Coffee Break

CSC688 Computer Science Project III
National University
Professor: Dr. Barbara Lauridsen

Presented By:
Kyle Glover(Project Lead)
Stephen Okeke(Project Lead)

Development Team

Kyle Glover – Team Lead | Project Manager | Front-End/UI | Backend | Tester

Stephen Okeke - Software Developer | Front-End/UI | Tester

Agenda

- Project Introduction
- Design Architecture
- Testing
- Product Demonstration
- Tasks For the Future
- Lessons Learned

Introduction

- Problem Statement
- Technology Overview
- Customer Market
- Statement of Scope
- Prototype Capabilities

Technology Overview

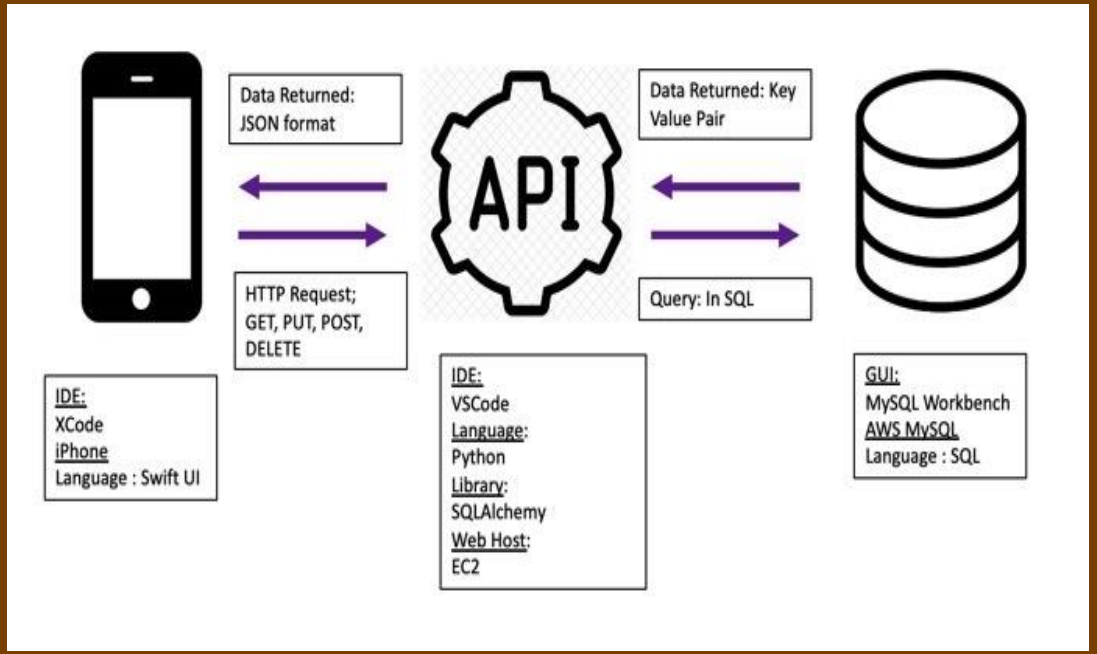
- Mobile Platform: iOS
- Integrated Development Environment: Xcode, Visual Studio Code
- Front-end languages: Swift UI
- Web Service APIs: Representational State Transfer (Rest) methodology
- Back-end languages: Python
- Hosted Services by AWS: MySQL, EC2
- Version Control: Git, GitHub

Project Statement

- Statement of scope :
  - Anxiety of non-avid coffee drinkers
  - A cold coffee experience
  - Complicated group ordering

Structure

- Used Flask REST API design and create Database.
  - IDE: VSCode
  - Language: Python
  - Database: SQLite
  - Framework: SQLAlchemy, Marshmallow, JWT
- Used Swift UI to design and create User Interface
  - IDE: Xcode
  - Language: Swift UI (Released Sept. 2019)
- Constraints
  - iOS Application (iPhone Exclusive)
  - Swift UI applications works only on iOS 13.0



Database Components

- Object Relational Mapper: SQLAlchemy
  - Generates database as Objects in Python classes.
  - Creates relationships to models and SQL Queries.
- Marshmallow Flask database-agnostic framework library for creating REST APIs.
  - Serializes and deserializes data.

