# Running the Application

## App Architecture

The app has an Angular frontend, found in /client, and a Node.js server, found in /server.

To get the application up and running, a couple of steps are first needed.

## Database Setup

You must first set-up a local instance of the database. This can be done by

1. Downloading PgAdmin here.
2. Set up a server and remember the username and password you use during set-up.
3. Create the entities:
    - Set-Up SQL:

```
CREATE DATABASE budgetingapp

CREATE EXTENSION pgcrypto;

CREATE TYPE public.payment_type AS ENUM (
    'credit',
    'debit',
    'cheque',
    'cash',
    'bank_transfer'
);

CREATE TYPE public.repeat_type AS ENUM (
    'daily',
    'weekly',
    'monthly',
    'yearly'
);

CREATE TYPE public.transaction_type AS ENUM (
    'expense',
    'income',
    'savings'
);

CREATE TYPE public.category_type_enum AS ENUM (
    'expense',
    'income',
    'savings'
);
```

- Users:

```
create table public.users (
    user_id serial not null,
    username character varying(30) not null,
    email character varying(255) not null,
    password character varying(255) not null,
    default_currency character varying(3) not null,
    starting_balance numeric(20, 2) not null default 0.00,
    constraint users_pkey primary key (user_id),
    constraint users_email_key unique (email),
    constraint users_username_key unique (username)
) TABLESPACE pg_default;
```

- Category:

```
create table public.category (
    category_id serial not null,
    name character varying(255) not null,
    category_type public.category_type_enum not null,
    constraint category_pkey primary key (category_id)
) TABLESPACE pg_default;
```

- Budget:

```
create table public.budget (
    budget_id serial not null,
    user_id serial not null,
    category_id serial not null,
    amount numeric(20, 2) not null,
    constraint budget_pkey primary key (budget_id),
    constraint unique_user_category unique (user_id, category_id),
    constraint budget_category_id_fkey foreign KEY (category_id)
references category (category_id),
    constraint budget_user_id_fkey foreign KEY (user_id) references
users (user_id) on delete CASCADE
) TABLESPACE pg_default;
```

- Savings_Goal

```
create table public.savings_goal (
    goal_id serial not null,
    user_id serial not null,
```

```
    goal_amount numeric(20, 2) not null default 0.00,
    starting_savings numeric(20, 2) not null default 0.00,
    goal_due_date date null,
    name character varying(255) not null,
    ranking bigint not null,
    constraint savings_goal_pkey primary key (goal_id),
    constraint savings_goal_user_id_fkey foreign KEY (user_id)
references users (user_id) on delete CASCADE
) TABLESPACE pg_default;
```

- Transaction:

```
create table public.transaction (
    transaction_id serial not null,
    user_id serial not null,
    category_id serial not null,
    type public.transaction_type not null,
    name character varying(255) not null,
    transaction_date date not null,
    amount numeric(20, 2) not null,
    shop character varying(255) null,
    payment_method public.payment_type null,
    repeat boolean null,
    repeat_schedule public.repeat_type null,
    end_date date null,
    repeat_group_id uuid null,
    constraint transaction_pkey primary key (transaction_id),
    constraint transaction_category_id_fkey foreign KEY (category_id)
references category (category_id),
    constraint transaction_user_id_fkey foreign KEY (user_id)
references users (user_id) on delete CASCADE
) TABLESPACE pg_default;
```

## Setting Up The Env File

In /server create a .env file. You will need to fill in the following environment variables:

```
DB_USER=[your pgadmin username]
DB_PASSWORD=[your pgadmin password]
DB_HOST=[your pgadmin database host]
DB_PORT=[the port your pgadmin database is running on]
DB_DATABASE=budgetingapp
JWT_SECRET=[will come back to this]
NODE_ENV=development
```

To create a JWT secret, run the following in a terminal and copy the output into the JWT_SECRET value:

```
node -e "console.log(require('crypto').randomBytes(64).toString('hex'));"
```

## Downloading Dependencies

cd into client and run:

```
npm install
```

Do the same in the server folder.

## Running the Application

When in the client folder, run:

```
npm run start
```

When in the server folder, run:

```
npm run start-dev
```

## Testing the Application

### Frontend

To run the unit tests for the frontend, once in the client folder, run:

```
npm test
```

To get code coverage for frontend tests, run:

```
npm run test-coverage
```

The coverage report can be found at client/coverage/evergreen-budgeting-app/index.html

### Server

To run tests for the server, run:

```
npm test
```

This will produce the coverage too, which can be found in server/coverage/lcov-report/index.html.