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

Assignment

Group Project

Submitted to

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the

Bachelor of Science Special Honors Degree in Information Technology

Group No: Y3S1.01.1(IT)_Group08

Table of Contents

1.Introduction

"Gadget Badget (GB)" is a company who funds inventors and let them sell their products in companies' online platform. Our task was to create the online platform by covering the whole scope of the company. We used java JAX-RS Jersey, tomcat and MySQL as our tools to build the platform. Simply when the inventor invents a product and publish in the company's online platform after getting the approvement from the company the customers can view the product and if they are interested, they can do the payment and purchase the product. The platform is simple and easy to understand.

2.Member's Details

IT Number	Name	Web Service	Description of the Web Service
IT19091426	Avinash W.V. K	Customer Management Payment Management	Insert, Update, Delete and View Customer Details Insert, Update, Delete and View Payment Details
IT19117942	Ganepola A.M.T. G	Inventor Management Product Management	Insert, Update, Delete and View Inventor Details. Insert, Update, Delete and View Products Details.
IT19126784	Wijerathne P.S. A	Funder Management	Insert, Update, Delete and View Funder Details.
IT19117010	Jayasundara J.M.N.K. B	Approvement Management	Insert, Update, Delete and View Approvement Details.

Clickable link

https://github.com/kulanaavinash/GadgetBadget.git

SE Methodology

Agile Methodology is used for the development purposes

The combination of iterative and the incremental models is referred to agile methodology. Due to its adaptiveness the complex projects are very easily to handle. This model will provide chance for the project team to break the task and put into small iteration.

Pros:

- People interaction is frequently happening here, so it very help for people communication.
- Adaption can have any time and any phase

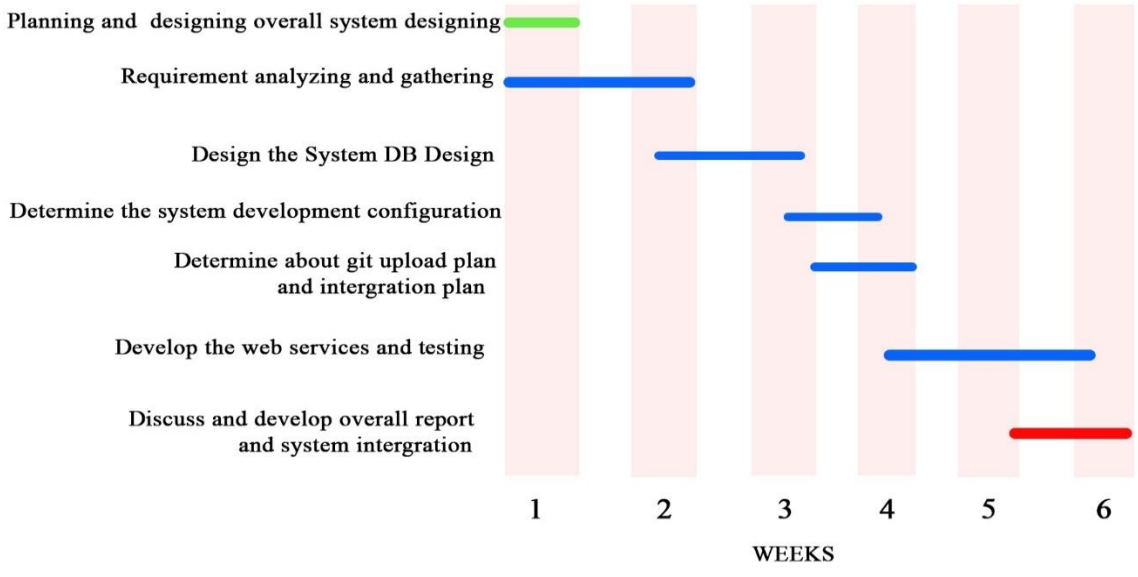
Cons:

- Lack of emphasize designing and documentation.
- Project direction can off track when the discussion gone wrong.

We used the unit testing and integrated testing methodologies for testing purposes.

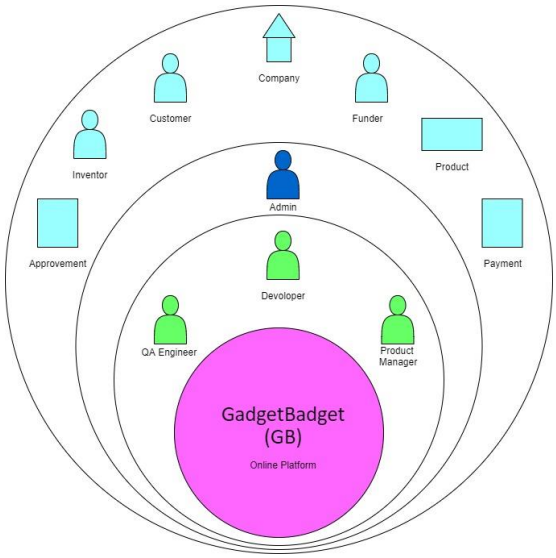
1. Unit testing
   • We use this for Unit-testing to test the services like individual component or unit. In every Unit-testing micro service tested as unit.
2. Integrate testing
   • After Unit-testing all the services are tested as unit. In integration testing the tested microservice will tested as one service and the service order will decided.

3.Time schedule (Gantt chart)



System's overall design

- Stakeholder analysis (onion diagram)



Requirement's analysis (Functional, Non-functional, Technical requirements)

Functional Requirements

1. Customer Management: Add, Update, Delete and View Customer Details
2. Payment Management: Add, Update, Delete and View Payment Details
3. Inventor Management: Add, Update, Delete and View Inventor Details
4. Product Management: Add, Update, Delete and View Product details
5. Approvement Management: Add, Update, Delete and View Approvement Details
6. Funder Management: Add, Update, Delete and View Funder Management Details

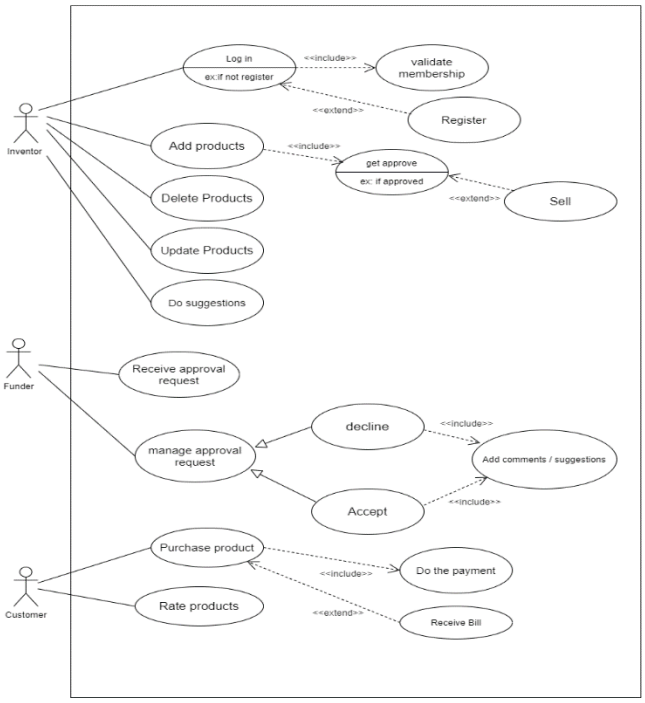
Nonfunctional Requirements

1. Quality, Efficiency, Maintainability, Reliability, Privacy, Accessibility, Compatibility, Extensibility

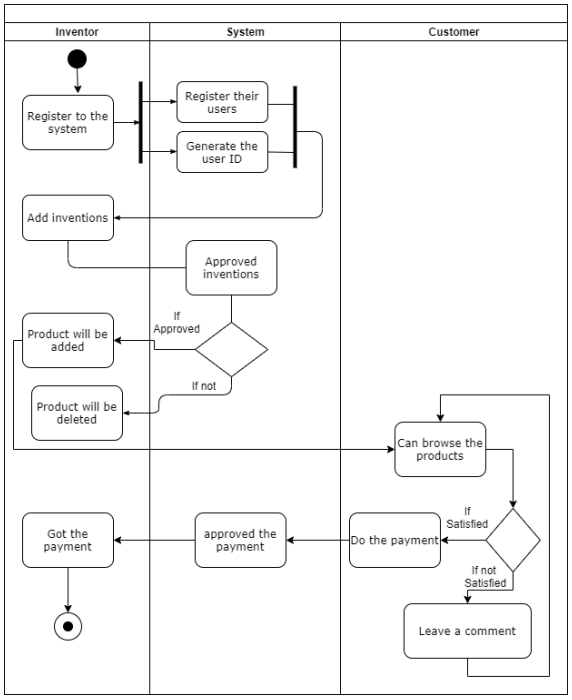
Technical Requirements

The backend is developed using Restful web services **JAX-RS, Jersey, Java on Tomcat** via Eclipse IDE in a windows operating system. My SQL is used in creating database and Postman is an API Client which uses for testing.

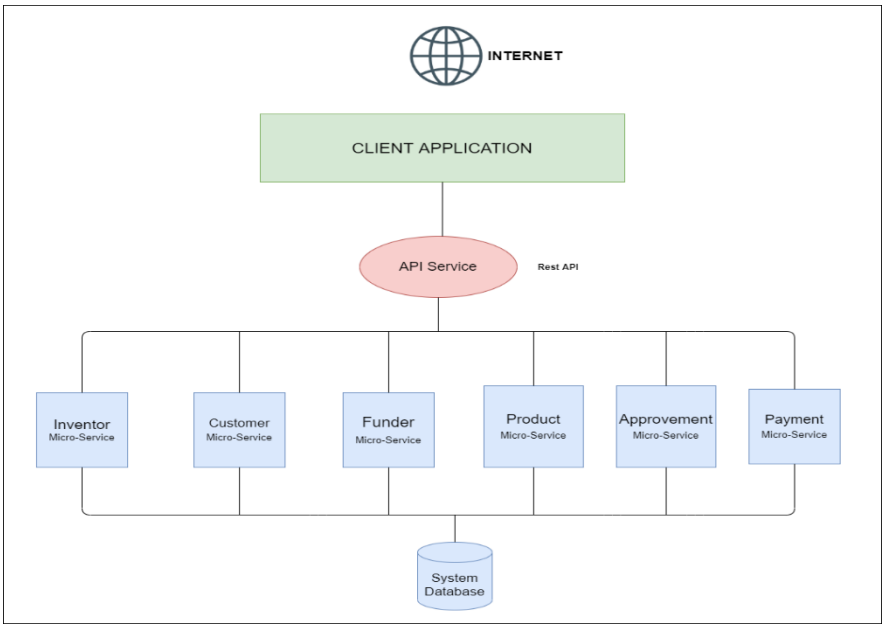
Requirements modelling (Use Case Diagram)



