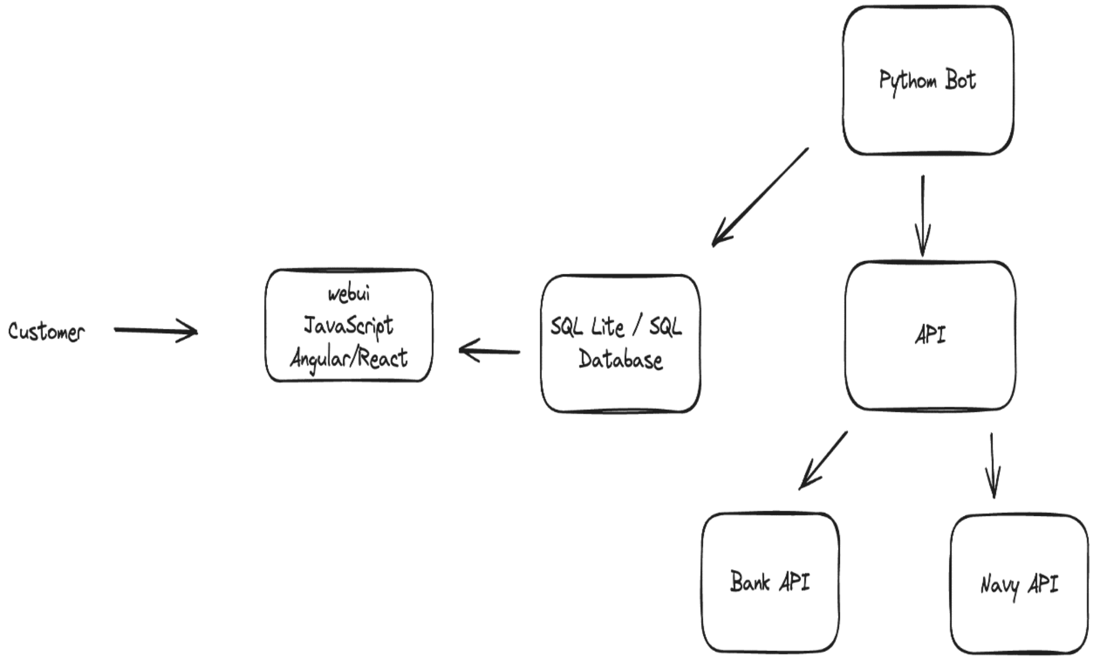
FUSIONBANK

The Universal Banking Application

Naseem Brady - Kush Desai - Zahir Humphries - John Pluznyk

Chapter 1 Team Vision:

Our vision is to create a universal banking application that will allow the consumer to interact with several of their bank accounts through one app. Building upon this foundation, we are dedicated to integrating advanced Python bot functionality alongside the bank APIs and SQL Lite technology. This holistic approach will enable users to seamlessly manage multiple accounts, access comprehensive financial insights, and execute transactions with ease. By consolidating disparate banking experiences into a unified platform, we aspire to streamline financial management and enhance user convenience. Our overarching goal is to redefine the banking experience, fostering greater efficiency, security, and accessibility for all users. This vision is a simple one but can positively impact many people.



Chapter 2 Team Proposal:

Team:

We have divided the project into distinct components, with each team member tasked with specific responsibilities to ensure its completion:

Naseem: Responsible for handling SQL Database management and API calls.

Zahir: Responsible Developing the Python Bot functionality.

John: Responsible for Designing and implementing the Web UI.

Kush: Responsible for Developing the Bank System and API.

Vision:

Making our vision come to light our team will focus on four aspects for this project a web UI, SQL database, python bot, and our own banking system. The web Ui will allow consumers to interact with our system directly, that will allow them to seamlessly check the balances of several of their bank accounts. Our python bot should be up and running 24/7. This allows our bot to continuously retrieve updated data from user's accounts and store data within our SQL database.

The web UI will directly interact with our database through a backend server. When users access the web UI and perform actions such as viewing account balances or transaction histories, the UI will send requests to the backend server. The backend server will then process these requests, retrieve the relevant data from the database (which stores user account information, transaction history, etc.), and send the requested information back to the web UI to be displayed to the user. Creating the web UI, we will use JavaScript & HTML and CSS to create a user-friendly website. We will also use node.js which will allow us to directly interact with our SQL database.

