GoBird

Justin Linwood, Lieu Phung, Chaughn Robin, Cole Snyder

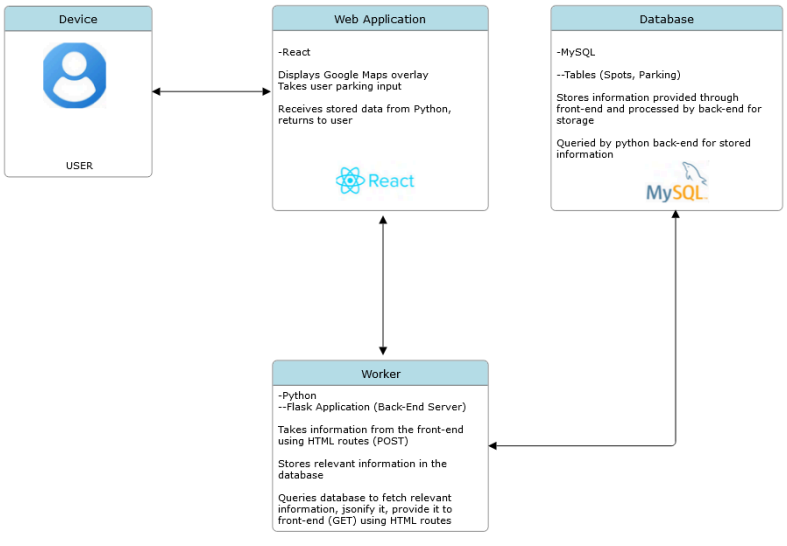
CHAPTER 1:

Purpose: The goal of our project is to create an attendance tracker for cars parked in West Chester University spaces, GoBird.

Navigate parking at West Chester University with ease using our app tailored to the WCU area, simplifying your campus commute and ensuring you're always steps away from where you need to be. GoBird is a parking app that will allow students and faculty at West Chester University to find open parking spots with ease, reducing the stress of searching for parking.

Key Features:

- ➢ Real Time Updates - GoBird will deliver real-time parking availability updates to the user by incorporating Google Maps API, which makes use of real-time refreshing, as well as data crowdsourced from users. The user will have a clear image of what areas have availability and what areas do not while using familiar mapping to get there.
- ➢ Geolocation - Using Google Maps API, GoBird will use geolocation to display relevant information as well as collect information from users.
- ➢ Google Maps API - The app will use Google Maps API to have a clean, familiar interface that is very widely used to make use of customized interactive maps with Google's large amounts of data.
- ➢ User-Friendly Interface - A simple interface utilizing a map service most users are familiar with will make using the app an easy process and minimize the cost of educating users.



CHAPTER 2:

Implementation:

- ➢ React - React will be used to build the front-end user interface. It will handle user interactions such as logging in/out and parking detail information. It will also fetch data from the Python backend using API calls. The information that has been fetched will be processed and integrated with the Google Maps API to display the information on the user's screen. It will also display relevant user information, parking times, and available locations.

- ➢ MySQL - MySQL will be used to store the user's username/password, parking location, parking space number, parking zone, and parking time information.

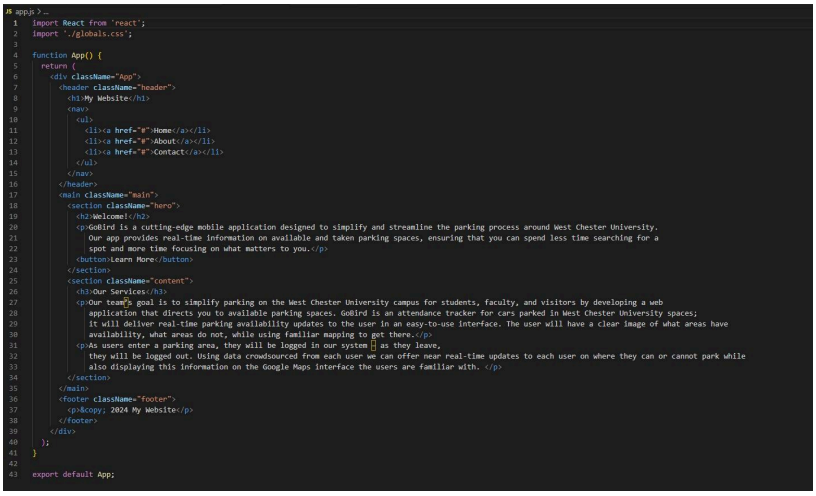
- ➢ Python - Python is the primary language for the back-end. This allows us to build logic for processing user information and Google Maps API. It also handles data needed to push/pull from MySQL.

