Pocket Stock

Software Design

CSCI-P465/565 (Software Engineering I)

Project Team

Matthew Pletcher

Aravind Bharatha

Angad Beer Singh Dhillon

Raghottam Dilip Talwai

---

1. Introduction

This website is being implemented via the model-view-template (MVT) architecture using Django.

1.1 System Description

Pocket Stock aims to create a universal asset management system where a user can quickly view the success/viability of stock/cryptocurrency/other types of investments they have made. It will also have a portal for users to purchase/sell assets

1.2 Design Evolution

This section is intended to document the rationale behind the selected design solution.

1.2.1 Design Issues

Requirements:

- Operating System: Any operating system will suffice.
- Environment: Django framework, PostgreSQL.
- A web browser is necessary to access the system.
- A disk space of around 10GB, RAM of 1GB and 2GHz of processor

1.2.2 Candidate Design Solutions

We have decided to build most of the components in the system using Python Django framework. We are looking to deploy the application on the IU server using IU web hosting.

1.2.3 Design Solution Rationale

Django framework is a popular choice since it is a full stack framework and more mature than other frameworks available. Django also has a very vast and high quality open source documentation which would be helpful for us to look up in case of any issues while coding.

IU web hosting is a reliable medium to host cgi scripts on the IU server without much hassle. There are documentations available in the knowledge base to assist in hosting.

1.3 Design Approach

1.3.1 Methods

Django framework is based on the MVT framework. MVT separates business logic and UI, encouraging separation of the user experience from database and scripting. Django's ORM(Object relation mapper) uses high level abstraction efficiently allowing a developer to write a Python code instead of a SQL to create or modify schemas in a database. All the database interaction is handled independently by the framework itself which allows faster prototyping with focus on a single programming language instead of switching to a SQL paradigms time and again. Eventually, using this object oriented approach will help our team to emphasis more on building up the features and caring less about database coding.

Currently our prototype includes

- A basic template based on which our webpages would be designed.
- A login and a registration page
- A landing page
- A basic design of how to integrate O-auth from a third party website and duo authentication for user authentication while logging into the website.

For testing, we would be using unit and regression testing. Unit tests are for stability and code predictability, while regression testing is for evaluating of how users interact with the code.

1.3.2 Standards

Django, by default encrypts passwords using PBKDF2 algorithm with a SHA256. This is standard security protocol regarding passwords. We are adhering to the Django standards and practices as mentioned in the official documentation. The core component in Django is called the 'app', which is a logical collection of code which implements a specific feature of our website. Each app will have its own models, views and templates conforming to the design of a MVT framework.

Apart from these, we are using Bootstrap 3's standards and practices to design our HTMLpages.

1.3.3 Tools

- Editor – Sublime, Pycharm
- Webserver- Django
- Database- PostgreSQL
- Hosting- IU web hosting

2. System Architecture

2.1 System Design

At a higher level, our system will have App for each component implementing a specific feature. At a lower level, each app has it's own Models, Logic, URL Routing, and Unit Tests.

We have designed a basic template for our HTML pages having the same header and footer. This template will be inherited for all the HTML pages that we create. In our case the *base_generic.html* file contains the template which will be inherited for every HTML page we design here on.

2.2 External Interfaces

We are using OAuth using a third party Facebook API. Users can login and register using their Facebook credentials.

We have used duo authentication to set up the two-factor authentication. Users can be complete the two - factor authentication either by sending a push, calling the registered mobile number or by feeding in the token number