SQLAlchemy class Example

```python
class group_order(db.Model):
    __tablename__ = 'group_order'

    group_id = db.Column('group_id', db.Integer, primary_key=True, autoincrement=True)
    name = db.Column('name', db.String(100), nullable=False)
    order_location = db.Column('order_location', db.String(100), nullable=False)
    order_time = db.Column('order_time', db.String(50), nullable=False)
    date = db.Column('date', db.String(75), nullable=False)
    address = db.Column('address', db.String(75), nullable=False)
    is_active = db.Column('is_active', db.Boolean, nullable = False)
    created_at = db.Column('created_at',db.DateTime, default=datetime.utcnow)

    # Foreign key linking profile to the admin of a group order
    admin = db.Column('admin', db.String, db.ForeignKey('profile.username'), nullable=False)

    # relationship for the members of the group order to store in table
    member = db.relationship('member', cascade="all,delete", backref ='order_member', lazy=True)


    def __init__(self, name, order_location, order_time, admin, is_active, date, address):
        self.name = name
        self.admin = admin
        self.order_location = order_location
```

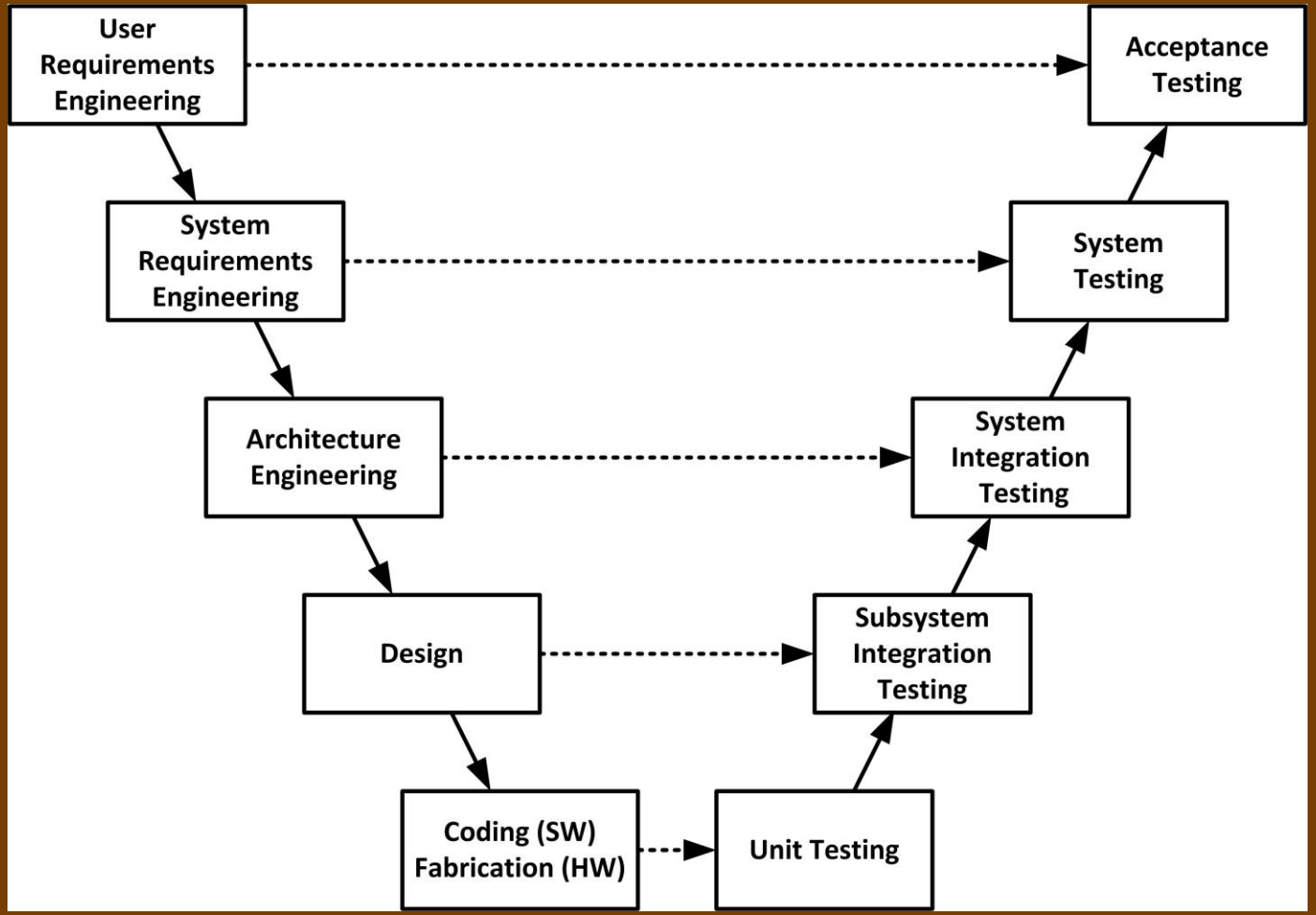
Database API Endpoints

- Allow for the mobile application Coffee Break to communicate with the database using API Endpoints.
- They follow the CRUD methodology where each endpoint is designed to perform a certain task.
