Global Distributed Software Development
Master Team Project WS 2022
“ReservEat”
Milestone 4
Team 1:

Utsav Shrestha
Team Lead, Frontend Lead
utsav.shrestha@informatik.hs-fulda.de

Muhammad Bilal
Frontend Developer
muhammad.bilal@informatik.hs-fulda.de

Sanjay George
Backend Lead, DevOps
sanjay.varkey-george@informatik.hs-fulda.de

Omer Zogubi
Backend Developer, GitHub Admin
omer.zogubi@informatik.hs-fulda.de

Parthiv Jani
Frontend Developer
parthiv-yogesh-kumar.jani@informatik.hs-fulda.de

Kristijan Lazeski
Backend Developer, Database admin
kristijan.lazeski@informatik.hs-fulda.de

Date Submitted	Date Revised	Revision Summary

1. Product Summary

Introducing ReservEat, the ultimate restaurant booking website that will change the way you dine out! ReservEat is a vendor for multiple restaurants, giving you access to a wide range of dining options. With ReservEat, you can easily search for restaurants by name, cuisine, food, and rating. Our search results can be filtered and sorted, making it easy to find the perfect restaurant for your taste.

ReservEat allows you to reserve a table and specify the number of seats at any given restaurant. You can reserve one or multiple tables, and the number of booked seats is specified per table. Additionally, every booking is associated with an account, so you can keep track of all your reservations in one place.

Our platform allows restaurant owners to add, remove, and update their restaurant information. You can view menus of the restaurants and post reviews of your dining experience. You can also chat with the restaurant host, making it easy to communicate any special requests or preferences.

With ReservEat, you can cancel or change your reservation up to 2 hours before the reservation time. This grace period ensures that you have the flexibility you need in case your plans change. Additionally, for every reservation you make with ReservEat, you accumulate reward points that can be redeemed for discounts at participating restaurants.

We understand the importance of safety and security when it comes to your personal information. That's why one account can only be associated with one user, ensuring that your data is kept safe and secure.

Experience the convenience of ReservEat and discover new dining experiences today. Start searching for your next dining destination with ReservEat!

List of Functional Requirements:

ID	Functional Requirement	Status (Pass/Fail)
1	The Application is a Vendor for Multiple Restaurants	
2	User can Search for Restaurants by name, cuisine, food, and rating	
2.1	Search results can be filtered additionally and sorted	
3	User can reserve a table and specify the number of seats in a given Restaurant	
3.1	User reserves one or Multiple Tables at the Restaurant	

3.2	User specifies the number of seats	
4	Restaurants can be added, removed and updated by Restaurant owners	
4.1	Restaurants can be Added by the Restaurant Owners	
4.2	Restaurants can be Removed by the Owners of given Restaurant	
4.3	Restaurants can be Updated by the owners of said Restaurant	
5	There is a Grace Period to Cancel or update the booking of the table	
5.1	Every User can Cancel the reservation up to 2h before said reservation	
6	The number of booked seats are specified per table	
7	Users can view Menus of the Restaurants	
8	Users search for restaurants, make/change/cancel reservations and also post reviews.	
8.1	User can cancel Reservation	
8.2	User can change Reservation	
8.3	User can Post Reviews	
9	User can View the Restaurant Reviews	
10	Every booking is associated with an account	
11	One Account can not be associated with multiple users	
12	User can chat with the Restaurant Host	
13	User accumulates Reward Points for every reservations with the application	

2. Usability Test Plan

Objective: The objective of this usability test is to evaluate the ease of use, functionality, and user experience of the ReservEat website.

Participants: The test participants will consist of 6 individuals who frequently dine out and are comfortable using technology. Participants will be chosen based on their accessibility and willingness to take part on the test with restaurant booking websites.

