Attend & Learn

Team Members: Michael Burns, Joseph Haywood, Hayden Kanek
Anthony Mosley, Aroum Zombra

https://github.com/AJMosley345/CSC468-Project

Chapter 1:

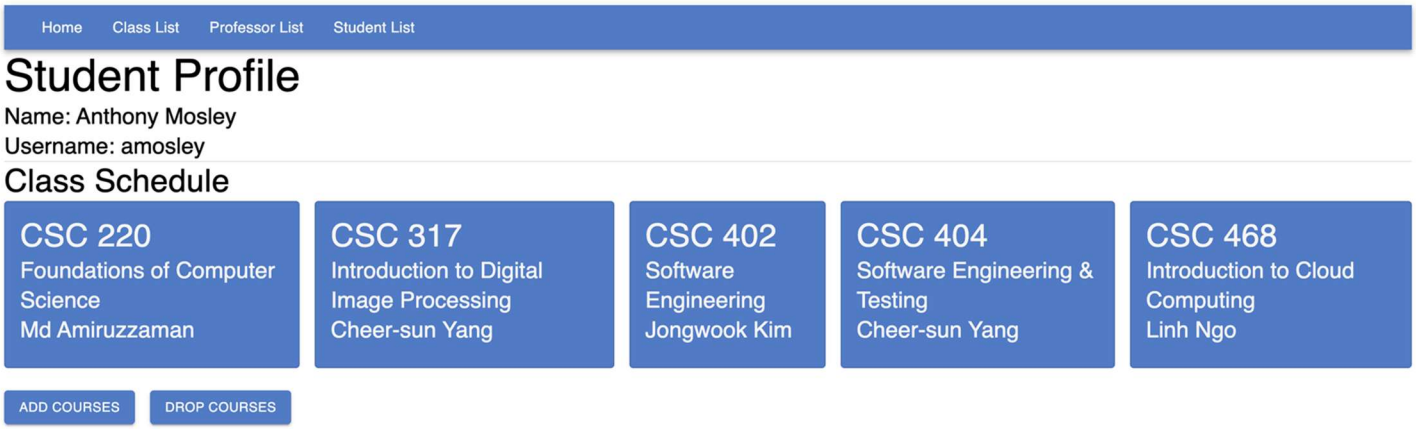
Vision:

A MyWCU type web app. This web app would mimic MyWCU in many ways. From adding and dropping classes, to being able to see financials. The original idea was to have professors make meetings for students within their classes and take attendance. The students would be able to check in with their assigned id and the professor would be able to confirm it. A student would be able to add and drop classes, see which professor is teaching it, class times, etc. All of these features would combine under a nice and easy to navigate user interface that would be easy on the eyes compared to MyWCU.

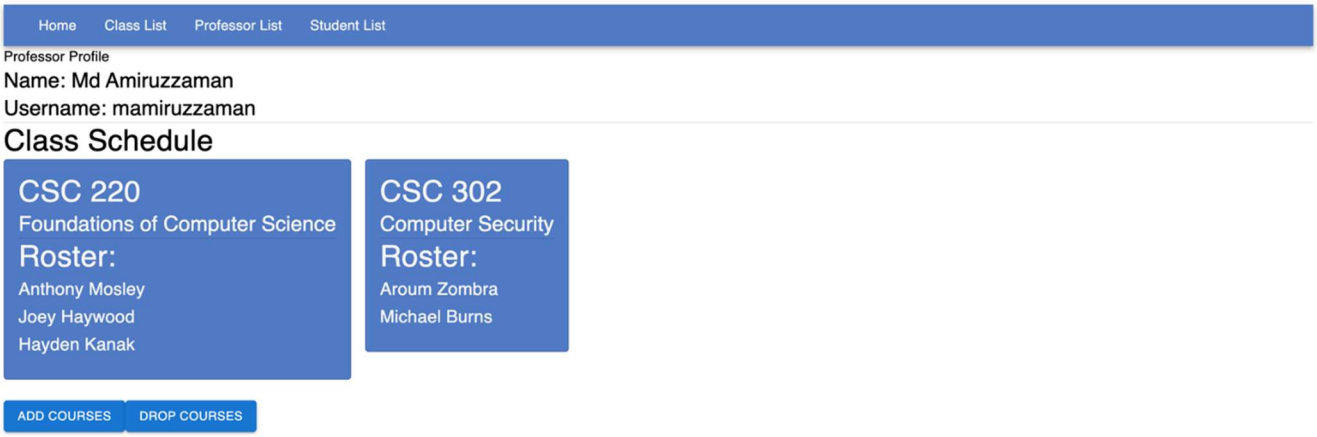
Features:

Within the time frame of the project, we were able to implement a decent number of features for both the students and professors. We weren't able to get the main feature, the attendance tracker, completely nailed down, but what we do have is a nice groundwork for what a better version of MyWCU could be.

- For Students:
    - They are able to see a details page that includes what classes they have, and what professors are teaching them.
    - They are able to select classes on their first login with a given username and password. Plans to implement a meeting/attendance taking feature never came to fruition due to time constraints.
    - The view below consists of what a student would see upon login and adding multiple classes to their profile. Within this view they can navigate to add courses or drop courses.
    - Student View:



- For Professors:
    - Professors can see what classes they are teaching, as well as students that are in them.
    - Plans to make a meeting feature, where professors would make a meeting for their class, then take attendance within that meeting never came to fruition due to time constraints.
    - Within the view shown below, a professor, upon first login, can select classes that they teach. Once these classes are added to their profile, they are also able to see which students are in those classes
    - Professor View:



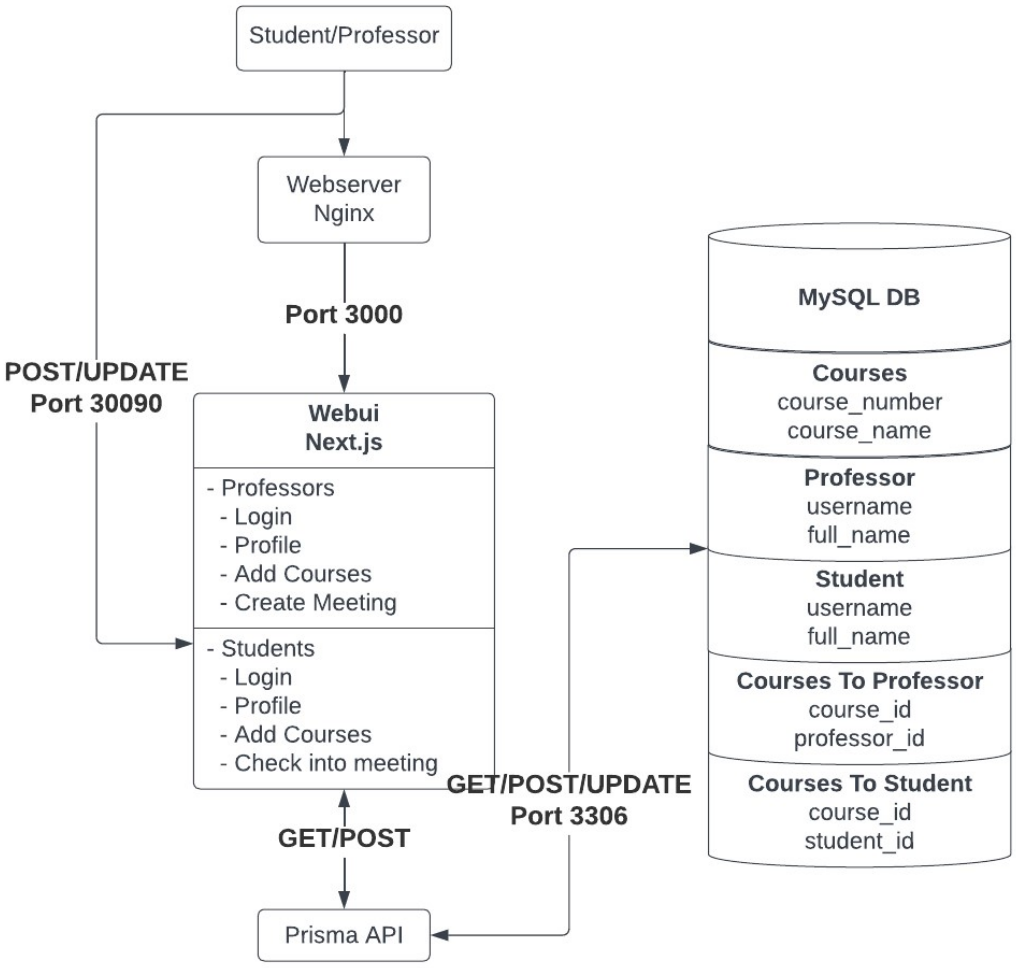
Chapter 2:
Technologies used in our project:
Next.js/Prisma (Frontend), Nginx (Webserver), and MySQL (Database) Stack:

- **Next.js**:
    - Next.js provides a host of features that we used in our project. From server-side rendering to its' powerful page routing capabilities. The way we implemented it will be detailed below:
    - The structure of Next.js is simple, all your "views" are stored in the "pages" directory. From here you can navigate to any page without having to define the routes yourself. This made it very easy to scaffold a multi-page website quickly. Within the "pages" directory, there is the "api" folder. There is how we used Prisma to make queries on our database.
    - The website is broken into two "flows", a student flow and a professor flow.
        - Student Flow:
            - The original idea was for a student to log in, add classes and be able to join a meeting created by a professor. What is present is the ability to go to a student's profile, add and drop courses and see those courses and professors that teach them.
        - Professor Flow:
            - Similarly, the professor was supposed to be able to log in, add courses they were teaching, and create meetings for those courses. It is just like the student's view, just that they can see the full roster of students that are in that class.
- **Prisma**:
    - Along with Next.js, we used Prisma to interact with our database. Prisma allowed us to quickly pull, modify, and delete data as we needed straight from the frontend. This saved us a lot of time, since we didn't have to define queries, or think about which ones we needed, they were built right in. One of its biggest positives was that it could create relational tables easily, without us having to define much of anything.
    - Prisma allowed us to quickly and efficiently scaffold a database schema, push it to the database itself, and make on the fly adjustments as we needed without having to do any raw SQL (except in seed.ts).

- **Nginx**:
    - We chose Nginx due to its capabilities serving static and dynamic content. Combining this with Next.js' capabilities in getting server-side content and static content, it made a very efficient and quick application. Nginx made it incredibly easy to configure anything we needed for the application. All it does is listen for the webui on port 3000 and goes from there.
- **MySQL**:
    - We chose MySQL for its reliability, popularity and support. It's quick to start up, powerful and easy to use. Combining this with Prisma's excellent integration, it made an easy to set up, use and change database. If we were able to implement it, MySQL would

be a great option for authentication, the eventual log-in system and meeting system we planned on creating.

Application Design



Chapter 3:

Progress/Accomplishments:

Database: We have completed the tables needed for mySQL database including students, courses, and professors. Student/professor tables utilize ids that auto generate in a record when created, and an assigned username. For the courses table, we have an id that auto generates a record when created, the course name and number, and two relational tables that connect the courses with students that are taking them and professors that are teaching them.

Frontend: We are utilizing Next.js for the WebUI. Next.js has provided a very easy way to deal with page routing, and its integration with Prisma and many other libraries has made it very simple for us to set up the project in a way where everything just works.

Webserver: The webserver we have been using, Nginx, has been hit or miss the entire project. It mostly works with a very simple configuration, but having nothing more than a single layer separating it and the webui has led to issues. We decided to shelve it at this point, since kubernetes provides a way to expose the webui without much configuration at all.

Prisma: We are utilizing Prisma to connect to mySQL database. Prisma has been very helpful in allowing us to quickly and efficiently pull, create, change, etc. All of the data within our database.

Dockerfiles:

Webui Dockerfile:

```
FROM node:18-alpine

# Set the working directory to /app
WORKDIR /app

# Copy package.json and package-lock.json to the container's working directory
COPY package.json yarn.lock ./

# Copy the rest of the application to the container's working directory
COPY . .

EXPOSE 3000
```

1. Prior versions of this Dockerfile used the regular Node image from DockerHub, now it is using the 18-alpine image, making the size smaller and making it quicker to build and push.
2. It does a lot less of the work now, only creating a work directory, copying package.json and yarn.lock
3. It copies the rest of the application and exposes port 3000.