Our bot will interact with the database using SQL queries. When users interact with the bot by asking for account balances or transaction details, for example, the bot will parse the user's request and formulate SQL queries to retrieve the necessary information from the database. Once the database returns the requested data, the bot will format it appropriately and present it to the user.

Due to constraints on accessing actual bank APIs, we will develop a custom API tailored specifically to our project's requirements. The development of our own API enables us to circumvent the limitations imposed by the unavailability of direct access to bank APIs. Our custom API will mimic the functionalities of real bank APIs, providing endpoints for retrieving account information, transaction history, and initiating transactions. By creating our own API, we retain full control over the integration process and can ensure compatibility with our bot's functionality. This approach allows us to maintain security standards and data privacy while delivering a seamless user experience.

The API will interact with our banking system by establishing secure connections to the banks' servers. When a user requests information or initiates a transaction through our bot, the bot will send requests to the respective banking APIs. These requests will include authentication credentials and specific parameters related to the user's request. The banking APIs will then process these requests, verify the user's identity and permissions, and interact with the banks' systems to retrieve the requested information or perform the requested actions. The API will then return the results of these interactions (such as account balances or transaction confirmations) back to our bot, which will present them to the user.

Deployment:

The computing resources that our group will be utilizing will be provided to us by Cloud Lab. The service that Clouds Lab offers to us as software developers is IaaS (Infrastructure-as-a-Service). Cloud Lab provides us the control and visibility all the way down to bare metal. This will let us configure our environments as needed to develop our application. By creating customized python configuration files, located within our teams GitHub repo, we will be able to ask Cloud Lab for the computing resources needed for application. This configuration file will also allow us to install software such as Kubernetes and Docker, giving us the ability to configure our components as needed.

To deploy our application to Cloud Lab's computing resources we will be using Kubernetes. Kubernetes is an open-source container orchestration platform used for automating the deployment, scaling, and management of containerized applications. Containers are lightweight, portable, and consistent environments that package applications and their dependencies, allowing them to run consistently across different computing environments. Kubernetes enables us to create and manage a cluster of the these running containers, which will be seamlessly orchestrated, scaled, and monitored to ensure deployment, optimal resource utilization, and high availability for our applications.

The containerization software that our group will be using to make our components is Docker. Docker is an open platform for developing, shipping, and running applications. Docker will enable us to separate each of our applications from our infrastructure. By using Docker, we will be able to develop each of our components without the headache of having to worry about the details of our infrastructure. Docker also empowers us to use the bare minimum resources needed to create our application. Lastly, Docker also enables us to pack all the necessary components for our application into our own custom Docker image. From there we will be able to deploy and manage our components using Kubernetes.

Chapter 3 Current Progress:

In order to construct our custom docker images our team utilized ChatGPT. By communicating with ChatGPT we were able to create our own custom docker images. ChatGPT was used as an assistant to help us resolve any issues or errors we ran into during the creation of our custom docker files. We also found that, yes, you can google this information and use the docker website to help build your custom docker files, but we found that using ChatGPT provided us with the same results if not better. Using ChatGPT allowed us to cut down on the research time we needed to do to build our custom docker files.

The MySQL server is by far one of the most crucial parts within our project. The MySQL server will create the database and necessary tables needed for our application to store our data. The docker file needed for the creation of our MySQL docker image first pulls the latest version of MySQL from docker hub. After pulling the latest version of MySQL server it sets the root password to a customized password and sets up a MYSQL user which allows us to connect to the server from outside the local network. Next, the image creates a database called FusionBank which is the database that will be utilized to store all of our application's info. From there we grant the privileges necessary to the user to access the databases and then copy a file called init.sql from the same directory. The init.sql files is needed to set up the necessary tables within our database. Our web application requires three tables. One that will store the login data for our application. Another table will store the necessary banking information needed to connect the bank account to FusionBank and lastly account information from the connected bank accounts. The file needed to create these tables within our MySQL server is located within the init.SQL file which contains the commands necessary needed to create our tables.