Activity Diagram

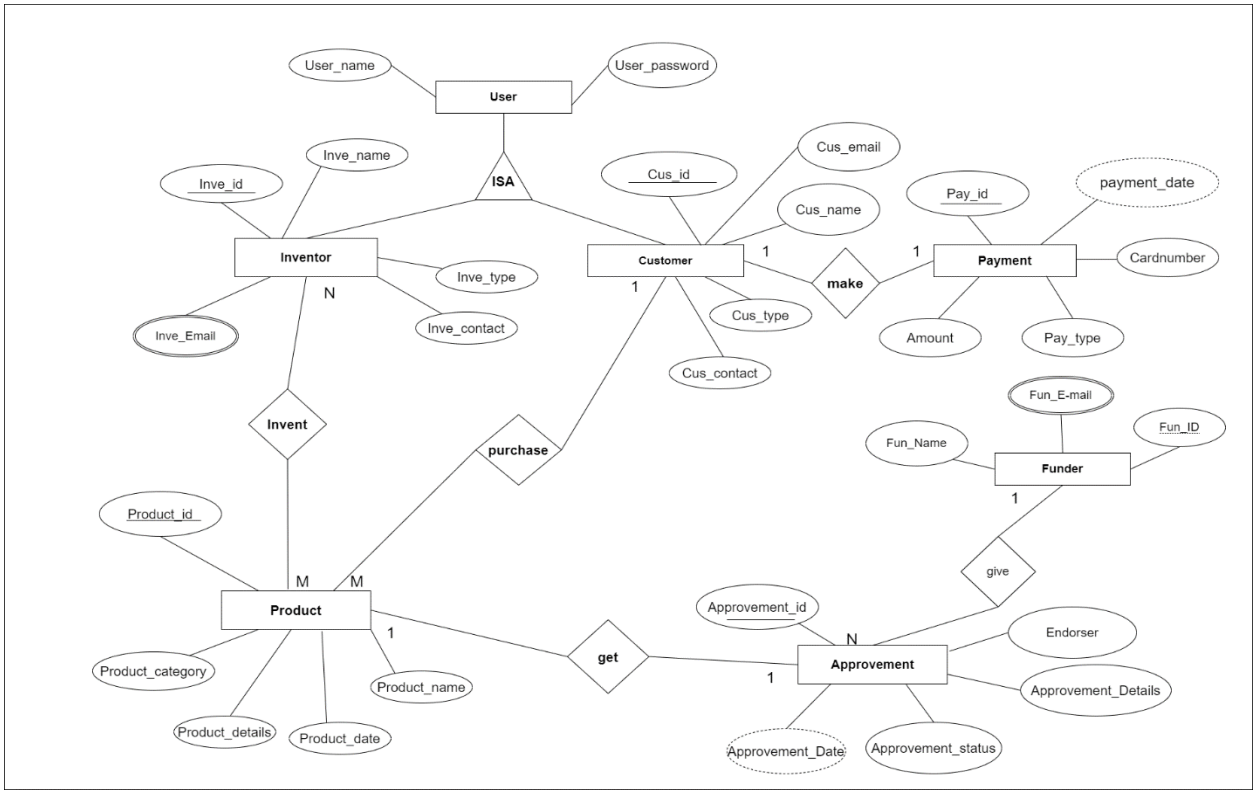


Overall Architecture

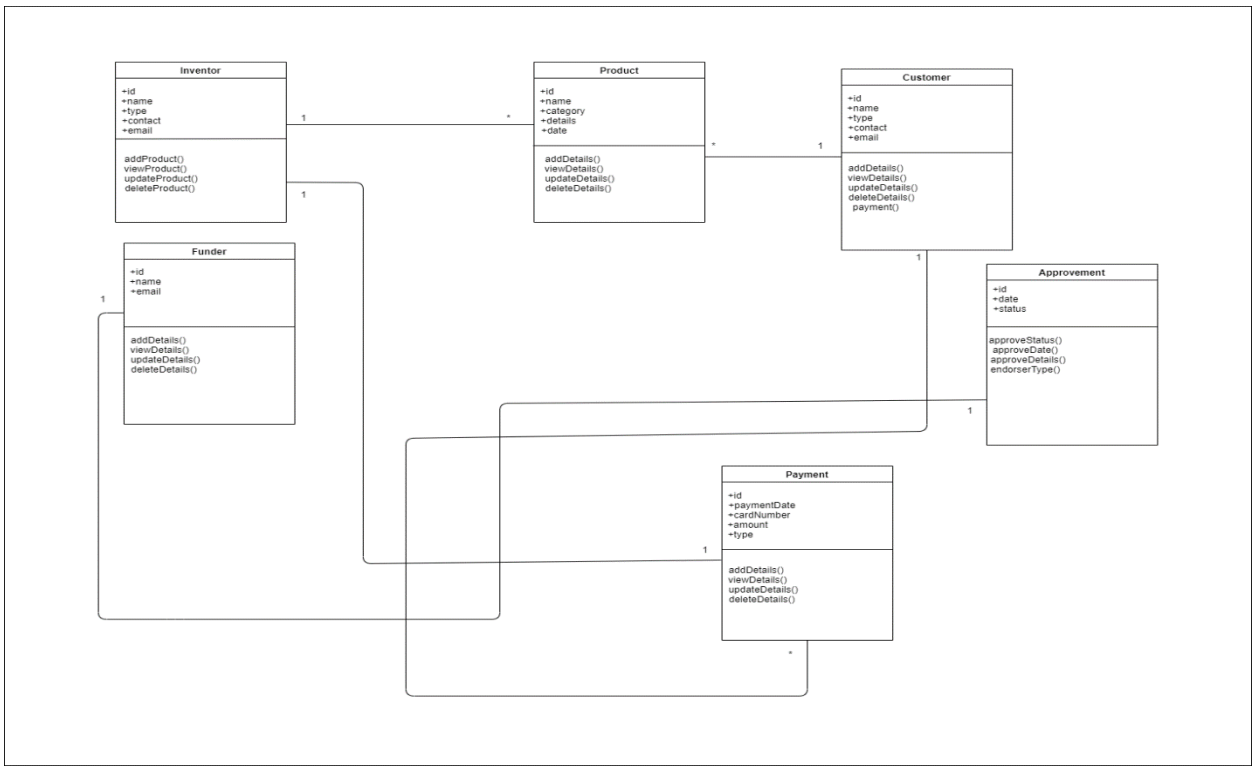


Our task is to create an online platform for a company which provide funds for innovative projects which were done by young inventors. Also, the company will help the inventors to sell their products via the company's online platform. We have allowed the customers to connect with the inventors so inventors can sell the inventions easily through our platform.

ER diagram



Class diagram



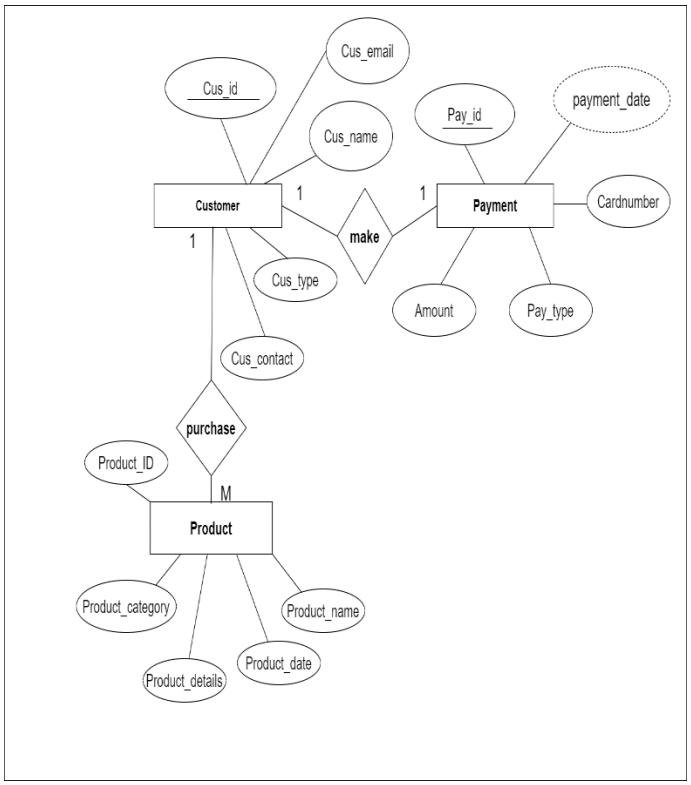
IT19091426 (Avinash W.V.K) –URL: http://localhost:8090/GadgetBadget/CustomerService/Customers

http://localhost:8090/GadgetBadget/PaymentService/Payments

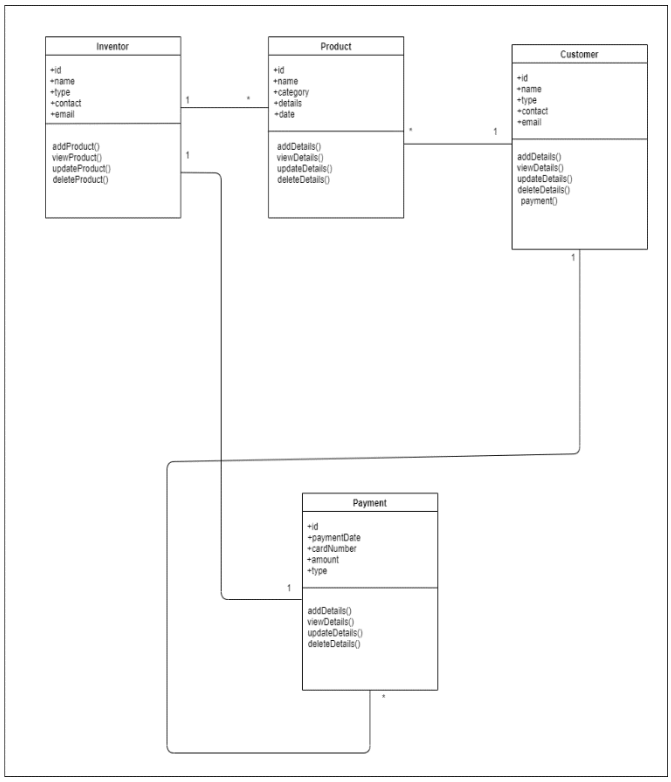
Customer Management and Payment Management

Here the full scope of the customer and Payment management is completed. The respected parties can perform their function successfully though this platform. In customer management the customers can insert, update, delete and view customer details and in payment management the parties can insert, update, delete and view the payment details.

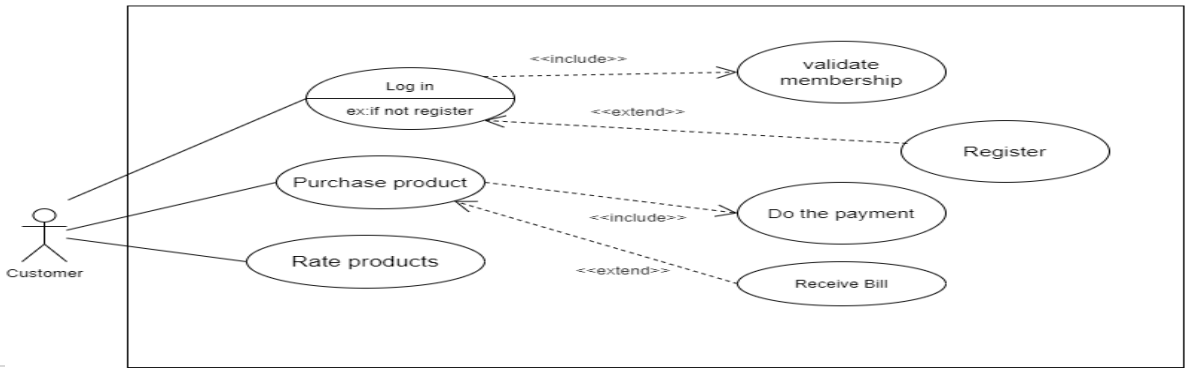
Er Diagram



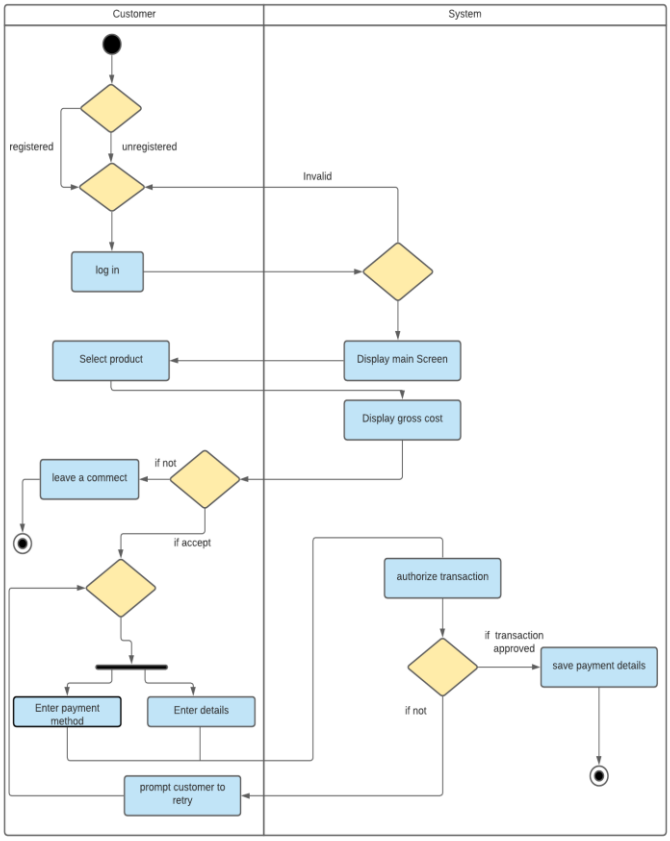
Class Diagram



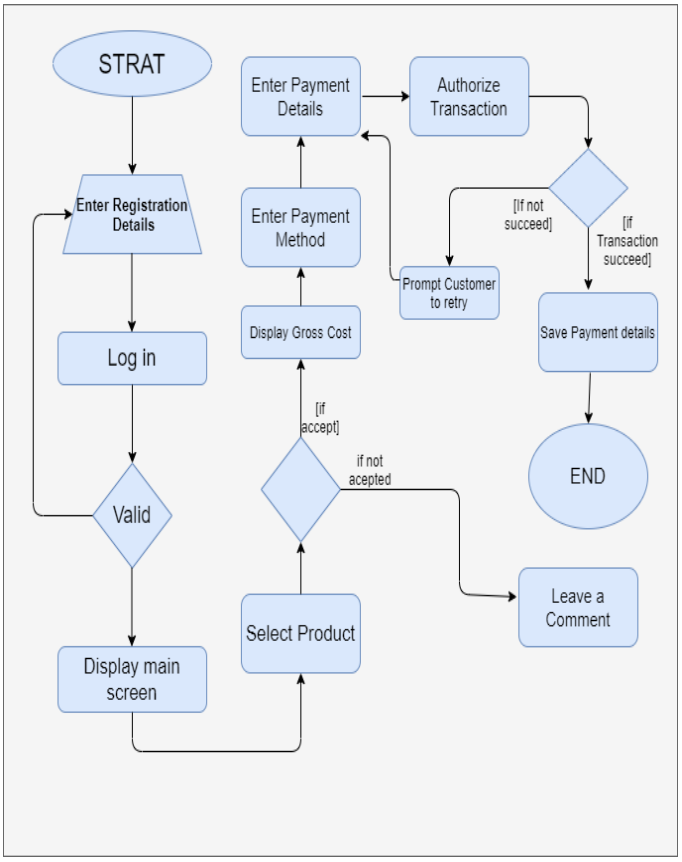
Usecase Diagram



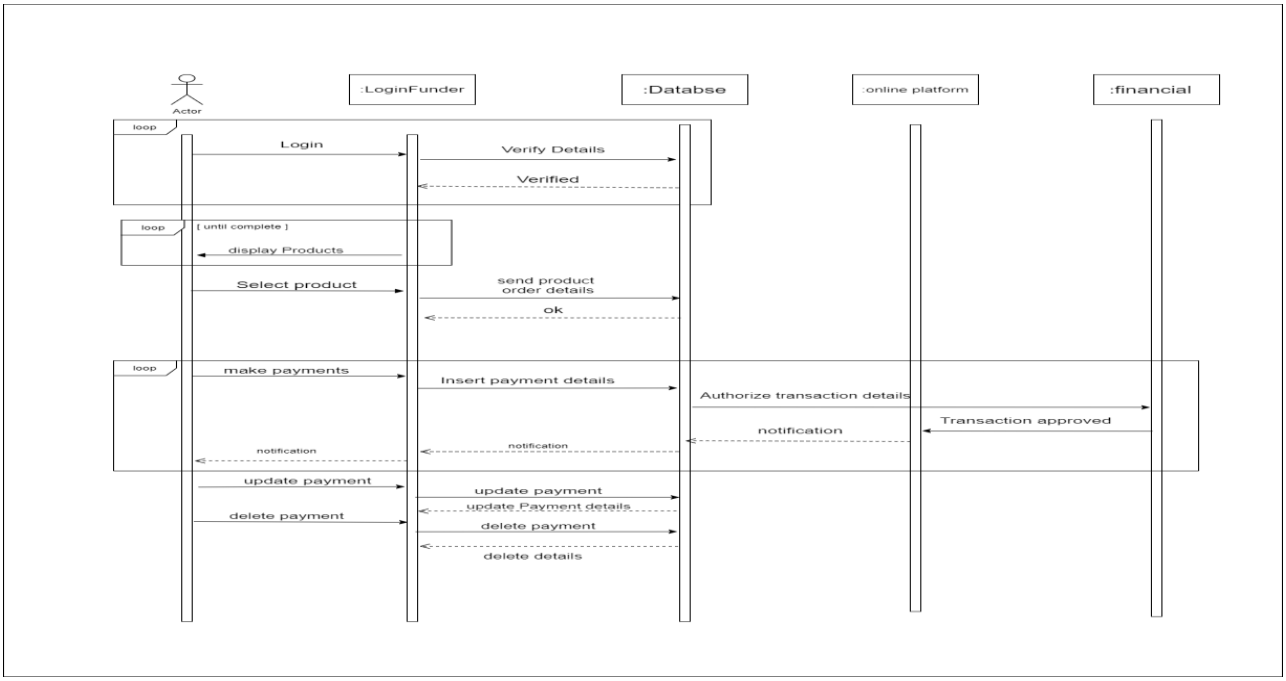
Activity Diagram



Flow Chart

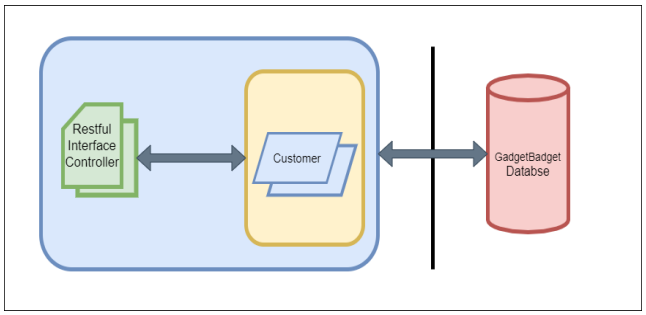


Sequence Diagram

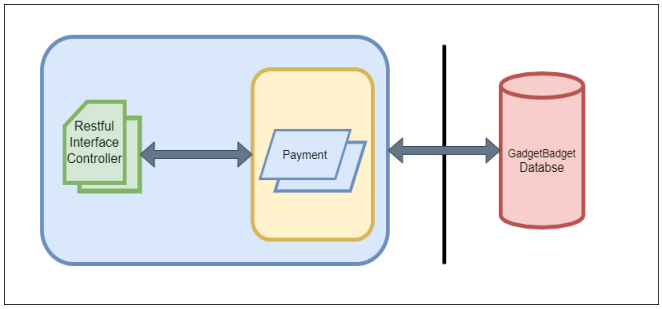


Internal workflow

Customer:



Payment:



API Design Relational

Customer management and payment management is one of the main parts in this system. The Customers can purchase any product which are added by the inventors and with the systems payment management system the customers can easily do their payment. The system is user friendly and low complicated.

