Ringer – Virtual stock app

ECE-GY-9953-BK30

Group C :

Chia-An Kuo(Net ID: cak580)

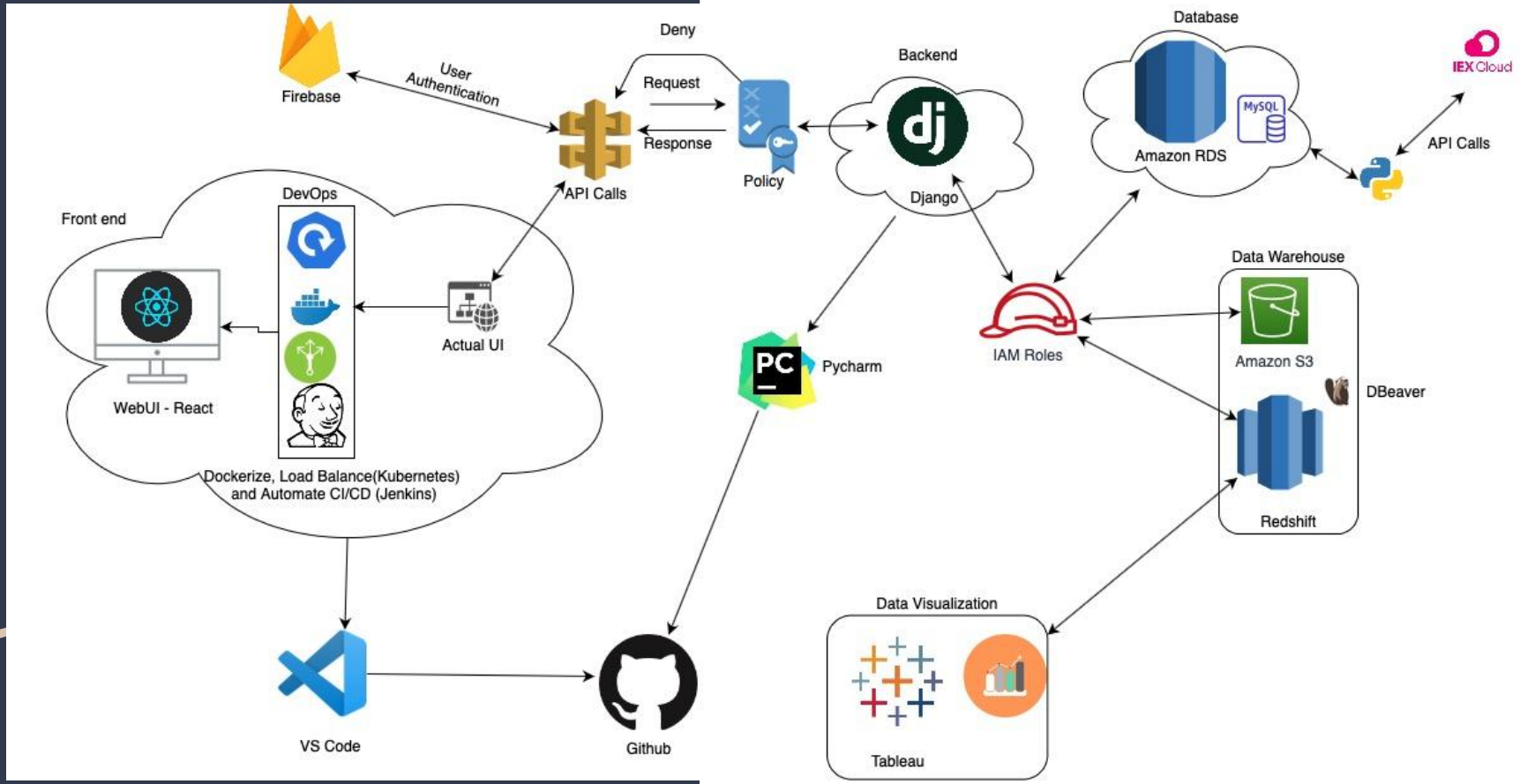
Sai Kiran(Net ID: vc2118)

