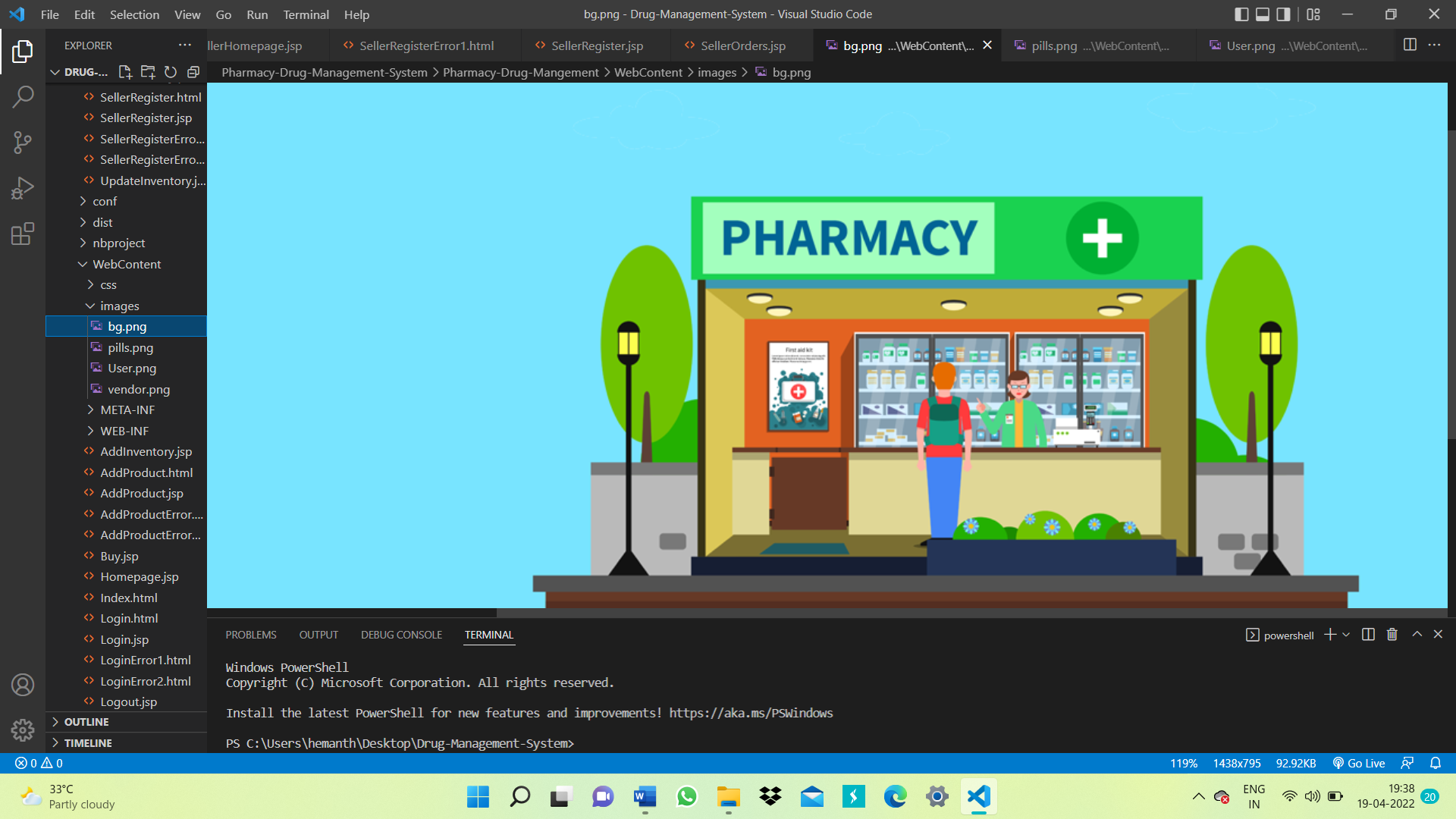
**Pharmacy Management System**



Date: 14/04/2022

Current Document Version : [1.0]



**DOCUMENT APPROVAL**

**Approvers of this document**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Department** | **Role** | **Signature** | **Date** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**DOCUMENT CHANGE HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Version #** | **Author** | **Date** | **Description** |
| 1.0 |  |  | Pharmacy Management System |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[**1.** **Document Purpose**: 5](#_Toc101269909)

