H$G Virtual Portfolio
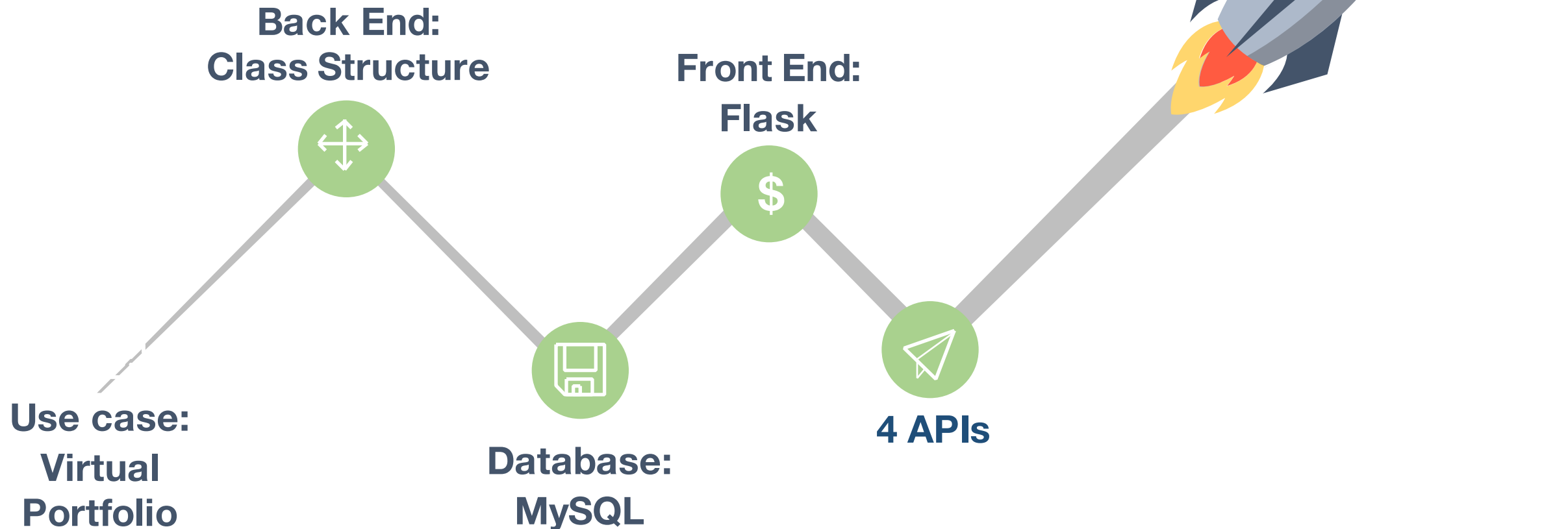
H$G Virtual Portfolio enables students to trade in real time and compare their portfolios to others