- Customers can insert, delete, update and view the customer details
- The payment details also can be inserted, viewed, deleted and updated by the respected party.

Service Development

- Technologies Used: Java - JAX-RS (Jersey) on Tomcat
- IDE : Eclipse
- Database : MySQL
- Testing : POSTMAN

API for get all the Customer details in the database (GET Request)

URL: - http://localhost:8090/GadgetBadget/CustomerService/Customers

Request : - { }

Response : - {CustomerID : "Auto generated integer value"}
{CustomerName =:"avinash"}
{CustomerEmail :avinash @gmail.com }
{CustomerType : Loyal customer"}
{CustomerContact : "0715534356"}

API for insert a new Customers to the database (POST Request)

`*URL:* - http://localhost:8090/GadgetBadget/CustomerService/Customers

Request : - {CustomerID : "Auto generated integer value"}
{CustomerName =:"kulana avinash"}
{CustomerEmail : kulanaavinash@gmail.com }
{CustomerType : Need-based customer"}
{CustomerContact : "0713333333"}

Response :- { Result = "Inserted successfully" }
{ CustomerID = "Auto generated integer value"
{ Error= "Error while inserting"}

API for update a Customers which is existing in database (PUT Request)

`*URL:* - http://localhost:8090/GadgetBadget/CustomerService/Customers

Request : - {CustomerID : "Auto generated integer value"}
{CustomerName =:"kulana avinash"}
{CustomerEmail : kulanaavinash@gmail.com }
{CustomerType : Need-based customer"}
{CustomerContact : "0713333333"}

Response :- { Result = "Updated successfully" }
{ CustomerID = "Auto generated integer value"
{ Error= "Error while Updating"}

API for delete a Customers which is existing in database (DELETE Request)

URL: - http://localhost:8090/GadgetBadget/CustomerService/Customers

Request :- { CustomerID = " Selected Customer ID from the service "}

Response :- { Result = "Deleted successfully" }
{Error = "Error while deleting the hospital" }

API for get all the Payment details in the database (GET Request)

URL: - http://localhost:8090/GadgetBadget/PaymentService/Payments

Request : - { }

Response : - {PaymentID = "Auto generated integer value"}

{ PaymentDate" = "12.09.2017"}

{CardNumber = "12334335454664"}

{Amount = "130000.00 "}

{PaymentType = "Cash”}

API for insert a new Payment to the database (POST Request)

URL: - http://localhost:8090/GadgetBadget/PaymentService/Payments

Request : - { PaymentID: "Auto generated integer value"}
{ PaymentDate" = "12.09.2017"}
{CardNumber = "43636754744343546"}
{Amount = "12000.00 "}
{PaymentType = "Cash”}

Response :- { Result = "Inserted successfully" }
{ PaymentID = "Auto generated integer value" }
{ Error= "Error while inserting"}

API for update a Payment which is existing in database (PUT Request)

URL: - http://localhost:8090/GadgetBadget/PaymentService/Payments

Request : - { PaymentID: "Auto generated integer value"}
{ PaymentDate" = "11.08.2019"}
{CardNumber = "54564365745734653"}
{Amount = "11000.00 "}
{PaymentType = "Check”}

Response :- { Result = "Updated successfully" }
{ PaymentID = "Auto generated integer value" }
{Error= "Error while Updating "}

API for delete a Payment which is existing in database (DELETE Request)

URL: - http://localhost:8090/GadgetBadget/PaymentService/Payments

Request :- { PaymentID = " Selected PaymentID from the service "}

Response :- { Result = "Deleted successfully" }

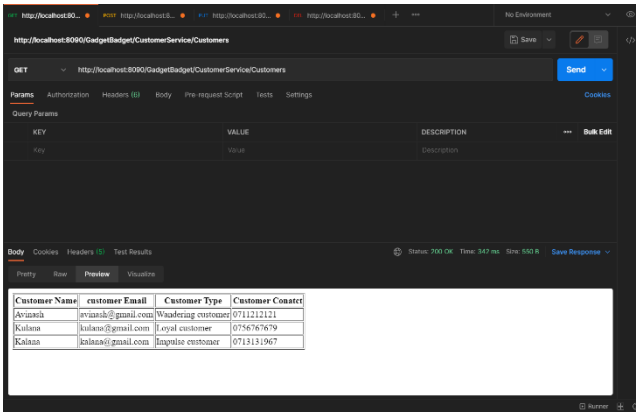
{Error = "Error while deleting the Payment Details" }

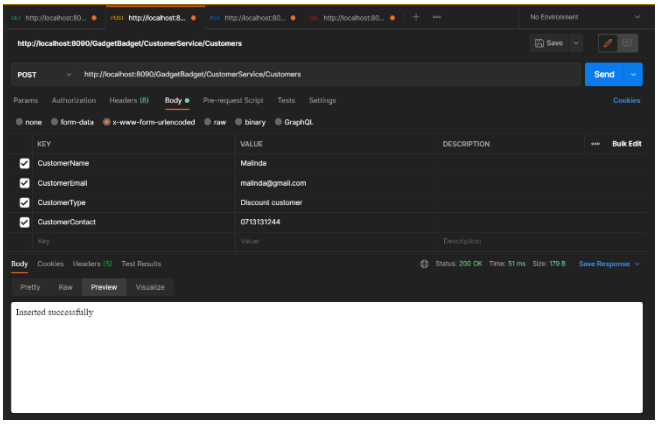
Test Cases

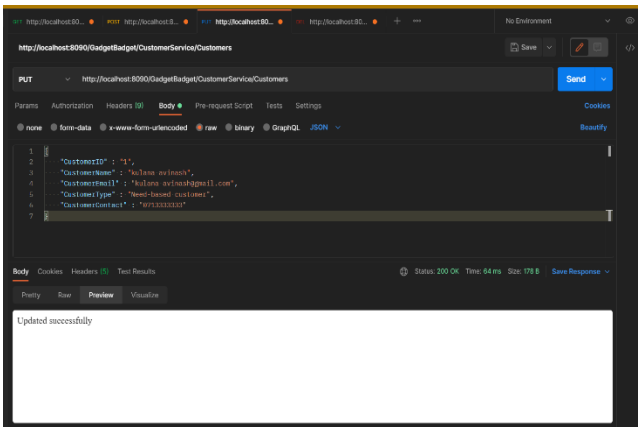
TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
1	Add Customer Details	{CustomerName :"avinash"} {CustomerEmail :avinash @gmail.com } {CustomerType :Loyalcustomer"} {CustomerContact:"0715534356"}	New Customer details are added to the database. Show message as "Inserted successfully".	New Customer details are added to the database. Show message as "Inserted successfully".	**PASS**
2	Update Customer Details	{ "CustomerID" : "1", "CustomerName" : "kulana avinash", "CustomerEmail" : "kulana avinash@gmail.com", "CustomerType" : "Need-based customer", "CustomerContact" : "0713333333" }	Customer details are updated according to relevant input Customer details. Show message as "Updated successfully".	Customer details are updated according to relevant input Customer details. Show message as "Updated successfully".	**PASS**
3	Delete Customer Details	<customerData> <CustomerID>1</CustomerID> </customerData>	Show message as “Deleted Successfully”	Customer Details are deleted successfully. Show message as “Deleted Successfully”	**PASS**

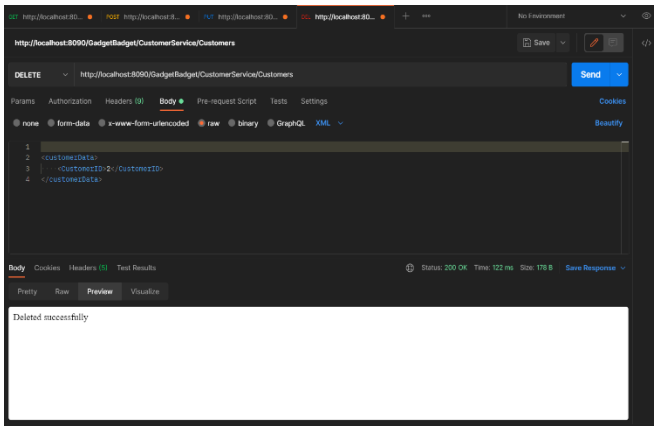
TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
1	Add Payment Details	{PaymentDate" = "12.09.2017"} {CardNumber = "12334335454664"} {Amount = "130000.00 "} {PaymentType = "Cash”}	New Payment details are added to the database. Show message as "Inserted successfully".	New Payment details are added to the database. Show message as "Inserted successfully".	**PASS**
2	Update Payment Details	"PaymentID" : "1", "PaymentDate" : "12.09.2017", "CardNumber" :"12334335454664", "Amount”: "130000.00 ", "PaymentType”: "Cash"	Payment details are updated according to relevant input Payment details. Show message as "Updated successfully".	Customer details are updated according to relevant input Payment details. Show message as "Updated successfully".	**PASS**
3	Delete Payment Details	<paymentData> <PaymentID>1</PaymentID> </paymentData>	Show message as “Deleted Successfully”	Payments Details are deleted successfully. Show message as “Deleted Successfully”	**PASS**

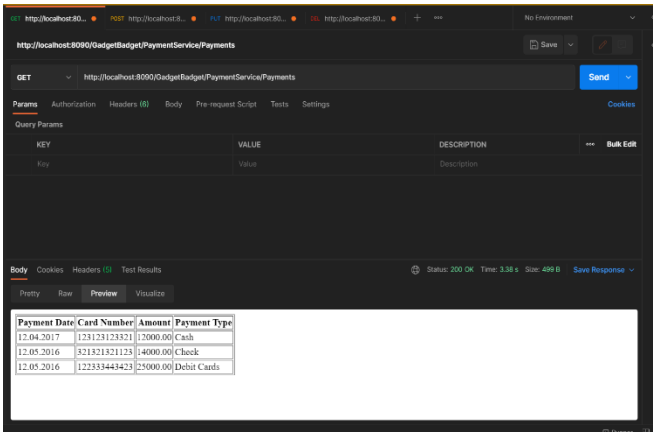
Testing

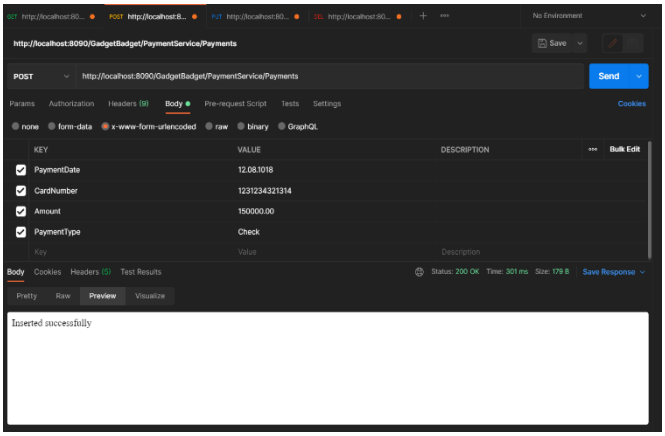


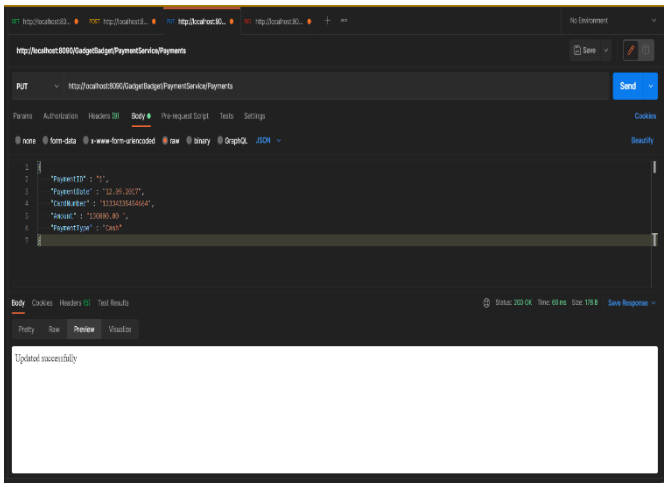


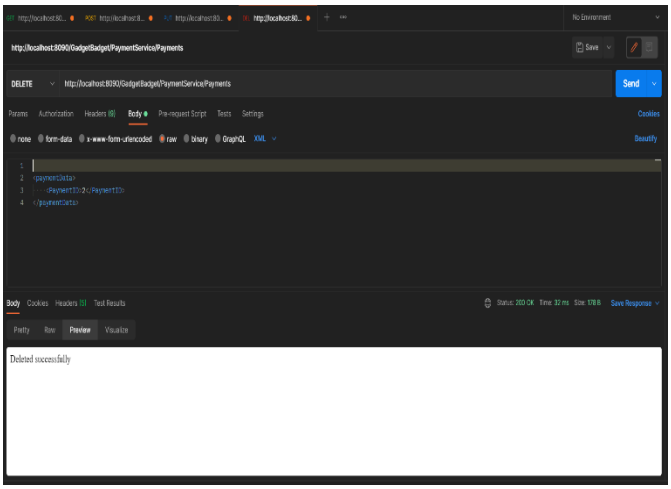

http://localhost:8090/GadgetBadget/CustomerService/Customers
PUT
Send
Updated successfully


http://localhost:8090/GadgetBadget/CustomerService/Customers
DELETE
Send
Deleted successfully


http://localhost:8090/GadgetBadget/PaymentService/Payments
GET
Send
Payment Date
Card Number
Amount
Payment Type
12.04.2017
123123123321
12000.00
Cash
12.05.2016
321321321123
14000.00
Check
12.05.2016
122333443423
25000.00
Debit Cards


http://localhost:8090/GadgetBadget/PaymentService/Payments
POST
Send
PaymentDate
12.08.1018
CardNumber
123123432321314
Amount
150000.00
PaymentType
Check
Inserted successfully


http://localhost:8090/GadgetBadget/PaymentService/Payments
PUT
Send
Updated successfully


http://localhost:8090/GadgetBadget/PaymentService/Payments
DELETE
Send
Deleted successfully

