Programming Applications and Frameworks (IT3030)
3rd Year, 1st Semester

Assignment 2021 S1 – Group Project
(Group S1.12.01.07)

GadgetBadget System

Submitted to
Sri Lanka Institute of Information Technology

Student Registration No.	Student Name
IT19033938	Samaranayake S.L.
IT19121116	Samarakkody S R S S U
IT19171302	T.V. Thimira Isiwara Vithanage
IT19037066	De Silva W.A.D.S.

In partial fulfillment of the requirements for the
Bachelor of Science Special Honors Degree in Information Technology.

25/04/2021

Contents

Section 1.01 Workload Distribution

Student ID	Student Name	Workload Distribution
IT19033938	Samaranayake S.L.	**Researcher Service** • This service is mainly focused on the management of research. It performs the CRUD operations of the research that are in the system. This is managed by the Researcher.
IT19121116	Samarakkody S R S S U	**Funding Company Service** • Focuses on the CRUD operations on adding and manage requirements of funding company. • This is managed by the relevant funding companies.
IT19171302	T.V. Thimira Isiwara Vithanage	**Buyer Service** • This service is mainly focused on managing the payment services when buyer buying a research project. • This is managed by the buyer.
IT19037066	De Silva W.A.D.S.	**User management and Rating Service** • In this service it mainly focused on the user appointment handling. Registered users are privileged to make appointments and handle their appointments. (CRUD)

<u>Section 1.02</u> <u>Public VCS Repository</u>

Git Repository Link:
https://github.com/shenaljfx/GadgetBadget.git

We added our projects to GitHub as separate branches and add them to the main repository through git bash coding. All the branches are named as Student ID numbers.

<u>Section 1.03</u> <u>SE Methods</u>

Understanding Customer Requirements - The purpose of the project is to create a platform for researchers to sell their research projects, so this platform should be user friendly, secured, simple, fast, and cost – effective. It deals with the collection of research information, payment details, etc.

Requirements Analysis - Here we involve exploring issues related to functional, non-functional specifications and technical requirements. We discussed the goals, deadlines, features like what we are going to implement in this project. We decided to engage the team via GitHub. We identified the possible web services (Mentioned on workload distribution) to be implemented as micro services. We identified RESTful Web service: Java - JAX-RS (Jersy) on Tomcat, DB: MySQL and testing tool postman as technical requirements to develop this. We mainly focused on the web services that we must decide through the requirements which are available.

Creating a Design - After analysis we designed the webservices as micro services that we decided in requirement analysis. GadgetBadget system is designed for connect researchers, buyers and funding companies in a same platform.

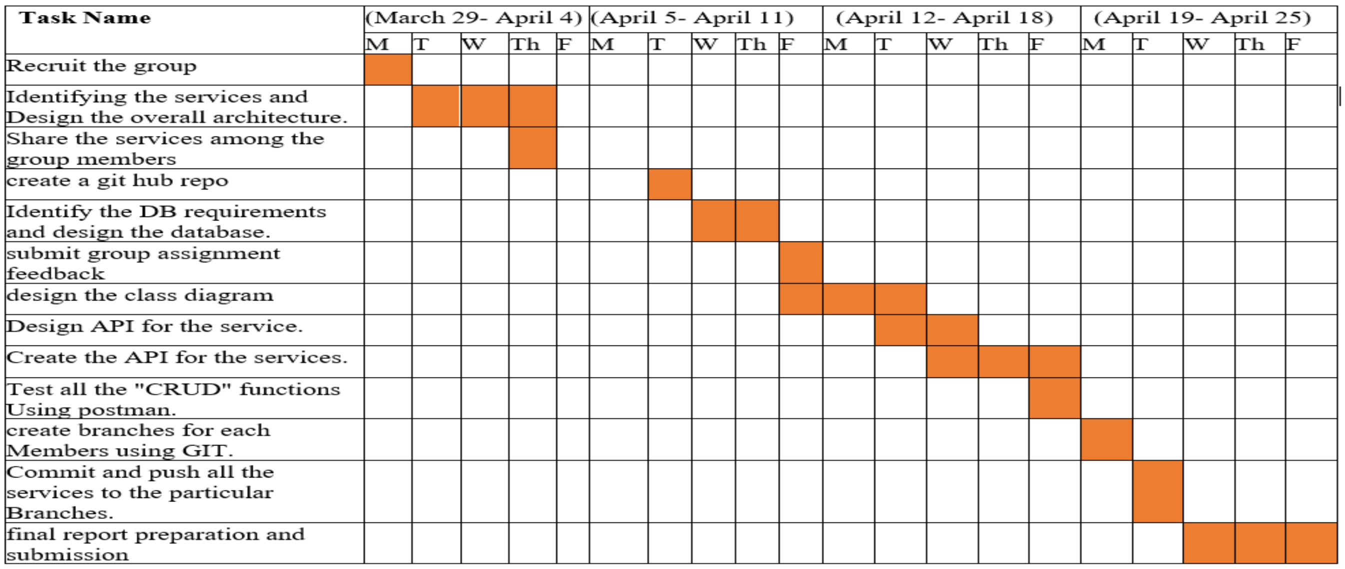
Coding, Testing, and Installation - We developed the system using RESTful Web service: Java - JAX-RS (Jersy) on Tomcat, DB: MySQL and tested it using postman tool. We wrote the code for the APIs, CRUD functions and created the database and connected it to the services to develop the system. We used Object Oriented Programming concepts for coding.

Maintenance - Maintenance is the application of each of the above steps to the existing modules in the system in order to modify or add new features, depending on what the customer needs.

For this project Incremental Model is used as the software Engineering Methodology.

Section 1.04 Time Schedule

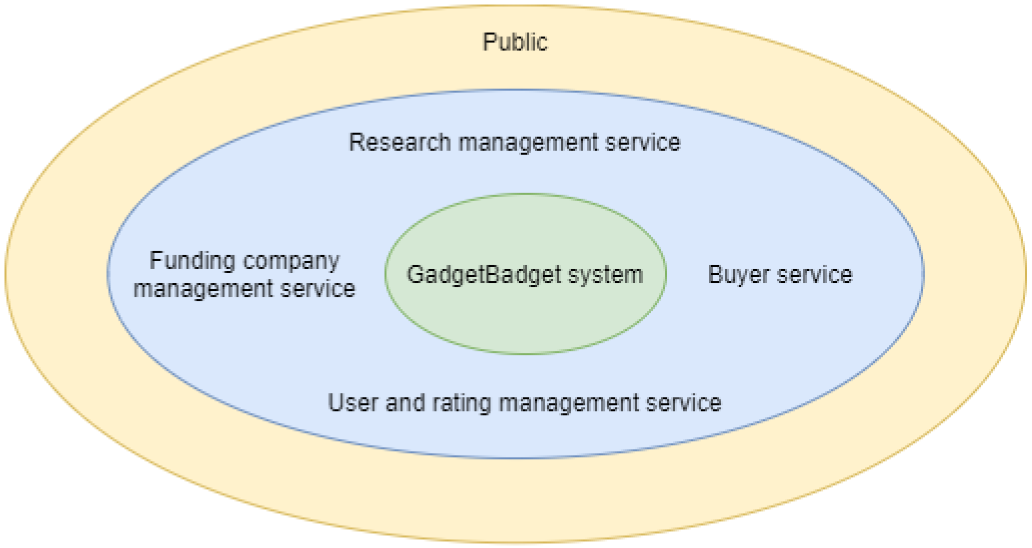
➢ Time Schedule



Task Name	(March 29- April 4)					(April 5- April 11)					(April 12- April 18)					(April 19- April 25)				
	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F
Recruit the group	🟧																			
Identifying the services and Design the overall architecture.		🟧	🟧	🟧																
Share the services among the group members				🟧																
create a git hub repo							🟧													
Identify the DB requirements and design the database.								🟧	🟧											
submit group assignment feedback										🟧										
design the class diagram										🟧	🟧	🟧								
Design API for the service.												🟧	🟧							
Create the API for the services.													🟧	🟧	🟧					
Test all the "CRUD" functions Using postman.															🟧					
create branches for each Members using GIT.																🟧				
Commit and push all the services to the particular Branches.																	🟧			
final report preparation and submission																		🟧	🟧	🟧

Section 1.05 Requirements

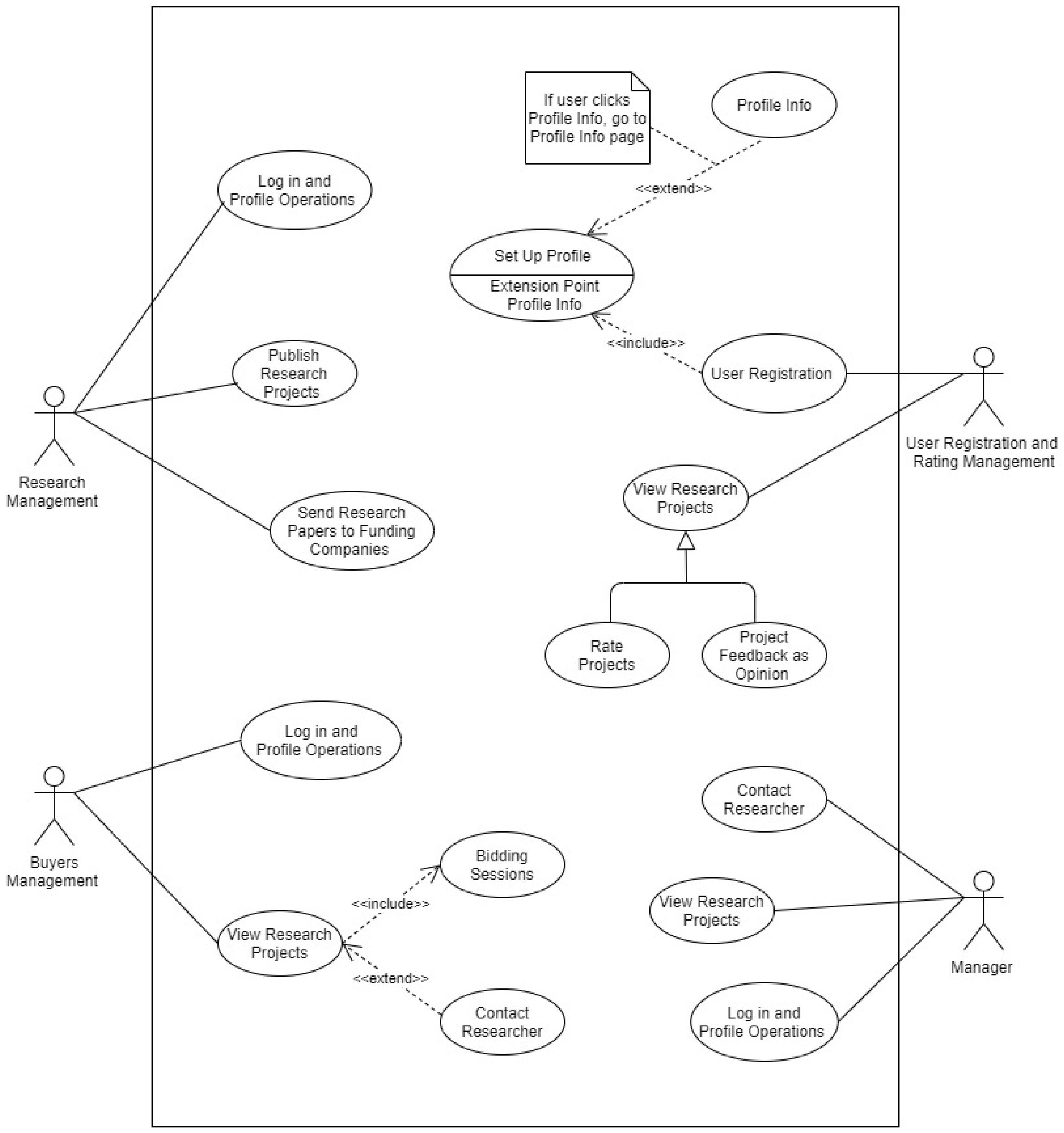
Stakeholder Analysis


Public
Research management service
Funding company management service
GadgetBadget system
Buyer service
User and rating management service

Requirements Analysis

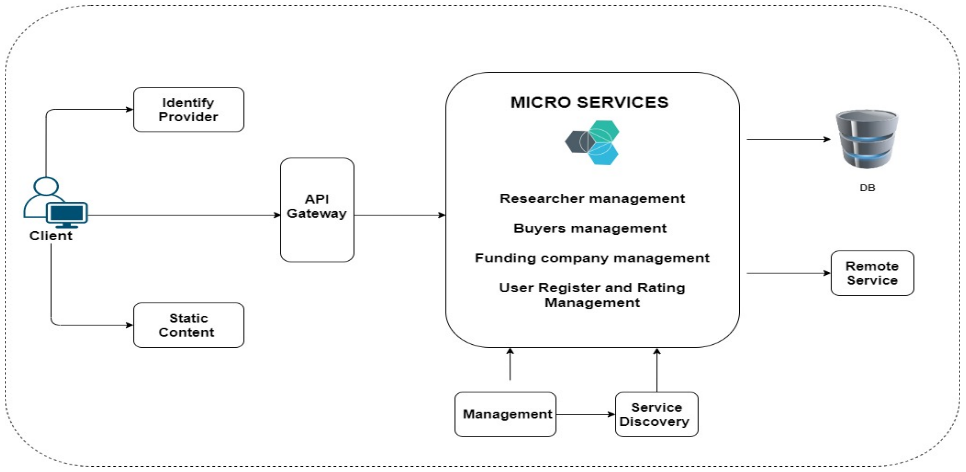
Here we involve exploring issues related to functional, non-functional specifications and technical requirements. We discussed the goals, deadlines, features like what we are going to implement in this project. We decided to engage the team via GitHub. We wrote detailed use case scenario to analyses the requirements like pre and post conditions, actors & basic flow etc. We identified the possible web services to be implemented as micro services. They are Researcher, Rating and user register, Buyer and Funding companies. We used RESTful Web service: Java - JAX-RS (Jersy) on Tomcat, DB: MySQL and testing tool postman as technical requirements to develop this.

Requirements modelling

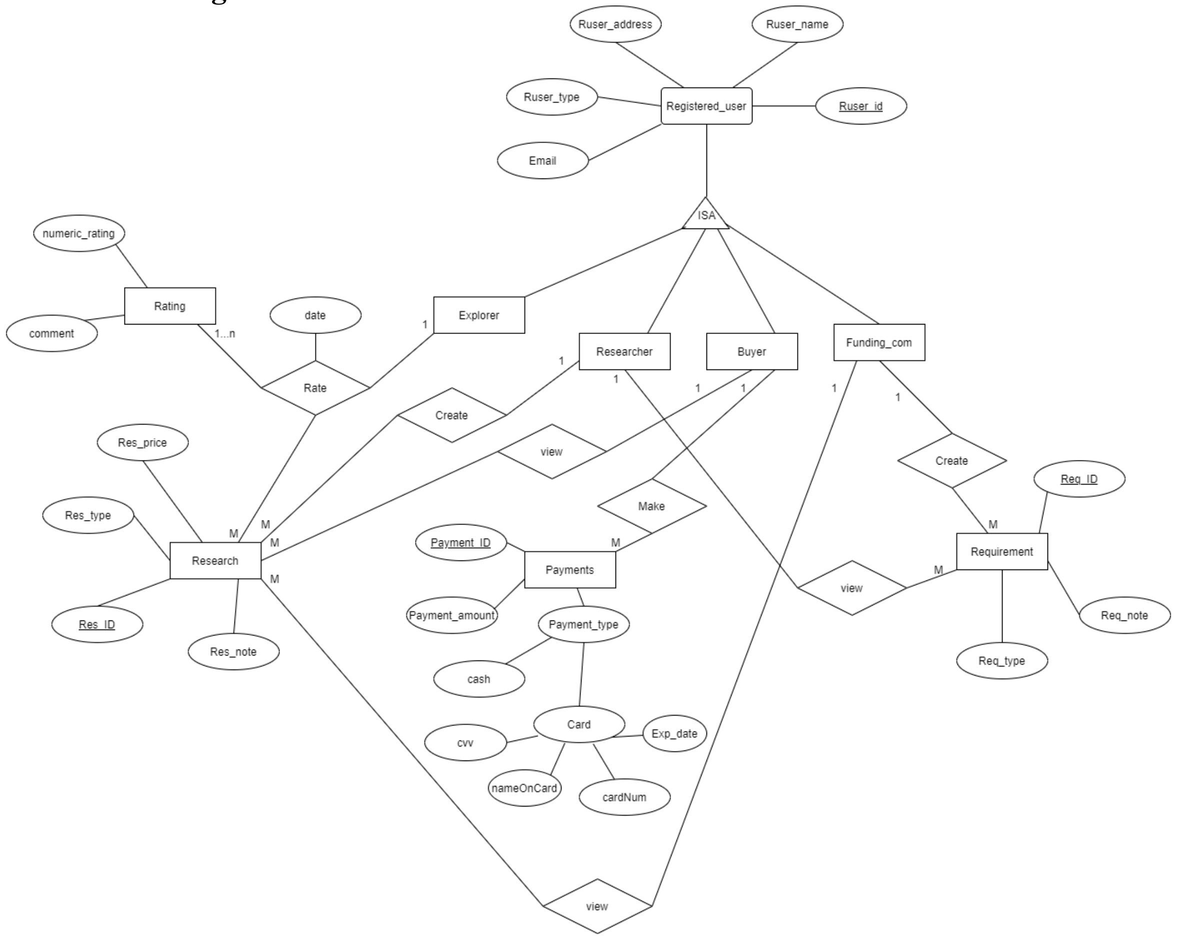

If user clicks Profile Info, go to Profile Info page
Profile Info
<<extend>>
Log in and Profile Operations
Set Up Profile
Extension Point Profile Info
<<include>>
User Registration
Publish Research Projects
User Registration and Rating Management
Research Management
Send Research Papers to Funding Companies
View Research Projects
Rate Projects
Project Feedback as Opinion
Log in and Profile Operations
Contact Researcher
Buyers Management
Bidding Sessions
<<include>>
View Research Projects
View Research Projects
Manager
<<extend>>
Contact Researcher
Log in and Profile Operations

Section 1.06 System's Overall Design

Overall Architecture



ER Diagram



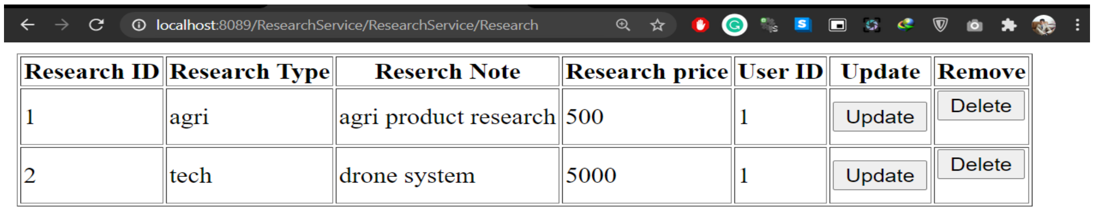
Section 1.07 Individual section

Reserch Service (Samaranayake S.L IT19033938)

Service Design

E.g. :http://localhost:8089/ResearchService/ ResearchService /Research

So, here is the API address which I run on my tomcat server in eclipse. Localhost:8089 is the address of the host server. ResearchService(1) is the name of the project. ResearchService is the URL-pattern given in the web.xml file. Research is the path to get the database connection to the project. If the connection is ok, the database will appear on the server window. If not, error message will be display on the window. All the CRUD functions are done by this and testing is below in my individual section.