[**2.** **Intended Audience:** 5](#_Toc101269910)

[**3.** **Project Background, Objective(s)** 5](#_Toc101269911)

[**3.1** **Project Background** 5](#_Toc101269912)

[**3.2 Project Objective** 5](#_Toc101269913)

[**4.** **Design Pattern** 6](#_Toc101269914)

[**5.** **Solution Diagram** 6](#_Toc101269915)

[**6.** **Solution Steps** 7](#_Toc101269916)

[**6.1** **Admin** 8](#_Toc101269917)

[**A.** **Admin Registration** 8](#_Toc101269918)

[**B.** **Admin Features** 8](#_Toc101269919)

[**6.2** **Doctor** 10](#_Toc101269920)

[**A.** **Doctor Registration:** 10](#_Toc101269921)

[**B.** **Doctor Features:** 10](#_Toc101269922)

[**7.** **Microservices/Classes/Interface/Method** 11](#_Toc101269923)

[**7.1 Classes** 11](#_Toc101269924)

[**7.2 Methods** 12](#_Toc101269925)

[**8.Validation** 15](#_Toc101269929)

[**9. Data Model/Table** 15](#_Toc101269930)

[**10. API Canvas** 16](#_Toc101269931)

[**11. Env Variables** 18](#_Toc101269932)

[**11.1 Admin Microservices** 18](#_Toc101269933)

[**11.2 Doctor Microservices** 18](#_Toc101269934)

[**11.3 Order Microservices** 18](#_Toc101269935)

[**11.4 Drug-Inventory Microservice** 19](#_Toc101269936)

[**11.5 Supplier Microservice** 19](#_Toc101269937)

[**11.6 Eureka-Server Microservice** 19](#_Toc101269938)

[**11.7 Zuul-gateway Microservice** 19](#_Toc101269939)

[**11.8 Payment-Gateway Microservice** 20](#_Toc101269940)

[**12 Integration** 20](#_Toc101269941)

[**13. HTTP Status Code** 21](#_Toc101269942)

[**15.0 Unit Testing** 21](#_Toc101269943)

[**1. Admin Registration/login** 21](#_Toc101269944)

[**2. Admin Features** 24](#_Toc101269945)

[**3. Doctor Registration/login** 26](#_Toc101269946)

[**16. Request** 28](#_Toc101269947)

[**17. Response** 28](#_Toc101269947)

# **Document Purpose**:

This document describes the solution of Microservice architecture for the Pharmacy Management System.

# **Intended Audience:**

This document is intended as a reference for the role of admin and doctor who are the users of the Pharmacy Management System.

|  |  |
| --- | --- |
| **Role** | **Nature of Engagement in web application** |
| Admin | Admin has access to all the functionality related to drugs, suppliers, and Orders. |
| Doctor/Customer | Doctors have access to view and buy drugs. |

# **Project Background, Objective(s)**

## **Project Background**

A pharmacy management system is a management system that is designed to improve accuracy and enhance safety and efficiency in the pharmaceutical store. It is a computer-based system that helps the Pharmacist to improve inventory management, cost, medical safety, etc.

