Task&BudgetTracking

The task and budget tracking system is a web app for small businesses and non-profits to track employees and volunteers tasks and the organizations spending.

Contents

[**Architecture** 1](#_Toc84425110)

[React 1](#_Toc84425111)

[Flask/PostgreSQL 1](#_Toc84425112)

[Nginx 1](#_Toc84425113)

[**Setup** 1](#_Toc84425114)

[Create your config files 1](#_Toc84425115)

[Build and Start the application 2](#_Toc84425116)

[Run the application tests 2](#_Toc84425117)

# **Architecture**

The application consists of three parts. All three need to be built into a Docker Compose deployment.

## React

The Front End is written in React and utilizes React Bootstrap to give a reactive UI.

## Flask/PostgreSQL

The Back End is written in Flask and utilizes FlaskSQLAlchemy to store and manage a PostgreSQL database.

It also stores images in the file system of its container.

## Nginx

The application uses an Nginx proxy server to handle directing of requests and encryption.

# **Setup**

You will need docker, docker compose (these come bundled together if you are using docker desktop), and git installed to follow these instructions.

In order to get started with this application pull a clone of the repository to your local environment.

## Create your config files

Sample files have been provided for ease of quick setup in their respective files.

1. Create a docker-compose.yml file following the instructions in docker-compose.yml.patern

2. If you are deploying to a production environment and planning on using an ssl certificate create a nginx.prod.conf file from the nginx.prod.conf.patern file in the Nginx folder.

## Build and Start the application

1. Make sure docker is installed.
2. Build the images. (if the front end fails to build run “npm install” from the react sub-folder in your command terminal before building)
3. Then run the docker compose file from the root folder of the project with `docker-compose up`
4. If the application does not generate the database tables, follow the following steps to manually run the database migration
   1. Delete the contents of the flask/migrations/versions folder
   2. Attach to the running back-end container. Then run the following commands
   3. Python -m manage db init
   4. Python -m manage db migrate
   5. Python -m manage db upgrade
   6. Python -m manage create\_admin
   7. Python -m manage run
5. Stop the docker containers then restart them with the docker-compose up command

## Run the application tests

This process will clear any data in your local container volumes only run this sequence in your dev environment if you have made modifications to the source code and want to revalidate the unit tests

1. Rename the docker-compose.test.yml docker-compose.yml

2. Build the compose file

3. Run the compose file with `docker-compose up --abort-on-container-exit`

These commands will produce a testing.txt file and a htmlcov folder in the flask folder. The results of the tests can be viewed in the testing.txt file and the coverage report can be viewed by opening intex.html in the htmlcov folder. After you are done change the file names on the docker-compose.yml files. Make sure you rebuild the images before continuing.