Tasks:

1. Search for a restaurant by name, cuisine, food, and rating.
2. Filter and sort the search results to find a suitable restaurant.
3. Reserve a table and specify the number of seats at a given restaurant.
4. Cancel or change a reservation up to 2 hours before the reservation time.
5. Post a review of a restaurant and view other restaurant reviews.
6. Chat with a restaurant host to communicate any special requests or preferences.
7. Accumulate reward points for every reservation with the application.

Metrics:

1. Success rate: The percentage of participants who are able to complete each task successfully.
2. Time on task: The time taken by participants to complete each task.
3. Error rate: The number of errors made by participants while completing each task.
4. Satisfaction: The overall satisfaction of participants with the ReservEat website.

Procedure:

1. Brief the participants about the purpose of the test and provide them with a consent form.
2. Ask the participants to complete each task while thinking aloud, and record their responses.
3. Collect metrics such as success rate, time on task, error rate, and satisfaction score for each task.
4. After the completion of all tasks, ask participants to provide feedback on the website's usability, functionality, and user experience.
5. Debrief participants and thank them for their time.

Analysis: Analyze the data collected from the test and identify any areas of improvement for the website. Use the feedback provided by participants to make necessary changes and modifications to enhance the website's usability, functionality, and user experience.

3. QA test plan

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Status (Pass/Fail)
1	Search for a restaurant by name	1. Enter restaurant name in search bar	Search results should display the restaurant		
		2. Click on search button			
		3. Verify that the correct restaurant is displayed in results			
2	Filter search results by cuisine type	1. Select a cuisine type from the filter dropdown	Search results should display restaurants of that cuisine		
		2. Verify that only the selected cuisine type is displayed			
3	Sort search results by rating	1. Select "rating" from the sort dropdown	Search results should be sorted by rating		
		2. Verify that the highest rated restaurants are displayed			
4	Reserve a table at a restaurant	1. Select a restaurant from search results	The reservation page should be displayed		

		2. Enter the number of seats needed	The reservation should be confirmed		
		3. Click "reserve table" button			
		4. Verify that reservation is confirmed			
5	Cancel or change a reservation	1. Click on "My Reservations" tab	The user's reservation history should be displayed		
		2. Select the reservation to be cancelled or changed	The reservation details should be displayed		
		3. Click on "cancel" or "change" button	The reservation should be cancelled or changed		
6	Post a review of a restaurant and view other restaurant reviews	1. Select a restaurant from search results	The restaurant page should be displayed		
		2. Click on "write a review" button	The review form should be displayed		
		3. Enter the review details and click on "submit" button	The review should be posted and displayed on the page		

		4. Verify that the posted review is displayed on the page			
7	Chat with a restaurant host	1. Select a restaurant from search results	The restaurant page should be displayed		
		2. Click on "chat with host" button	The chat window should be displayed		
		3. Enter the chat message and click on "send" button	The chat message should be sent and displayed on the page		
8	Accumulate reward points for every reservation	1. Make a reservation at a restaurant	Reward points should		

4. Code Review

1. Kristijan
    a. Add Hochschule email validation and encrypt password Done

```
43  -            const accessToken = jwt.sign(data[0], "" + process.env.ACCESS_TOKEN_SECRET);
44  -            res.send({ msg: 'Login successful', data: data[0], accessToken: accessToken });
    51 +         const validPass = await bcrypt.compare(password, data[0].password);
```

kikolazeski Pending

this should be done in the login function not here.