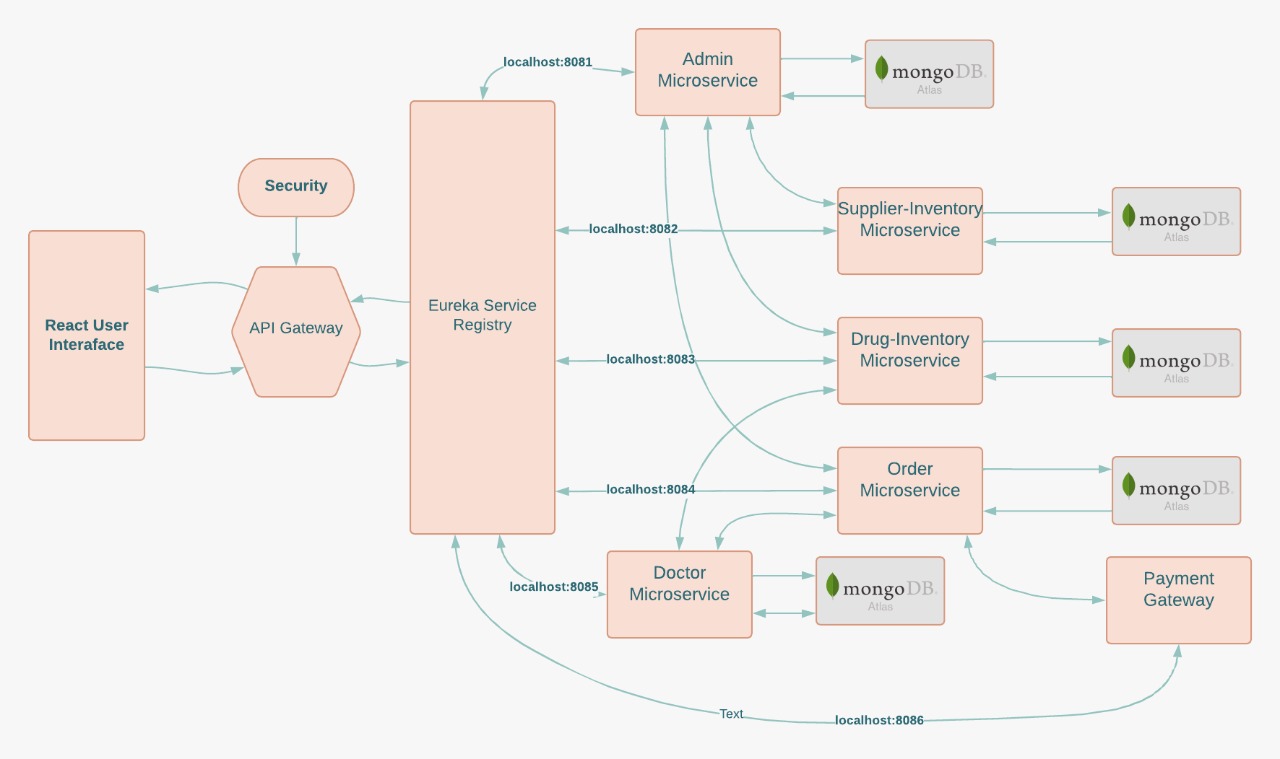
## **3.2 Project Objective**

The main objective of developing this (online Pharmacy Management System project) is to manage a medical shop like this online medical where an admin can add the stock in the shop (Application). And a customer view that medicine and request it to purchase. The user can only view and buy medicine only.

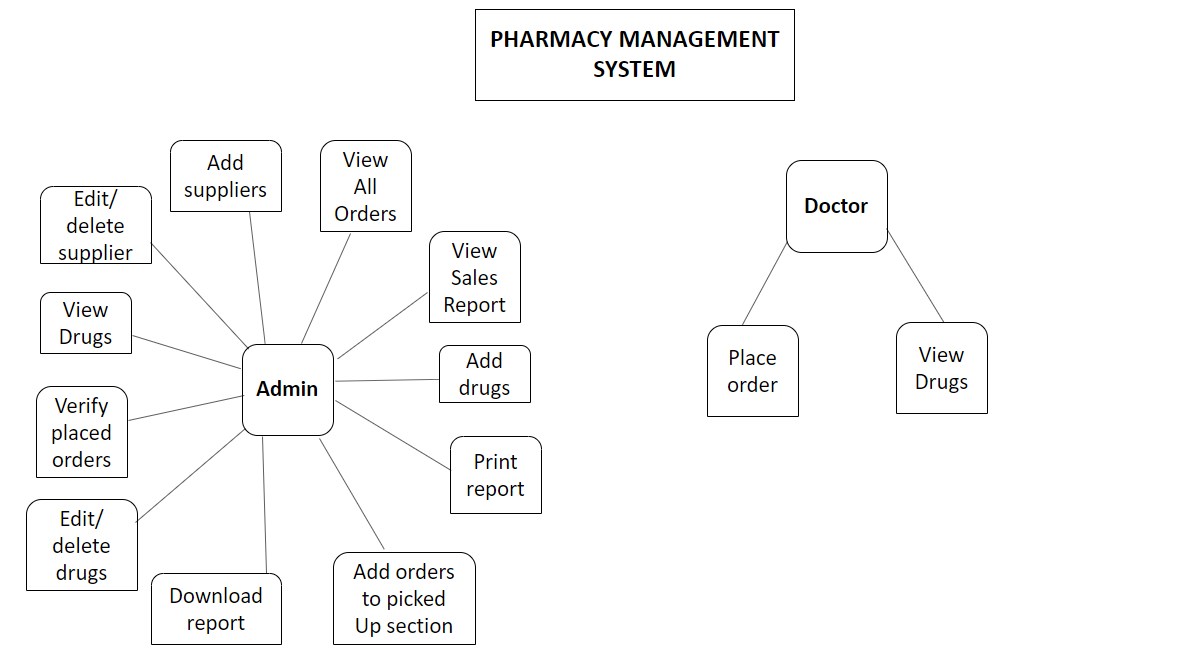
# **Design Pattern**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Name** | **Description** |
| 1 | Admin Microservice | Using HTTP requests, we will authorize the respective action of the admin to trigger various operations |
| 2. | Doctor Microservice | Using HTTP requests, we will authorize the respective action of the doctor to trigger various operations |
| 3. | Orders Microservice | Using HTTP requests, we will perform order-related operations. |
| 4. | Drug-Inventory Microservice | Using HTTP requests, we will perform Drug-Inventory related operations. |
| 5. | Supplier Microservice | Using HTTP requests, we will perform Supplier-Inventory related operations. |
| 6. | Eureka-Server Microservice | This microservice will set as a Eureka server and rest microservice will be set as a eureka client. At the runtime all the microservices will be register on eureka server. |
| 7. | Zuul-gateway Microservice | It is an API Gateway application. It handles all the requests and performs the dynamic routing of microservice applications. It works as a front door for all the requests. |
| 8. | Payment-gateway Microservice | In this microservice, all the payment related configuration will be done here. We are using Paytm as a mode of payment. |

