Programming Applications and Frameworks (IT3030)

3rd Year, 1st Semester

GadgetBadget

Report

Group Number: Y3S1.02.1(IT)_20

Student Registration No.	Student Name
IT19010472	K.Mithusha
IT19012834	Naveen S
IT19111834	Cooray P.L.R.K
IT19118246	Wijesekera S.M
IT19122588	Gunarathne N. U

GitHub - https://github.com/RajinduSE/GadgetBadget.git

Content

1. Members' details

	Student Registration Number	Student Name	Workload distribution
1	IT19010472	K.Mithusha	<u>Product Service</u> • Insert new product details. • Update the product details. • Delete the product details. • View the product details.
2	IT19012834	Naveen S	<u>Payment Service</u> • This service is focused on managing the payment details for the bought products. Buyer (who is a registered user) gets the privileges to make payments for their purchased products, edit, Delete, view their Payment details (CRUD).
3	IT19111834	Cooray P.L.R.K	<u>Auth Service</u> • This service can be used to login to the system, to verify the user is logged before doing any operation, to check whether the user has necessary permission to perform the operation and logout from the system. <u>Research Service</u> • This service can be used to manage research papers of users (CRUD), approve a research, download a research paper like functionalities.
4	IT19118246	Wijesekera S.M	<u>Order service</u> Order service is related to management of orders placed by the buyers. Functions related to the order service are, • Adding order details • Updating order details • Deleting order details • Retrieving order details

5	IT19122588	Gunarathne N. U	<u>User Service</u> This service is related to registering and managing a user/researcher. The functions done by the service are as follows, • Adding User Details • Updating User Details • Viewing User Details • Deleting User Details

2. SE methodology/methods

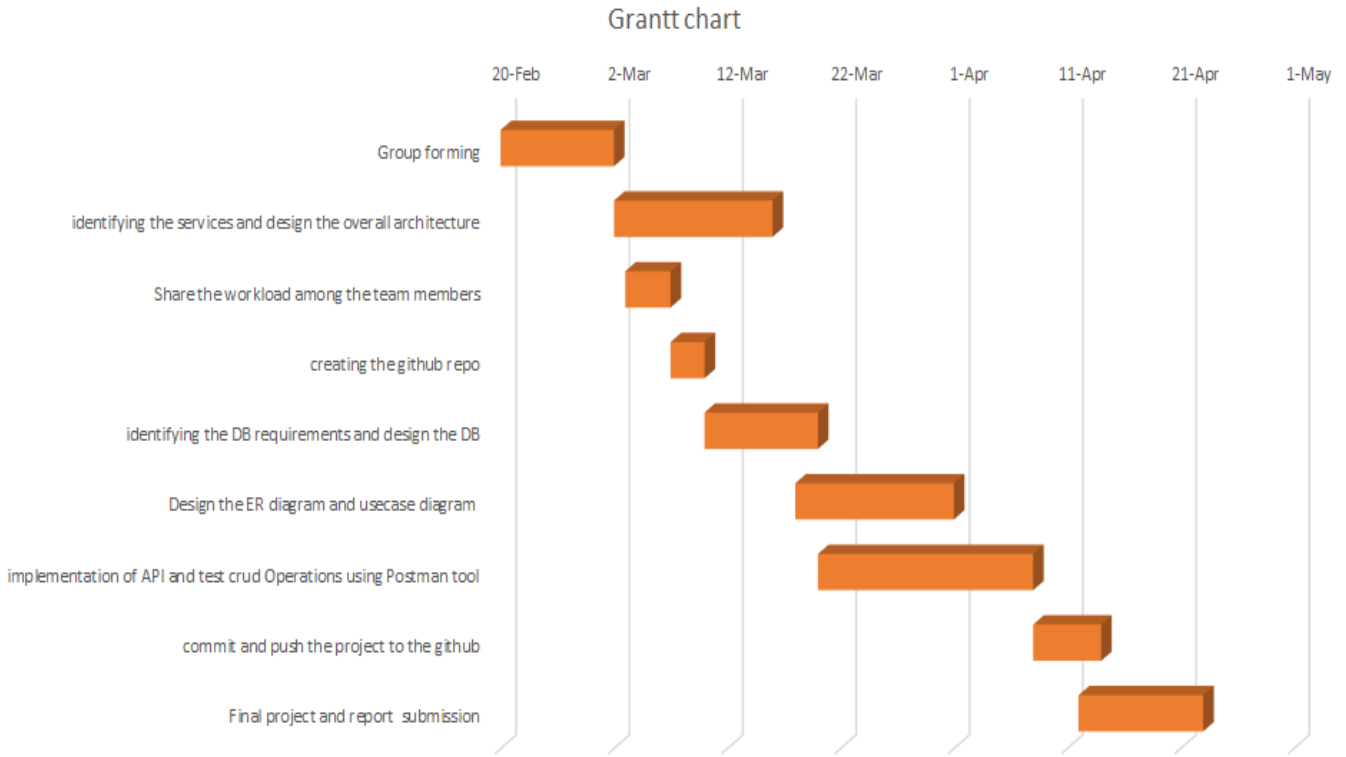
- **Requirement Analysis -** At first, we analyzed the issues related to the functional, Non-functional and technical requirements. Afterwards we maintained our project up to date as we have shown below in the Gantt chart. We identified the web services for this project and chose one service each, among the five members and implemented each as micro services. Since we have used GitHub in previous projects, we decided to create a repository and connect eclipse and collect our projects using GitHub. We identified RESTful Web service: Java - JAX-RS (Jersy) on Tomcat, DB: MySQL and testing tool postman (Client Side) as technical requirements to develop this project.

- **Creating a design -** This System is designed for researchers to reach potential buyers and sell their projects through a computerized platform. After a proper requirement analysis, we decided to implement the web services as micro services.

- **Coding, Testing & Installation -** We used RESTful Web service: Java - JAX-RS (Jersy) on Tomcat, DB: MySQL for database and tested it using postman (Client-Side testing) tool for this implemented system. We wrote the code for the APIs, CRUD functions and then created the database and connected it to the services to develop the system. After finishing the coding part, we formatted it in eclipse.

- **Maintenance -** This is the final step. In this part, if we want to add features or modify the code, we can give updates according to the customer/user needs. Incremental model is being used for this project as a SE Methodology.

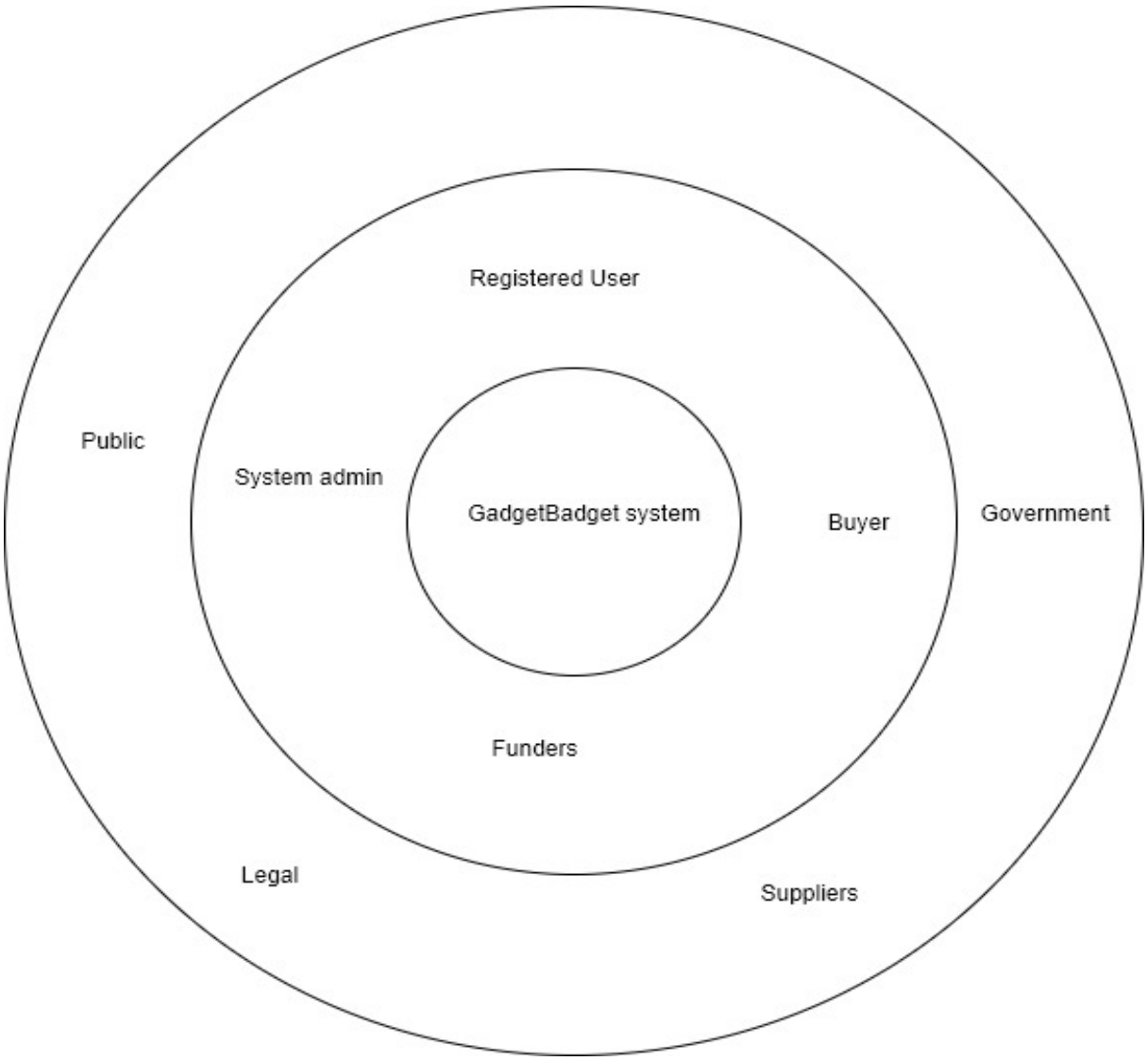
3. Time schedule (Gantt chart)



	Final project and report submission	commit and push the project to the github	implementation of API and test crud Operations using Postman tool	Design the ER diagram and usecase diagram	identifying the DB requirements and design the DB	creating the github repo	Share the workload among the team members	identifying the services and design the overall architecture	Group forming
Start date	12-Apr	8-Apr	20-Mar	18-Mar	10-Mar	7-Mar	3-Mar	2-Mar	20-Feb
Duration	11	6	19	14	10	3	4	14	10

4. Requirements

4.1 Stakeholder analysis (onion diagram)



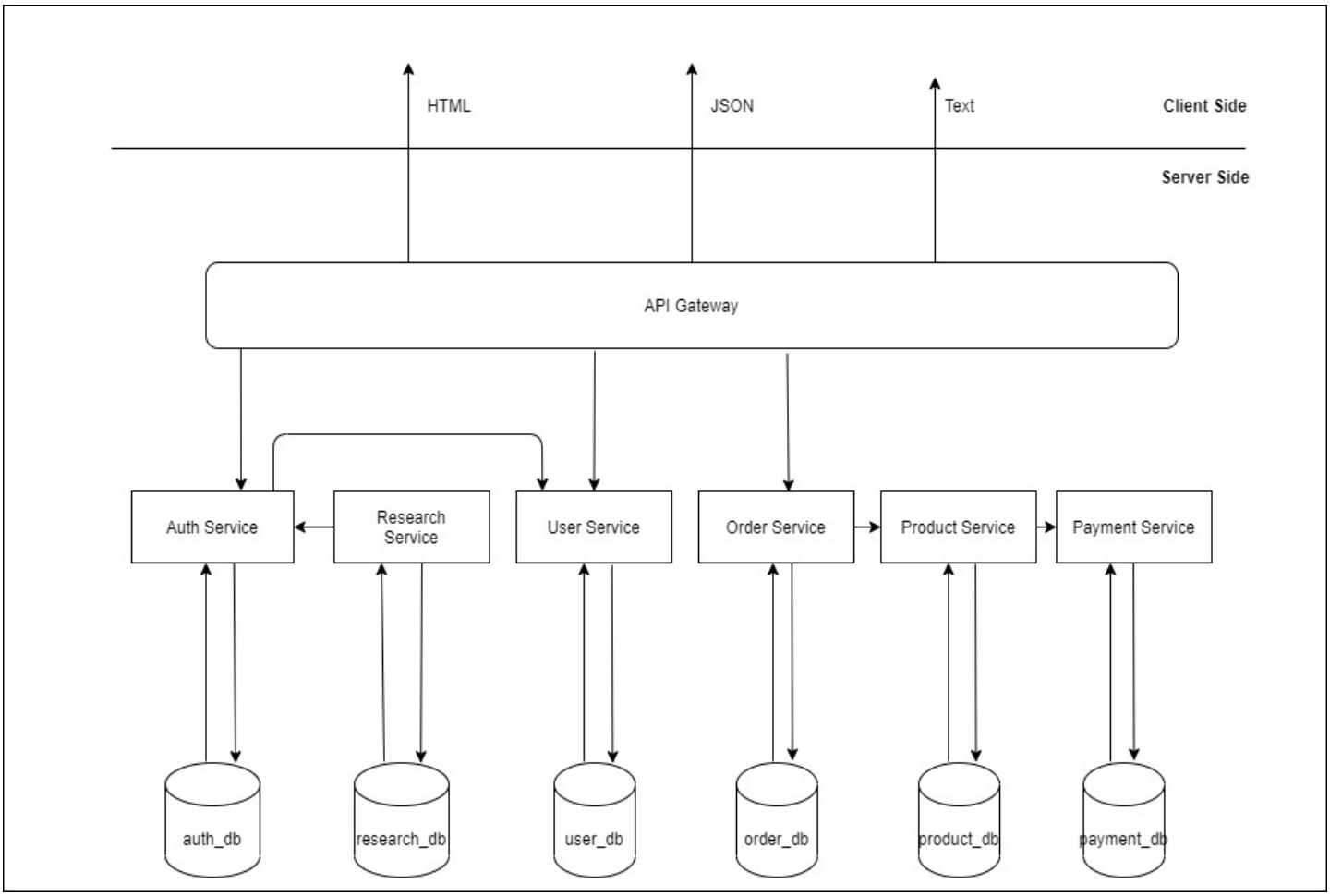
4.2 Requirements analysis

Function	Functional Requirement	Non-functional Requirement
Payment Service	Add new Payment details Delete Payment details View Payment details Update Payment details	-Automated testing tool -VCSGitHub(Recoverability, integrity) -Privacy -Scalability -Response in time
Product Service	Insert new products Edit product details Delete product details Retrieve product details	Privacy, Usability, Recoverability, Response in time
Auth Service	Login to the system, Verify a user, Logout from the system	Scalability, Reliability, Integrity, High Security and Privacy
Research Service	Add research details Delete research details Update research details Approve research Download research View approved researches View pending researches	Scalability, High Security and Privacy, Well Supported, Developer friendly High availability
Order Service	Add order details Update order details Delete order details Retrieve all order details Retrieve order details by ID	Reliability Availability Data integrity Usability Maintainability Recoverability Scalability
User Service	Add user details Update user details Delete user details Retrieve user details View research details	Scalability Availability Security & Privacy Usability

4.3 Requirements modelling (Use case diagram)

5. System's Overall design

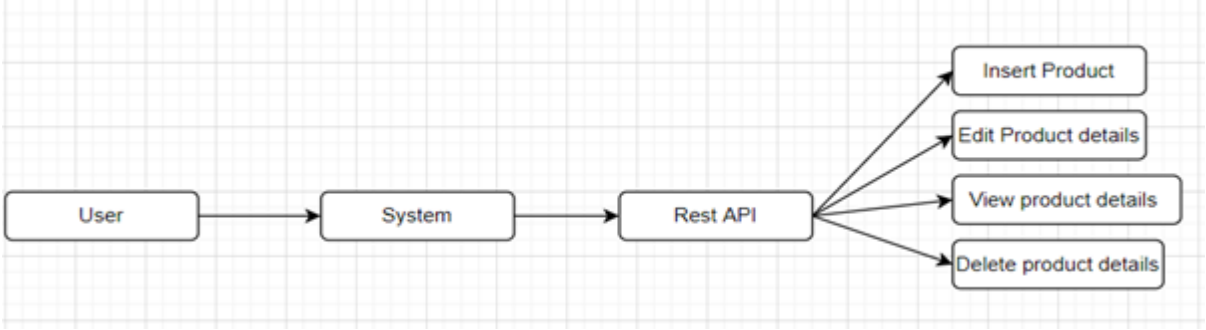
5.1 Overall architecture



6. Individual sections

1.Product Service (K.Mithusha IT19010472)

API for the product Service



GET – Read Product details

URL – http://localhost:8080/ProductAPI/ProductService/Products/
Resource – Product Service API
Request – GET Product Service/Products
Response –

```
{
    "proID" :"1",
     "proCode" :"p01",
     "proName" :"ttt",
     "proPrice" :"200",
     "proQty" : "30",
      "proDesc" : "For sell"
}
```

POST – Add Product

URL – http://localhost:8080/ProductAPI/ProductService/Products/
Resource – Product Service API
Request – POST Product Service/Products
Media – Form data – URL encoded.
Data – proCode
proName
proPrice
proQty
proDesc
Response –String Status Message
"Added successfully" or "Error while Adding."

