

## **Deli / Chicken Roll Web App Project**

**CIARAN ROCHE - G00376934**

**MILOSZ MILEWSKI - G00376658**



**Professional Practise in IT**

**Computing in Software Development Year 3 project**

**Supervisor: Martin Hynes**

## Submission Links

### Documentation:

<https://github.com/CiaranRoche203/DeliWebApp>

### Deli App Code:

<https://github.com/CiaranRoche203/Deli-App-Front-End>

<https://github.com/ChickenDeliApp/Deli-App-BackEnd>

### Demonstration Video:

<https://www.youtube.com/watch?v=ieC3LZfCGAQ>

## **Table of Contents**

- 1. Introduction**
- 2. System Requirements**
- 3. Technology used and why**
- 4. Design Methodology Applied and Software Development Life Cycle**
- 5. Architecture of the Solution and Features of the Implementation**
- 6. Limitations and Known Bugs**
- 7. Testing Plans**
- 8. Recommendations for Future Development**
- 9. Conclusion**
- 10. References**

## **Introduction:**

For our project, we decided that we would implement the design of a chicken roll or deli app. We decided on a react application which is connected to an SQL database with a number of different tables. The app will allow the user to create an account, create a review of a deli, search reviews, search for delis, rate and like other reviews and delis. In this document we will talk about the system requirements, technologies used, design methodologies, features, limitations, and bugs and finally recommendations for future development.

Our main objectives were as follows:

- Login system based on information stored in SQL
- Submit reviews
- Rate delis
- Search for delis
- Improve our teamworking skills
- Practise project management using different tools such as Jira
- Practise different aspects of project management by using the likes of:
  - Sprints to complete stages
  - Burndown charts on Jira
  - Using GitHub to store documentation and the project
  - Weekly meetings as a group and with our mentor
  - Weekly documentation of our weeks progress

We have both taken on all aspects of the project. We each did some share for each aspect. Ciaran did the majority of the front end; Milosz did the majority of the back end. For the documentation we each took paragraphs to write and submitted them to the GitHub documentation section. We both also submitted weekly documents to the GitHub documentation section.

- Front end – Ciaran
- Back end – Milosz
- Documentation – Both

**System Requirements:**

As the project is still in early phases and not put online just yet. In order to run the project, you will need the following:

- Open a command prompt by typing cmd
- Clone the repository to your pc or desktop – git clone and the URL
- Install the required modules. You do this by typing npm install
- Then type npm start
- This is needed to be done for both the backend and the front end in order for it to run correctly
- The latest version of node needs to be installed along with an editor such as visual studio code

Once the website is made available online you will not need to follow the steps above as the user will be able to interact with it like any other website on the internet.

We will take you through the user's perspective of the website in a heading below called features of the implementation.

### **Technology Used and Why:**

Below is a list of technologies used. We will give a brief explanation as to why we chose each.

#### **1. JavaScript**

JavaScript is the language that we have decided to use as our programming language for the application. We chose this language as we both have good experience and enjoy immensely working with JavaScript. Secondly, it is a very simple and efficient language to use. There is lots of great resources to use with many different features that we can implement through JavaScript.

#### **2. React**

We decided to use a react application for the project. There were a number of reasons for choosing this. As part of semester one we studied react and both of us have a great interest in it. React seemed the most appealing route to take. Another reason for choosing this was due to our knowledge of React. We both feel confident in our abilities to be able to use it. However, there is still a huge learning scope for us. There is a lot of new things we learned when working on the app. This made the application both rewarding and challenging.

#### **3. Bootstrap**

We used Bootstrap as a way of designing the web application. We chose this as we found there was a lot of unique aspects that were superior to other designs. In terms of front end, it made the styling sleeker and very easy to use. It was very easy to implement as well.

#### **4. MySQL**

We decided that the best database management system is MySQL. We were originally looking at MongoDB but with all the different tables and relationships between said tables, MySQL was clearly the better option for us. We are also learning MySQL for a number of years, so it was very easy to implement and design.

#### **5. Jira**

We used Jira as our project management tool. We chose this as we felt it was the most useable and flexible tool available to us. We could set up sprints, we had a roadmap and backlog. It was also very useful for assigning people tasks. This tool is very useful as it was a guide as to where we are and where we should be on the project.

### **Design Methodology Applied and Software Development Life Cycle:**

We started our project by having a meeting and setting out a timeline for the project. We did this by setting up a Jira board. We first set the project into 4 stages:

- Front end
- Database
- Backend
- Documentation

We used these as a basis for our sprints. We then decided on a design or possible design that we would pursue. We did this by creating some freehand designs and ultimately creating a wireframe. We decided to make 3 GitHub repositories. Front end, Back end and documentation repositories. This made it easier to separate out the work. The reasoning for separating the front end and back end was so that there was no clash between the front end and back end. Also, it was so that we could separate them in order to make the website go live.