2021/05/08

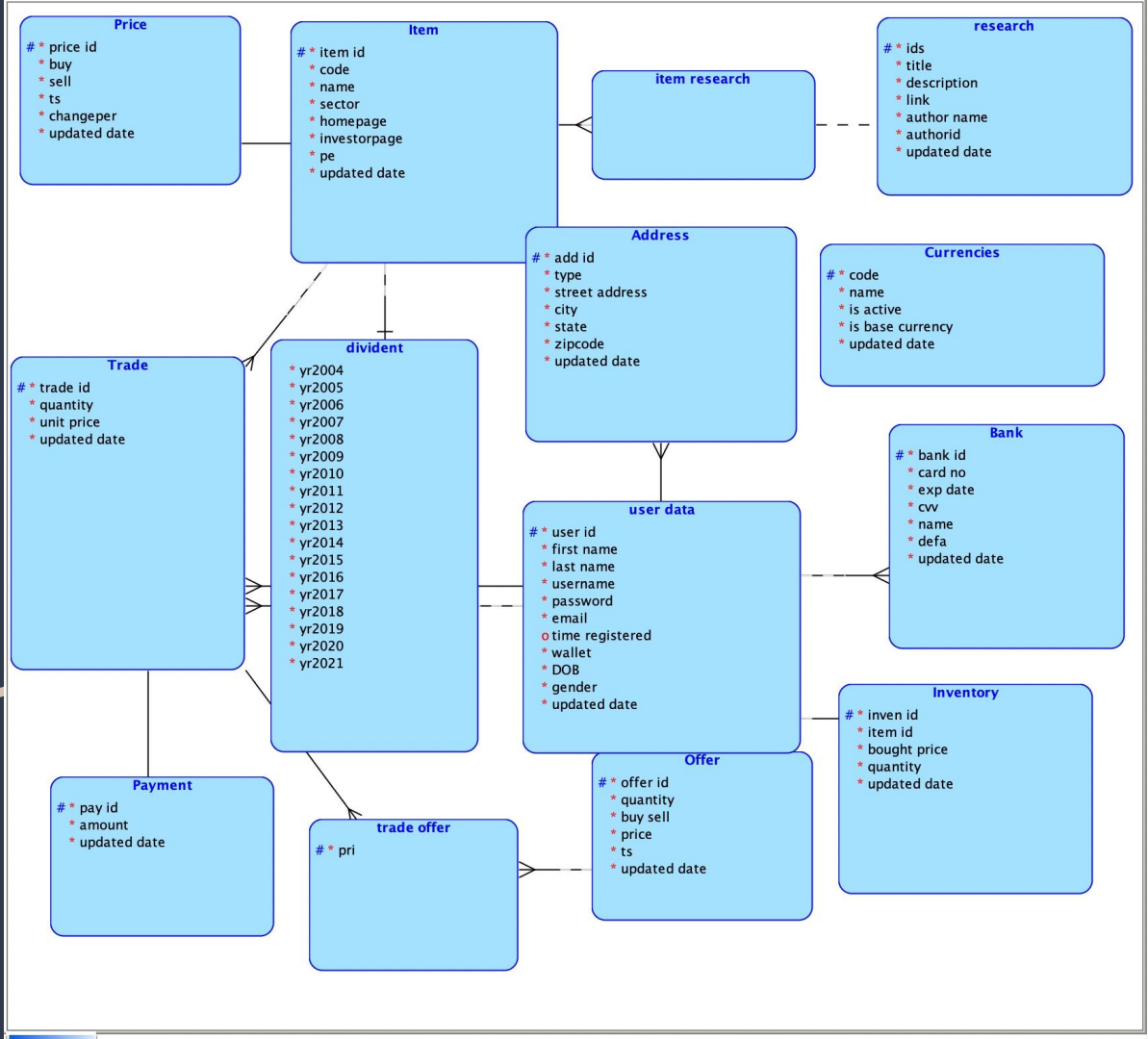
Business Case

This is a virtual stock trading web application made using React.js hook. This application intends to help future investors learn the ups and downs of the stock market and build a portfolio of stocks for themselves. Beginning with a sum of $1000 you get a chance to get accustomed to the buying and selling of stocks, track your progress over a period of time and make yourself ready to conquer the stock market. The app will be containerized, and could be deployed on kubernetes. We are using jenkins for the CI/CD pipeline. Whenever the code is modified, the app will deploy the latest version automatically.