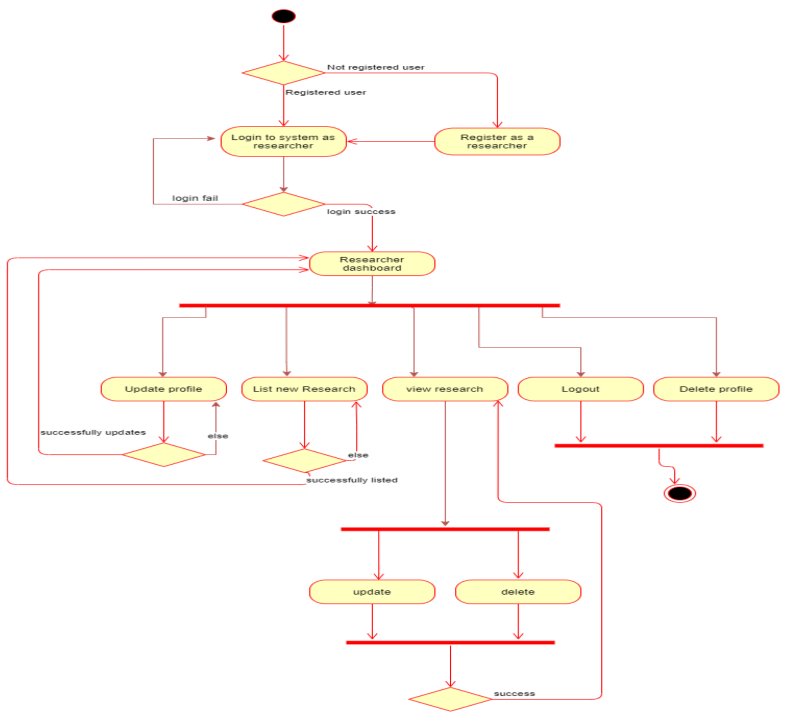


localhost:8089/ResearchService/ResearchService/Research

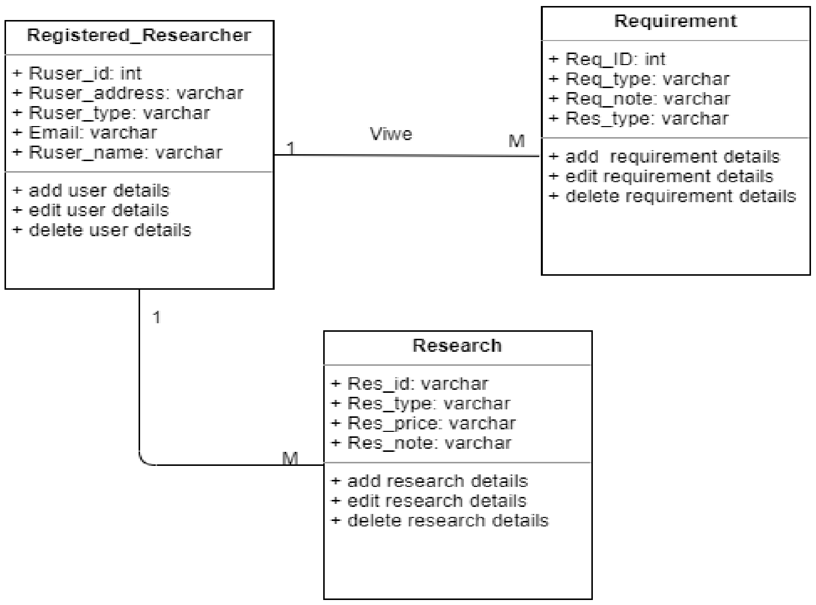
Research ID	Research Type	Reserch Note	Research price	User ID	Update	Remove
1	agri	agri product research	500	1	Update	Delete
2	tech	drone system	5000	1	Update	Delete

Internal Logic

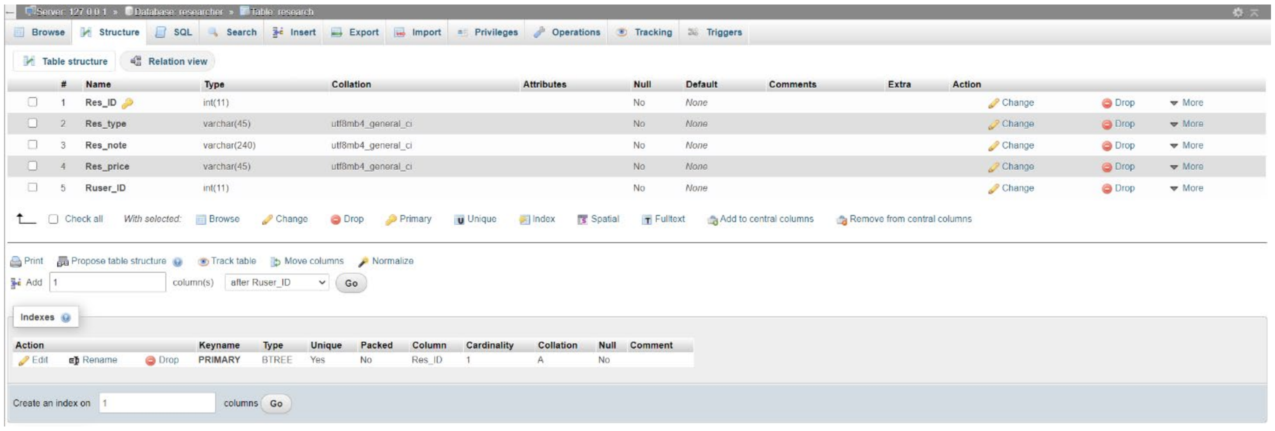
Microservice architectures will use libraries, but their primary way of componentizing their own software is by breaking down into services (The services which we divided like above.). We define libraries as components that are linked into a program and called using in-memory function calls, while service (Research service) are out-of-process components who communicate with a mechanism such as a web service request. Usually the best approach is to use what is known as an API Gateway.



Here the user first login to the system. After successfully login to the system, the user redirect to the Research service dashboard. At the user can update their details and they can delete it if necessary. They can also list a new research details and view all listed research details. After that it will redirect to the home page.



DB Structure


Server: 127.0.0.1 » Database: researcher » Table: research
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
Triggers
Table structure
Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Res_ID	int(11)			No	*None*			Change Drop More
2	Res_type	varchar(45)	utf8mb4_general_ci		No	*None*			Change Drop More
3	Res_note	varchar(240)	utf8mb4_general_ci		No	*None*			Change Drop More
4	Res_price	varchar(45)	utf8mb4_general_ci		No	*None*			Change Drop More
5	Ruser_ID	int(11)			No	*None*			Change Drop More

Check all
With selected: Browse Change Drop Primary Unique Index Spatial Fulltext Add to central columns Remove from central columns
Print Propose table structure Track table Move columns Normalize
Add 1 column(s) after Ruser_ID Go
Indexes

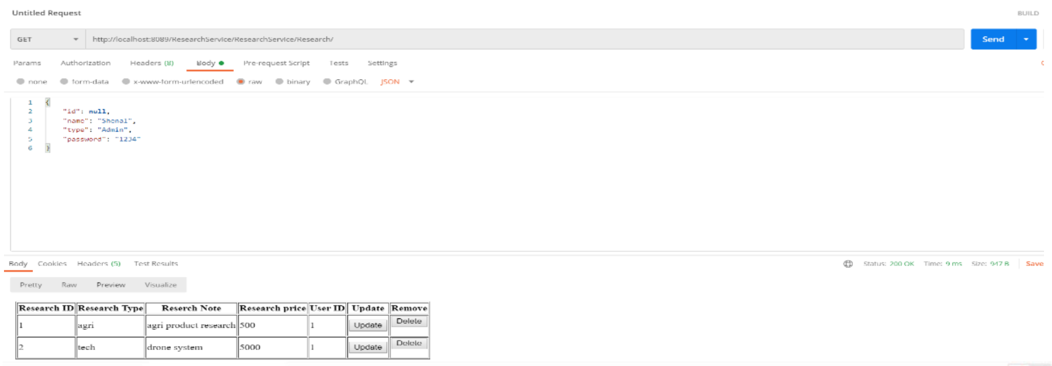
Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Rename Drop	PRIMARY	BTREE	Yes	No	Res_ID	1	A	No	

This is the DB design for my service. The DB is named as Researcher and the research is the table. Res_ID is the primary key of the table. All other attributes are above

Service development and Testing

Maven is a build automation tool used primarily for Java projects. Dependency management tool that I used is Maven. “Postman” is the testing tool that I used. Testing methodology and the results are below. (CRUD functions that I tested)

- **GET** - use to retrieve resources
- **POST** - accept the entity enclosed in the request as a new resource
- **PUT** - support the updating of REST resources
- **DELETE** – delete RESTful resources

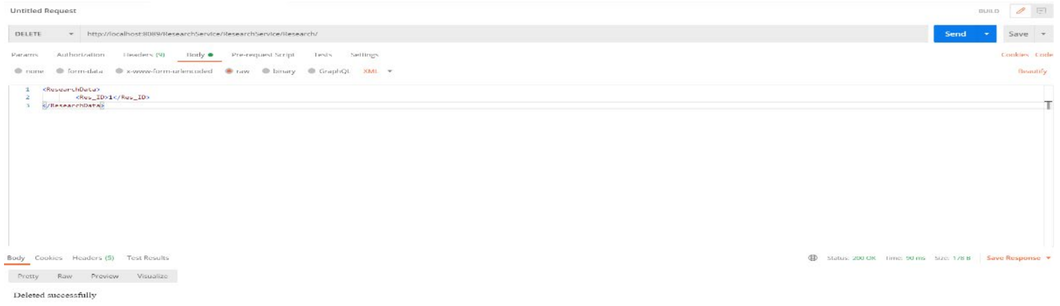


```
GET http://localhost:8089/ResearchService/ResearchService/Research/
```

```
{
    "id": null,
    "name": "Shenal",
    "type": "Admin",
    "password": "1234"
}
```

Status: 200 OK Time: 9 ms Size: 947 B

Research ID	Research Type	Reserch Note	Research price	User ID	Update	Remove
1	agri	agri product research	500	1	Update	Delete
2	tech	drone system	5000	1	Update	Delete

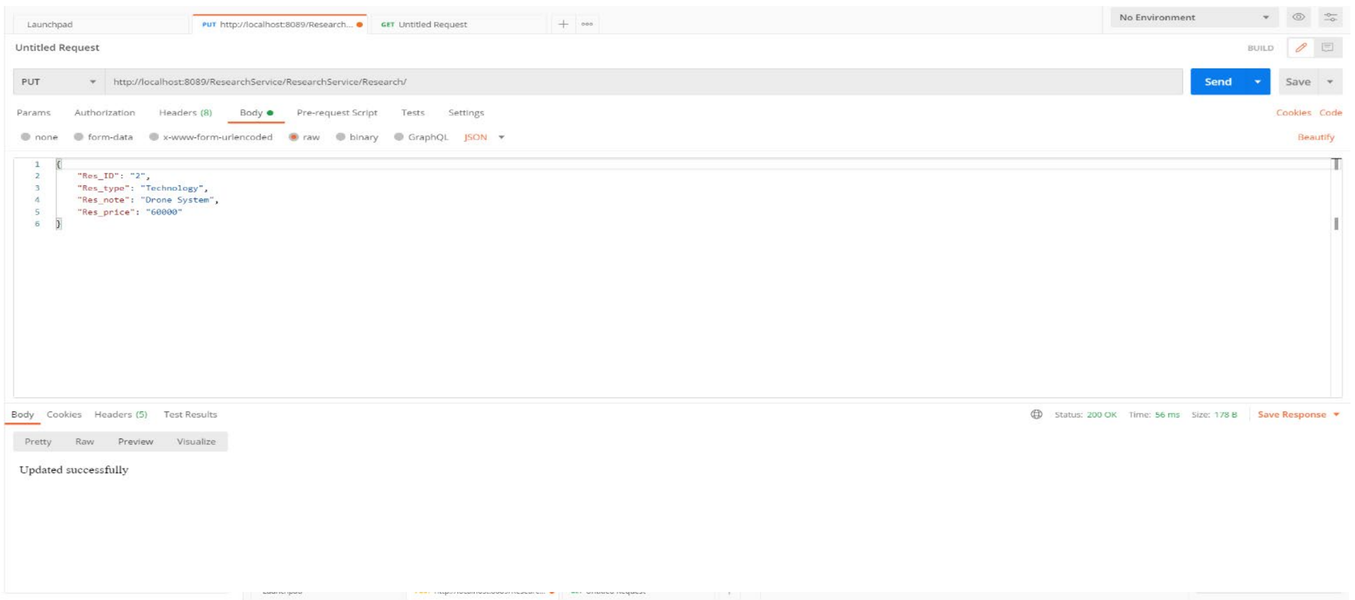


```
DELETE http://localhost:8089/ResearchService/ResearchService/Research/
```

```
<ResearchData>
        <Res_ID>1</Res_ID>
</ResearchData>
```

Status: 200 OK Time: 90 ms Size: 178 B

```
Deleted successfully
```



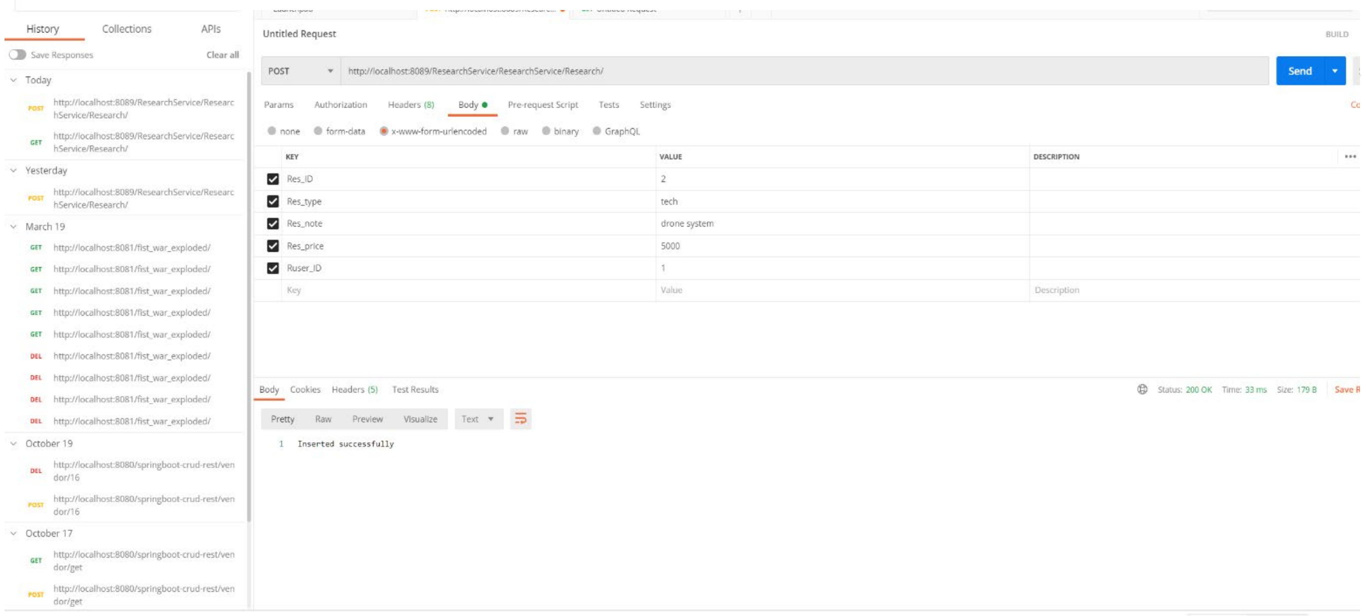
Untitled Request

PUT http://localhost:8089/ResearchService/ResearchService/Research/

```
{
    "Res_ID": "2",
    "Res_type": "Technology",
    "Res_note": "Drone System",
    "Res_price": "60000"
}
```