#### *Sprint 1:*

We then proceeded to set up the front end. This was done by creating a react app. All the necessary modules were installed and the skeleton for the front end was ready. Setting up and designing the front end was our first sprint. The front-end sprint took approximately 2 to 3 weeks.

In this time, we created the design for:

- Login Page
- Register Page
- Home Page
- Add Chain page
- Add Restaurant Page
- Submit a review Page

We got the design of the pages up and running. We discussed different designs and decided on the best one to use.

#### *Sprint 2:*

Our second sprint was all based around the database. During this time, we designed the Entity Relationship Diagram and then set up the scripts for the SQL database. We both did this over meetings and collaborated to decide on the best design for the SQL tables. This sprint took in total around a week.

#### *Sprint 3:*

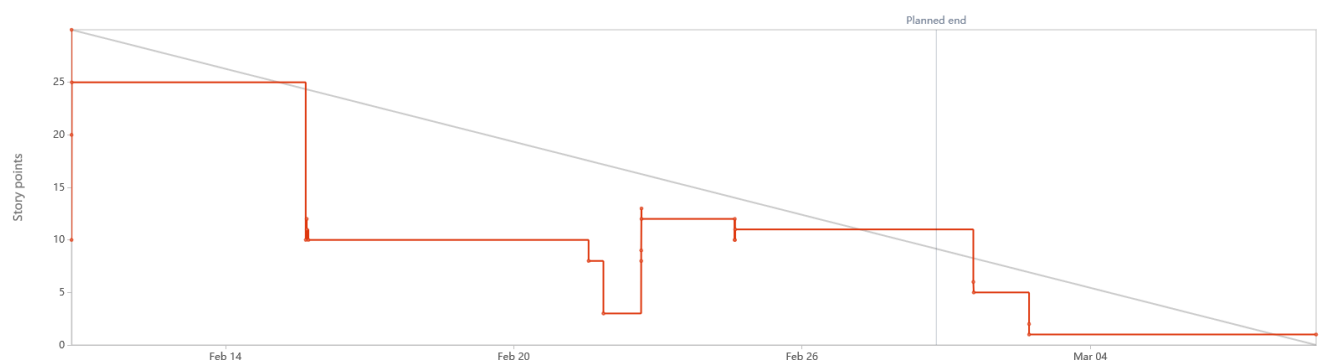
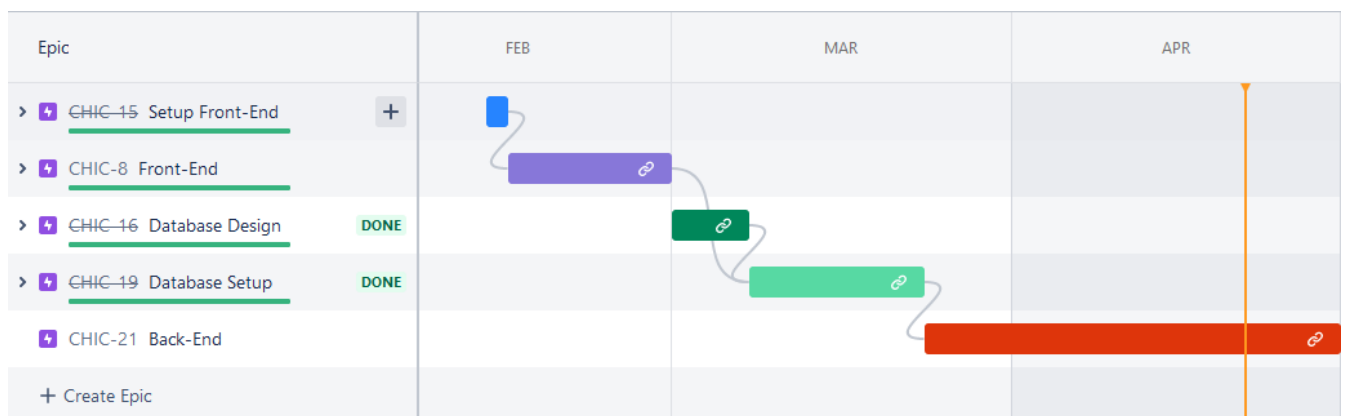
Our third sprint was for the Backend design of the project. This was probably the most complex part of the project with us learning a few new things as we went along. This was essentially where we set up the connection to the front end. This was to link up the entire project and we could see real progress at this point. This was also the point where we updated how the SQL database works. Instead of having a script that created the table, the backend was designed in a way so that it generates it automatically. We will talk about this more in the features section below.