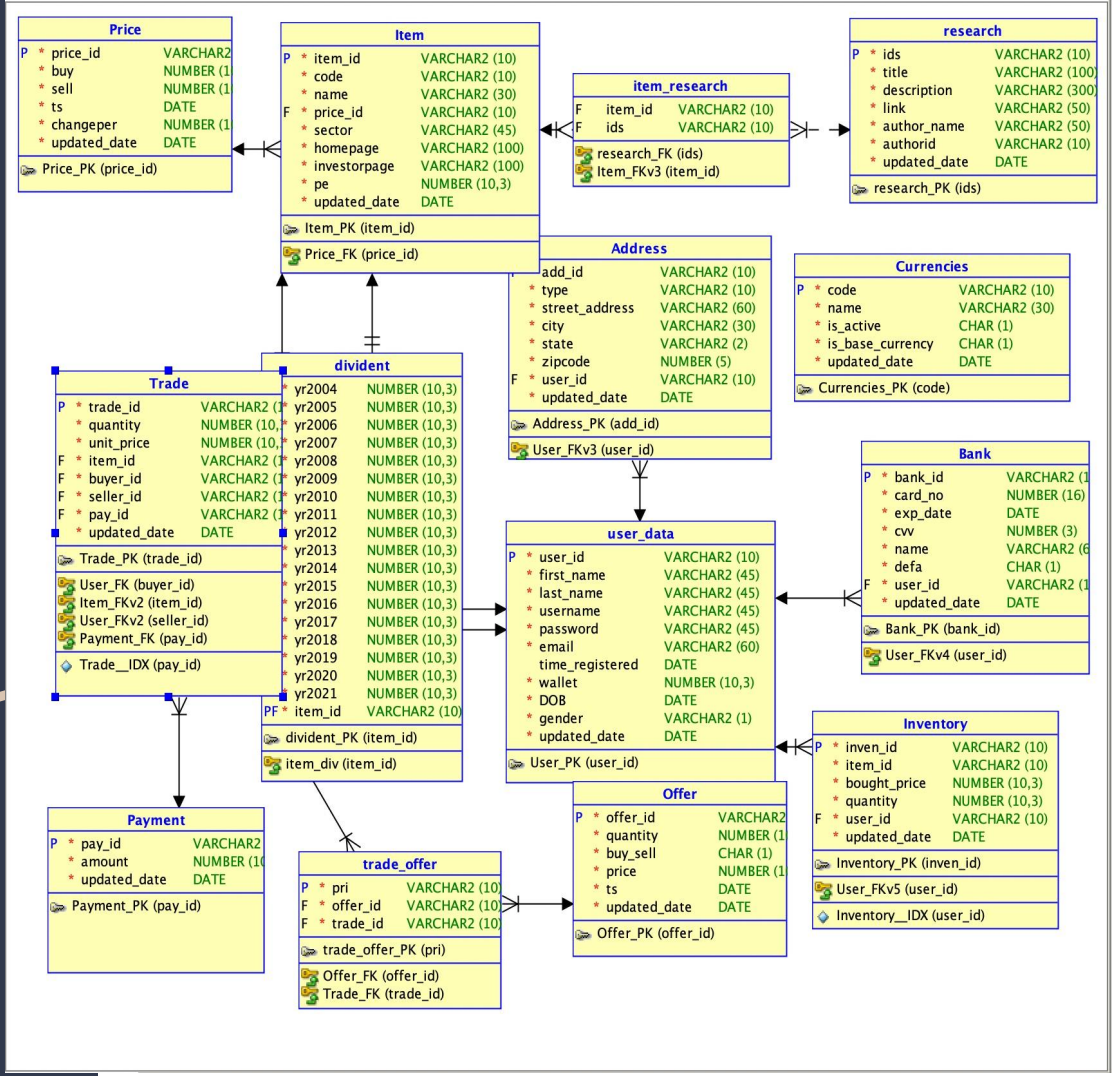
Topology:


Firebase
User Authentication
Deny
Request
Response
API Calls
Policy
Backend
Django
Database
Amazon RDS
MySQL
IEXCloud
API Calls
Front end
DevOps
Actual UI
WebUI - React
Dockerize, Load Balance(Kubernetes) and Automate CI/CD (Jenkins)
Pycharm
IAM Roles
Data Warehouse
Amazon S3
DBeaver
Redshift
Data Visualization
Tableau
VS Code
Github