**Back End:
Class Structure**

**Front End:
Flask**
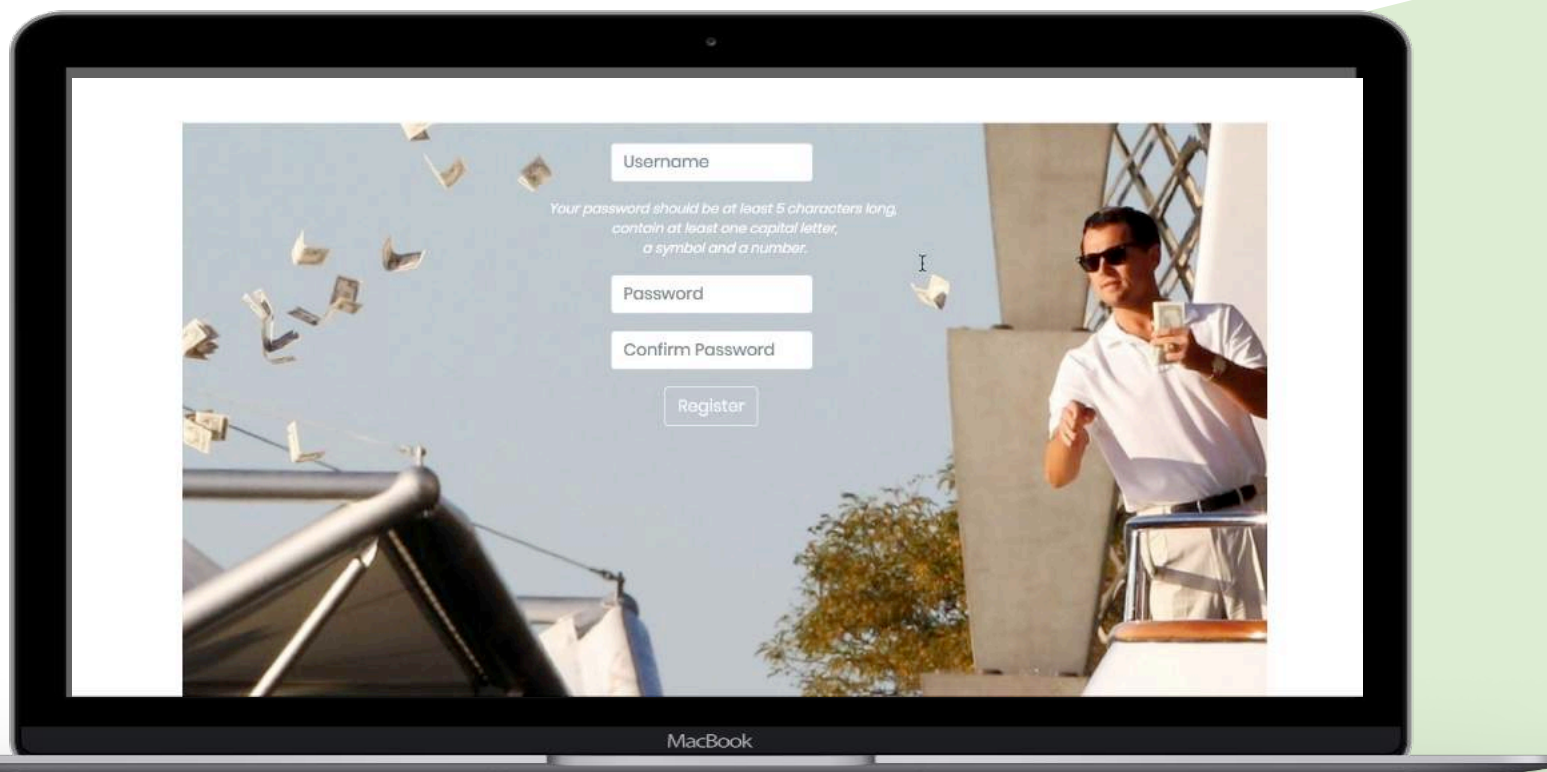
**Use case:
Virtual
Portfolio**

**Database:
MySQL**

4 APIs

Front End –
