CS572 Final Project

During the pandemic COVID-19, farmers are no longer able to sell their products and customers are no longer able to buy fresh local food.
You will build a web application for farmers to add their products, browse orders, and contact customers when orders are ready for contactless pickup. Customers will shop for products and make orders.

Web Application Workflow

- Farmers will sign up/sign in before they can use the application.
- Main farmers' functionalities are
  - Add, Delete, Update, Retrieve products (as inventory).
  - Orders have 3 status, pending, ready, complete.
  - Farmers can see all orders and filter them by status.
  - Once the order is prepared and ready for pick up, the farmer will update the order status to 'ready', and an automatic email will be sent to customers with the pick-up date and time.
  - Once the order is picked up, farmers will update the order status to 'complete'.
- Customers will sign up/sign in before they can use the application.
- Customers can browse all farmers. Farmers should be sorted by reputation. Check the farmer's reputation algorithm below.
- Customers can browse all products in the inventory for the selected farmer.
- Customers can add products from one farmer at a time to their shopping cart. (An order cannot contain products from more than 1 farmer).
- Customers can check out the shopping cart, then an order should be created with the status 'pending' and update the inventory, an automatic email should be sent to both farmer and customer with the order details.
- Customers can see their orders history and filter by date and status.
- Customers can pick up their orders and pay the farmers.
- After the order status is changed to 'complete', customers may leave an optional rating and feedback. The rating can be one of three values: excellent, good, and bad.

Farmers Reputation Algorithm

The system generates farmers' reputation automatically based on customers rating as following:

An excellent rating adds 1 point to the farmer's reputation score.
A bad rating deducts 1 point from the farmer's reputation score.
A good rating does not change the farmer's reputation score.

Superuser Account (Optional)
The user collection in the database must have a **superuser** account. (Role is **superuser**). Superuser may log in to the Web Application (similar to farmers and customers) and see a dashboard with the following functionalities.

- List all farmers and customers' accounts, activates/deactivates and reset their password.
- List all transactions and filter them by date.
- List all requests in the log file. (check the technical details)

Technical Details (for standard or custom projects)

- All pictures must be uploaded and stored on either Amazon S3 or Google Cloud Storage.
- Use JWT for authentication and authorization.
- You need to follow REST convention to build the server application.
- The backend API documentation contains the following:
    - Entity, HTTP Verb, Request Header and Body, Response Header, and Body (You may use Swagger to generate this API documentation).
- Use may GitHub issue tracking to plan your daily schedule/tasks.
- Use feature branches for each task/issue/feature.
- Log all the backend API requests to a file.

Important Notes

- This is a group project. Create one Repo for the team and add members as collaborators.
- Should you need any help, please contact the teaching faculty between 10 AM - 12:30 PM on Teams.
- Do not push any private key of any service to Git. (If found, your account will be terminated and you may be legally sued by service providers). Write your keys in a config file (npm i dotenv) and add the filename to .gitignore
- Do not spend more than two hours on a problem, move on, or find an alternative.
- A daily push is required for each member to track your code and performance. If you miss a push that will affect your final grade.
- Even though projects will be developed in teams, every team member will be graded individually based on their contribution and their quality of work. Team leads will be responsible for leading, mentoring their teammates and making the project succeed.
- Remember to respect the code honor submission policy. All written code must be original. Presenting something as one's own work when it came from another source is plagiarism and is forbidden.
- Plagiarism is a very serious thing in all American academic institutions and is guarded against vigilantly by every professor.

Technical Video Tutorial (Extra Point +5)
Students may prepare and record 1 technical video tutorial that features a certain functionality in their project. This is optional and will add five extra points to their course grade total.

Example: https://www.youtube.com/watch?v=Ur6MNStwXlQ

Upload the video tutorial to YouTube and send me the video link by email. (Videos can be published as 'public', or 'unlisted' so only those with the link can watch it).

Evaluation Criteria

- REST design conventions
- Authentication/Authorization for all server-side, and client-side applications
- Cloud services integration
- MongoDB design
- Design patterns
- Code quality and logic
- UX/UI, using UI kit
- Proper using of GitHub and daily pushes
- Reusability of the code

Optional requirements

- Use a payment gateway to process credit-cards electronically.
- Deploy the server-side application and the client web application to the Cloud.

Final Evaluation

Project deadline is **Wednesday 12 pm**. Team leads will have to send me an email with the following:

1. List your team members names, email and id.
2. Attach your detailed project plan (daily schedule for each member and what they will be responsible for – tasks/roles). I'll use this project plan to evaluate each member's performance. Every team member is expected to perform at least one daily Git push summarizing their daily contribution.
3. Your project git link.
4. Attach or copy/paste your config file content (all secret keys).

Good luck and happy coding!