GadgetBadget System
Programming Applications and Frameworks(IT3030) 2021
IT3030PAF2021_GroupProject_11
Main Group 01(WeekEnd)

- **Background**
  GadgetBadget (GB) is a company who funds innovative projects and help the young researches to sell their products via company's online platform.
- **Problem**
  The number of stakeholders of GB (researchers, funding bodies, buyers, etc…) is growing and the current system is not scalable.
- **Aim**
  The aim of the project is to develop a highly scalable online platform for the stakeholders of GB to interact with each other.
- **Submitted by:**

Registration No	Name
IT19110462	Miyanadeniya M.K.M.P
IT19060804	Kumari T.N
IT19052298	Hewabathma C.D
IT19062952	P.H.D Sigera

Table of contents

Members Details

Student ID	Student Name	Workload Distribution
IT19110462	Miyanadeniya M.K.M.P	Funding Body Service
IT19060804	Kumari T.N	Buyer Service
IT19052298	Hewabathma C.D	Researcher Service
IT19062952	P.H.D Sigera	Courier Service

Clickable link to the public VCS repo

https://github.com/ChamodiDH/GadgetBadget_PAF.git

SE methodology/methods (with justification)

Requirement Gathering and analysis – Gathering the requirements of stakeholders.
System Design – Designing the;

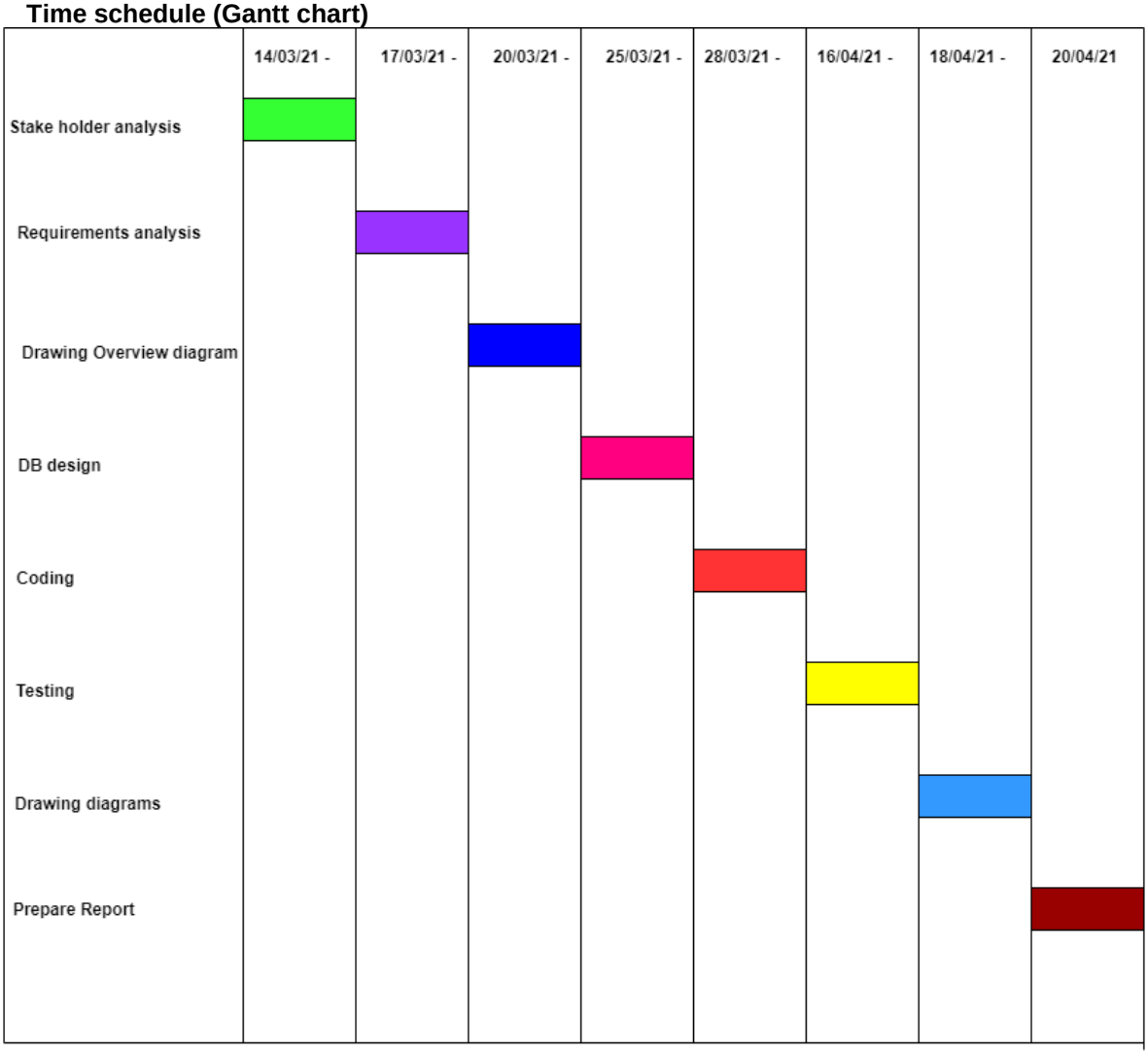
- Overview diagram.
- Onion diagram.
- Class diagram.
- Sequence diagram.
- Use case diagram.
- activity diagram.

Implementation – Implementation of the services;

- Funding Body Service
- Buyer Service
- Researcher Service
- Courier Service

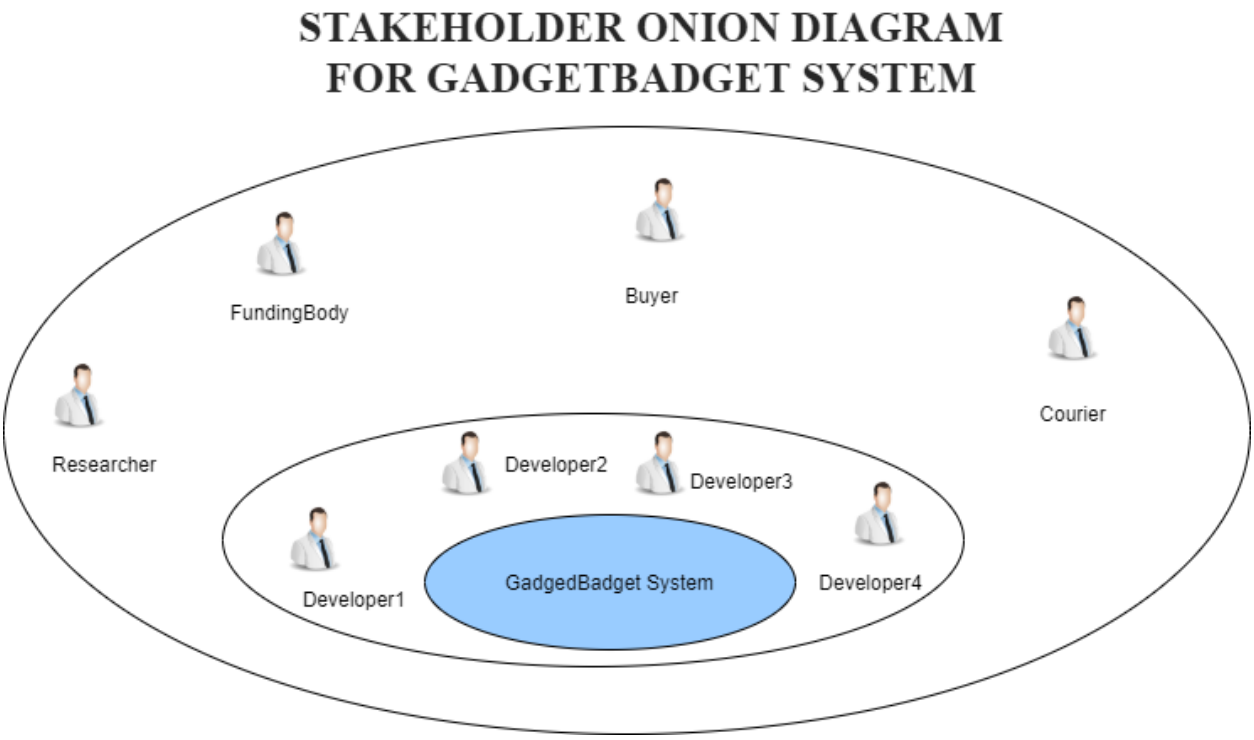
Integration and Testing – Integration the services of the GadgetBadget system, Testing the system using a client.

Time schedule (Gantt chart)



Requirements.

- Stakeholder analysis (onion diagram).



This diagram shows the dependencies among the parts of the Gadgetbadget system. The chart displays items in concentric circles, where the items in each ring depend on the items in the smaller rings. The outermost layer shows the stakeholders of the system such as Researcher, Buyer, Funding Body and Courier. The innermost layer shows the developers who has developed the Researcher Service, Buyer Service, Funding Body Service, Courier Service.

- Requirements analysis (, , Technical requirements).

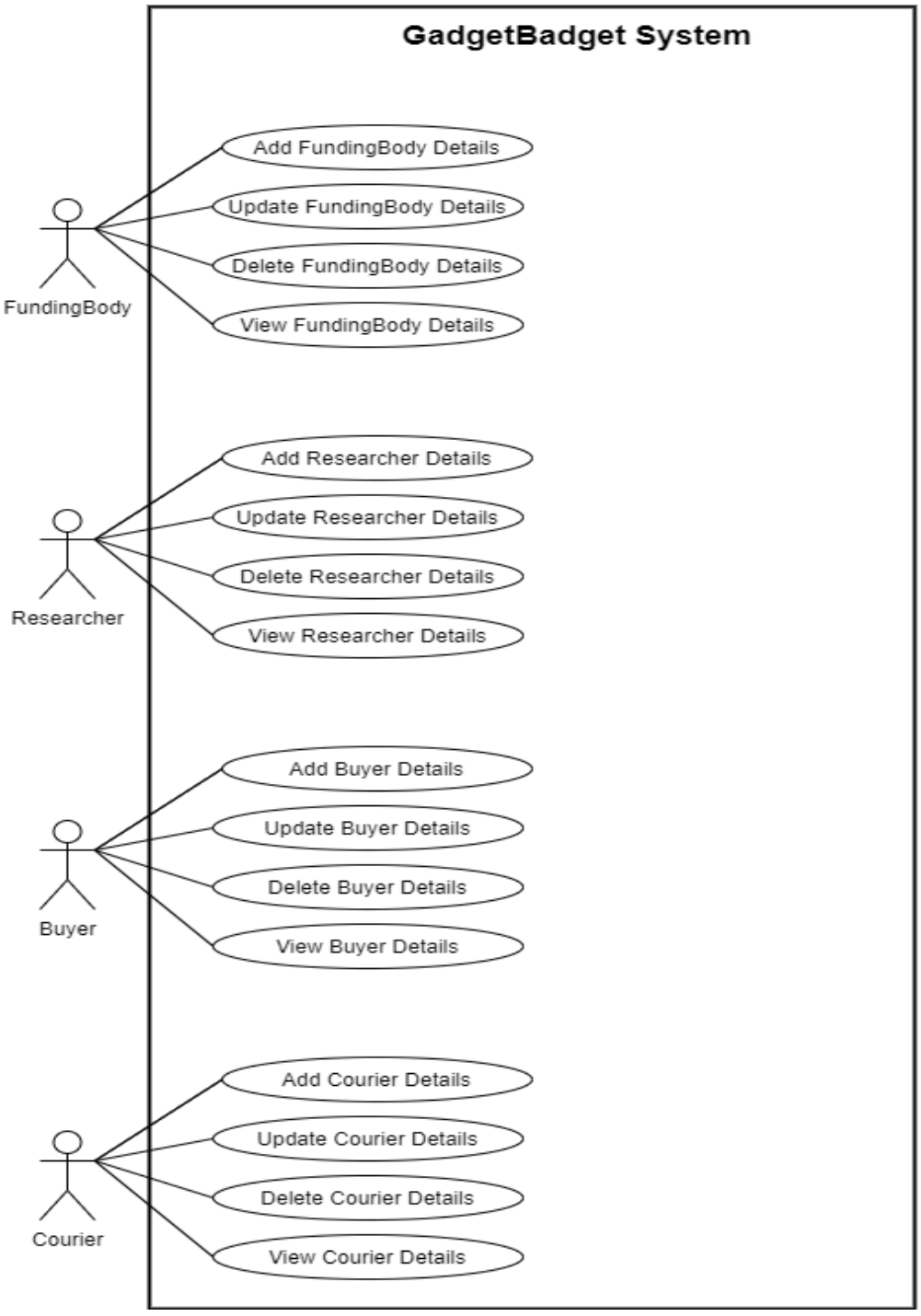
1.Functional Requirements
Add Researcher Details.
Update Researcher Details.
Delete Researcher Details.
Retrieve Researcher Details.
Add Funding Body Details.
Update Funding Body Details.
Delete Funding Body Details.
Retrieve Funding Body Details.
Add Buyer Details.
Update Buyer Details.
Delete Buyer Details.
Retrieve Buyer Details.