# **Solution Diagram**



**6. Flow Diagram of Pharmacy-Management-System:**



# **Solution Steps**

## **Admin**

### **Admin Registration**

1. **Login/Signup:** Admin will enter the required details such as name, contact, email, and password and click submit button browser directs the request to admin Microservice.
2. call reaches the API gateway.
3. **For Sign-Up:** API gateway does the routing and forwards the request to controller.java class of admin and calls signUp() method.

After clicking submit it will redirect to the login page.

1. **For Login:** Admin will enter username and password and click submit button. API gateway does the routing and forwards the request to Controller.java class of admin and it will call login() method of admin.
2. login () method will validate username and password
   1. If username and password are valid admin will redirect to the home page of the application.
   2. Else admin will notify with an alert that he/she has entered the wrong credentials.

### **Admin Features**

**1.** **To View Drugs:**

* Admin will click on the view drug button.
* Then API gateway does the routing and forwards the request to controller.java class of Drug-inventory API.
* It will call the “viewDrugs()” method from the controller class.
* “viewDrugs()” method will retrieve drug details from database using DrugRepository.java class. Which is extending Mongo Repository.

**2. To Edit Drugs:**

* For updating the price of a drug, the admin will click on “edit drug price” and will enter the drug name with the updated price.
* Then API gateway does the routing and forwards the request to controller.java class of Drug-inventory API.
* It will call updateDrugs() method from the controller.java class.
* Update Drug method will update drug price in the database.

**3. To add Drugs:**

* For adding a new drugs admin will click on “add drug” and enter the details of the drugs.
* Then API gateway does the routing and forwards the request to controller.java class of Drug-inventory API.
* It will call addDrugs() method from the controller.java class.
* Add drug method will add a new drug to the database.

**4. To delete drugs:**

* For deleting drugs admin will click on “delete drugs” and enter the id/name of drugs.
* Then API gateway does the routing and forwards the request to controller.java class of Drug-inventory API.
* It will call the “deleteDrugs()” method from the controller.java class.
* “deleteDrugs()” method will delete specified drugs from the database.

**5. To view orders:**

* Admin will click on the View Order button.
* Then API gateway does the routing and forwards the request to controller.java class of Order API.
* It will call the “viewOrder()” method from the controller class.
* “viewOrder()” method will retrieve Order details from database using OrderRepository.java class. Which is extending Mongo Repository.