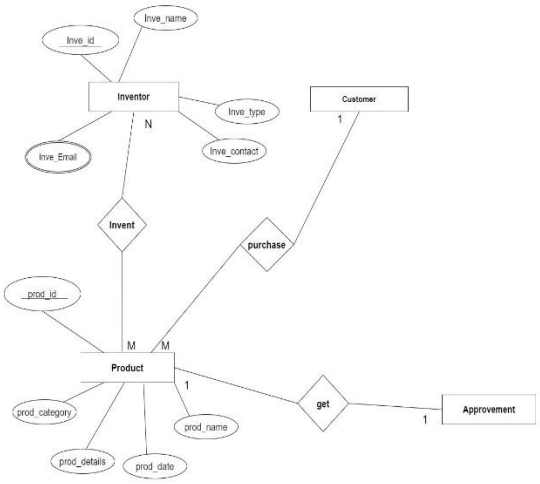
IT19117942 (Ganepola A.M.T.G)
URL:http://localhost:8090/GadgetBadget/InventorService/Inventors

http://localhost:8090/GadgetBadget/ProductService/Products

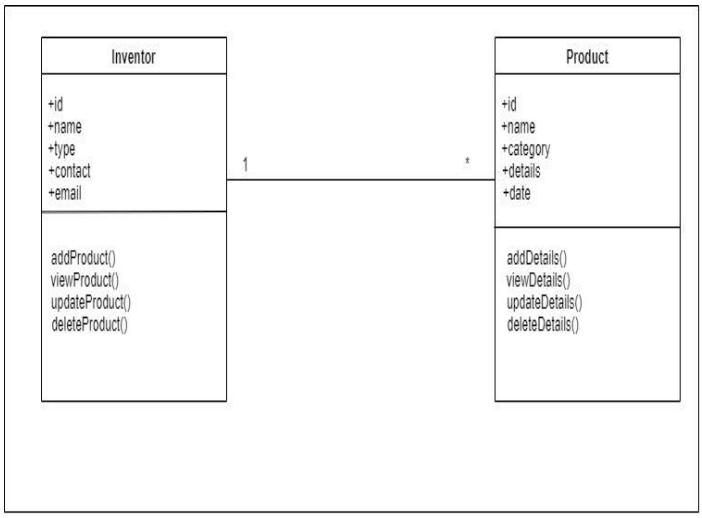
Inventor Management and Product Management

Inventors are the group of users who invent the products, so in this the full scope of inventor management and product management is completed. The inventors can insert, update, delete and view the inventor details and, they can insert, update, delete and view the respected product details.

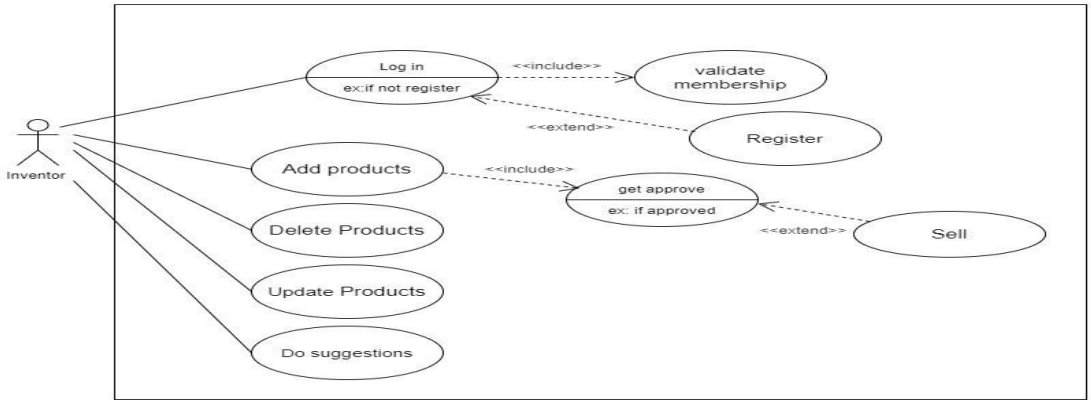
Er Diagram



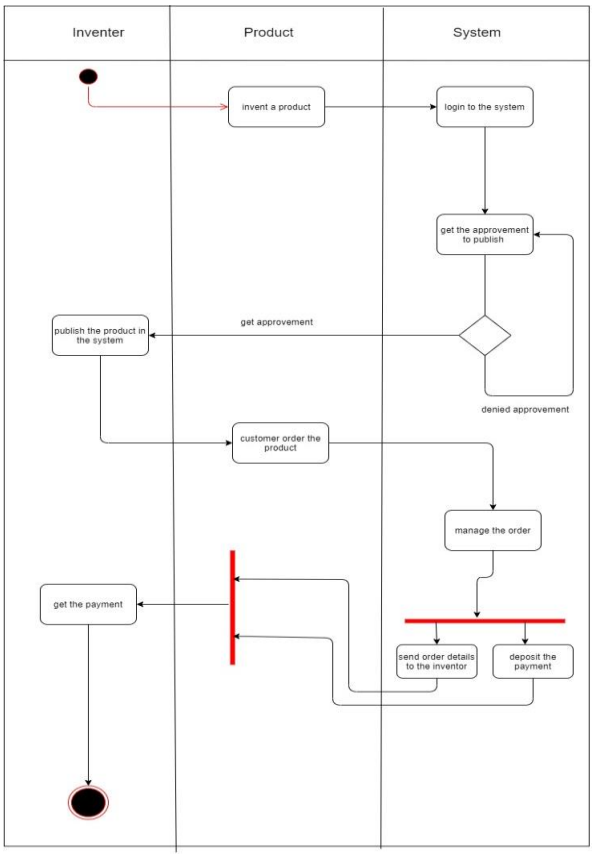
Class Diagram



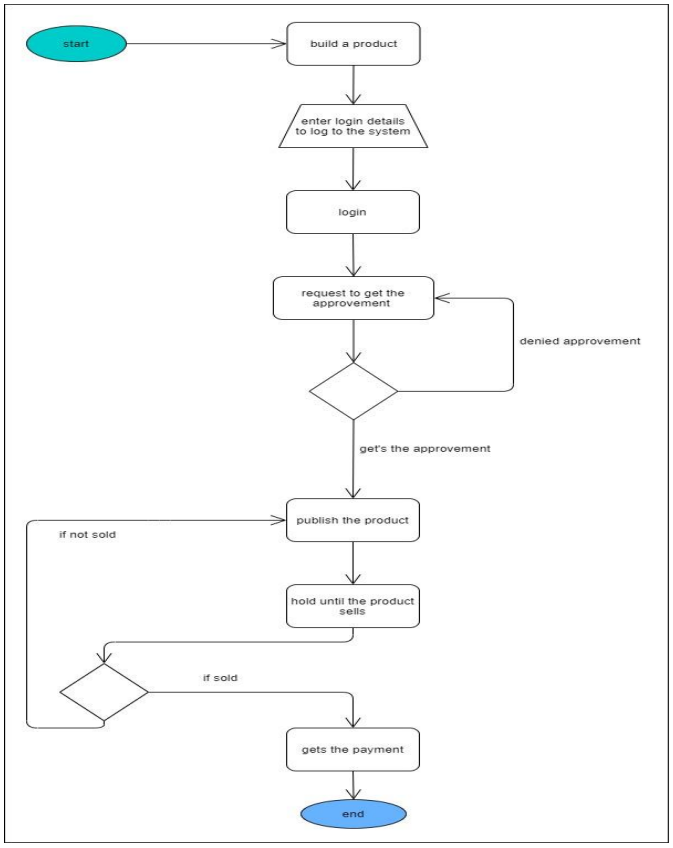
Usecase Diagram



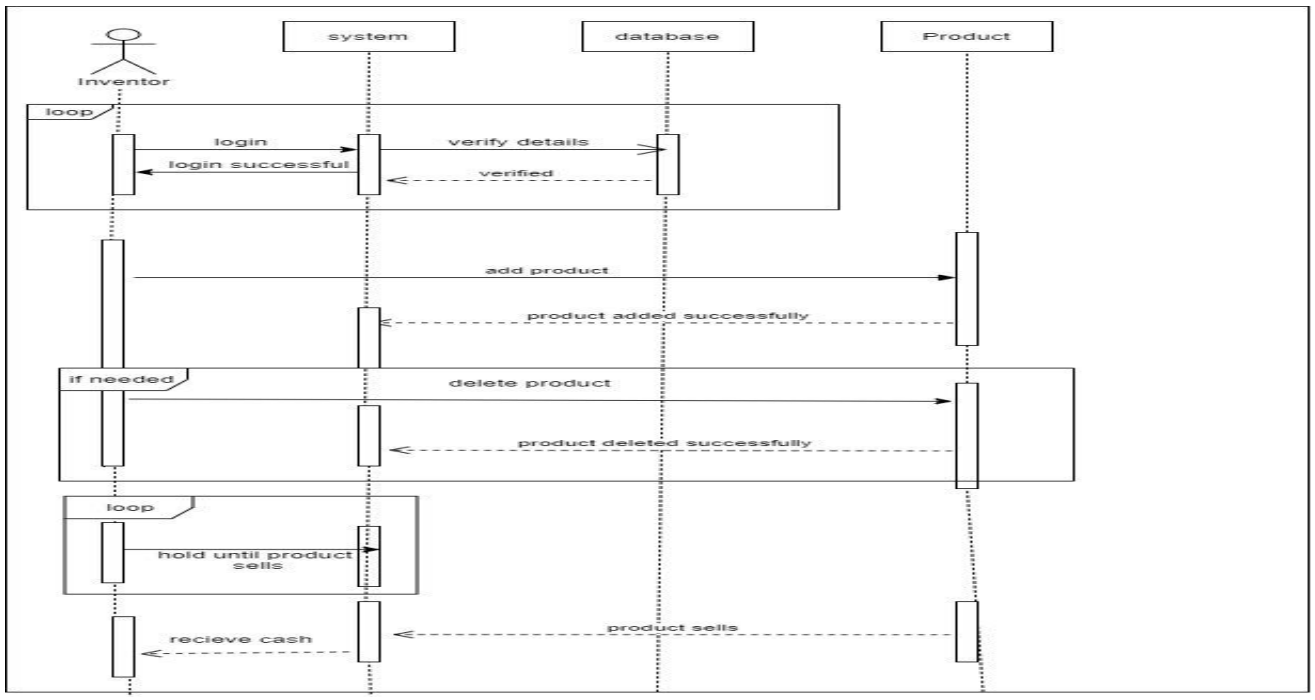
Activity Diagram



Flow Chart

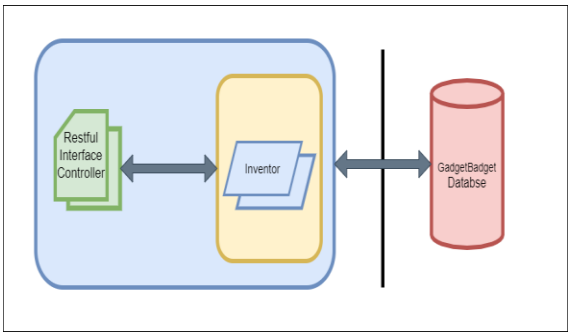


Sequence Diagram

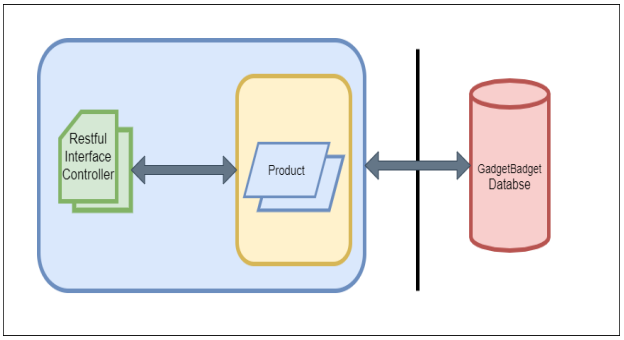


Internal workflow

Inventor:



Product:



API Design Relational

In inventor and product management the inventors can add their developed inventions as products to the system so the customers can purchase them. The inventors can easily register to the system and get the approvement from the company in order to add their products to the system. The inventor and payment management systems are user friendly and low complicated.

- Inventors can insert, delete, update and view the inventor details.
- In the product management the inventors can add, view, delete and update the product details easily.