Add Courier Details.
Update Courier Details.
Delete Courier Details.
Retrieve Courier Details.

2. Non-functional Requirements

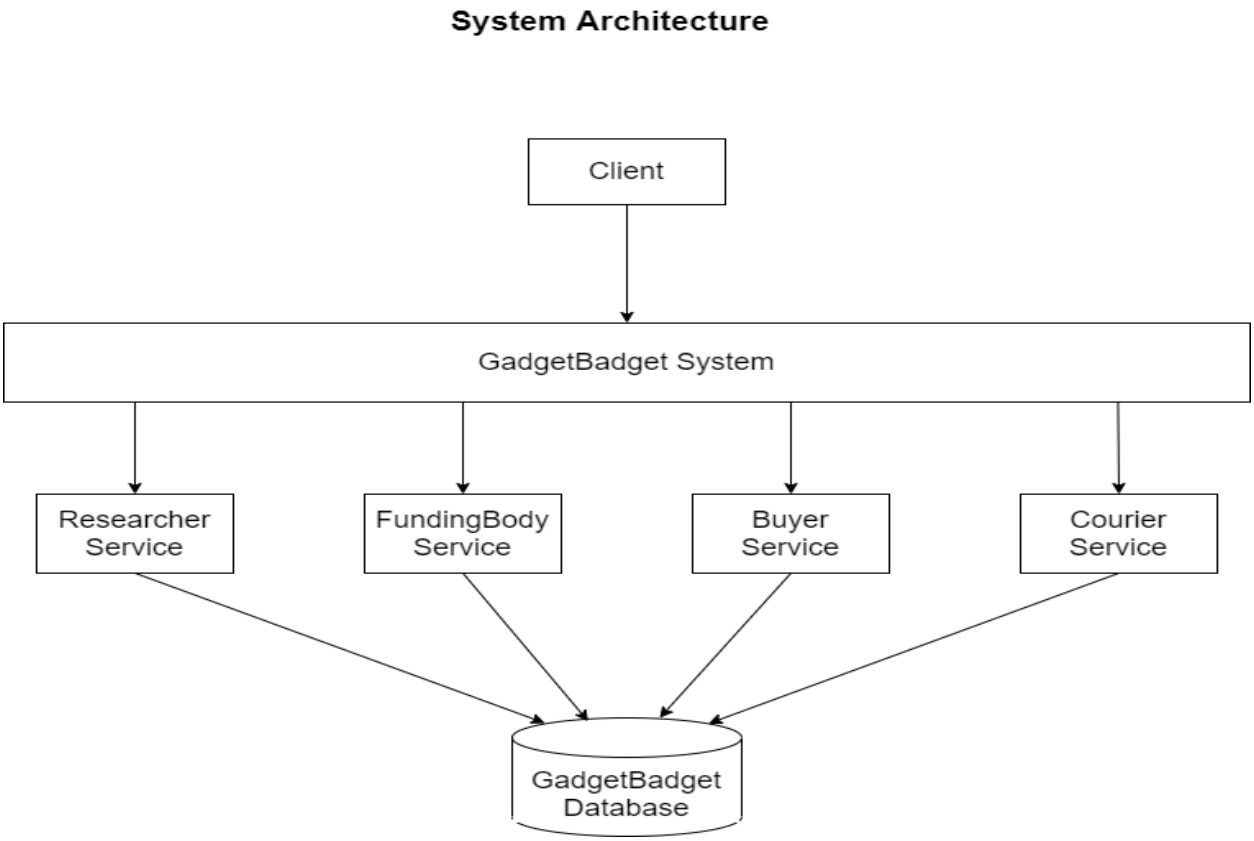
- Scalability
- Capacity
- Availability
- Reliability
- Recoverability
- Maintainability
- Serviceability
- Security
- Regulatory
- Manageability
- Environmental
- Data Integrity
- Usability

Requirements modelling (Use case diagram)



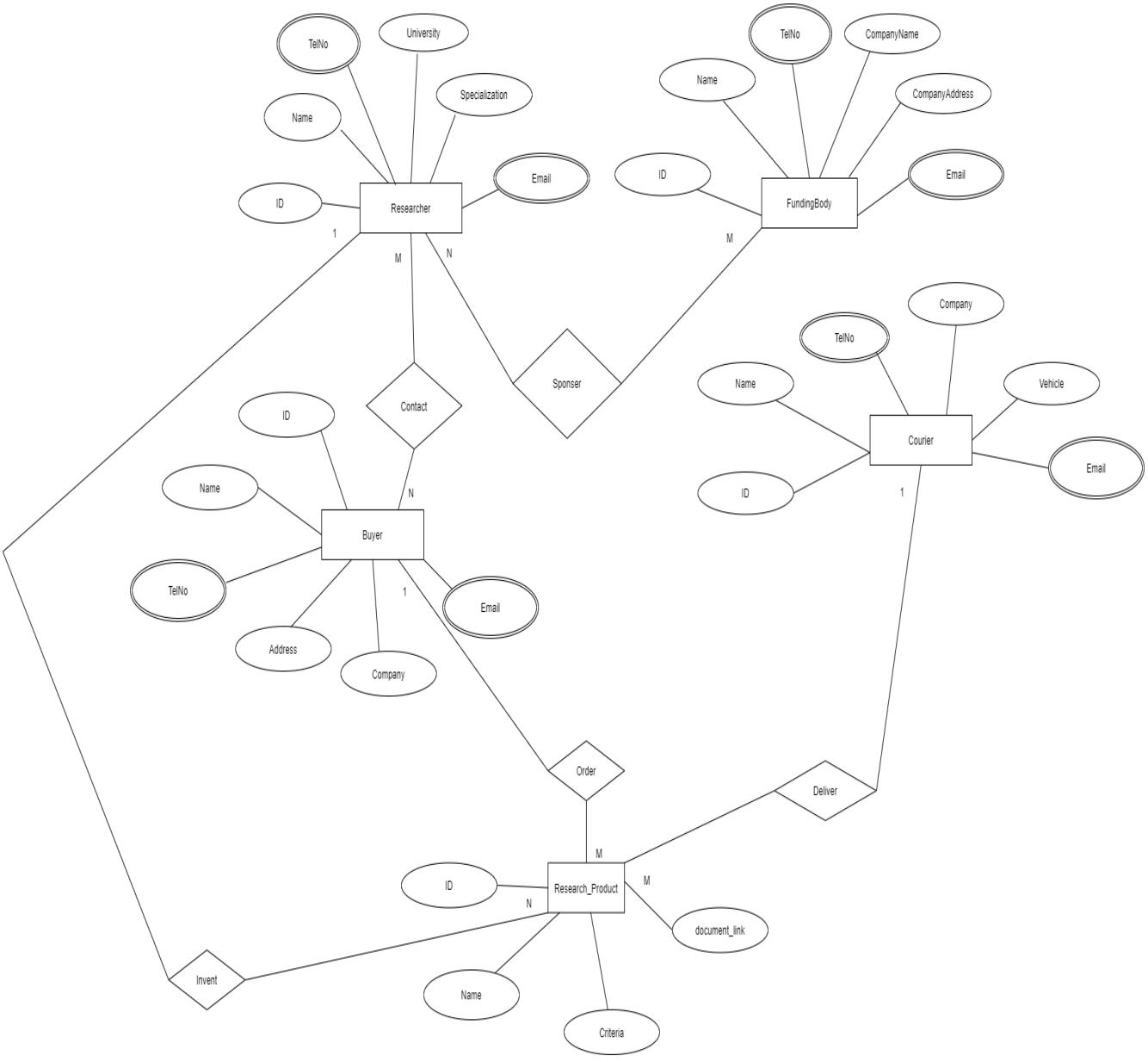
System’s overall design

Overall architecture.



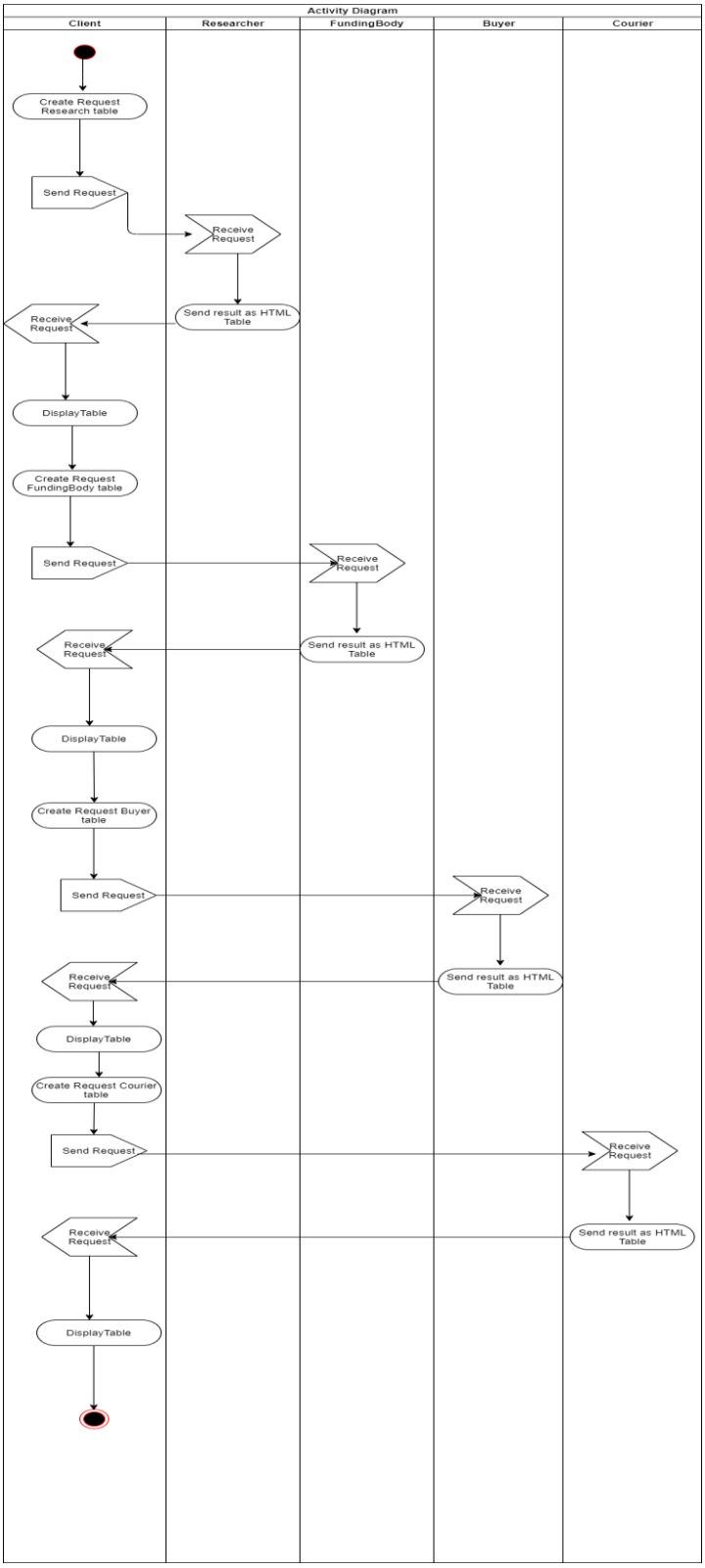
The above diagram shows the system architecture of Gadgetbadget System which is used by client to connect with four services such as Researcher Service, Funding Body Service, Buyer Service, Courier Service.