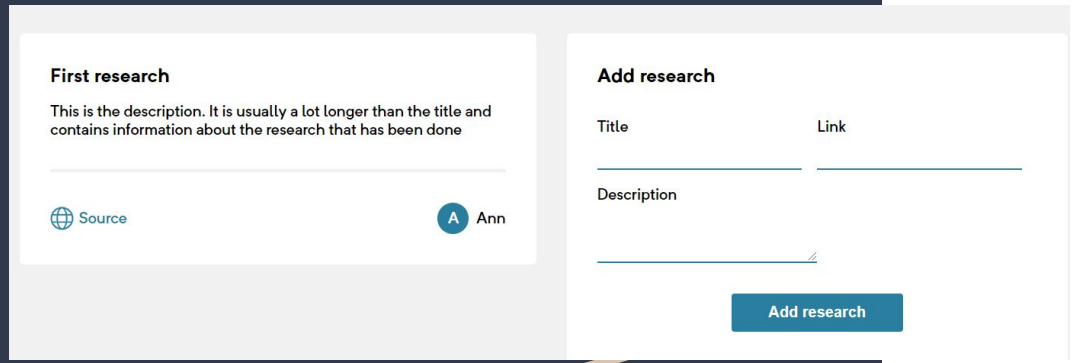
Logical model of OLTP

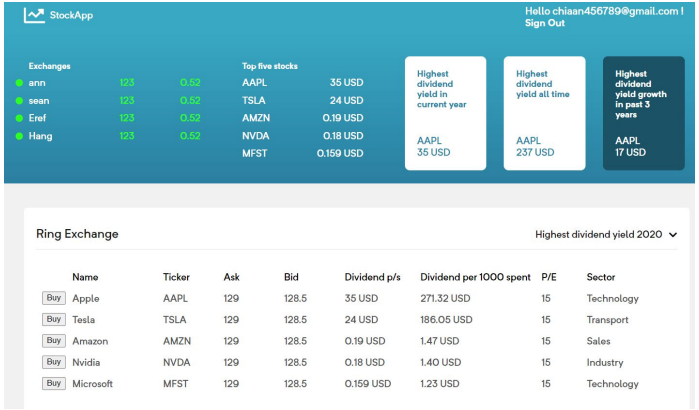


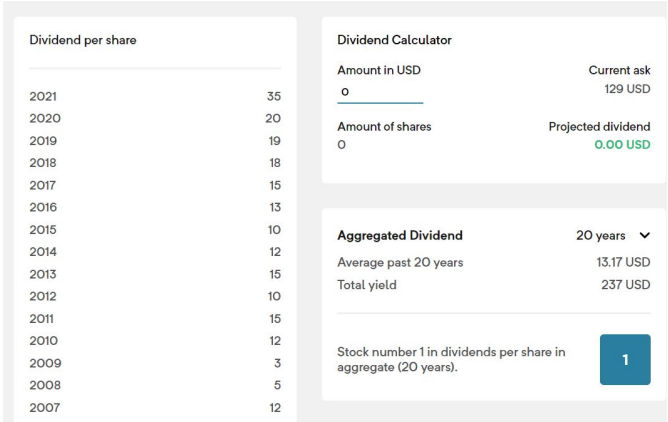
Relational model of OLTP



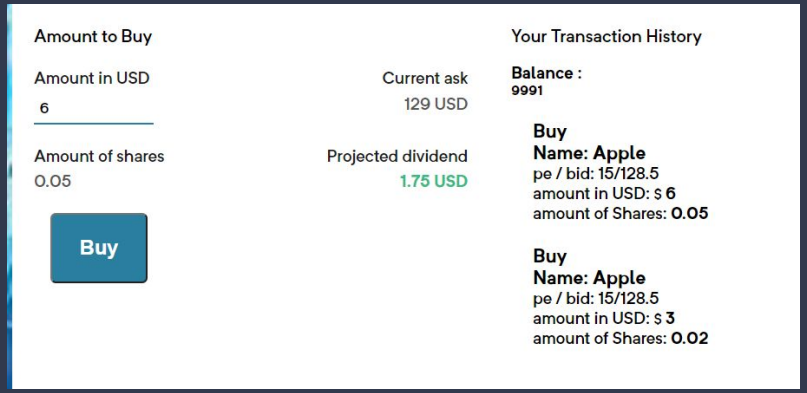
Web application

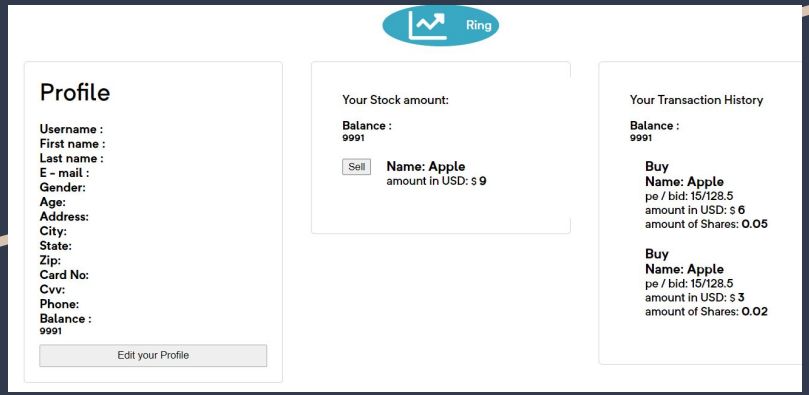


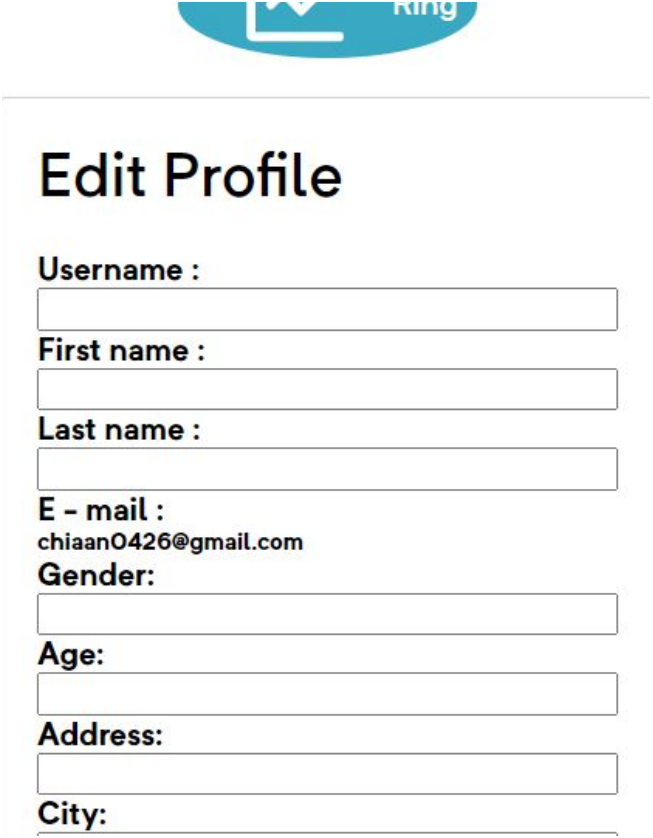




Web application

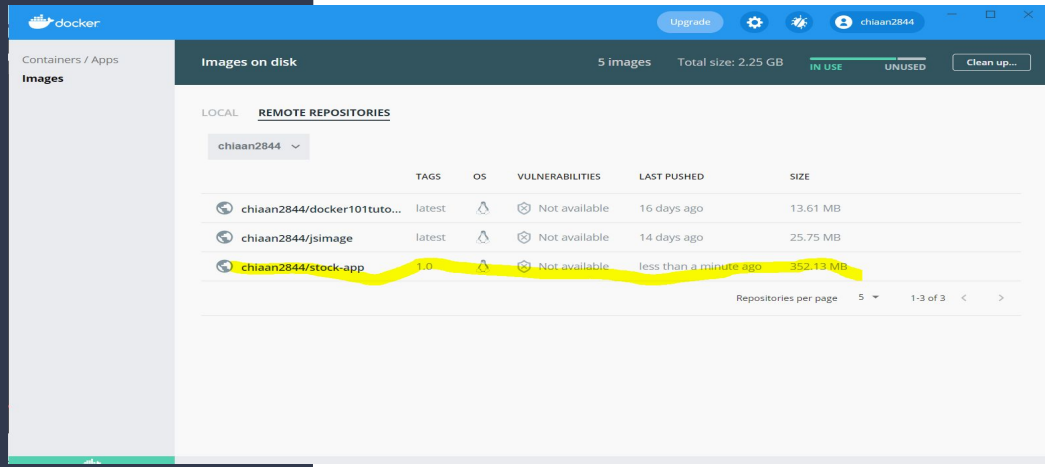


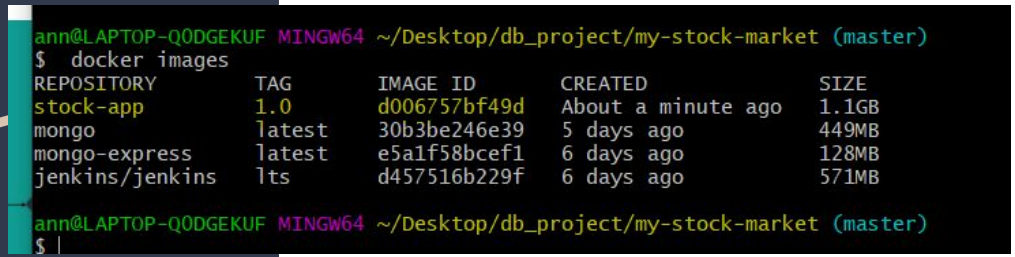




Docker

```
# pull official base image
FROM node:13.12.0-alpine

# set working directory
WORKDIR /app

# add `/app/node_modules/.bin` to $PATH
ENV PATH /app/node_modules/.bin:$PATH

# install app dependencies
COPY package.json ./
COPY package-lock.json ./
RUN npm install --silent
RUN npm install react-scripts@3.4.1 -g --silent
RUN npm install react-router-dom --silent
RUN npm install classnames --silent
RUN npm install react-util-kit --silent
RUN npm install --save firebase

# add app
COPY . ./

# start app
CMD ["npm", "start"]
```