Service Development

- Technologies Used: Java - JAX-RS (Jersey) on Tomcat
- IDE : Eclipse
- Database : MySQL
- Testing : POSTMAN

API for get all the Inventor details in the database (GET Request)

URL: - http://localhost:8090/GadgetBadget/InventorService/Inventors

Request : - { }

Response : - {InventorID : "Auto generated integer value"}
{InventorName =:"Malinda"}
{InventorEmail :malinda@gmail.com }
{InventorContact : 0714444464"}
{InventorType: "School Leaver"}

API for insert a new Inventor to the database (POST Request)

`URL: - http://localhost:8090/GadgetBadget/InventorService/Inventors

Request : - {InventorID : "Auto generated integer value"}
{InventorName :"Malinda Ganepola"}
{InventorEmail: malindaganepola@gmail.com }
{InventorContact: “0965434665"}
{InventorType: "School Leaver"}

Response :- { Result = "Inserted successfully" }
{ InventorID = "Auto generated integer value"
{ Error= "Error while inserting"}

API for update Inventor which is existing in database (PUT Request)

`URL: - http://localhost:8090/GadgetBadget/InventorService/Inventors

Request : - { InventorID: "Auto generated integer value"}
{ InventorName =:"malinda "}
{ InventorEmail: malinda@gmail.com }
{ InventorContact: 0713333333"}
{ InventorType: "School Leaver"}

Response :- { Result = "Updated successfully" }
{ InventorID = "Auto generated integer value"
{ Error= "Error while Updating"}

API for delete a Inventor which is existing in database (DELETE Request)

URL: - http://localhost:8090/GadgetBadget/InventorService/Inventors

Request :- { InventorID = " Selected InventorID from the service "}

Response :- { Result = "Deleted successfully" }
{Error = "Error while deleting the Inventor details" }

API for get all the Products details in the database (GET Request)

URL: - http://localhost:8090/GadgetBadget/ProductService/Products

Request : - { }

Response : - {ProductID : "Auto generated integer value"}
{ProductName =:"Driverless car"}
{ProductDate : “12.11.2016”}
{ProductDetails : “Tesla is among many tech companies "}
{ProducType: "Other"}

API for insert a new Products to the database (POST Request)

`*URL: -* http://localhost:8090/GadgetBadget/ProductService/Products

Request : - {ProductID : "Auto generated integer value"}
{ProductName :"Driverless car"}
{ProductDate : “12.11.2016”}
{ProductDetails : “Tesla is among many tech companies "}
{ProductType: "Other"}

Response :- { Result = "Inserted successfully" }
{ ProductID = "Auto generated integer value"
{ Error= "Error while inserting"}

API for update Products which is existing in database (PUT Request)

`*URL: -* http://localhost:8090/GadgetBadget/ProductService/Products

Request : - { ProductID: "Auto generated integer value"}
{ ProductName =:"Hidden Camera "}
{ ProductDate: “12.08.2019”}
{ ProductDetails: “Vision care hearing ai solution"}
{ ProductType: "Medical"}

Response :- { Result = "Updated successfully" }
{ ProductID = "Auto generated integer value" }
{ Error= "Error while Updating"}

API for delete a Products which is existing in database (DELETE Request)

URL: - http://localhost:8090/GadgetBadget/ProductService/Products

Request :- { ProductID = " Selected ProductID from the service "}

Response :- { Result = "Deleted successfully" }

{Error = "Error while deleting the Product details”}

Test Cases

TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
1	Add Inventor Details	{InventorName =:"Malinda"} {InventorEmail :malinda@gmail.com } {InventorContact : 0714444464"} {InventorType: "School Leaver"}	New Inventor details are added to the database. Show message as "Inserted successfully".	New Inventor details are added to the database. Show message as "Inserted successfully".	**PASS**

2	Update Inventor Details	{
		"InventorID" : "2",
		"InventorName" : "malindaganepola",
		"InventorEmail" : " malindaganepola @gmail.com",
		"InventorContact" : "0345566771 ",
		"InventorType" : "School Leaver"
		}
			Inventor details are updated according to relevant input Inventor details. Show message as "Updated successfully".	Inventor details are updated according to relevant input Inventor details. Show message as "Updated successfully".	**PASS**

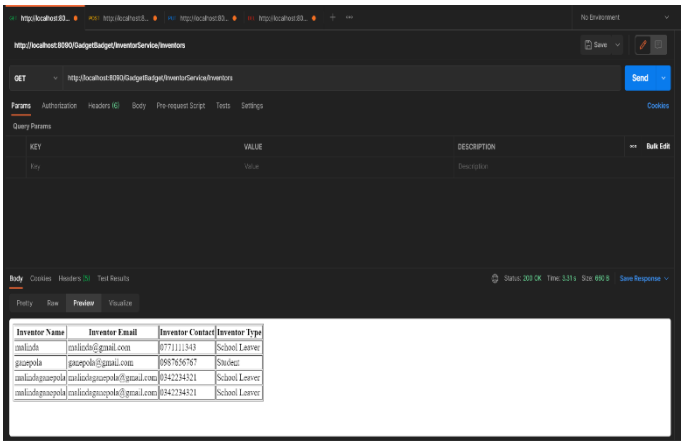
3	Delete Inventor Details	<inventorData> <InventorID>2</InventorID> </inventorData>	Show message as “Deleted Successfully”	Inventor Details are deleted successfully. Show message as “Deleted Successfully”	**PASS**

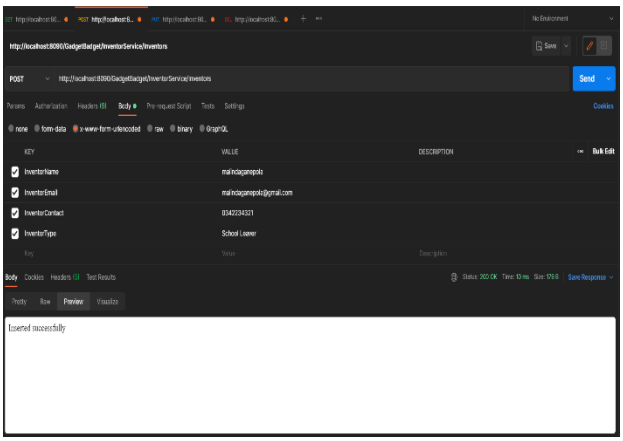
TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
1	Add Products Details	{ProductName =:"Driverless car"} {ProductDate : “12.11.2016”} {ProductDetails : “Tesla is among many tech companies "} {ProducType: "Other"}	New Products details are added to the database. Show message as "Inserted successfully".	New Products details are added to the database. Show message as "Inserted successfully".	**PASS**

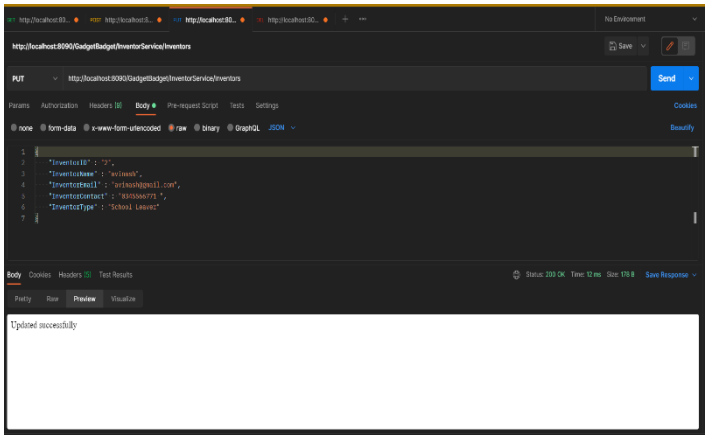
2	Update Products Details	{
		"ProductID" : "1",
		"ProductName" : "Driverless car",
		"ProductDate" : "12.11.2016",
		"ProductDetails" : "Tesla is among many tech ",}
		"ProductCategory" : "Vehicle"
		}
			Products details are updated according to relevant input Products details. Show message as "Updated successfully".	Products details are updated according to relevant input Products details. Show message as "Updated successfully".	**PASS**

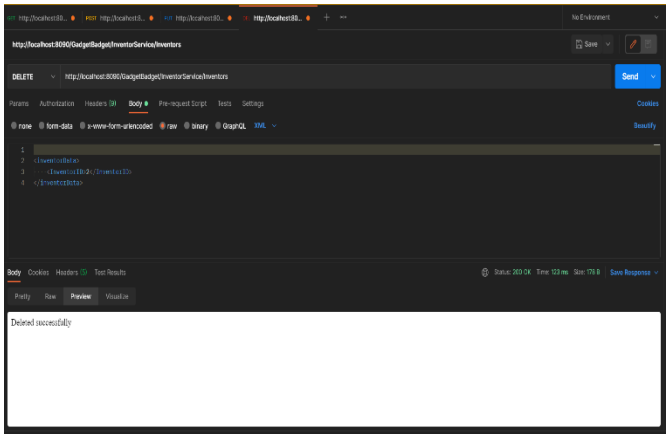
3	Delete Products Details	<inventorData> <InventorID>2</InventorID> </inventorData>	Show message as “Deleted Successfully”	Products Details are deleted successfully. Show message as “Deleted Successfully”	**PASS**

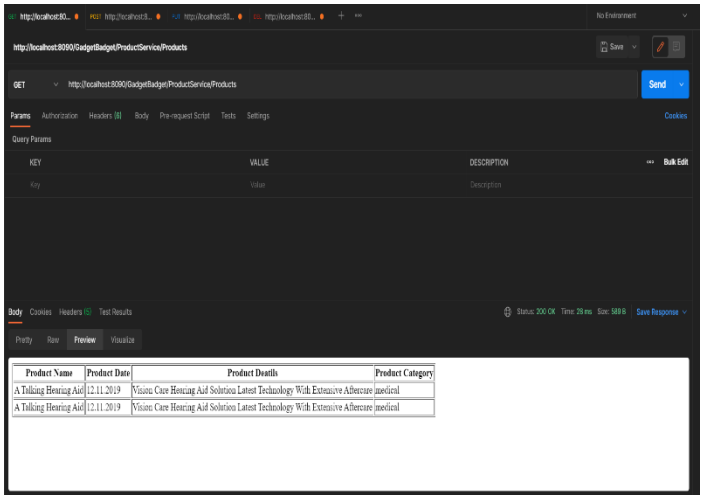
Testing

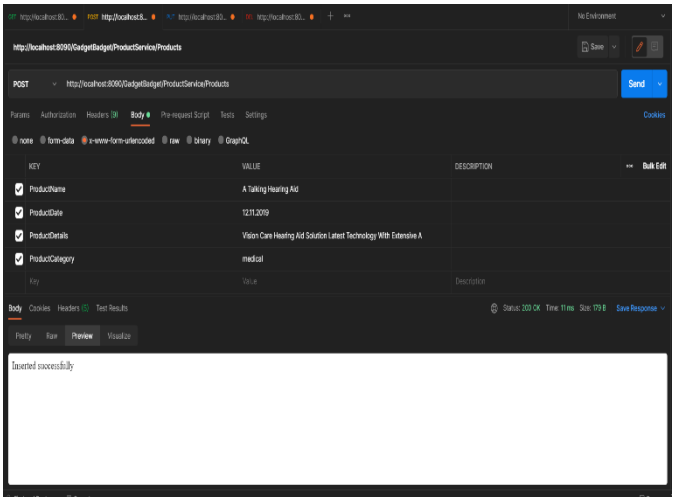


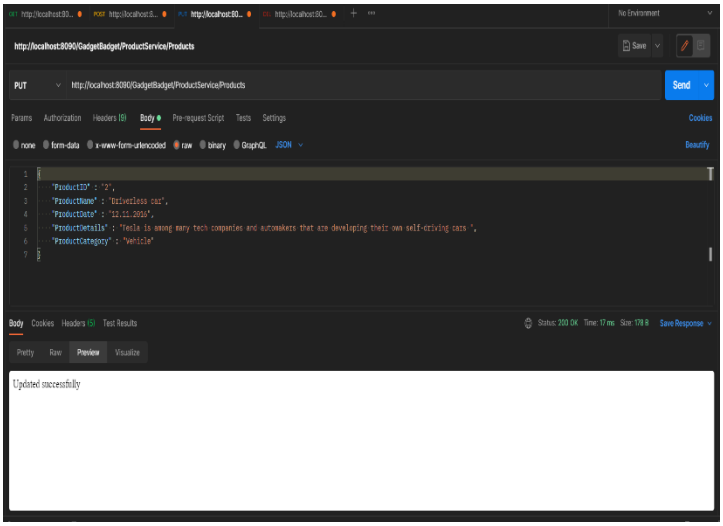


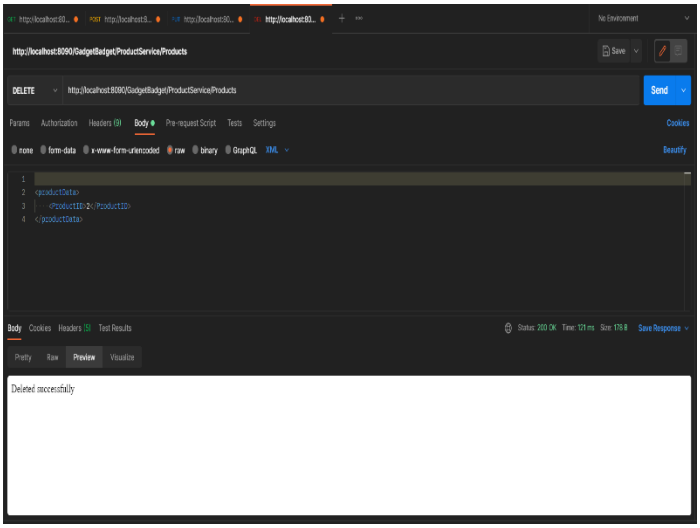

http://localhost:8090/GadgetBadget/InventorService/Inventors
PUT
Send
Updated successfully


http://localhost:8090/GadgetBadget/InventorService/Inventors
DELETE
Send
Deleted successfully


http://localhost:8090/GadgetBadget/ProductService/Products
GET
Send
Product Name
Product Date
Product Deatils
Product Category
A Talking Hearing Aid
12.11.2019
Vision Care Hearing Aid Solution Latest Technology With Extensive Aftercare
medical


http://localhost:8090/GadgetBadget/ProductService/Products
POST
Send
ProductName
A Talking Hearing Aid
ProductDate
12.11.2019
ProductDetails
Vision Care Hearing Aid Solution Latest Technology With Extensive A
ProductCategory
medical
Inserted successfully


http://localhost:8090/GadgetBadget/ProductService/Products
PUT
Send
Updated successfully

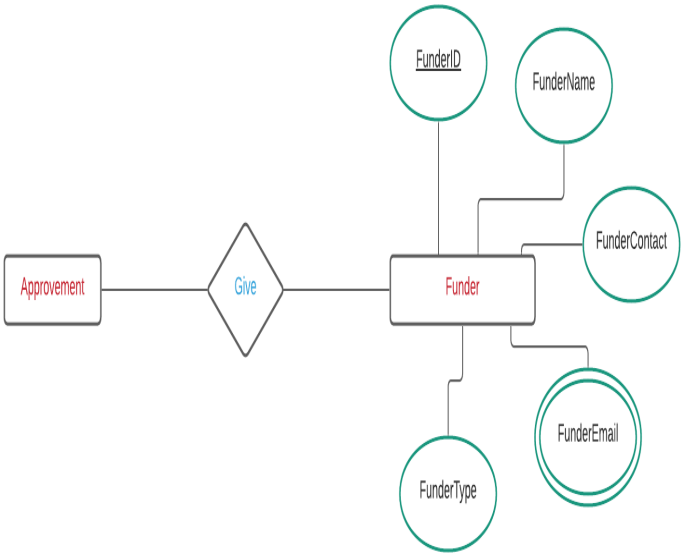

http://localhost:8090/GadgetBadget/ProductService/Products
DELETE
Send
Deleted successfully

IT19126784 (P.S.A.Wijerathne) –URL: http://localhost:8090/GadgetBadget/FunderService/Funders

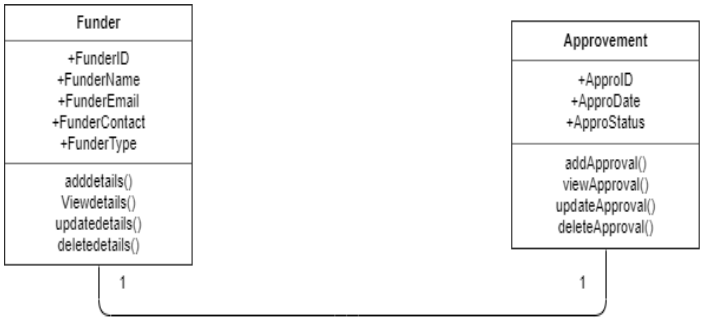
Funder Management

Inhere the full scope of funder management is completed where the funding part of the company can be successfully completed through this management system. In funder management the details can be inserted, updated, deleted and viewed.