Overall DB design (ER diagram)



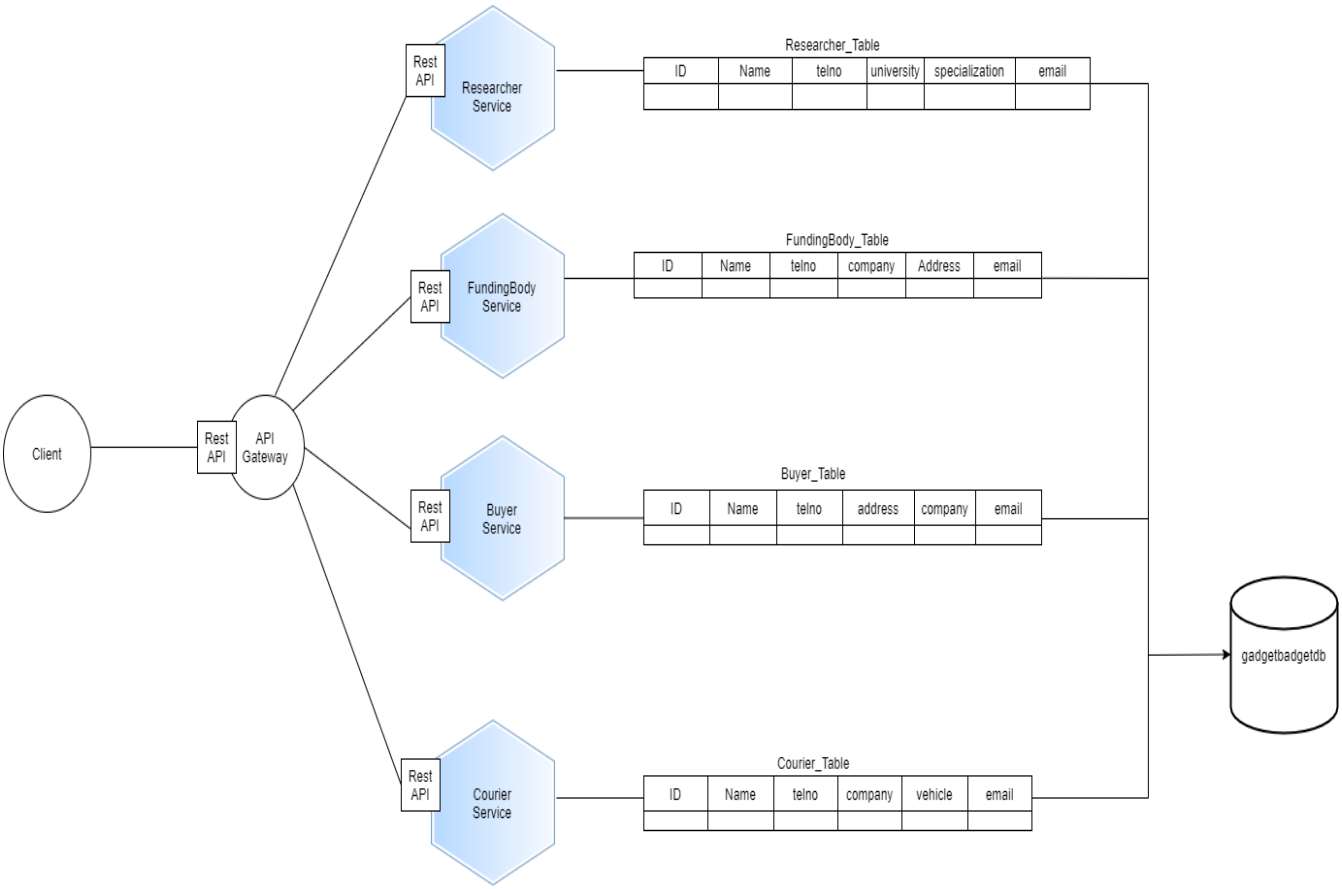
The above ER diagram is drawn for the Gadgetbadget System which shows the relationships among the entities and their attributes.

Activity diagram.



This activity diagram visually presents the series of actions which is conducted in the Gadgetbadget System.

Overview diagram

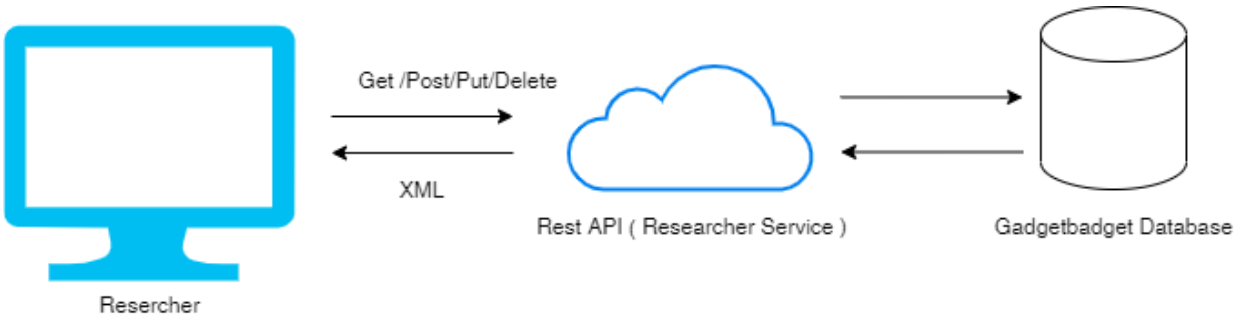


This Overview Diagram shows how the client can obtain the services of the Gadgetbadget System via REST API. The services provided by the Gadgetbadget System are Researcher Service, Funding Body Service, Buyer Service and Courier Service.

Individual sections. – Researcher Service

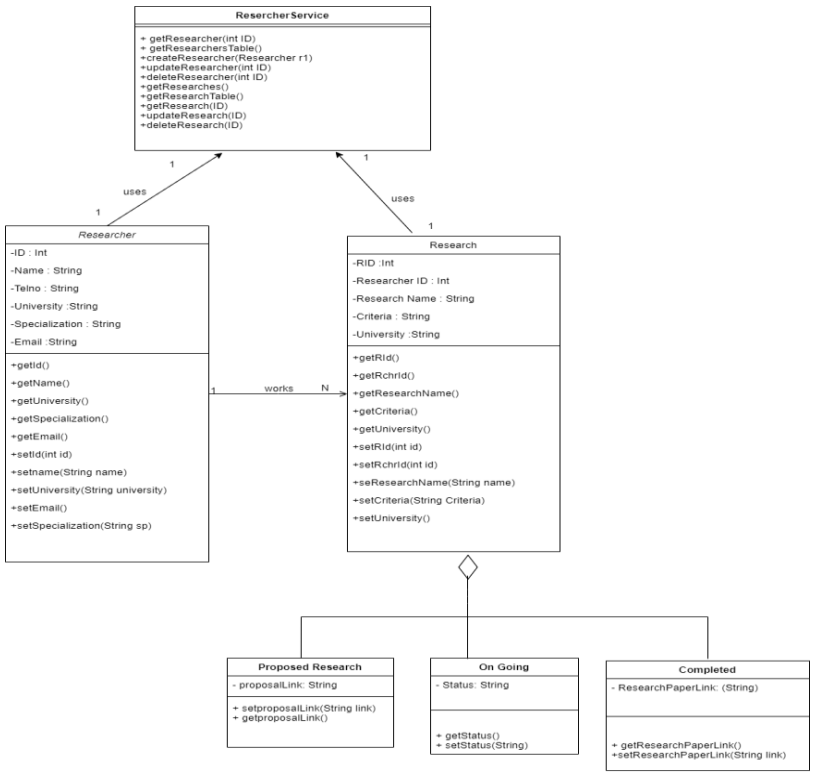
Service design.

- API of the service.

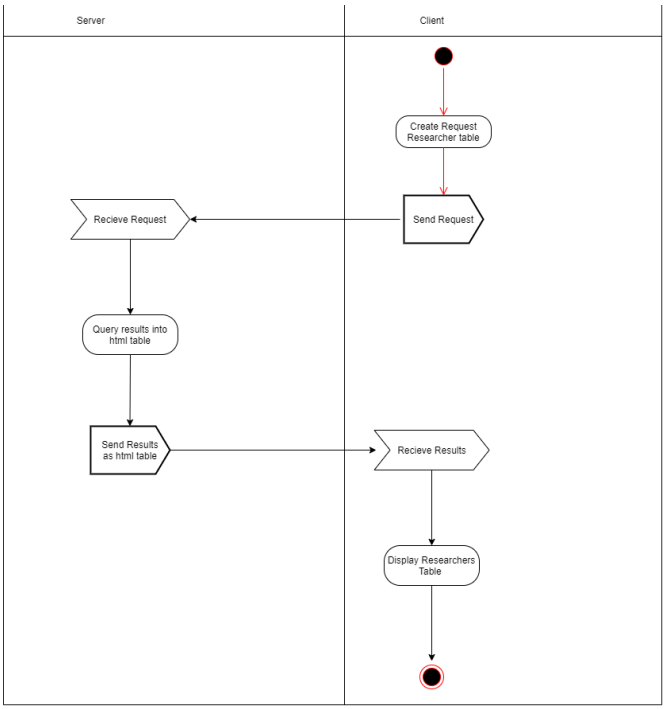


Rest API is used by researcher to communicate with other services in the Gadgetbadget System via Researcher Service.