- Example: Using the POST method is used to create an instance (row) for a table.

API Endpoint Example:

```python
#Sign Up a user
@app.route('/signUp', methods=['POST'])
def add_user():

    username = request.json['username']
    first_name = request.json['first_name']
    last_name = request.json['last_name']
    hashed_password = bcrypt.generate_password_hash(request.json['password']).decode('utf-8')
    email = request.json['email']
    description = request.json['description']

    existing_email = profile.query.filter_by(email=email).first()
    existing_username = profile.query.filter_by(username=username).first()

    if not existing_email and not existing_username:
        new_user = profile(username,first_name, last_name, email, hashed_password, description)
        db.session.add(new_user)
        db.session.commit()
        return {'message': 'User has been created!.'}, 200

    if existing_email:
        return {'message': 'Email is already in use.'}, 400

    if existing_username:
        return {'message': 'Username is already in use.'}, 400

    return {'message': 'Error was not caught.'}, 400
```

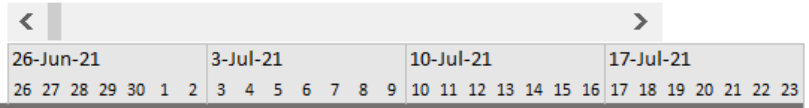
Testing Model



Gantt Chart Scheduler

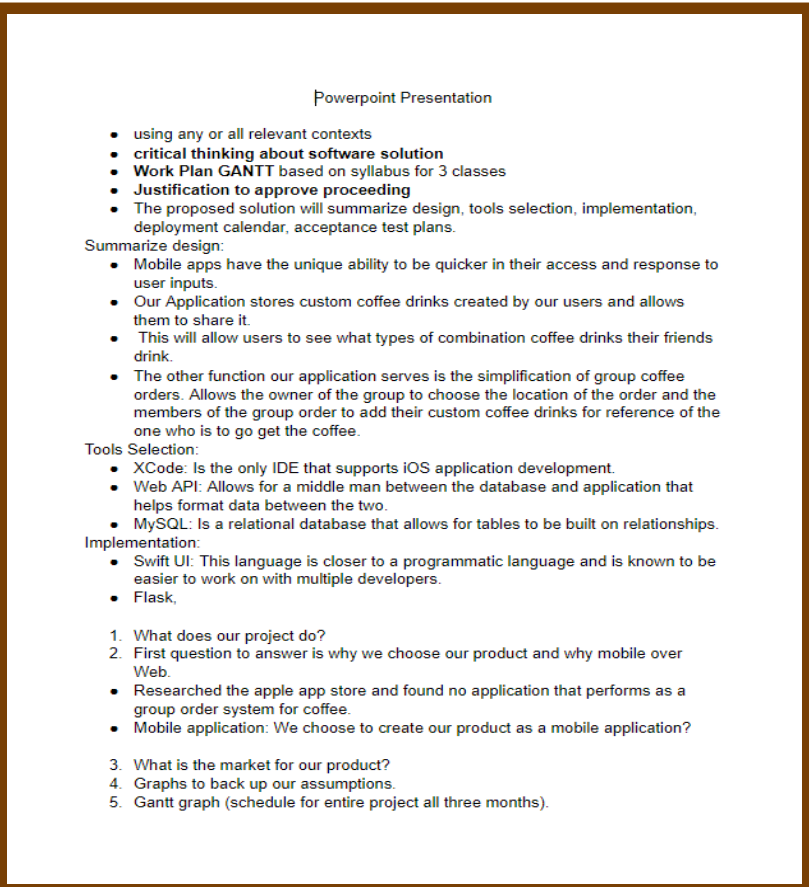
Coffee Break

Project Lead: Kyle Glover (KG)
Teammate: Stephen Okeke (SO)
Project Start: Sat, 1-May-2021
Display Week: 0