### Sprint 4:

Our final sprint was where we made this document and tied up all the loose ends of the project. This was essentially where the bulk of the work was done but it was now time to clean it up, comment the code more efficiently and see the final submission of our project.

### Jira:

Jira played a vital role in our project. We used this as our main way of setting our goals, timelines and assigning tasks to each other for the project. Jira is where we set up the details for each sprint. We set up tasks that needed to be completed for each sprint and assigned these tasks to each other. When we had completed a task, we would then move that task from the to do list to the done section. This was key to keeping on schedule and tracking what was needed to be done in the remainder of the sprint. Below we have attached a screenshot of the roadmap that was set out for our project. This roadmap feature was also extremely helpful as we could adjust the sprint timelines as we saw fit. For example, we spent a bit more time on the backend than originally planned, so we moved the sprint to start at an earlier date. The other screenshot is of one of the sprint burndown charts that was generated in the reports section of the project.





**Weekly Documents:**

As part of the project we wrote weekly documents for our mentor where we described what we had done during the past week and what we plan to do over the next week. This documented most of our work during the project and kept our mentor informed as to what our progress was looking like.

**Weekly meetings:**

As part of the project we took part in weekly meetings with our mentor on teams. Much like the weekly document, these meetings were done to check our progress and ensure that we were kept on our toes and not let our progress fall. Our mentor kept us on the right track throughout the duration of the project and was a major help on getting the project done in a timely fashion.

### Architecture of the solution and Features of the Implementation:

Here we will give you a look at our project. We will talk about some of the aspects that a user can utilise, and we will also provide some screenshots of the project.

#### **Register:**

The first feature we would like to talk to you about is how the user is first greeted at the opening of the application/ website. This is the register page. In order for the user to fully access and utilise the project/ application, they must register an account. The user is asked to enter their:

- Email address
- Username
- Password
- Date of Birth
- Must tick the acceptance of terms and conditions

The details are then stored on the SQL server. The back end was set up so that we could take these details and store them here. There are a number of constraints such as:

- The email must be unique and a valid email address
- The username must be unique.
- The password must be a minimum of 5 characters and maximum of 60.

The error messages displayed show the constraints that must be met. Below you can see a number of images from the register page detailing all the above.

## Register a New Account

Email

roheciar37@gmail.com

Username

rohe1

Password

●●●●●●●●●●

Date of Birth

06 / 04 / 2021

A-Z, numeric, 5-60 characters long

☒ I agree to the Terms of Service

Proceed

## Register a New Account

Email

roheciar37@gmail.com

Username

roche

Password

••••



Must be between 5 and 60 characters long

A-Z, numeric, 5-60 characters long

Date of Birth

14/04/2021

☒ I agree to the Terms of Service

Proceed

Invalid password

```
mysql> select * from users;
```

username	email	birth	password	isAdmin	joinDate
roche1	roheciar37@gmail.com	2021-04-06	\$2b\$04\$VZPDHxTRqWwVdsqPZnmFi.RXfjtZq.LecXw4wqHoGwRlNnnNZ1DKmu	0	2021-04-16 20:12:48

1 row in set (0.04 sec)

Registered user in database.

### Login:

The next feature is the login page. The user has just registered their account. They may now navigate to the login page. Here they will be asked to enter their email and password. The user must provide the correct details in order to successfully login. These details are stored in the user table that was created on the back end of the project. Only details that are stored here in that database may be used to log in. Once logged in, this creates a session for the client and ends when the user is logged out or the tab is closed. Below is a couple of screenshots of our login system.

## Account Login

Email

Password

[Proceed](#)


### Home Page:

The Home page is the users landing page once they are successfully logged into the account. Here the user can see all the relevant information to them. The first section of the page is a list of the months 10 best rated deli places. The second section is for newest reviews. These will dynamically update as new reviews are added to the database in the back end. Below is a couple of screenshots of the home page. You can also see in the second screenshot that a user is logged in. This is done by creating a session using SQLite. Once the user logs out, or after 1 hour, the session is ended.

### TOP 10 Reviewed Delis (Monthly)


Rank	Restaurant Name	Chain	Rating
1	Duggans Spar	Duggans Spar	2.5 (Based on 2 review)

### Newest Reviews



**Duggans Spar || Duggans Spar**  
Review by rako99 • 1 stars  
Bad deli, not yum :(

4/23/2021



**Duggans Spar || Duggans Spar**  
Review by rako99 • 4 stars  
Very nice deli, chicken is delicious

4/23/2021

Welcome, rako99

### **Post Review:**

This is the page where a user can submit a review for a deli. The user is required to enter the deli name and enter details about their review. The User selects the restaurant they want to review from a dropdown list, can enter the details of the review in the text box labelled review and use the scroller to set their rating of the review. The proceed button will then submit said review. The user can also use the map to find the restaurant they want to review. The details of the review are posted to the table for reviews in the back end of the application.