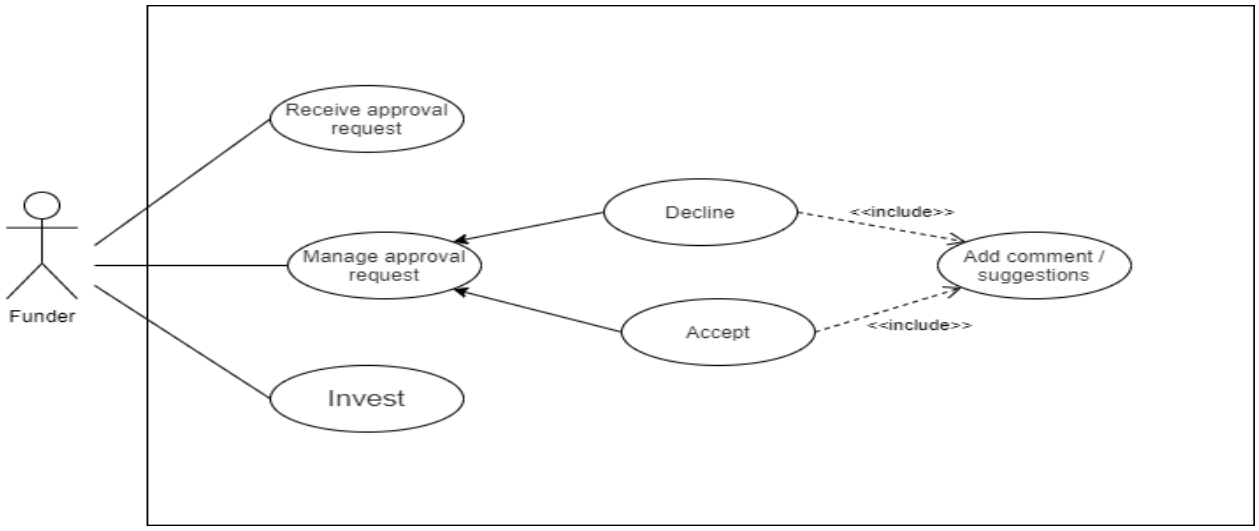
Er Diagram



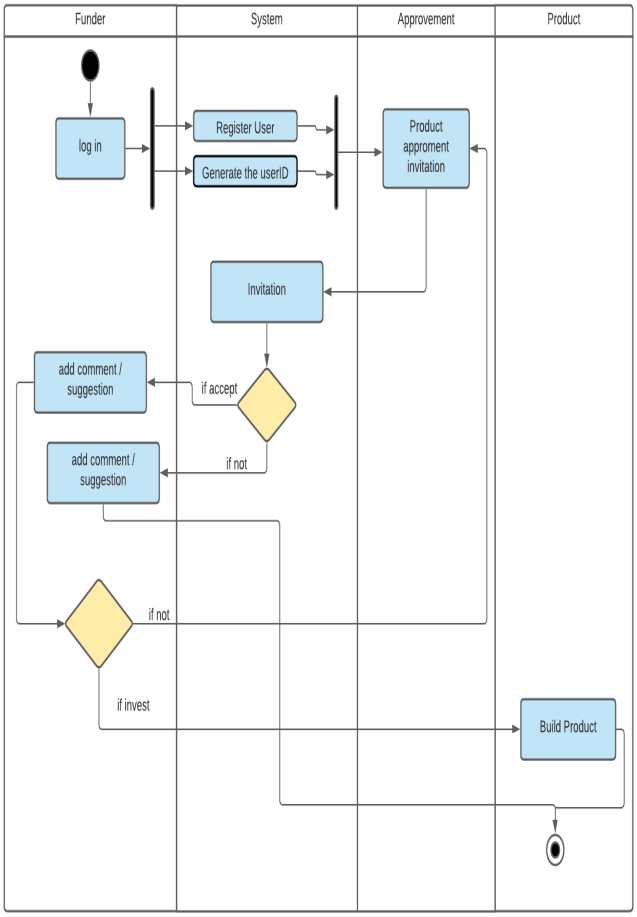
Class Diagram



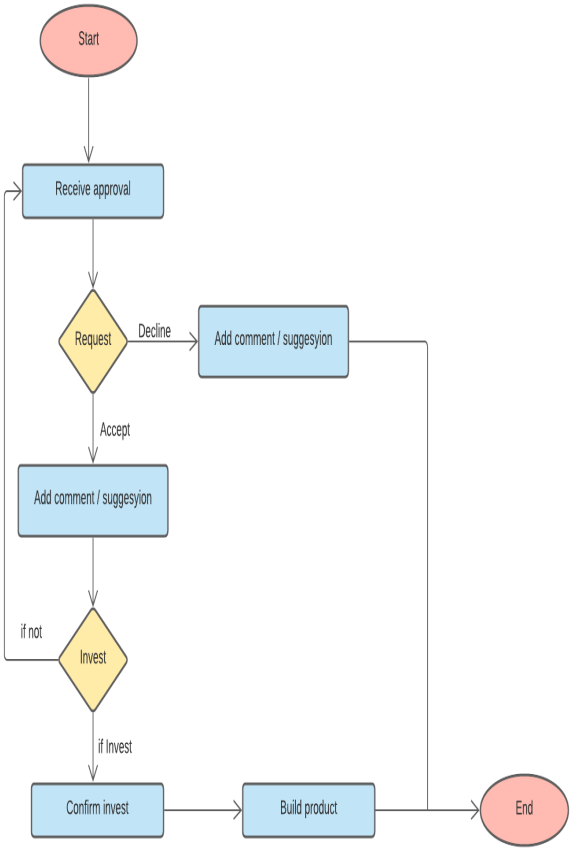
Usecase Diagram



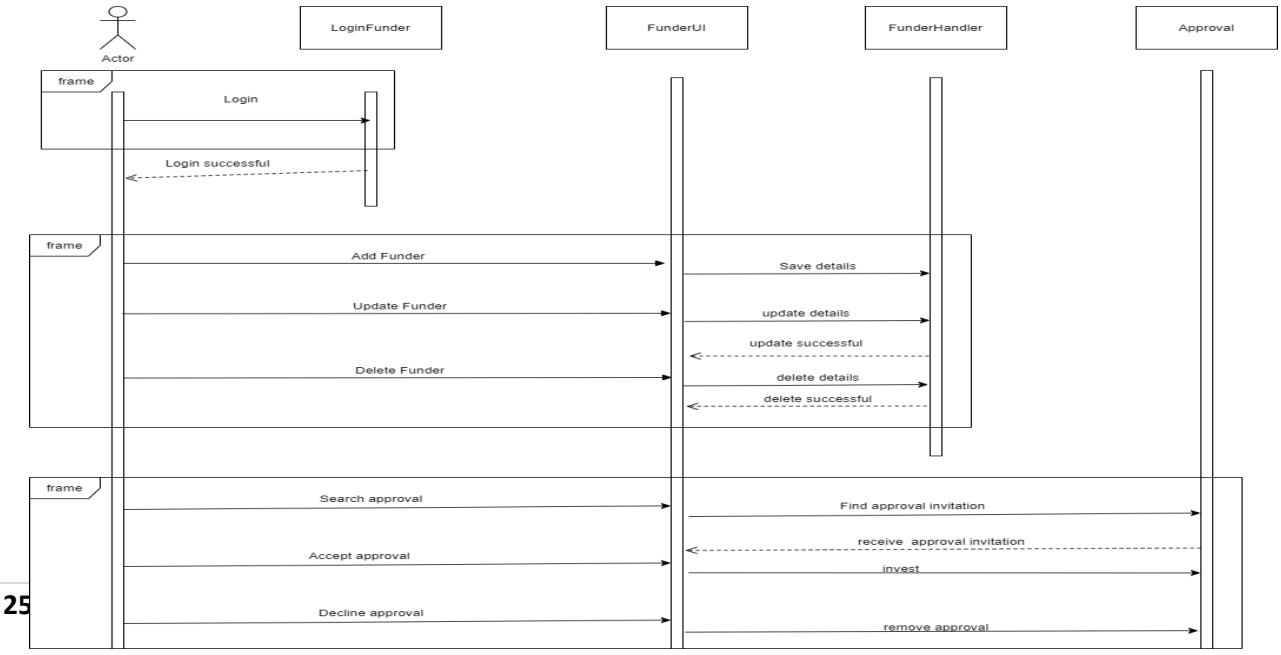
Activity Diagram



Flow Chart

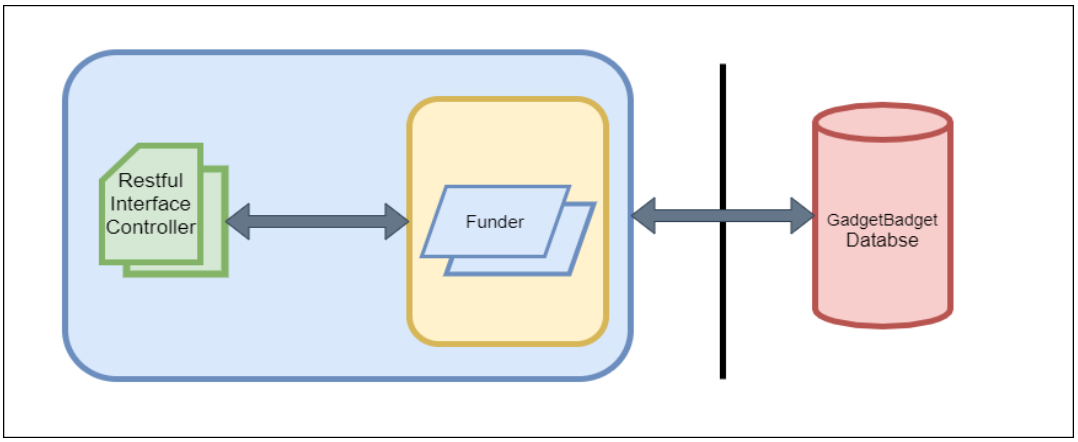


Sequence Diagram



Internal workflow

Funder:



API Design Relational

In funder management, funders are the party which funds the inventors to build their valued products. The funder management is user friendly and easy to understand with less complication.

- The funders can add their details, delete, view and update them.

Service Development

- Technologies Used: Java - JAX-RS (Jersey) on Tomcat
- IDE : Eclipse
- Database : MySQL
- Testing : POSTMAN

API for get all the Funder details in the database (GET Request)

URL: - http://localhost:8090/GadgetBadget/FunderService/Funders

Request : - { }

Response : - {FunderID : "Auto generated integer value"}
{FunderName =:"pathum "}
{FunderEmail :pathum@gmail.com }
{FunderContact : 0714444464"}
{FunderType: "Government"}

API for insert a new Funder to the database (POST Request)

`URL: - http://localhost:8090/GadgetBadget/FunderService/Funders

Request : - FunderID : "Auto generated integer value"}
{FunderName =:"pathum "}
{FunderEmail :pathum@gmail.com }
{FunderContact : 0714444464"}
{FunderType: "Government"}

Response :- { Result = "Inserted successfully" }
{ FunderID = "Auto generated integer value"
{ Error= "Error while inserting"}

API for update Funder which is existing in database (PUT Request)

*`URL: -*http://localhost:8090/GadgetBadget/FunderService/Funders

Request : - FunderID : "Auto generated integer value"}
{FunderName =:"pathumwijerathne "}
{FunderEmail :pathumwijerathne@gmail.com }
{FunderContact : 07189844464"}
{FunderType: "Government"}

Response :- { Result = "Updated successfully" }
{ FunderID = "Auto generated integer value"
{ Error= "Error while Updating"}

API for delete a Funder which is existing in database (DELETE Request)

URL: - http://localhost:8090/GadgetBadget/FunderService/Funders

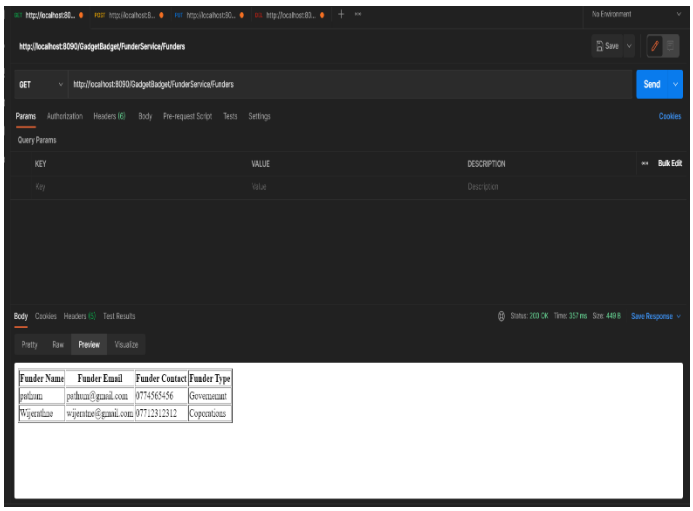
Request :- { FunderID = " Selected FunderID from the service "}