Reply...

```
    52 +         if (validPass) {
    53 +             const accessToken = jwt.sign(data[0], "" + process.env.ACCESS_TOKEN_SECRET);
    54 +             delete data[0].password;
```

kikolazeski Pending

why is password getting returned from the login function data

Reply...

```
    55 +             res.send({ msg: 'Login successful', data: data[0], accessToken: accessToken });
    56 +         } else {
    57 +             res.status(500).send({ msg: 'Wrong password' },);
    58 +         }
45  59       } else {
46  60           res.status(500).send({ msg: 'email or password is in incorrect' },);
47  61       }
```

```
31  32  };
32  33  const login = (email, password) => {
33      -   const query = `SELECT us.id, us.firstName, us.lastName, us.phonenumber, us.email, r.name as role FROM users us INNER J
            = '${password}'`;
    34  +   const query = `SELECT us.id, us.firstName, us.lastName, us.phonenumber, us.email, us.password, r.name as role FROM use
            ='${email}'`;
```

kikolazeski Pending

check the password decryption here. and check if password and email matches and then query this

Reply...

```
34  35      return new Promise((resolve, reject) => {
35  36          pool.query(query, function (error, results) {
36  37              if (error) reject(error);
```

11 package-lock.json

2. Utsav Shrestha
    a. Implement restaurant time (Pull request)

omerzogubi requested a review from **Utsav170** last week

Utsav170 reviewed last week

View changes

libs/database/restaurants.js

```
+   const query = `
+           SELECT *
+           FROM restaurantTime
+           WHERE restaurantID=${restaurantID}
```

Utsav170 last week

and you need to add `availability=true`

Reply...

Resolve conversation

libs/database/booking.js Outdated

```
-
+   var tableBookedSeats = `SELECT seatsBooked FROM restaurantTime WHERE restaurantId =${restaur
+   var totalSeats = tableBookedSeats + numberOfSeats;
+   const updateRestaurantTime = `UPDATE restaurantTime SET seatsBooked = '${totalSeats}' WHERE
```

Utsav170 last week

here first you need to get the number of seats available in a restaurant and then add the condition if the totalNumberOfSeats >= totalSeats only after that you should update here

omerzogubi last week Author

But you should not allow to have that in the FE, there is an endpoint to get seats for each time for the restaurant you can check and allow/disallow user from selecting it, we can have a safety check yes, but I guess user shouldn't progress if there is no place for that hour

Reply...

Resolve conversation

libs/database/booking.js Outdated

```
22  24      return new Promise((resolve, reject) => {
23  -         pool.query(sql, (error, result, fields) => {
    25  +     pool.query(sql, updateRestaurantTime, (error, result, fields) => {
```

Utsav170 last week

it cannot take two querys remove updateRestaurantTime from here

Utsav170 last week

create a new pool.query and execute that updateRestaurantTime

Reply...

Resolve conversation

libs/database/booking.js Outdated

```
42  +
43  +   var reservationNumberOfSeats = `SELECE numberOfSeats FROM reservations WHERE userId = '${use
44  +   var reservationRestaurantID = `SELECE restaurantId FROM reservations WHERE userId = '${user
45  +   var reservationTime = `SELECE time FROM reservations WHERE userId = '${userId}' AND id = ${
```

Utsav170 last week

first thing select spelling is wrong. second you can get all of the above in the same query `SELECT time, numberOfSeats, restaurantId  FROM reservations WHERE userId = '${userId}' AND id = ${reservationId}'`

Reply...