Successfully created a new Review!

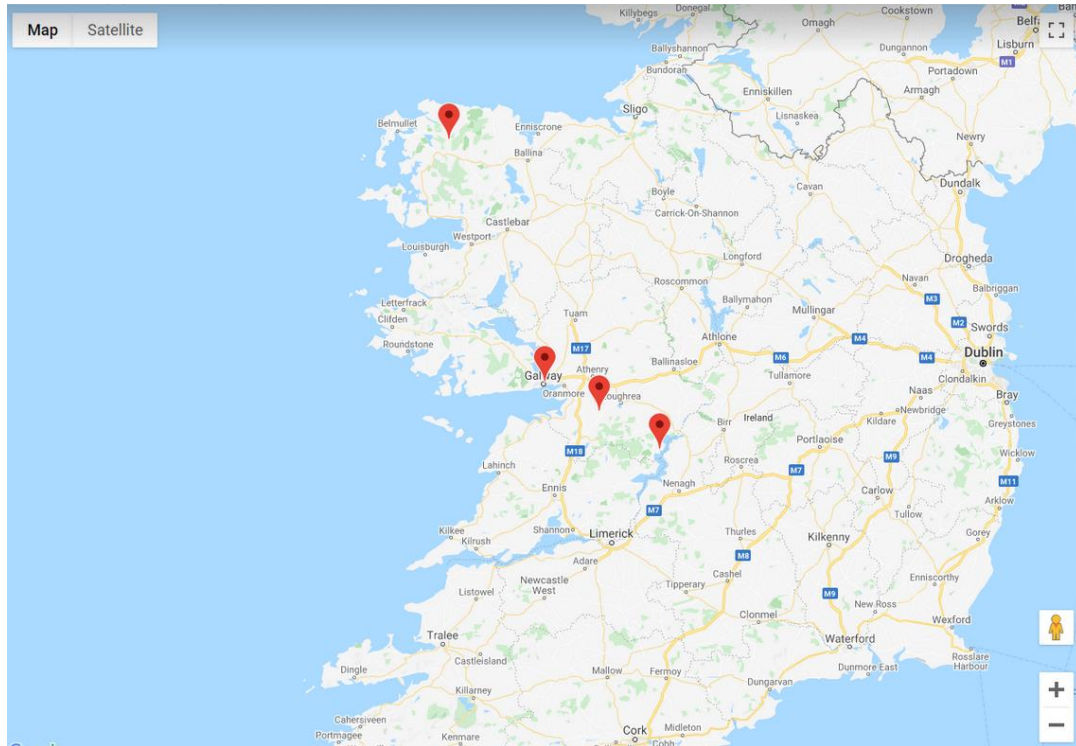
## Post a new Review

Restaurant

Review

Rating

Proceed



### Add a new Chain:

The user can add a restaurant chain. On the right of the screenshot you can see the existing chains. This is where the front end connects with the back end and takes all the chains that exist in the database and display it here. The chain of restaurants essentially is where we say what franchise or company a shop is belonging to. The resulting chain will also be needed in order for the user to add a restaurant to the database.

### Add a new Restaurant Chain

Restaurant Chain Name

By submitting of a Restaurant Chain you acknowledge our Terms of Service agreement.

Submit 😊

### Existing Restaurant Chains

Before adding a new Restaurant Chain, please ensure it does already not exist.

Chain Name	Claimed by
No data 😞	

### Add a new Restaurant Chain

Restaurant Chain Name

By submitting of a Restaurant Chain you acknowledge our Terms of Service agreement.

Submit 😊

### Existing Restaurant Chains

Before adding a new Restaurant Chain, please ensure it does already not exist.