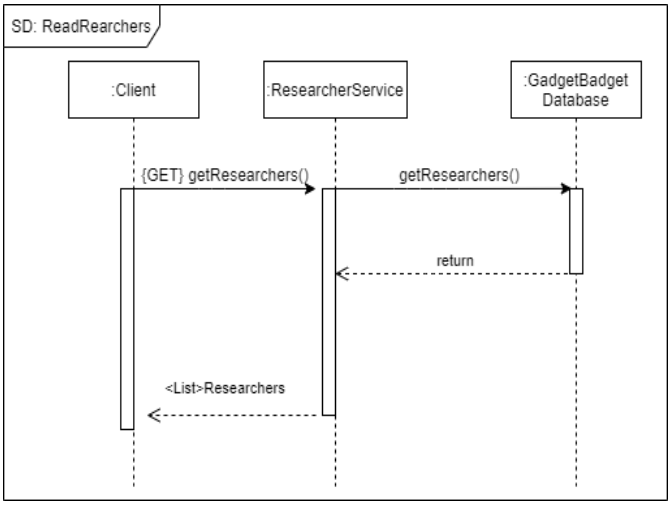
Internal logic (Class diagram)

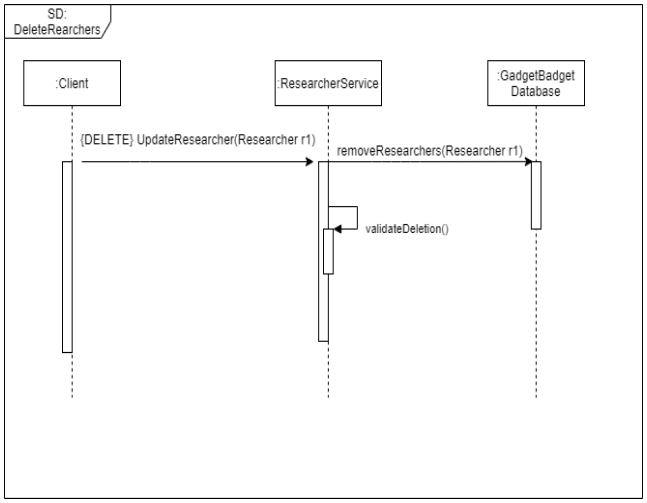


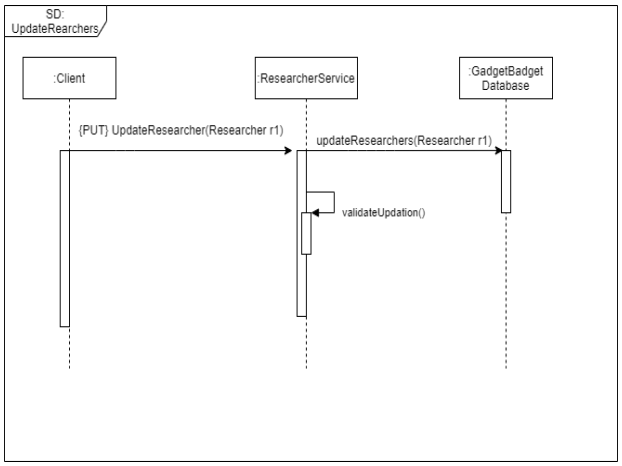
Activity Diagram

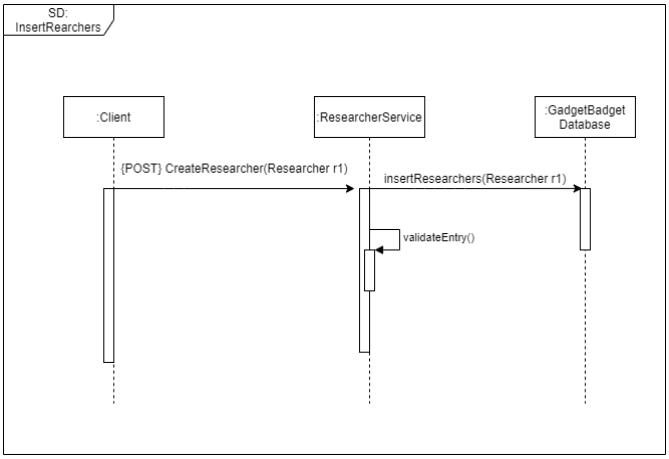


Sequence diagram









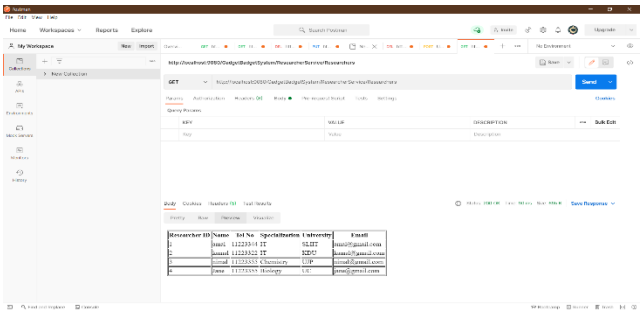
- Service development and testing.
  - Tools used.

- Dependency management tools. – Maven
- IDE – Eclipse
- Client – Postman

Testing methodology and results.

Testing CRUD.

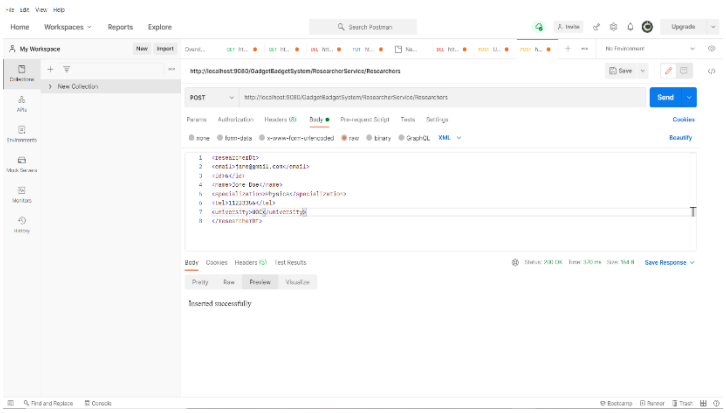
- GET



1. Start the tomcat Server in Eclipse.
2. Open Postman.
3. Select GET from the Postman.
4. Give the following link.
   http://localhost:9080/GadgetBadgetSystem/ResearcherService/Researchers/

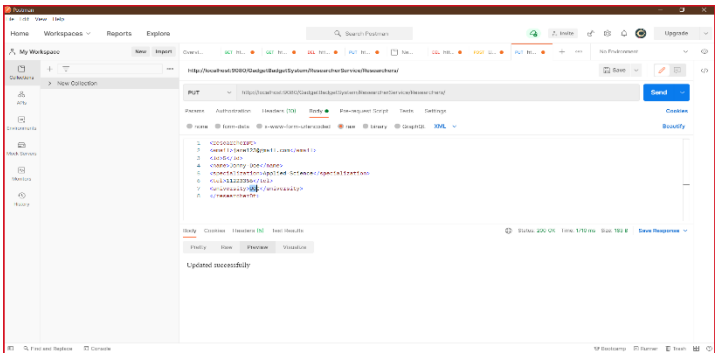
Result : HTML table with researchers details will be displayed to the user.

- POST



1. Start the tomcat Server in Eclipse
2. Open Postman.
3. Select POST from the Postman.
4. Give the following XML query.
5. <researcherDt>
   <email>jane@gmail.com</email>
   <id>6</id>
   <name>Jone Doe</name>
   <specialization>Physics</specialization>
   <tel>11223356</tel>
   <university>UOC</university>
   </researcherDt>
   Result : “Inserted Successfully” message will be displayed to the user in the Postman. Data will be added to the database.

- PUT

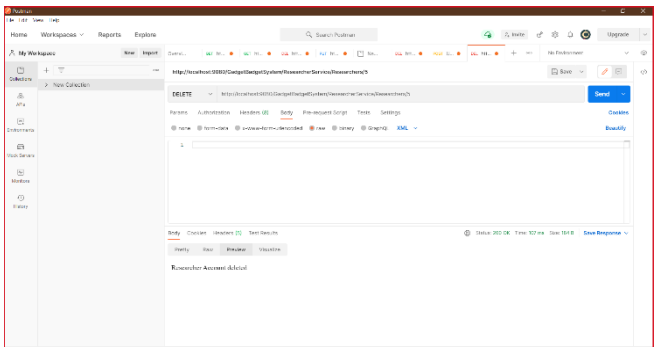


1. Start the tomcat Server in Eclipse
2. Open Postman.
3. Select PUT from the Postman.
4. Give the following XML query.

```
<researcherDt>
 <email>jane123@gmail.com</email>
 <id>5</id>
 <name>Jonny Doe</name>
 <specialization>Applied
Science</specialization>
 <tel>11223356</tel>
 <university>UOC</university>
</researcherDt>
```

Result : “Updated Successfully” message will be displayed to the user in the Postman. Data will be Updated in the database.

- Delete

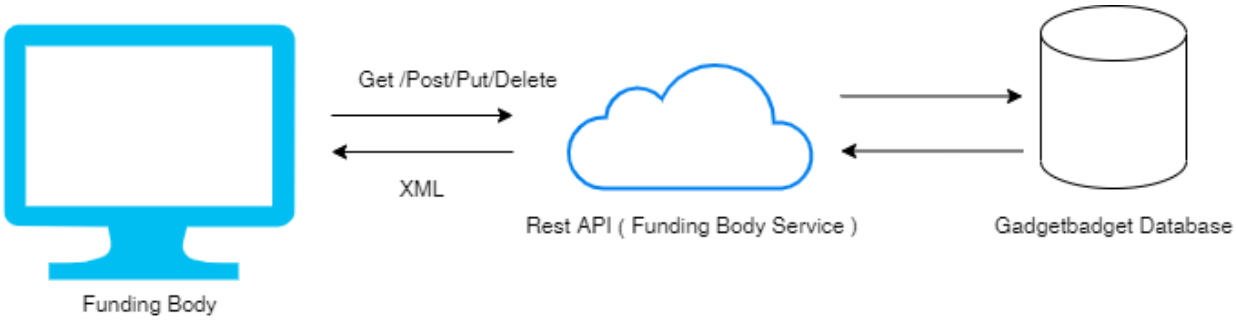


1. Start the tomcat Server in Eclipse
2. Open Postman.
3. Select DELETE from the Postman.
4. Give the following link.
   http://localhost:9080/GadgetBadgetSystem/ResearcherService/Researchers/5
   Result : “Researcher Account Deleted ” message will be displayed to the user in the Postman. Data will be Deleted from the database.