**6. To verify orders:**

* Admin will click on the Verify button which will be present below each order by this admin is validating order and it will go to the pickup section.
* By clicking on the verify button it will call “orderVarify()” method.
* “orderVarify()” will verify order .

**7. To view sales report:**

* Admin will click on view sales report button.
* It will call viewsalesreport() method and report will be shown.

**8. To view Suppliers:**

* Admin will click on the “view supplier” button.
* Then API gateway does the routing and forwards the request to controller.java class of supplier-inventory API.
* It will call the “viewSupplier()” method from the controller class.
* “viewSupplier()” method will retrieve suppliers and drugs details from SupplierRepository.java class. Which is extending Mongo Repository.

**9. To add supplier:**

* For adding a new supplier’s admin will click on “add new supplier” and enter the details of the supplier with the details of the supplied pharmacy goods.
* Then API gateway does the routing and forwards the request to controller.java class of supplier-inventory API.
* It will call the “addNewSupplier()” method from the controller.java class.
* “addNewSupplier()” method will add new supplier information with the supplied pharmacy goods in the database.

**10. To delete suppliers:**

* For deleting supplier admin will click on “delete supplier” and enter the id/name of the supplier.
* Then API gateway does the routing and forwards the request to controller.java class of supplier-inventory API.
* It will call the “deleteSupplier()” method from the controller.java class.
* “deleteSupplier()” method will delete specified suppliers along with their details from the database.

## **Doctor**

### **Doctor Registration:**

1. **Login/Signup:** The doctor will enter the required details such as name, contact, email, and password and click submit button browser directs the request to Doctor API.
2. call reaches the API gateway.
3. **For Sign-Up:** API gateway does the routing and forwards the request to controller.java class of doctor and calls signUp () method.
4. After clicking submit it will redirect to the login page.
5. **For Login:** the doctor will enter their username and password and click submit button. API gateway does the routing and forwards the request to Controller.java class of doctor and it will call login() method of the doctor.
6. login() method will validate username and password
7. If the username and password are valid doctors will redirect to the home page of the application.
8. Else doctor will notify with an alert that he/she has entered the wrong credentials.

### **Doctor Features:**

**1.To view Drugs:**

* Doctor will click on the view drug button.
* Then API gateway does the routing and forwards the request to controller.java class of supplier-inventory API.
* It will call the “viewDrugs()” method from the controller class.
* “viewDrugs()” method will retrieve drug details from DrugRepository.java class database using Which is extending Mongo Repository.

**2. To Order Drugs:**

* Doctor will click on the Order button.
* Then API gateway does the routing and forwards the request to controller.java class of “Order” API.
* It will call the “Order ()” method from the controller class.
* “Order()” method will Add drug Orders in the database of Order API

# **Microservices/Classes/Interface/Method**

## **7.1 Classes**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Microservices** | **Classes** | **description** |
| 1 | Admin Microservice | AdminApplication.java | It is a class with annotation “Springbootapplication”. To run admin microservice, we will run this class. It will run with embedded “Apache Tomcat server”. |
| AdminController.java | It is a controller class, request mapping of all uri’s related to admin with the require methods will be happen here. Main working of this class is related to login and signup. |
| Admin.java | It is a attribute class defines all the variable and their getters, setters and Constructor. |
| AdminRepository.java | This interface extending interface Mongo Repository so that we can perform all crud operations depending on requirements. |
| 2 | Doctor Microservice | DoctorApplication.java | It is a class with annotation “Springbootapplication”. To run doctor microservice, we will run this class. It will run with embedded “Apache Tomcat server”. |
| DoctorController.java | It is a controller class, request mapping of all uri’s related to admin with the require methods will be happen here. Main working of this class is related to login and signup. |
| Doctor.java | It is a attribute class defines all the variable and their getters, setters and Constructor. |
| DoctorRepository.java | This interface extends interface Mongo Repository so that we can perform all crud operations depending on requirements |
| 3 | Order microservice | OrderApplication.java | It is a class with annotation “Springbootapplication”. To run order microservice, we will run this class. It will run with embedded “Apache Tomcat server”. |
| OrderController.java | It is a controller class, request mapping of all uri’s related to order with the require methods will be happen here. Operations like viewing order will be done here and by using “RestTemplate” Admin microservice will be accesing this operation |
| OrderRepository.java | This interface extends Mongo Repository so that we can perform all crud operations depending on requirements |
| Order.java | It is a attribute class defines all the variable and their getters, setters and Constructor. |
| 4 | Drug inventory microservice | DrugInvenApplication.java | It is a class with annotation “Springbootapplication”. To run Supplier-Inventory microservice, we will run this class. It will run with embedded “Apache Tomcat server”. |
| DrugInvenController.java | It is a controller class, in this class request mapping of all uri’s related to drugs and supplier with the require methods will be happen here. CRUD operations related to drugs and suppliers will be performed here. |
| DrugInven.java | It is a attribute class defines all the variable and their getters, setters and Constructor. |
| DrugInvenRepository.java | This interface extends Mongo Repository so that we can perform all crud operations depending on requirements |
| 5 | Supplier inventory | SuppInvenApplication.java | It is a class with annotation “Springbootapplication”. To run Supplier-Inventory microservice, we will run this class. It will run with embedded “Apache Tomcat server”. |
| SuppInvenController.java | It is a controller class, in this class request mapping of all uri’s related to drugs and supplier with the require methods will be happen here. CRUD operations related to drugs and suppliers will be performed here. |
| SuppInven.java | It is a attribute class defines all the variable and their getters, setters and Constructor. |
| SuppInvenRepository.java | This interface extends Mongo Repository so that we can perform all crud operations depending on requirements |