Chain Name	Claimed by
Apache Pizza	Nobody
Duggans Spar	Nobody
McDonalds	Nobody

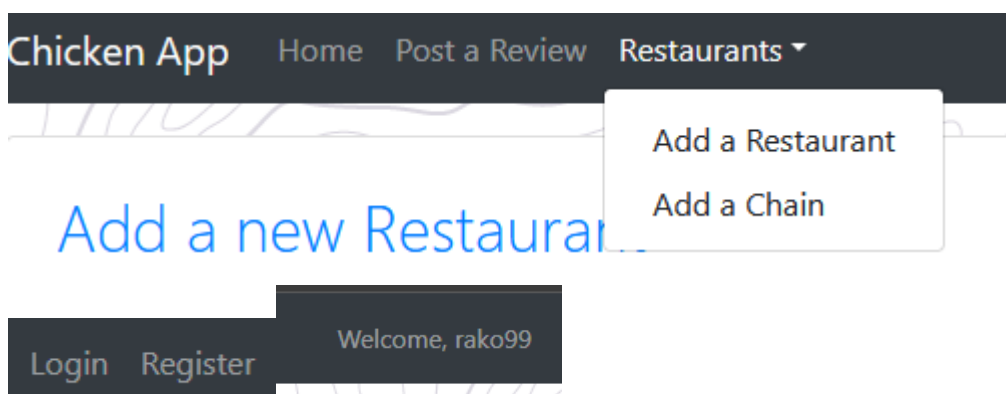
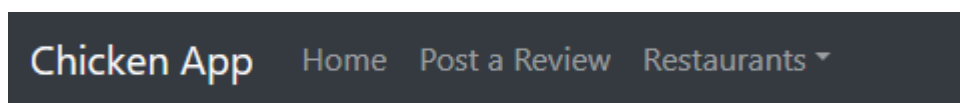
< 1 >

### Add a new Restaurant

We have given the user the option of adding a new restaurant/deli to database through the web application. The user will need to provide the restaurant name, the chain it belongs to, the co-ordinates of the restaurant (this can be done by interacting with the map itself) and the address of the restaurant. After this has been done, the user can then submit a review of the newly added restaurant to the database in the back end.

### Navigation Bar:

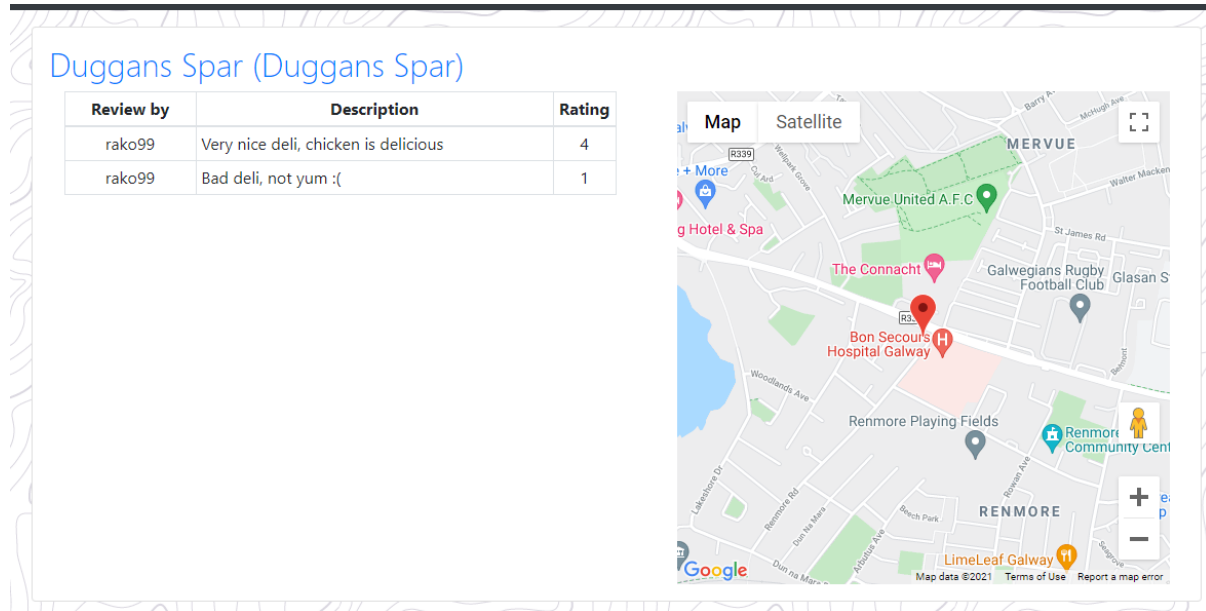
The navigation bar is a feature we have implemented that can be seen at the very top of the application. This allows the user to navigate around the website with efficiency and ease. The Links on the navigation bar bring the user to the different pages on the application. Below are the screenshots of the navigation bar. Also, there is two screenshots of the navigation bar, one without a user logged in and one with a user logged in.





### Individual Restaurant Page:

The individual restaurant page is where the user can see the list of reviews associated with said restaurant. As you can see from the screenshot below this particular deli has 2 reviews, one good, one not so good. The list of reviews for a restaurant are got from the back end of the application and displayed here on the front end. As a new review is added, it will be added to the back-end database and then, once refreshed, will be visible on the individual restaurant page.



