

# Budget App (Ruby on Rails + PostgreSQL + Docker)

This is a simple Budget Management application built with Ruby on Rails.

It allows users to manage their categories and records (e.g., groceries, bills).

The app uses PostgreSQL for data storage and is containerized using Docker and Docker Compose.

## Technologies Used:

- Ruby 3.1.2
- Ruby on Rails 7.0.1
- PostgreSQL 14
- Docker & Docker Compose
- Devise (User authentication)

## Prerequisites:

- Docker installed
- Docker Compose installed
- Git

## 1. Clone the Repository

```
git clone https://github.com/evans22j/Budget-App.git
```

```
cd budget-app
```

## 2. Dockerfile – Build your Rails Image

This file tells Docker how to build your app image.

## 3. docker-compose.yml - Define Multi-Container App

This file tells Docker how to run your web app and database together.

## 4. entrypoint.sh - Fix PID Issues in Docker

Rails sometimes keeps a server.pid file, causing the server to crash when restarting. This script deletes it on container start.

NOTE: Make sure this file is executable

## 5. config/database.yml - Connect Rails to Postgres

This file tells Rails how to connect to the database using the ENV variables from docker-compose.yml.

## 6. Steps to Build and Run the App

### Build the Docker image

```
docker-compose build --no-cache
```

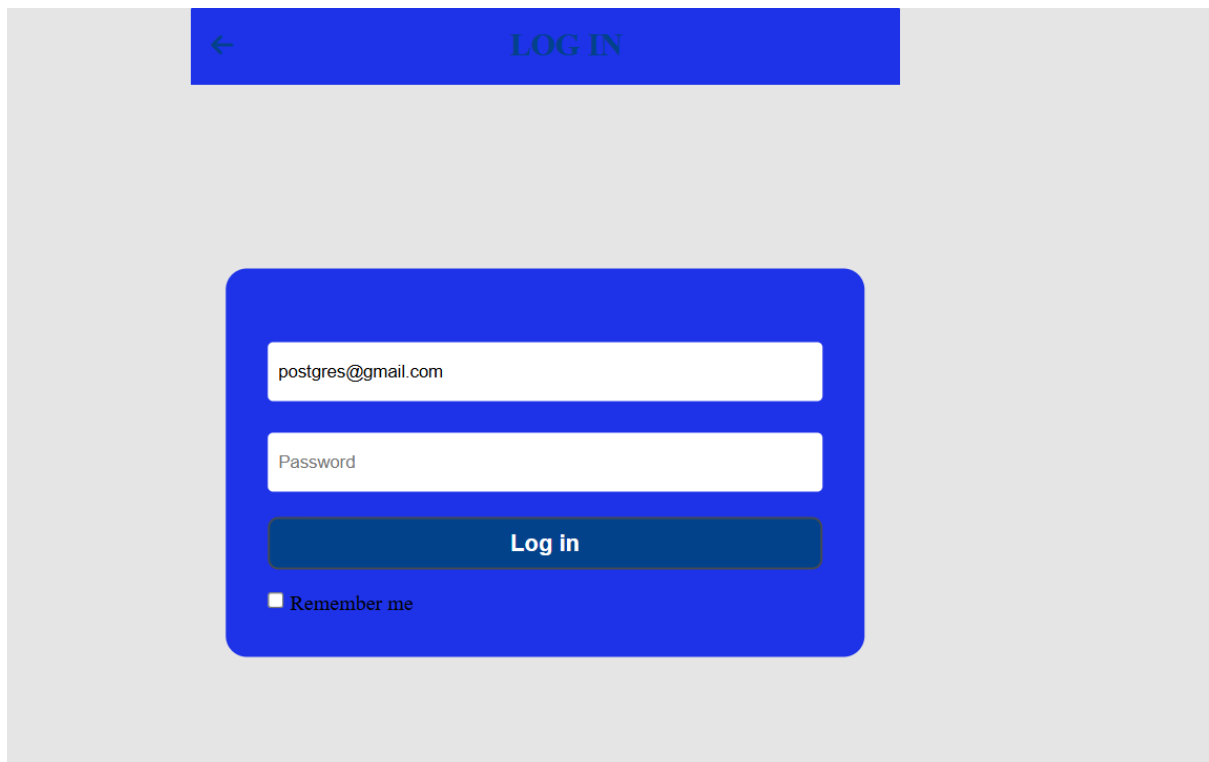
### Start the containers

```
docker-compose up
```

### Setup the database

```
docker-compose run web rails db:create
```

```
docker-compose run web rails db:migrate
```



← LOG IN

postgres@gmail.com

Password

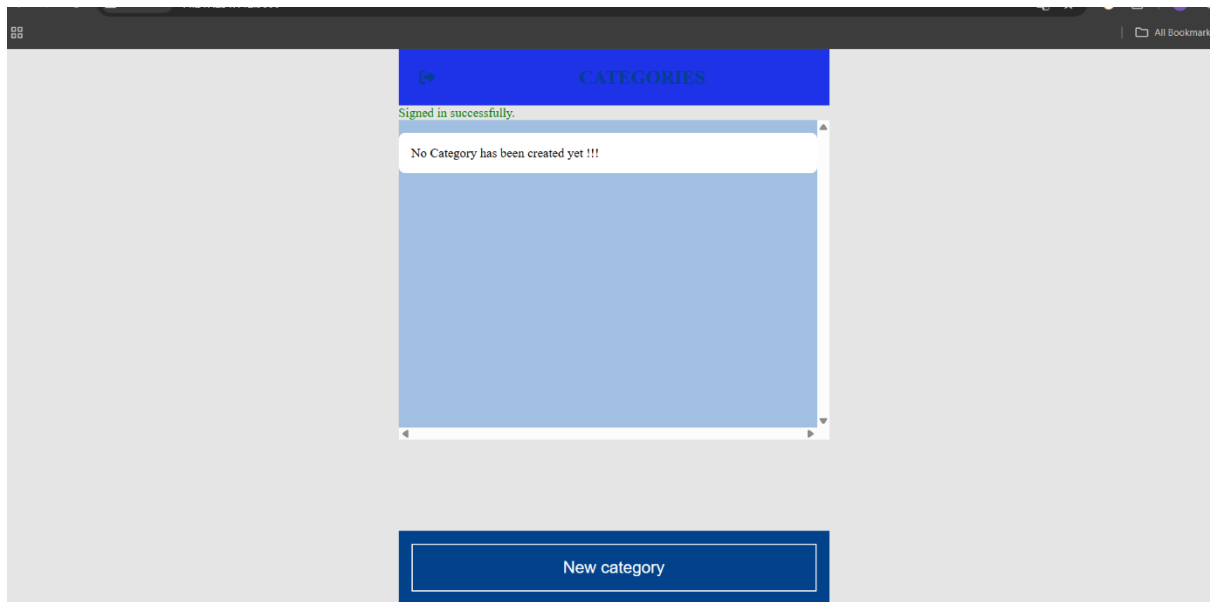
Log in

☐ Remember me

### Create admin user

```
docker-compose run web rails console
```

```
User.create!(name: "Admin", email: "admin@example.com", password: "password",  
password_confirmation: "password")
```



Add a New Category in rails console

# First, check the user

User.first

# If user exists, create a category

Category.create!(name: "Groceries", user\_id: 1)

To display the added category in your Rails app, you need to make sure it is rendered inside the view — typically in `app/views/categories/index.html.erb` using the partial `_category.html.erb`