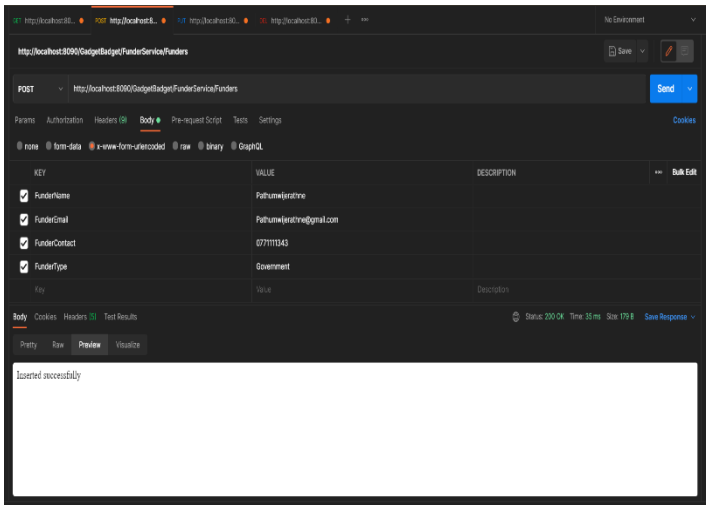
Response :- { Result = "Deleted successfully" }
{Error = "Error while deleting the Funder details" }

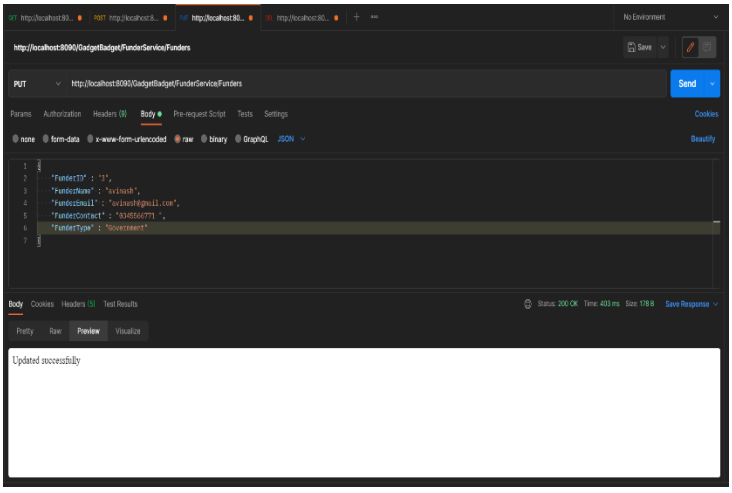
Test Cases

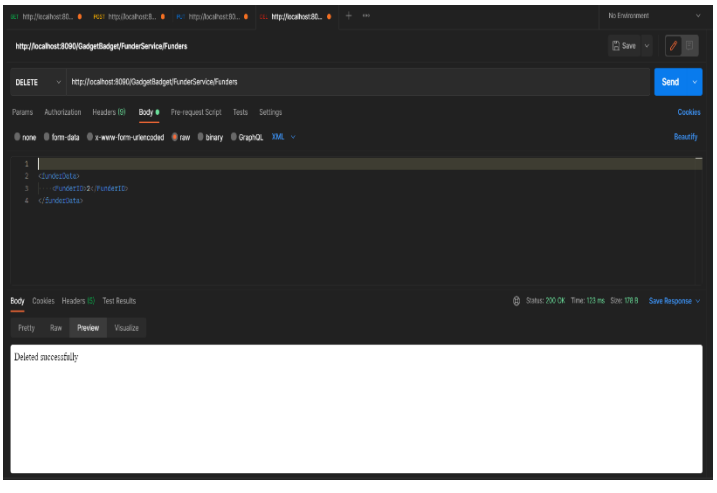
TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
1	Add Funder Details	{FunderName =:"pathum "} {FunderEmail :pathum@gmail.com } {FunderContact : 0714444464"} {FunderType: "Government"}	New Funder details are added to the database. Show message as "Inserted successfully".	New Funder details are added to the database. Show message as "Inserted successfully".	**PASS**
2	Update Funder Details	{"FunderID" : "3", "FunderName" : "avinash", "FunderEmail" : "avinash@gmail.com", "FunderContact" : "0345566771 ", "FunderType" : "Government" }	Funder details are updated according to relevant input Funder details. Show message as "Updated successfully".	Funder details are updated according to relevant input Funder details. Show message as "Updated successfully".	**PASS**
3	Delete Funder Details	<funderData> <FunderID>2</FunderID> </funderData>	Show message as “Deleted Successfully”	Funder Details are deleted successfully. Show message as “Deleted Successfully”	**PASS**

Testing


http://localhost:8090/GadgetBadget/FunderService/Funders
GET
http://localhost:8090/GadgetBadget/FunderService/Funders
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize
Funder Name Funder Email Funder Contact Funder Type
pathum pathum@gmail.com 0774565456 Government
Wijerathne wijerathe@gmail.com 0771212312 Coperations


http://localhost:8090/GadgetBadget/FunderService/Funders
POST
http://localhost:8090/GadgetBadget/FunderService/Funders
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL
KEY VALUE DESCRIPTION Bulk Edit
FunderName Pathum(jerathne
FunderEmail Pathum(jerathne@gmail.com
FunderContact 0771111343
FunderType Government
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize
Inserted successfully


http://localhost:8090/GadgetBadget/FunderService/Funders
PUT
http://localhost:8090/GadgetBadget/FunderService/Funders
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
{
"FunderID" : "1",
"FunderName" : "avinash",
"FunderEmail" : "avinash@gmail.com",
"FunderContact" : "0345567771 ",
"FunderType" : "Government"
}
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize
Updated successfully

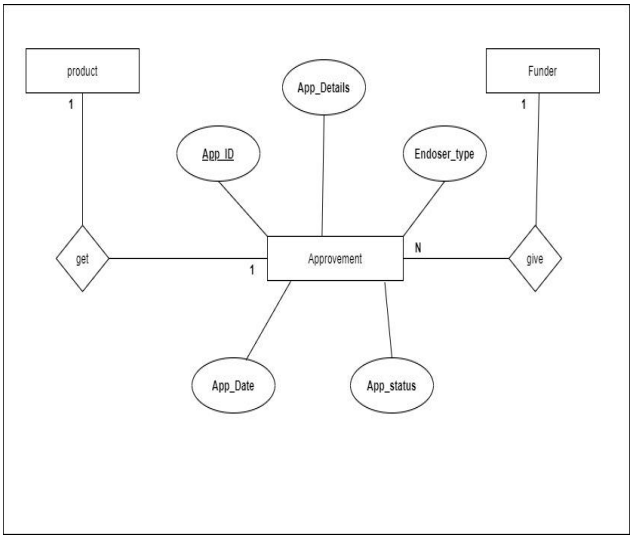

http://localhost:8090/GadgetBadget/FunderService/Funders
DELETE
http://localhost:8090/GadgetBadget/FunderService/Funders
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL XML
<FunderData>
<FunderID>2</FunderID>
</FunderData>
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize
Deleted successfully

IT19117010 (Jayasundara J.M.N.K.B) –URL:
http://localhost:8090/GadgetBadget/ApprovementService/Approvements

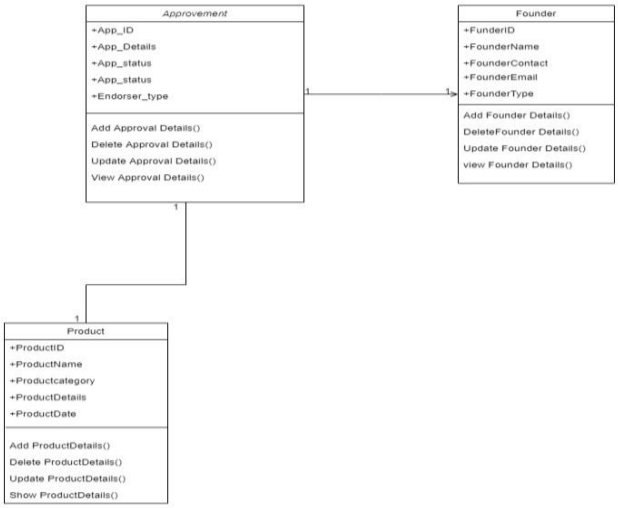
Approve Management

For the inventors who invent the products need the approvement to publish their product in the company's online platform. In this approvement management system the company has the full access to insert, update, delete and view the approvement details.