Resolve conversation

libs/database/booking.js Outdated

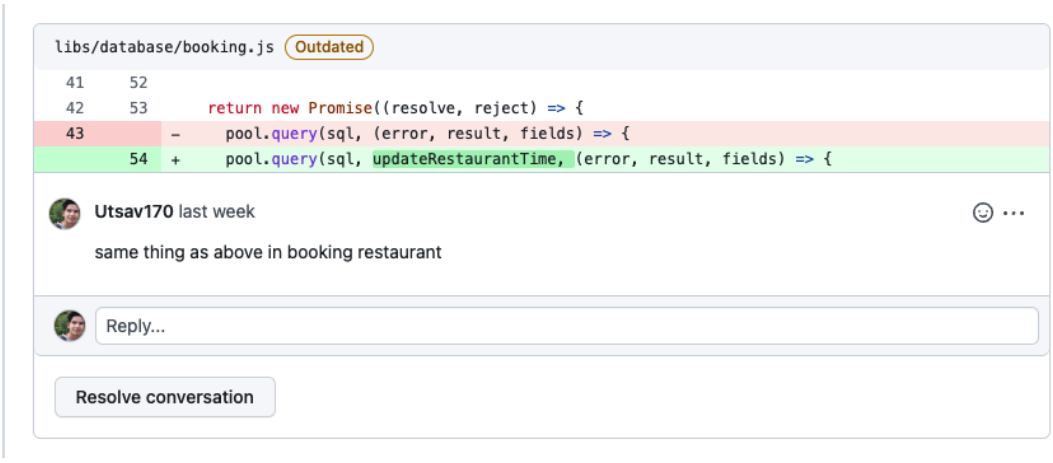
```
44  +   var reservationRestaurantID = `SELECE restaurantId FROM reservations WHERE userId = '${user
45  +   var reservationTime = `SELECE time FROM reservations WHERE userId = '${userId}' AND id = ${
46  +   var tableBookedSeats = `SELECT seatsBooked FROM restaurantTime WHERE restaurantId =${reserva
47  +   var totalSeats = tableBookedSeats - reservationNumberOfSeats;
```

Utsav170 last week

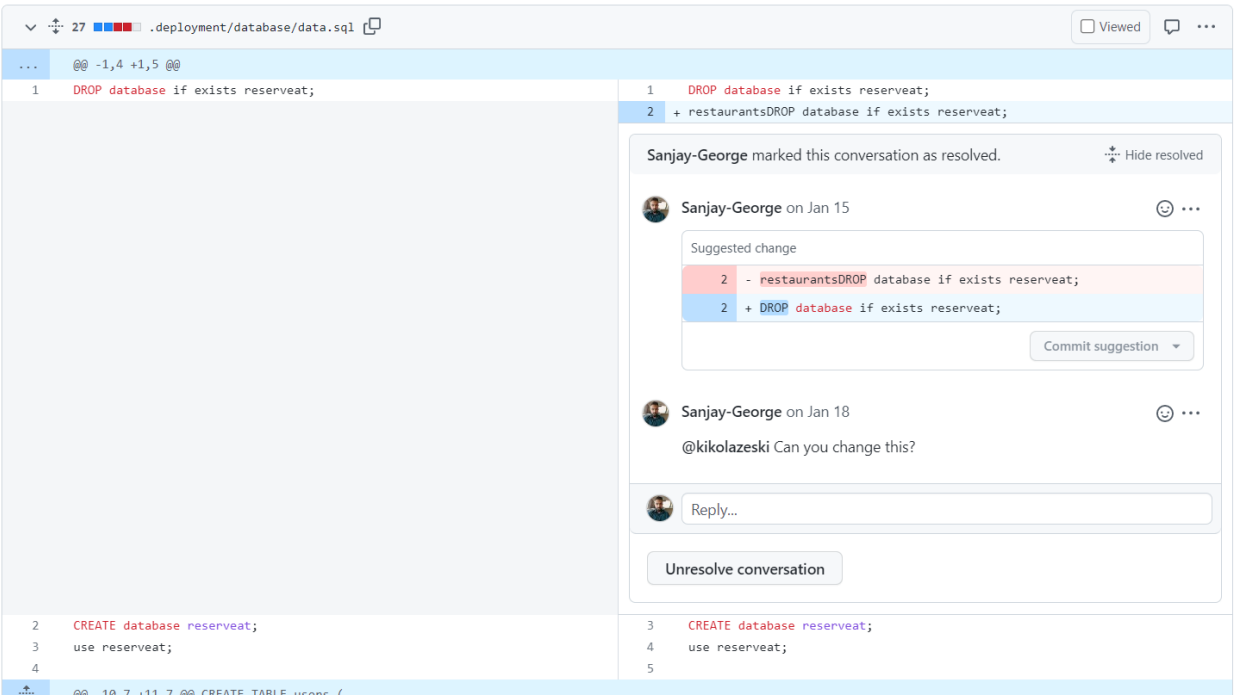
here `reservationNumberOfSeats` this query is not yet executed yet. execute the query first and then from the results you can get it.