Webserver Dockerfile:

```
FROM nginx

COPY ./default.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

1. Uses the official nginx docker image as a base
2. Copies the default configuration that is defined in the webserver folder
3. Exposes port 80
4. Runs nginx once the container is created.

Database Schema:

```
model Courses {
  course_id     Int          @id @unique @default(autoincrement())
  course_number String      @db.VarChar(250)
  course_name   String      @db.VarChar(250)
  taught_by     Professor[]
  taken_by      Student[]
}

model Professor {
  id             Int       @id @unique @default(autoincrement())
  username       String    @unique @db.VarChar(250)
  pass           String    @unique @db.VarChar(250)
  full_name      String    @db.VarChar(250)
  courses_taught Courses[]
}

model Student {
  id            Int       @id @unique @default(autoincrement())
  username      String    @unique @db.VarChar(250)
  pass          String    @unique @db.VarChar(250)
  full_name      String    @db.VarChar(250)
  courses_taken Courses[]
}
```

This file is Prisma's way of defining the schema for a database and for using the different fields and tables in the project.

1. Courses Model:
    a. Fields:
        i. course_id: id that is assigned to a course once it is created in the table
        ii. course_number: Course number as defined by the person inputting it, simple string

iii. course_name: Same as course_number
iv. taught_by/ taken_by: This field creates a one-to-many relationship between the Courses table and the Professor/Student table. It makes it so each course has an array of Professors that teach it and Students that take it.

2. Professor/Student Models:
    a. Fields:
        i. id: Each table has a unique id that is assigned to record when it is created
        ii. username/pass/full_name:: Username/password/full name that is assigned to a Professor/Student
        iii. courses_taught/courses_taken: This field is how the Professor and Student models are connected to Courses respectively

Data Creation:

Using Prisma's ability to execute raw SQL, we made a seed.ts file that seeds the database with the information that was previously in init.sql. Since the database schema would now be defined by Prisma (explained later), we needed to have the data be inserted after the schema was defined and created in the database.

- Courses: Inserts a list of classes that were scraped from the WCU course page

```
const courses = await prisma.$executeRaw`
INSERT INTO Courses (course_number, course_name)
    VALUES
    (${'CSC 112'},${'Programming & Data Science'}),
    (${'CSC 115'},${'Introduction to Computer Programming'}),
    (${'CSC 141'},${'Computer Science I'}),
    (${'CSC 142'},${'Computer Science II'}),
    (${'CSC 220'},${'Foundations of Computer Science'}),
    (${'CSC 231'},${'Computer Systems'}),
    (${'CSC 240'},${'Computer Science III'}),
    (${'CSC 241'},${'Data Structures and Algorithms'}),
    (${'CSC 242'},${'Computer Organization'}),
    (${'CSC 301'},${'Computer Security & Ethics'}),
    (${'CSC 302'},${'Computer Security'}),
    (${'CSC 317'},${'Introduction to Digital Image Processing'}),
    (${'CSC 321'},${'Database Management Systems'}),
    (${'CSC 331'},${'Operating Systems'}),
    (${'CSC 335'},${'Data Communications and Networking I'}),
    (${'CSC 336'},${'Data Communications and Networking II'}),
    (${'CSC 345'},${'Programming Language Concepts and Paradigms'}),
    (${'CSC 400'},${'Internship'}),
    (${'CSC 402'},${'Software Engineering'}),
    (${'CSC 404'},${'Software Engineering & Testing'}),
    (${'CSC 416'},${'Design and Construction of Compilers'}),
    (${'CSC 417'},${'User Interfaces'}),
    (${'CSC 466'},${'Distributed and Parallel Programming'}),
    (${'CSC 467'},${'Big Data Engineering'}),
    (${'CSC 468'},${'Introduction to Cloud Computing'}),
    (${'CSC 471'},${'Modern Malware Analysis'}),
    (${'CSC 472'},${'Software Security'}),
    (${'CSC 476'},${'Game Development'}),
    (${'CSC 481'},${'Artificial Intelligence'}),
    (${'CSC 490'},${'Independent Project'}),
    (${'CSC 495'},${'Topics in Computer Science'}),
    (${'CSC 496'},${'Topics in Complex Systems'}),
    (${'CSC 497'},${'Topics in Computer Security'}),
    (${'CSC 499'},${'Independent Study in Computer Science'});
`;
```

- Students/Professors: Creates some basic information