PUT – Edit Product details

URL – http://localhost:8080/ProductAPI/ProductService/Products
Resource – Product Service API
Request – PUT Product Service/Products
Media – Application JSON

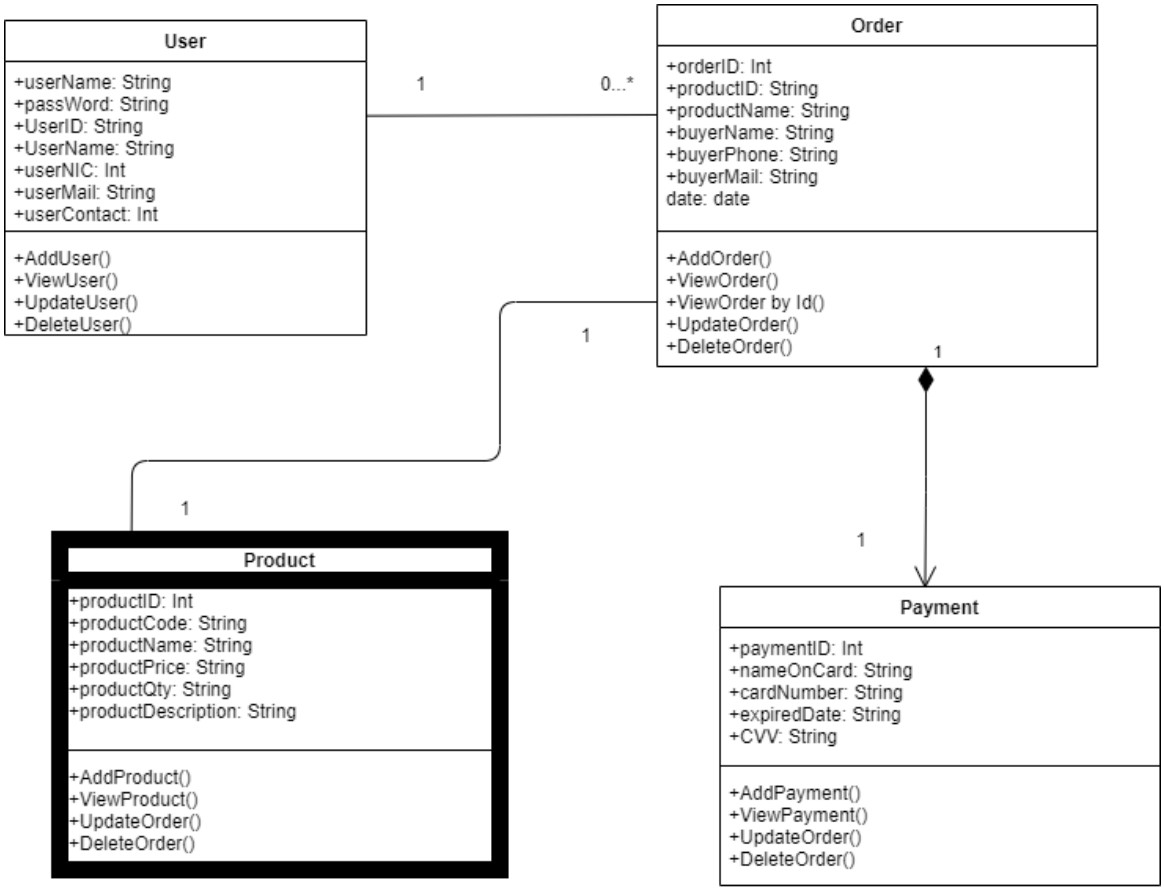
Data – proCode,
proName
proPrice
proQty
proDesc
Response – String Status Message
"Edited successfully" or "Error while Editing the product".

DELETE – Delete Product details

URL – http://localhost:8080/ProductAPI/ProductService/Products
Resource – Product Service API
Request – DELETE Product Service/Products
Media – Application XML
Data – proID
Response – String Status Message
"Deleted successfully" or "Error while deleting the product".

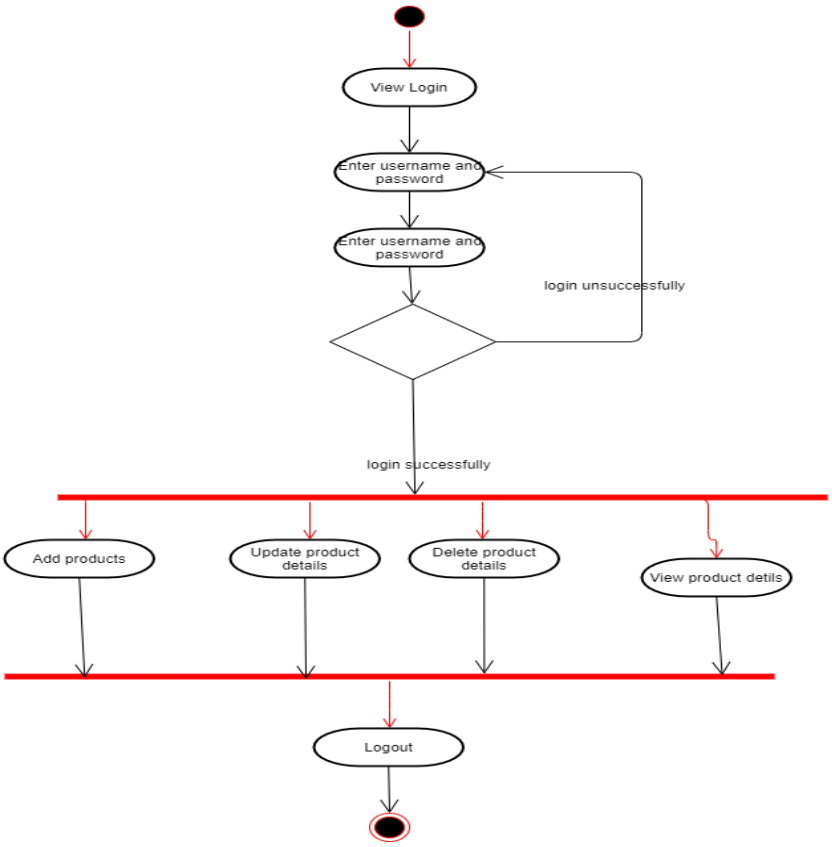
Internal Logic

a. Class Diagram

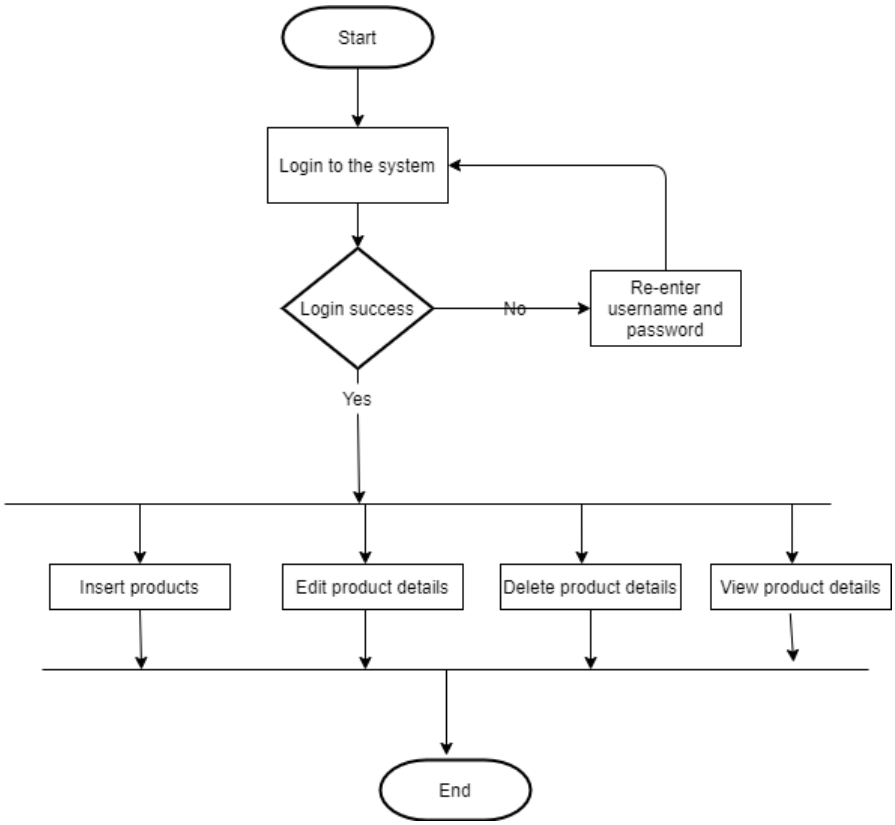


This is my class diagram

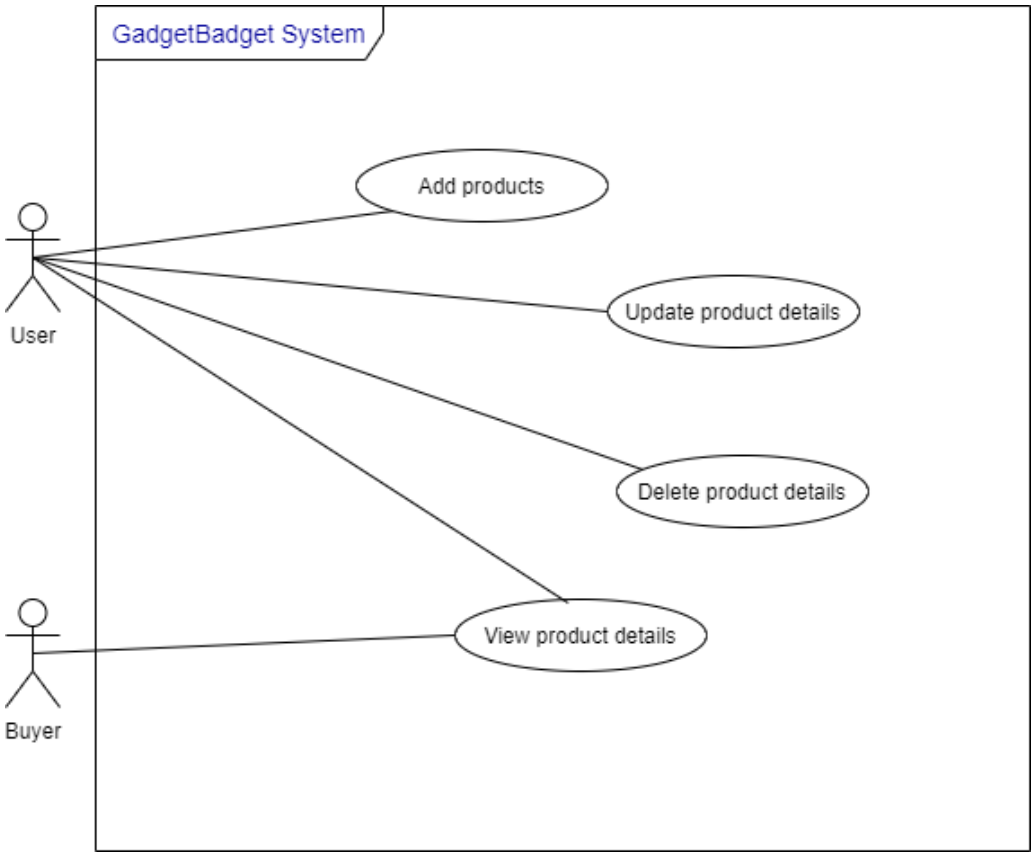
b. Activity Diagram



C. Flow Chart



d. Use Case Diagram



Testing Methodology and results

TestID	Description	Inputs	Excepted Outputs	Actual Outputs	Result(Pass/Fail)
1	Add new products	proCode, proName, proPrice, proQty, proDesc	Display message as "Added Successfully"	Display message as "Added Successfully"	Pass
2	Edit products details	proCode, proName, proPrice, proQty, proDesc	Display message as "Edited Successfully"	Display message as "Edited Successfully"	Pass
3	Delete products details	proID	Display message as "Deleted Successfully"	Display message as "Deleted Successfully"	Pass
4	Read product details		Display all product details	Display all product details	Pass

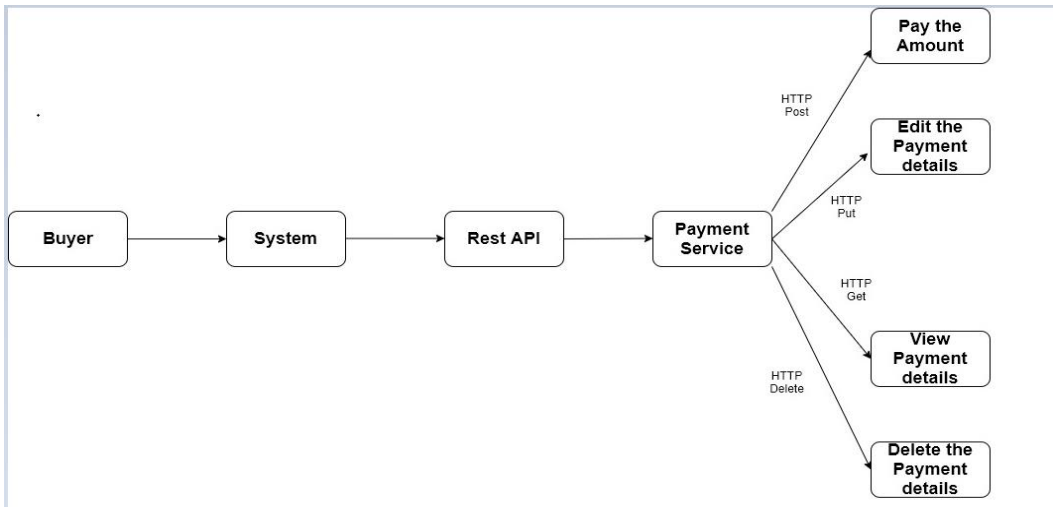
Service Development and Testing

- IDE: Eclipse
- Dependency management tool - Maven
- Server – Tomcat 9.0
- Database – PHP My Admin
- RESTful Web Service – Java - JAX-RS
- Back End: Java
    - Maven
- Testing tool: Postman

Naveen .S

- **API of the service**

Payment Service (Naveen .S IT19012834)



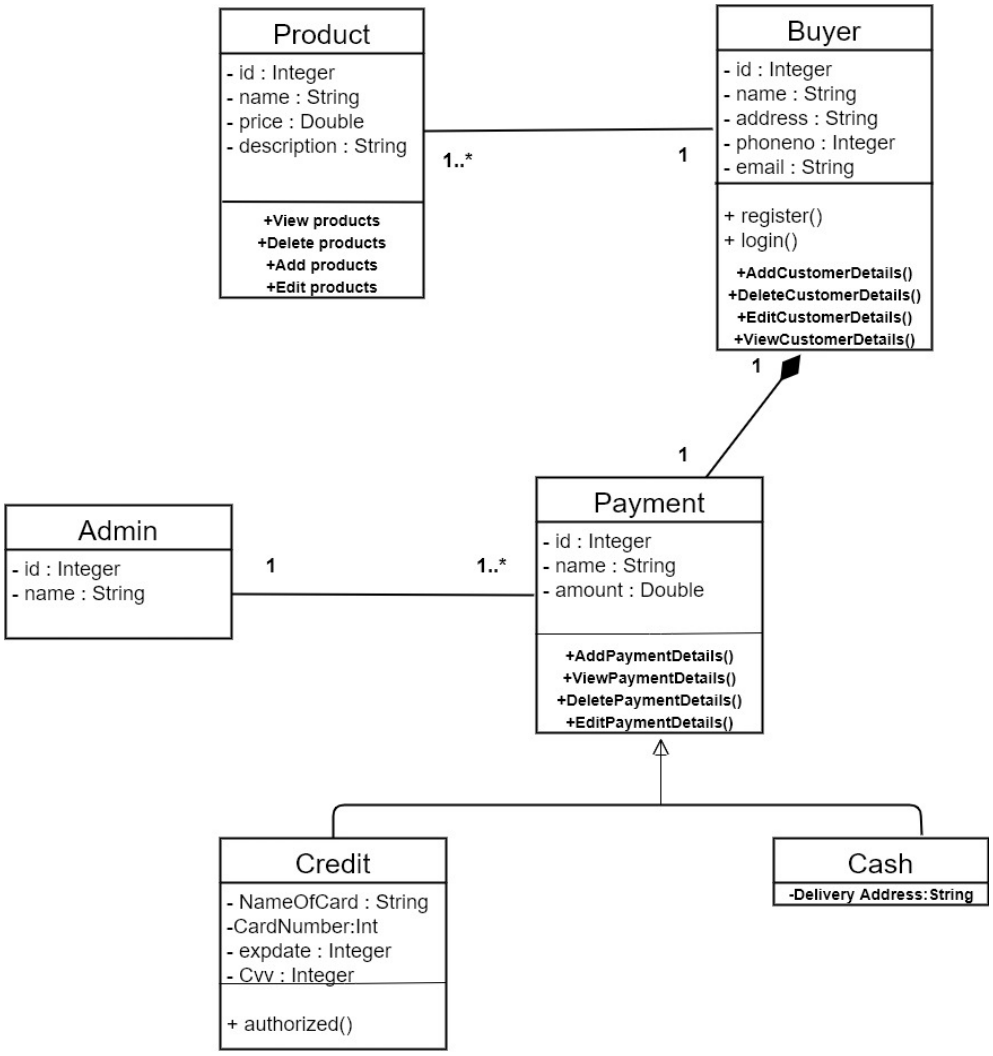
- **Design rationale(Discussion)**

Payment system of a GadgetBadget system is an important service. It helps to maintain thousands of records in payment in a reliable ,Quick accessing and methodical manner. Reduce the complexity of entering the data and user-friendly design and efficiency are the main targets of this designing process. RESTful Web service: Java – JAX-RS (Jersey) on Tomcat, DB: phpMyAdmin(xammp) were the development Environment used to implement this project. GET/PUT/POST/DELETE are the main functions of the system. Maven based project object model was used when implementing this project. It really helped to go into the source code ,adding dependencies . it can add all the dependencies required for the project automatically by reading pom file. We used ”Postman’’ tool to test the functionalities.