Reply...

Resolve conversation



3. Sanjay George
    a. Review on Pull Request containing backend changes to add new restaurants



libs/controllers/store.js Outdated

Hide resolved

```
+     const { id, name, plz, street, town, cuisine, seatAmount, seatsFree, text, timeSlot, time } = req.b
+     const sql = `INSERT INTO restaurants (name, location, cuisine) VALUES ('${id}', '${name}', '${plz}'
+
+     connection.query(sql, (error, res) => {
+         if (error) throw error;
+         res.send({ message: 'Restaurant added successfully.' });
+     });
```

Sanjay-George on Jan 18

move this into restaurants DB. Files in controller should only handle requests and return responses.

Reply...

libs/controllers/store.js Outdated

Hide resolved

```
-         console.error(ex);
-                 res.sendStatus(500);
-     }
+ router.post('/add-restaurant', (req, res) => {
```

Sanjay-George on Jan 18

Keep the path : `POST /api/restaurants/` which means that we are POSTing to restuarants entity (basically creating a new restaurant).

Sanjay-George on Jan 18

Also, I think we already have a controller and DB file for restaurants, we can use that itself. All CRUD operations on restaurants can be in thoese files

dbms.txt Outdated

Hide resolved

```
-     Role varchar(255),
-     Password varchar(255),
-     Phonenumber varchar(255)
+ restaurantsDROP database if exists reserveat;
```

Sanjay-George on Jan 15

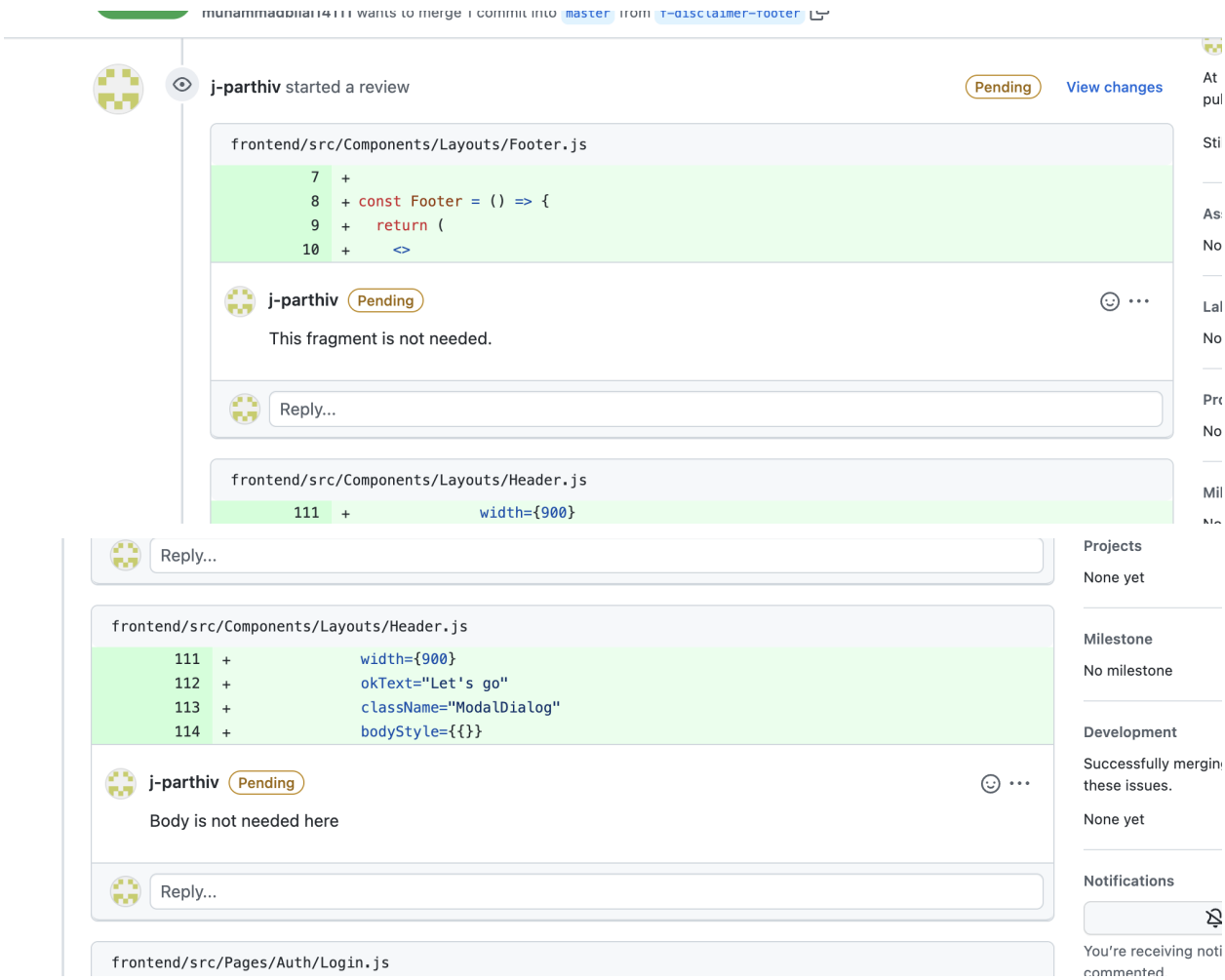
We can get rid of this file, then you don't have to update both files.