To create our web server, we utilized the Nodejs docker image. Nodejs can act as a web server that connects to our MySQL server and serves our front end to the user. When configuring the Dockerfile our Nodejs server it was fairly simple. We simply had to set the workdir to the following path "/usr/src/app" this allowed to store the files we need within the directory that Node looks for when starting the server. The next command is "COPY package*.json ./ this copies all of the necessary .json files within the directory of the Dockerfile. Next we run the commands "RUN apt update" and "RUN npm install" which allows us to install npm (Node Package Manager). Next we run the command "COPY .." which copies the rest of the needed javascript, html, and css files needed to setup our webserver and front end. Lastly we expose the port 3000 and run the application with the commands "EXPOSE 3000" and "CMD ["node", "app.js"].

Currently we have tested our MySQL server Dockerfile and Nodejs Dockerfile locally. The MySQL server Dockerfile is able to configure correctly and setup a MySQL server that our Nodejs webserver can connect to. A docker-compose.images.yml file is currently under construction. At this moment it is only setup for the MySQL server and Nodejs server. In the future the docker-compose.imgae.yml file will automatically build all the necessary docker images needed for FusionBank. A service and deployment file were made in order to allow outside users to connect to our project. Currently our python bot and banking service is still underconstruction.

Zahir Humphries: Some of the struggles I faced was setting up the container within Docker, then currently now integrating my files inside the cloud. Figure out how to incorporate the API calls within the python bot was interesting then running Linux commands to make sure everything was up, and running was a headache. But overall I am enjoying facing these technical issues since this is a simulation of what to expect within the real workspace. Implementing my Python bot caused me some issues.

In response to the hurdles presented by the unavailability of authentic bank APIs, our team is diligently working on developing a replica banking API solution. Harnessing Python, JSON, and Fast API, we're constructing a simplified yet comprehensive suite of banking functionalities. Throughout the development process, we've encountered various challenges, and to overcome these obstacles, we're actively tapping into online resources.

Chapter 4 Final Results:

After working on the project for the past three months, our team came up with a working demo. In order to construct the final results of our project we utilize a MySQL server, NodeJS server, and our own banking server. The MySQL server stores the user's login information and the account balance of the account they wish to pull from on our baking server. The NodeJS server first serves the user a login/sign-up page. Here the user can sign up for our open banking platform and then login. Once they login they will then be met with a home/landing page. The home page has a button that will allow the user to enter in the bank URL, user id, and password for the account they wish to connect. If entered correctly they will be able to see the user balance from the bank account, they have just entered. The banking server was done using python FAST API and just contains the barebones for a banking server that we can pull data from.

One of the biggest challenges that our group faced when working on this project was understanding docker and the need for containerizing software. At first the idea of containerizing software was confusing but was soon made clear as we furthered progressed into the semester and started working with Cloud Lab. By using docker within our Cloud Lab experiments we were able to easily create the necessary environments needed for our software to run on.

Another challenge that our group faced was communication with one and other. At first none of us really had a clear picture on what we should do for our cloud application and waited till the last minute to come up with this idea. We initially thought that this project would be much simpler and less tedious than it had turned out to be. As the semester progressed and lack of communication continued none of really raised any concerns with one and other as each of us continued our part. This lack of communication ultimately led to the failure of trying to create a python bot for our cloud application.

The most notable milestone that our group missed is not being able to implement the python bot that our group had promised. At first, we believed that we could just have a simple python bot that would fetch the data our user submitted on the home/landing page of our web app. But as our time ran low, we realized that there were too many issues that we were facing with this bot and decided it best to put the production of our bot on hold. Instead, we were able to compromise with some client-side

JavaScript code that updates the home/landing page with the data our user requests. However, this version is not ideal as it does not meet the goals we had set for this project.

Chatper 5 Conclusion:

To conclude, the creation of Fusion Bank has taught us much in both the worlds of cloud computing and software engineering. By utilizing docker our team was able to package the necessary software needed for Fusion Bank. Our team was able to come up with the needed docker-compose files and docker files that allowed us to build each component of our application. Then by utilizing Kubernetes our team was able create deployment files and a service file that allowed for us to properly deploy our web app on Kubernetes.