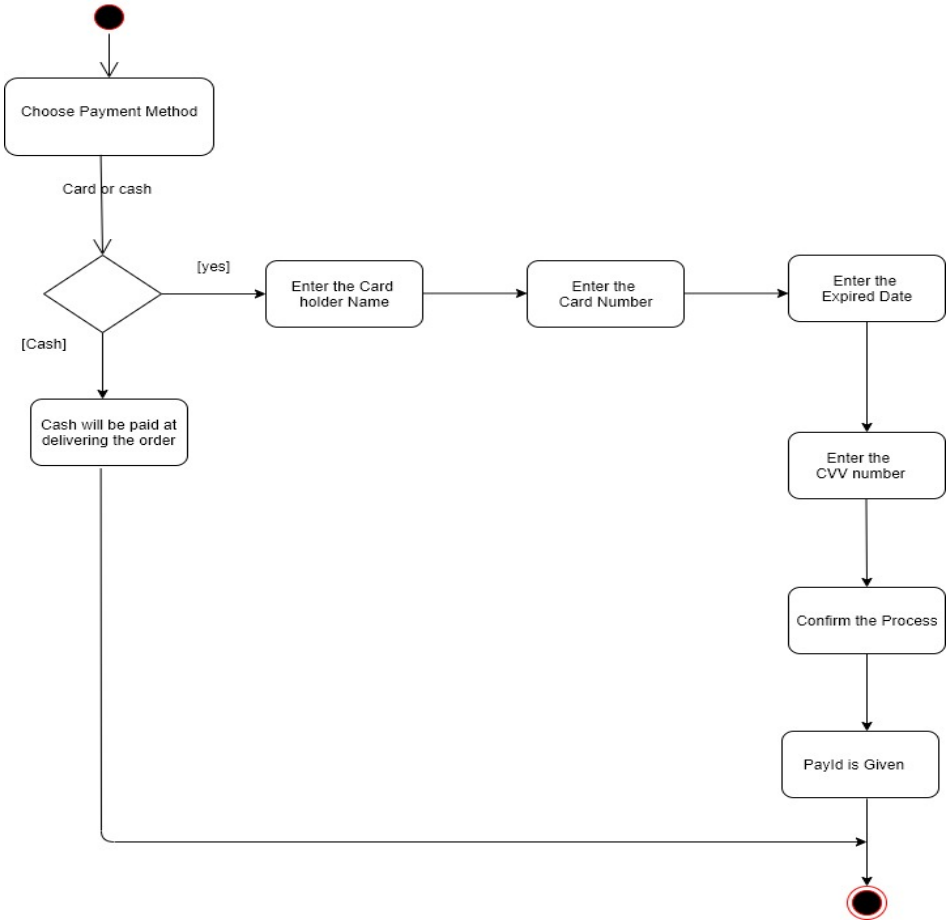
URL- pattern directs to the database of this project and provides the connection. If the connection is successful, it shows the success message and fields in the screen else if it is unsuccessful an error message will be appearing in the screen. Screen shot of the successful connection and the database shown here. and with the help of this API user can make payments for particular product and buyer.

- **Internal logic (Class diagram, activity diagram, flowchart)**

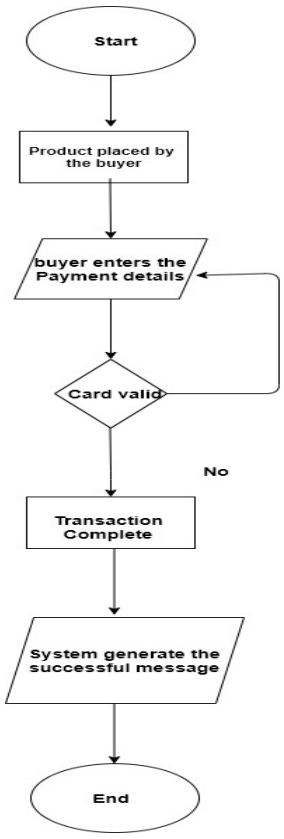
- **Class diagram**



- **activity diagram**



- **flowchart**



Service development and testing.

- Tools used, including justifications for their selection

1) Technical Environment tools

IDE: Eclipse

Database: MySQL (PhpMyAdmin)

Server: Tomcat

Maven

JAX-RS

2) dependency management tools

Maven project management

3)Code quality checking tools

Postman

- Testing methodology and results.

Test ID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
01	View All Payment Details.	• URL for the API • Send GET request.	Display Payment details.	Display Payment details.	Pass
02	Delete a Payment Details	PayId of the appointment to be deleted.	Display message as "Deleted successfully"	Display message as "Deleted successfully"	pass
03	Add a Payment.	NameOnCard, CardNumber, ExpiredDate, cvv	Display message as "Inserted successfully"	Display message as "Inserted successfully"	pass
04	Update Payment Details.	Attributes to be updated along with the PayId.	Display message as "Updated successfully"	Display message as "Updated successfully"	pass

- **Assumptions**

- A buyer can buy only one product at once

Cooray P.L.R.K (IT19111834)

There are two APIs

1. Researches API
2. Auth API

API for the Service

Auth API

Login

- Resource : Auth
- Request : GET AuthService/Auth/login
  - ✓ Media : Application JSON
  - ✓ Data :

```
{
    "username":"rajindu",
    "password":"1234"
}
```

- Response : String status message

This service is used by the user to login to the system. Auth service use the User service(Created by other member) to get username and passwords of registered users.

The internal logic of login is it inputs username, password, and user list(Calling a user service). First it converts this user list to Map<String, String> and using for each it validates username and password. If validation fails, then it returns "Invalid Login" as response. Since rest is a stateless architecture, we could not use sessions and cookies. So, for a successful login JWT token will be generated by using username of the user and using an expiration time.

User Verify

- Resource : Auth
- Request : GET AuthService/Auth/verify
- Response : String status message

This call is used by Research Service(Created by me) to verify the user has logged into the system before performing any user authorized operation.

In internally this will check is there a non-expired JWT token for that user.

Admin Verify

- Resource : Auth
- Request : GET AuthService/Auth/adminVerify
- Response : String status message

This call is used by Research Service other than the user verify call to confirm this is an admin profile or not.

Internally this will check whether the JWT is created using the admin username.

Logout

- Resource : Auth
- Request : GET AuthService/Auth/logout
- Response : String status message

This call is used to logout from the system. Internally it deletes the JWT token.

Researches API

Add Research

- Resource : Research
- Request : POST ResearchService/Researches
  - ✓ Media : Application JSON
  - ✓ Data:

```
{
    "title":"Object Detection",
    "author":"Rajindu",
    "catergory":"computer science",
    "year":2021,
    "path":"C:\\Users\\Rajindu\\Documents\\SLIIT\\Y3S1\\PAF\\pdfs\\sample.pdf"
}
```

- Response : String status message

From this service a user can add a research to the system. They can also upload a pdf of the research to the system. To use this service the user must log into the system(Auth API - verify).

Pending Researches

After adding a research, that research will be in pending state and that should approve by the admin. Admin can see all pending researches using the following call. Only admin is authorized to use this call.(Checked using Auth API – Admin verify)

- Resource : Research
- Request : GET ResearchService/Researches/pending
- Response : HTML table with Research title, Author, category, year and action buttons for download and approve the research.

When adding a research there is a field called status in reseaches_db and that field is set to false. In pending researches call, this will show all rows with status equals to false.

Approve Researches

Only admin is authorized to perform this action(Checked Using Auth API – Admin verify).

- Resource : Research
- Request : PUT ResearchService/Researches/approve
    - ✓ Media : Application JSON
    - ✓ Data :

```
{
    "id":10
}
```

- Response : String status message

Internally this will set status column in researches_db to true.

Download Research

For approving process, the admin needs to read the research uploaded by the user. So, this can be used to download the uploaded research.

- Resource : Resource
- Request : GET ResearchService/Researches/download
    - ✓ Media : Application JSON
    - ✓ Data :

```
{
    "id":10,
    "path":"C:\\Users\\Rajindu\\Documents\\SLIIT\\Y3S1\\PAF\\pdfs\\",
    "file":"check5.pdf"
}
```

- Response : String status message

This will save file into specified location in path attribute.

Approved Researches

After research is approved by the admin it will display to user.

- Resource : Research
- Request : GET ResearchService/Researches
- Response : HTML table with research title, author, category, year and actions buttons for update and delete.

Update Research

- Resource : Research
- Request : PUT ResearchService/Researches
  - ✓ Media : Application JSON
  - ✓ Data :

```
{
    "id":10,
    "title":"Object Identification",
    "author":"Cooray",
    "catergory":"AI",
    "year":2020
}
```

- Response : String status message

Delete Research

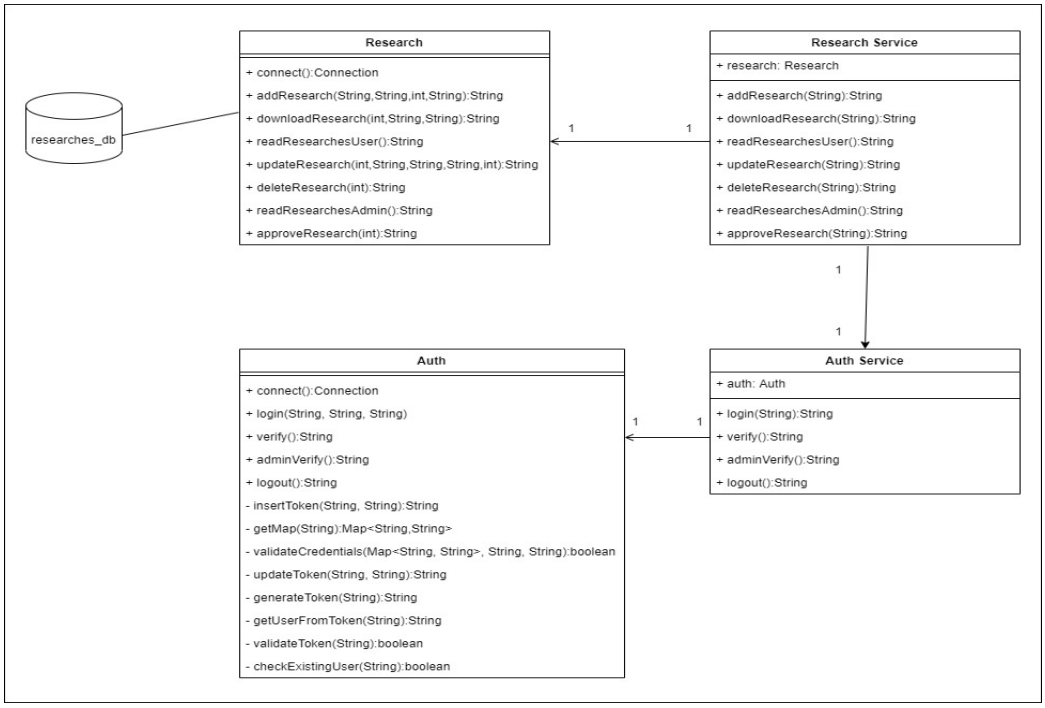
- Resource : Research
- Request : DELETE ResearchService/Researches
  - ✓ Media : Application JSON
  - ✓ Data :