```
const professors = await prisma.$executeRaw`
    INSERT INTO
        Professor (username, pass, full_name)
    VALUES
        (${"rburns"}, ${"Z14VdAlyK"}, ${"Richard Burns"}),
        (${"lngo"}, ${"9fbzmBWeL"}, ${"Linh Ngo"}),
        (${"schen"}, ${"XS98PtClR"}, ${"Si Chen"}),
        (${"cyang"}, ${"XqetLj4bO"}, ${"Cheer-sun Yang"});
`

const students = await prisma.$executeRaw`
    INSERT INTO
    Student (username, pass, full_name)
    VALUES
        (${"amosley"}, ${"J98xWOEEo"}, ${"Anthony Mosley"}),
        (${"azombra"}, ${"oAm1yI8SQ"}, ${"Aroum Zombra"}),
        (${"mburns"}, ${"4aPJACpup"}, ${"Michael Burns"}),
        (${"jhaywood"}, ${"bVYFdjWB3"}, ${"Joey Haywood"}),
        (${"hkanak"}, ${"4W6l9q9w3"}, ${"Hayden Kanak"});
```

Chapter 4:

Throughout our project, working with Kubernetes has been frustrating. While it provides amazing functionality with rapid deployment and testing, some errors and overall quirks make it very difficult to work efficiently. After much testing, and hardship, we finally nailed down a pair of config files that work for our database and webui.

db-deployment:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: db
  name: db
spec:
  selector:
    matchLabels:
      app: db
  template:
    metadata:
      labels:
        app: db
        network: "true"
    spec:
      containers:
        - env:
            - name: MYSQL_DATABASE
              value: project
            - name: MYSQL_ROOT_PASSWORD
              value: test1234
          image: mysql:latest
          name: db
          ports:
            - containerPort: 3306
      restartPolicy: Always
```

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: db
  name: db-service
spec:
  ports:
    - name: "3306"
      port: 3306
      targetPort: 3306
      protocol: TCP
  selector:
    app: db
status:
  loadBalancer: {}
```

- Starting with the db-deployment.yaml file, it does most of what the Database dockerfile used to do. As you can see, we started using the base mysql image from Docker Hub, as we only needed to define the environment variables and nothing else. This file just defines what the base deployment will be and sets the container port to be 3306, allowing for any other pods in the cluster to access it on that port.
- db-service: This part defines the service portion of the database deployment. This service allows all the pods and associated services within them to access the database at http://db-service:3306. This allows connections between each pod to go smoothly and all services to be able to talk to each other easily.

webui-deployment:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webui
spec:
  selector:
    matchLabels:
      app: webui
  template:
    metadata:
      labels:
        app: webui
    spec:
      containers:
        - name: webui
          args:
            - sh
            - -c
            - sleep 10
            - cd /app
            - yarn install
            - npx prisma migrate deploy
            - npx prisma db seed
            - yarn build
            - yarn start
          image: ajmosley345/468-project:webui
          env:
            - name: DATABASE_URL
              value: mysql://root:test1234@db-service:3306/project?schema=public
          ports:
            - containerPort: 3000
```

- This file defines our webui deployment. It creates a pod using the dockerfile image created by us that was shown earlier and sets an environment variable for the database url for Prisma to use.
- It defines the container port for all other pods to be able to use and is setup to run multiple commands before starting the application properly. These had to be run after the pods creation as they require the database to be fully setup and started before they could be run.
- webui-service: We got the webui service started up by simply running “kubectl expose deploy/webui --port=3000 --type=NodePort -n project” within the project startup script. This assigns a random port to the application and exposes it outside of the cluster.

nginx-deployment:

- Nginx was hit or miss during most of the development time. Sometimes it would work fine and be able to see all the services, and sometimes it couldn’t see anything. We ended up using it during the docker container/ docker compose part of the project, but shelved it once we got to kubernetes. We still include it as part of the stack because in an ideal world it would be what we would use for a reverse proxy.