Status: 200 OK Time: 56 ms Size: 178 B

```
Updated successfully
```



Untitled Request

POST http://localhost:8089/ResearchService/ResearchService/Research/

Body: x-www-form-urlencoded

KEY	VALUE	DESCRIPTION
Res_ID	2	
Res_type	tech	
Res_note	drone system	
Res_price	5000	
Ruser_ID	1	

Status: 200 OK Time: 33 ms Size: 179 B

```
Inserted successfully
```

History

Today
- POST http://localhost:8089/ResearchService/ResearchService/Research/
- GET http://localhost:8089/ResearchService/ResearchService/Research/

Yesterday
- POST http://localhost:8089/ResearchService/ResearchService/Research/

March 19
- GET http://localhost:8081/fist_war_exploded/
- GET http://localhost:8081/fist_war_exploded/
- GET http://localhost:8081/fist_war_exploded/
- GET http://localhost:8081/fist_war_exploded/
- GET http://localhost:8081/fist_war_exploded/
- DEL http://localhost:8081/fist_war_exploded/
- DEL http://localhost:8081/fist_war_exploded/
- DEL http://localhost:8081/fist_war_exploded/
- DEL http://localhost:8081/fist_war_exploded/

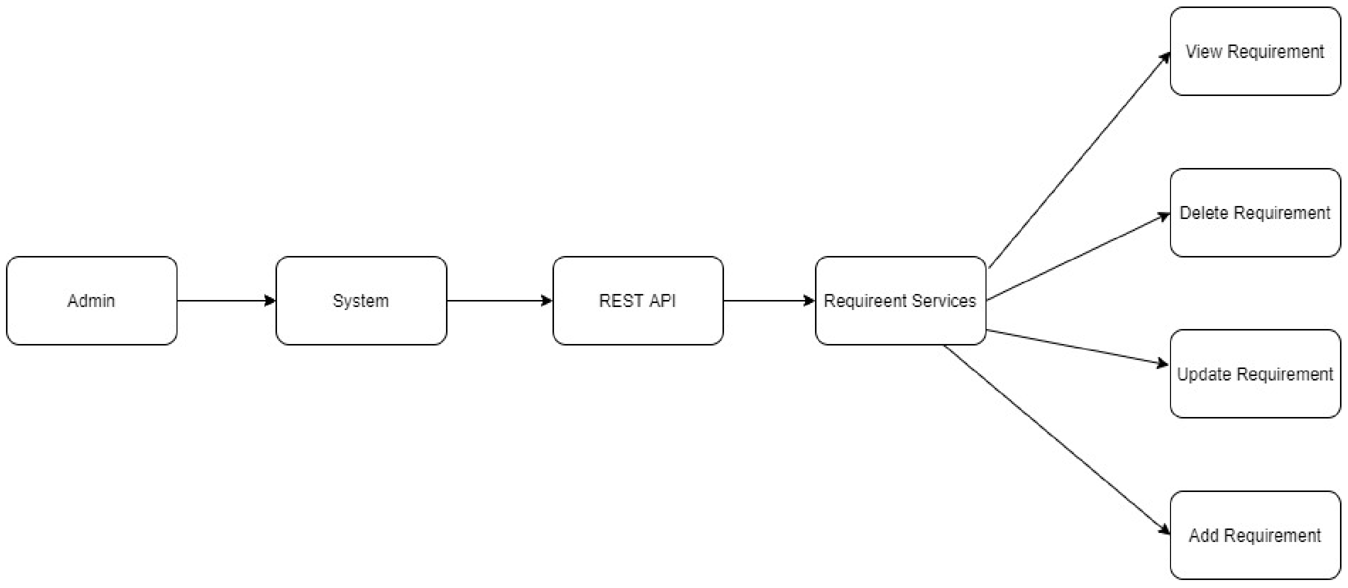
October 19
- DEL http://localhost:8080/springboot-crud-rest/vendor/16
- POST http://localhost:8080/springboot-crud-rest/vendor/16

October 17
- GET http://localhost:8080/springboot-crud-rest/vendor/get
- POST http://localhost:8080/springboot-crud-rest/vendor/get

RESTful Web service: Java - JAX-RS (Jersy) on Tomcat, DB: MySQL are the technical tools which I used to develop this system. I used Maven project management tool for build the project dependency. For the testing purposes I used "Postman" 3rd party testing tool. Testing methodology and the results are above.

Funding Company Service (Samarakkody S R S S U – IT19121116)

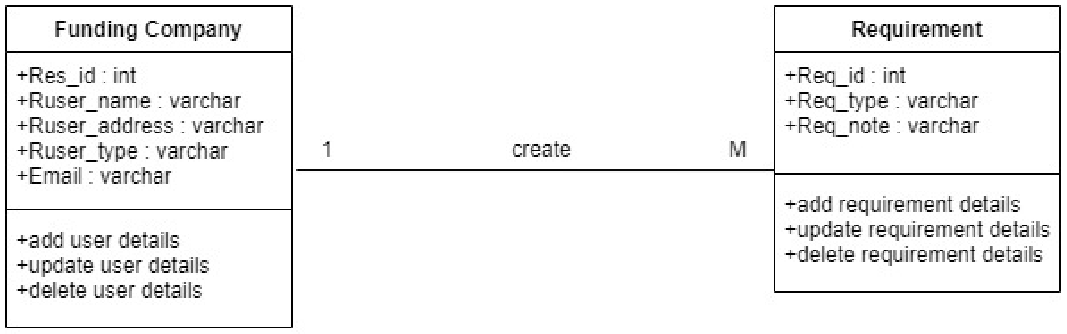
API for the Requirement Service



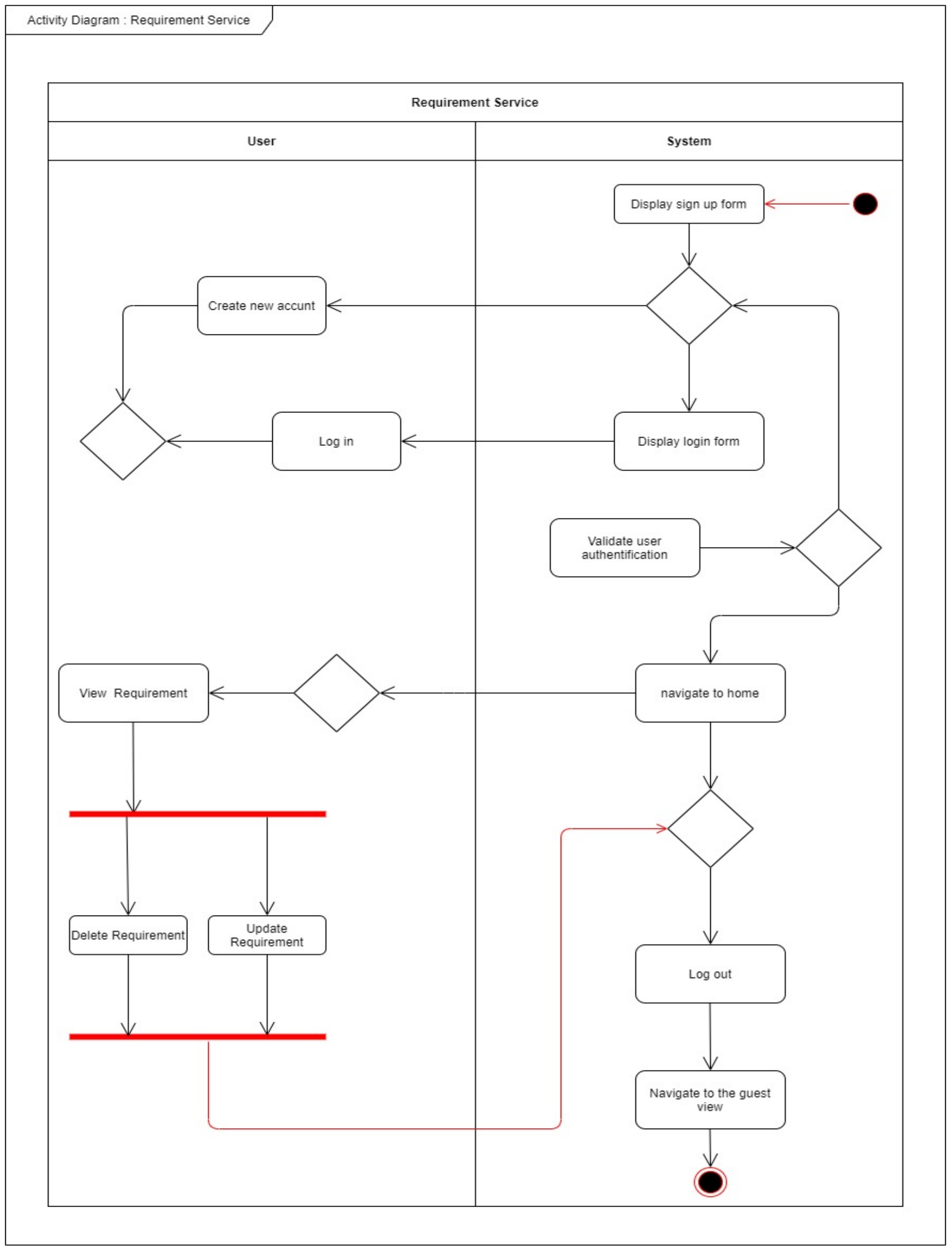
POST – Add Requirement	GET – View Requirement
URL - http://localhost:8090/FundingCompany/reqServices/req **Resource :** fundServices API **Request :** POST **addRequirement** **Media:** Form data -URL encoded **Data:** `resId , reqType , reqNote` **Response:** String status message `" Inserted successfully"` or `" Error while inserting"`	**URL –** http://localhost:8090/FundingCompany/reqServices/req **Resource :** fundServices API **Request :** GET **ViewRequirement**
PUT – Update Requirement	**DELETE – Delete Requirement**
URL - http://localhost:8090/FundingCompany/reqServices/req **Resource :** FundServices API **Request:** PUT **updateRequirement** **Media:** Application JSON **Data:** `reqId , resId , reqType , reqNote` **Response:** String status message `" [ID : ] Updated successfully"` or `" Error while updating "`	**URL -** http://localhost:8090/FundingCompany/reqServices/req **Resource :** FundServices API **Request:** DELETE **_deleteRequirement** **Media:** Application XML **Data:** `reqId` **Response:** String status message `" [ reqId : ] Deleted successfully"` or `"Error while deleting : "`

Internal Logic

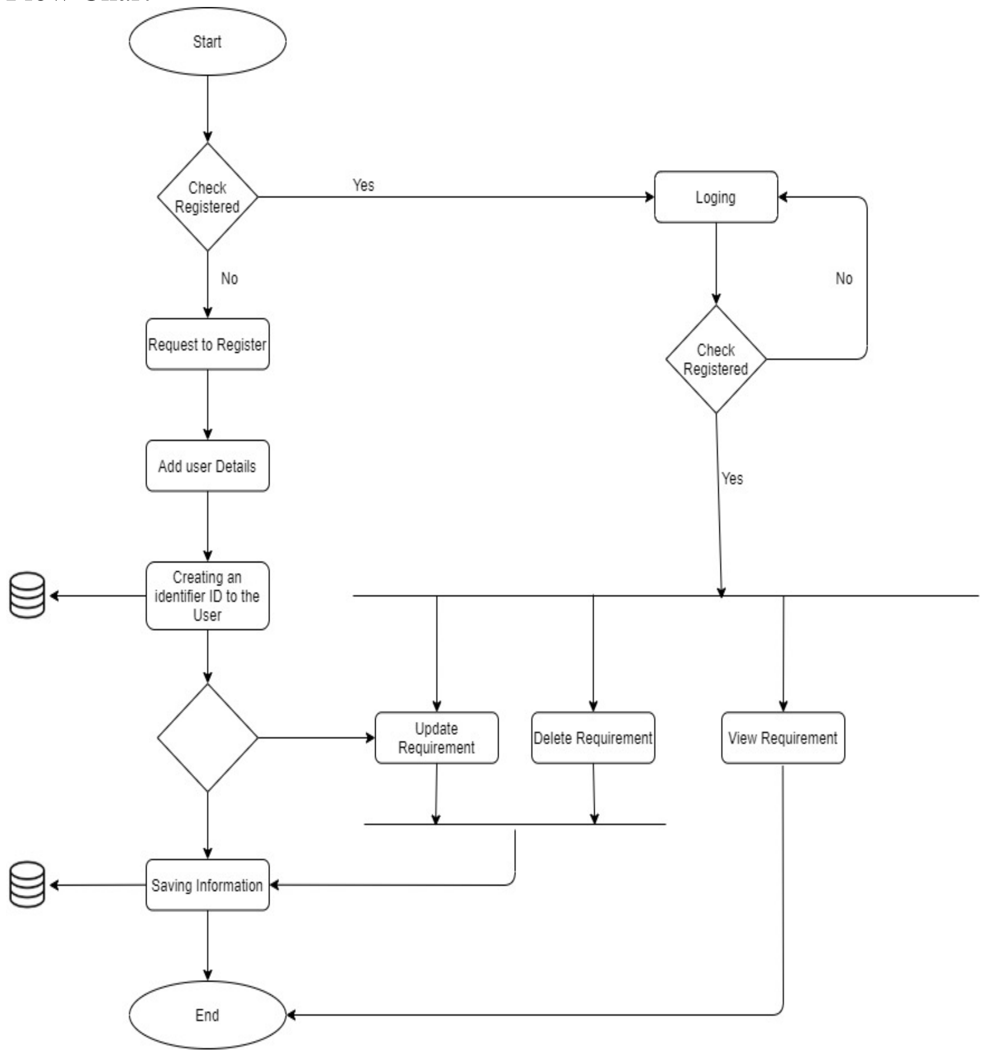
Class Diagram



Activity Diagram



Flow Chart



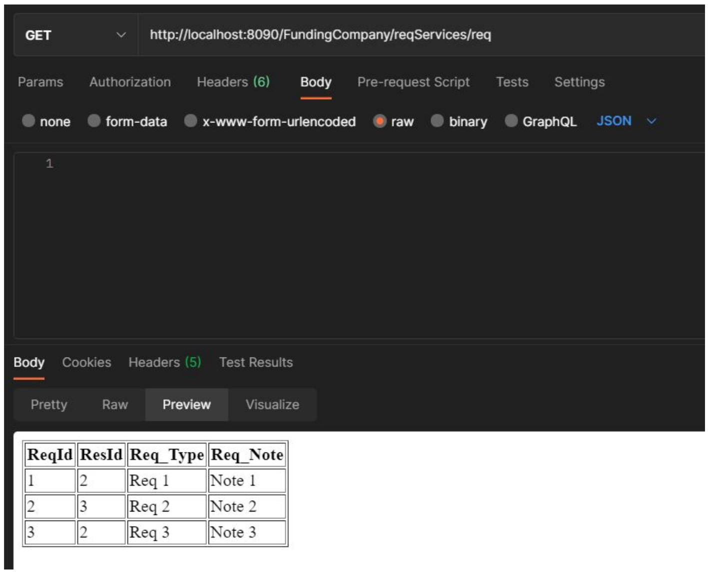
Service Development and Testing

- Dependency management tools
  IDE: Eclipse
  Database: MySQL
  Back End: Java
    - Maven

- Testing tool
  Postman

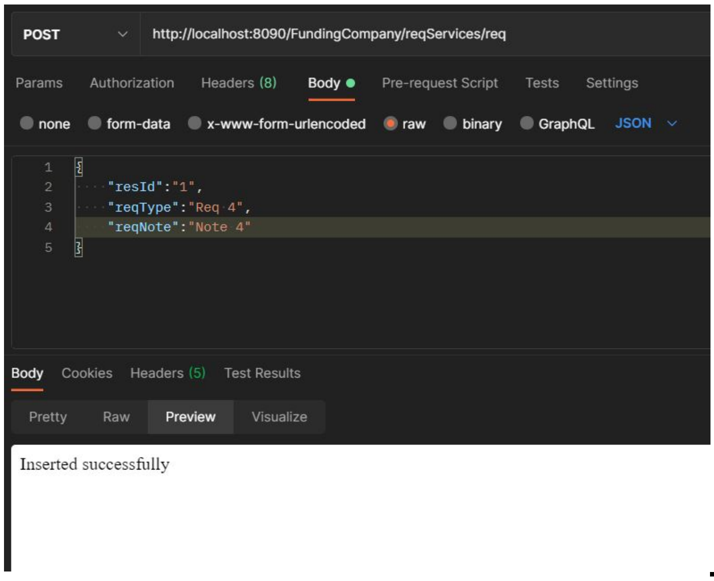
Testing methodology and results

Test ID	Description	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
01	Adding a new Requirement	`resId , reqType , reqNote`	`Inserted successfully`	`Inserted successfully`	Pass
02	View Requirement Details	URL for the API	Fund Details list	Fund Details list	Pass
03	Update Requirement Details	`reqId , resId , reqType , reqNote`	`[ID : ] Updated successfully`	`[ID : ] Updated successfully`	Pass
04	Delete Requirement	`reqId`	`[ reqId : ] Deleted successfully`	`[ reqId : ] Deleted successfully`	Pass


