Spotted

Team 4: Sean Whoriskey, David Rockwell, Zachary Smith, Mike Collins, and Brianna Ayala

https://github.com/seanwho88/CSC468Project4.git

Chapter 1

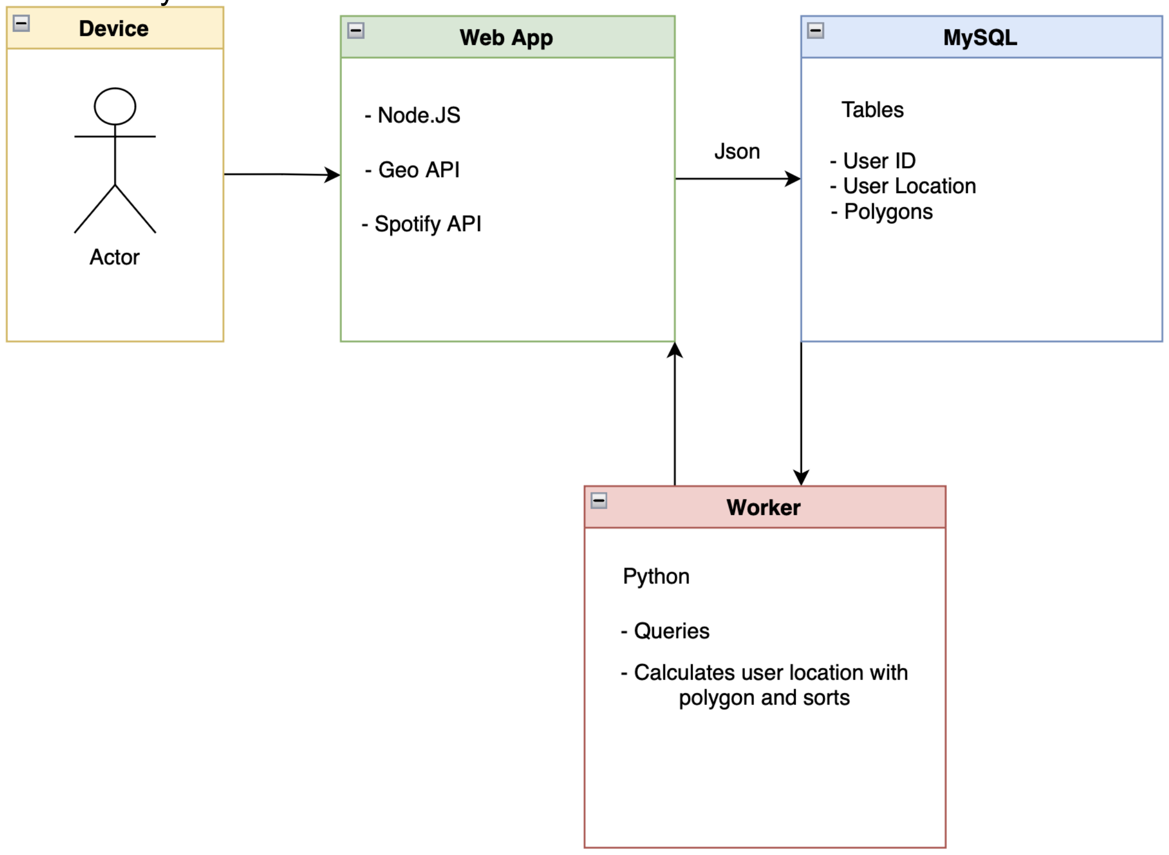
Team Vision/Design

Introduction: The goal of this project is to develop and deploy a cloud-based music sharing web application that allows users to connect with others who share their musical interests. The app will utilize real-time geolocation to find other users in proximity and display the music they are currently listening to.

Features:

- **Connect with Spotify:** Users will connect their Spotify account to the app, which will then display their current playing song for other users to see.
- **Geolocation-based Sharing:** The app will use real-time geolocation to find other users in proximity and display their currently playing song.
- Social Interaction: As a long term goal, users will be able to like, comment, or follow other users, providing a way to socialize through music.
- **Music Discovery:** The app will provide a new way for users to discover new music by browsing other users' currently playing songs.
- **User Interface:** The app will have a simple, intuitive interface, allowing users to quickly connect their Spotify account and start sharing their music.