Chapter 5:

Our current goal from when we started the project has changed. Originally, we were going make a cloud based bus pass service around West Chester. We reached out to Transloc (the company who designed the WCU Shuttle app) to try to get access to their API's. If this would have worked out, we would've had current-time GPS data from the buses. Transloc threw us the cold shoulder, and didn't reach back out to us even with a no.

We then tried to expand the scope of our project, with a recommendation from Dr. Ngo. Our next idea was to make a larger scale buss bass idea for the Delco area using the SEPTA transit system API. We planned on making individual user accounts with different levels of access, such as student, commuting worker or government. We were able to find the SEPTA API on GitHub. The problem was that the API was outdated, and to get the data we would have to manually input all the routes. That would've taken a while, and we felt this was out of the scope of our capability.

We still wanted to create an application with the university in mind. Hence, we decided on a cloud-based class attendance system. We wanted to use a lot of the same technologies and programming languages, however most of our vision has changed. All of us had proficiency in different languages, and we are trying to implement the languages most compatible with each other. Our first idea for a database was to use MongoDB, but MySQL ended up working better with our front and backend.

Final Project Assessment:

We were able to successfully manage and deploy a working service that is similar to that of MyWCU but with an easy-to-navigate UI that could help students and professors view their class schedule easier. With Attend & Learn, students are able to add and drop classes but we were not able to successfully create an attendance tracker that could essentially create a "meeting" for each class that professors could create and students could verify that they attended classes. We were not able to create these "meetings" due mainly to time constraints, but also, implementation issues such as students actually being in class to track the attendance (i.e. integrity that students actually show up to class instead of just tracking it and not showing up).

Anthony Mosley

Oxford, Pennsylvania
ajmosley345@gmail.com - (717)-371-0907

Experience

Stoner Inc. - May 2022 - Aug 2022/Dec 2022-Jan 2023
Junior Network Administrator/IT Support

- Executed smaller tasks to aid the team with their projects
- Helped set up their new phone system (RingCentral)
- Assisted multiple departments with technology setup and issues
- Setup 30+ machines to be deployed across the company
- Worked with their servers to convert an old camera server into a virtual machine

Education

West Chester University of Pennsylvania - Aug 2020 – Dec 2023
Bachelor of Science - BS, Computer Science

Skills

- Systems Administration
  - Server Administration/Maintenance
  - Computer Networking
- Backend Development
  - MySQL
  - Nginx
- Software Engineering
  - Python
  - JavaScript

Michael Burns

Bath, PA 18014
Email: mburns0163@gmail.com
Phone: 484 892 0163

Work Experience

IT Technician - September 2022 to Present
IT Edge - West Chester, PA

- Configure all incoming client orders to IT Edge technology standards.
- Install devices with all required software and troubleshoot for any issues before getting ready for delivery.
- Troubleshoot and repair hardware and software for client equipment.
- Assist with on-site setup and configuration, as well as troubleshooting client issues.
- Contact clients with updates on repairs, and work towards the most effective solution.
- Document and enter all work done each shift.

Education

West Chester University of Pennsylvania - January 2022 to Present
Bachelor of Science – BS, Computer Science
Northampton Area Community College - Sep 2019 – Dec 2021
Applied Associate of Science – AAS, Information Technology

Skills

- Computer Operation
- Microsoft Office
- C++ (1 year)
- C/C++
- Technical support
- Software troubleshooting

Joseph Haywood

Quakertown, PA 18951
Email: joey.haywood49@outlook.com
Phone: 267-347-0880

Experience:

Intelligence Sergeant (35F10), Pennsylvania Army National Guard
S2 Staff Section, 56th Stryker Brigade Combat Team
August 2020-Present

- Accountable of all section equipment and vehicles
- Assists with physical security protocols and handling of classified materials
- Supports junior enlisted soldiers in military and civilian careers

Intelligence Analyst (35F), Pennsylvania Army National Guard
Deployed to Poland in S2 Staff Section, 278th Armored Cavalry Regiment March 2019-March 2020
October 2016-August 2020

- Handled classified material on daily basis
- Performed maintenance checks on vehicles and equipment
- Created and briefed mission sensitive reports to Squadron Commander and NATO Allies
- Reviewed and updated SCAR (security clearance access roster)