## **7.2 Methods**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Classes | Methods | | Description | |
| 1 | AdminController.java | 1. authentication() | This method will match the credentials and will authenticate request. | |
| 1. viewDrug() | It will fetch all the drugs info using RestTemplate from Supplier-Inventory microservice. | |
| 1. addDrug() | It will add drugs in supplier inventory database using RestTemplate. | |
| 1. editDrug | It will edit drug info from supplier inventory database using RestTemplate. | |
| 1. deleteDrugs() | It will delete drug from supplier inventory database using RestTemplate. | |
| 1. viewSupp() | It will view suppliers detail from supplier inventory database using RestTemplate. | |
| 1. addSupp() | It will add suppliers in supplier inventory database using RestTemplate. | |
| 1. deleteSupp() | It will delete supplier from supplier inventory database using RestTemplate. | |
| 1. viewOrders() | It will view orders from order database using RestTemplate. | |
| 1. verifyOrder() | It will verify orders, which are make by doctor. | |
| 1. pickup() | It will place verify order to pickup section. | |
| 1. viewSaleReports() | Sales report can be seen by using this method. | |
| 1. downloadSalesReport() | By using this method report can be download. | |
| 2 | DoctorController.java | i) signUp() | | This method will store the details which are receive from new user and it will store in Doctor database. | |
| ii)login() | | This method will match the credentials and will authenticate request. | |
| iii) viewDrug() | | It will fetch all the drugs info using RestTemplate from Supplier-Inventory microservice. | |
| iv) orderDrug() | | Doctor will order drugs from available options and after clicking on order button that order will be store in database of Order microservice. | |
| 3 | OrderController.java | i) viewOrder | | This method will show the orders available in the order database. This method will be use by Admin microservices using RestTemplate. | |
| ii) Addorder | | This method will add the orders in database and this method will be use by Admin and Doctor microservice using RestTemplate. | |
| 4 | DrugInvenController.java | i. viewDrug() | | This method will show the drugs available in database. This method will be use by both admin and doctor microservice using RestTemplate. | |
| ii. addDrug() | | This method will add the drugs in database of supplier-inventory microservice. This method will be used by admin microservice using RestTemplate. | |
| iii. editDrug | | This method will edit drug info. This method will be used by admin microservice using RestTemplate . | |
| iv. deleteDrugs() | | This method will delete drugs from database. This method will be used by admin microservice using RestTemplate. | |
| 5 | SuppInvenController.java | i. viewSupp() | | This method will show the details of suppliers from database. This method will be used by admin microservice using RestTemplate. | |
| ii. addSupp() | | This method will add the suppliers from database. This method will be used by admin microservice using RestTemplate. | |
| iii. deleteSupp() | | This method will delete the details of suppliers from database. This method will be used by admin microservice using RestTemplate. | |

# **8.Validation**



# **9. Data Model/Table**

Admin Microservice

|  |  |
| --- | --- |
| **Admin** |  |
| id | Int |
| name | String |
| email | String |
| phoneNumber | Long |