- ➢ Google Maps API - Google Maps API will be used for geolocation of the user, customized maps for interaction and display, as well as location data which will be stored in MySQL.

The overall idea is the user will have a service presented to them similar in look to FlowBird or Google Maps. The map presented to the user will display available and unavailable spots in zones overlaid on the map similar to FlowBird's zones.

INTERMEDIATE MILESTONES (CHAPTER 3):

We have worked on creating the front end for our app (GoBird) by using React. We have started making this by creating a new App.js file and creating a base template to build the WebUI. At the moment it is just a page with simple headers, paragraphs, lists, and buttons.



```
app.js > ...
import React from 'react';
import './globals.css';

function App() {
  return (
    <div className="App">
      <header className="header">
        <h1>My Website</h1>
        <nav>
          <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Contact</a></li>
          </ul>
        </nav>
      </header>
      <main className="main">
        <section className="hero">
          <h2>Welcome!</h2>
          <p>GoBird is a cutting-edge mobile application designed to simplify and streamline the parking process around West Chester University.
            Our app provides real-time information on available and taken parking spaces, ensuring that you can spend less time searching for a
            spot and more time focusing on what matters to you.</p>
          <button>Learn More</button>
        </section>
        <section className="content">
          <h3>Our Services</h3>
          <p>Our team's goal is to simplify parking on the West Chester University campus for students, faculty, and visitors by developing a web
            application that directs you to available parking spaces. GoBird is an attendance tracker for cars parked in West Chester University spaces;
            it will deliver real-time parking availability updates to the user in an easy-to-use interface. The user will have a clear image of what areas have
            availability, what areas do not, while using familiar mapping to get there.</p>
          <p>As users enter a parking area, they will be logged in our system as they leave,
            they will be logged out. Using data crowdsourced from each user we can offer near real-time updates to each user on where they can or cannot park while
            also displaying this information on the Google Maps interface the users are familiar with. </p>
        </section>
      </main>
      <footer className="footer">
        <p>&copy; 2024 My Website</p>
      </footer>
    </div>
  );
}

export default App;
```

There is also a file called index.js which initializes our React application. index.js imports all of our necessary libraries, defines our root element, and renders the main app within the root element. We have created our Dockerfile as well as our .yaml files for our WebUI as well. We plan on implementing the Google Maps API with our WebUI to show the user's current location. We also plan on having an overlay over the API display to show the parking space locations. This is one of our current struggles. We are able to display our map on our website but the overlay still needs a lot of work, also we have other technical issues like hiding our API key in our public GitHub repository. In addition, we plan on using maps to count and create templates to mark out the parking spots rather than having someone go down each street and parking lot to count all of the parking spots and map it out.