Sanjay-George on Jan 18

@kikolazeski Let's delete this file. It's redundant

Reply...

4. Parthiv Yogesh Kumar Jani
    a. Add Disclaimer Message and Responsiveness



5. Muhammad Bilal
    b. Make User Profile

24 frontend/src/Pages/User-Profile/profile.js

Viewed

```
@@ -0,0 +1,24 @@
+ import { Image } from "antd";
+ import "./profile.css";
+
+ const profile = () => {
+   return (
+     <div className="profile">
+       <div className="profile-image">
+         <Image
+           width={200}
+           src="https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png"
+         />
+       </div>
+       <div className="user-data">
```

muhammadbilal14111 Pending

Make this dynamic. Dont use static values.

Reply...

```
+         <h3>Name: Parthiv Jani</h3>
```

6 frontend/src/App.js

Viewed

```
@@ -22,13 +22,19 @@ import Reservation from "./Pages/Details/Reservation";
import Login from "./Pages/Auth/Login";
import Register from "./Pages/Auth/Register";
import RestaurantApproval from "./Pages/RestaurantApprovals/RestaurantApproval";
+ import Index from "./Pages/User-Profile/index";
```

muhammadbilal14111 Pending

Index is default component that is get taken from the folder. you dont need to specify it. Just write "./Pages/User-Profile"

Reply...

```
const routes = [
```

6. ÖMER ZOĞUBİ
   a. Rewards API

Rewads API #79

Open kikolazeski wants to merge 1 commit into `master` from `f-backend-rewards`

Conversation 4 | Commits 1 | Checks 0 | Files changed 4

kikolazeski commented last week

No description provided.

Rewads API 679b575

omerzogubi reviewed 6 minutes ago View changes

libs/controllers/booking.js

```
+        const { userId, restaurantId, numberOfSeats, dateOf, time, extraServiceId,
+            lastUpdate, reservations } = req.body;
+
+        if((lastUpdate - 0000) === 2 ) return res.status(400).json('Sorry your Reservation can not be u
```

omerzogubi 6 minutes ago

Why do you subtract 0000 from lastUpdate? It doesn't make any sense.