### Database design:

We designed the database in the back end of the project. We essentially created a model in the back end that creates the database and associated tables when the server starts up. We have 4 tables in our database. They are as follows:

- Users
- Deli Chain
- Deli Restaurant
- Review

The Deli chain can have many Deli Restaurants, a Restaurant can have many reviews. A user can make many reviews.

- Users ---<Review
- Deli Chain ---< Deli Restaurants
- Restaurant ---< Reviews

Below you can see the structure of the user table. The primary key is the email. This ensures that when a user registers that an email can only be used once. The username is set to unique in order to avoid duplication. We have set the type to varchar for both of these as the lengths of emails and usernames can vary. Birth is set as a type date. Password is set to varchar 60. The reason being is that the password is hashed in the back end. So, when the data is inserted into the SQL tables, the password comes back hashed as seen in the screenshot above where we discuss the register page.



We have set the isAdmin to a Boolean value. A simple yes or no is all that is needed. This can also be null.

```
mysql> describe users;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| username | varchar(50) | NO | UNI | NULL |  |
| email | varchar(255) | NO | PRI | NULL |  |
| birth | date | NO |  | NULL |  |
| password | varchar(60) | NO |  | NULL |  |
| isAdmin | tinyint(1) | YES |  | 0 |  |
| joinDate | datetime | NO |  | NULL |  |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.05 sec)
```

Users table.

Much of the design is similar in terms of types throughout so we will just take you through some of the types that we haven't discussed so far that are new in the other tables. Below is the deli restaurants table. As you can see there is a row called geo\_location. It is marked with a type point. The reason for this is quite simple. This is where the location of a deli is stored. It will be stored in the form of co-ordinates. Point allows us to use the x and y coordinates from a map. It is a geometry data type.

```
mysql> describe delirestaurants;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id | int(11) | NO | PRI | NULL | auto_increment |
| restaurantName | varchar(80) | YES |  | NULL |  |
| geo_location | point | YES |  | NULL |  |
| address | varchar(255) | NO |  | NULL |  |
| postCode | varchar(10) | NO |  | NULL |  |
| createdAt | datetime | NO |  | NULL |  |
| restaurantChain | int(11) | YES | MUL | NULL |  |
+-----+-----+-----+-----+-----+-----+
```

Deli restaurants table.

### **Back End:**

On the back end there are some very cool features implemented. The first is that of how we create the database. Instead of using scripts and running them physically ourselves in say WAMP or MySQL workbench, we actually create the tables in the back end of the database. Below is a screenshot of the code in question. We used an ORM for converting the SQL database into a mongo style database so that we could perform queries with more ease and efficiency.

```

require("dotenv").config()
const { Sequelize, Model, DataTypes, DECIMAL, INTEGER } = require("sequelize")
const bcrypt = require("bcrypt")

// https://github.com/sequelize/sequelize/issues/9786
const wkx = require('wkx')
Sequelize.GEOMETRY.prototype._stringify = function _stringify(value, options) {
  return `ST_GeomFromText(${options.escape(wkx.Geometry.parseGeoJSON(value).toWkt())})`;
}
Sequelize.GEOMETRY.prototype._bindParam = function _bindParam(value, options) {
  return `ST_GeomFromText(${options.bindParam(wkx.Geometry.parseGeoJSON(value).toWkt())})`;
}
Sequelize.GEOGRAPHY.prototype._stringify = function _stringify(value, options) {
  return `ST_GeomFromText(${options.escape(wkx.Geometry.parseGeoJSON(value).toWkt())})`;
}
Sequelize.GEOGRAPHY.prototype._bindParam = function _bindParam(value, options) {
  return `ST_GeomFromText(${options.bindParam(wkx.Geometry.parseGeoJSON(value).toWkt())})`;
}

const sequelize = new Sequelize({
  dialect: "mysql",
  host: process.env.HOST || "localhost",
  port: process.env.PORT || 3306,
  database: "chicky",

  username: process.env.DB_USERNAME || "root",
  password: process.env.DB_PASSWORD || ""
})

class User extends Model {
  verifyPassword(password){
    return bcrypt.compareSync(password, this.password);
  }
}

User.init({
  username: {
    type: DataTypes.STRING(50),
    allowNull: false,
    unique: true
  },

  email: {
    type: DataTypes.STRING(255),
    allowNull: false,
    primaryKey: true,

    validate: {
      isEmail: {
        msg: "Must be a valid e-mail"
      }
    }
  },

  birth: {
    type: DataTypes.DATEONLY,
    allowNull: false
  },

  password: {

