**A) What I learned from QuickPay**

1.Spring boot :

* Intro: What is spring boot , spring container , spring vs spring boot , beans , annotations .
* properties file
* MVC
* Dependency injection
* How to link data base with spring boot
* How open Postgres in cmd
* Repository pattern , JPA repository interface and JPQL code
* Package organization
* Changing a class into database table
* Join two tables and using foreign keys and relations @one to many
* A screen shot of a computer

  Description automatically generatedWhat is serialization and how to prevent stack overflow exception because of infinite serialization using @JsonIgnore and @JsonBackReference

The infinite serialization happens because of bidirectional relationship mapped with each other

* Storing data URL in the DB
* What are Sessions

2. REST API , JSON and postman

3.Front end :

CSS : position , Z-index , padding , margin , flex , opacity , display , border , box-shadow , focus , flex-direction , white-space : No wrap , media query

HTML : tables , header , footer , div , viewing the html and DOM as tree

JS : Fetch API , how to append variables to the URL , validations , getting elements in the DOM ,

Dynamic html , DATA URL and BLOB URL , local storage , how to add a pfp

**B) bugs I faced**

Backend : the familiarity of most backend bugs like

1.mistakes in properties files

2.stack over flow exception

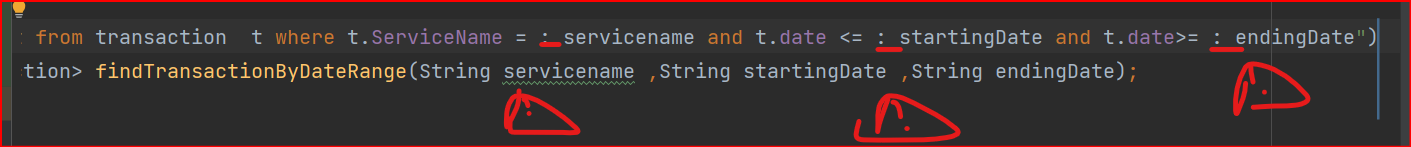
3. wrong JPQL code

4.wrong packaging (the main class should be outside of all the packages )

5.null values in the table attributes (not adding setters and getters so they are not saved when I use Save() ) also JSON in the java script should be written the same as it is in the class attributes otherwise this attribute will be saved as null

6. storing Data URL without adjusting the column size because var char is not enough @column( length = ###### )

7. don’t add spaces in JPQL after ‘: ’



8.any updates u make in the code re save it to update otherwise it is gonna be a local update

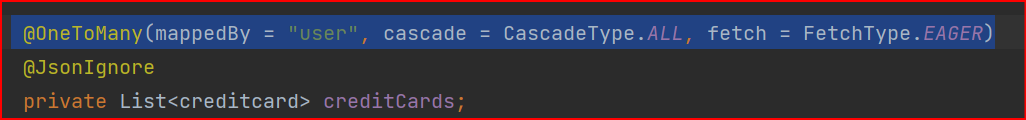
A screen shot of a computer code

Description automatically generated

9.LazyInitialization exception:

I changed fetch type from Lazy to Eager

Important : avoid fetching too many collections eagerly to prevent performance issues



Front end mistakes :

1.Fetch mistakes :

a . don’t call response variable more than one time

b. Don’t take values from the field when the page is loaded as it is still empty so no values u should take it just exactly before sending because you cant send an empty payload u will get error 400.. this image is wrong Tho

A screen shot of a computer

Description automatically generated

**C ) How the website works in summary :**

1. Search : this Api displays all the services stored in the backend a Lil trick here : is that I made Boolean called fake page which add the service name in the local storage

So when I transfer to the page of the selected service it appears as if I’m in the actual page but in fact I only have one page for mobile internet >> We and it assembles 4 different sub services inside the selection bar

Also I remove the selection bar based on this

2. Summary button : which brings all the transactions made it is like a notification

3. Favorite button: displays a table that gathers the fav services to give easier access and user experience

**\*First 3 are implemented in the dashboard.js\***

4. QuickPay page which load all the credit cards in the DB the credit card can be deleted also there the type of the card is identified based on the card number regex and security for it when displayed if no cards in your account then u need to add one so a form will be shown so u can add your information and save the CC to the DB

**\*Quickpay.js and creditcardform.js\***

5.myHistory gets all the payment transactions in the DB based on the date and service name

**\*myHistory.js\***

6.Help&support page sends complaints or questions to the admin

7.settings contains my Profile and security there u can update your information and add a profile picture

8.payment:

* New payment button :

This will let u choose between different services u pick until u reach to the payment step where u must select a credit card to pay u can have more than one if not u cant

Pay then u will be transferred to quick pay to add a credit card . if u have a cc you pick it your pick is sent to the server side so the payment calculations occurs because there might be more than one , then you write the amount and the payment calculations are made in the server side and finally a receipt with the service name and service type will be displayed and the order number

(using local storage took the info through the picking steps services.js)

* Pay from favorite list here u should use the fake page Boolean stored in the local storage that will transfer you to the service page directly and covers the drop down menu and button all is left is to add your number and then go to payment at receipt.js I deleted the fake page from the local storage to prevent interruptions and logical errors

* Search menu when u type
* All the scenarios of fake pages are handled :

1. If u go from search to the desired page if u continued to the receipt step it will be deleted there once the receipt.js is loaded
2. If u change your mind without continuing to the receipt step then an alternative deletion will happen at the dashboard page (dashboard.js) to delete the fake page Boolean
3. If u go from favorite page and changed your mind and choose the path of new button then also the fake page is deleted

In summary it is going to be deleted when u change your mind once u go the dashboard page … u cant go any of the previous paths unless u start with the dashboard page so the reset happens there

**D) Remaining implementations**

1. media query so the website be responsive

2.lanuage switch (Arabic-English)

3.Animations in the home page

5.Fake bank controller (not imp)

6.Fake Vodafone or company to accept the payment (not imp)