GET http://localhost:8090/FundingCompany/reqServices/req

Body: raw, JSON

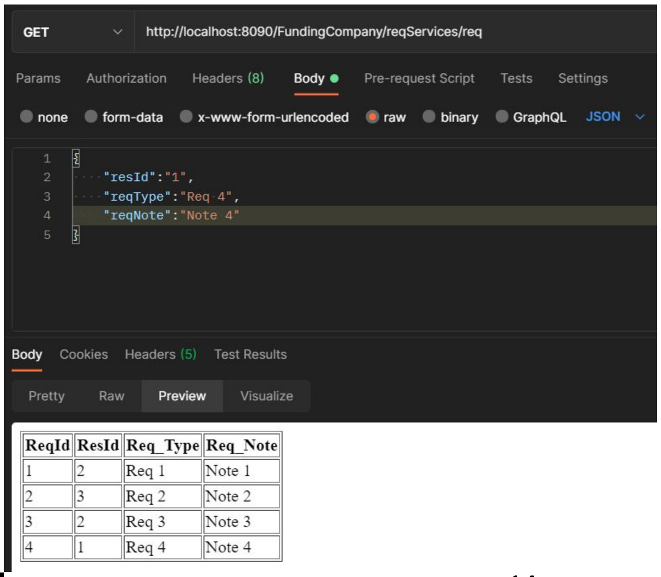
ReqId	ResId	Req_Type	Req_Note
1	2	Req 1	Note 1
2	3	Req 2	Note 2
3	2	Req 3	Note 3


POST http://localhost:8090/FundingCompany/reqServices/req

Body: raw, JSON

```
{
    "resId":"1",
    "reqType":"Req 4",
    "reqNote":"Note 4"
}
```

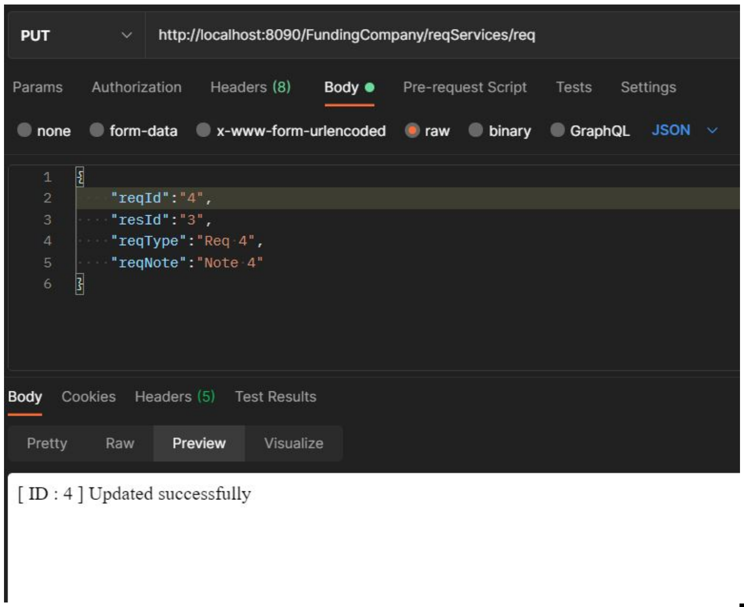
Inserted successfully


GET http://localhost:8090/FundingCompany/reqServices/req

Body: raw, JSON

```
{
    "resId":"1",
    "reqType":"Req 4",
    "reqNote":"Note 4"
}
```

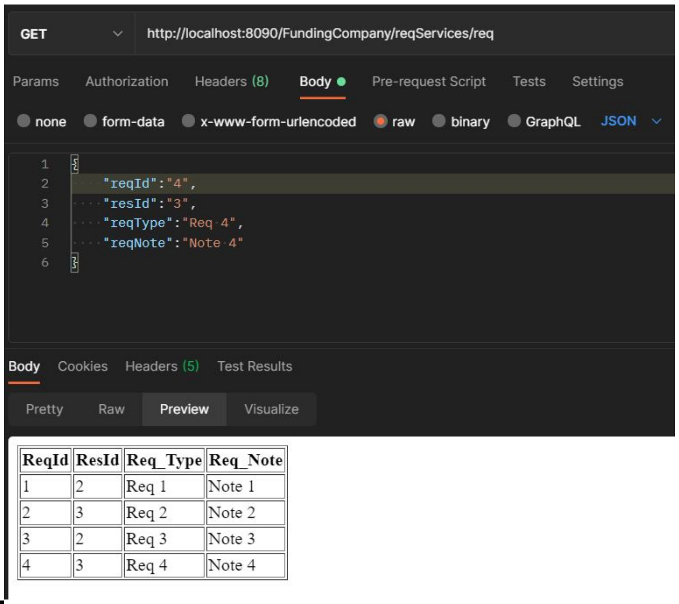
ReqId	ResId	Req_Type	Req_Note
1	2	Req 1	Note 1
2	3	Req 2	Note 2
3	2	Req 3	Note 3
4	1	Req 4	Note 4



http://localhost:8090/FundingCompany/reqServices/req

```
{
    "reqId":"4",
    "resId":"3",
    "reqType":"Req 4",
    "reqNote":"Note 4"
}
```

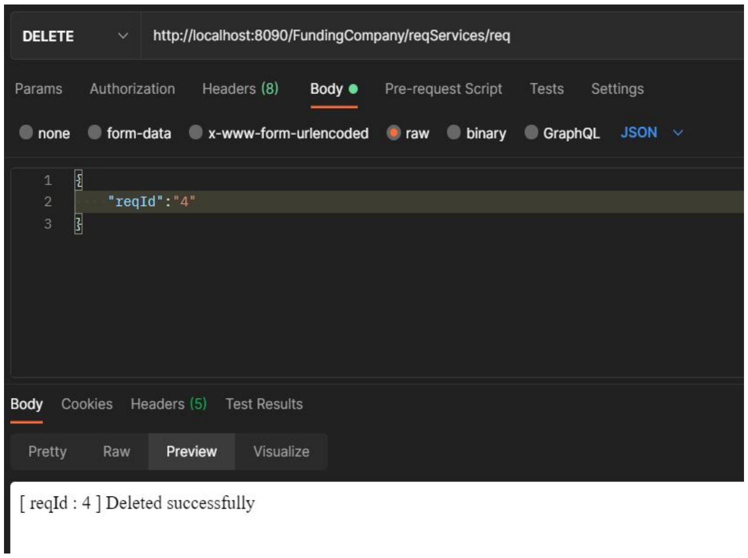
[ID : 4] Updated successfully



http://localhost:8090/FundingCompany/reqServices/req

```
{
    "reqId":"4",
    "resId":"3",
    "reqType":"Req 4",
    "reqNote":"Note 4"
}
```

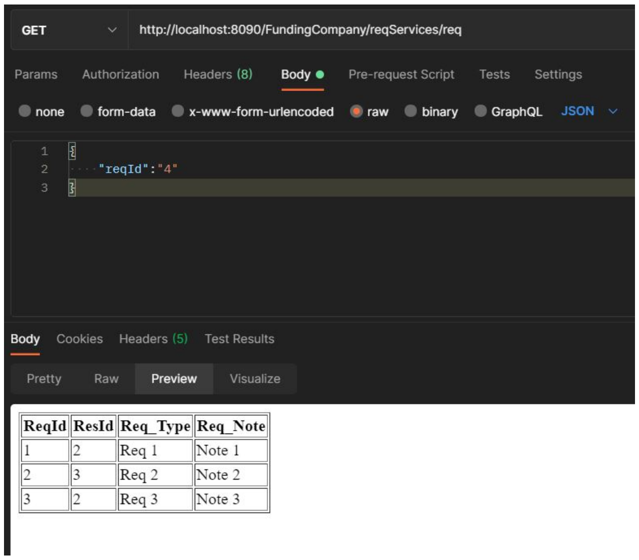
ReqId	ResId	Req_Type	Req_Note
1	2	Req 1	Note 1
2	3	Req 2	Note 2
3	2	Req 3	Note 3
4	3	Req 4	Note 4



http://localhost:8090/FundingCompany/reqServices/req

```
{
    "reqId":"4"
}
```

[reqId : 4] Deleted successfully



http://localhost:8090/FundingCompany/reqServices/req

```
{
    "reqId":"4"
}
```

ReqId	ResId	Req_Type	Req_Note
1	2	Req 1	Note 1
2	3	Req 2	Note 2
3	2	Req 3	Note 3

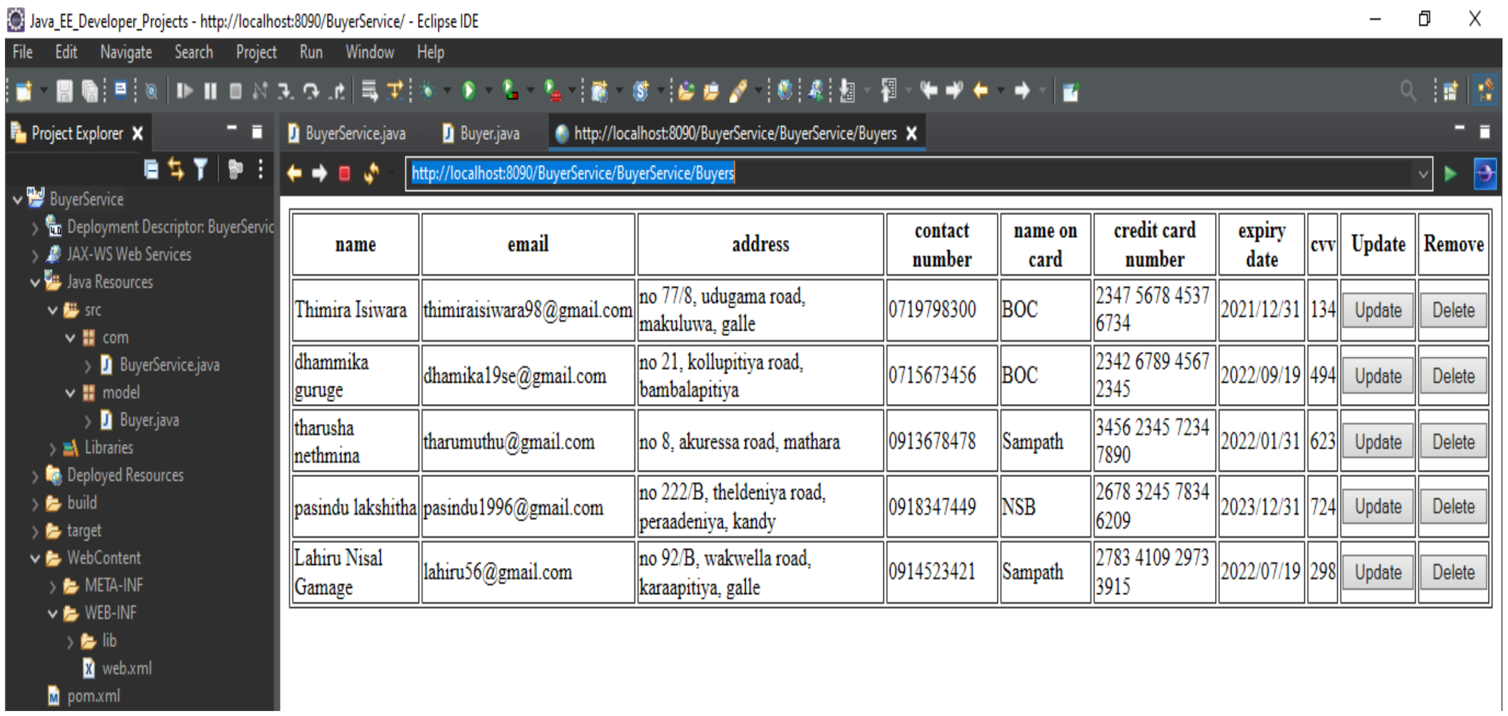
Buyer Service (T.V.Thimira Isiwara Vithanage. - IT19171302)

➢ Service Design

Eg: http://localhost:8090/BuyerService/BuyerService/Buyers

I activated the **RESTful** web service for the Order service using the Tomcat server. The database used was **MySQL**. **RESTful API** is an architectural style for application application interface (API) that uses HTTP requests to access and use data. That data can be used for **GET, PUT, POST** and **DELETE** data types, which means reading, updating, inserting and deleting resource related operations. I used **MAVEN** to create this web service.

This is the IP address of Tomcat server in eclipse IDE. **"localhost:8090"** is the machine name of the host server. **BuyerService** is my project name and URL-pattern given in the web.xml file. **Buyers** is the path to get the database connection to the project. If the connection is correct, the database will appear on the server window. If not, the error message "Error while connecting to the database for reading" will appear on the window. It performs all CRUD functions such as Read / Insert / Update / delete and is at the below of my Buyers database.



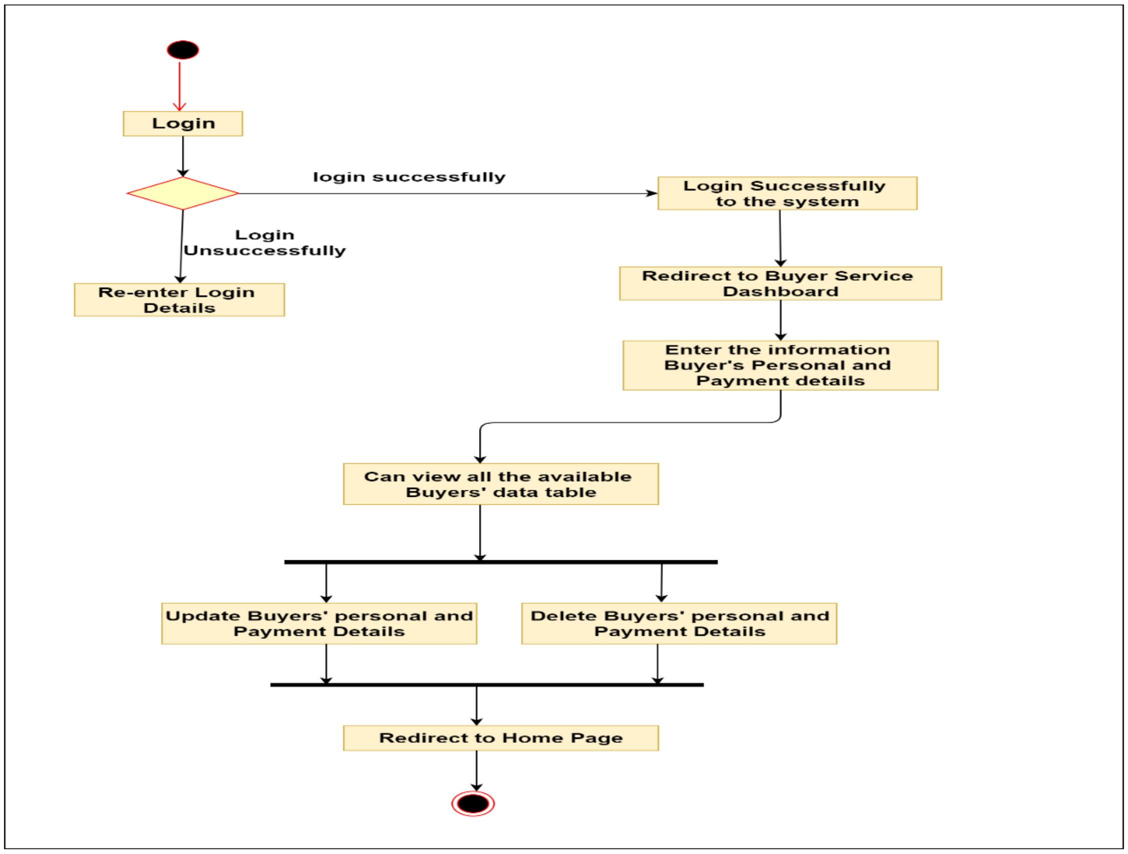
Java_EE_Developer_Projects - http://localhost:8090/BuyerService/ - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
BuyerService.java
Buyer.java
http://localhost:8090/BuyerService/BuyerService/Buyers
BuyerService
Deployment Descriptor: BuyerServic
JAX-WS Web Services
Java Resources
src
com
BuyerService.java
model
Buyer.java
Libraries
Deployed Resources
build
target
WebContent
META-INF
WEB-INF
lib
web.xml
pom.xml

name	email	address	contact number	name on card	credit card number	expiry date	cvv	Update	Remove
Thimira Isiwara	thimiraisiwara98@gmail.com	no 77/8, udugama road, makuluwa, galle	0719798300	BOC	2347 5678 4537 6734	2021/12/31	134	Update	Delete
dhammika guruge	dhamika19se@gmail.com	no 21, kollupitiya road, bambalapitiya	0715673456	BOC	2342 6789 4567 2345	2022/09/19	494	Update	Delete
tharusha nethmina	tharumuthu@gmail.com	no 8, akuressa road, mathara	0913678478	Sampath	3456 2345 7234 7890	2022/01/31	623	Update	Delete
pasindu lakshitha	pasindu1996@gmail.com	no 222/B, theldeniya road, peraadeniya, kandy	0918347449	NSB	2678 3245 7834 6209	2023/12/31	724	Update	Delete
Lahiru Nisal Gamage	lahiru56@gmail.com	no 92/B, wakwella road, karaapitiya, galle	0914523421	Sampath	2783 4109 2973 3915	2022/07/19	298	Update	Delete

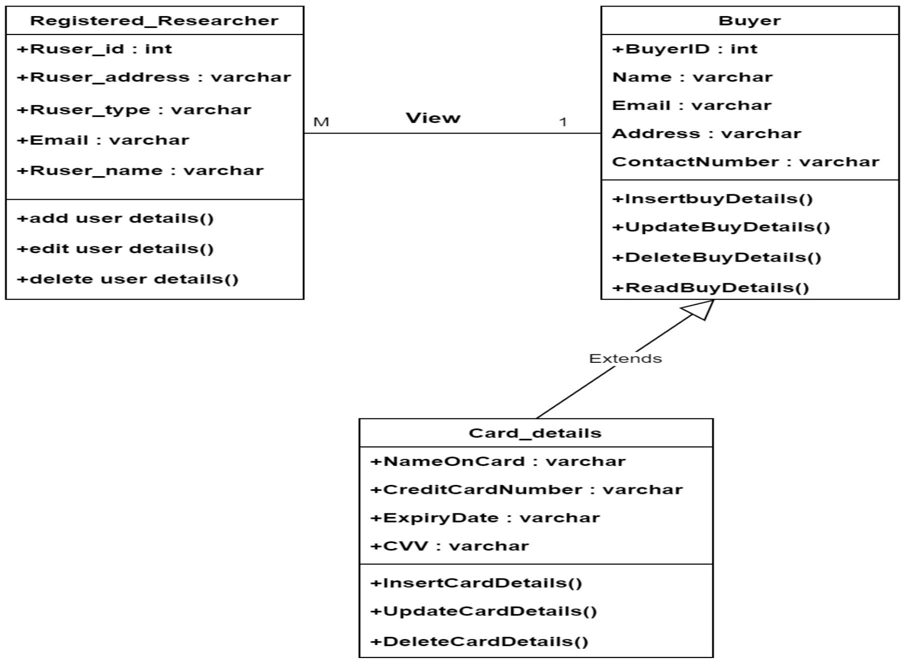
➢ Internal Logic

Micro-services architecture will use libraries, but the basic method of integrating their own software component is to break it down into services. We refer to the library as components that are connected to a program and use active calls in memory, while the service (buyer service) is the out-of-process component that communicates with a mechanism such as a web service request. Usually the best approach is to use something called an API portal.

