Executive Summary

Introduction

Our task was designing a software which is a small part of our main project, traders platform. It provides the basic functionality of the traders' platform. The functions can be grouped into three parts mainly: registration, parity, and manual investments. For registration, the software provides basic sign-up and sign-in functionalities. For parities, the software provides functions like listing parities, and listing current and historical exchange rates. For manual investments, there are functions for creating, deleting, listing manual investments and displaying their related profits individually or in the aggregated form. The software has a simple and user-friendly interface to display these functionalities.

Work Done

We used Django Framework to work on our API. One of our teammates was experienced on this platform so he sent us some tutorials and helped us when we faced some problems. First the system requirements were determined so we could see our path more clearly. Then the "models.py" was created so we could use it when writing "views.py". We distributed the requirements while coding, so that each of us could contribute. Everyone would either write their own tests or open an issue to test their part of the code. These tests were also shared among team-members. We tried to finish our coding quickly so our friends were able to use our code to shape the frontend. The frontend was created and we deployed our system to an AWS EC2 t3.micro instance with Ubuntu 18.04 installed. We then continued to work on the design of the API with UML diagrams. We wrote a brief description of software, a simple user manual, and deployment notes.

Moving Forward

Next semester, when it comes to implementing the project that we developed all semester, we plan to focus on teamwork even more. It is hard when all the burden of the project is carried by a few people so all of us should be more active coding, even though we may not be as experienced as others. We should give constructive feedback to each other so we can all develop and become better and better thus our project will become better as well. Next semester, we will continue to meet weekly and actively follow Github activities. These two have been really useful in our project because this kept us going, seeing our teammates work hard gave us the enthusiasm to work and contribute more to the project. At last, we believe that we will create a great product in the end with great teamwork and each member will be proud to be a part of this group.

Challenges

This project was really difficult for us compared to our other tasks so far. We only had one member who was actually experienced with the concepts. He spent a lot of his time teaching us about many subjects from Django Framework to deployment and much more. Our two meetings were mostly the rest of us getting familiarized with the concepts of the project. We spent a full day having a coding meeting in which everyone could ask for help from more experienced teammates. I believe this meeting was immensely helpful for most of us because we easily overcame some problems which would be really time-consuming otherwise. Everyone tried to do their part to contribute to the final form of the project. There were some problems with frontend and it didn't turn out the way we wanted but nonetheless, we were able to provide a working frontend with working functions when the deadline came.

Deliverables

Deliverable:	**Status:**	**Update Frequency**	**Description**
1- GitHub Wiki	Complete	Weekly	Accessible github wiki pages, up-to-date information related to project can be found.
2- GitHub Readme	Complete	As improvement needed	GitHub readme is constantly updated, includes up-to-date information about the group.
3- GitHub Issues	Complete	As improvement needed	Preparing the issue labelings and usage of the github system. We update it as we need improvements.
4- Meeting Notes	In Progress	Weekly	Publishing meeting notes in GitHub Wiki. All of them can be found in wiki.
5- Project Plan	In progress	Complete	Preparing the project plan
6- API System Requirements	Complete	Complete	System requirements for the practice API
7- API Design Diagrams	Complete	Complete	Design diagrams for the practice API
8- Collaboration Workflow Documentation	Complete	As improvement needed	Usage of templates, pull request and issue templates

Deliverable:	Status:	Update Frequency	Description
9- API Code	Complete	As improvement needed	Backend and frontend of the API
10- Deployed API	Complete	As improvement needed	Deployment of the API to the EC2.

1- GitHub Wiki: We are still in good shape in terms of github wiki. Team members are utilizing it to both keep track of the project, and demonstrate the work done. The wiki is at least weekly updated with meeting notes, in addition to that we upload whatever is done with the other assignments. Overall, the group is still good at maintaining the wiki page and utilizing it to keep everyone involved.

2- GitHub Readme: Got a good explanation of the work done in the readme, that explains what we are up to. Also, the practice-app readme has the deployed app and features, how to setup, install the requirements etc.

3- GitHub Issues: Still we are utilizing issues well. In addition to previous labels, we added labels such as backend, devops, frontend etc. to indicate exactly what has been requested in an issue.

4- Meeting Notes: We have been good at keeping meeting notes and utilizing them, and this habit of ours continues. We use the notes not only for just the sake of writing something, but keeping well intracommunication, returning the previous weeks when needed, to remember what we have discussed, etc.

5- Project Plan: We have struggled a bit in terms of writing a project plan that is following the instructors' opinions. It generally helped us see what's ahead of us and what we have done so far. It helped us question our workflow, and what could we do better in the next steps of the project.

6- API System Requirements: The first thing we did before implementation phase was to write system requirements. We decided to provide basic investments and parity functionalities, and wrote down the requirements accordingly. Then, we used this to share the responsibilities between us. We created 10 requirement items and each member was supposed to implement one of them. We have done this to provide equal opportunity of implementation to everyone.

7- API Design Diagram: We have use case diagram, class diagram and sequence diagrams. The class diagrams was almost a subset of the class diagram we did before, with minor modifications. The use case diagram and sequence diagram was also similar to the previous deliverables we have had. All in all, as usual, preparing these diagrams helped us predict and build the workflow of the API: models, methods etc.