Education

West Chester University **GPA: 3.54**
B.S. Computer Science & Minor in Applied Statistics
August 2020 – May 2024

Cochise College **GPA: 3.48**
Intelligence Operation Studies
October 2017 – February 2018

Skills

- Programming: Java, R, C, and Haskell
- Database management systems: mySQL
- Information security, physical security, and communication security

Aroum Zombra

Phone: 267 530 8083
Email: zombraahmed12@gmail.com
LinkedIn: https://www.linkedin.com/in/zombraahmed

Education:

West Chester University **West Chester, PA**
Bachelor of Sciences in Computer Sciences September 2021
Expected Graduation: May 2023
Blackwood Community college: New Jersey, NJ
Associate in Computer Science May 2021
Relevant Courses Worked:
Foundation in C++, Java Programming, Data Structures & Algorithms, Programming Language Concepts and Paradigms, Computer Security & Ethics, Calculus I&II, Physics I&II, User Interface, Software Testing

Skills:

- **Programming Languages:** Java, Python, C++
- **User Interface:** JavaScript, HTML5, CSS, Node.js, React
- **Software Engineering:** Object Oriented Programming, Software Design, UML diagram

Work Experience:

Uber Technology/ Delivery & Passengers Driver

- Driving passengers safely to their destination
- Managing frequent issues related to the trips
- Good leadership and communication skills

Technical Project:

- Achieved some projects Front-End web application
- Achieved some projects Back-End web application
- Tackled Data Structure with Binary Search Tree

Hayden Kanak

Computer Science

Contact

Address
West Chester, PA, 19382

Phone
215-301-9681

E-mail
haydenk1708@gmail.com

Skills

- Java (iGRASP, Eclipse) — Very Good
- Python (Spyder) — Average
- C — Good
- IOS App Development (Swift) — Good
- Software Security — Very Good
- Data Structures & Algorithms — Very Good

Intuitive honors computer science student, with a 3.8 GPA, who excels at programming in Java and C. Seeking an intern experience that will leverage my software engineering and cybersecurity background but eager to work in a variety of other fields. Coming with excellent communication skills and knowledge of statistical concepts and principles.

Education

2020-08 - Current

Bachelor of Science: Computer Science

West Chester University - West Chester, PA

- *Minor in Applied Statistics*
- *Minor in Communication Studies*
- *Computer Security Certificate*

Honors/Awards

- Dean's List - Honors College (Spring 2021 - current)
- Ann's Choice Scholars Program Scholarship (Fall 2020 - current)
- West Chester University Academic Excellence Scholarship (Fall 2020 - current)

Work History

2021-02 - Current

Team Member

Tropical Smoothie Cafe, West Chester, PA

- Operate the register/automated order system including entering customer orders, accurately handling cash, and being responsible for the drawer.
- Train new team members to Tropical Smoothie standards.
- Work efficiently with team to prepare food/smoothies and ensure the customer is completely satisfied.

2022-06 - 2022-08

Front End Clerk

Lowes Home Improvement, Willow Grove, PA

- Performed clerical duties to process returns, manage

procedures related to Centralized Return to Vendor and appropriately dispose of returned items.
- Utilized Lowe's web-based inventory database to research customer inquiries, helping to find the correct product replacement.
- Worked quickly to handle customer orders, returns, & other requests efficiently and in a friendly manner during difficult situations.

2018-02 - 2021-08

Wait Staff Member

Ann's Choice Retirement Community, Warminster, PA

- Practiced safe, sanitary food handling for preparation and service to maximize pleasant dining experience for residents
- Utilized ordering management database to input and track resident food orders against their accounts for billing purposes.
- Trained new team members on all Banner's Cafe procedures.

Volunteering & Fundraising

Honors Student Association Member

- Volunteer for various community & honors events such as Adopt-a-Block, Aid to South Africa, Barclay Friends, & Brandywine Valley ASPCA

Applications

- Microsoft Word
- Microsoft Excel
- Microsoft PowerPoint
- Microsoft Access
- Microsoft Project
- Google email (Gmail)
- Google Apps (Docs, Sheets, Slides etc.)
- Dropbox
- Collaboration: MS Teams, Slack,
- Zoom, WebEx
- Asana