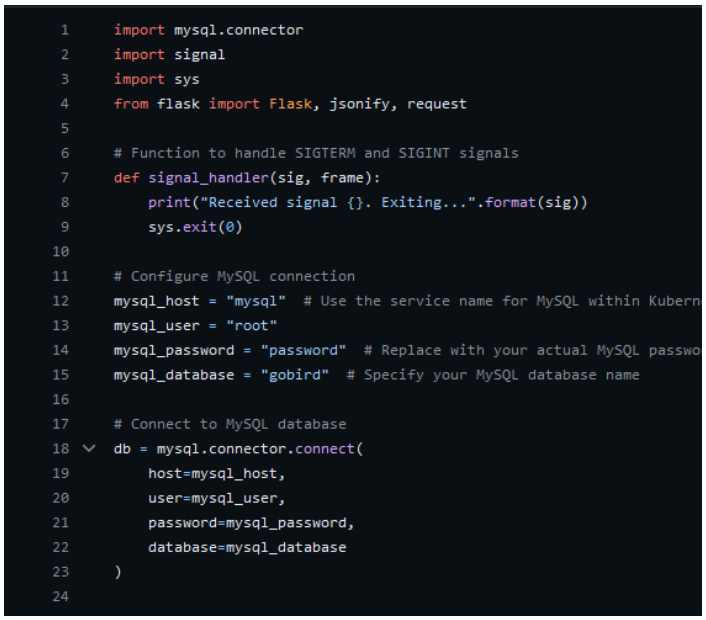
Our database Container has been created using MySQL in Docker.

```
CREATE DATABASE gobird;
use gobird;

CREATE TABLE spots (
  spots_number int NOT NULL,
  spots_street varchar(50) NOT NULL,
  spots_status varchar(50) NOT NULL,
  PRIMARY KEY (spots_number)
);

CREATE TABLE parking (
  park_spot_number int NOT NULL,
  park_time_in varchar(50) NOT NULL,
  park_time_out varchar(50) NOT NULL,
  PRIMARY KEY (park_spot_number),
  FOREIGN KEY (park_spot_number) REFERENCES spots(spots_number)
);

INSERT INTO spots (spots_number, spots_street, spots_status)
VALUES ("1", "S Church", "open"), ("2", "S Church", "taken"),("3", "S Church", "taken"),("4", "S Church", "open");

INSERT INTO spots (spots_number, spots_street, spots_status)
VALUES ("21", "University Ave", "open"), ("22", "University Ave", "taken"),("23", "University Ave", "taken"),("24",



CREATE USER 'root'@'%' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON gobird.* TO 'root'@'%';
FLUSH PRIVILEGES;
```

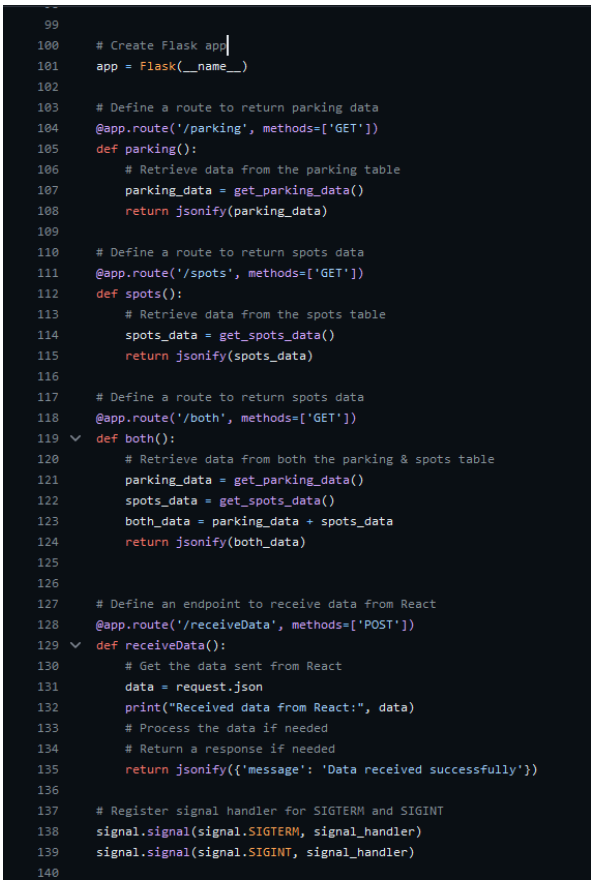
We have worked on creating the back end for our app by using Python. Our Dockerfile builds a Python image and installs necessary dependencies (mysql-connector, Flask) listed in the requirements.txt file. It also exposes ports for the other containers to communicate with. Currently, we are deciding whether to use Flask to communicate with the React container or whether we will just use the HTTP libraries built-in python. Flask may be more user-friendly and have more features, but depending on our use cases, HTTP may be sufficient. We will have more input into which method to use as we build the React container. Choosing to use Flask may require some API in React, but looks like the correct choice. It will allow us to more easily route and process HTTP requests from React, push HTTP responses to React, process data and insert into mySQL, handle json data, authentication (if necessary), and error handling. Dealing with all of these issues ourselves would be much more time costly and less feasible.