```

```

    password: {
      type: DataTypes.STRING(60),
      allowNull: false
    },

    isAdmin: {
      type: DataTypes.BOOLEAN,
      defaultValue: false
    }
  }, {
    sequelize,
    createdAt: "joinDate",
    updatedAt: false,
    timestamps: true
  })

class DeliChain extends Model {}
DeliChain.init({
  chainName: {
    type: DataTypes.STRING(80),
    allowNull: false,
    primaryKey: true
  },

  // managedBy: {
  //   type: DataTypes.STRING(255),

  //   references: {
  //     model: User,
  //     key: "email"
  //   }
  // }
}, {
  sequelize,
  updatedAt: false
})
// User.hasOne(DeliChain, {foreignKey: "managedBy"})
DeliChain.belongsTo(User)

class DeliRestaurant extends Model {}
DeliRestaurant.init({
  restaurantName: {
    type: DataTypes.STRING(80)
  },

  // restaurantChain: {
  //   type: DataTypes.INTEGER,

  //   references: {
  //     model: DeliChain,
  //     key: "id"
  //   }
  // }
}, {
  sequelize,
  timestamps: false
})

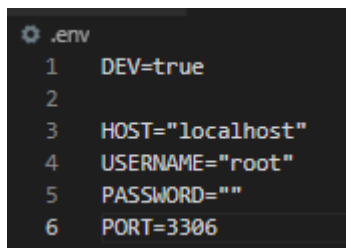
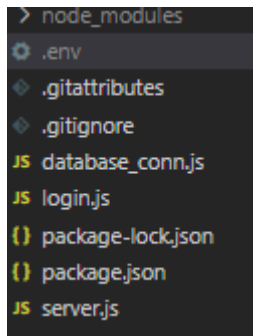
```

We used Passport JS as a way of creating the login system. Passport JS is essentially an authentication middleware for Node JS. It is used for authenticating requests which it does through plugins. It is very flexible and efficient and was a great help for us on this project.

## **Limitations and Known Bugs:**

### ***Database Error:***

One error that was occurring for us when we were making this project was to do with database connectivity. The error was to do with us having different credentials on our machines. For example, we both use root as the host. One of us has a password, the other does not. This would then sometimes cause errors when trying to connect to the database. However, we did find a very quick and easy fix. What we found is that if we added a .env file, we could then set our credentials there, without ever interfering with the other persons. We have a screenshot below to show you.



### ***Front end to back end connectivity:***

When in a meeting displaying our work so far, we encountered an error that refused us connection to the database. The error occurred as we had not plugged up the form correctly. In essence, we tried to register a user without a date of birth as we had not correctly set up the date of birth. We managed to fix the error fairly quickly but was still another error that we had come across.

### ***Node JS***

An error that we encountered in the project was to do with Node JS. The node we were using was 12.18 version. When we tried to submit something on the front end it would bring up an error. The reason being for this was that the version of node we were using did not support the functionality we were trying to implement. As a result, we had to download the latest version of node. This then fixed the error but certainly did cause some trouble and slowed down the progress made that day ever so slightly.

Most of the errors that we encountered were resolved within a reasonable rate of time and some were fixed very easily. However, these bugs still caused some trouble and slowed down our progress ever so slightly.

## Testing Plans:

As part of our testing plans, we mainly used white box testing. This was as we both had a knowledge of what was happening with the code and knew about the internal structure of the application. We constantly tested the project throughout each sprint and tried to curb as many errors as possible as they happened.

### White Box Testing:

White box testing is where the user essentially has a knowledge of the code. As both of us were working on the project we both had a great knowledge of the code. The code was visible to us at all times. This was a good way for us to test things as we knew how we could go about essentially breaking the application so we could try and find bugs. As it was us who coded up the project, we knew beforehand, when testing, what would or could potentially break the application. This meant our testing of the application was quick, efficient and accurate as we spent a lot of time testing every possible scenario.

### Test Case Plan:

We created a test case plan. This is so we would write out all the different tests that we would carry out. In this plan we wrote down every test that was performed and their outcome. This was done as a way of monitoring any bugs that appear and we would then focus on fixing them. Below is a screenshot of a plan that we have written. Test cases were written for every aspect of functionality on the website. We also created a Business requirements document to go with this. Both of these can be seen on the documentation GitHub.

### Selenium:

We plan to test the application after all loose ends have been tied up with selenium. We have been studying selenium as part of the module Software Testing and we think it would be appropriate to incorporate a tool that is fresh in our minds and seems very relevant to the testing that will be at hand with this application.