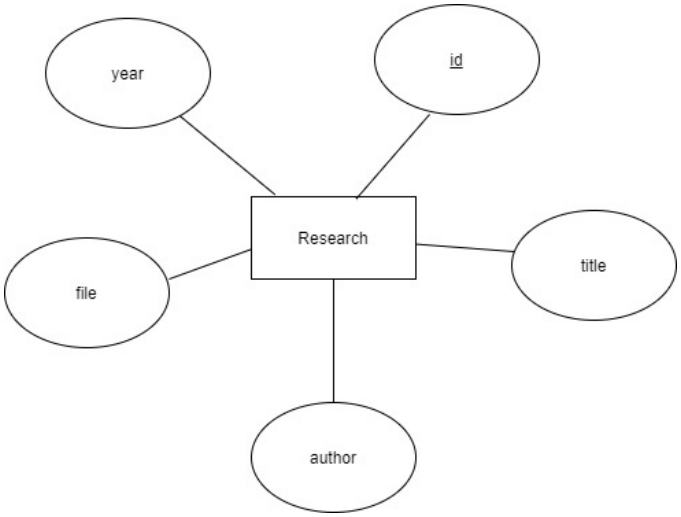
```
{
    "id":10
}
```

- Response : String status message

Internal Logic

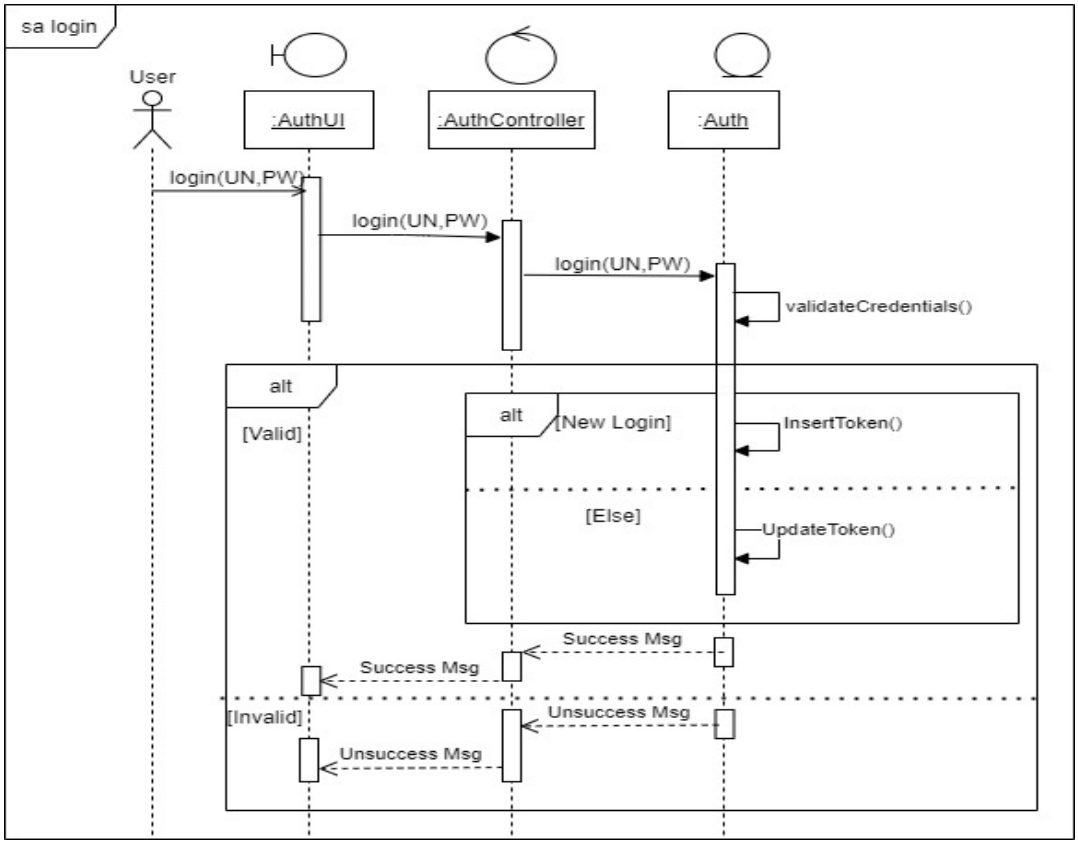
Class Diagram

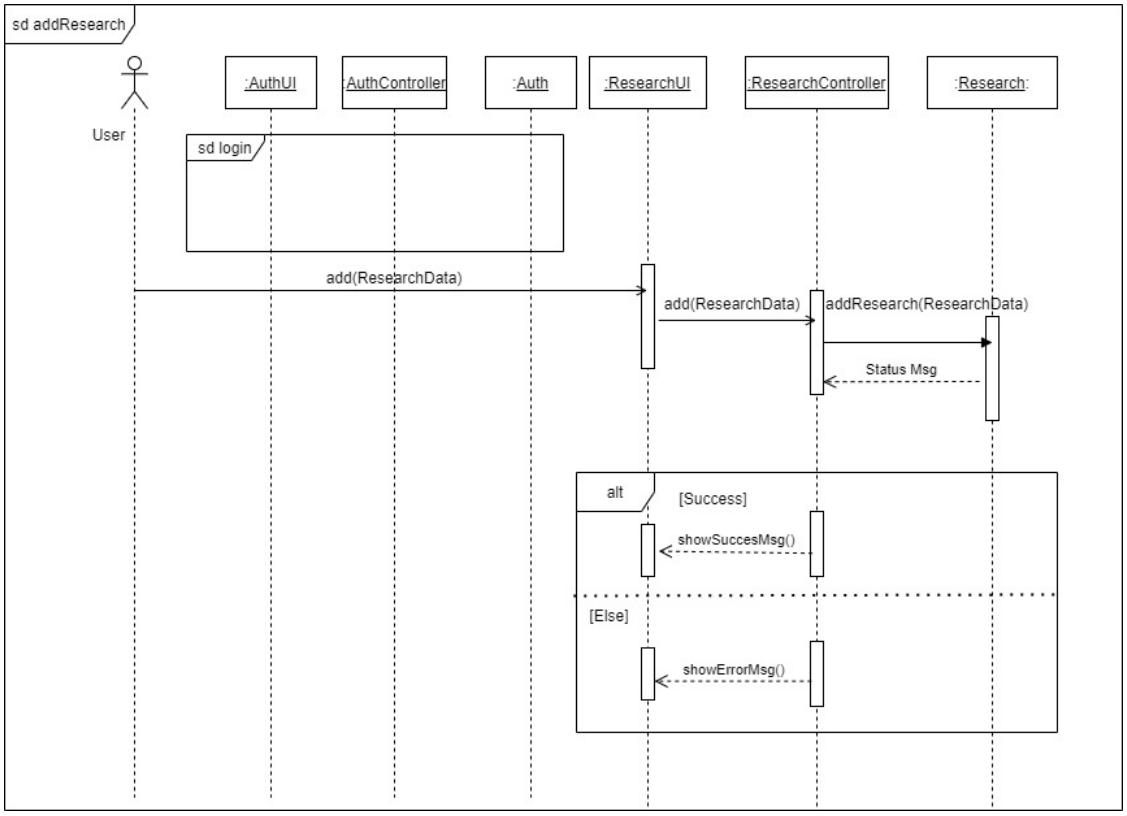


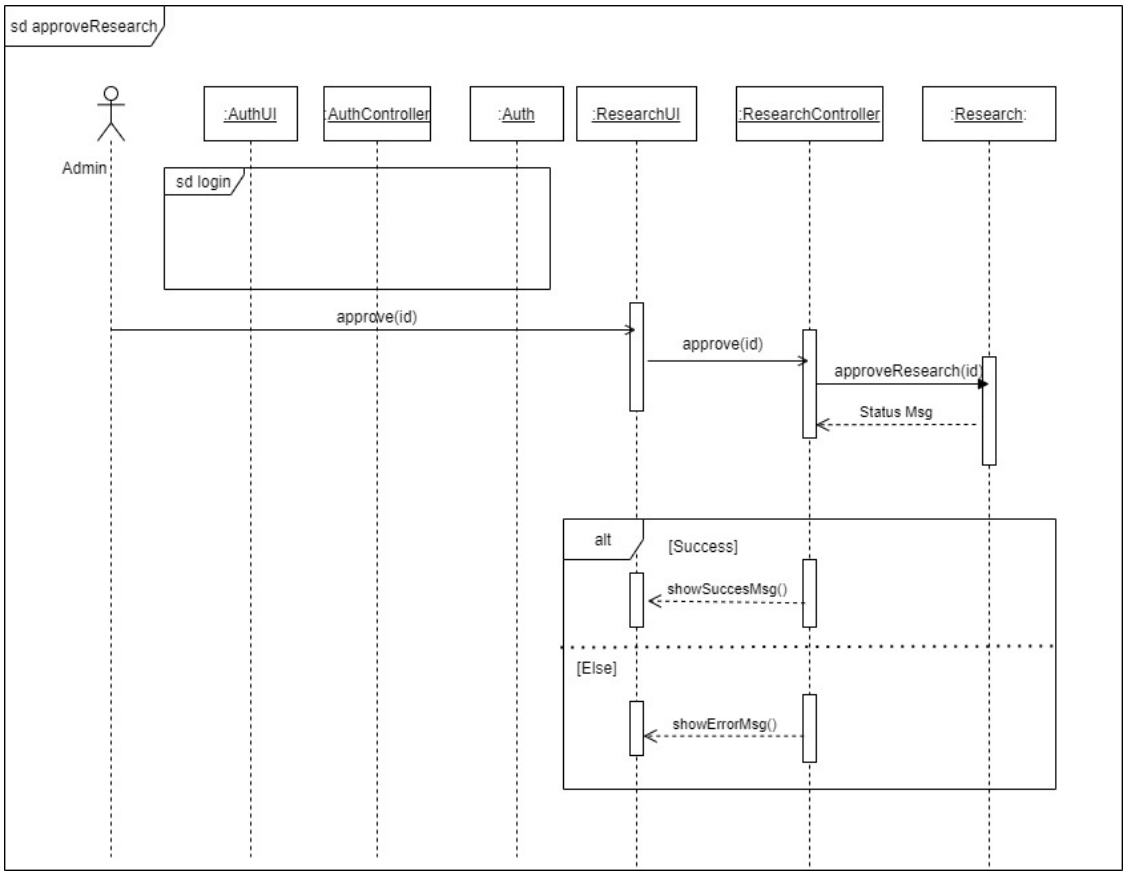


ER for researches dB

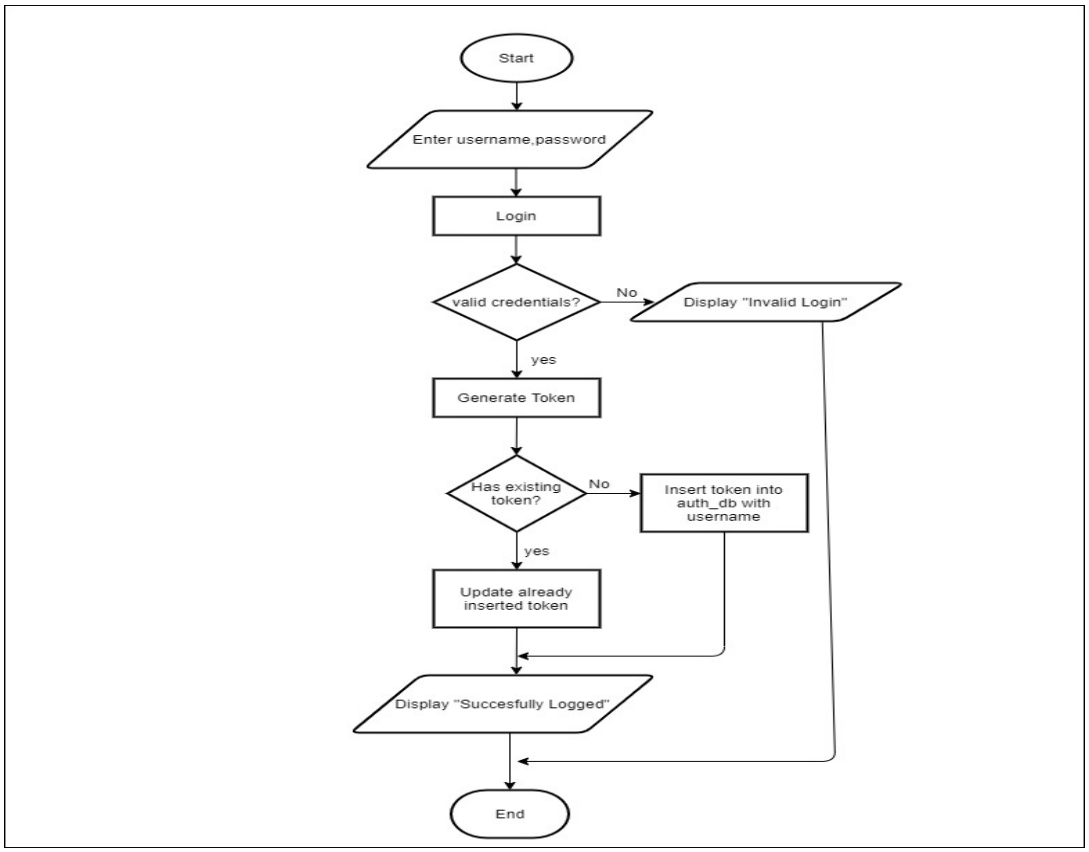
Sequence Diagrams

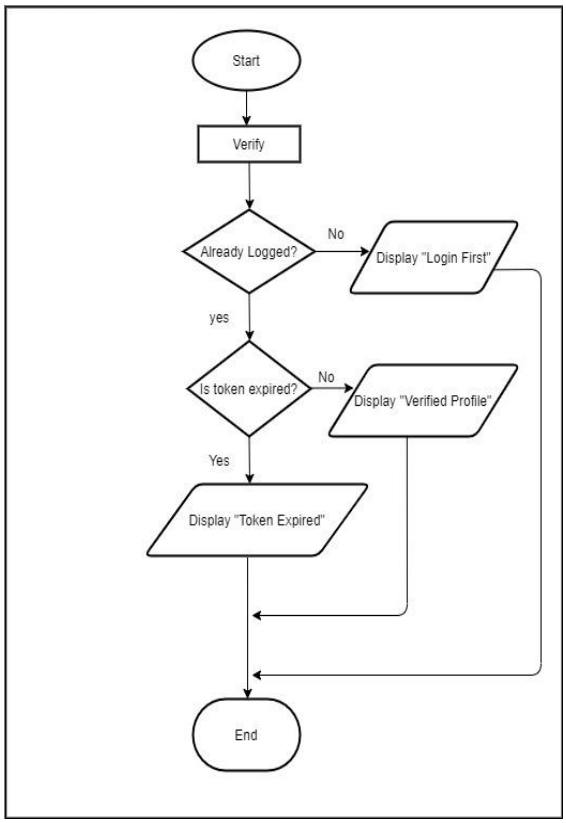


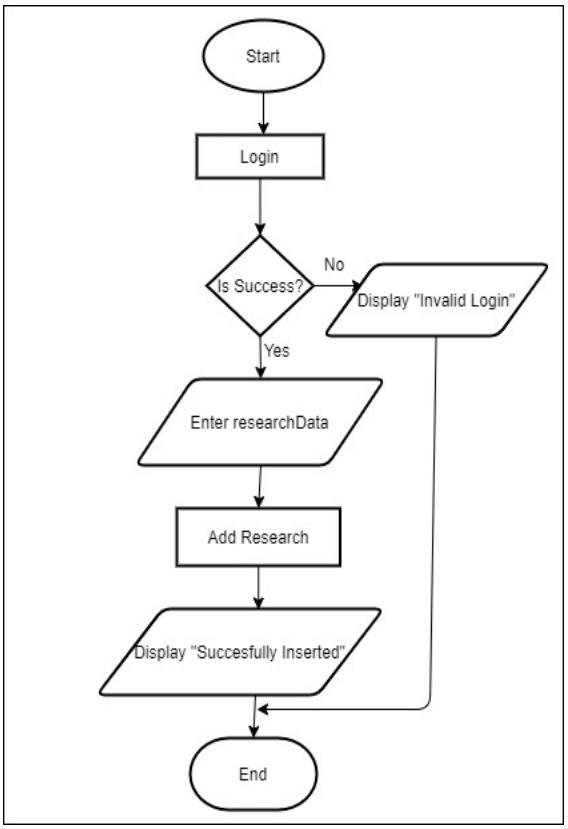

sd addResearch
User
:AuthUI
:AuthController
:Auth
:ResearchUI
:ResearchController
:Research:
sd login
add(ResearchData)
add(ResearchData)
addResearch(ResearchData)
Status Msg
alt
[Success]
showSuccesMsg()
[Else]
showErrorMsg()

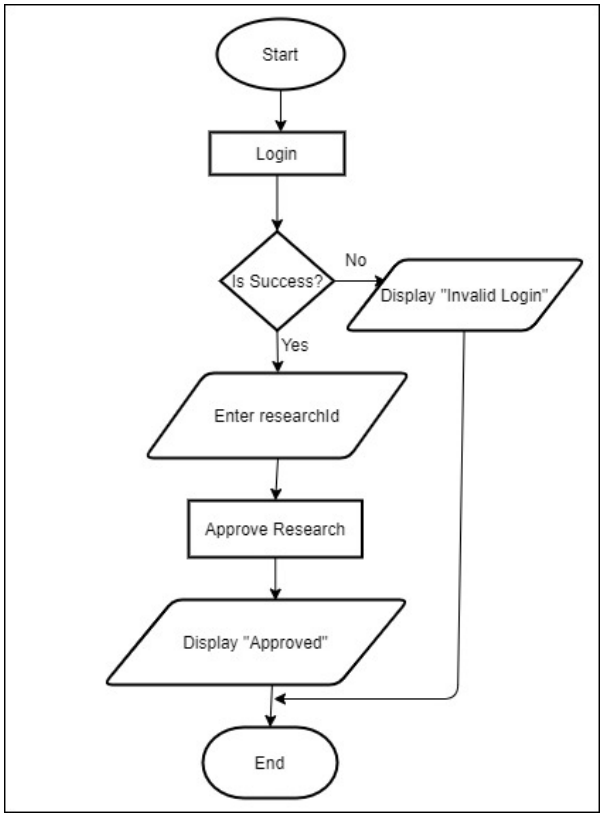

sd approveResearch
Admin
:AuthUI
:AuthController
:Auth
:ResearchUI
:ResearchController
:Research:
sd login
approve(id)
approve(id)
approveResearch(id)
Status Msg
alt
[Success]
showSuccesMsg()
[Else]
showErrorMsg()

Flow Charts


Start
Enter username,password
Login
valid credentials?
No
Display "Invalid Login"
yes
Generate Token
Has existing token?
No
Insert token into auth_db with username
yes
Update already inserted token
Display "Succesfully Logged"
End


Start
Verify
Already Logged?
No
Display "Login First"
yes
Is token expired?
No
Display "Verified Profile"
Yes
Display "Token Expired"
End


Start
Login
Is Success?
No
Display "Invalid Login"
Yes
Enter researchData
Add Research
Display "Succesfully Inserted"
End



Service Development and Testing

- IDE: Eclipse
- Dependency management tool - Maven
- Server – Tomcat 9.0
- Database – MySQL
- RESTful Web Service – Java - JAX-RS
- Back End: Java
  - Maven

Testing tool: Postman(Screenshots are in appendix section)

Testing methodology and results

Test ID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/ Fail)
01	Add research details	Title, author, category, year, path	Display message as “Inserted successfully”	Display message as “Inserted successfully”	Pass
02	Try to insert details without login	Title, author, category, year, path	Display message "Login First”	Display message as "Login First”	Pass