Activity Diagram for Buyer Service

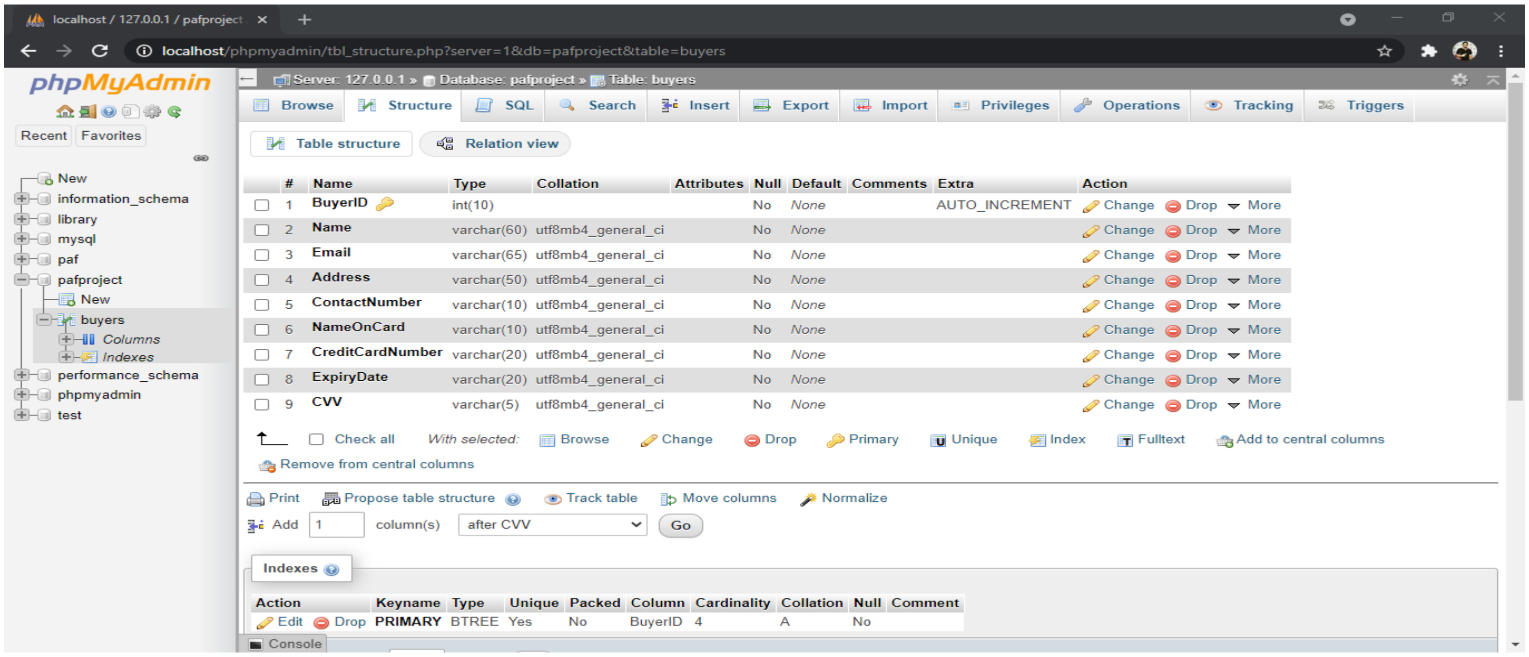

Login
login successfully
Login Successfully to the system
Login Unsuccessfully
Re-enter Login Details
Redirect to Buyer Service Dashboard
Enter the information Buyer's Personal and Payment details
Can view all the available Buyers' data table
Update Buyers' personal and Payment Details
Delete Buyers' personal and Payment Details
Redirect to Home Page

Class Diagram for Buyer Service



➢ DB Structure

This is my DB design for the service. My DB name is **pafproject** and **buyers** is my table name. **BuyerID** is the primary key.



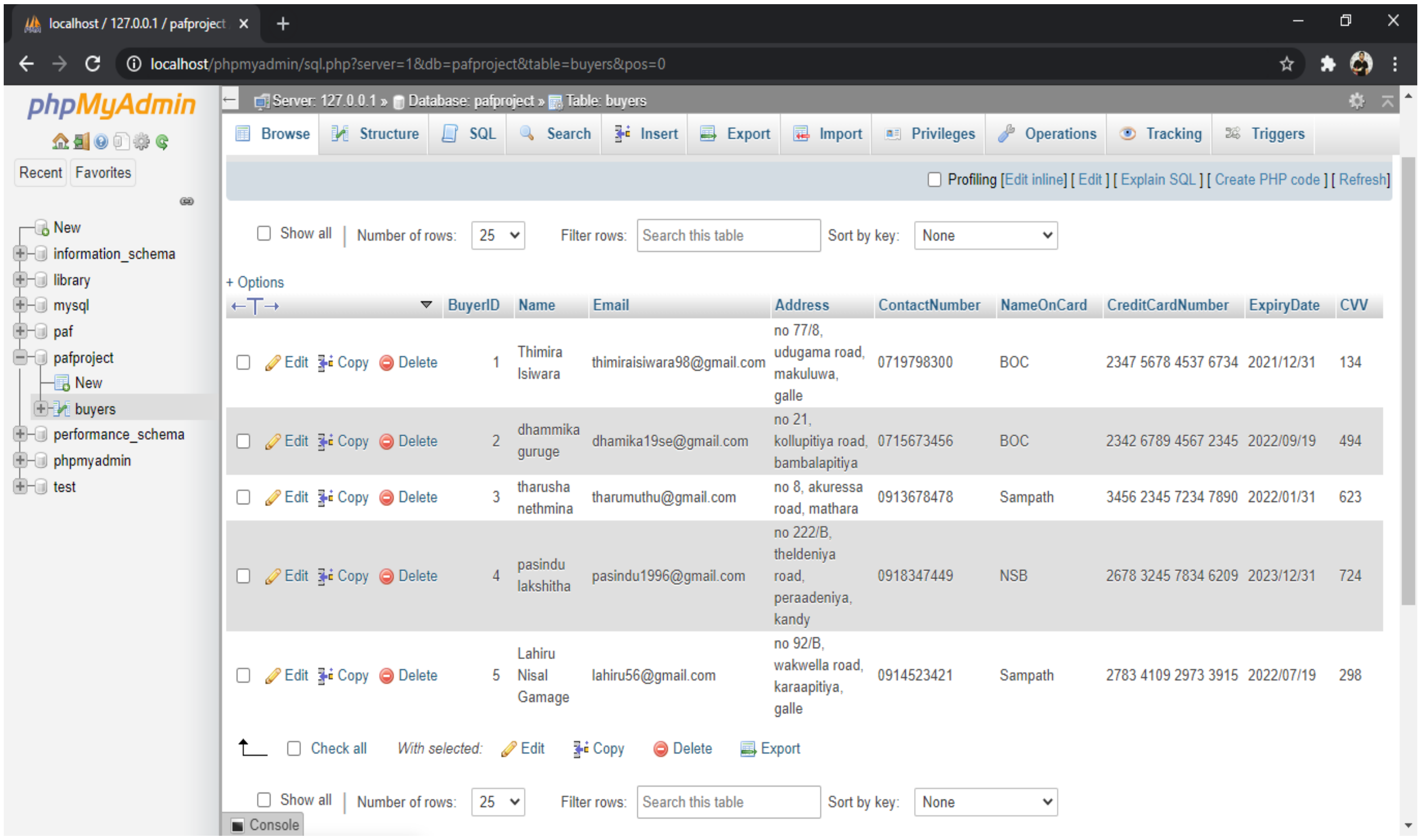
localhost/phpmyadmin/tbl_structure.php?server=1&db=pafproject&table=buyers

Server: 127.0.0.1 » Database: pafproject » Table: buyers

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	BuyerID	int(10)			No	None		AUTO_INCREMENT	Change Drop More
2	Name	varchar(60)	utf8mb4_general_ci		No	None			Change Drop More
3	Email	varchar(65)	utf8mb4_general_ci		No	None			Change Drop More
4	Address	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
5	ContactNumber	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
6	NameOnCard	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
7	CreditCardNumber	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
8	ExpiryDate	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
9	CVV	varchar(5)	utf8mb4_general_ci		No	None			Change Drop More

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	BuyerID	4	A	No	



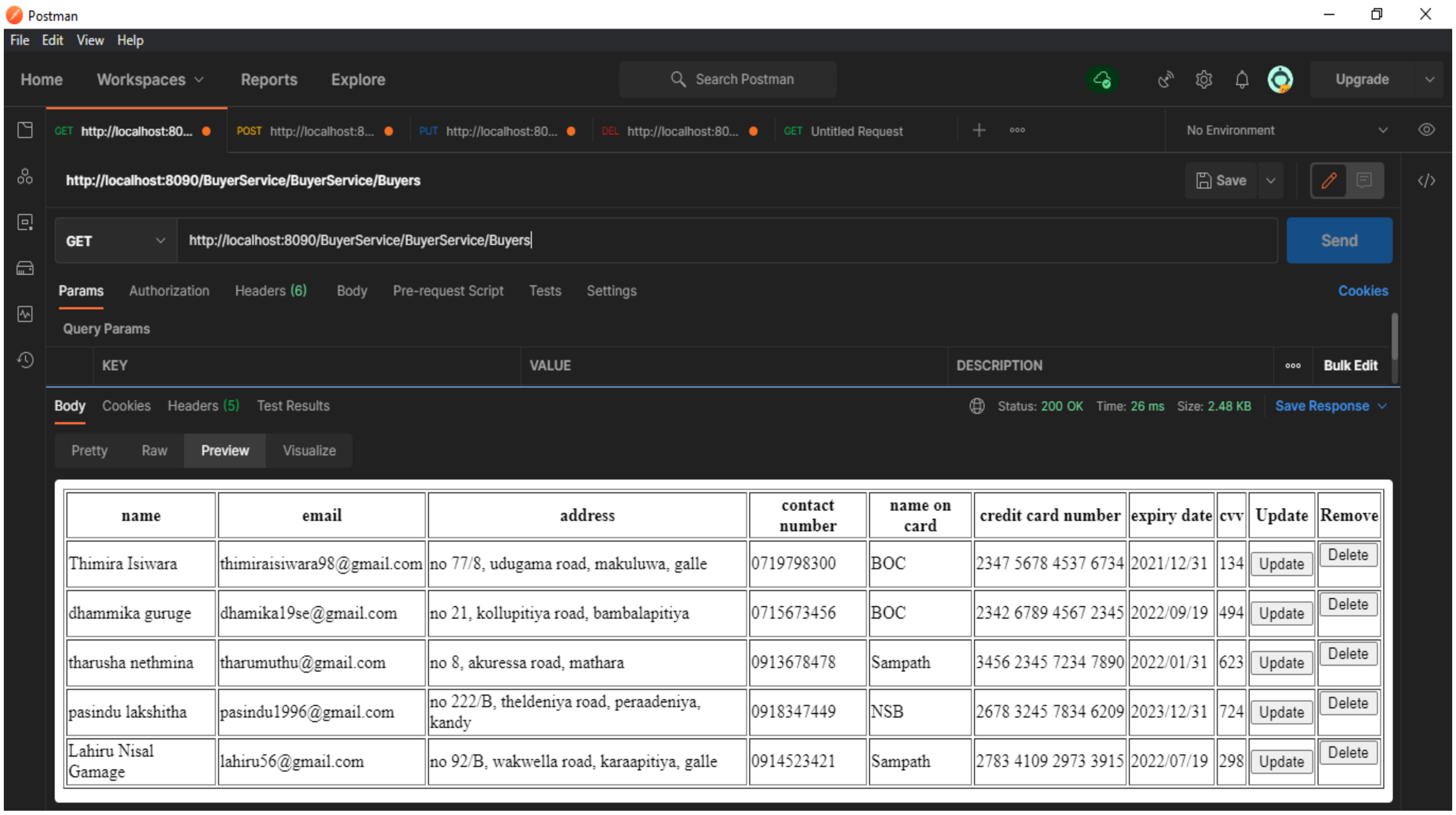
localhost/phpmyadmin/sql.php?server=1&db=pafproject&table=buyers&pos=0

Server: 127.0.0.1 » Database: pafproject » Table: buyers

	BuyerID	Name	Email	Address	ContactNumber	NameOnCard	CreditCardNumber	ExpiryDate	CVV
Edit Copy Delete	1	Thimira Isiwara	thimiraisiwara98@gmail.com	no 77/8, udugama road, makuluwa, galle	0719798300	BOC	2347 5678 4537 6734	2021/12/31	134
Edit Copy Delete	2	dhammika guruge	dhamika19se@gmail.com	no 21, kollupitiya road, bambalapitiya	0715673456	BOC	2342 6789 4567 2345	2022/09/19	494
Edit Copy Delete	3	tharusha nethmina	tharumuthu@gmail.com	no 8, akuressa road, mathara	0913678478	Sampath	3456 2345 7234 7890	2022/01/31	623
Edit Copy Delete	4	pasindu lakshitha	pasindu1996@gmail.com	no 222/B, theldeniya road, peraadeniya, kandy	0918347449	NSB	2678 3245 7834 6209	2023/12/31	724
Edit Copy Delete	5	Lahiru Nisal Gamage	lahiru56@gmail.com	no 92/B, wakwella road, karaapitiya, galle	0914523421	Sampath	2783 4109 2973 3915	2022/07/19	298

➢ Service development and testing

Using tool called postman we can check whether the insert, update, delete and read functions are working or not. After confirming the connection was correct, I copied the address to the **"postman".** After pasting the URL from the java EE IDE, select the **GET** method to connect to the database. We can then read the rest of the database by sending that URL.