Conclusion: Spotted (name in progress) is a fun and innovative way to connect with others who share your musical interests. With its real-time geolocation and social interaction features, it provides a new way to discover new music and socialize through the music you love.



Chapter 2

Team Design Implementation

JavaScript and Node.js: JavaScript will be used as the primary programming language for building the front-end user interface and the backend of the app. Node.js will be used as the runtime environment to execute JavaScript on the server-side. This will allow the app to handle multiple concurrent user requests.

Geolocation API: The app will use a Geolocation API to retrieve the real-time location of the user. This information will then be stored in the database so that calculation of user proximity can be calculated by the worker.

Spotify API: The app will use the Spotify Web API to retrieve the current playing song of the user, which will be stored and updated in the database.

MySql: MySql will be used as the database for storing user data, including their user location and current playing song from Spotify.

Python: Python will be used to implement the backend logic for handling user requests and interacting with the Spotify and Geolocation APIs. Python will also be used for any data processing tasks, such as filtering and aggregating data from the MySQL database.

Overall, the implementation of the web application, Spotted (in progress) will use a combination of JavaScript and Node.js for the front-end and back-end, a Geolocation API for retrieving real time location and storing on the database.

To begin we will start small-scale with manually inputting data and test with one set location. We are able to expand into more extensive areas and data as we meet the small goals locally.

Chapter 3

Docker Images

Web App: We created the Dockerfile for the “webapp” component using the node:14-alpine image. The Dockerfile sets up the work directory to /webapp and copies any package.json files. The necessary dependencies for the web app are then installed. The application files are then copied into the container, and the application is exposed on port 3000. We use CMD to run npm start at the end.

We ran into issues regarding our connection to the database through the web app as the database’s container took longer to initialize than the web app. To solve this we added a “depends-on” statement to the “webapp” service in our docker-compose.yaml file. We then added a loop to our web app to continuously attempt connection to the database until everything is initialized. Overall, the procedure for setting up this Dockerfile was relatively simple.

Database: We created the Dockerfile for the “database” component using the official mysql/mysql-server image, which provides our MySQL environment. We set environment variables for the root password, database name, database user, and user password. This information will be put into environment variables in the future for privacy reasons. Finally, we made a custom SQL script (init.sql) that is copied into the /docker-entrypoint-initdb.d/ directory. This script is executed during the container startup.

Overall, the procedure for building this Dockerfile was simple as well. As we worked on designing our database, the init.sql script continuously changed. At this point in the project, the init.sql script builds our users and location tables.