TASK	Assigned To	Progress	START	END
Week 9				
Draft Framework CH1, CH2	KG/SO	100%	28-Jun-21	8-Jul-21
Database Design	KG	90%	28-Jun-21	25-Jul-21
MyCoffee Tab	KG	80%	28-Jun-21	31-Jul-21
MyOrder Tab	KG	60%	28-Jun-21	31-Jul-21
MyProfile Tab	SO	50%	28-Jun-21	31-Jul-21
MyFriend Tab	SO	50%	28-Jun-21	31-Jul-21
Meetings 3/3	KG/SO	100%	28-Jun-21	5-Jul-21
Week 10				
2-on-1	KG/SO	100%	3 Jul-21	4 Jul-21
Decided to push all deliverables				
Presentation Slides	KG/SO	100%	15 Jul-21	31-Jul-21
Meetings 3/3	KG	100%	5 Jul-21	11 Jul-21
Week 11				
Meetings 3/3		100%	12-Jul-21	18-Jul-21
Week 12- Week 12.5				
Completed Presentation Slides		100%	12-Jul-21	
Finish Framework Doc		70%	19-Jul-21	1-Aug-21
Update OCD		70%	19-Jul-21	1-Aug-21
Update STP		70%	19-Jul-21	1-Aug-21
Update SDD		70%	19-Jul-21	1-Aug-21
Meetings 6/6		100%	19-Jul-21	31-Jul-21

Google Docs Documentation



Powerpoint Presentation

- using any or all relevant contexts
- **critical thinking about software solution**
- **Work Plan GANTT** based on syllabus for 3 classes
- **Justification to approve proceeding**
- The proposed solution will summarize design, tools selection, implementation, deployment calendar, acceptance test plans.

Summarize design:

- Mobile apps have the unique ability to be quicker in their access and response to user inputs.
- Our Application stores custom coffee drinks created by our users and allows them to share it.
- This will allow users to see what types of combination coffee drinks their friends drink.
- The other function our application serves is the simplification of group coffee orders. Allows the owner of the group to choose the location of the order and the members of the group order to add their custom coffee drinks for reference of the one who is to go get the coffee.

Tools Selection:

- XCode: Is the only IDE that supports iOS application development.
- Web API: Allows for a middle man between the database and application that helps format data between the two.
- MySQL: Is a relational database that allows for tables to be built on relationships.

Implementation:

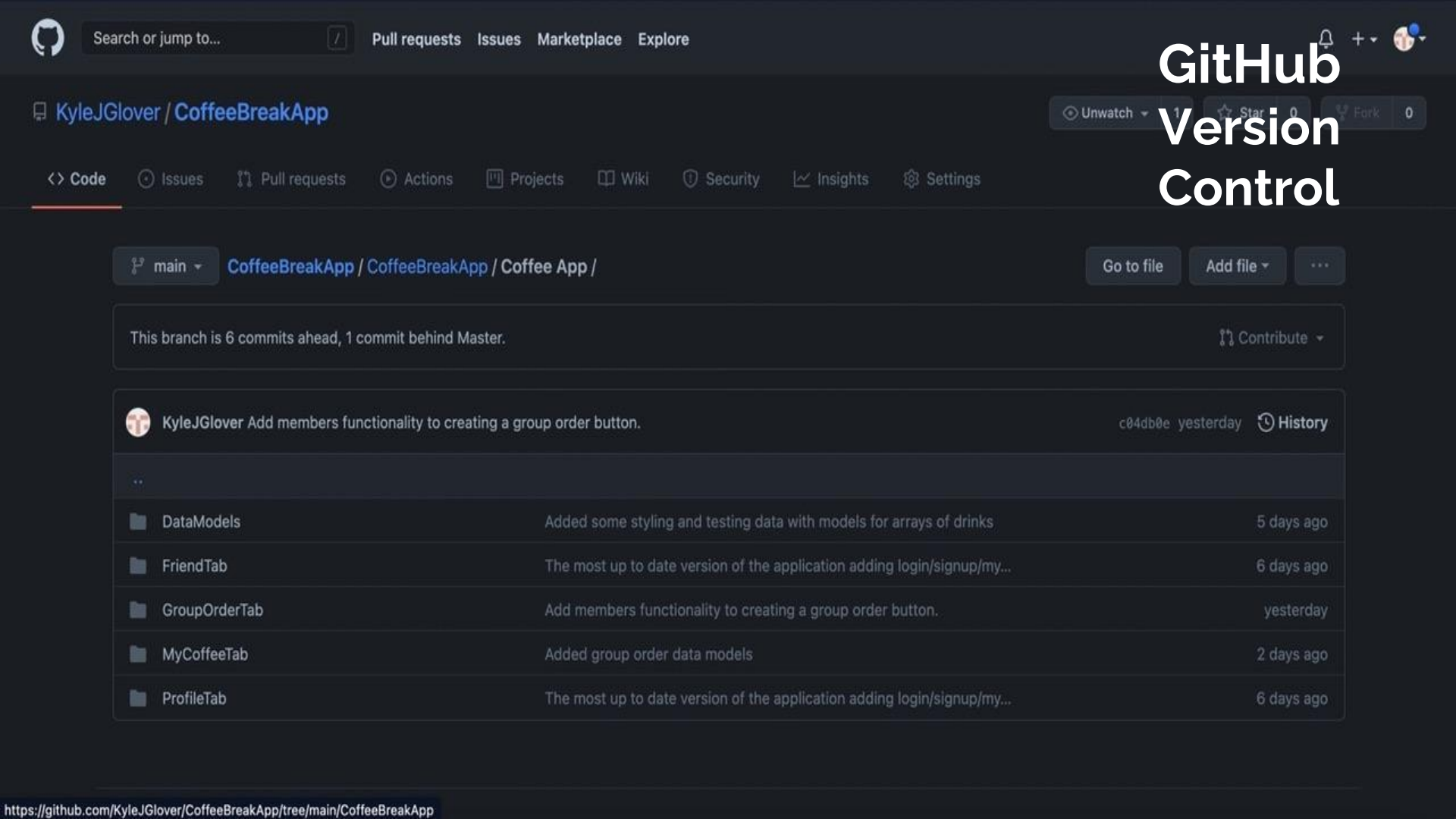
- Swift UI: This language is closer to a programmatic language and is known to be easier to work on with multiple developers.
- Flask,

1. What does our project do?
2. First question to answer is why we choose our product and why mobile over Web.

- Researched the apple app store and found no application that performs as a group order system for coffee.
- Mobile application: We choose to create our product as a mobile application?

3. What is the market for our product?
4. Graphs to back up our assumptions.
5. Gantt graph (schedule for entire project all three months).

By Kyle Glover



GitHub Version Control

Zoom Communication

- The Coffee Break Team uses Zoom as the main form of communication.
- Using this platform the team can speak to each other as if on a phone call.
- The added benefit of a Zoom call is the team can share screens to assist one another.