What
We
Do

Dynamic Multi-Functional Real-Time Trading Platform bound to three APIs

4

Register, Login, Logout

Quote Stock Information & News

Buy & Sell Stocks

Track your Portfolio Performance

Compare to your peers

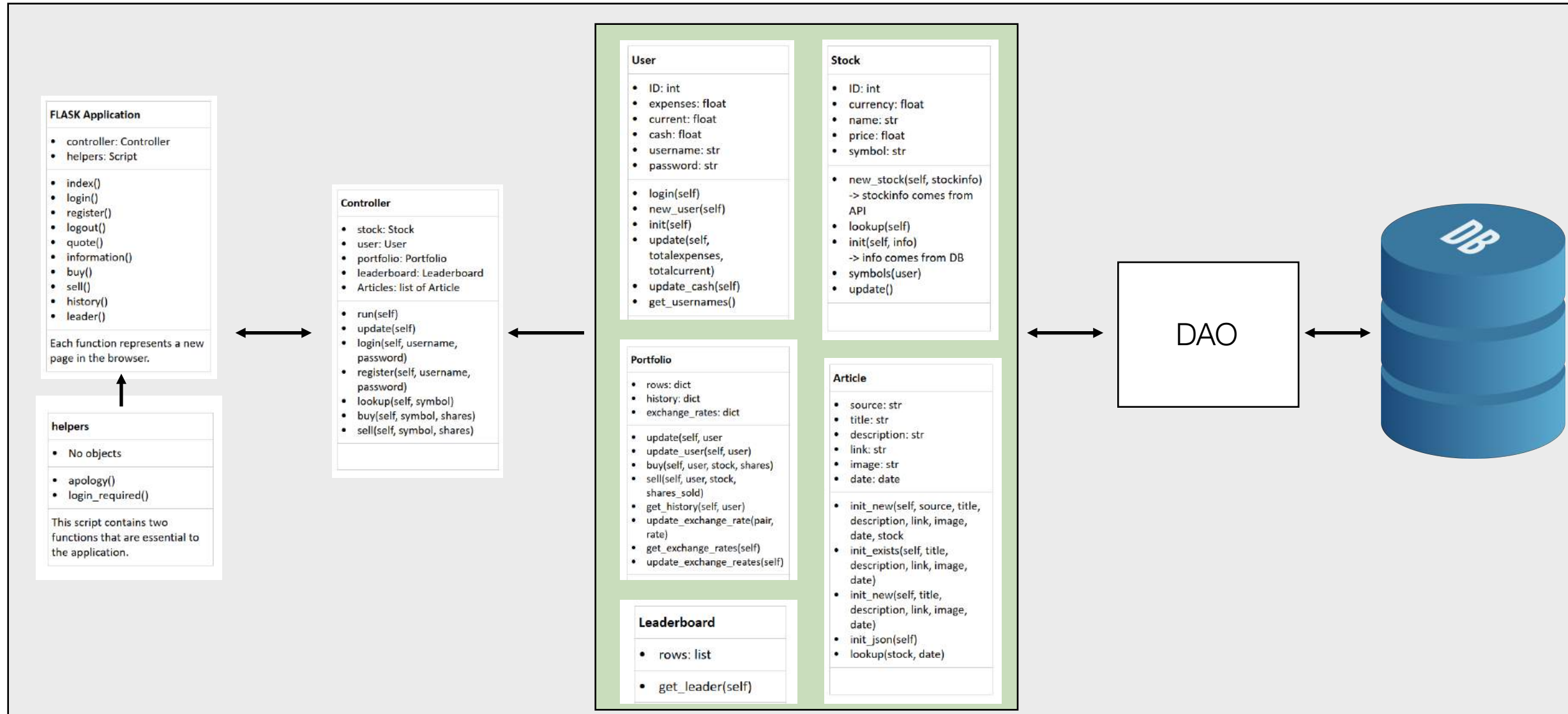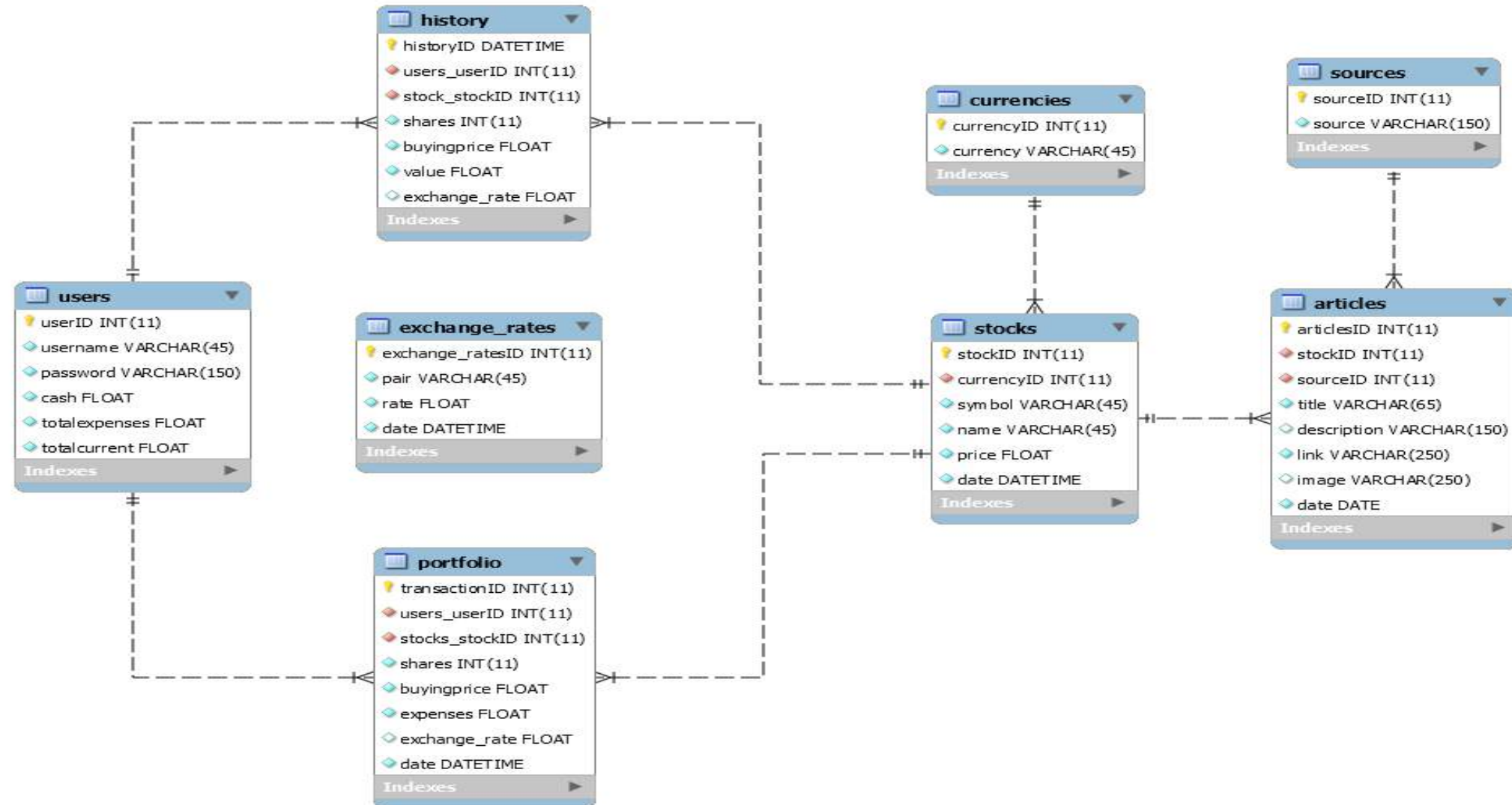# Flask, HTML, CSS, Bootstrap & Jinja

