Software Engineering project 2023/2024
Ca’ Foscari University of Venice

FinanceFlow

Installation Guide

AgileMasters

Document Informations

ProjectName	FinanceFlow	**Acronym**	FinFlow
Deliverable	Installation Guide		
Delivery date	15/01/2024		
Team leader	Alberto Tomasin		892614@stud.unive.it
Team members	André Ramolivaz Simone Dinato Mirco De Zorzi		891923@stud.unive.it 892539@stud.unive.it 891275@stud.unive.it

Document History

Version	Issue Date	Stage	Contributors (reviewer)
Final	10 /12/2023	Complete	Alberto Tomasin, Mirco De Zorzi, (Simone Dinato, Andrè Ramolivaz)

Indice

1. Introduction

FinancialFlow is an innovative trading application designed to cater to a diverse audience, ranging from amateur traders to experienced investors. At its core, FinancialFlow offers a dynamic and interactive trading experience, backed by real-time market data and a suite of analytical tools. The app's frontend is developed using Flutter, a versatile framework known for its native performance and seamless cross-platform functionality. On the backend, Django serves as the robust, high-level Python web framework, ensuring secure, fast, and scalable performance. This combination promises a seamless and responsive user experience, with the reliability and efficiency required in the fast-paced world of trading.

1.1. Structure

This guide is crafted to assist you, the user, in downloading, installing, and initially configuring FinancialFlow on your device. Aimed primarily at facilitating developers and first-time users, this document will provide:

- Step-by-step instructions on installing FinancialFlow.
- Guidance for initial setup and configuration.
- Essential information for getting started with the app.

2. Running using Docker Compose

2.1. Prerequisites

To run FinanceFlow using Docker Compose, you need to have Docker and Docker compose installed and working in your machine. You can install them here.

2.2. Installation

2.2.1. Clone the repository

Clone the repository into your local machine using the following command:

```
Unset
git clone https://github.com/3Lance/CT0090
```

2.2.2. Navigate into the directory

By using

```
Unset
cd path/to/the/cloned/repo
```

you can navigate to the cloned repository.

Then, use your favourite text editor to open it and view source code.

2.2.3. Configuring environment variables

Open the **docker.env** file located in the project's root directory to customise the environment variables to your specific requirements.

2.2.4. Start the containers

Build and start the Docker containers using the following command:

```
Unset
docker-compose up --build
```

This command is responsible for pulling necessary images, building all the services and starting all the containers.

2.2.5. Access to the backend

Once the container are up and running, you can access the backend by the exposed urls:

- **Backend:** http://localhost:80000
- **Websocket**: ws://localhost:8001

2.2.6. Default admin credentials

Use the following credentials to access the admin panel:

- **Usr:** admin@smolstock.com
- **Pw:** smolstock

2.2.7. Troubleshooting

- Open the **entrypoint.sh** file using your favourite text editor
- Change the line ending sequence to LF (Line Feed). You can usually do this by configuring your text editor to save the file with LF line endings.
- Save changes to the **entryopint.sh** file.
- Rebuild containers using the **docker-compose up –build** command.
- Grant execute permission to the entrypoint.sh file using the following command:

```
Unset
sudo chmod +x entrypoint.sh
```

3. Running using a local server

3.1. Prerequisites

In order for the app to run using a local server, you need to have some tools installed and working:

- **PostgreSQL**: you can download the latest version of PostgreSQL here. To run the app you need to have PostgreSQL version 15 or later.
- **Redis**: make sure you have Redis installed and working in your local machine, if not you can install it here
- **Python**: the backend is entirely written using python 3.11, but a recent version works fine.

3.2. Installation

3.2.1. Clone the repository

Clone the repository in your local machine using the following command or equivalent:

```
Unset
git clone https://github.com/3Lance/CT0090
```

3.2.2. Navigate into the directory

By using

```
Unset
cd path/to/the/cloned/repo
```

you can navigate to the cloned repository.

Then you can open it using your favourite text editor. We suggest using Visual Studio Code to edit the backend code and Android Studio for the frontend.

3.2.3. Create and activate a virtual environment

```
Unset
pip install virtualenv
virtualenv venv
venv/scripts/activate # on Windows
source venv/bin/activate # on Unix / Mac
```

3.2.4. Install dependencies

In the folder you will find a **requirements.txt** text file, it contains all the python packages used in the project:

```
Unset
pip install -r requirements.txt
```

3.2.5. Configure the environment variables

The app needs a **.env** file containing your configurations. Navigate in the **smolstock** folder, create the file and add the following variables:

```
Unset
SECRET_KEY=
DEBUG=

DATABASE_NAME=
DATABASE_USER=
DATABASE_PASSWORD=
DATABASE_HOST=

TWILIO_ACCOUNT_SID=
TWILIO_AUTH_TOKEN=
TWILIO_DEFAULT_CALLERID=

EMAIL_HOST_USER=
EMAIL_HOST_PASSWORD=

CYPTOCOMPARE_API_KEY=
```

3.2.6. Create the database

Create a PostgreSQL database and connect to it using the credentials specified inside the **.env** file. Once connected, in a terminal with the environment activated, by running the following command you will create the necessary tables:

```
Unset
python manage.py migrate
```

3.2.6.1. Populate coins

In a terminal with the environment activated, and the valid **CRYPTOCOMPARE_API_KEY**, you will populate the database with the coins.

```
Unset
python manage.py add_coins_to_db
```

3.2.7. Create a superuser

In a terminal with the environment activated, by executing the following command you will create a superuser:

```
Unset
python manage.py createsuperuser
```

Follow the steps to create a superuser. Alternatively, you can add a superuser using the command:

```
Unset
python manage.py add_superuser --email <email> --password <password>
```

3.2.8. Run the backend server

In a terminal with the environment activated, run the following command:

```
Unset
python manage.py runserver
```

Access the endpoints in your browser by connecting to http://localhost:8000 and the admin console at http://localhost:8000/admin

Use the superuser credentials to access.

3.2.9. Run websocket server

In a separate terminal window with the environment activated, launch the websockets with the command:

```
Unset
python websockets.py
```

You can then connect to the websockets from the shell using the following command:

```
Unset
python -m websockets ws://localhost:8001/
```

3.2.10. Run Redis

To run Redis, you simply need to open a terminal and run the following command:

```
Unset
redis-server
```

3.2.11. Run celery worker

In a separate terminal window, with the environment activated, run the following command:

```
Unset
celery -A smolstock worker --pool=solo -l info
```

3.2.12. Run celery beat

In a separate terminal window, with the environment activated, run the following command:

```
Unset
celery -A smolstock beat -l info
```

4. Run the frontend

4.1. Prerequisites

To run the app you need to have some tools installed:

- **Flutter**: the app has been written using Flutter version 3.16.7, but a newer version works fine, Flutter sdk can be found here.
- **Dart:** the app uses dart 3.2.4, but a newer version works fine. In case you don't have dart, the most recent version can be downloaded here.
- **Android Studio**: It's the IDE of choice to develop android applications, it can be found at the following link.

4.2. Install libraries

Inside Android Studio, open **pubspec.yaml** file in the root directory of the project and download the dependencies.

Alternatively you can do it by running the following command:

```
Unset
flutter pub get
```

4.3. Configure emulator

Open the emulator manager in Android Studio, and create a device.

To develop the app we used a Pixel 7 pro with Android 14.0

Start the device and wait for it to boot up.

4.4. Run the app

By running the **main.dart** file the app will build directly in the emulated device.