We have attached links to our GitHub repositories at the end in the references section. If you would like to take an in depth look at the test cases, they are in the section under Project Doc.

				<b>Purpose:</b> This set of tests is intended to check main functionality developed									
<b>Login validation For Deli Web App</b>				Microsoft Internet Explorer 9 Windows 7 Microsoft Internet Explorer 10 Windows 7 Microsoft Internet Explorer 11 Windows 7 Microsoft Internet Explorer 8 Windows XP Microsoft Internet Explorer 7 Windows XP Firefox latest version Apple Safari Windows XP									
<b>Requirement ID</b>	<b>Sprint</b>	<b>Test Case Ref</b>	<b>Test Case Name</b>	<b>Descriptions / Instructions</b>	<b>Checks</b>	<b>Expected Result</b>	<b>Actual Result</b>	<b>Results Pass/Fail</b>	<b>Bug #</b>	<b>Comments</b>			
1.00	1	TC.001	Login Heading	GUI: 1. Launch Register page	Panel displays in middle of webpage	Panel is displayed centrally	Panel is displayed centrally	Pass					
2.00	1	TC.002	Nav Bar is present	GUI: 1. Launch Home/login/register page	Nav bar is present at top of page	Nav bar is there with working links	Nav bar is there with working links	Pass					
3.00	1	TC.003	Register	GUI: 1. Create valid password and username 2. Use a valid email. 3. Select a date of birth 4. Accept the terms and conditions button	Successful register	Successful register, be able to login	Successful register, be able to login	Pass					
4.00	1	TC.004	Login	GUI: 1. Click to the login page link 2. Enter a valid email address 3. Enter a valid password	Successful login	redirect to home page	redirect to home page	Pass					

**Recommendations for Future Development:**

There are a number of different ways on which we can enhance the project as follows:

On the front end of the application, we can always make the design more modern, sleek and eye catching for the user. At the moment we are very happy with the design, however, in future we may decide that we would like to change the colour scheme. This is easily done but for the moment and sake of the project it was not necessary. We can also add new features in the future. We will undoubtedly have new ideas that we would like to add or pursue and would be a fun way to pass time.

On the database end we can always add more tables. This of course will depend on us adding new features to the website.

We can always look at expanding the website. In the future we don't necessarily have to limit it to chicken rolls or delis and restaurants. It is an area that has a wide variety and so many different types of cuisines out there. For example, we can always adopt an approach at looking at cafes, bakeries and more. The website can always expand into other areas and become like a TripAdvisor.

In terms of software, there is always a possibility for upgrading our code per say. There may be an efficient new way at doing things that we discover and may decide to adopt that into our project.

### **Conclusions - what you learned from this project:**

Overall, we are extremely happy with the outcome of the project. We found that we worked very well together as a team. We organised ourselves well, kept as close to the project plan as possible, kept a good schedule of getting a certain amount of work done each week and had a great time developing the project. We believe that we achieved our goals that we set out from the beginning by making a functional and fun to use deli application. We greatly improved our “soft skills” such as communication skills and teamwork skills.

We also made a great improvement in our coding skills and using tools such as Jira. We both learned lots of cool new features that we can adapt in future projects. We also both learned from each other as we both brought different yet useful skills to the table. We learned a lot of new skills from working with the different pieces of software that we incorporated into the project. GitHub and Jira taught us both different things that we can now use in future projects.

The experience of working as a team was extremely useful. It was something that we both have greatly improved at now. As the project developed, we grew more confident in our abilities to work closely together and ensure of a good end product.

The guidance we got on a weekly basis from our mentor was extremely helpful. Knowing that we were on the right track in terms of a time frame and that we were executing the project correctly was very reassuring and gave us both more confidence.

In our opinions the project was a success. It is something that we can always look to improve on in the future. Both of us would be very excited to continue working on this project some more and even working on new projects together in the future.

### **Learnings/Findings:**

- Improved soft skills such as teamwork and communication
- Improved coding skills
- Front end and back end connectivity – Great improvements in our ability to do this
- Improved project planning skills
- We worked very well as a team
- We were good at keeping on schedule
- Learned new techniques – ORM for databases, Passport JS etc
- Thoroughly enjoyed working on this type of project
- Have great experience going forward in how to prepare and carry out future projects

**References and links to GitHub**

<https://github.com/CiaranRoche203/DeliWebApp>

<https://github.com/CiaranRoche203/Deli-App-Front-End>

<https://github.com/ChickenDeliApp/Deli-App-BackEnd>

<https://reactjs.org/docs/create-a-new-react-app.html>

<http://www.passportjs.org/>

<https://getbootstrap.com/>