Individual sections. – Funding Body Service

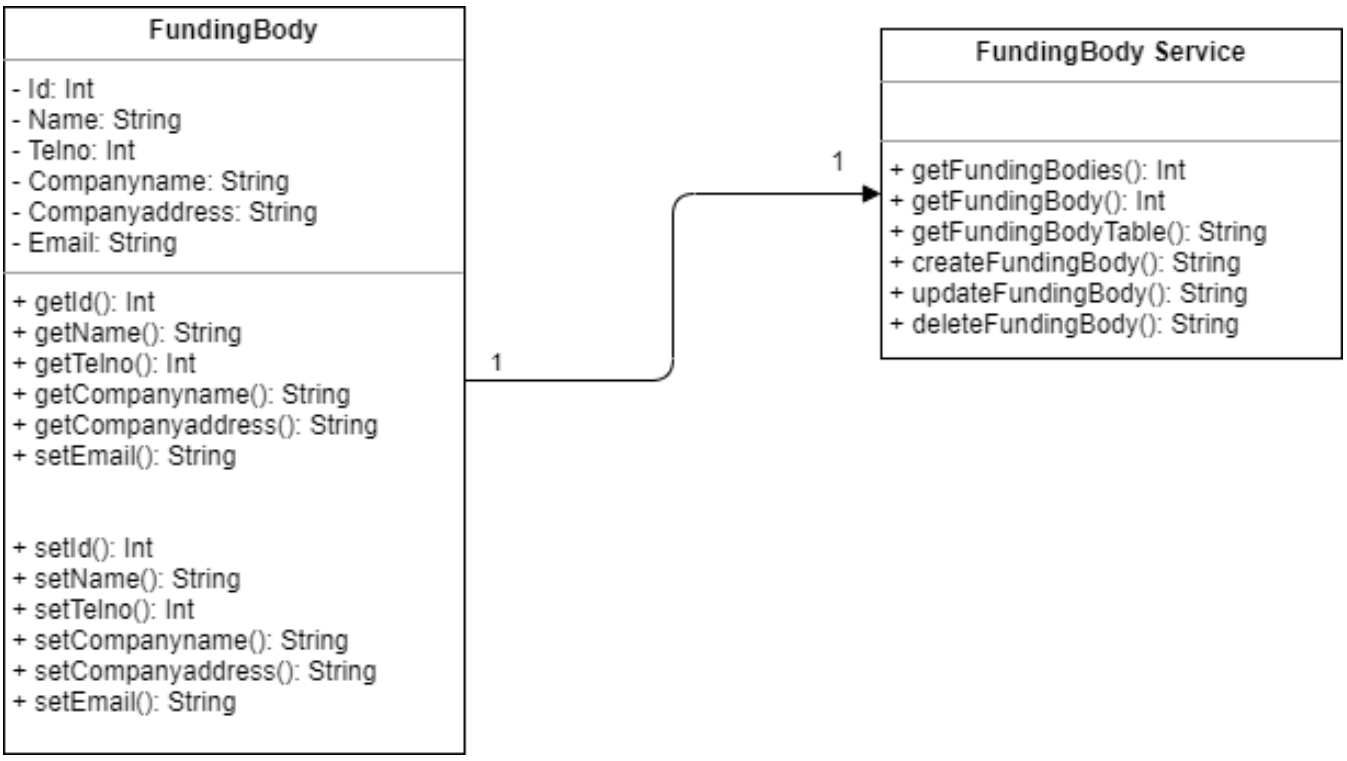
Service design.

- API of the service.

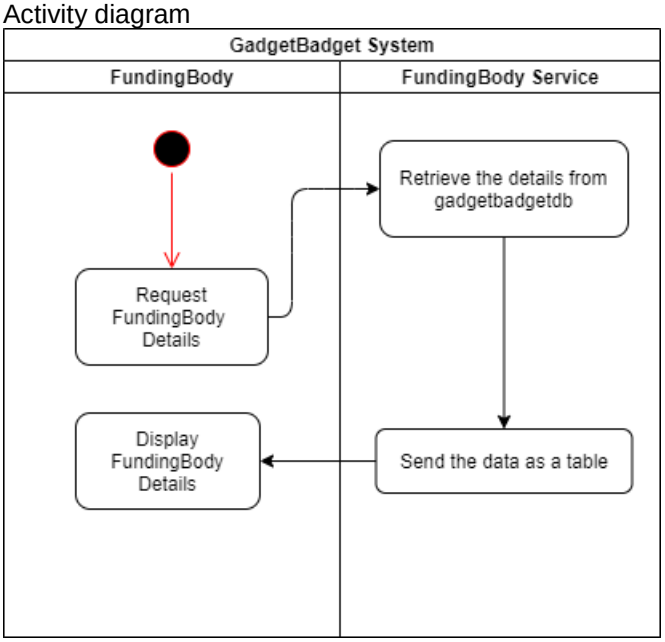


Rest API is used by Funding Body to communicate with other services in the Gadgetbadget System via Funding Body Service.

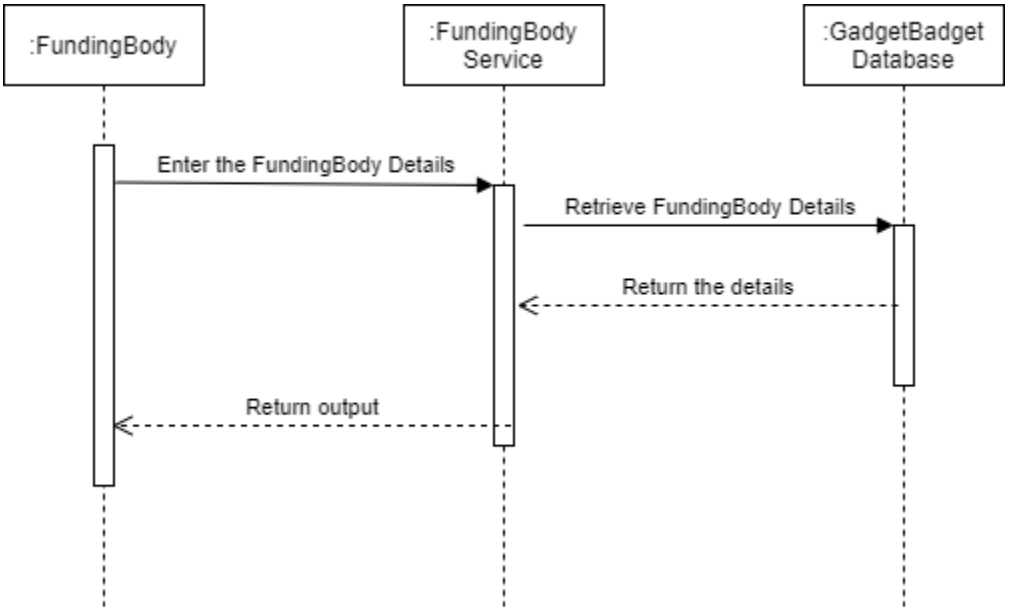
Internal logic (Class diagram)

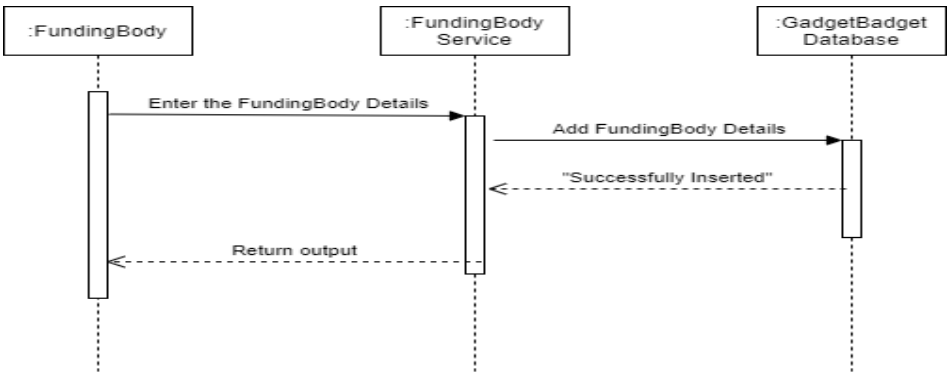


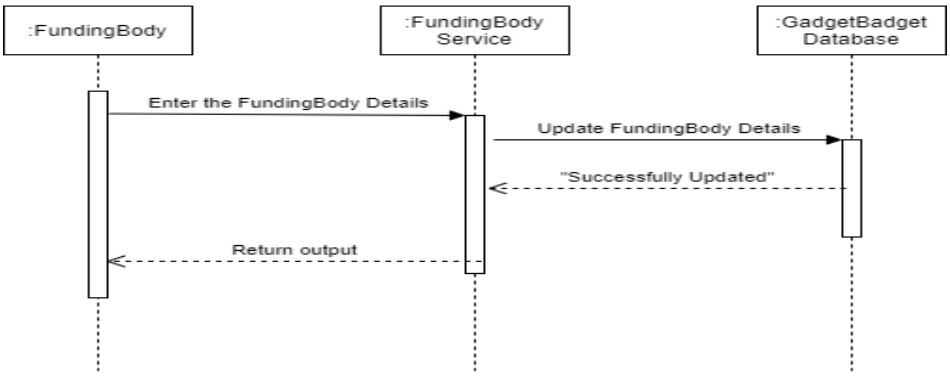
Activity diagram

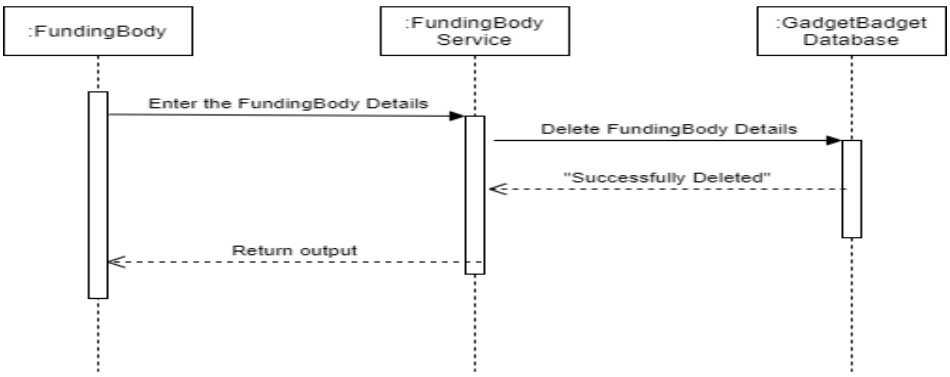


Sequence diagram









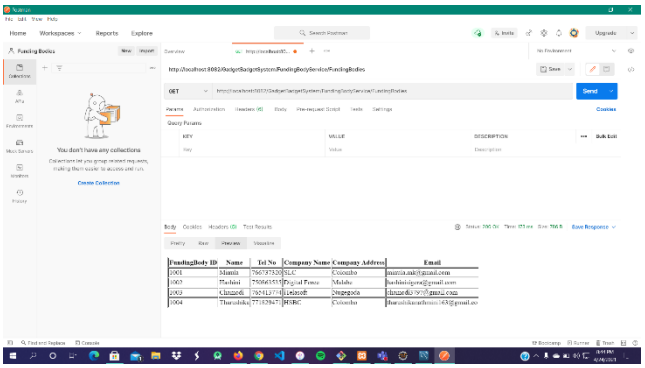
- Service development and testing.
  - Tools used.

    - Dependency management tools. – Maven
    - IDE – Eclipse
    - Client – Postman

Testing methodology and results.

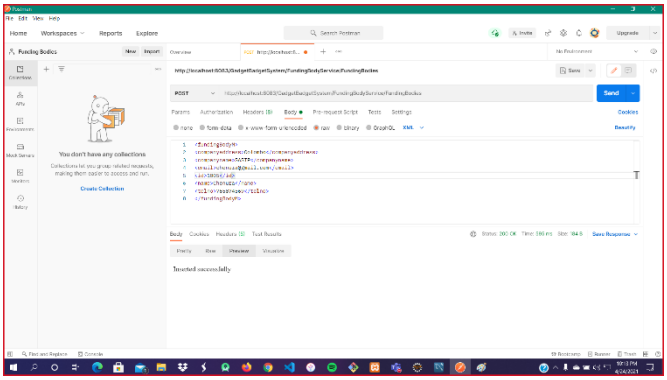
Testing CRUD.

- GET



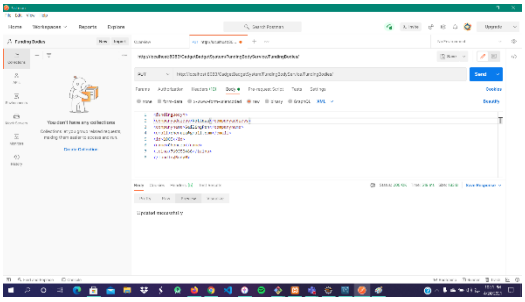
1. Start the tomcat Server in Eclipse.
2. Open Postman.
3. Select GET from the Postman.
4. Give the following link.
   http://localhost:8082/GadgetBadgetSystem/FundingBodyService/FundingBodies/
   Result : HTML table with Funding Body details will be displayed to the user.