By Stephen

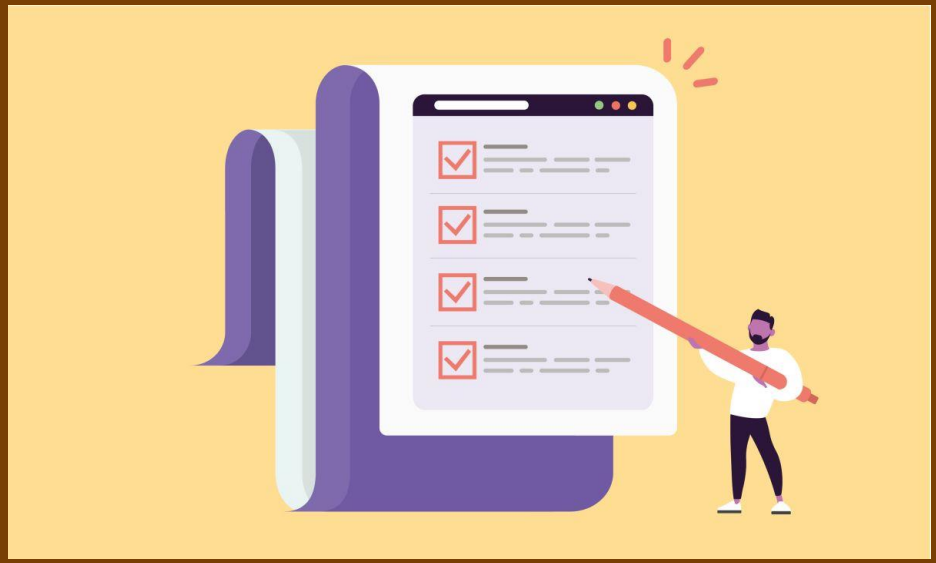
Verification & Audit

- Using Google Docs to note & monitor Changes
- We use GitHub, Zoom, and Google Docs to check that we are both on the same agreed upon standard.

By Stephen

Tasks

- Creating Delete function for Order Tab
- Add members functioning for the Order's.
- AWS MySQL Server Host.
- EC2 Cloud based host for API.
- Beautification
- Friends Tab
- Profile Tab
- User testing/ feedback



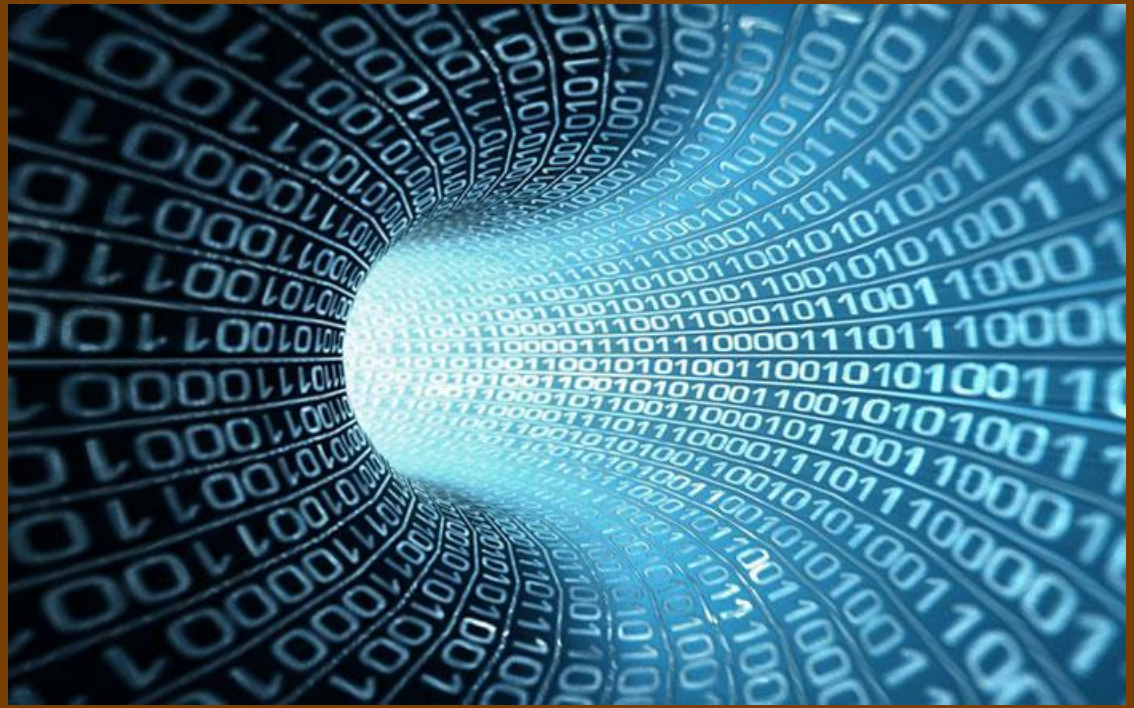
Future Tasks

- Creating Delete function for Order Tab
- Add members functioning for the Order’s.
- AWS MySQL Server Host.
- EC2 Cloud based host for API.
- Beautification
- Friends Tab
- Profile Tab

Lessons Learned –Kyle

- How to create an Entire application front to back.
- The struggle of working with a timeline.
- How to design and implement an API.
- How to design applications to dynamically call databases.
- Working on both software and documentation side of software engineering.
- Dealing with complications.

Lessons Learned –Stephen



Questions?