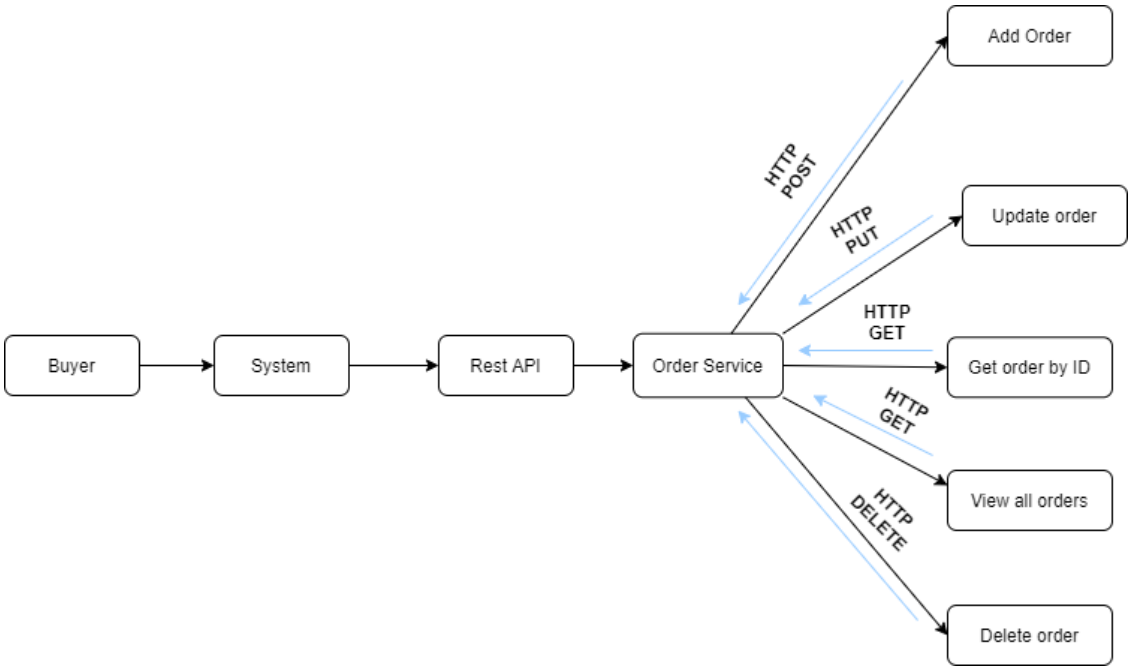
03	Update research details.	Attributes to be updated along with the id.	Display message as “Updated successfully”	Display message as “Updated successfully”	pass
04	Delete a research	id of the record to be deleted.	Display message as “Deleted successfully”	Display message as “Deleted successfully”	pass

Assumptions and Discussions

- Cannot add multiple researches at same time
- File should be in pdf format

IT19118246-Wijesekera S.M

API for the Service



Resource: OrderService API **<u>Request: GET (Get all orders)</u>** Media: *TEXT_HTML* Response: String status message `"Error while reading the orders.”`	Resource: OrderService API **<u>Request: GET (Get orders by Customer ID)</u>** Media: *TEXT_HTML* Data: orderID Response: String status message `"Error while reading the orders "`

Resource: OrderService API
 	
<u>Request: POST (Add order)</u>
 	
Media: *APPLICATION_FORM_URLENCODED*
 	
Data: productID, productName, buyerName, buyerPhone, buyerMail, date
 	
Response: String status message
 	
`"Inserted successfully"`
 	
`"Error while inserting the order."`
	Resource: OrderService API
	
	<u>Request: PUT (Update order)</u>
	
	Media: *APPLICATION JSON*
	
	Data: productID, productName, buyerName, buyerPhone, buyerMail, date
	
	Response: String status message
	
	`"Updated successfully"`
	
	`"Error while updating the order."`

Resource: OrderService API
 	
<u>Request: DELETE (Delete order)</u>
 	
Media: *APPLICATION XML*
 	
Data: orderID
 	
Response: String status message
 	
`“Deleted successfully.”`
 	
`"Error while deleting the order"`	

<u>URL for API request</u>

GET - http://localhost:8090/Order_Service/OrderService/Orders/

GET - http://localhost:8090/Order_Service/OrderService/Orders/getOrderbyID/1

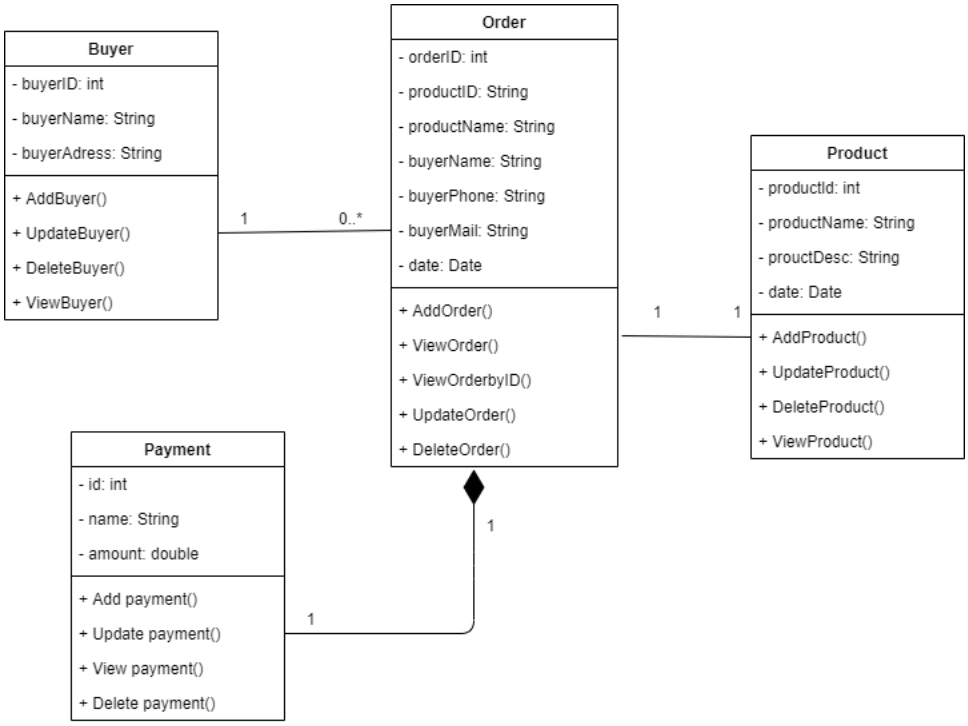
POST - http://localhost:8090/Order_Service/OrderService/Orders/add

PUT - http://localhost:8090/Order_Service/OrderService/Orders/update

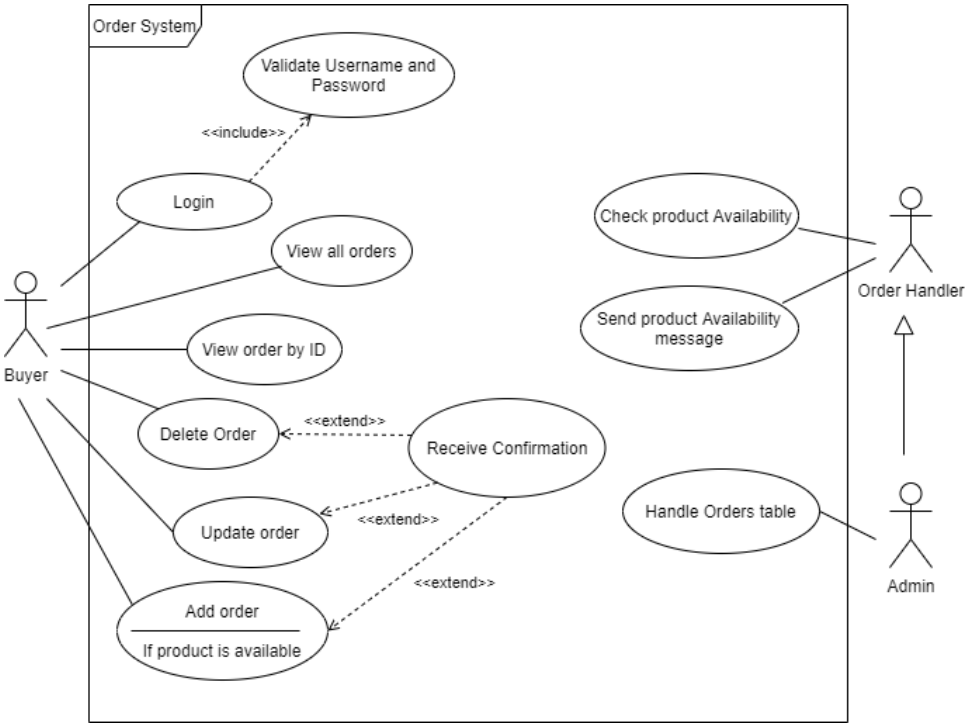
DELETE - http://localhost:8090/Order_Service/OrderService/Orders/delete

Internal logic design

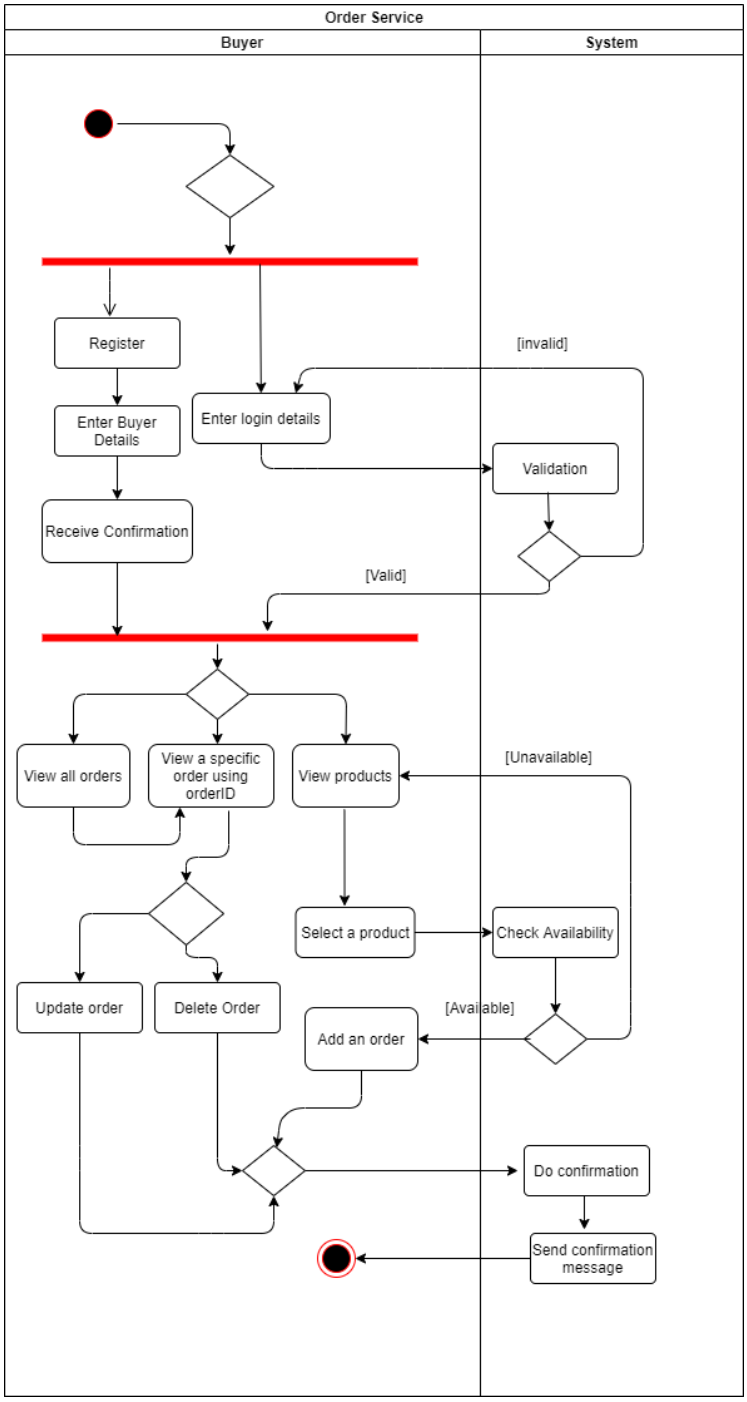
Class Diagram



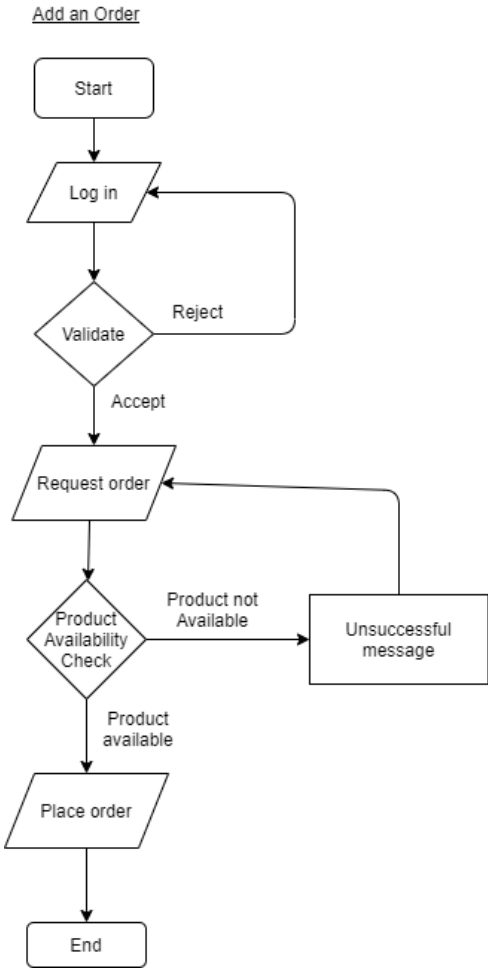
Use case Diagram

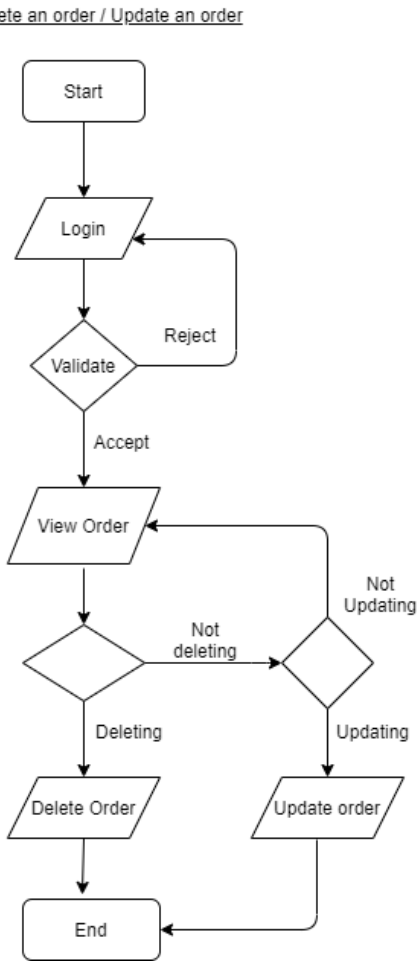


Activity Diagram



Flow Charts





Service Development and Testing

I. Dependency management tools

- IDE: Eclipse

  Database: MySQL

  Back End: Java

- Maven
- JAX-RS

II. Testing Tools

- Postman

III. Code quality checking tools

Testing methodology and results

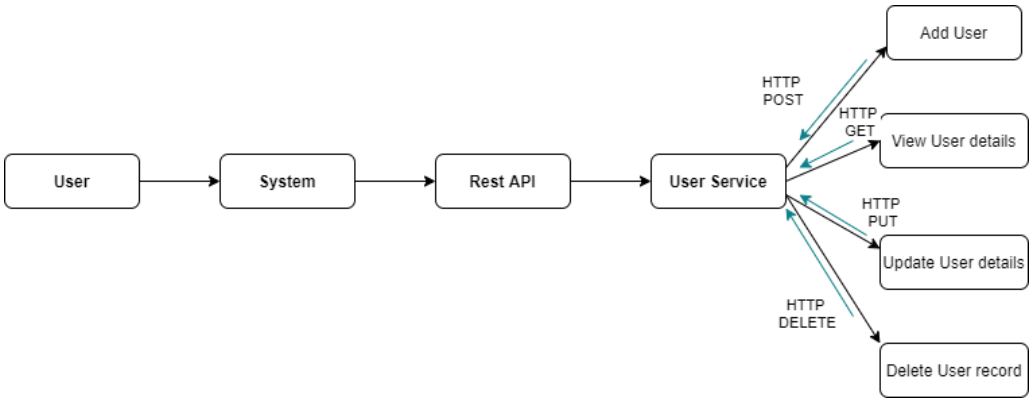
Test ID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
01	View All Orders.	URL for the API Send GET request.	Display orders details.	Display Orders details.	Pass
02	View Order By ID	orderID for the URL with a GET request	Display Order details of the given orderID	Display Order details of the given orderID	Pass
03	Add an order.	productID, productName, buyerName, buyerPhone, buyerMail, date	Display message as "Inserted successfully"	Display message as "Inserted successfully"	pass
04	Update an order.	Attributes to be updated along with the orderID.	Display message as "Updated successfully"	Display message as "Updated successfully"	pass
05	Delete an order.	orderID of the appointment to be deleted.	Display message as "Deleted successfully"	Display message as "Deleted successfully"	pass

Assumptions and Discussions

- Multiple products cannot be added in one order. (An order can be made only for a one product at once).
- Buyer can request a specific order detail by giving the order ID.
- A buyer can only place an order to a specific product after checking its availability.
- If a product already has an order, it cannot be reordered by another buyer.
- System sends confirmation message after confirmation of an order.
- Buyer receives confirmation messages after deleting or updating an order.

