2024

Final Year Project

BUDGETING WEB APP

RACHEL RING

Tasks and Uploads

Iterations and Releases

Iteration 1 can be found here.

Iteration 2 can be found here.

Iteration 3 can be found here.

Iteration 4 can be found here.

Iteration 5 can be found here.

Iteration 6 can be found here.

Iteration 7 can be found here.

Iteration 8 can be found here.

Documentation

UCLA and Research documentation can be found in the Documentation directory in my Final Year Github.

Code Quality

Used `@angular-eslint/eslint-plugin` v17.4.1.

Used `SonarLint` v4.5.1 extension on Visual Studio Code.

All tracked in #44.

Technologies

Angular: v17.0.3

Typescript: v5.2.2

Jasmine: v5.1.0

Golang: v1.20.2

C#: v12.0

.NET: 8.0

Entity Framework Core: v8.0.3

IDE:

- Visual Studio Code: v1.89.1
- Visual Studio: v17.9.2

Frontend – client-side

For the frontend I used Angular web framework. For the project I used Agular's standalone components as it is becoming industry standard to use this instead of `ngModule`.

List of components and their uses

Component Name	Use
BudgetPlannerComponent	- If the user is not authenticated, they are encouraged to log in or register. - If the user is authenticated: Creates a form for the user to fill in their income and expenses. - If the user is Authenticated and they have a budget, their budget values will be loaded into the form as yearly values. - Budget can be saved – sent to c# backend. - User can view an initial breakdown of their income and expenses in doughnut charts.
DataVisualisationsComponent	- Creates Income charts using data from Go bakend - Creates Expense charts using data from Go backend - The expense charts can be filtered by household size - If the user is authenticated: they are shown personalised recommendations on how to save money if their expenses are over the national average for the chosen household size
ExpenseRecommendationComponent	- Child component to DataVisualisationsComponent - Displays the component red or green based on whether the user is under or above average for that expense category
RecommendationListComponent	- Child component to ExpenseRecommendationComponent - Displays the links to help the user save in that expense category.
NavbarComponent	- Shows menu options for the user. - If the user is authenticated: they are shown a menu option to view their budget breakdown and in the user menu the option to reset their password and log out. - If the user is not authentication: they are shown options in the user menu to log in or register.
LoginUserComponent	- Creates form for user to log in. - On submit send request to c# backend for authentication and logs the token and user email to application storage in the browser.

PasswordResetComponent	- Creates form for logged in users to change their password. - Uses custom password validator to make sure the passwords match.
RegisterUserComponent	- Creates form for new user to register. - Uses custom password confirmation validator. - Sends request to the c# backend and if there's an error it is displayed to the user
UserBudgetComponent	- Shows the user a breakdown of their finances. - Shows the user's income in a sortable table. - Shows the users expenses in a sortable table. - Breakdown of expenses and how much they contribute to the overall expenditure.
WelcomePageComponent	- Landing page for the user

List of Services and their uses

Service	Use
PasswordConfirmationValidatorService	Custom validator that checks if the password and the confirm password are the same.
AuthenticationService	General HTTP requests used all over the application.

How to Run

```
$ cd code-and-docs-rachelrring/client-side/budget-app
```

```
$ npm install
```
←Only run this if it's your first time running the code

```
$ ng serve --port 8080
```

Backend

Go Backend

This backend handles all the requests for data from the CSO.

Originally, I wanted to request the data from CSO but no matter what way I formed the struct to unmarshall the data, it would not work.

I ended up having to download the .csv files and reading from the file directly.

My Golang Gin server has 5 GET endpoints.

Endpoint	Use
/hs067	Responds with all the data from hs067.csv
/hs067Region	Responds with the data from hs067.csv sorted by Region
/hs208	Returns all the expenditure data from hs208.csv
/hs208OverView	Returns chosen rows from hs208.csv

/hs208Recommendations	Returns chosen rows from hs208.csv

How to run

```
$ cd code-and-docs-rachelrring/server-side
```

```
$ go run .
```

The Go backend will automatically run on `localhost:8070`

C# Backend

This backend handles user functionality and budget data.