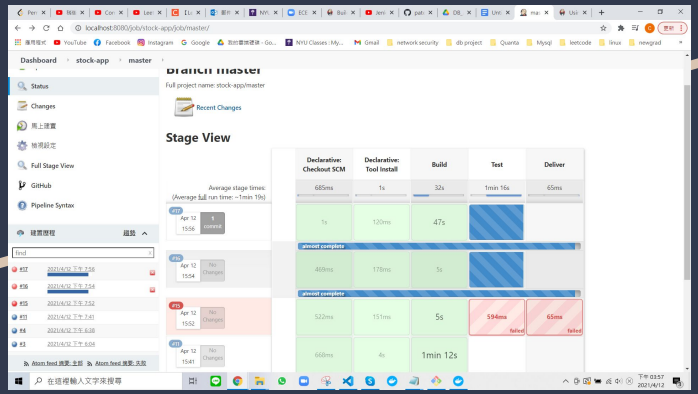




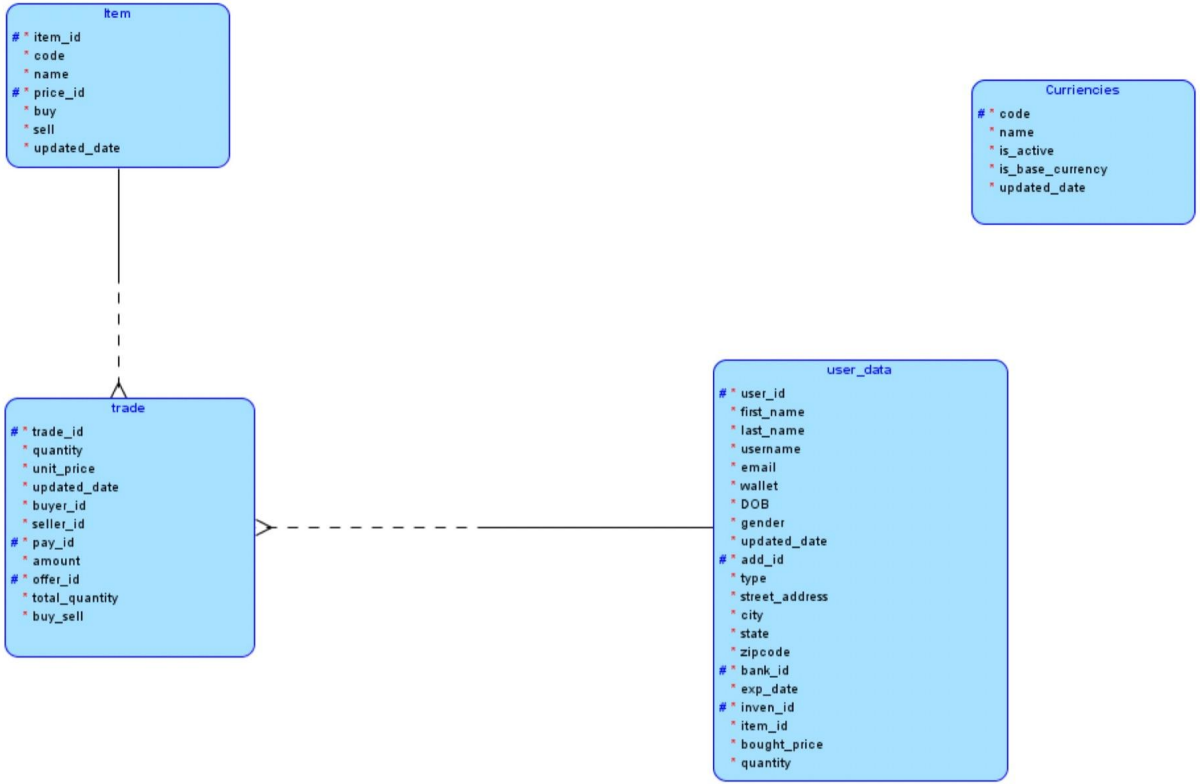
CI/CD pipeline

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: stock-app-service
  labels:
    app: stock-app
spec:
  type: NodePort
  ports:
  - port: 8000
    protocol: TCP
    targetPort: 8000
    nodePort: 32121
  selector:
    app: stock-app
```

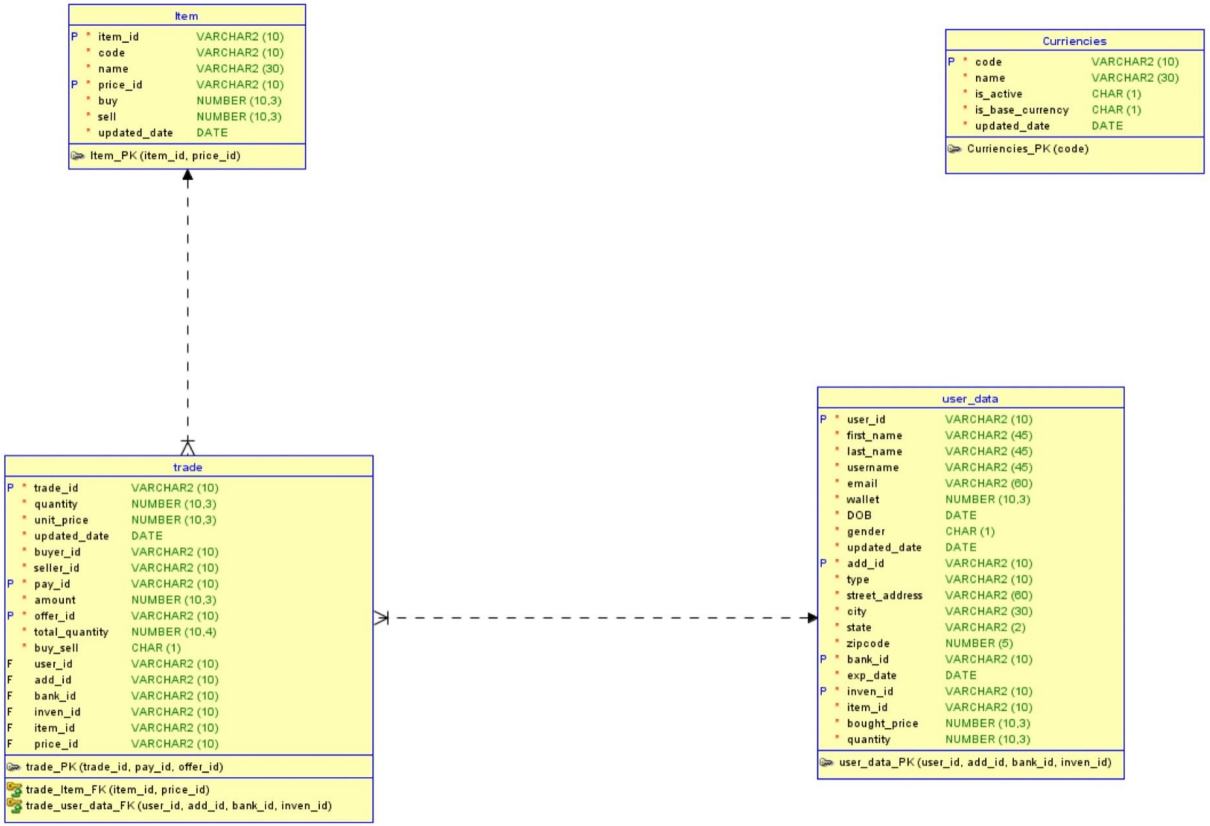
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: stock-app
  labels:
    app: stock-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: stock-app
  template:
    metadata:
      labels:
        app: stock-app
    spec:
      containers:
      - name: stock-app
        image: chiaan2844/stock-app
        ports:
        - containerPort: 8000
---
```