8- Collaboration Workflow Documentation: This was one of the best things we did as a team. We prepared issue templates and pull request templates. In the pull request templates, we have 2 main things, first is making sure that the author has done everything that she was supposed to do, such as adding 2 reviewers, self-reviewing and using the format(PEP8) we agreed to use, and secondly describing what is the nature and philosophy of the implementation, to let the reviewers grasp the material easier. We did the same thing with the issue templates, with the exact same purpose. Using these templates let us be on the same page using pull requests and issues system. Overall this is one of the things we consider the highlight of our group.
In addition to this, we have used Travis and the autotesting to make our workflow even safer. It not only lets us check the recent code and recents tests, but also keeps checking the previous tests and makes us sure that we are not breaking something affecting previous features. In the future, we are also planning to use Travis to help us with the continuous integration.

9- API Code: We used Django to implement the backend of the app, and used javascript jquery to implement the frontend. We have basically 2 family of functionalities, one is investments and the other family is parity. We provide creating, deleting, listing investments and profit calculation. We also provide listing history of parities, latest parity values. We also have register and login functionalities, secured by JWT Authentication. One thing we did good is the cron job of our backend, the job basically sends request to the API we have used(https://exchangeratesapi.io/) and updates the database accordingly. When the system is initialized, we query the API using our own-written django command to fetch the history of selected parities, then cronjob constantly updates the database with new values. Hence, the parity latest and historic endpoints can provide the information fast and robust, instead of querying another API when the request sent to us. We may even directly use these systems once we start next term, since these parts are working and solid right now.

10- Deployed API: When deploying our application, we tried to apply modern software development practices as much as we could afford. We currently use an AWS EC2 t3.micro instance as our production server. To enable each team member to remotely access to the server, we have collected SSH public keys and authorized them.

We containerized each individual part of our system as Docker containers and combined them using Docker Compose. Containerization helped us continuously deploy our application as we developed and led us to version server-side configurations and software dependencies in our project repository. It also provides portability in terms of simply downloading and running application with no concern about the software environment or configurations.

We registered traiders-practice.tk domain name for our website with freenom.com. We added three subdomains (traiders-practice.tk, www.traiders-practice.tk, api.traiders-practice.tk) to resolve to the IP address of our production server.

We also enabled HTTPs on our website by getting an SSL certificate from https://letsencrypt.org to secure the communications between clients and the server. The web server (Nginx) is configured to redirect every request coming from HTTP to HTTPs.

In the future, we plan to enable continuous deployment so that the master branch gets deployed automatically whenever we merge a pull request. We will also develop a backup strategy for our database to be able to recover from disasters.

Evaluation of Tools and Processes Used

- **Amazon Web Services:** We used an AWS EC2 t3.micro instance to deploy our application. As the server is located in the U.S., there is a bit of latency when connecting to the website. We are going to choose a region in Europe in the next semester.
- **Django and Django REST Framework:** We loved Django and DRF as they provide so many builtin features for API development.
- **Docker and Docker Compose:** Docker with Docker Compose helped us easily configure our production environment.
- **exchangeratesapi.io:** The API that we used was very straightforward to work with. But it only provides data for foreign exchange rates and the data is updated daily. We will use a more advanced API next semester.
- **Github (PRs and Issues):** We think we utilized Github pull requests and issues very well throughout the development period of the project. We tried to use Github's features for all our reviews and comments. We have created issue and pull request templates to better define our cycle of development. We have opened a new branch for each feature implemented and created a pull request.
- **jQuery:** We developed our frontend using jquery. Though a bit old, jQuery is still a very powerful library for making user interfaces. But we think we should use a more sophisticated javascript framework next semester such as React or Vue.js.
- **JWT:** JSON Web Tokens makes it easy to create secure web tokens without worrying about storing the tokens in your database. We used JWT to authorize users on our system when they create or delete manual investments.
- **Nginx:** We use nginx both as a proxy and a static content server. It stands as a proxy between the backend and clients to decipher SSL encrypted requests. It serves the static content of the frontend.
- **PyCharm:** Pycharm made our lifes easy with its builtin features. We used it for PEP8 format checking for Python code. It is also very helpful for resolving merge conflicts.
- **Travis:** We utilized Travis to automatically test the pull requests. It helped us maintain a smooth codebase by avoiding broken branches to get merged into

master. We also plan to use it for continious integration next semester.
- **PostgresSQL:** PostgresSQL is a well documented and widely used open source DBMS. It works well with Django.

Work Done by Each Member

Team Member	Contribution / Work Done
Adil Numan Çelik	• I implemented list investments endpoint.
Buse Gilederesi	• I registered to AWS and created an EC2 instance. • I collected SSH keys from team members and added them to the machine. • I implemented the API endpoint for list of parities, it was later reviewed and changed. • I implemented the test for total profit/loss calculation and test for individual profit/loss calculation with invalid symbol. • I designed the use case diagram for the API. • I created a user manual for the API. • I reviewed the test for individual profit/loss calculation and gave feedback. • I reviewed the test for adding and deleting investments and gave feedback. • I reviewed the PR template.
Fatih İver	• I implemented the cronjob functionality to get exchange rates periodically from a 3rd party API. • I prepared the class diagram for the practice app. • I wrote the brief description of the app and the endpoints we provide. • I contributed to the preparetion of the project plan. • I wrote a unit test for the "create investment" API endpoint. • I reviewed the issue template and the profit calculation end point.