3. Component Design

Component Name

Login and registration

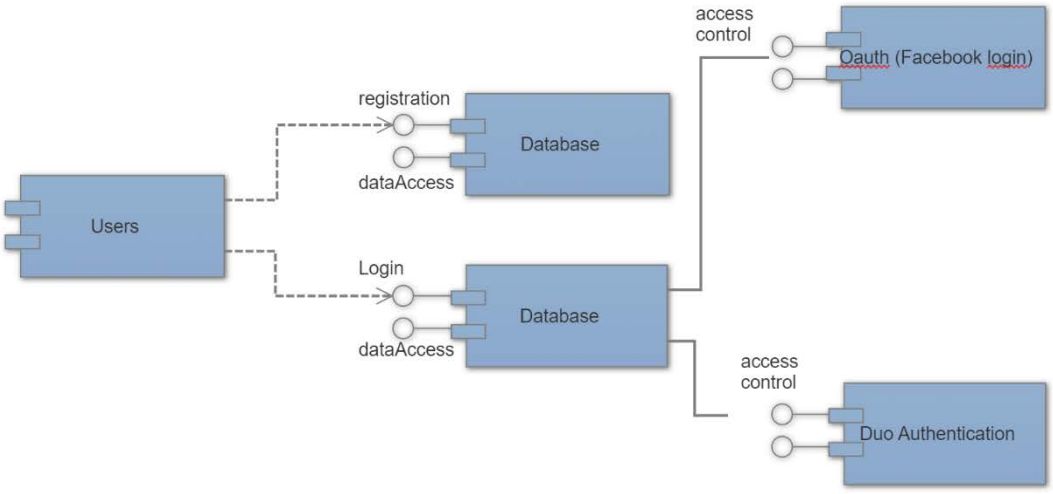
Component Description

The Login form allows the users to login with the credentials. When providing the correct credentials the user will be logged into the home page. Once the user is logged into the website, the website provides an option to logout of the web page. If the user is inactive for a particular amount of time they will be logged out accordingly. We have also implemented OAuth using a third party API such as Facebook to enable users to directly register with their Facebook credentials. The registration form will allow users who don't want to use their Facebook credentials to register by filling in their details. We have used duo authentication to set up the two factor authentication. Users can be complete the two factor authentication either by sending a push, calling the registered mobile number or by feeding in the token number.

Responsible Development Team Member

Matthew is responsible for development of this component

Component Diagram



Component Objects

- Django has built in apps to handle login and registration functionalities.

Component Interfaces (internal and external)

The component will validate internally the credentials of users.

The component will check with the third party Facebook API to validate the credentials.

The component will check with the Duo authentication to validate the two factor authentication.

Component Error Handling

If a user is trying to login with invalid credentials such as Invalid username or Invalid password, they are handled in the component.

Component Name

This component is the User Profile Component

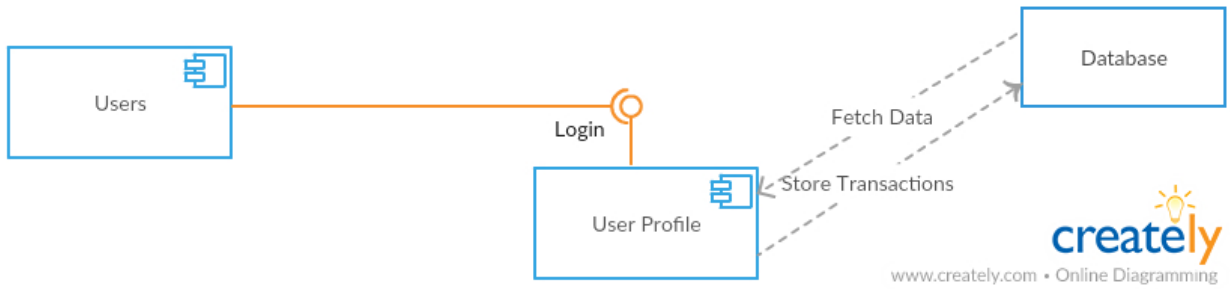
Component Description

This component can be further divided into dashboard and user actions sub components. The dashboard sub component allows the users to see the details of the stocks which have been purchased and their other investments. The user action sub component allows the user to perform transactions such as buying and selling of stocks.

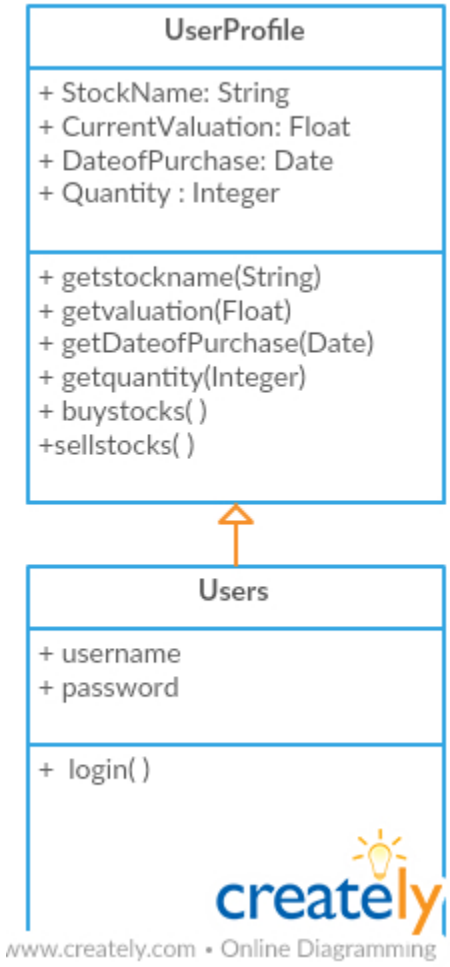
Responsible Development Team Member

Aravind Bharatha is responsible for development of this component

Component Diagram



Component Objects



Component Interfaces (internal and external)

The component will interact with the database extensively in populating the dashboard of users and also store the transactions such as buying or selling of stocks

Component Error Handling

The user cannot access their user profile without logging in. If any attempt is made to access, they are redirected to the login page.

Revision History

Revision	Date	Change Description
Sprint 1	10/1/17	Added Login and Registration Component
Sprint 2	10/15/17	Added User Profile Component