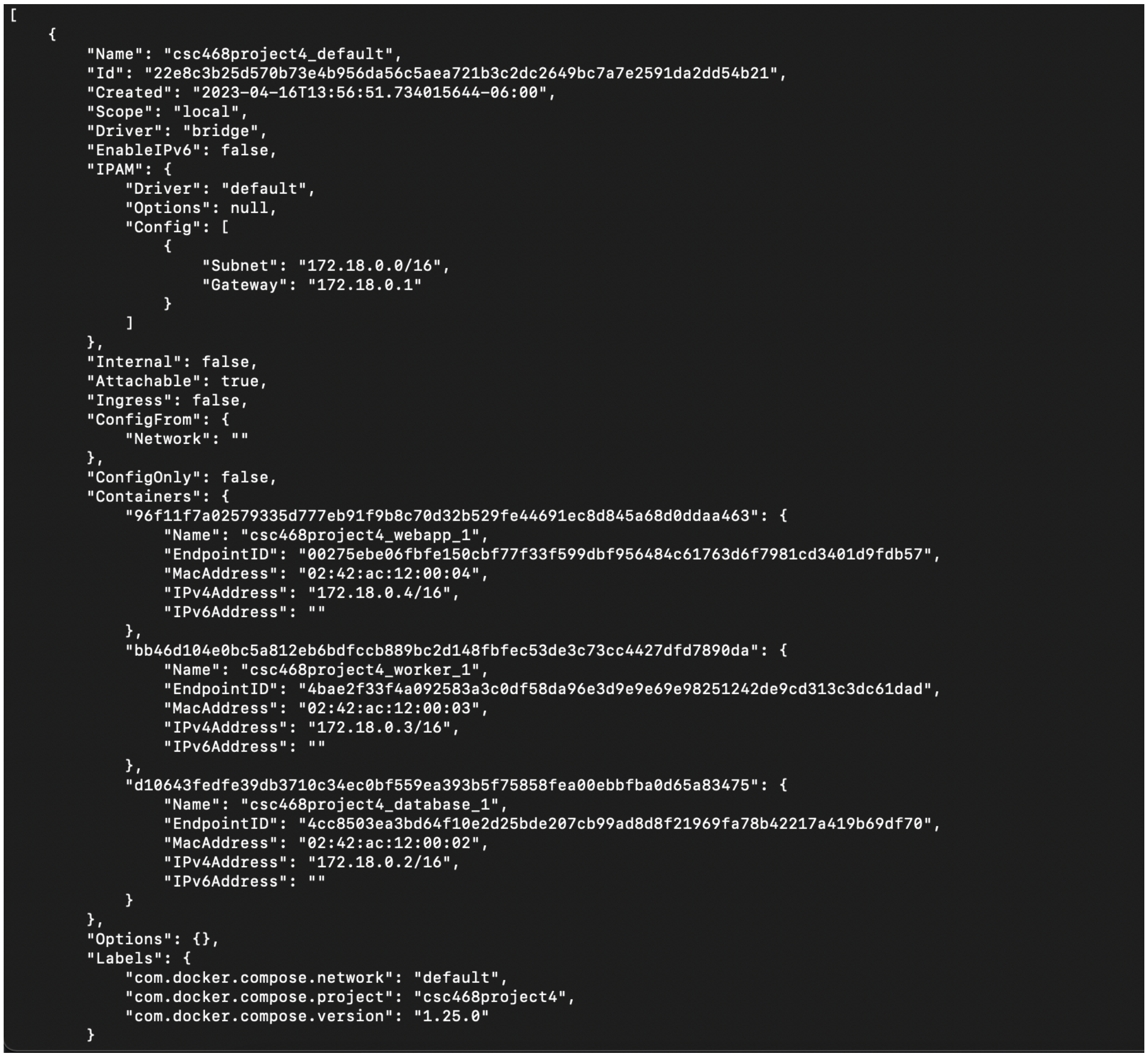
Worker: We created the Dockerfile for our “worker” component using the python:3.9 official image. The working directory is set to /worker, and the requirements.txt file is copied into the container. The required Python packages are installed through that requirements.txt file, and the application files are copied into the container. We expose the worker on port 5000, and then use CMD to start the worker using the run.sh script.

The creation of the “worker” Dockerfile has been the most challenging so far, as we have faced troubles with orchestrating the multiple files that make up our worker component. For example, as new files were added we created a script “run.sh” to start all of the worker scripts when the container is built.

Chapter 4:

Docker Networking

Our docker-compose.yaml file defines our service's exposed ports. We map port 3000 on the host machine to port 3000 in our "webapp" container. We do the same for port 3306 and 5000 for our database and worker respectively. By default, Docker Compose creates a network for the services defined in the compose file. In our case, you can see the network is named "csc468project4_default". Internally, the "webapp" and "worker" services can communicate with the database using the unique ip address for each container and the exposed port 3306. In the image below you can see more details regarding our project's docker network.



```
[
    {
        "Name": "csc468project4_default",
        "Id": "22e8c3b25d570b73e4b956da56c5aea721b3c2dc2649bc7a7e2591da2dd54b21",
        "Created": "2023-04-16T13:56:51.734015644-06:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "96f11f7a02579335d777eb91f9b8c70d32b529fe44691ec8d845a68d0ddaa463": {
                "Name": "csc468project4_webapp_1",
                "EndpointID": "00275ebe06fbfe150cbf77f33f599dbf956484c61763d6f7981cd3401d9fdb57",
                "MacAddress": "02:42:ac:12:00:04",
                "IPv4Address": "172.18.0.4/16",
                "IPv6Address": ""
            },
            "bb46d104e0bc5a812eb6bdfccb889bc2d148fbfec53de3c73cc4427dfd7890da": {
                "Name": "csc468project4_worker_1",
                "EndpointID": "4bae2f33f4a092583a3c0df58da96e3d9e9e69e98251242de9cd313c3dc61dad",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            },
            "d10643fedfe39db3710c34ec0bf559ea393b5f75858fea00ebbfba0d65a83475": {
                "Name": "csc468project4_database_1",
                "EndpointID": "4cc8503ea3bd64f10e2d25bde207cb99ad8d8f21969fa78b42217a419b69df70",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {
            "com.docker.compose.network": "default",
            "com.docker.compose.project": "csc468project4",
            "com.docker.compose.version": "1.25.0"
        }
```

Our attempts to map our components went smoothly for the most part. We had issues with orchestrating the database container's initialization and the webapp's connection to the database. We fixed this by continuously attempting to connect to the database until a connection is finally made. We also ran into issues setting up the hostname for our web app. We fixed this by creating a script called “hostname.sh”. This script is used to change any instance of the word “hostname” or “mysql” to the actual hostname of our machine. This streamlined our ability to instantiate our project on Docker.

David Rockwell

270 Valley Road, Media, PA 19063

davidrockwell7310@gmail.com **|** 484-844-4674

linkedin.com/in/david-rockwell-83b49222b

EDUCATION

West Chester University | West Chester, Pennsylvania | August 2021 - May 2023 (expected)

- Bachelor of Science, Computer Science, Specialization: Cyber Security
- GPA: 3.5

Delaware County Community College | Media, Pennsylvania | August 2019 – May 2021

WORK & INTERNSHIP EXPERIENCE

TA Technologies | New Castle, Delaware | May 2022 – August 2022

Software Developer Intern

- Developed an automated Azure DevOps CD/CI pipeline for code generation and test tool automation.
- Created Azure Cloud virtual machines (VM) to test before deployment.
- Application deployment and automated testing using Test Complete as part of CD/CI pipeline.
- Made heavy use of PowerShell to implement Azure DevOps environment.
- Implemented SSH based password-less communication between pipeline and VMs.

DR Technologies | Media, Pennsylvania | September 2021 – January 2022

Software Test Engineering Intern

- Developed an IoT mesh network to monitor temperature and events for warehouse application.
- Worked on Python backend and Javascript frontend of system.
- Tested device configuration and calibration interfaces and helped debug the integrated system.
- Used virtual machines to run server code and debug Python application.
- Used developer tools through browser to debug user interface.

West Chester University | West Chester, Pennsylvania| January 2022 - Present

IT Help Desk Consultant

VOLUNTEER & INVOLVEMENT

Computer Science Club | West Chester, Pennsylvania | September 2021 – Present

- Work with other club members on computer science related projects
- Discuss different topics in the Computer Science field

Cyber Security Club | West Chester, Pennsylvania | September 2021 – Present

- Participate in local CTF events
- Discuss news stories and emerging topics in the Cyber Security field

First Robotics Competition | Media, Pennsylvania | September 2013 – April 2022

- Participated in robotics competitions from Middle School through High School
- During college, volunteered mentoring younger competitors

TECHNICAL SKILLS

Languages and Applications: Java, C++, C, Python, Javascript, HTML, VirtualBox, SceneBuilder, NetBeans, IntelliJ, Visual Studio Code, JSON serializer/deserializer, Flask server, Bootstrap, web sockets, PowerShell, Azure DevOps, Azure Cloud, Bomgar, ServiceNow, Virtual Machines, SSH