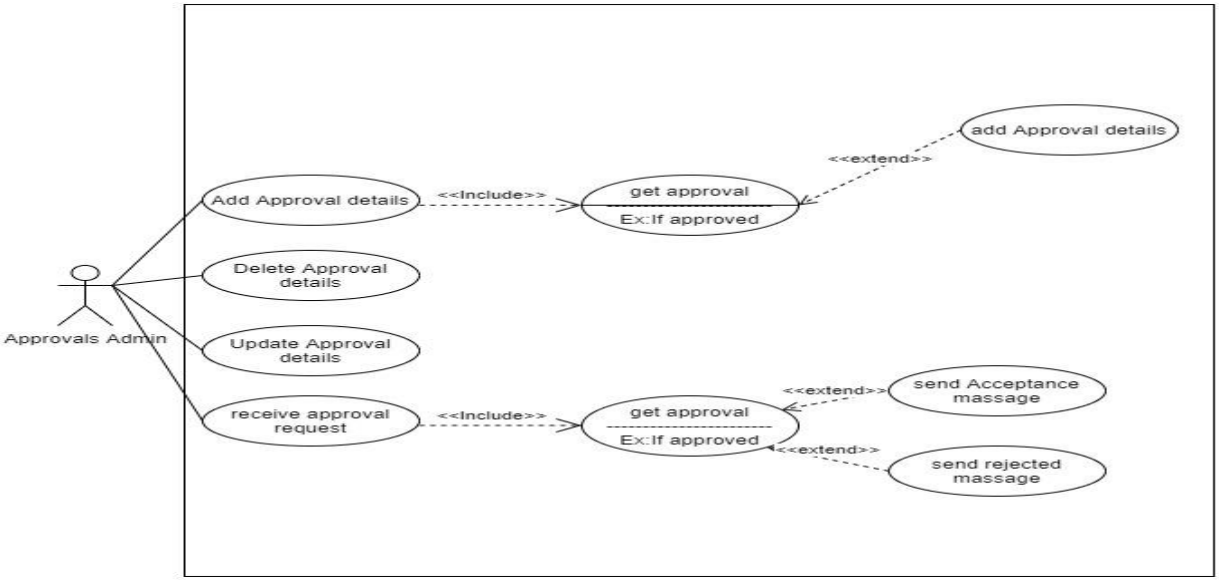
Er Diagram



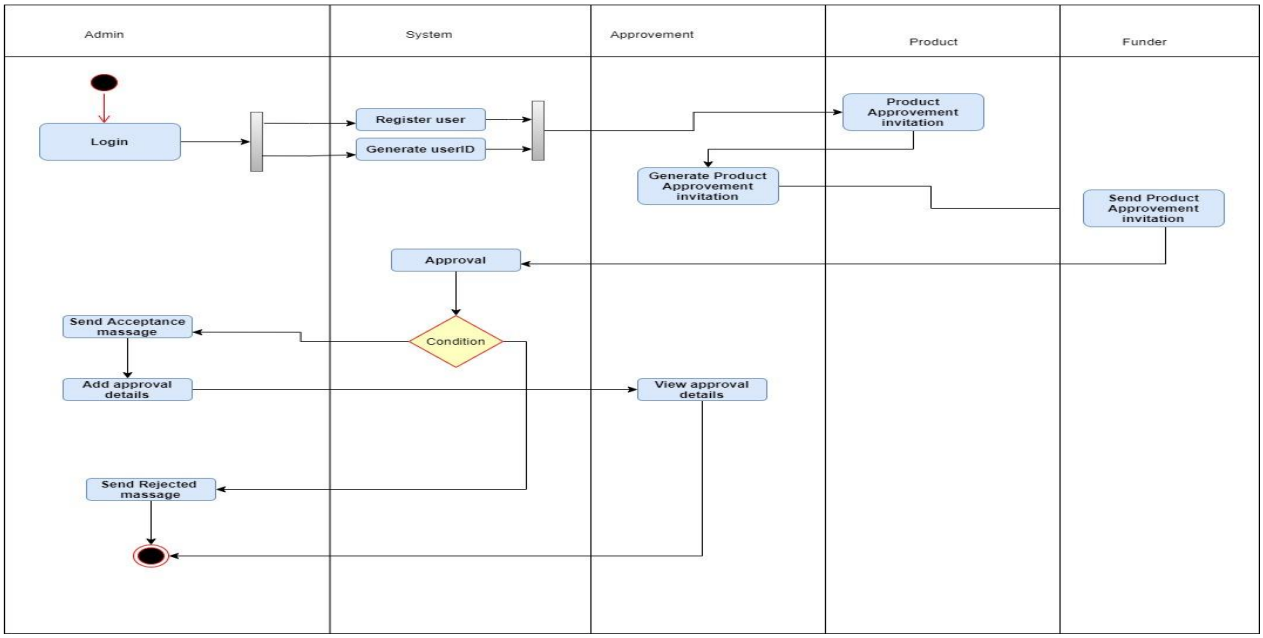
Class Diagram



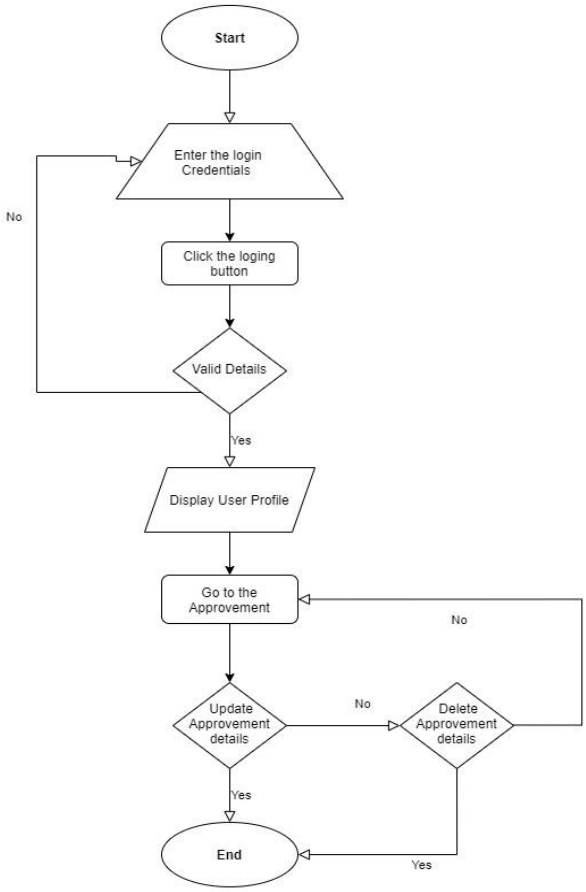
Usecase Diagram



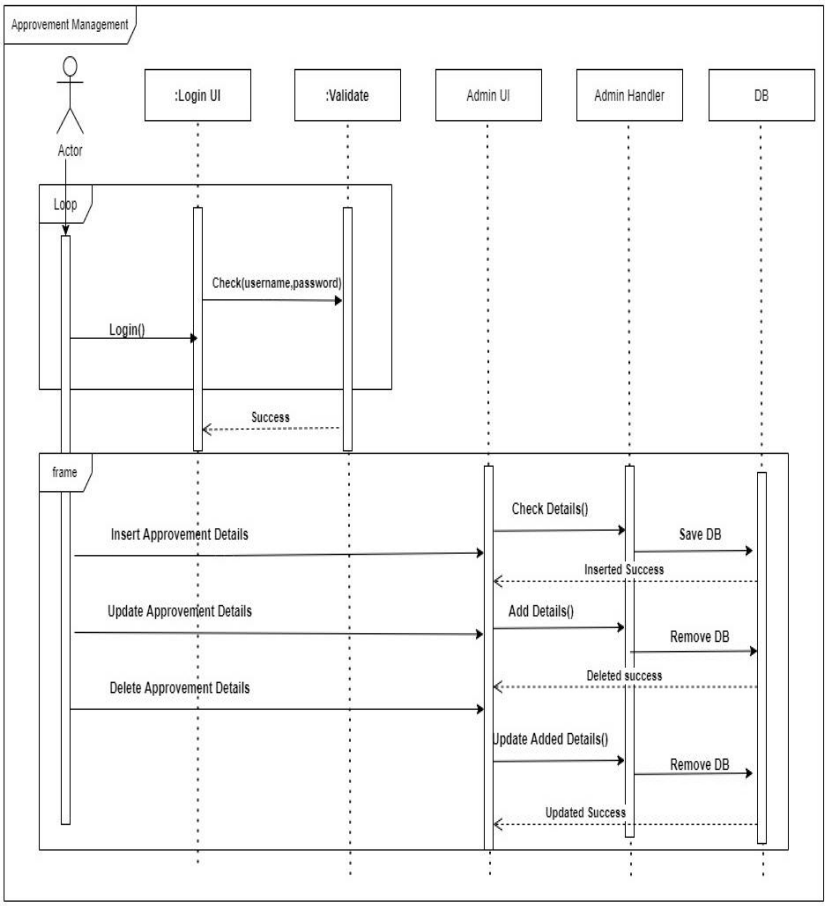
Activity Diagram



Flow chart

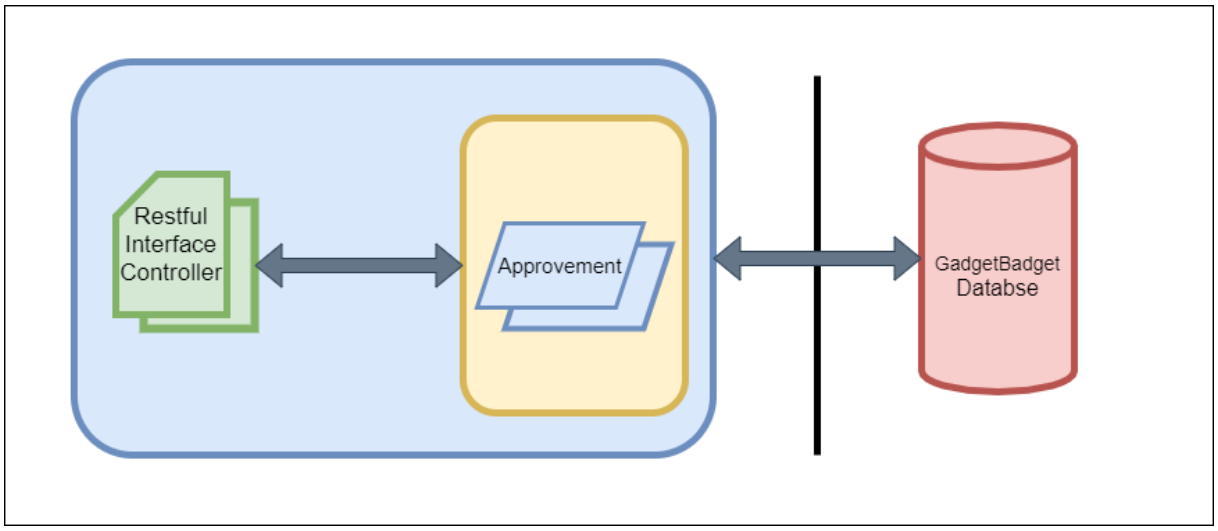


Sequence Diagram



Internal workflow

Approvement :



API Design Relational

In approvement management system the company can give the approvement to the product which are added by the inventors. The approvement management part is simple and less complicated.

- In approvement management the approvement details can be added, updated, viewed and deleted.

Service Development

- Technologies Used: Java - JAX-RS (Jersey) on Tomcat
- IDE : Eclipse
- Database : MySQL
- Testing : POSTMAN

API for get all the Approvement details in the database (GET Request)

URL: -

http://localhost:8090/GadgetBadget/ApprovementService/Approvements

Request : - { }

Response : - {ApproveID : "Auto generated integer value"}
{ApproveStatus =:"Approved"}
{ApproveDate = :"12.09.2018" }
{ApproveDetaills =: "071444464"}
{Endorser: "Kaush"}

API for insert a new Approvement details to the database (POST Request)

`URL: -
http://localhost:8090/GadgetBadget/ApprovementService/Approvements

Request : - ApproveID : "Auto generated integer value"}
{ApproveStatus =:"Approved"}
{ApproveDate : 12.09.2018 }
{ApproveDetaills : 071444464"}
{Endorser: "Kaush"}

Response :- { Result = "Inserted successfully" }
{ ApproveID = "Auto generated integer value"
{ Error= "Error while inserting"}

API for update Approvement which is existing in database (PUT Request)

`URL: -
http://localhost:8090/GadgetBadget/ApprovementService/Approvements

Request : - ApproveID : "Auto generated integer value"}
{ApproveStatus =:"Disapproved"}
{ApproveDate :"12.09.2018" }
{ApproveDetaills : 0718994464"}
{Endorser: "KaushJaysundara"}

Response :- { Result = "Updated successfully" }
{ ApproveID = "Auto generated integer value" }
{ Error= "Error while Updating"}

API for delete a Approvement which is existing in database (DELETE Request)

URL: -
http://localhost:8090/GadgetBadget/ApprovementService/Approvements

Request :- { ApproveID = " Selected ApproveID from the service "}

Response :- { Result = "Deleted successfully" }
{Error = "Error while deleting the Approvement details" }

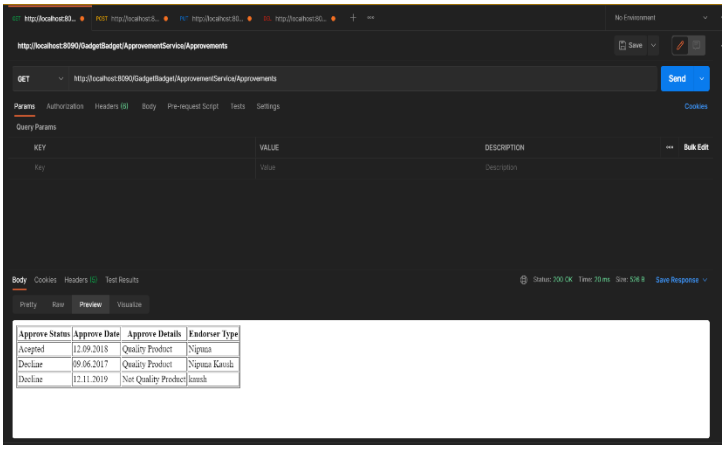
Test Cases

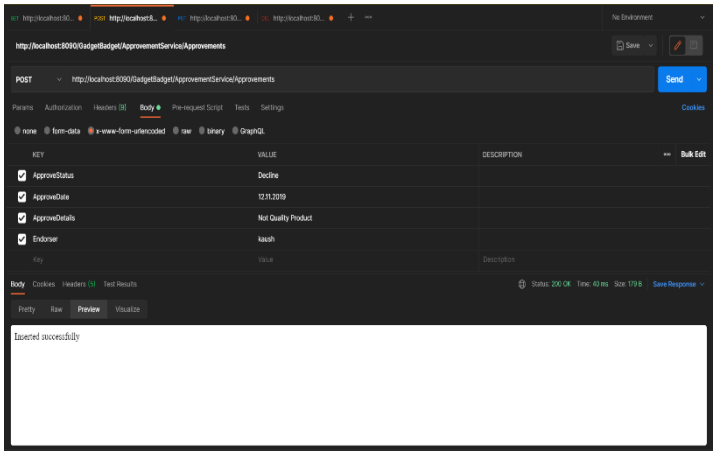
TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
1	Add Approvemnt Details	{ApproveStatus =:"Approved"} {ApproveDate = :"12.09.2018" } {ApproveDetaills =: "071444464"} {Endorser: "Kaush"}	New Approvemnt details are added to the database. Show message as "Inserted successfully".	New Approvemnt details are added to the database. Show message as "Inserted successfully".	**PASS**

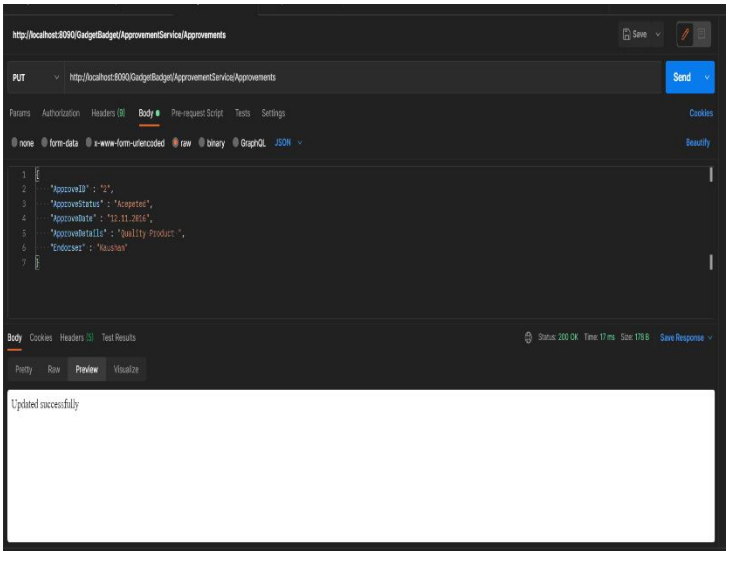
2	Update Approvemnt Details	{
		"ApproveID" : "2",
		"ApprovementStatus": "Disapproved",
		" ApproveDate " : "12.06.2014 ",
		"ApproveDetails" : "Not Quality Product",
		" Endorser " : "Kaush Jayasundara"
		}
			Approvemnt details are updated according to relevant input Approvemnt details. Show message as "Updated successfully".	Approvemnt details are updated according to relevant input Approvemnt details. Show message as "Updated successfully".	**PASS**

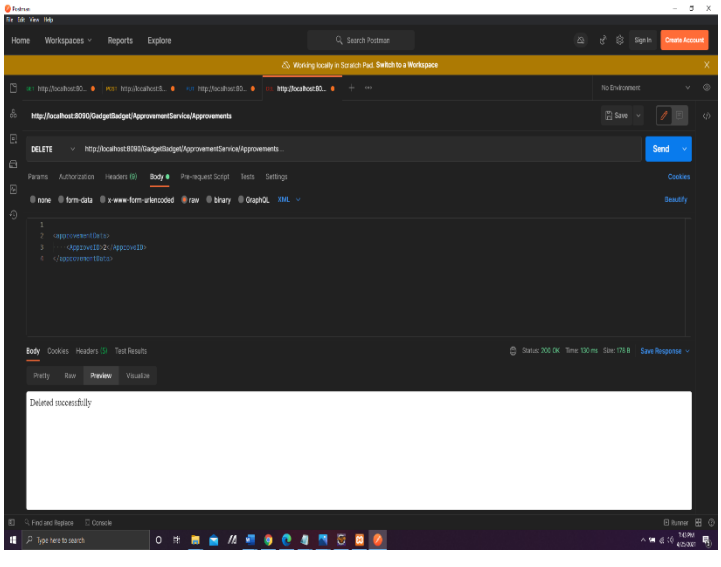
3	Delete Approvemnt Details	<approvementData> <ApprovemntID>2</ApprovementID> </ approvementData >	Show message as "Deleted Successfully"	Approvemnt Details are deleted successfully. Show message as "Deleted Successfully"	**PASS**

Testing


http://localhost:8090/GadgetBadget/ApprovementService/Approvements
GET
Send
Approve Status | Approve Date | Approve Details | Endorser Type
Accepted | 12.09.2018 | Quality Product | Nigma
Decline | 09.06.2017 | Quality Product | Nigma Kaush
Decline | 12.11.2019 | Not Quality Product | kaush


http://localhost:8090/GadgetBadget/ApprovementService/Approvements
POST
Send
ApproveStatus | Decline
ApproveDate | 12.11.2019
ApproveDetails | Not Quality Product
Endorser | kaush
Inserted successfully


http://localhost:8090/GadgetBadget/ApprovementService/Approvements
PUT
Send
{
"ApproveID" : "2",
"ApproveStatus" : "Accepted",
"ApproveDate" : "12.11.2014",
"ApproveDetails" : "Quality Product ",
"Endorser" : "Kaushan"
}
Updated successfully


http://localhost:8090/GadgetBadget/ApprovementService/Approvements
DELETE
Send
<approvementDetails>
<ApproveID>2</ApproveID>
</approvementDetails>
Deleted successfully

.**Tools Used**

- Java-JAX-RS (Jersey): Used for backend development
- Eclipse: Use as IDE for developing this project
- MVC Architecture: used to easily code our project
- Apache Tomcat V.09: Used as the server
- My SQL Used as our database

References:

https://www.youtube.com/watch?v=pCK6prSq8aw
https://www.youtube.com/watch?v=zid-MVo7M-E
https://courseweb.sliit.lk/course/view.php?id=4668

Appendix:

Use case and sequence diagrams have been added in earlier slides, each and every individual user have separately added their appendix in their respected part.