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Upgrade
GET http://localhost:80...
POST http://localhost:8...
PUT http://localhost:80...
DEL http://localhost:80...
GET Untitled Request
No Environment
Save
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Body Cookies Headers (5) Test Results
Save Response
Pretty Raw Preview Visualize

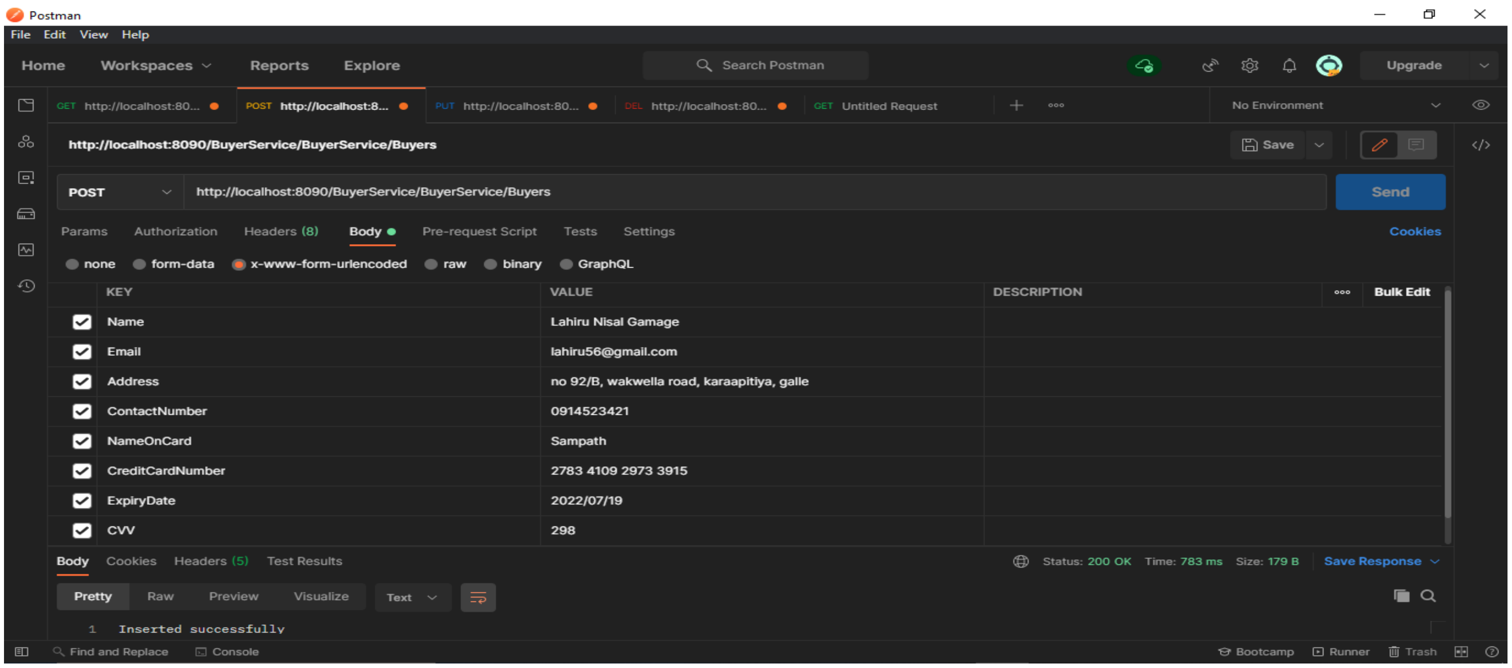
http://localhost:8090/BuyerService/BuyerService/Buyers

GET http://localhost:8090/BuyerService/BuyerService/Buyers

Status: 200 OK Time: 26 ms Size: 2.48 KB

name	email	address	contact number	name on card	credit card number	expiry date	cvv	Update	Remove
Thimira Isiwara	thimiraisiwara98@gmail.com	no 77/8, udugama road, makuluwa, galle	0719798300	BOC	2347 5678 4537 6734	2021/12/31	134	Update	Delete
dhammika guruge	dhamika19se@gmail.com	no 21, kollupitiya road, bambalapitiya	0715673456	BOC	2342 6789 4567 2345	2022/09/19	494	Update	Delete
tharusha nethmina	tharumuthu@gmail.com	no 8, akuressa road, mathara	0913678478	Sampath	3456 2345 7234 7890	2022/01/31	623	Update	Delete
pasindu lakshitha	pasindu1996@gmail.com	no 222/B, theldeniya road, peraadeniya, kandy	0918347449	NSB	2678 3245 7834 6209	2023/12/31	724	Update	Delete
Lahiru Nisal Gamage	lahiru56@gmail.com	no 92/B, wakwella road, karaapitiya, galle	0914523421	Sampath	2783 4109 2973 3915	2022/07/19	298	Update	Delete

Second, open a new tab and select the link **POST**. In the insertion method, the request data type must be **X-WWW-FORM-URLENCODED**. Select the body and then add all the attributes except the primary key (**BuyerID**) to the space provided above. Now enter the data we want to enter into the database. If the data is entered correctly, a successful message "**Inserted Successfully**" will appear.



POST http://localhost:8090/BuyerService/BuyerService/Buyers

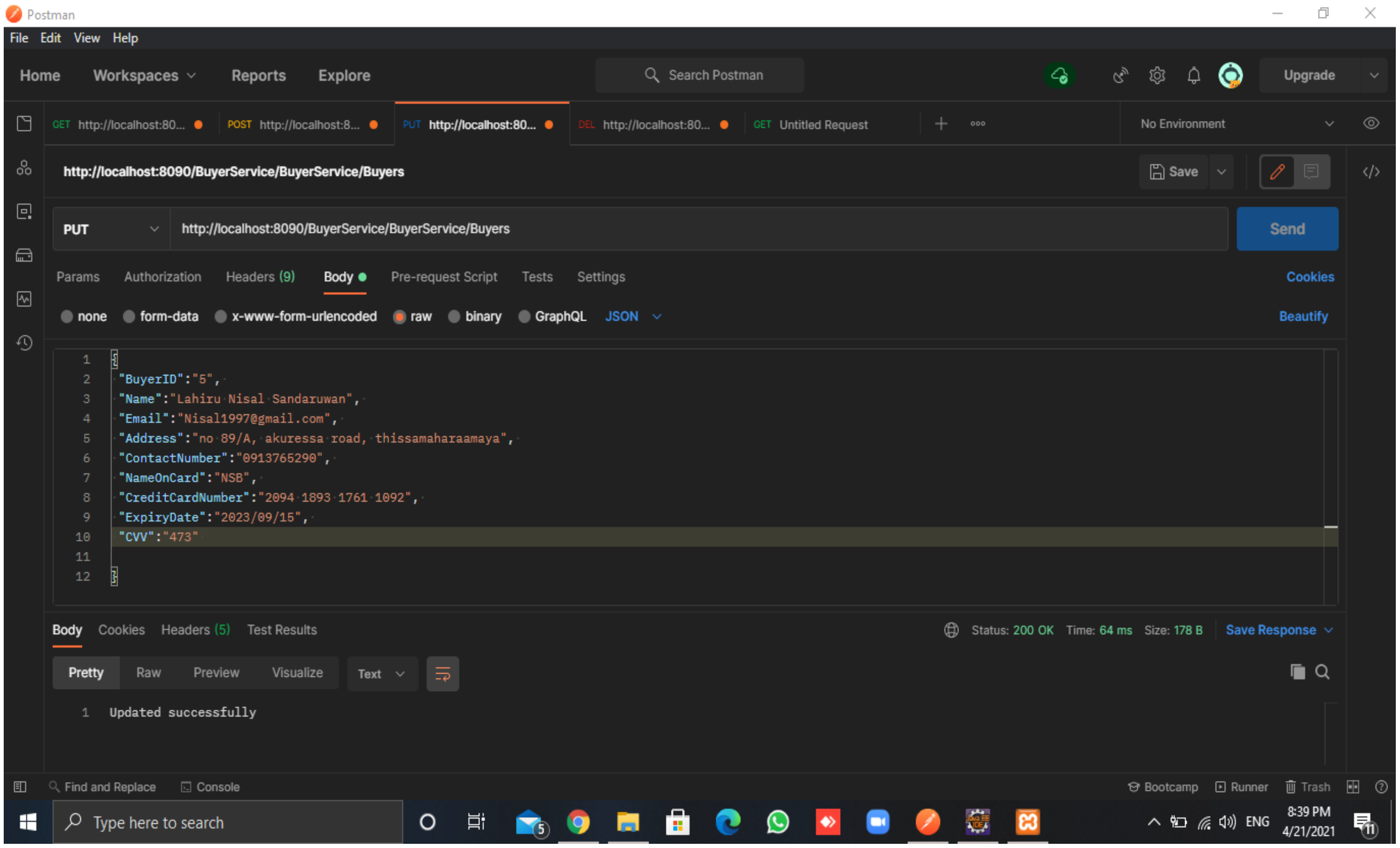
Body: x-www-form-urlencoded

KEY	VALUE	DESCRIPTION
Name	Lahiru Nisal Gamage	
Email	lahiru56@gmail.com	
Address	no 92/B, wakwella road, karaapitiya, galle	
ContactNumber	0914523421	
NameOnCard	Sampath	
CreditCardNumber	2783 4109 2973 3915	
ExpiryDate	2022/07/19	
CVV	298	

Status: 200 OK Time: 783 ms Size: 179 B

```
Inserted successfully
```

Third, open a new tab and select the **PUT** method and then click the link to update the data. Then we need to send the request field names and update values as **APPLICATION / JSON** in the body part and then add the primary key (**BuyerID**) we need to update and enter new information. If you entered the data correctly, an "**Update Successfully**" update message will appear, and an incorrect message will appear if incorrect data is entered.



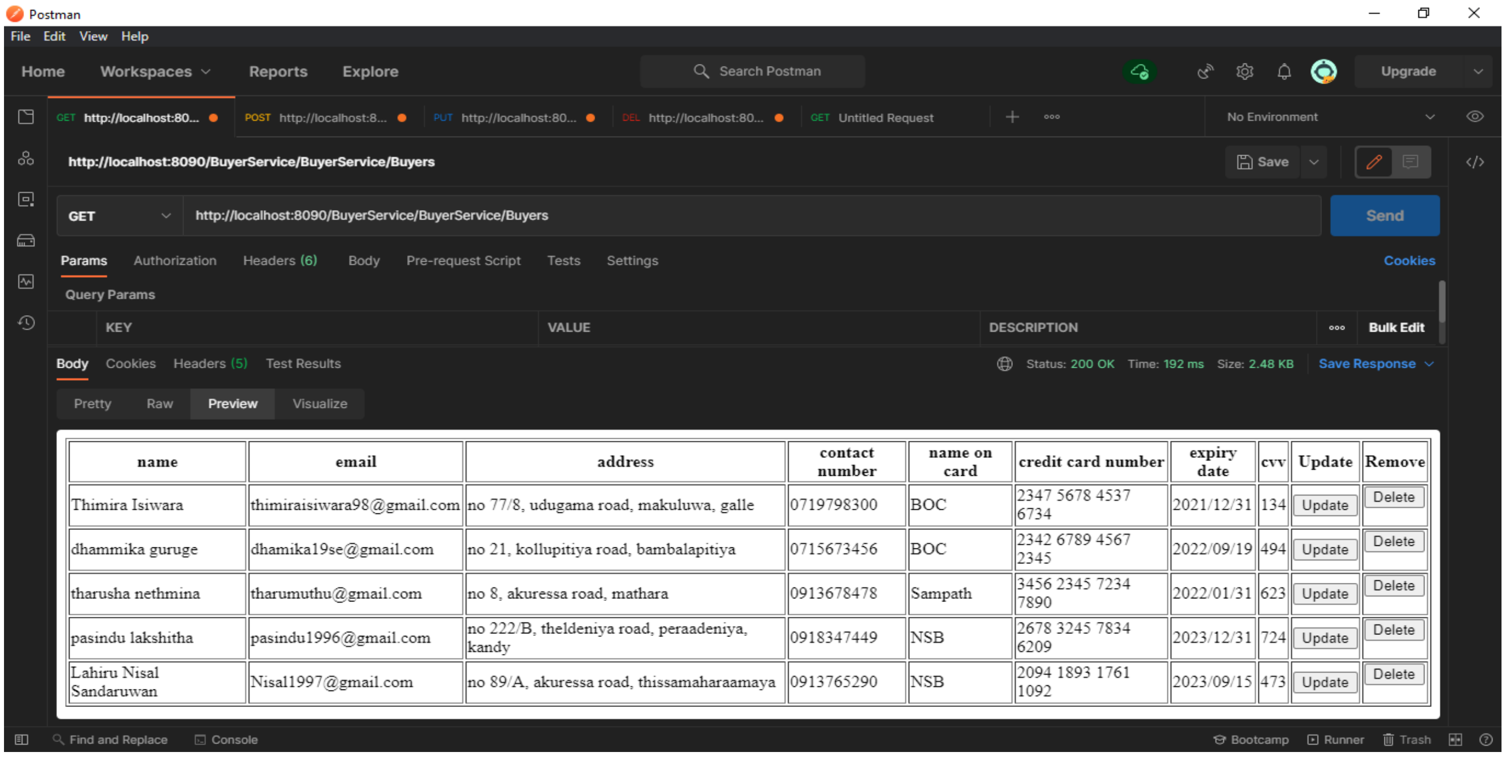
PUT http://localhost:8090/BuyerService/BuyerService/Buyers

```
{
"BuyerID":"5",
"Name":"Lahiru Nisal Sandaruwan",
"Email":"Nisal1997@gmail.com",
"Address":"no 89/A, akuressa road, thissamaharaamaya",
"ContactNumber":"0913765290",
"NameOnCard":"NSB",
"CreditCardNumber":"2094 1893 1761 1092",
"ExpiryDate":"2023/09/15",
"CVV":"473"

}
```

Status: 200 OK Time: 64 ms Size: 178 B

```
Updated successfully
```

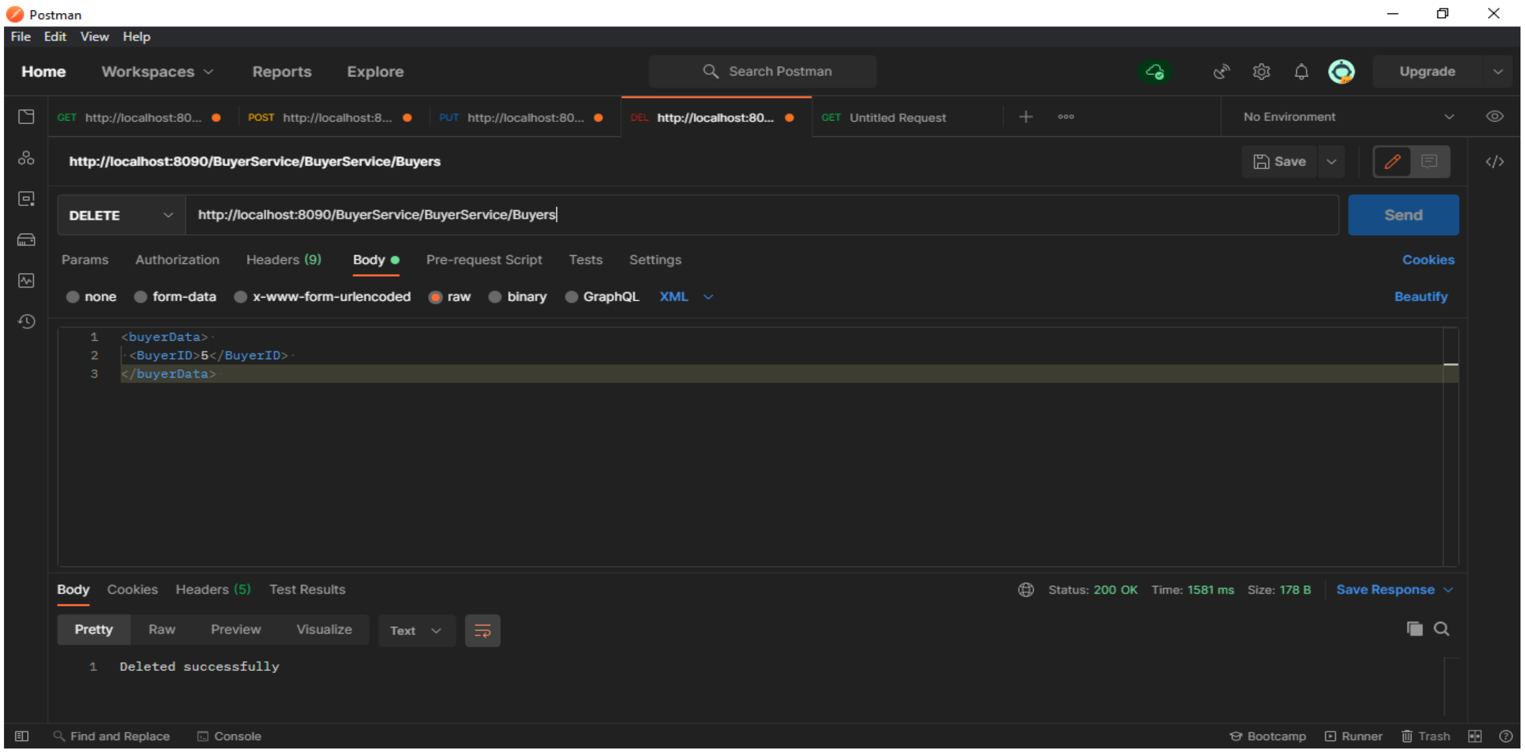


GET http://localhost:8090/BuyerService/BuyerService/Buyers

Status: 200 OK Time: 192 ms Size: 2.48 KB

name	email	address	contact number	name on card	credit card number	expiry date	cvv	Update	Remove
Thimira Isiwara	thimiraisiwara98@gmail.com	no 77/8, udugama road, makuluwa, galle	0719798300	BOC	2347 5678 4537 6734	2021/12/31	134	Update	Delete
dhammika guruge	dhamika19se@gmail.com	no 21, kollupitiya road, bambalapitiya	0715673456	BOC	2342 6789 4567 2345	2022/09/19	494	Update	Delete
tharusha nethmina	tharumuthu@gmail.com	no 8, akuressa road, mathara	0913678478	Sampath	3456 2345 7234 7890	2022/01/31	623	Update	Delete
pasindu lakshitha	pasindu1996@gmail.com	no 222/B, theldeniya road, peraadeniya, kandy	0918347449	NSB	2678 3245 7834 6209	2023/12/31	724	Update	Delete
Lahiru Nisal Sandaruwan	Nisal1997@gmail.com	no 89/A, akuressa road, thissamaharaamaya	0913765290	NSB	2094 1893 1761 1092	2023/09/15	473	Update	Delete

To delete, open a new tab and select the **DEL** method. Then we need to send the request field names and values for deletion to the body part as **APPLICATION / XML** and then enter the primary key to delete the row of the table containing the relevant primary key information. The data will then be erased and the message **"Delete Successfully"** will appear.



