (abnormal value) in the IncomeForm and ExpenseForm components then clicking on the submit button	Error message "Transaction name should be less than or equal to 20 characters long" should be displayed, and the form submission should be cancelled
	Ensuring that transaction dates follow the "MM/DD/YYYY" format during transactions editing	Editing the date field for a transaction in the EditTransactionModal component and inputting a date in an incorrect format (DD/MM/YYYY) then clicking on the "Update" button	An error message alerts the user with "Date must be in MM/DD/YYYY format" and the edit does not proceed
	Ensuring that category names do not exceed 25 characters	Enter a category name 30 characters in length (abnormal value) in the CategoryTable and then clicking on the "Add" button	An error message alerts the client with "Category name cannot be longer than 25 characters," and the category should not be added
	Verifying that the user receives feedback when a transaction is successfully added to the database	Create an income or expense with normal data values and submit the form	A success message appears, confirming that the income or expense has been added to the database for the client

	Ensuring that the user is notified when a transaction is edited successfully	Edit a transaction in the EditTransactionModal using normal values and clicking on the "Update" button	A success message appears, and the updated transaction values should be reflected in the database
	Ensuring that the user receives error feedback when invalid data prevents a transaction from being updated	Edit a transaction with invalid data (e.g., name longer than 20 characters or amount greater than 100,000) in the EditTransactionModal and click the "Update" button	The client receives an error message explaining the issue relating to the specific field in which the abnormal value is present in (ex. "Transaction amount should be less than or equal to 100,000") and the transaction is not updated
Success Criterion 11: All changes (adding, editing, or deleting transactions and categories) must be updated in real-time across the client's devices and reflected immediately in the application. 11. All changes (adding, editing, or deleting transactions and categories) must be updated in real-time across the client's devices and reflected immediately in the application.	Adding a transaction is reflected in real-time across all devices	Add a transaction using the IncomeForm component on one device. Immediately check the same transaction on a different device	The newly added transaction should be visible in the transactions table on both devices without the need for a page refresh
	Editing a transaction is reflected in real-time across all devices	Edit an existing transaction using the EditTransactionModal component on one device. Immediately check the updated transaction on a different device	The edited transaction with updated values should be visible on both devices in the transactions table without requiring a page refresh
	Deleting a transaction is reflected in real-time across all devices	Delete a transaction using the DeleteTransactionsModal component on one device. Immediately check if the transaction has been removed from a different device	The deleted transaction should no longer appear in the transactions table on both devices without needing a page refresh
	Adding a category is reflected in real-time across all devices	Add a new transaction category using the CategoryTable component on one device. Immediately	The newly added category should appear in the categories dropdowns and lists across both devices without a page refresh

		check the category list on a different device	
	Editing a category is reflected in real-time across all devices	Edit an existing transaction category using the EditCategoryModal component on one device. Immediately check the category on a different device	The edited category name should be updated in the categories list on both devices without a page refresh
	Deleting a category is reflected in real-time across all devices	Delete a category using the DeleteCategoriesModal component on one device. Immediately check the category list on a different device	The deleted category should no longer appear in the categories dropdowns and lists across both devices without needing a page refresh. Any transactions associated with the deleted category should also no longer appear in any component across both devices
	Filtering transactions by category should update in real-time across devices	Apply a category filter using the CategoryFilter component on one device. Immediately check if the filter is applied consistently on a different device	The filtered list of transactions should be displayed correctly across both devices
	Filtering transactions by amount should update in real-time across devices	Apply an amount filter using the AmountFilter component on one device. Immediately check if the filtered results are reflected on a different device	The filtered list of transactions by amount should appear correctly on both devices without needing a page refresh
	Adding a large number of transactions should update in real-time across devices without delay	Add a batch of 50 transactions using the IncomeForm and ExpenseForm components on one device. Check for any delays or missing transactions on a different device	All added transactions should appear in real-time on both devices without any delay