- POST



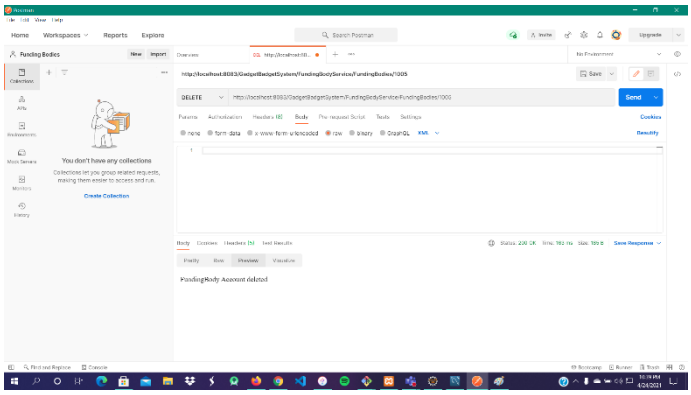
1. Start the tomcat Server in Eclipse.
2. Open Postman.
3. Select POST from the Postman.
4. Give the following XML query.
   <fundingBodyM>
   <companyaddress>Colombo</companyaddress>
   <companyname>SASIP</companyname>
   <email>chenura@gmail.com</email>
   <id>1005</id>
   <name>Chenura</name>
   <telno>765874569</telno>
   </fundingBodyM>
   Result : “Inserted Successfully” message will be displayed to the user in the Postman. Data will be added to the database.

- PUT



1. tart the tomcat Server in Eclipse
2. Open Postman.
3. Select PUT from the Postman.
4. Give the following XML query.
   <fundingBodyM>
   <companyaddress>Kottawa</companyaddress>
   <companyname>SailingPen</companyname>
   <email>chenura@gmail.com</email>
   <id>1005</id>
   <name>Chenura</name>
   <telno>769350466</telno>
   </fundingBodyM>
   Result : “Updated Successfully” message will be displayed to the user in the Postman. Data will be Updated in the database.

- DELETE

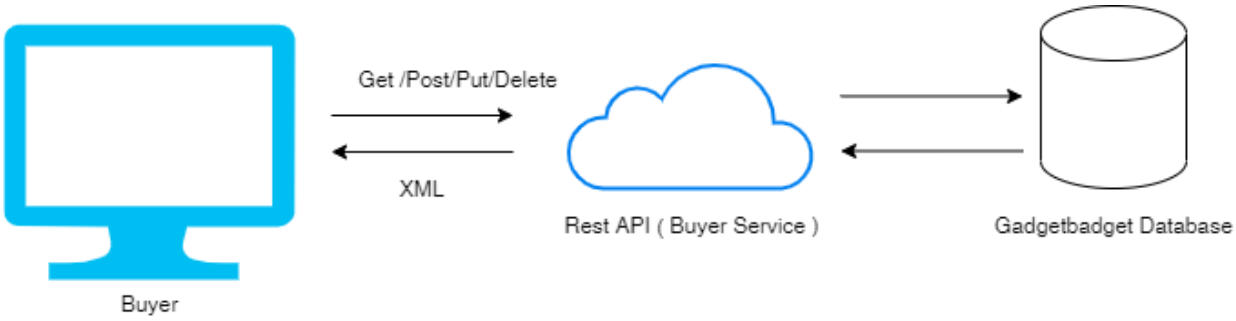


1. Start the tomcat Server in Eclipse
2. Open Postman.
3. Select DELETE from the Postman.
4. Give the following link.
   http://localhost:8083/GadgetBadgetSystem/FundingBodyService/FundingBodies/1005
   Result : “FundingBody Account Deleted ” message will be displayed to the user in the Postman. Data will be Deleted from the database.

Individual sections. – Buyer Service

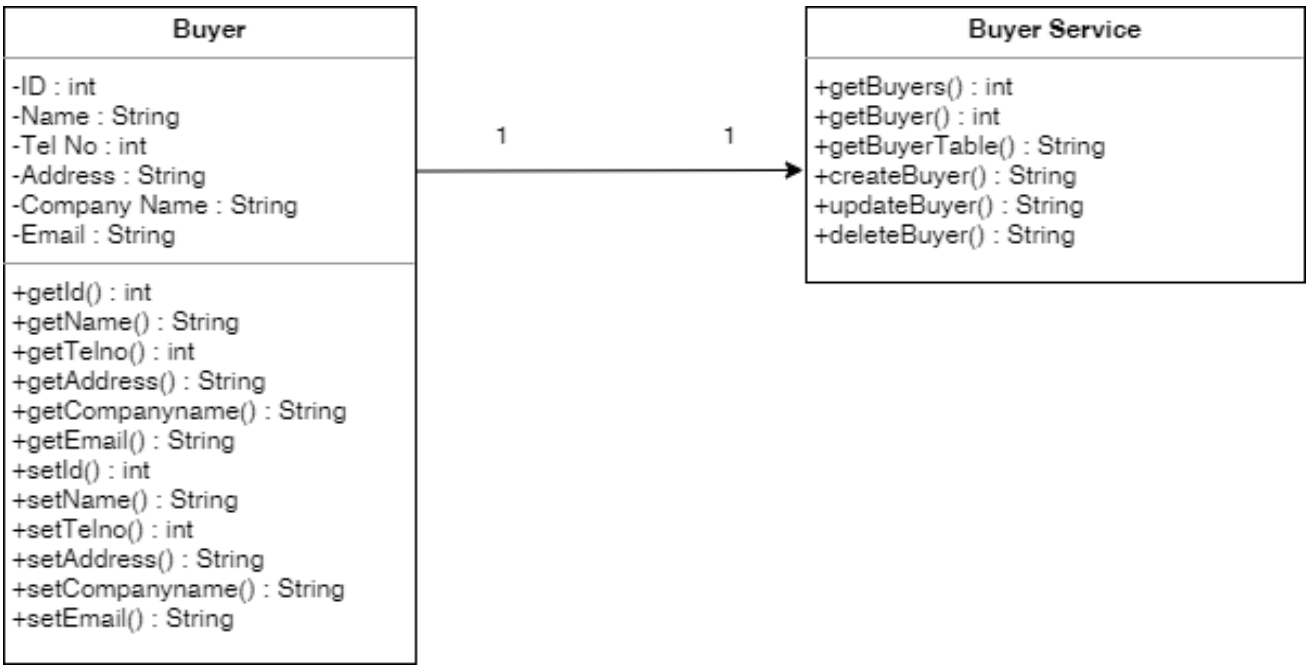
Service design.

- API of the service.

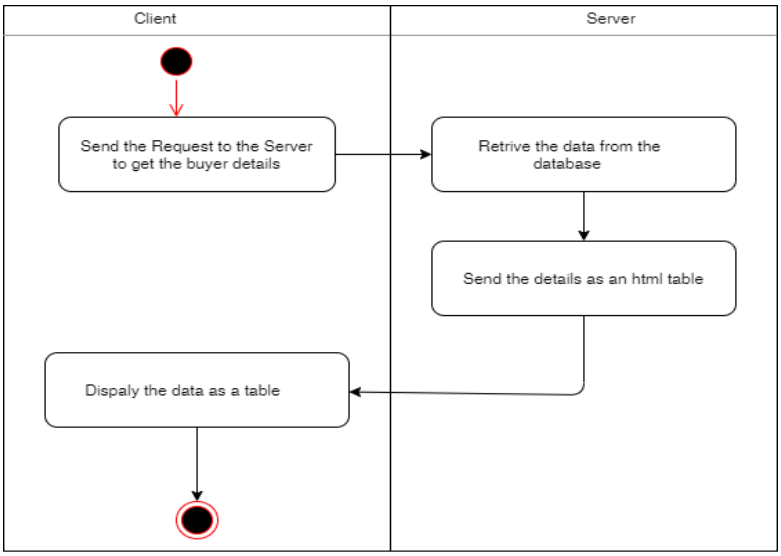


Rest API is used by Buyer to communicate with other services in the Gadgetbadget System via Buyer Service.

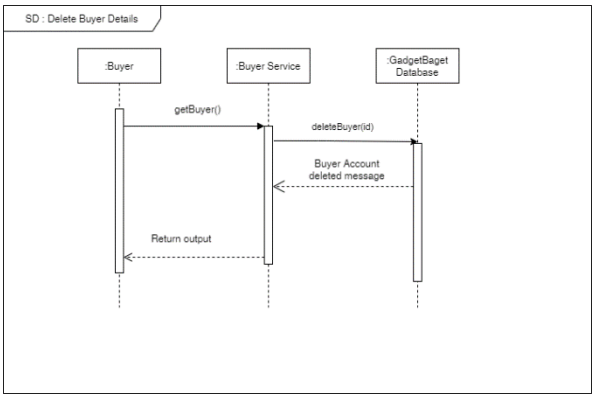
Internal logic (Class diagram)

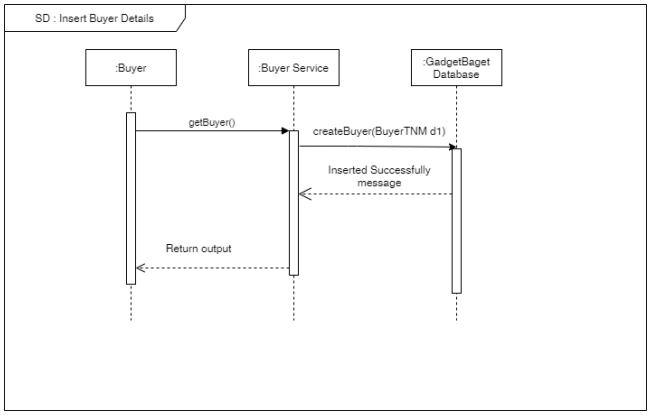


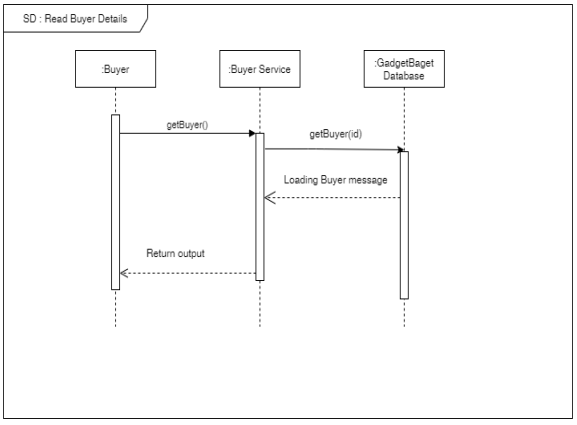
Activity diagram

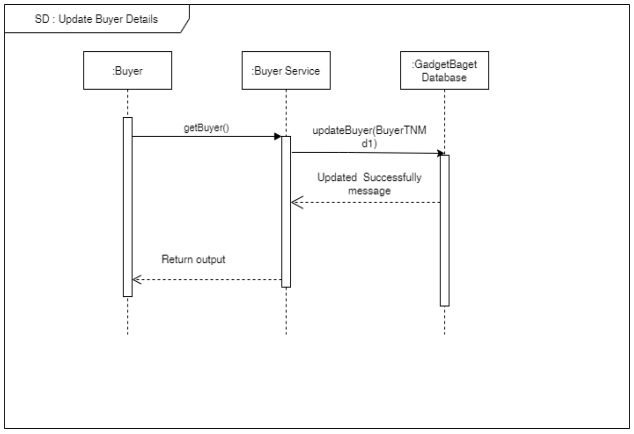


Sequence diagram









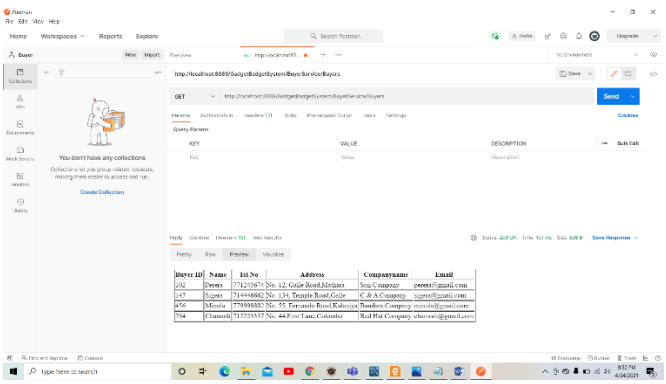
- Service development and testing.
  - Tools used.