Gunarathne N. U - IT19122588

API for the Service



Resource: UserService API **<u>Request: GET (Get user details)</u>** Media: *TEXT_HTML* Response: String status message "Error while reading the users."	Resource: UserService API **<u>Request: POST (Add user details)</u>** Media: *APPLICATION_FORM_URLENCODED* Data: userID, name, nic, email, contact Response: String status message "Inserted successfully" "Error while inserting the user."
Resource: UserService API **<u>Request: PUT (Update User details)</u>** Media: *APPLICATION JSON* Data: userID, name, nic, email, contact Response: String status message "Updated successfully" "Error while updating the user."	Resource: UserService API **<u>Request: DELETE (Delete User record)</u>** Media: *APPLICATION XML* Data: userID Response: String status message "Deleted successfully" "Error Deleting"

URL for API request

GET: http://localhost:8080/UsersAPI/UserService/Users/all

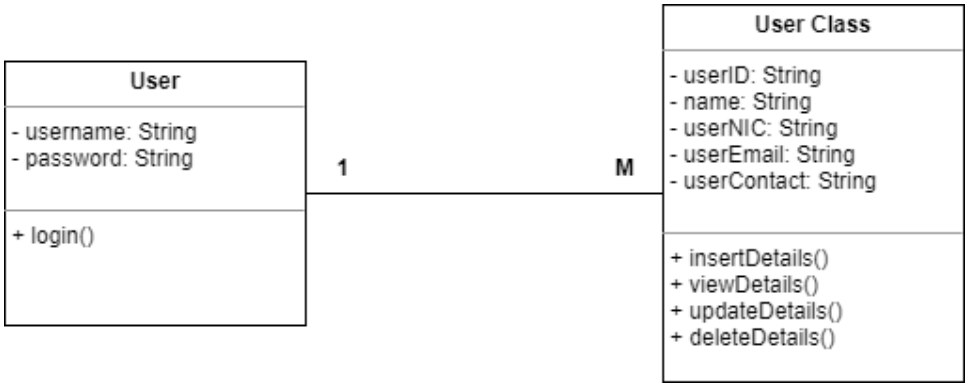
POST: http://localhost:8080/UsersAPI/UserService/Users

PUT: http://localhost:8080/UsersAPI/UserService/Users/

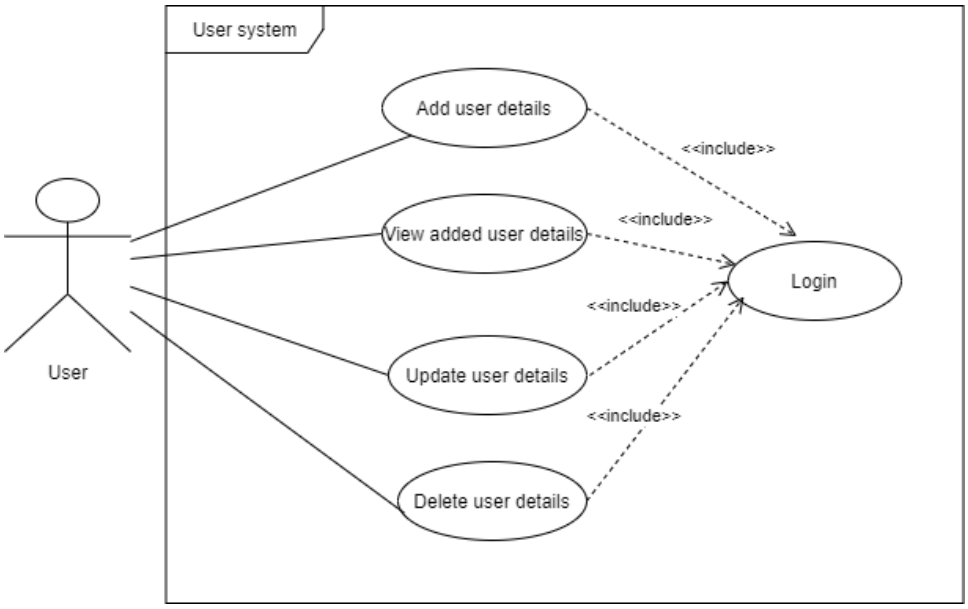
DELETE: http://localhost:8080/UsersAPI/UserService/Users

Internal logic design

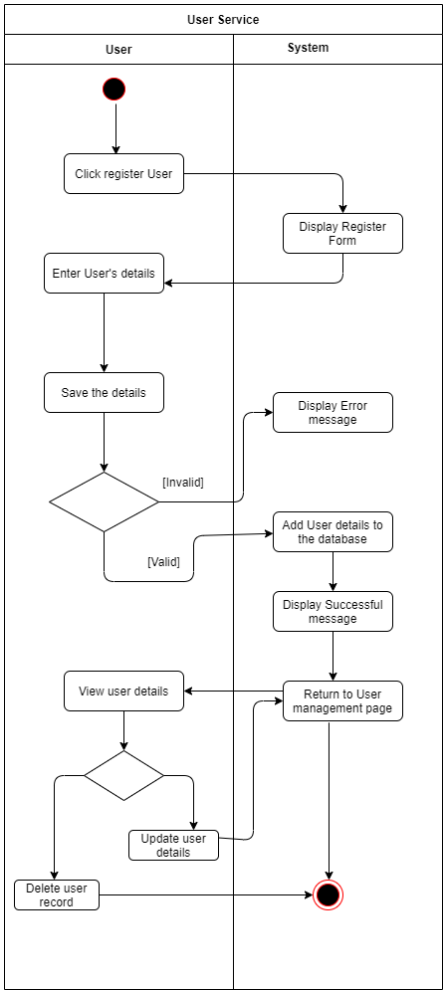
Class Diagram



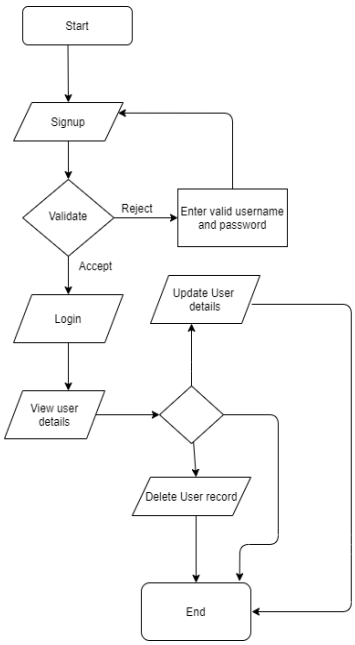
Use case Diagram



Activity Diagram



Flow Charts



Service Development and Testing

Testing methodology and result Methodology is testing APIs by sending request to the web server and getting the response back. It allows us to set up all the headers and cookies the API expects and checks the response. I have used mainly four request methods frequently, which are as below,

- POST Request - for inserting data
- PUT Request – For Updating data
- GET Request - For Retrieving/Fetching data
- DELETE Request - For Deleting data Tools used

Eclipse IDE: Programming environment

Postman: testing Tool

Git: for Version controlling

MySQL: to manage database

Tomcat server: to serve the API

Testing methodology and results

Test ID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/ Fail)
01	Add User details	userID name userNIC userEmail userContact	Display message as “Inserted successfully”	Display message as “Inserted successfully”	Pass
02	Invalid User details entered to the system	name: 12356	Display message "Error while inserting the user."	Display message as "Error while inserting the user."	Pass
03	Update User details.	Attributes to be updated along with the userID.	Display message as “Updated successfully”	Display message as “Updated successfully”	pass
04	Delete a user record	userID of the record to be deleted.	Display message as “Deleted successfully”	Display message as “Deleted successfully”	pass

Assumptions and Discussions

- User can view user details as well as the project/research details of his/her
- A confirmation message is received by the user upon registering to the system
- User can delete the whole record of the user details from the system
- Researchers can be added after registering to the system as a User, therefore user registration is a must

7 . References

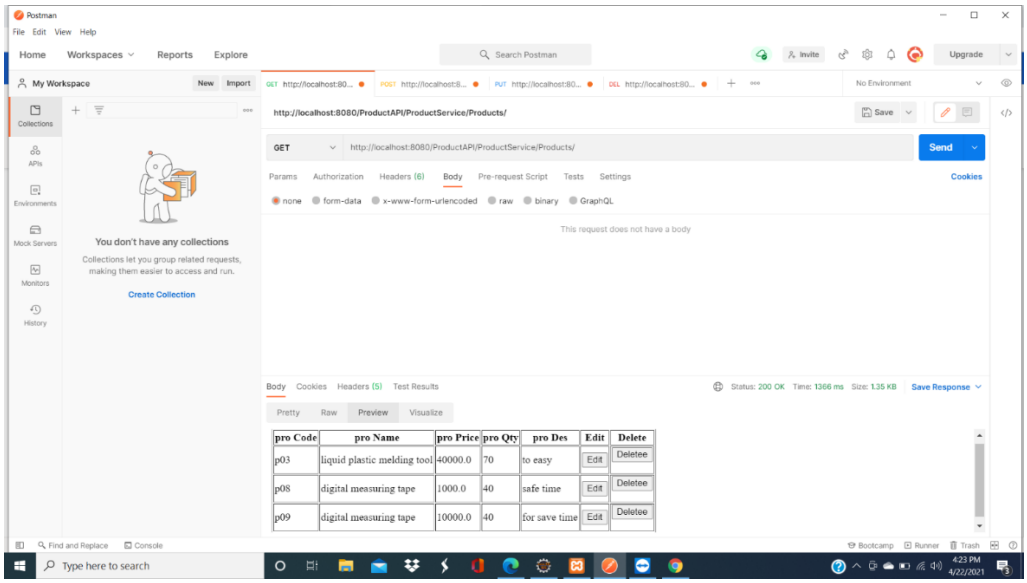
Lab sheets 5, 6/ Lecture 5

Maven Documentation - Maven – Maven Documentation (apache.org)

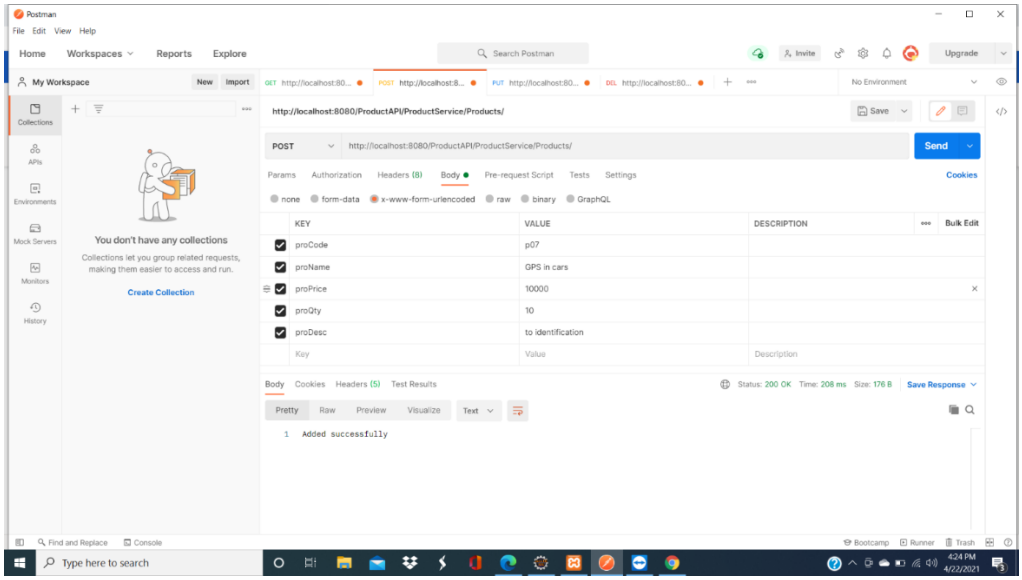
8. Appendix

1.Product Service(IT19010472 K.Mithusha)