I used the Identity class supplied by ASP .NET core framework to create my user class. Because I used the identity class, it autogenerated a lot of endpoints that can be used out of the box:

- /register
- /login
- /refresh
- /confirmEmail
- /resendConfirmationEmail
- /forgotPassword
- /resetPassword
- /manage/2fa
- /manage/info

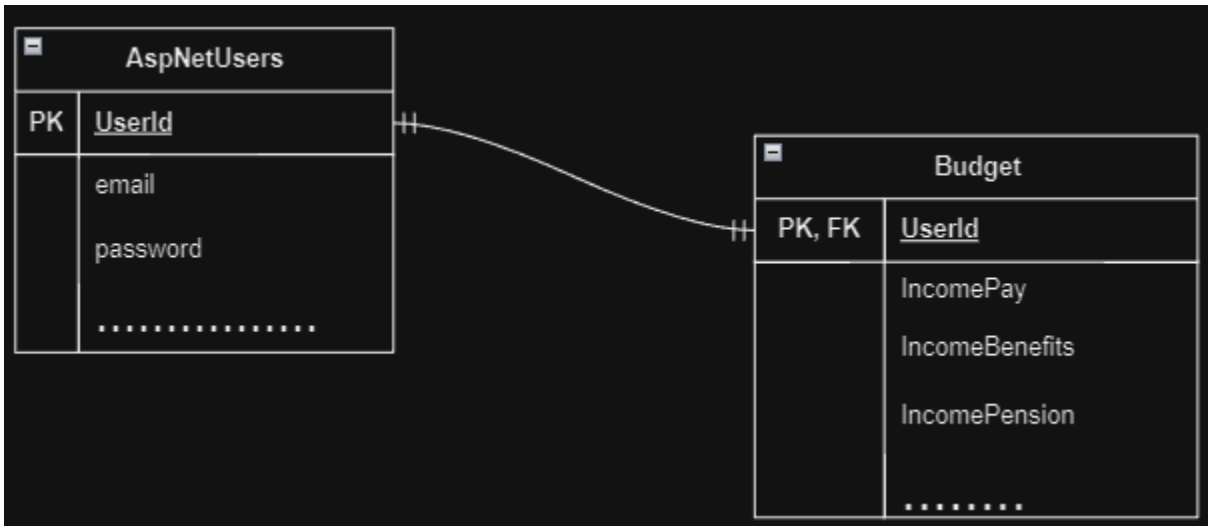
The following endpoints were written by me.

Method	Endpoint	Use
POST	/api/Accounts/Registration	Creates a new user and inserts it to the database
POST	/api/Accounts/NewBudget	Request has newBudget in the body. Budget is written to the database. It can create or update an existing budget.
POST	/api/Accounts/PasswordReset	Change password of a user.
GET	/api/Accounts/Budget/{userEmail}	Get user budget given their email.

Database

The database is connected to the C# backend. It was created using entity framework core – code first approach. It is stored in an SQL database locally.

Entity Relational Diagram



The relationship between a user and a budget is 1:1. A user can optionally have a budget, but a budget cannot be made without the user being made first.

Testing

Frontend

Frontend tests were written in typescript and run using Karma test runner.

In the entire client-side application, there are 83-unit tests.

To run all tests

```
$ cd code-and-docs-rachelrring/client-side/budget-app
```

```
$ ng test
```

Backend

Go backend

Go backend tests are written with go testing package. There are a total of 6 unit tests written for the go backend.

Issues I want to mention

Deployment

Issue #62 – this issue tracks my problem deploying this project.

I couldn't deploy this project on Azure because the college disabled my subscription as it was past the end of the college year.

SonarCloud

Issue #44

I couldn't use SonarCloud for this project as I don't own the repository. As a replacement, I have installed the Sonar Lint extension in Visual Studio code.

Further Improvements

- Users can input their own custom expenses and income.
- Find a more reliable API to get expenses data for Ireland.
- Implement Email Sender in C# backend.