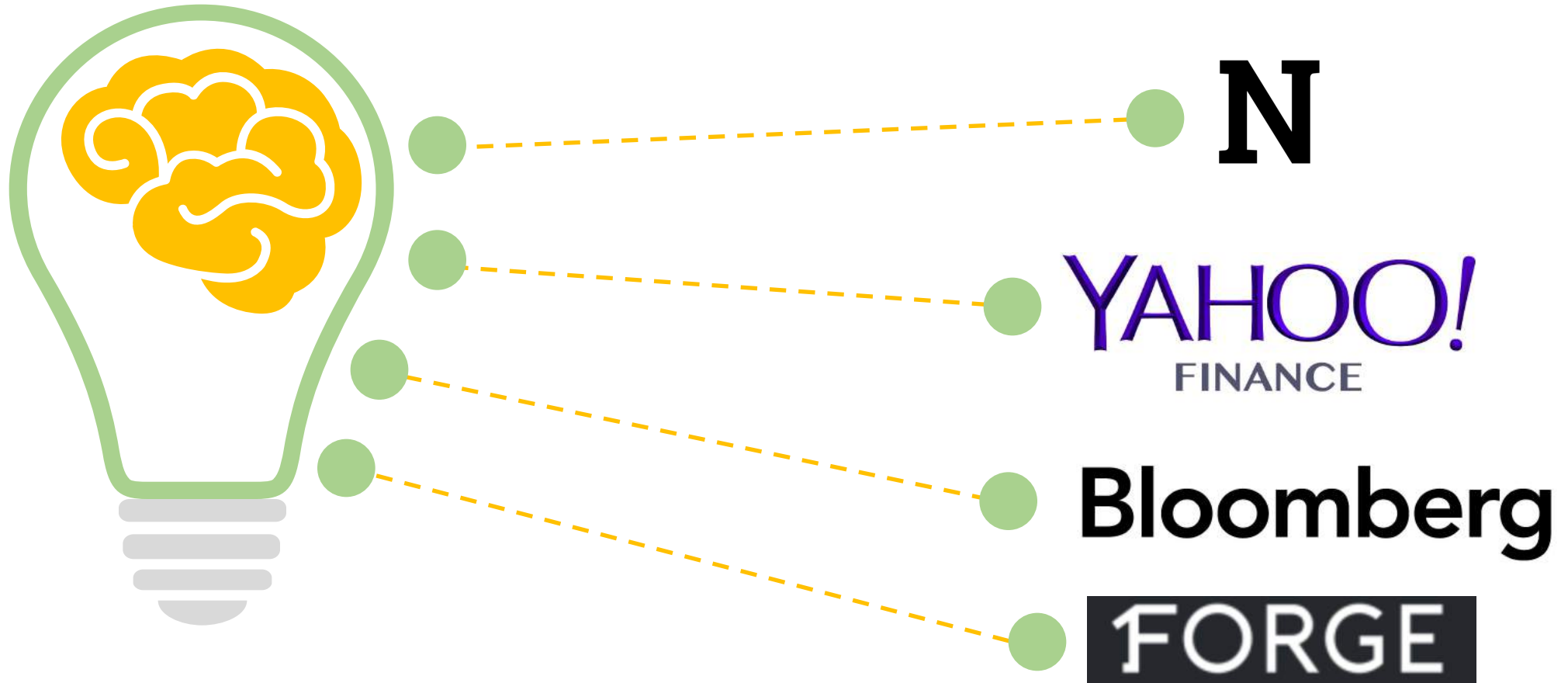Needed an Alternative for Console

Flask to create a website like user experience

Renders HTML templates

Bootstrap – Free Front-End framework

HTML, CSS for design

Jinja – template engine for python programming

Backend

OOP Classes

FLASK Application

- controller: Controller
- helpers: Script

- index()
- login()
- register()
- logout()
- quote()
- information()
- buy()
- sell()
- history()
- leader()

Each function represents a new page in the browser.

helpers

- No objects

- apology()
- login_required()

This script contains two functions that are essential to the application.

Controller

- stock: Stock
- user: User
- portfolio: Portfolio
- leaderboard: Leaderboard
- Articles: list of Article

- run(self)
- update(self)
- login(self, username, password)
- register(self, username, password)
- lookup(self, symbol)
- buy(self, symbol, shares)
- sell(self, symbol, shares)

User

- ID: int
- expenses: float
- current: float
- cash: float
- username: str
- password: str

- login(self)
- new_user(self)
- init(self)
- update(self, totalexpenses, totalcurrent)
- update_cash(self)
- get_usernames()

Stock

- ID: int
- currency: float
- name: str
- price: float
- symbol: str

- new_stock(self, stockinfo) -> stockinfo comes from API
- lookup(self)
- init(self, info) -> info comes from DB
- symbols(user)
- update()

Portfolio

- rows: dict
- history: dict
- exchange_rates: dict

- update(self, user
- update_user(self, user)
- buy(self, user, stock, shares)
- sell(self, user, stock, shares_sold)
- get_history(self, user)
- update_exchange_rate(pair, rate)
- get_exchange_rates(self)
- update_exchange_reates(self)

Article

- source: str
- title: str
- description: str
- link: str
- image: str
- date: date

- init_new(self, source, title, description, link, image, date, stock)
- init_exists(self, title, description, link, image, date)
- init_new(self, title, description, link, image, date)
- init_json(self)
- lookup(stock, date)

Leaderboard

- rows: list

- get_leader(self)

DAO

DB

Database

APIs

Problems

What we struggled with

No documentation YAHOO API
Trial and Error

Updating DB
Raspberry Pi

Coordinating Workflow
Github Repository

Understanding Flask, HTML, Jinja, CSS
Tutorials, Documentations, Trial and Error

Scalability with increasing data volume
Future: greater server capacity and code efficacy

NOW CHECK IT OUT YOURSELF! ☺