Team Member	**Contribution / Work Done**
Mert Yüksekgönül	• I have created pull request template and issue template for the workflow. • I have created the travis configuration for autotesting and to be used in CI in the future. • Yunus Emre and I wrote down the system requirements. • I have implemented models(database) for the API. • I have implemented create investment endpoint. • I have implemented delete investment endpoint. • I have implemented the endpoint for calculating an investment's profit. • I have implemented the endpoint for calculating total profit for a user. • I have reimplemented the parity/latest endpoint for enhancing its use, letting handling multiple cases. • I have implemented a unit test for profit calculation. • I have implemented the history generation command, which we run when initializing the db and fetching historical data. • I was the reviewer for 12 PRs.
Muhammet Furkan Gök	• I was responsible for implementing login API with Ozgur Solak. • I implemented some invalid test cases for login API. • I implemented some front-end codes which is not in use for now, will be used later. • I reviewed some parts of the codes of the login API which was written by my partner.
Ozgur Solak	• I implemented the login API with my partner Furkan Gok. •I implemented different test cases for login API.
Yunus Emre İnci	• I implemented the historic parity data API. • I wrote unittests for all of the /parity endpoint. • I improved the speed of the parity list endpoint. • I developed the frontend code. • I implemented a Django management command to generate fake parity data for development. • I deployed the project in the production server by making necessary configurations and settings. • I reviewed all of the code that has been written. • I merged several pull requests by resolving conflicts.

Team Member	Contribution / Work Done
Dilruba Köse	• I implemented the register API. •I wrote different unit tests -valid and invalid- for register API. • I reviewed the test for create investment code which is written by Fatih. • I created sequence diagrams for API design.

API Requirements

Table of Contents

1. System Requirements

1.1 Registration and Authorization

- **1.1.1.** System shall be able to register a user given a username, an email address and a password.
- **1.1.2.** System shall be able to provide an endpoint for a user to sign in with their email and password.

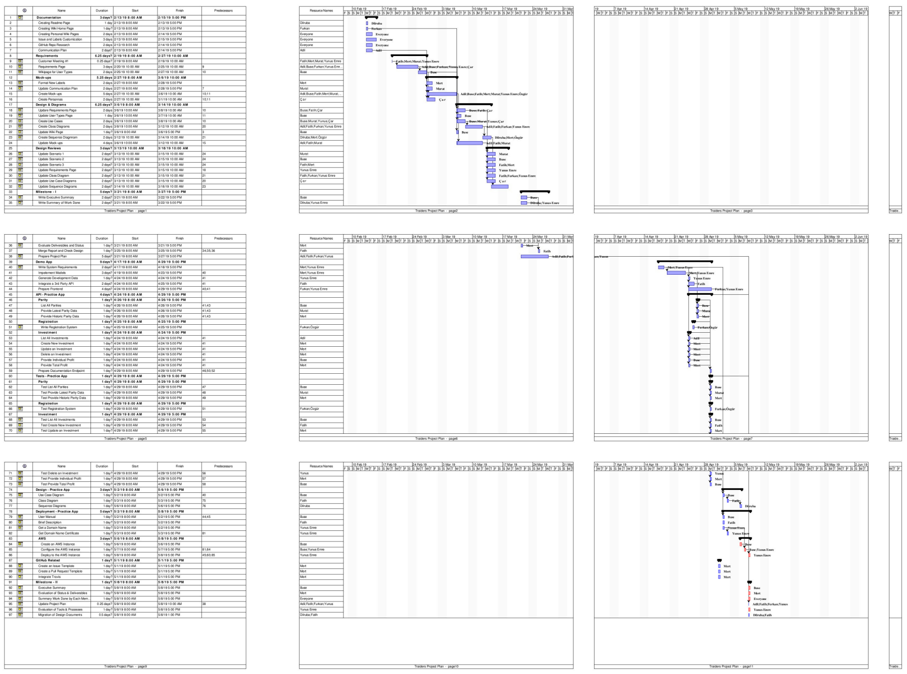
1.2 Parity

- **1.2.1.** System shall be able to provide historic parity data.
- **1.2.2.** System shall be able to provide up-to-date parity data.
- **1.2.3.** System shall be able to provide the list of all of the existing parities in the system.

1.3 Investment

- **1.3.1.** System shall provide a list of all investments for a user.
- **1.3.2.** System shall allow a user to create a new investment.
- **1.3.3.** System shall allow a user to delete an existing investment.
- **1.3.4.** System shall provide a user current profit/loss for a given investment.
- **1.3.5.** System shall provide a user total profit/loss for all investments.

Project Plan



API URL

API Documentation: https://api.traiders-practice.tk/doc/
Frontend: https://traiders-practice.tk/