Reply...

Resolve conversation

omerzogubi reviewed 5 minutes ago View changes

libs/controllers/booking.js

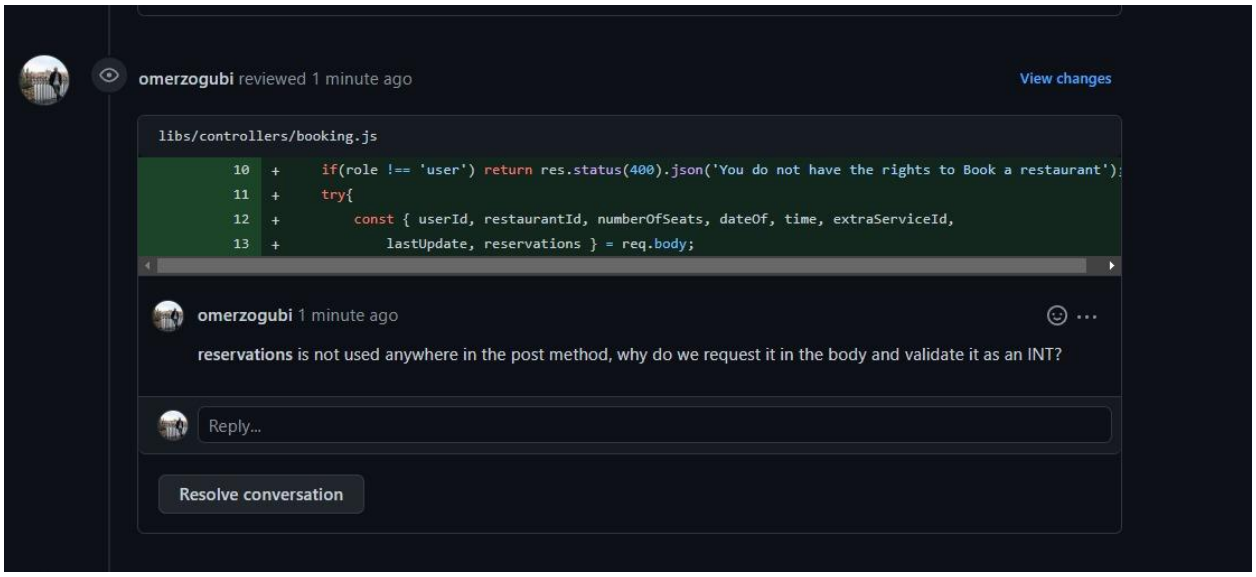
```
+        if(!validateInt(numberOfSeats) || !validateInt(reservations)) {
+            return res.status(400).json('Invalid numberOfSeats or reservations');
+        }
+        if(!validateInt(dateOf) || !validateInt(time) || !validateInt(lastUpdate)) {
```