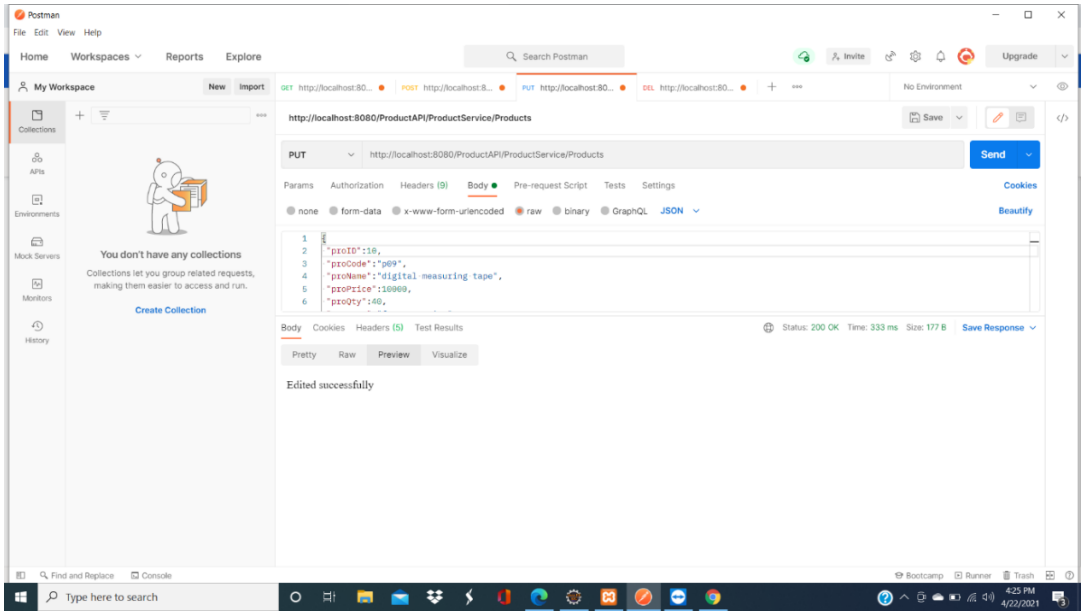
a. GET



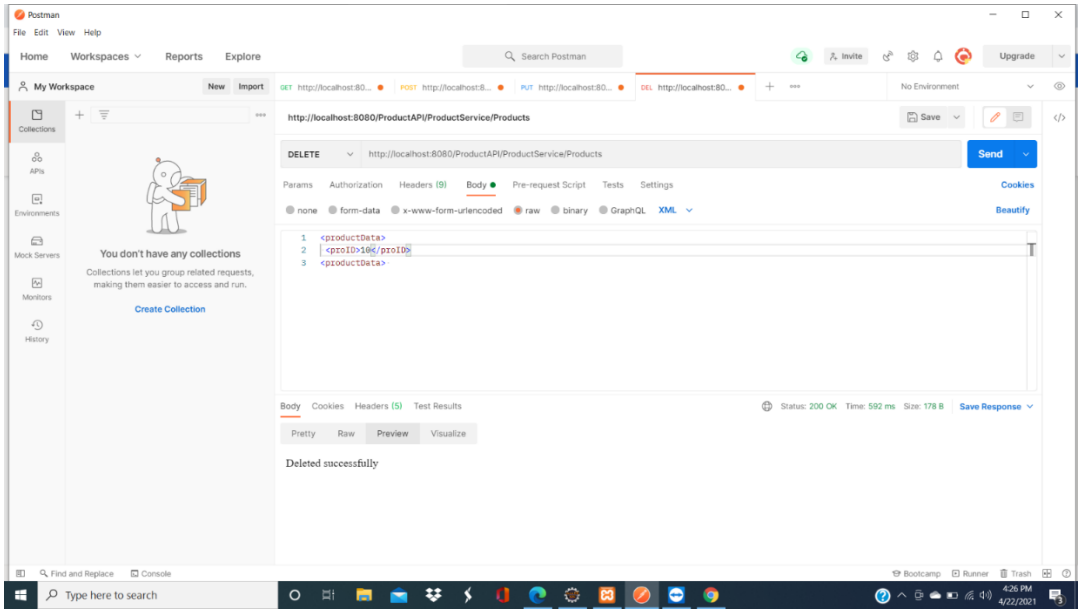
b.POST



C.PUT

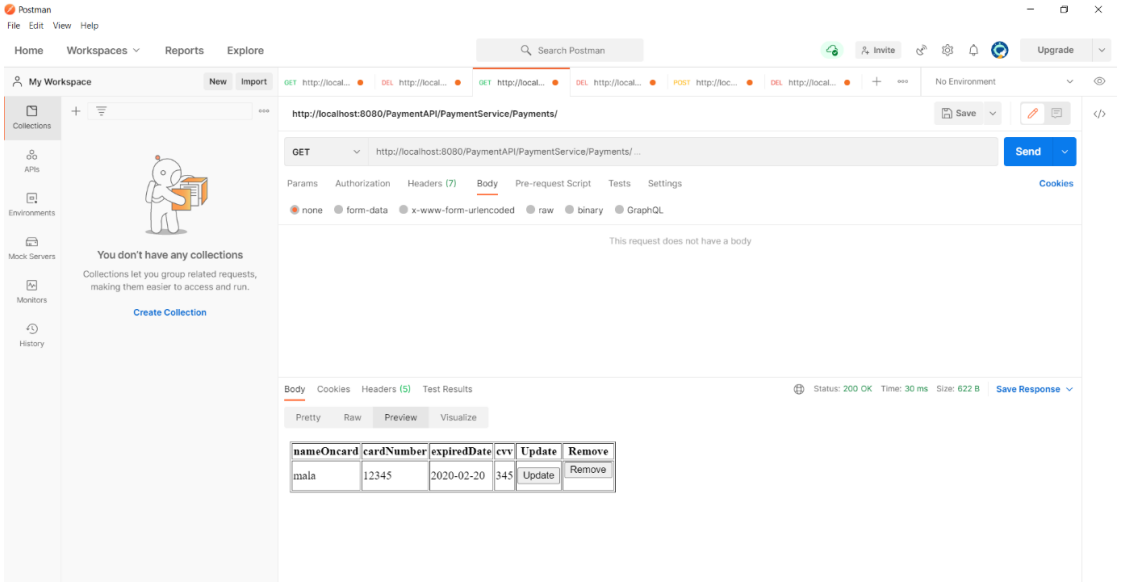


d.DELETE

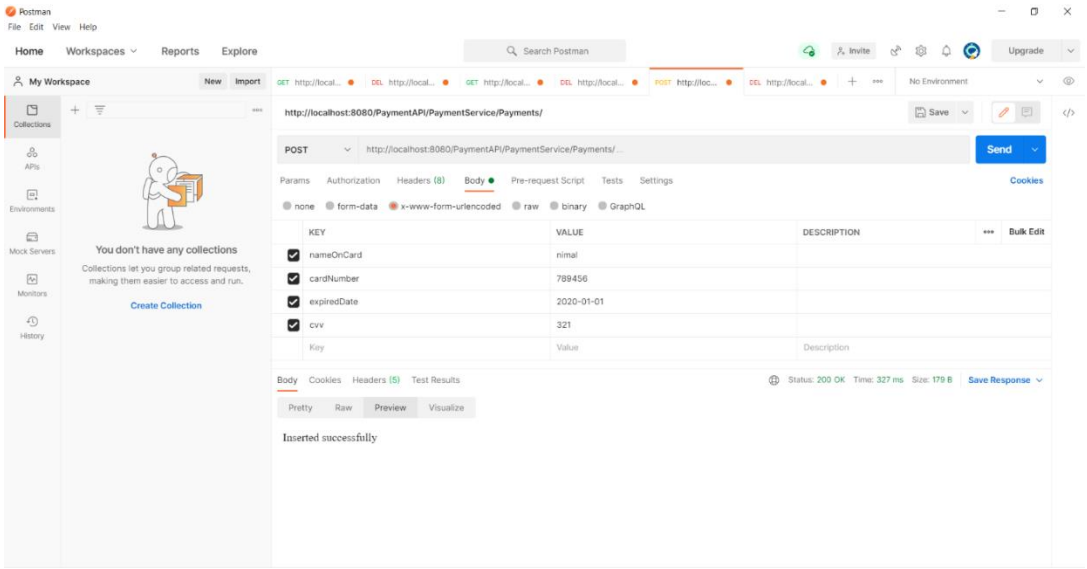


2.Payment Service(IT19012834 -Naveen S)

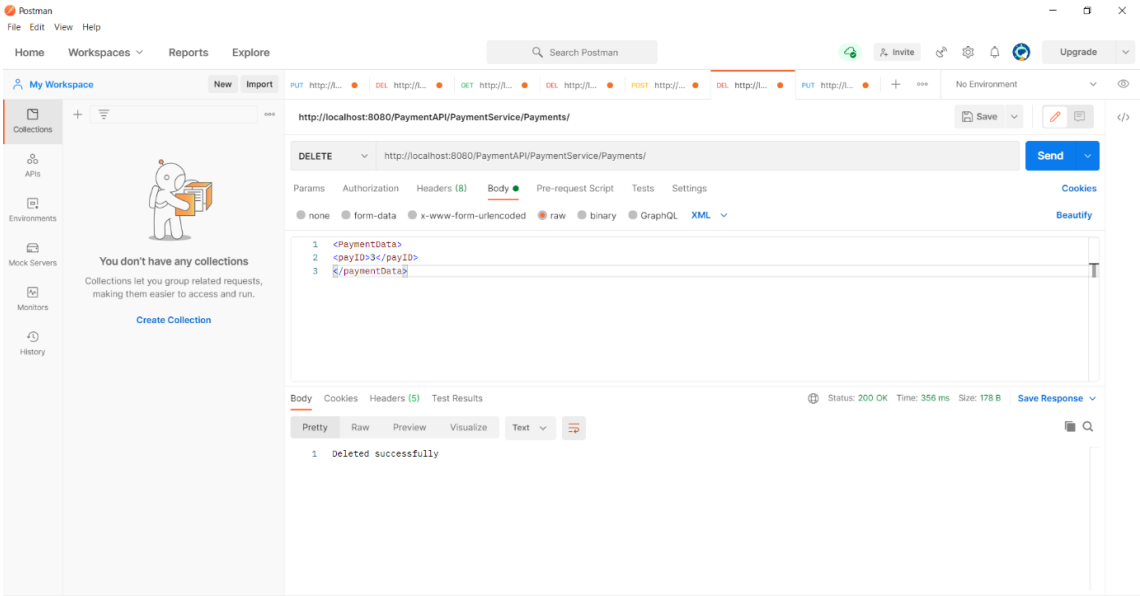
a)Get



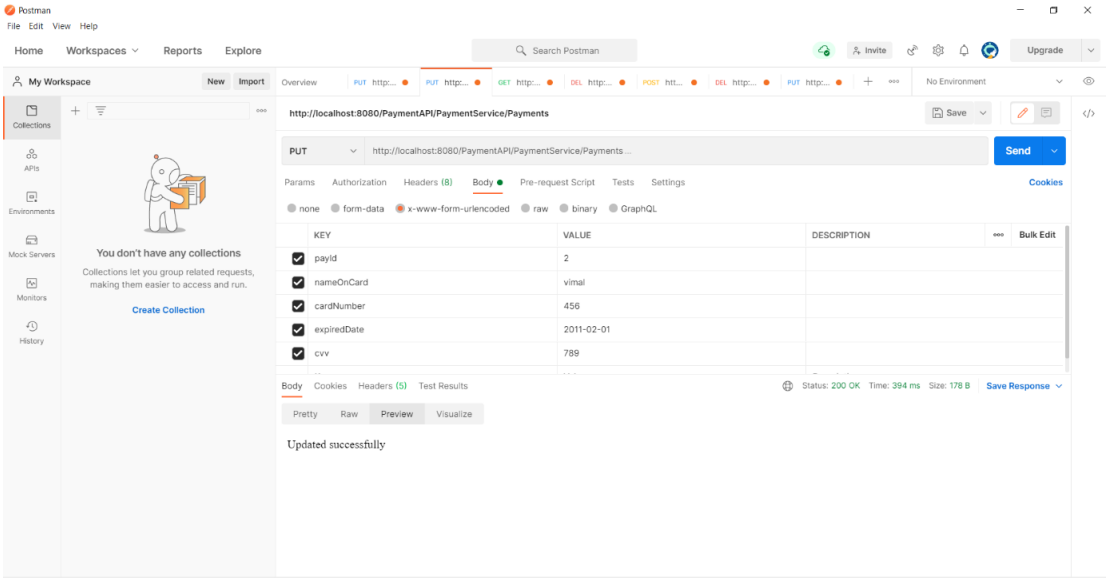
b)Post



c)Delete



d)Put



Cooray P.L.R.K – IT19111834 (Auth and Researches Services)

Login

PAF / **AuthAPI - userLogin**

GET http://localhost:8080/AuthenticationAPI/AuthService/Auth/login

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "username":"rajindu",
3     "password":"1234"
4 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
1 Successfully Logged
```

Verify User Profile

PAF / **AuthAPI - verifyLogin**

GET http://localhost:8080/AuthenticationAPI/AuthService/Auth/verify

Params Authorization Headers (6) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
1 Verified Profile
```

Verify Admin Profile

PAF / **AuthAPI - verifyAdmin**

GET http://localhost:8080/AuthenticationAPI/AuthService/Auth/adminVerify

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE
	Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
1  Verified Admin Profile
```

Logout

PAF / **AuthAPI - logout**

GET http://localhost:8080/AuthenticationAPI/AuthService/Auth/logout

Params Authorization Headers (6) Body Pre-request Script Tests Settin

Query Params

	KEY	VA
	Key	Va

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
1  Logout successfully
```

Add Research

PAF / **ResearchesAPI - addResearch**

POST http://localhost:8080/ResearchesAPI/ResearchService/Researches

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "title":"Object Detection",
3     "author":"Rajindu",
4     "catergory":"computer science",
5     "year":2021,
6     "path":"C:\\Users\\Rajindu\\Documents\\SLIIT\\Y3S1\\PAF\\pdfs\\sample.pdf"
7 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
1 Inserted successfully
```

Pending Approvals

PAF / **ResearchesAPI - pendingApprovals**

GET http://localhost:8080/ResearchesAPI/ResearchService/Researches/pending

Params Authorization Headers (6) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize

Research Title	Author	Catergory	Year	Action	Download
sample research	sample author	computer science	2021	Approve	Download
sample research2	sample author2	computer science	2021	Approve	Download
sample research3	sample author3	computer science	2021	Approve	Download
Object Detection	Rajindu	computer science	2021	Approve	Download

Download

PAF / **ResearchesAPI - download**

GET http://localhost:8080/ResearchesAPI/ResearchService/Researches/download

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "id":11,
    "path":"C:\\Users\\Rajindu\\Documents\\SLIIT\\Y3S1\\PAF\\pdfs\\",
    "file":"check5.pdf"
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
Downloaded Successfully C:\Users\Rajindu\Documents\SLIIT\Y3S1\PAF\pdfs
```

Approve Research

PAF / **ResearchesAPI - approveResearch**

PUT http://localhost:8080/ResearchesAPI/ResearchService/Researches/approve

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "id":11
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
Approved
```

Approved Researches

PAF / ResearchesAPI - approvedResearches

GET http://localhost:8080/ResearchesAPI/ResearchService/Researches

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize

Research Title	Author	Catergory	Year	Update	Delete
sample research3	sample author3	computer science	2021	Update	Remove
Object Detection	Rajindu	computer science	2021	Update	Remove

Update Research

PAF / ResearchAPI - updateResearch

PUT http://localhost:8080/ResearchesAPI/ResearchService/Researches

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "id":10,
    "title":"Object Identification",
    "author":"Cooray",
    "catergory":"AI",
    "year":2020
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
Updated successfully
```

Delete Research

PAF / **ResearchesAPI - deleteResearch**

DELETE http://localhost:8080/ResearchesAPI/ResearchService/Researches

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "id":11
3 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
1 Deleted successfully
```

Order Service (IT19118246-Wijesekera S.M)

1)GET- all orders

GET http://localhost:8090/Order_Service/OrderService/Orders/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Headers 6 hidden

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK Time: 118 ms Size: 1.27 KB Save

Pretty Raw Preview Visualize

OrderID	ProductID	Product Name	Buyer Name	Phone Number	Email	Date
2	P002	product2	buyer2	0712345674	buyer2@gmail.com	2021-01-07
3	P234	product3	buyer3	0912345670	buyer3@gmail.com	2021-01-12
4	P348	product4	buyer4	0912345345	buyer4@gmail.com	2021-12-12
8	P555	product8	buyer8	0774555983	buyer8@gmail.com	2021-02-20
9	P475	product9	buyer9	0764046667	buyer9@gmailcom	2021-03-05
10	P233	product10	buyer10	0784556677	buyer10@gmail.com	2021-03-07
15	P245	product15	buyer15	0784512377	buyer15@gmail.com	2021-12-04
16	P324	product16	buyer16	0784559677	buyer16@gmail.com	2021-12-04

2)GET- order by ID

GET http://localhost:8090/Order_Service/OrderService/Orders/getOrderbyID/3 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Headers 6 hidden

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK Time: 22 ms Size: 436 B Save

Pretty Raw Preview Visualize

OrderID	ProductID	Product Name	Buyer Name	Phone Number	Email	Date
3	P234	product3	buyer3	0912345670	buyer3@gmail.com	2021-01-12

3)POST

POST http://localhost:8090/Order_Service/OrderService/Orders/add Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
productID	P233	
productName	product	
buyerName	buyer	
buyerPhone	0784556677	
buyerMail	buyer@gmail.com	
date	2021-12-04	

Body Cookies Headers (5) Test Results Status: 200 OK Time: 304 ms Size: 179 B Sa

Pretty Raw Preview Visualize Text

```
1  Inserted successfully
```

4)PUT

PUT http://localhost:8090/Order_Service/OrderService/Orders/update Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1  {
2      "orderID":"4",
3      "productID":"P234",
4      "productName":"updatedproduct4",
5      "buyerName":"updatedbuyer4",
6      "buyerPhone":"091234567",
7      "buyerMail":"updatedbuyer@gmail.com",
8      "date":"2021-12-12"
9  }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 35 ms Size: 178 B Save

Pretty Raw Preview Visualize Text

```
1  Updated successfully
```

5)DELETE

DELETE http://localhost:8090/Order_Service/OrderService/Orders/delete Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL XML