From the software engineering aspect of this project each of us got to choose individual components that we worked on. The real learning curve here was learning how to integrate each of these components within our web application. Thankfully by utilizing online resources and documentation each of us were able to integrate our parts within one and another to get a working demo. This experience not only increased our technical skills but also emphasized the importance of collaboration and effective communication in software development projects.

Intentionally Blank

pluznyk02@gmail.com John Pluznyk

EDUCATION

West Chester University, West Chester, PA
Bachelor of Science, Computer Science (August 2020 – present; graduation May 2024)

- GPA: 3.6
- Member of the Computer Science Honor Society

WORK EXPERIENCE

West Chester University, West Chester, PA
Computer Science 2 Tutor (January 2022 – May 2022)

- Reinforced the fundamentals of coding and how to form simple programs.
- Communicated with students and assisted them in developing a learning plan.
- Helped students by listening to their struggles and generating solutions.

Mary Barness Tennis & Swim Club, Warrington, PA
Lifeguard, Swim Instructor (May 2016 – August 2020, seasonal)

- Monitored the wellbeing of guests and maintained a clean and safe pool.
- Taught patrons how to improve their swimming skills and how to be safe in water.

CERTIFICATES

Computer Security Certificate – From West Chester University

PROJECTS

MySQL JAVA GUI – This project is a simple GUI (Graphical User Interface) that allows the users to select the data table of their choosing from a database. I used Java's JFrame GUI interface to allow the user to select a table from a database. Upon selection of table, the entire table will be displayed. https://github.com/JohnPluznyk/JavaMySQL-GUI
Shell Interface C – For this project I created a simple shell interface for terminal users in Linux. https://github.com/JohnPluznyk/LinuxShell
Java Compiler (Back End) – For this project I used JavaCC as a parser generator and lexical analyzer generator to create my own Java like language. I then created a semantic analyzer to check for syntactical errors and make sure programs adhere to the intended semantics of my language. Lastly, after the Abstract Syntax Tree (AST) was created, I created target code generator that translated the AST in MIPS assembly.

CODING COURSEWORK

Design/Construction Compiler – Covers the basic topics in compiler design including lexical analysis, syntax analysis, error handling, symbol tables, intermediate code generation, and some optimization.
Data Base Management Systems – Studied characteristics of generalized database management systems and learned about the fundamentals of designing and implementing of a database system.
Software Engineering - Focused on more advanced topics in object-oriented programming, including project design, planning, and testing.
Data Structures and algorithms – Using object-oriented languages to explore data abstraction, recursion, lists, stacks, queues, linked lists, trees, hashing, and evaluating algorithm efficiency.
Operating Systems - Surveyed elements of an operating system with in-depth studies of certain features of specific operating systems (Linux). Focused on topics such as process scheduling and deadlock avoidance, memory management, and organization and protection of a file system.
Digital Image Processing – Focused on the fundamental concepts about the visualization of various data in the disciples of digital image processing, computer graphics, photometric processing, and image analysis.
Computer Systems – Exploring fundamental concepts of modern CPUs, memory, storage, networking, operating systems.

TECHNICAL SKILLS

Coding Languages: Java, C, MySQL, Python, HTML, CSS
Other: Linux, Docker, Kubernetes, data analytics
GitHub: https://github.com/JohnPluznyk
Linkedin: https://www.linkedin.com/in/john-pluznyk-4399b6225

kdd4203@gmail.com

Kush Desai

Computer Science Major at West Chester University
https://www.linkedin.com/in/kdesai442/

Education:

West Chester University West Chester, Pennsylvania
Bachelor of Science in Computer Science 2021 - 2025

Experience:

Target Target Dec 2019 & Dec 2021 – Jan 2022

- Inventory management and customer service

IS&T Student Intern West Chester University - Aug 2022 – Present

- Resolving various technological issues
- Collaborating with other student interns
- Pick up and recycling various electronic equipment around campus

Certifications:

Responsive Web Design	FreeCodeCamp – July 2020
JavaScript Algorithms and Data Structures	FreeCodeCamp – July 2020
Front End Development Libraries	FreeCodeCamp – October 2021
Data Visualization	FreeCodeCamp – October 2021
JavaScript Coding Fundamentals	Google Grasshopper – November 2021
JavaScript Coding Fundamentals II	Google Grasshopper – December 2021
Scientific Computing with Python	FreeCodeCamp – Currently in Progress

Technical Skills:

Java (Intermediate) HTML (Beginner) Python (Novice)
JavaScript (Beginner) C (Basics)

Involvement:

Computer Science Club 2021 - 2023

- Served as Vice President of the club for the Fall 2022 and Spring 2023 semesters

- Learned about and used various CS related topics outside of the university curriculum

Cyber Security Club 2021 - Present

- Introduced to various elements of cyber security
- Experimented with different applications such as WireShark
- Served as Treasurer of the club for the Fall 2023 and Spring 2024 semesters

NCAE Cyber Games
Feb 2022

- Collaborated as a team to set up, secure, protect, and maintain servers using Linux

Game Development Club 2022 - 2023

- Learned about software such as Unity for creating various types of games
- Created games and applications individually and as groups

Relevant Courses:

Computer Security and Ethics	Data Structures and Algorithms	Cloud Computing
Computer Systems	Foundations of Computer Science	Computer Language Concepts/Paradigms

Projects:

Unity Projects using C# Scripting

- Flappy Bird Clone
- Notepad Application

Interactive Discord Bot (Using Java and the Discord API) (In Progress)

- Typical Discord Bot with various messaging features and allowing user interaction

Python Projects (Using Python Library Imports)

- Snake Game
- YouTube Video Downloader (to MP4)
- Personal Password Manager with Encryption
- Slot Machine Casino Game

Naseem Brady

Comp. Sci. Major at West Chester University | Philadelphia, PA | (267) 581-3446 | nasbrady@gmail.com

OBJECTIVE

To complete this project.

EDUCATION

West Chester University, West Chester, PA
Bachelor of Science, Computer Science (August 2020 – present; graduation May 2024)

- GPA: 3.1

WORK EXPERIENCE

West Chester, Pa
1. The Giant Company – Produce Associate (Oct 2021 – June 2022)

PROJECTS

Text-Based Adventure-RPG "Unnamed": A project in development, made with Java, I started this project to showcase the skills, techniques, and concepts I've learned about programming after being in school for a couple years.

CODING COURSEWORK

Software Engineering - Focused on more advanced topics in object-oriented programming, including project design, planning, and testing.
Data Structures and algorithms – Using object-oriented languages to explore data abstraction, recursion, lists, stacks, queues, linked lists, trees, hashing, and evaluating algorithm efficiency.
Digital Image Processing – Focused on the fundamental concepts about the visualization of various data in the disciples of digital image processing, computer graphics, photometric processing, and image analysis.
Computer Systems – Exploring fundamental concepts of modern CPUs, memory, storage, networking, operating systems.

TECHNICAL SKILLS

Coding Languages: Java, Python, C++, HTML, CSS
Other: Linux, Mac, Windows

GitHub: https://github.com/Naseem-Brady

Zahir Humphries

Comp. Sci. Major at West Chester University | Philadelphia, PA | (215) 581-5099 |
zh921063@wcupa.edu

OBJECTIVE

To create a Python bot to pull bank information & display your current balance on the screen.

EDUCATION

West Chester University, West Chester, PA
Bachelor of Science, Computer Science (August 2020 – present; graduation May 2024)

- GPA: 3.3

WORK EXPERIENCE

West Chester, Pa
1. Google – Open-source with Norman Nanley

Flight Tracker, to track the most precious location of flight travel from a point A to point B

CODING COURSEWORK

Software Engineering - Focused on more advanced topics in object-oriented programming, including project design, planning, and testing.
Data Structures and algorithms – Using object-oriented languages to explore data abstraction, recursion, lists, stacks, queues, linked lists, trees, hashing, and evaluating algorithm efficiency.
Computer Science I, II, III – Java OOP programming

TECHNICAL SKILLS

Coding Languages: Java, Python, C++, HTML, CSS
Other: Linux, Mac, Windows
GitHub: https://github.com/ZHumphr