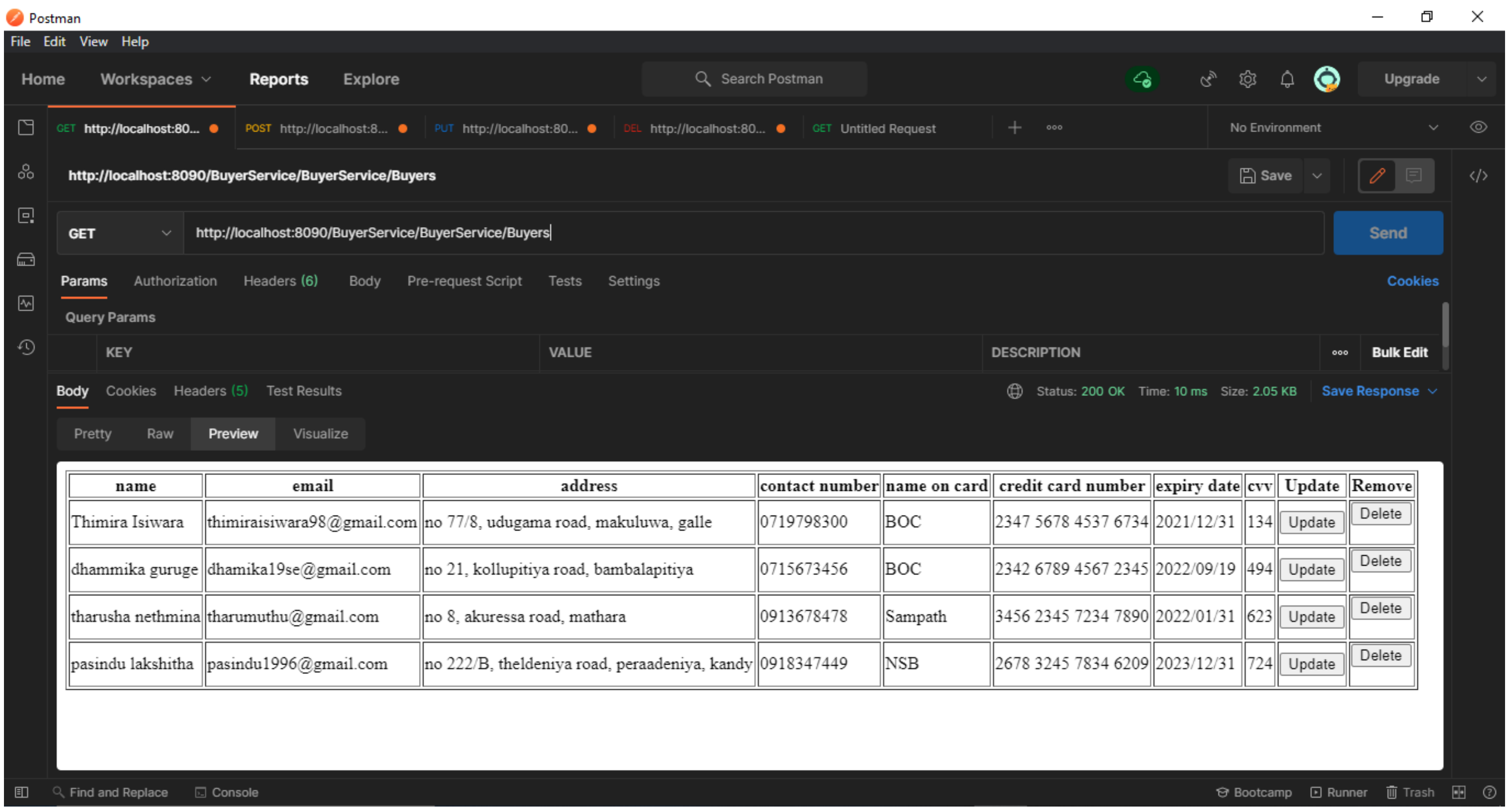
DELETE http://localhost:8090/BuyerService/BuyerService/Buyers

Body: raw, XML

```
<buyerData>
 <BuyerID>5</BuyerID>
</buyerData>
```

Status: 200 OK Time: 1581 ms Size: 178 B

```
Deleted successfully
```



GET http://localhost:8090/BuyerService/BuyerService/Buyers

Status: 200 OK Time: 10 ms Size: 2.05 KB

name	email	address	contact number	name on card	credit card number	expiry date	cvv	Update	Remove
Thimira Isiwara	thimiraisiwara98@gmail.com	no 77/8, udugama road, makuluwa, galle	0719798300	BOC	2347 5678 4537 6734	2021/12/31	134	Update	Delete
dhammika guruge	dhamika19se@gmail.com	no 21, kollupitiya road, bambalapitiya	0715673456	BOC	2342 6789 4567 2345	2022/09/19	494	Update	Delete
tharusha nethmina	tharumuthu@gmail.com	no 8, akuressa road, mathara	0913678478	Sampath	3456 2345 7234 7890	2022/01/31	623	Update	Delete
pasindu lakshitha	pasindu1996@gmail.com	no 222/B, theldeniya road, peraadeniya, kandy	0918347449	NSB	2678 3245 7834 6209	2023/12/31	724	Update	Delete

Technical tools which I used:-

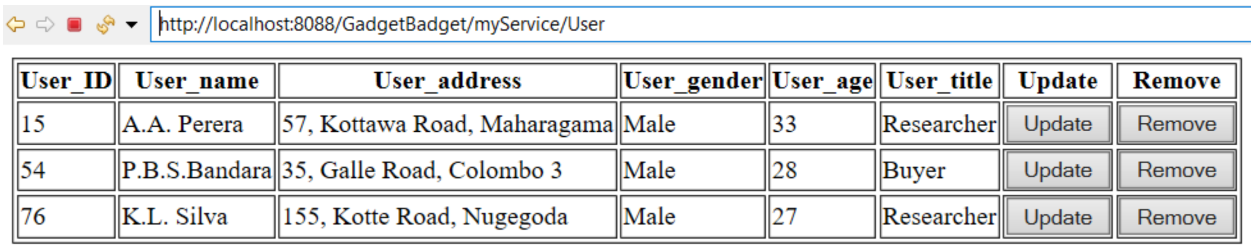
- **RESTful** web service : **Java – JAX-RS (Jersy)**
- Server – **Tomcat 9.0 Server**
- Database - **MySQL**
- Project Management Tool - **Maven**
- Testing tools – **Postman**

User management and Rating Service (De Silva W.A.D.S. – IT19037066)

Service Design 1 – User Registration Service

Application Programming Interface (API) of **User Registration Service** implemented according to the requirements of stakeholders. User Registration Service works consistently and accurately according to all CRUD Functions.

E.g. : http://localhost:8088/GadgetBadget/myService/User

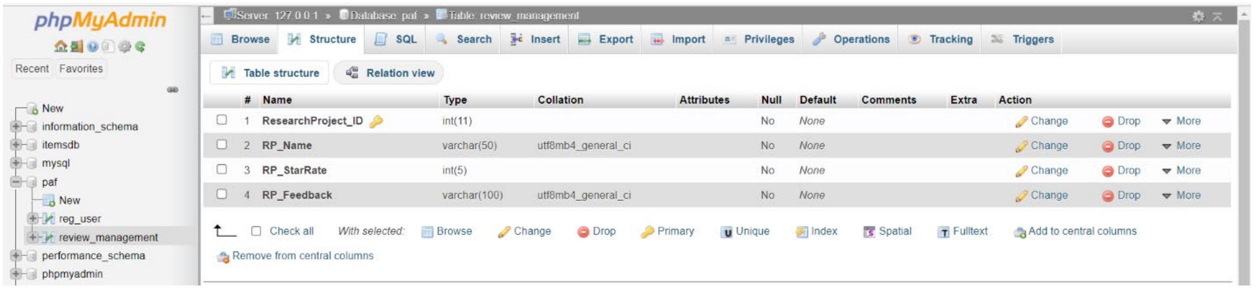


http://localhost:8088/GadgetBadget/myService/User

User_ID	User_name	User_address	User_gender	User_age	User_title	Update	Remove
15	A.A. Perera	57, Kottawa Road, Maharagama	Male	33	Researcher	Update	Remove
54	P.B.S.Bandara	35, Galle Road, Colombo 3	Male	28	Buyer	Update	Remove
76	K.L. Silva	155, Kotte Road, Nugegoda	Male	27	Researcher	Update	Remove

The API address that I run on my tomcat server in eclipse. Localhost:8088 is the address of the host server. 'GadgetBadget' is the name of the project. 'myService' is the URL pattern given in the web.xml file. 'User' is the path to get the database connection to the project. If the connection is ok, the database will appear on the server window. If not, the error message will be displayed on the window. All the CRUD functions are done by this and testing is below in my section.

DB Structure



Server: 127.0.0.1 » Database: paf » Table: review_management

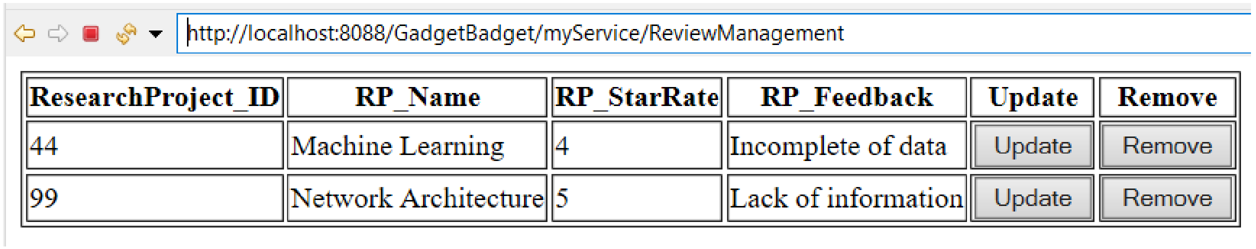
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ResearchProject_ID	int(11)			No	None			Change Drop More
2	RP_Name	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
3	RP_StarRate	int(5)			No	None			Change Drop More
4	RP_Feedback	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More

I used MYSQL to develop the database for my service. My DB name is 'paf' and the table name is 'reg_user'. The primary key is 'User_ID'. Other column names are 'User_name', 'User_address', 'User_gender', 'User_age', 'User_title'.

Service Design 2 – Review Management (Rating Service)

Application Programming Interface (API) of **Review Management (Rating Management) Service** implemented according to the requirements of stakeholders. Review Management (Rating Management) Service works consistently and accurately according to all CRUD Functions. This page mainly forced to rate research projects and give opinions as feedback.

E.g. : http://localhost:8088/GadgetBadget/myService/ReviewManagement

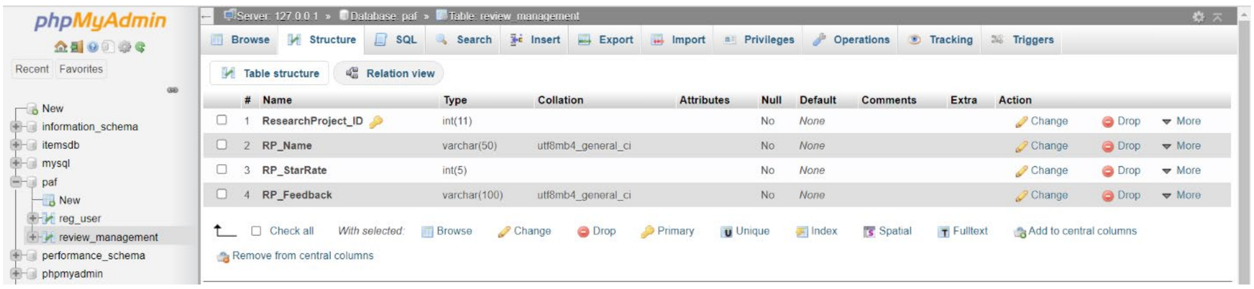


http://localhost:8088/GadgetBadget/myService/ReviewManagement

ResearchProject_ID	RP_Name	RP_StarRate	RP_Feedback	Update	Remove
44	Machine Learning	4	Incomplete of data	Update	Remove
99	Network Architecture	5	Lack of information	Update	Remove

The API address that I run on my tomcat server in eclipse. Localhost:8088 is the address of the host server. 'GadgetBadget' is the name of the project. 'myService' is the URL pattern given in the web.xml file. '`ReviewManagement`' is the path to get the database connection to the project. If the connection is ok, the database will appear on the server window. If not, the error message will be displayed on the window. All the CRUD functions are done by this and testing is below in my section.

DB Structure



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
1	ResearchProject_ID	int(11)			No	None			Change	Drop	More
2	RP_Name	varchar(50)	utf8mb4_general_ci		No	None			Change	Drop	More
3	RP_StarRate	int(5)			No	None			Change	Drop	More
4	RP_Feedback	varchar(100)	utf8mb4_general_ci		No	None			Change	Drop	More

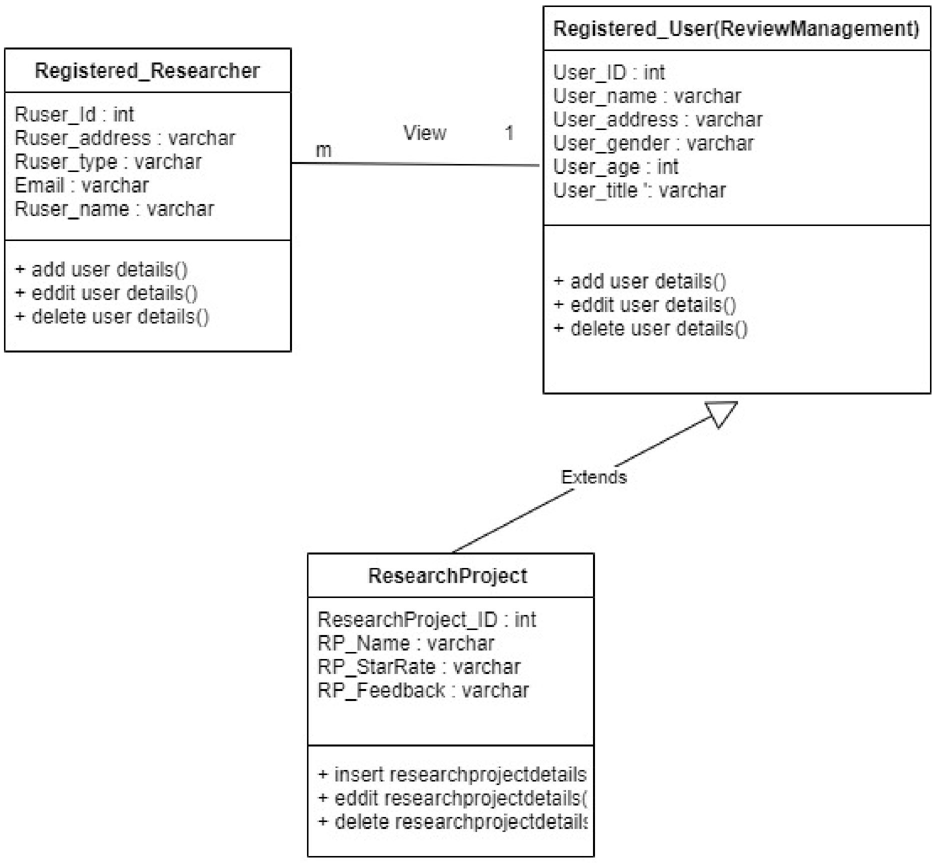
I used MYSQL to develop the database for my service. My DB name is 'paf' and the table name is 'review_management'. The primary key is 'ResearchProject_ID'. Other column names are 'RP_Name', 'RP_StarRate', 'RP_Feedback'.

Internal Logic

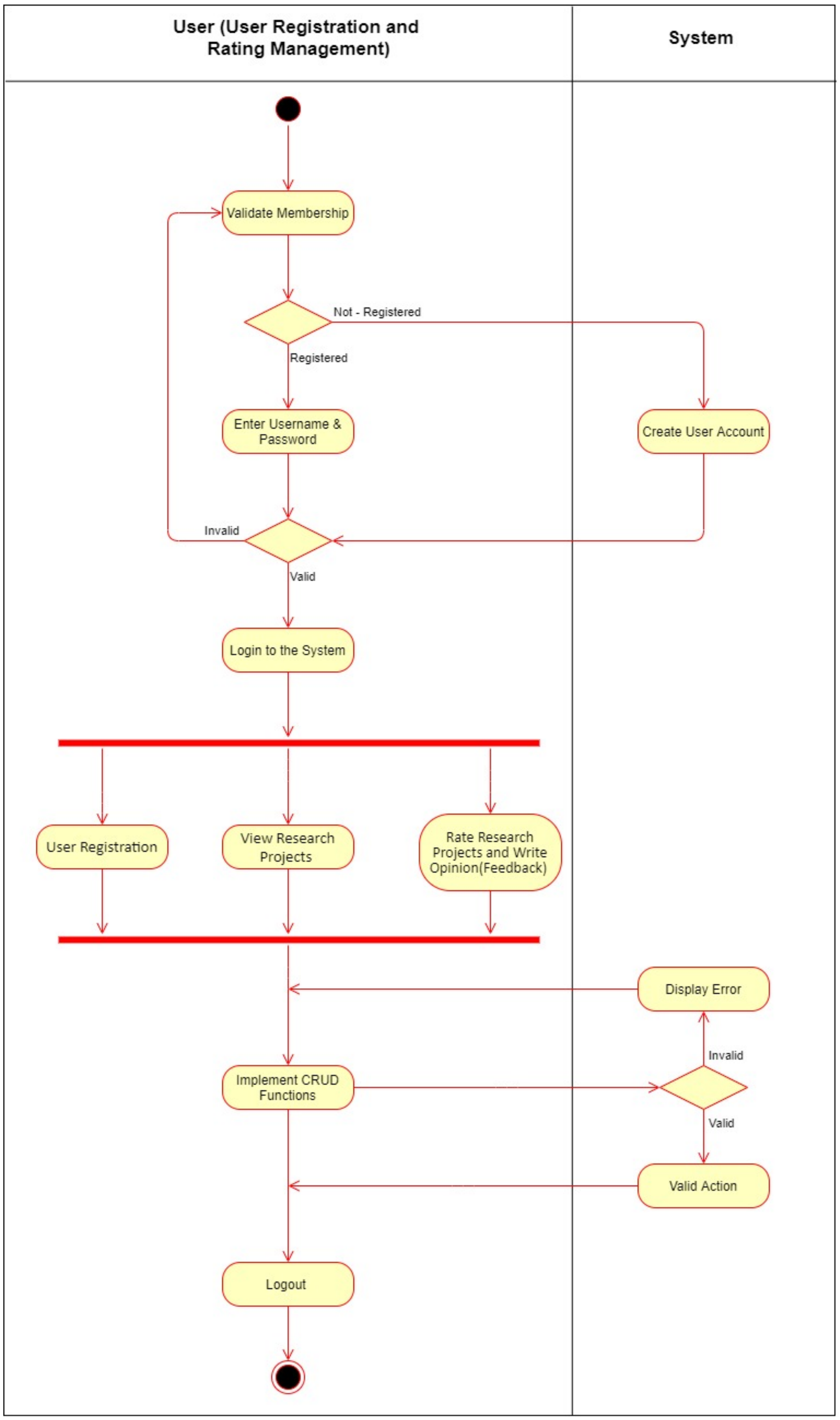
Here the user can register to the system at the beginning by entering the user details. Also, they can update and delete their profile. The user can appear the User service dashboard. They can update their user details or delete the details if necessary.