Logical model of DW systems



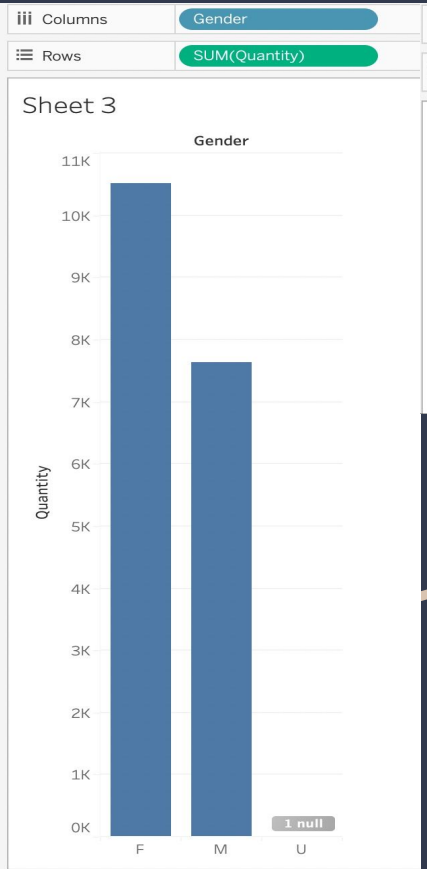
Relational model of DW systems

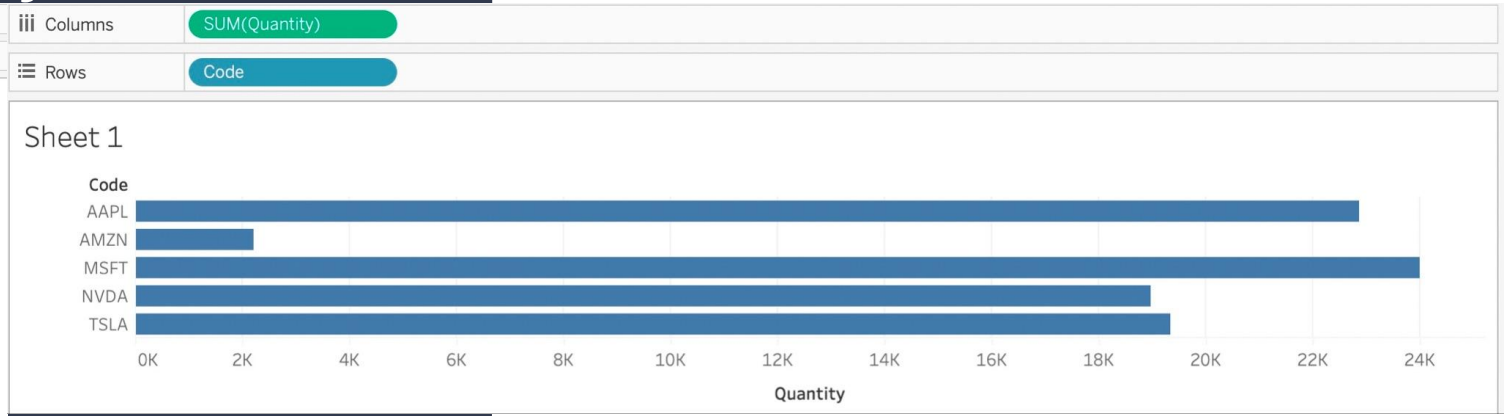


ETL approach

- Extract the columns in the same order as the DW tables by using joins on OLTP and save it as a csv file.
- Copy the CSV file to Amazon S3.
- Truncate the External tables.
- Load the CSV file to the Staging table in Redshift using COPY command.
- For initial load, load the whole data into DW tables.
- For incremental load, run a Procedure to check with the already present rows and, update them if they are modified or, insert the new rows.

Data analytics





Lesson learned

1. Learned a lot about Stock Trading.
2. Implementation and design of Data Warehouse.
3. Creation of History, External and Partition Tables.
4. Implementing ETL, and Visualising using BI tools.
5. AWS - RDS, S3, IAM Roles, Redshift.
6. Technologies like React, Firebase, Docker, Kubernetes, Jenkins (CI/CD), Django, Tableau, DBeaver, etc.
7. Learned a lot about cross platform communication using APIs.
8. Last but not the least, how to manage time, work with teammates and importance of deadlines.

Things didn’t go well:

We spent a lot of time choosing the right DB for the use case, as we spent almost ¾ th of the time working on Mongodb as a backend.