```
1  <orderData>
2      <orderID>16</orderID>
3  </orderData>
4
```

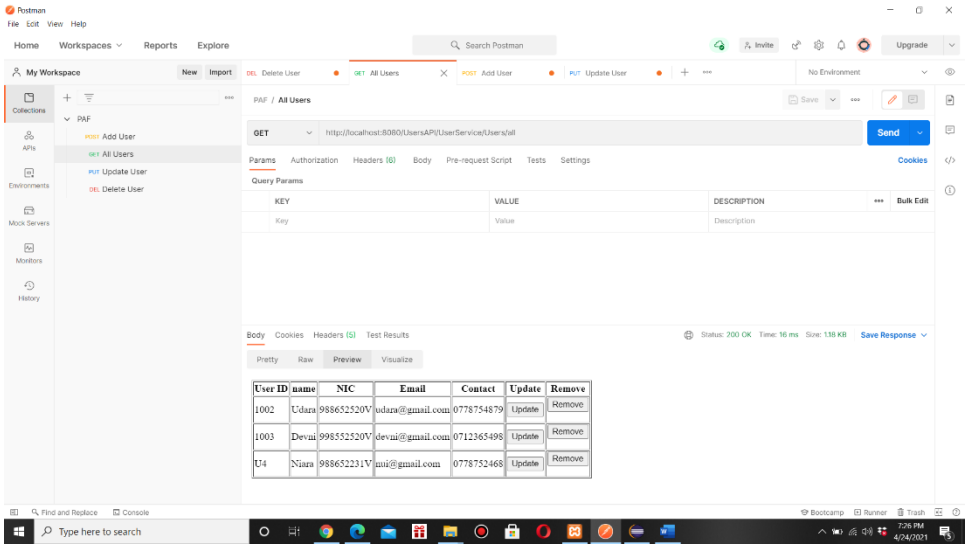
Body Cookies Headers (5) Test Results Status: 200 OK Time: 559 ms Size: 178 B Save

Pretty Raw Preview Visualize Text

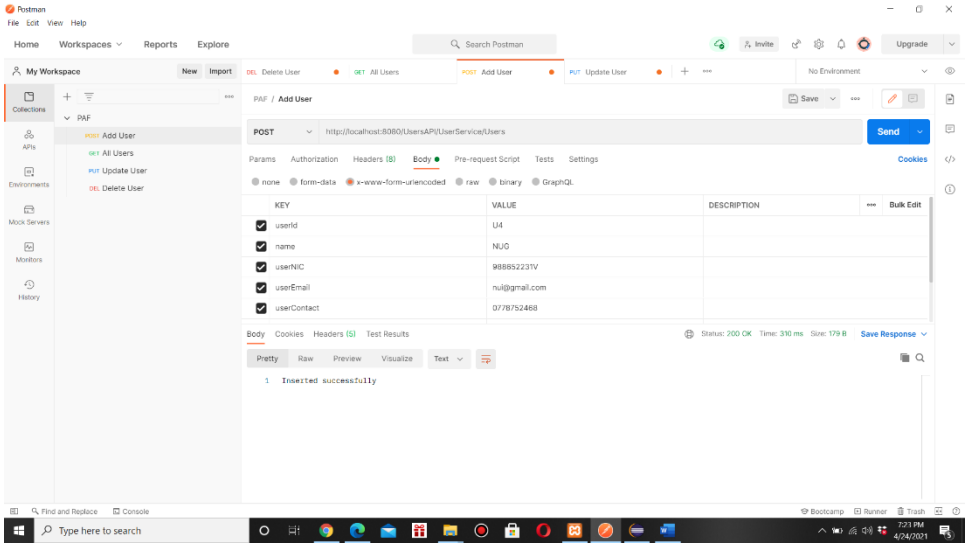
```
1  Deleted successfully
```

User Service (IT19122588- Gunarathne N.U.)

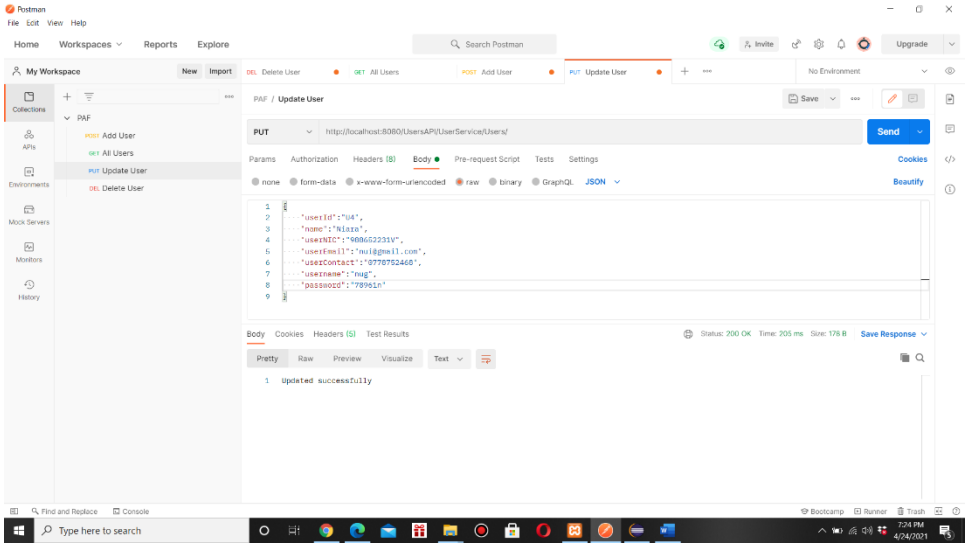
1)GET



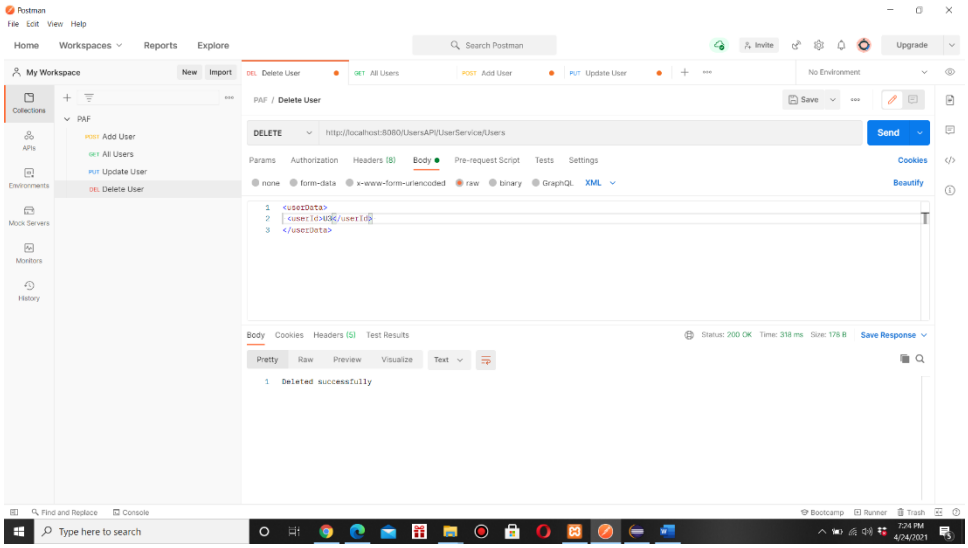
2)POST



3)PUT



4)DELETE



Commit Log

ShalaniMW committed 5 days ago — 937ce1f

Import packages to service class
ShalaniMW committed 5 days ago — 6ef0a08

Implement methods in the server-model
ShalaniMW committed 5 days ago — c3677e7

Commits on Apr 19, 2021

Implement database connection
ShalaniMW committed 5 days ago — b599baf

Create classes
ShalaniMW committed 5 days ago — 100e26c

Added dependencies
ShalaniMW committed 5 days ago — 31fb711

Commits on Apr 18, 2021

First Commit
ShalaniMW committed 6 days ago — 03b8b1d

Changed retrieving table format
ShalaniMW committed 2 days ago — a1e159a

Added readOrderbyID method
ShalaniMW committed 2 days ago — 107513b

Commits on Apr 21, 2021

Added date parameter
ShalaniMW committed 4 days ago — b1700c5

Added date parameter
ShalaniMW committed 4 days ago — 3aeabc0

Changed order class output alerts
ShalaniMW committed 4 days ago — cc0cd2f

Implement methods in service class ...
ShalaniMW committed 4 days ago — 60c24ab

Commits on Apr 20, 2021

Create order class object ...
ShalaniMW committed 4 days ago — 1abbfee

Servlet mapping
ShalaniMW committed 5 days ago — e066a3f

Servlet mapping
ShalaniMW committed 5 days ago — 937ce1f

Import packages to service class
ShalaniMW committed 5 days ago — 6ef0a08

Naveen Commit
RajinduSE committed 3 days ago — b9aee8c

Commits on Apr 21, 2021

Merge pull request #1 from RajinduSE/rajindu RajinduSE committed 3 days ago	Verified	a769d0d
Researches Service commit RajinduSE committed 3 days ago		10337a4
Researches API model commit RajinduSE committed 3 days ago		9acf3f7
Researches API web.xml RajinduSE committed 3 days ago		e68f324
Research API setup and POM RajinduSE committed 3 days ago		5d4b0bd
Authentication API web service commit RajinduSE committed 3 days ago		0d5bc94
Authentication API model class commit RajinduSE committed 3 days ago		f34885f
Authentication API web.xml commit RajinduSE committed 3 days ago		422a7c6
Authentication API Setup and POM RajinduSE committed 3 days ago		4f0addb

Commits on Apr 15, 2021

Initial commit	Verified	d9152b6

main

Commits on Apr 24, 2021

Merge pull request #4 from RajinduSE/nipuni Nipuni-UG committed 10 hours ago	Verified	23d5116
Nipuni Commit Nipuni-UG committed 10 hours ago		c36e1b4
Add files via upload ShalaniMW committed yesterday	Verified	962183a

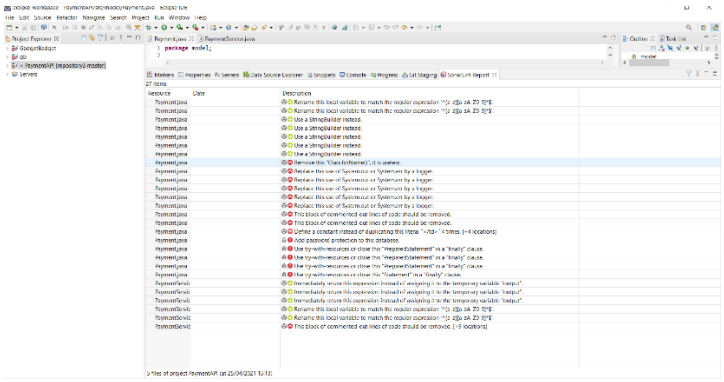
Commits on Apr 23, 2021

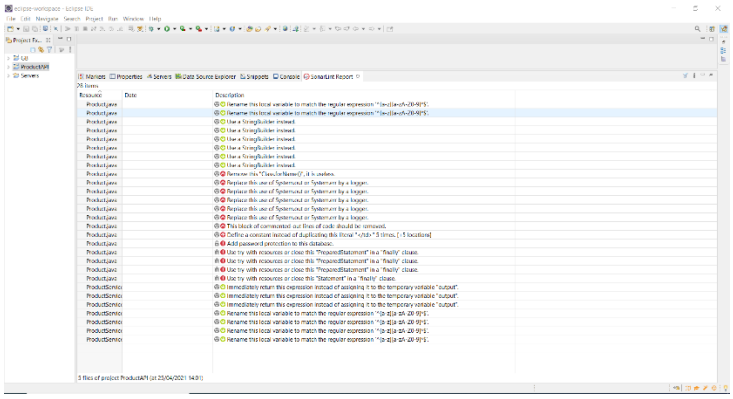
Merge pull request #3 from RajinduSE/Naveen RajinduSE committed yesterday	Verified	4ae4da6
Merge pull request #2 from RajinduSE/Mithusha Mit493 committed yesterday	Verified	409ec5c
Mithusha Commit Product API Mit493 committed yesterday		ad2b178

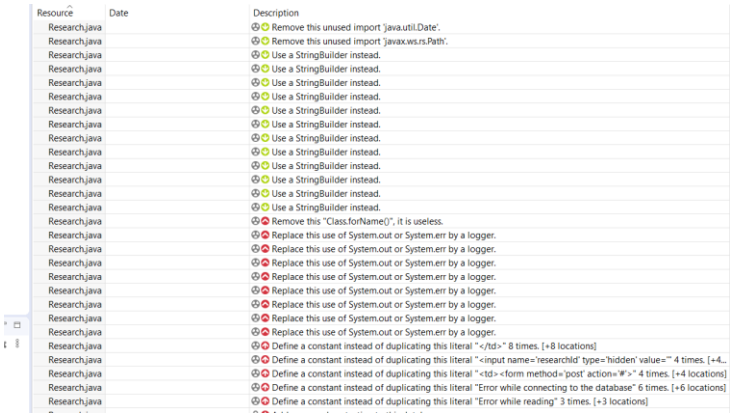
Commits on Apr 22, 2021

Naveen Final RajinduSE committed 3 days ago	9f45e62
Naveen Error RajinduSE committed 3 days ago	fa3a0ea
Naveen Commit	b9aee8c

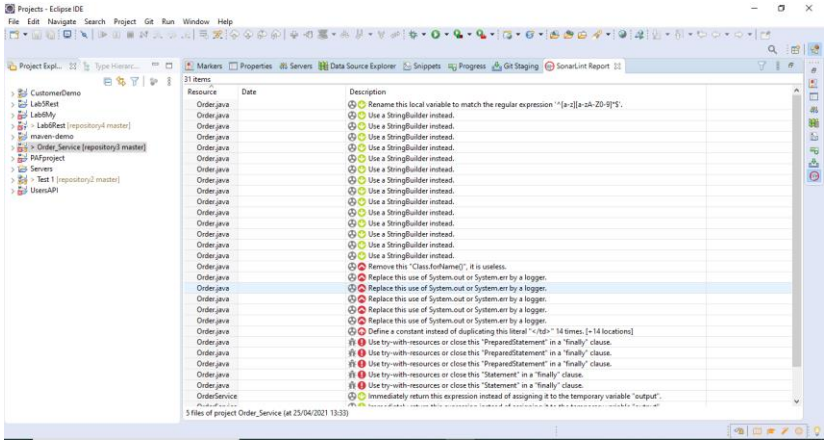
Code Quality Check(SonarLint)

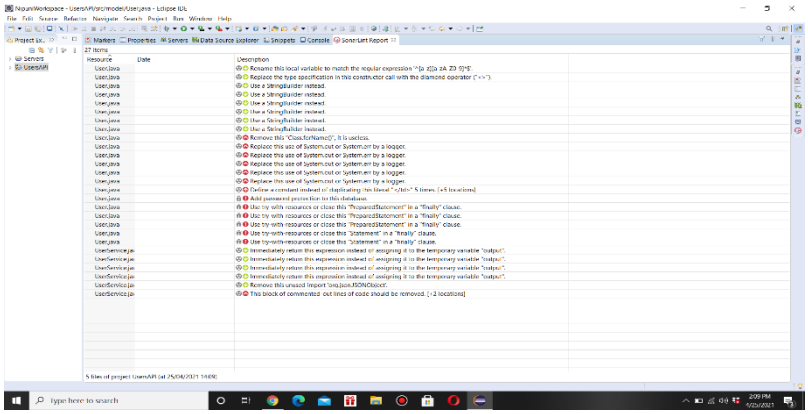






Resource	Date	Description
Research.java		Remove this unused import 'java.util.Date'.
Research.java		Remove this unused import 'javax.ws.rs.Path'.
Research.java		Use a StringBuilder instead.
Research.java		Use a StringBuilder instead.
Research.java		Use a StringBuilder instead.
Research.java		Use a StringBuilder instead.
Research.java		Use a StringBuilder instead.
Research.java		Use a StringBuilder instead.
Research.java		Use a StringBuilder instead.
Research.java		Use a StringBuilder instead.
Research.java		Use a StringBuilder instead.
Research.java		Use a StringBuilder instead.
Research.java		Use a StringBuilder instead.
Research.java		Remove this "Class.forName()", it is useless.
Research.java		Replace this use of System.out or System.err by a logger.
Research.java		Replace this use of System.out or System.err by a logger.
Research.java		Replace this use of System.out or System.err by a logger.
Research.java		Replace this use of System.out or System.err by a logger.
Research.java		Replace this use of System.out or System.err by a logger.
Research.java		Replace this use of System.out or System.err by a logger.
Research.java		Replace this use of System.out or System.err by a logger.
Research.java		Replace this use of System.out or System.err by a logger.
Research.java		Define a constant instead of duplicating this literal "</td>" 8 times. [+8 locations]
Research.java		Define a constant instead of duplicating this literal "<input name='researchId' type='hidden' value='" 4 times. [+4...
Research.java		Define a constant instead of duplicating this literal "<td><form method='post' action='#'>" 4 times. [+4 locations]
Research.java		Define a constant instead of duplicating this literal "Error while connecting to the database" 6 times. [+6 locations]
Research.java		Define a constant instead of duplicating this literal "Error while reading" 3 times. [+3 locations]


Projects - Eclipse IDE
File Edit Navigate Search Project Git Run Window Help
Project Expl...
Type Hierarc...
CustomerDemo
LabSRest
LabMy
> LabSRest [repository4 master]
maven-demo
> Order_Service [repository3 master]
PAFproject
Servers
> Test 1 [repository2 master]
UsersAPI
Markers
Properties
Servers
Data Source Explorer
Snippets
Progress
Git Staging
SonarLint Report
31 items
Resource
Date
Description
Order.java
Rename this local variable to match the regular expression '^[a-z][a-zA-Z0-9]*$'.
Use a StringBuilder instead.
Remove this "Class.forName()", it is useless.
Replace this use of System.out or System.err by a logger.
Define a constant instead of duplicating this literal "</td>" 14 times. [+14 locations]
Use try-with-resources or close this "PreparedStatement" in a "finally" clause.
Use try-with-resources or close this "Statement" in a "finally" clause.
OrderService
Immediately return this expression instead of assigning it to the temporary variable "output".
5 files of project Order_Service (at 25/04/2021 13:33)


NopaniWorkspace - UsersAPI/src/model/User.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Ex...
Servers
UsersAPI
Markers
Properties
Servers
Data Source Explorer
Snippets
Console
SonarLint Report
27 items
Resource
Date
Description
User.java
Rename this local variable to match the regular expression '^[a-z][a-zA-Z0-9]*$'.
Replace the type specification in this constructor call with the diamond operator ("<>").
Use a StringBuilder instead.
Remove this "Class.forName()", it is useless.
Replace this use of System.out or System.err by a logger.
Add password protection to this database.
Use try-with-resources or close this "PreparedStatement" in a "finally" clause.
Use try-with-resources or close this "Statement" in a "finally" clause.
UserService.jsp
Immediately return this expression instead of assigning it to the temporary variable "output".
Remove this unused import 'org.json.JSONObject'.
This block of commented-out lines of code should be removed. [+2 locations]
5 files of project UsersAPI (at 25/04/2021 14:09)
Type here to search
2:09 PM
4/25/2021