**CMPE 273 Lab-3 Report**

1. **Introduction:**

* **Goals:** The goal is to develop the “Prototype of GRUBHUB application” in which restaurant owners can post their menu and manage customer orders, while buyers can search food and place order.
* **Purpose** : The purpose of developing this application was to gain better

knowledge and understanding of GraphQL.

1. **System Design:**

Zookeeper

HTML, CSS,

Javascript

Kafka Backend

Node JS web

server

React JS,

Redux

Kafka

Broker

Express Web

Framework

GraphQL

MongoDB

(Database)

* **Client Side:** User friendly UI has been developed to integrate all the below functionalities into the prototype using react components.
* **Server Side:** Implemented Different functionalities as follows.
* Buyer/Owner Signup.
* Buyer/Owner Signin.
* Buyer/Owner Profile Update.
* Owner can create a section and add item to it.
* Listing items in different sections in Buyer/Owner account.
* **GraphQL:** Implemented graphQL for creating creating APIs.
* **Database Design:** Used MongoDB for storing data. I have created 3 models. They are: Buyer, Restaurant and Items.

**Collections:**

**Buyer**

|  |  |
| --- | --- |
| fname | String |
| lname | String |
| email | String |
| phone | String |
| password | String |

**Restaurant**

|  |  |
| --- | --- |
| name | String |
| res\_name | String |
| email | String |
| phone | String |
| password | String |
| res\_zipcode | String |
| cuisine | String |

**Items**

|  |  |
| --- | --- |
| item\_name | String |
| res\_email | String |
| cuisine | String |
| item\_desc | String |
| item\_price | String |
| menu\_sec | String |
| res\_name | String |

1. **Architecture Diagram:**

**A screenshot of a cell phone

Description automatically generated**

1. **Results:**

Buyer Side Screenshots:

* Signup:

A screenshot of a social media post

Description automatically generated

* Login:

A screenshot of a social media post

Description automatically generated

* Profile:

A screenshot of a computer

Description automatically generated

* Edit Profile:

A screenshot of a computer

Description automatically generated

A screenshot of a cell phone

Description automatically generated

* Menu (Display List of items in different sections):

A screenshot of a cell phone

Description automatically generated

* Restaurant (Owner) Side:

Signup:

A plate of food

Description automatically generated

* Login:

A screenshot of a social media post

Description automatically generated

* Profile and Edit Profile:

A screenshot of a social media post

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a social media post

Description automatically generated

* Add Section and Add Item:

A screenshot of a cell phone

Description automatically generated

* Menu Items Listed as per sections:

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A close up of a logo

Description automatically generated

1. **Questions:**
2. An architecture for using multi-part data in GraphQL without using any open source library from Git.

* Graphql is a query language and is used as alternative of REST for calling api. There are many options for using multi-part data in graphql. Specs are used in mutations to specify the file-type while using multi-part data in GraphQL. On the server side, this data is received and parsed using these specs. It also has keys for mapping the data. The files can be uploaded locally or on object storage like S3.

1. State any open source library for enabling multi - part data transfer using GraphQL

with sample code. Argue why do you think that this particular library is a good fit?

This can be done using apollo-upload-server with express , koa, or happi servers as a middleware or using apollo-upload-server with apollo-server. It is a freely available API which uses resolver function to manage uploading files and this returns a promise. It handles different file types.

**GIT CREDENTIALS:**

**Git Link:** [**https://github.com/Tithi286/Lab3-014344460**](https://github.com/Tithi286/Lab3-014344460)

**Git Username: Tithi286**

**Git Password: Ishan28696**