Our python worker currently can communicate with other containers, storing and pulling necessary information. It also builds a Flask back-end server for communicating with the React front-end. In Flask, we deal with functions that

correspond with specific jobs / functionalities – called endpoints. These endpoints route the requests from the React front-end. As we continue, we will build code for endpoints used by Flask. These will have logic for handling several different tasks, such as submitting forms, updating information, or asking the user for more information.



```python
import mysql.connector
import signal
import sys
from flask import Flask, jsonify, request

# Function to handle SIGTERM and SIGINT signals
def signal_handler(sig, frame):
    print("Received signal {}. Exiting...".format(sig))
    sys.exit(0)

# Configure MySQL connection
mysql_host = "mysql"  # Use the service name for MySQL within Kubern
mysql_user = "root"
mysql_password = "password"  # Replace with your actual MySQL passwo
mysql_database = "gobird"  # Specify your MySQL database name

# Connect to MySQL database
db = mysql.connector.connect(
    host=mysql_host,
    user=mysql_user,
    password=mysql_password,
    database=mysql_database
)

```



```python
# Create Flask app
app = Flask(__name__)

# Define a route to return parking data
@app.route('/parking', methods=['GET'])
def parking():
    # Retrieve data from the parking table
    parking_data = get_parking_data()
    return jsonify(parking_data)

# Define a route to return spots data
@app.route('/spots', methods=['GET'])
def spots():
    # Retrieve data from the spots table
    spots_data = get_spots_data()
    return jsonify(spots_data)

# Define a route to return spots data
@app.route('/both', methods=['GET'])
def both():
    # Retrieve data from both the parking & spots table
    parking_data = get_parking_data()
    spots_data = get_spots_data()
    both_data = parking_data + spots_data
    return jsonify(both_data)


# Define an endpoint to receive data from React
@app.route('/receiveData', methods=['POST'])
def receiveData():
    # Get the data sent from React
    data = request.json
    print("Received data from React:", data)
    # Process the data if needed
    # Return a response if needed
    return jsonify({'message': 'Data received successfully'})

# Register signal handler for SIGTERM and SIGINT
signal.signal(signal.SIGTERM, signal_handler)
signal.signal(signal.SIGINT, signal_handler)

```

The main struggles right now are building functionality within each respective container and then making those functionalities work together. Endpoints/functionality need to be constructed within the python code for properly handling requests, processing data, storing properly in mySQL and sending requests back to the front-end. Also incorporating whatever Google Maps API functions we wish to use. Lastly, the matter of properly running these things in Kubernetes / proper construction of yaml files (nodePorts, environment variables, etc.).

Our development process begins within our GitHub repository. Within our repository, we have set up multiple branches for each aspect of our design structure, for example we have a branch for web ui which is separate from our SQL database. When we begin to build the Docker experiments we have files that will use Kubernetes to create and manage our Docker containers. Once Kubernetes has

completed their tasks the Docker containers should be able to communicate with one another to complete the goal of our app.

FINAL RESULTS (CHAPTER 4):

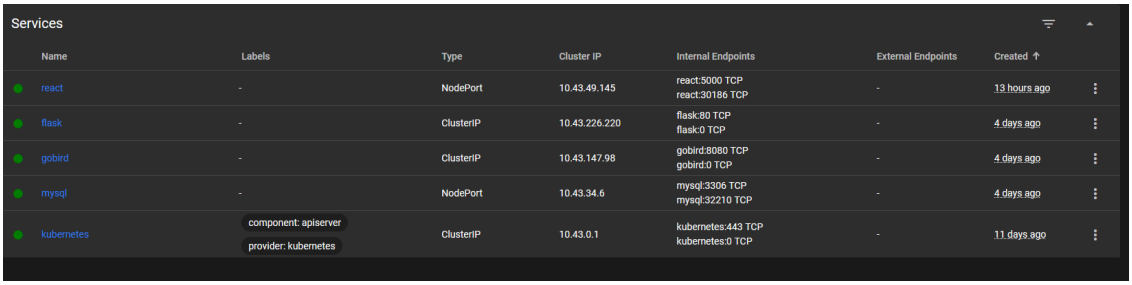
In summation, we have viable containers for the front-end (React), back-end (Python) and database (mySQL). The front-end and back-end communicate using a Flask application initialized in the Python worker, and run as a service on Kubernetes. The back-end and database communicate using mySQL-connector.