The Review Management can register to the system using valid information. Also, they can update and delete their profile. The user can appear the Review Management Services dashboard. The Review Management can view research projects and rate research projects give their opinion as comments.

Class Diagram



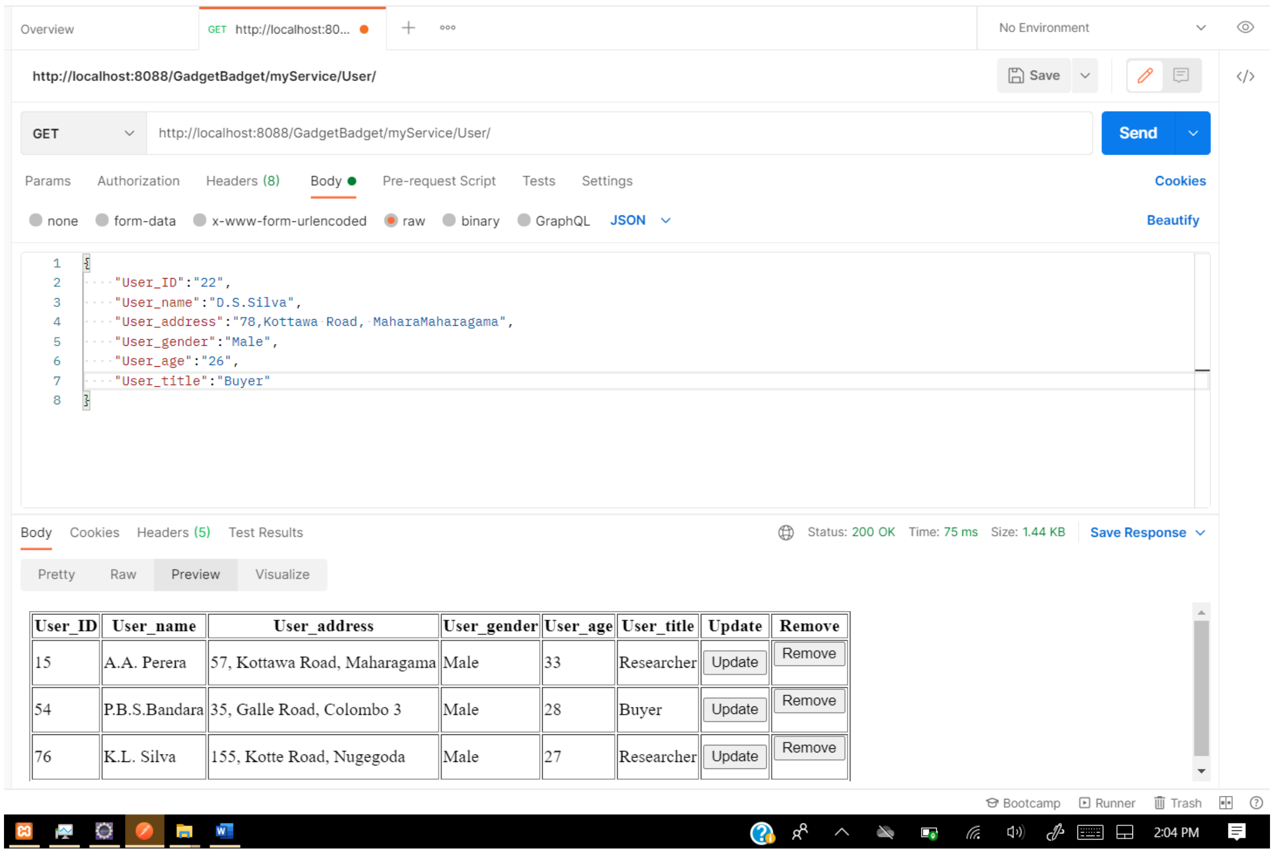
Activitiy Diagram



Service Development and Testing

Maven is a build automation tool used fundamentally for Java projects. The dependency management mechanism that I used is Maven. "Postman" is the testing tool that I used. Testing methodology and the results are below. (CRUD functions that I tested)

- GET - use to retrieve resources
- POST - accept the entity enclosed in the request as a new resource
- PUT - support the updating of REST resources
- DELETE – delete RESTful resources.



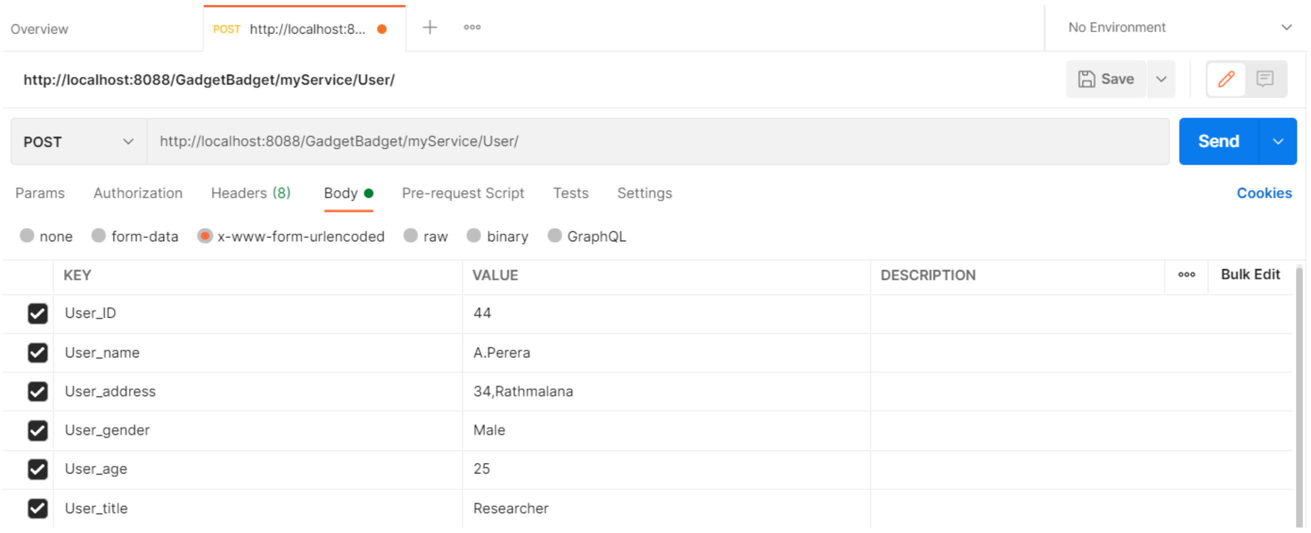
http://localhost:8088/GadgetBadget/myService/User/

GET http://localhost:8088/GadgetBadget/myService/User/

```
{
    "User_ID":"22",
    "User_name":"D.S.Silva",
    "User_address":"78,Kottawa Road, MaharaMaharagama",
    "User_gender":"Male",
    "User_age":"26",
    "User_title":"Buyer"
}
```

Status: 200 OK Time: 75 ms Size: 1.44 KB

User_ID	User_name	User_address	User_gender	User_age	User_title	Update	Remove
15	A.A. Perera	57, Kottawa Road, Maharagama	Male	33	Researcher	Update	Remove
54	P.B.S.Bandara	35, Galle Road, Colombo 3	Male	28	Buyer	Update	Remove
76	K.L. Silva	155, Kotte Road, Nugegoda	Male	27	Researcher	Update	Remove

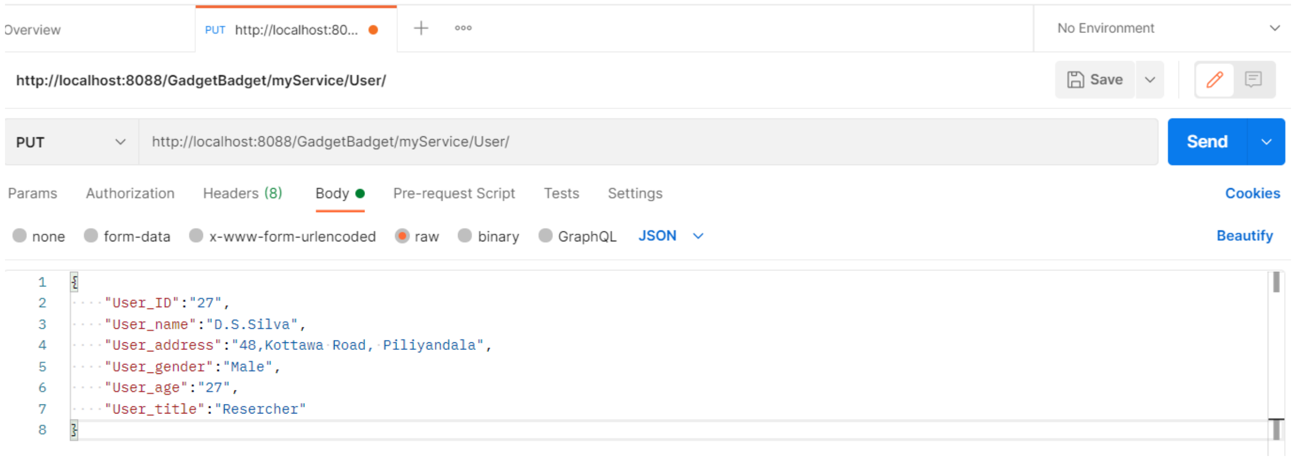


http://localhost:8088/GadgetBadget/myService/User/

POST http://localhost:8088/GadgetBadget/myService/User/

none | form-data | [✓ x-www-form-urlencoded] | raw | binary | GraphQL

KEY	VALUE	DESCRIPTION
✓ User_ID	44	
✓ User_name	A.Perera	
✓ User_address	34,Rathmalana	
✓ User_gender	Male	
✓ User_age	25	
✓ User_title	Researcher	

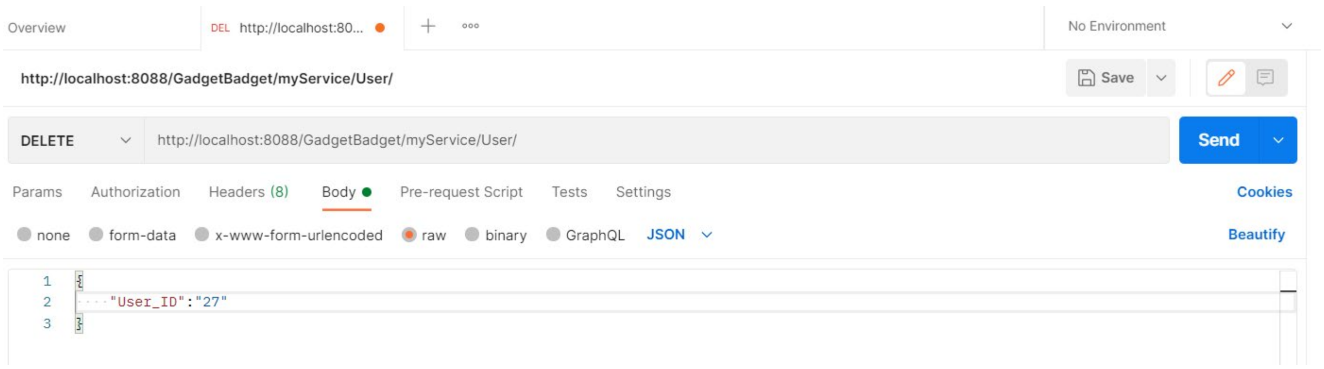


http://localhost:8088/GadgetBadget/myService/User/

PUT http://localhost:8088/GadgetBadget/myService/User/

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "User_ID":"27",
    "User_name":"D.S.Silva",
    "User_address":"48,Kottawa Road, Piliyandala",
    "User_gender":"Male",
    "User_age":"27",
    "User_title":"Resercher"
}
```

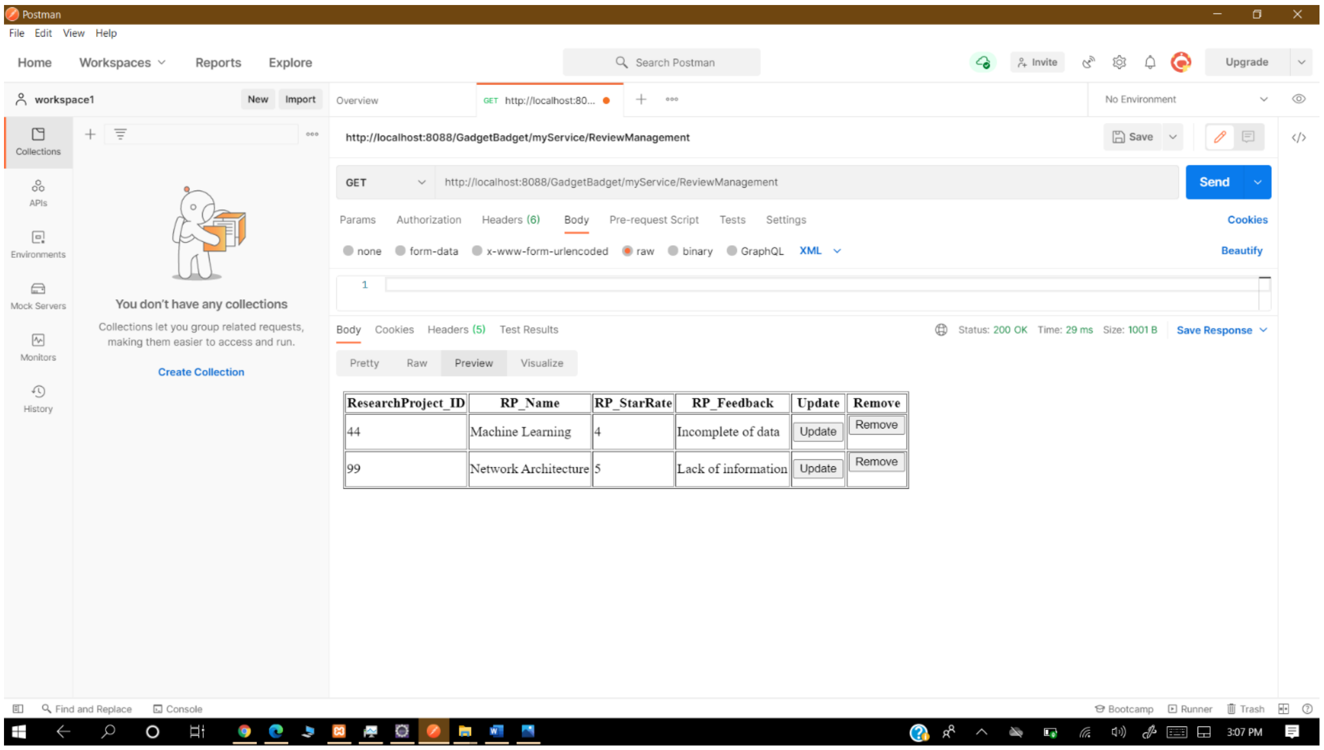


http://localhost:8088/GadgetBadget/myService/User/

DELETE http://localhost:8088/GadgetBadget/myService/User/

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "User_ID":"27"
}
```


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Invite
Upgrade
workspace1
New Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
Overview
No Environment
Save
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
Beautify
Body Cookies Headers (5) Test Results
Save Response
Pretty Raw Preview Visualize
Find and Replace
Console
Bootcamp Runner Trash
3:07 PM

http://localhost:8088/GadgetBadget/myService/ReviewManagement

GET http://localhost:8088/GadgetBadget/myService/ReviewManagement

Body: raw XML

Status: 200 OK Time: 29 ms Size: 1001 B

ResearchProject_ID	RP_Name	RP_StarRate	RP_Feedback	Update	Remove
44	Machine Learning	4	Incomplete of data	Update	Remove
99	Network Architecture	5	Lack of information	Update	Remove

Technical tools which I used:-

- RESTful web service : Java – JAX-RS (Jersy)
- Server – Tomcat 9.0 Server
- Database - MySQL
- Project Management Tool - Maven
- Testing tools – Postman

Section 1.08 System’s Integration Details

Microservices API gateway can greatly reduce coding efforts, make your applications far more efficient, reduce errors all at that same time. We create new maven project and clone every microservice to that using git clone so that all members can run all microservices on their PC. We tested it by using “Postman”.

Section 1.09 References

https://microservices.io/patterns/microservices.html
https://www.infoq.com/articles/application-integration-service-mesh/
https://dzone.com/articles/principles-for-microservices-integration
https://medium.com/hashmapinc/the-what-why-and-how-of-a-microservices-architecture4179579423a9
https://blog.newrelic.com/technology/microservices-what-they-are-why-to-use-them/