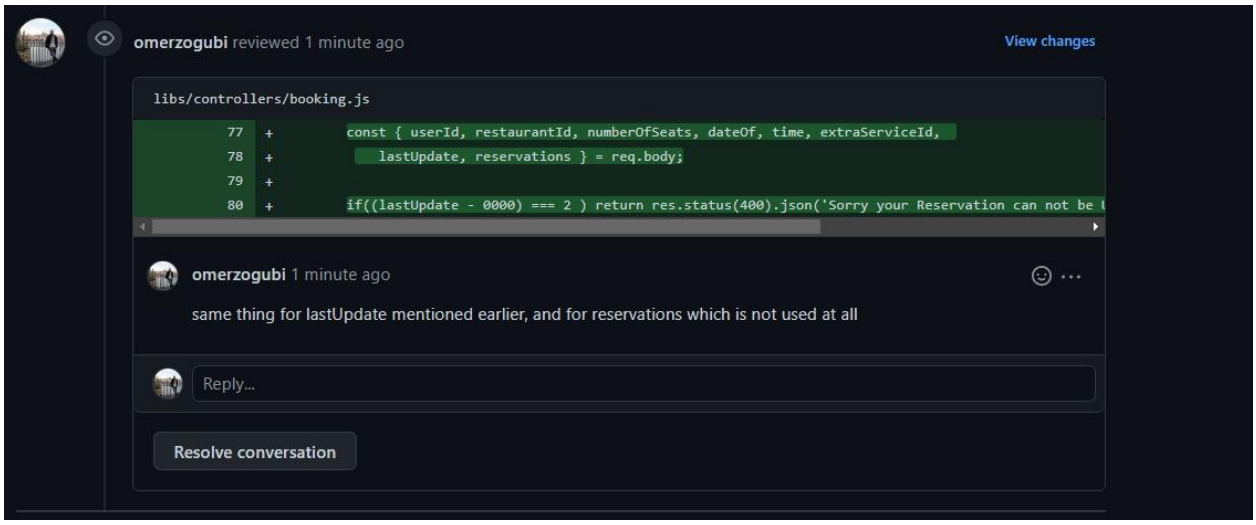
omerzogubi 5 minutes ago

(dateOf) must be validated using date type not integer

Reply...

Resolve conversation





5. Self-check on best practices for security

Assets:

1. Customer data (e.g. names, contact information)
2. Restaurant data (e.g. menus, location, availability)
3. Website functionality (e.g. booking forms)

Threats:

1. Data breaches or hacks targeting customer information
2. Malicious attacks on the website or server infrastructure
3. Payment fraud or theft

Protection measures:

1. Using strong encryption to protect customer data in transit and at rest, and implementing robust access controls to limit access to sensitive information
2. Regularly testing and updating website security measures, such as firewalls and intrusion detection systems, to protect against attacks
3. Implementing secure payment processing systems, such as tokenization or two-factor authentication, to reduce the risk of payment fraud.

Confirmations:

1. Yes, it is important to encrypt passwords in the database to protect against unauthorized access in the event of a data breach.
2. For the search bar input validation, the code might include checks for the length of the input string, as well as any special characters or other characters that may be invalid or could indicate an attempted attack (e.g. SQL injection).

As Security is the most important part of each Website that is being hosted on the Web, we as a Project should stay realistic and implement as much Security mechanism as we can after we finish all of our First Priority Functional Requirements.

6. Self-check: Adherence to original Non-functional specs – performed by team leads

ID	Functional Requirement	Status
1	The Application is a Vendor for Multiple Restaurants	DONE
2	User can Search for Restaurants by name, cuisine, food, and rating	DONE
2.1	Search results can be filtered additionally and sorted	DONE
3	User can reserve a table and specify the number of seats in a given Restaurant	DONE
3.1	User reserves one or Multiple Tables at the Restaurant	DONE
3.2	User specifies the number of seats	DONE
4	Restaurants can be added, removed and updated by Restaurant owners	DONE

4.1	Restaurants can be Added by the Restaurant Owners	DONE
4.2	Restaurants can be Removed by the Owners of given Restaurant	DONE
4.3	Restaurants can be Updated by the owners of said Restaurant	DONE
5	There is a Grace Period to Cancel or update the booking of the table	DONE
5.1	Every User can Cancel the reservation up to 2h before said reservation	DONE
6	The number of booked seats are specified per table	DONE
7	Users can view Menus of the Restaurants	DONE
8	Users search for restaurants, make/change/cancel reservations and also post reviews.	DONE
8.1	User can cancel Reservation	DONE
8.2	User can change Reservation	DONE
8.3	User can Post Reviews	DONE
9	User can View the Restaurant Reviews	DONE
10	Every booking is associated with an account	DONE
11	One Account can not be associated with multiple users	DONE
12	User can chat with the Restaurant Host	DONE
13	User accumulates Reward Points for every reservations with the application	DONE