**Week 1:**

**Team Member 1:**

Step 1: Loading Data into MongoDB Atlas

Load the "Sample Restaurant Data" into MongoDB Atlas.

Step 2: Building a Web API - Part 1

Install necessary packages (express, cors, mongoose).

* db.initialize
* db.addNewRestaurant(data)
* db.getAllRestaurants(page, perPage, borough)

Implement 3 promise-based functions for the Web API.

Set up the initial project folder.

Initialize the module for interacting with MongoDB (db.initialize method).

**Team Member 2:**

Step 4: Add Security Features - Part 1

Implement environment variable setup for the MongoDB connection string.

Research and determine the strategy for user authentication and authorization (JWT, Session, Cookie, etc.).

**Week 2:**

**Team Member 1:**

Step 2: Building a Web API - Part 2

Implement remaining 3 promise-based functions for the Web API.

* db.getRestaurantById(Id)
* updateRestaurantById(data,Id)
* deleteRestaurantById(Id):

Define and implement routes for the Web API.

Step 2: Building a Web API - Part 3

Finalize implementation and testing of API routes:

* POST /api/restaurants
* GET /api/restaurants
* GET /api/restaurants/:id
* PUT /api/restaurants/:id
* DELETE /api/restaurants/:id

Step 6: Pushing to Cyclic, Documentation, and Front-End Design:

Ensure application functionality and readiness for deployment.

Deploy the application to Cyclic or a chosen platform and update documentation with deployment details.

Design a JS front-end (jQuery, Native JS, etc.) to interact with the developed API, incorporating Bootstrap for enhanced visuals.

**Team Member 2:**

Step 4: Add Security Features - Part 2

Finalize environment variable usage for sensitive data protection.

Begin implementation of chosen security measures (e.g., JWT, Session, Cookie) within the codebase.

Bonus Tasks:

Implement GraphQL to extract and manipulate restaurant data, enhancing data extraction or manipulation routes within the API