    - Dependency management tools. – Maven
    - IDE – Eclipse
    - Client – Postman

Testing methodology and results.

Testing CRUD.

- GET

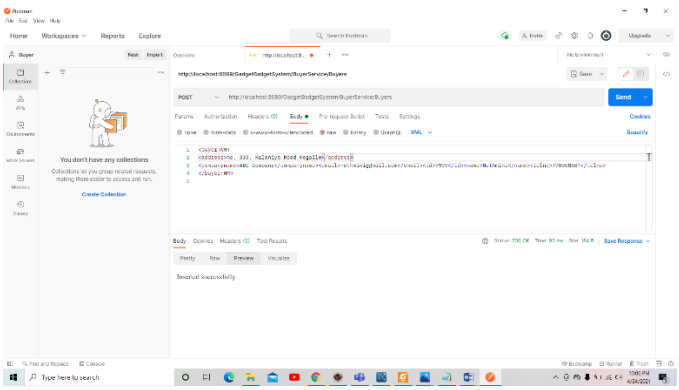


1. Start the tomcat Server in Eclipse.
2. Open Postman.
3. Select GET from the Postman.
4. Give the following link.

http://localhost:8089/GadgetBadgetSystem/BuyerService/Buyers/

Result : HTML table with Buyers details will be displayed to the user.

- POST

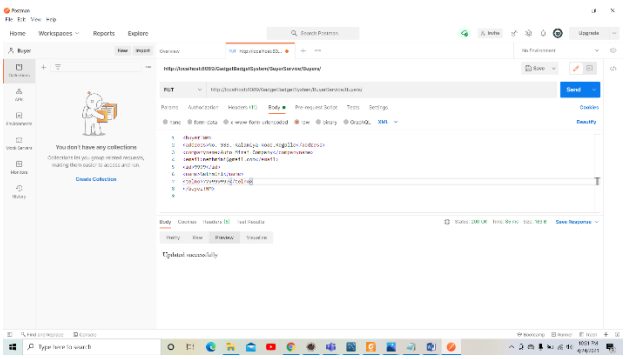


1. Start the tomcat Server in Eclipse
2. Open Postman.
3. Select POST from the Postman.
4. Give the following XML query.

```
<buyerTNM>
<address>No. 333, Kalaniya Road,Kegalle</address>
 <companyname>ABC Company</companyname>
<email>nethmini@gmail.com</email>
<id>9999</id>
<name>Nethmini</name>
<telno>770008884</telno>
</buyerTNM>
```

Result : “Inserted Successfully” message will be displayed to the user in the Postman. Data will be added to the database.

- PUT



1. Start the tomcat Server in Eclipse
2. Open Postman.
3. Select PUT from the Postman.
4. Give the following XML query.

<buyerTNM>

<address>No. 333, Kalaniya Road,Kegalle</address>

<companyname>Auto Miraj Company</companyname>

<email>nethmini@gmail.com</email>

<id>9999</id>

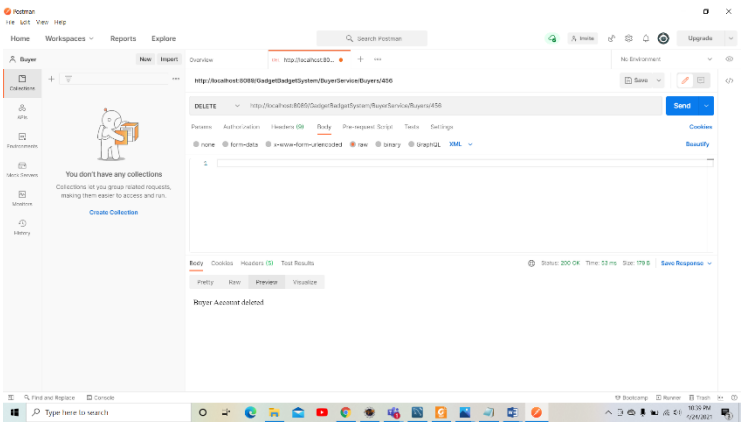
<name>Nethmini</name>

<telno>779999997</telno>

</buyerTNM>

Result : “Updated Successfully” message will be displayed to the user in the Postman. Data will be Updated in the database.

- DELETE



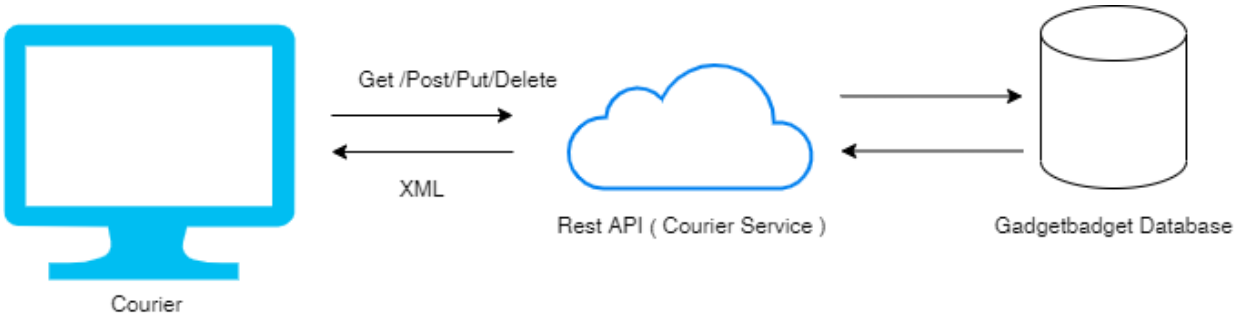
1. Start the tomcat Server in Eclipse
2. Open Postman.
3. Select DELETE from the Postman.
4. Give the following link. http://localhost:9080/GadgetBadgetSystem/ResearcherService/Researchers/456

Result : “Buyer Account Deleted ” message will be displayed to the user in the Postman. Data will be Deleted from the database.

Individual sections. – Courier Service

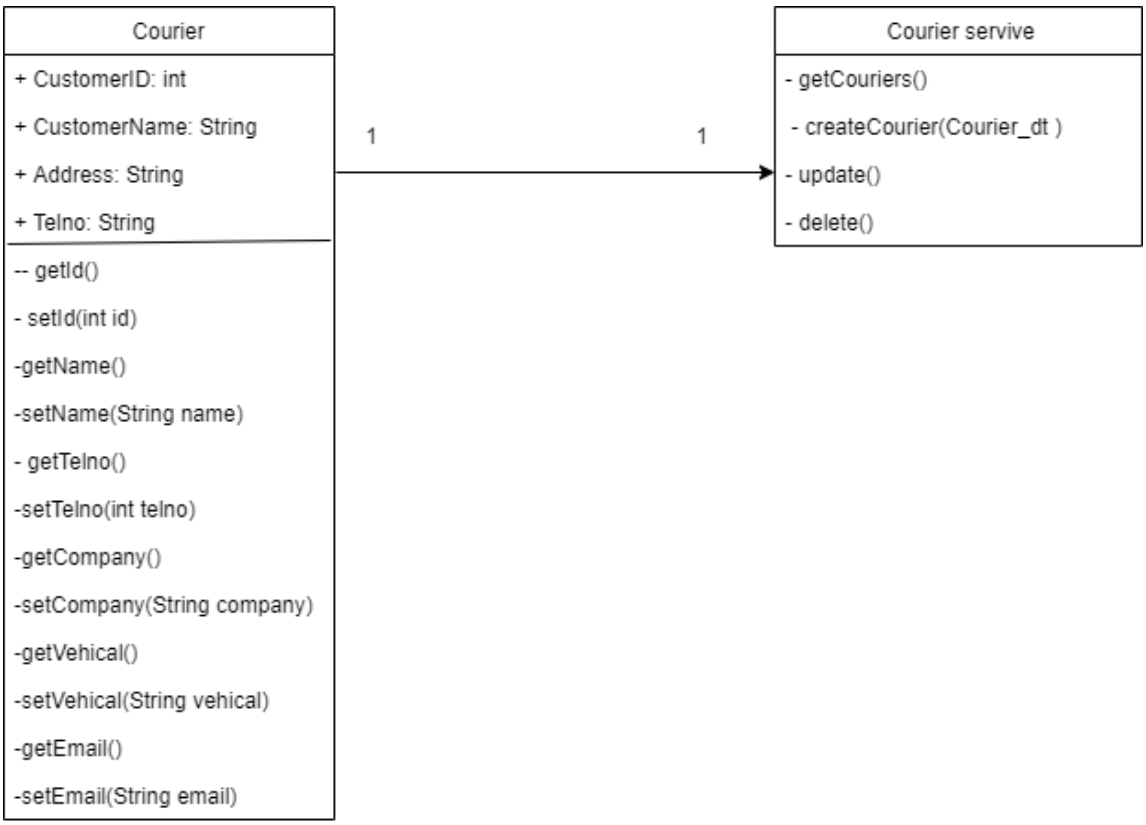
Service design.

- API of the service.