The python worker can take input from the front-end (POST) sent through HTML endpoints established with Flask and accessed by the React front-end. The worker interprets this data, storing relevant information in the database in tables associated with the user. The worker also pulls relevant information from the database, jsonifys it and serves it back to the front-end using (GET) HTML requests.

The database holds tables for parking information (parking number, arrival/departure time) and spot information (spot status, spot street).

Some of the main issues encountered surrounded general technical knowledge of Kubernetes and the interaction between containers. At one point, connecting the worker and database proved difficult until we used the proper host, IP, and ran the right commands in MySQL to allow the right privileges. Another issue relevant to Kubernetes was after having a working python worker container and being able to properly access the MySQL container, the backend container was still crashing (crashLoopBackOff). We discovered this was because of signals that Kubernetes sends that needed to be properly handled by the Python code. After adding code to manage K8's signals, the backend container stopped failing and continued to work as necessary.

The milestone we missed was deploying a React container that did not encounter any crashes. Our logs for our React container show it successfully running and connecting to the Flask service, but the container crashes without any logged errors. We have tried several different fixes for this, and with more time we would solve this issue with more debugging.



CONCLUSION:

We have a working deployment that can take input from the user, interpret what is needed in the Python worker, store it in the database, and reverse that process (pull from the database and serve it to the user). The front-end service has an efficient layout that has fields for relevant information. It also features a familiar google maps view with an easy to understand overlay that shows streets with and without available parking.

From this project, we have all learned basic technical knowledge of setting up containers in a linux environment, using/building Dockerfiles, deploying on and using kubernetes, using docker, and steps needed for different containers to interact.

This project has lots of opportunities for future development. Given more time, we would like to correct Kubernetes deployment of our front-end container. With more time and testing, we may update what information is stored and requested from the user. With further iterations, a more user friendly look would also be added.

Justin Linwood

Warminster, PA 18974
215-915-0111 | justin.linwood@verizon.net

Education

West Chester University, West Chester PA

Bachelor of Science – Major Computer Science, Minor Mathematics
Graduation Date: May 2024
GPA: 3.368

Technical Skills

Programming Languages

- Strong understanding and comfort coding complex projects in Java.
- Understanding of C and C++
- Understanding of Python and how to use it working close to hardware

Development Tools

- IntelliJ, JGrasp, Eclipse, Linux, Windows, GitHub, CloudLab, Jenkins

Relevant Coursework

Computer Science courses

- Cybersecurity · Cybersecurity II · Program Concepts and Paradigms · Datacom/Networking I · Software Engineering, Software Security, Artificial Inteligence, Intro to Cloud, Modern Malware
- · Calculus I - III · Linear Algebra, Discrete Mathematics, Statistics, Differential Equations

Certifications

Security+, CompTIA, January 2024
Computer Security, West Chester University, May 2024

Experience

Tony's Place, 4/17 – present

- Bartender
  - Duties included: taking orders, preparing drinks, providing high quality customer service in a high paced environment

References

Jason Howard – Manager - Tony's Place
Jayhoward77@gmail.com | 267-516-0765

John Sedlacsik – Teacher - Hatboro-Horsham School District
jsedlacs@hatboro-horsham.org | 267-475-9402

Lieu Phung

484-425-4683 | lieunphung@gmail.com

Education

West Chester University — West Chester, PA
Bachelor of Science in Computer Science — *Aug. 2023 – May 2025*

- Related coursework:
  Computer Science I
  Computer Science II
  Computer Science III
  Computer Security & Ethics

Lehigh Carbon Community College — Schnecksville, PA
Associate of Science in Computer Science — *Aug. 2020 – May 2023*

- GPA of 3.82

Extracurricular

Programming Contest | *Java* — September 2023

- Participated in West Chester University's programming contest to improve coding skills
- Solved problems correctly in a timely manner

Computer Science Club | — Aug. 2023 – Present

Work History

Recreation Attendant — August 2023 – Present
West Chester University — *West Chester, PA*