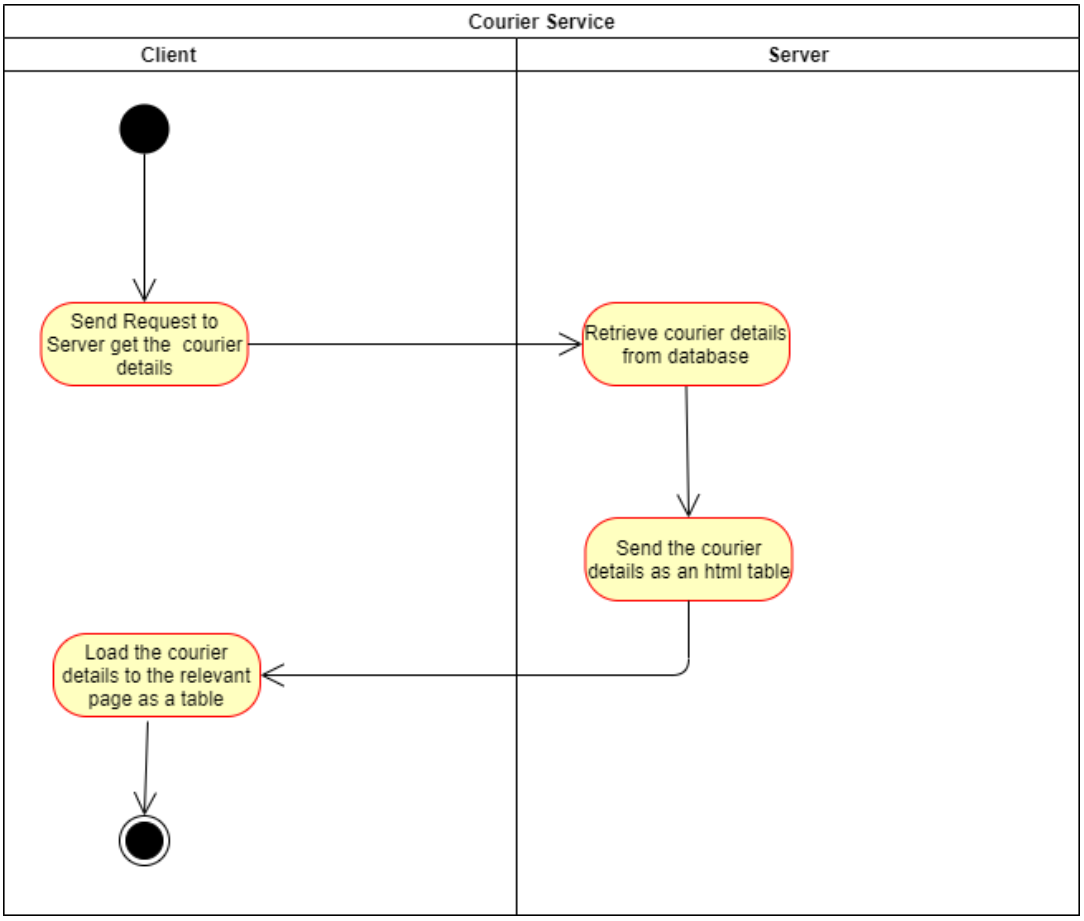


Rest API is used by Courier to communicate with other services in the Gadgetbadget System via Courier Service.

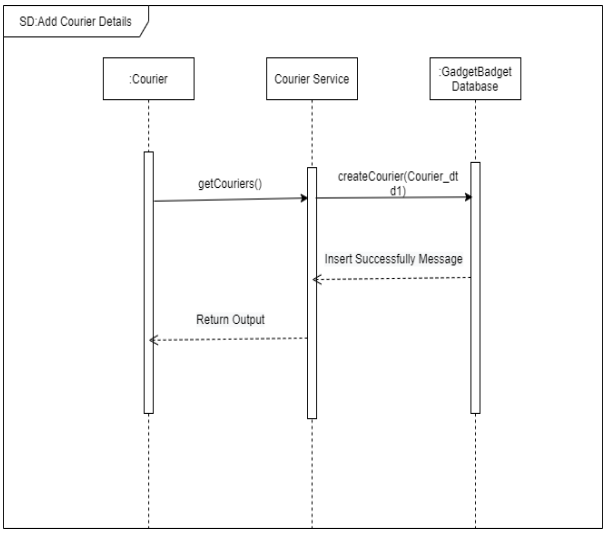
Internal logic (Class diagram)

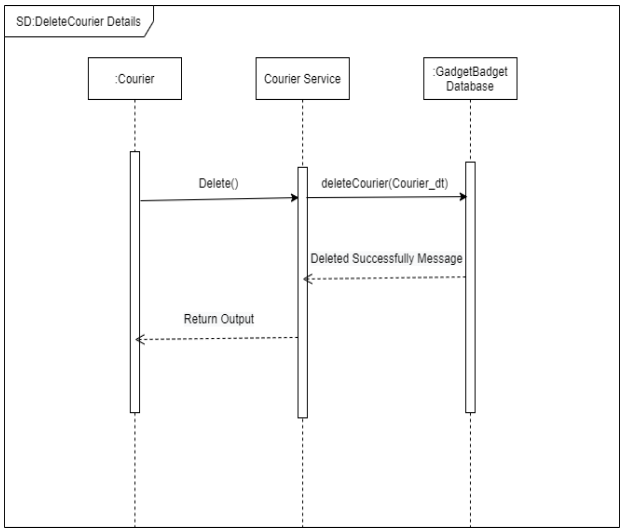


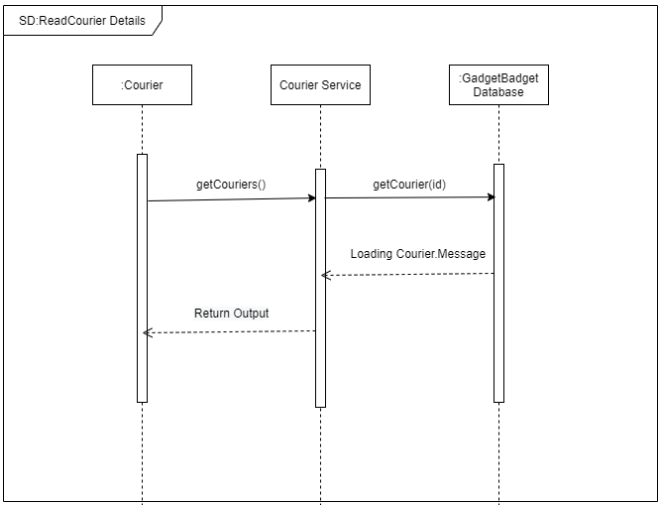
Activity diagram

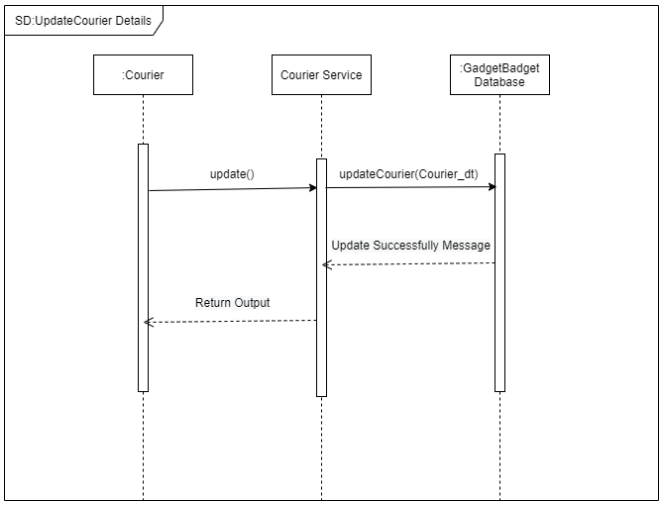


Sequence diagram









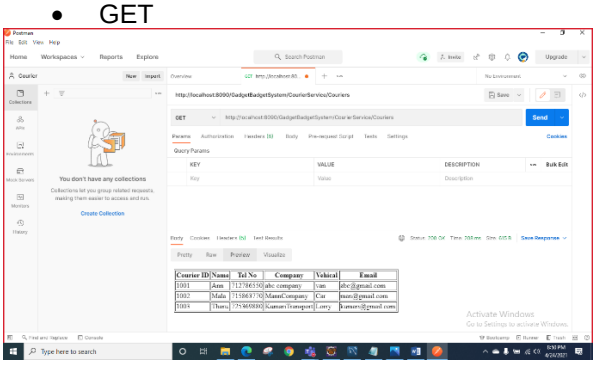
- Service development and testing.
  - Tools used.

    - Dependency management tools. – Maven
    - IDE – Eclipse
    - Client – Postman

Testing methodology and results.

Testing CRUD.

- GET

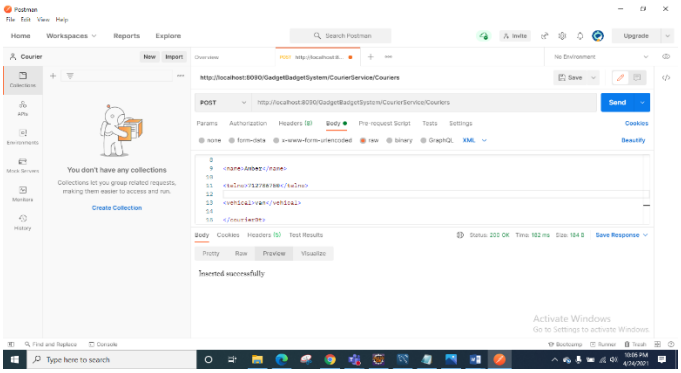


1. Start the tomcat Server in Eclipse.
2. Open Postman.
3. Select GET from the Postman.
4. Give the following link.

http://localhost:8090/GadgetBadgetSystem/CourierService/Couriers

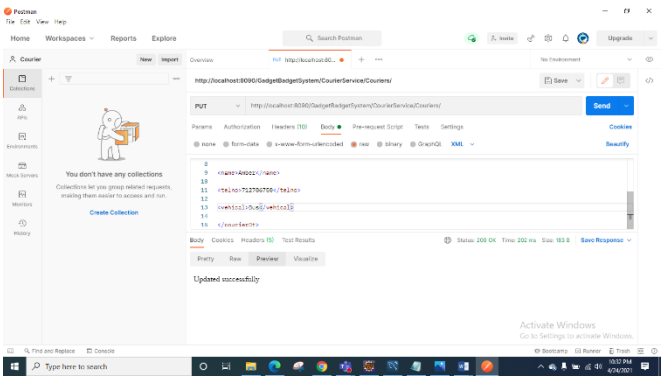
Result : HTML table with courier details will be displayed to the user.

- POST



1. Start the tomcat Server in Eclipse
2. Open Postman.
3. Select POST from the Postman.
4. Give the following XML query.
   <courierDt>
   <company>Amber company</company>
   <email>amber@gmail.com</email>
   <id>1004</id>
   <name>Amber</name>
   <telno>712786750</telno>
   <vehical>Bus</vehical>
   </courierDt>
   Result : “Inserted Successfully” message will be displayed to the user in the Postman. Data will be added to the database.

- PUT



1. Start the tomcat Server in Eclipse
2. Open Postman.
3. Select PUT from the Postman.
4. Give the following XML query.

<courierDt>

<company>Amber company</company>

<email>amber@gmail.com</email>

<id>1004</id>

<name>Amber</name>

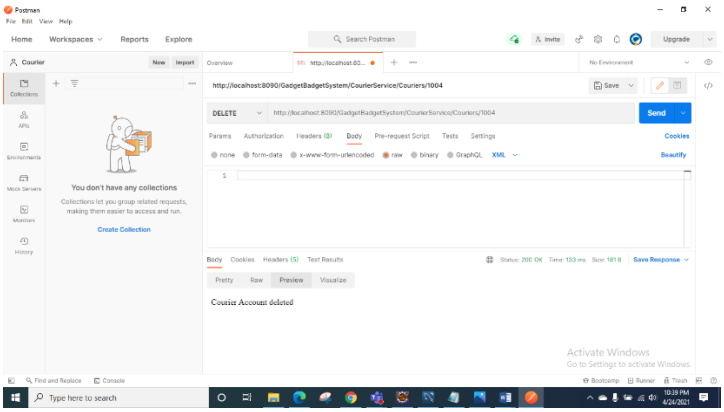
<telno>712786750</telno>

<vehical>Van</vehical>

</courierDt>

Result : “Updated Successfully” message will be displayed to the user in the Postman. Data will be Updated in the database.

- DELETE



1. Start the tomcat Server in Eclipse
2. Open Postman.
3. Select DELETE from the Postman.
4. Give the following link.

   http://localhost:8090/GadgetBadgetSystem/CourierService/Couriers/1004

   Result : “Courier Account Deleted ” message will be displayed to the user in the Postman. Data will be Deleted from the database.

System’s integration details.

Techniques

Integrated using VCS (Git Hub)

Tested

Tested the final project by merging all the branches in the repository to the mainbranch.

References :

1. https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/
2. PAF Lab Sheet 06.
3. PAF Lecture Slides