- Ensure patrons have valid identification
- Monitoring patrons to ensure a safe environment
- Clean strength and conditioning equipment
- Properly check-out/in equipment
- Act as a first responder in the event of an emergency

Stocking and Unloading Associate — Aug. 2020 – Aug. 2023
Walmart — *Whitehall, PA*

- Provide customers with any assistance needed in the store
- Unload general merchandise trucks according to each department
- Stock merchandise in its proper location

Technical Skills

Languages: Java
Developer Tools: Visual Studio Code

Chaughn Robin

chaughnr@outlook.com :: {484} 723-9366

EDUCATION

West Chester University of Pennsylvania, West Chester PA
Bachelor of Science in Computer Science, December 2024
GPA: 3.7/4.0 | Dean's List
Computer Science Club, Cyber Security Club

RELATED COURSEWORK:

- *Data Structures and Algorithms*
- *Computer Security and Ethics*
- *Computer Systems*
- *Digital Image Processing*

CURRENT COURSEWORK:

- *Database Management Systems*
- *Programming Language Concepts/Paradigms*
- *Introduction to Cloud Computing*

WORK HISTORY

West Chester University of Pennsylvania, West Chester, PA
Computer Technician **January 2024 – Current**

- Analyzed and processed information from students
- Collaborating with students to facilitate a variety of introductory programming courses
- Accommodated and instructed students in different levels of Computer Science curriculum with diverse concepts

PROJECTS

LibGDX (Java Game-Development Framework)
Word-Based iPhone/Android App **January 2024 - Current**
Developer

- Programmed a unique "hangman-like" word game based on acronyms and a large library of possible words using Java
- Created icons and artwork for the game using Paint.net
- Utilized LibGDX's portability features to create releases for PC, Android, and iPhone
- Produced several in-game tracking tools to measure player metrics and data for playtesting and further improvement / monetization using Java

Sidescroller Role-Playing Shoot 'Em Up PC Game **August 2023 - Current**
Developer

- Constructed several in-game systems for tracking player, player aim, different forms of movement, hitboxes, events and input based on player as well as A.I. characters using Java
- Created fully rigged 3D and 2D models and animations using Blender and Spine
- Recorded audio effects and voice lines using Audacity

RELATED SKILLS

Languages
Java, C++, Python, C#, Haskell, SQL
Tools & Services
Eclipse, MS Visual Studio, Docker, Kubernetes, Blender, Spine, MS Access, SPSS, MySQL
Operating Systems
Windows, Linux (Kali & Tails), macOS
Skills

- Conceptualizing and designing efficient algorithms for implementation
- Understanding and implementing appropriate data structures
- PC Building and Hardware

Cole Snyder

Phone: 484-467-8451
Email: 20snyderc@gmail.com

Education

West Chester University, West Chester, PA *Exp. Grad. Dec 2024.*

- Bachelor of Computer Science
- **Current GPA: 3.431/4** (Deans list)

Experience

Coding *Started August 2020*

- Partook in Computer Science classes at West Chester University.
  - CSC 141 Computer Sci I (Java).
  - CSC 142 Computer Sci II (Java).
  - CSC 240 Computer Sci III (Java).
  - CSC 231 Computer Systems (C & Linux).
  - CSC 241 Data Structures and Algorithms (Java).
  - CSC 301 Computer Security & Ethics.
- Completed a code academy course that covered Java.
- Completed a code academy course that covered HTML.
- Experience with object orientated programming (Java).
- Won the 2023 CSTA Hackathon at WCU.
- Attended WCPC Coding challenges.

Freelance I.T. *Started April 2019*

- Successfully completed computer repair and maintenance.
- Executed computers construction.
- Delegated software instillation, management, and set up.
- Administered peripheral repair.

Skills

- Meeting Deadlines.
- Managing Projects.
- Identifying, Analyzing, and creatively solving problems.
- Multi-tasking.
- Time Management.
- Accepting responsibility.
- Ability to work in a team and collaborate effectively.
- Proficient in Microsoft Word and Excel.
- Proficient in Google Workspace applications.
- Proficient and Java.
- Adapting and working with problems with new problems as they arise.