Doctor Microservice

|  |  |
| --- | --- |
| **Doctor** |  |
| id | Int |
| name | String |
| email | String |
| phoneNumber | Long |
| password | String |

Order Microservice

|  |  |
| --- | --- |
| **Order** |  |
| id | Int |
| Drugname | String |
| DrugPrice | Long |

Drug-Inventory Microservice

|  |  |
| --- | --- |
| **Drug** |  |
| id | Int |
| Drugname | String |
| DrugPrice | Long |

Supplier-Inventory Microservice

|  |  |
| --- | --- |
| **Supplier** |  |
| id | Int |
| name | String |
| email | String |
| phoneNumber | Long |
| Drugname | String |
| DrugPrice | Long |

# **10. API Canvas**

Admin Microservice

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Microservice** | **Path**  **(/admin/\*)** | **Mapping** | **API Description** | **Role** | **Auth** |
| Admin | /login | Post | Admin Login | Admin | True |
| Admin | /drug/view | Get | To get drug list | Admin | True |
| Admin | /drug/add | Post | To add drug | Admin | True |
| Admin | /drug/edit | Put | To update drug info | Admin | True |
| Admin | /drug/delete | Delete | To delete drug | Admin | True |
| Admin | /supp/view | Get | To get supplier list | Admin | True |
| Admin | /supp/add | Post | To add supplier | Admin | True |
| Admin | /supp/delete | delete | To delete supplier | Admin | True |

Doctor Microservice

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Microservice** | **Path**  **(/doctor/\*)** | **Mapping** | **API Description** | **Role** | **Auth** |
| Doctor | /signup | Post | Doctor signup | Doctor | True |
| Doctor | /login | post | Doctor login | Doctor | True |
| Doctor | /view | Get | To view drugs | Doctor | True |
| Doctor | /order | Post | To order drugs | Doctor | True |

Order Microservice

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Microservice** | **Path**  **(/order/\*)** | **Mapping** | **API Description** | **Role** | **Auth** |
| Order | /view | Get | To show all orders | Doctor/Admin | True |
| Order | /add | post | To add order | Doctor/Admin | True |
| Order | /viewreport | Get | To view report | Admin | True |
| Order | /viewreport/download | Get | To download report | Doctor | True |

Drug-Inventory Microservice

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Microservice** | **Path**  **(/drug/\*)** | **Mapping** | **API Description** | **Role** | **Auth** |
| Drug-Inventory | /view | Get | To get drug list | Admin/Doctor | True |
| Drug-Inventory | /add | Post | To add drug | Admin/Doctor | True |
| Drug-Inventory | /edit | Put | To update drug info | Admin | True |
| Drug-Inventory | /delete | Delete | To delete drug | Admin | True |

Supplier-Inventory Microservice

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Microservice** | **Path**  **(/supp/\*)** | **Mapping** | **API Description** | **Role** | **Auth** |
| Supplier-Inventory | /view | Get | To get supplier list | Admin | True |
| Supplier-Inventory | /add | Post | To add supplier | Admin | True |
| Supplier-Inventory | /delete | delete | To delete supplier | Admin | True |

# **11. Env Variables**

## **11.1 Admin Microservices**

* Server.port = 8081
* Spring.application.name = Admin-service
* management.endpoint.web.exposure.include = hystrix
* Spring.data.mongodb.uri= uri of mongodb database
* Spring.mvc.pathmatch.matching-strategy = ant-path-matcher
* Eureka.client.register-with-eureka= true
* Eureka.client.fetch-registry= true

## **11.2 Doctor Microservices**

* Server.port = 8082
* Spring.application.name = Doctor-service
* management.endpoint.web.exposure.include = hystrix
* Spring.data.mongodb.uri= uri of mongodb database
* Spring.mvc.pathmatch.matching-strategy = ant-path-matcher
* Eureka.client.register-with-eureka= true
* Eureka.client.fetch-registry= true

## **11.3 Order Microservices**

* Server.port = 8083
* Spring.application.name = Order-service
* management.endpoint.web.exposure.include = hystrix
* Spring.data.mongodb.uri= uri of mongodb database
* Spring.mvc.pathmatch.matching-strategy = ant-path-matcher
* Eureka.client.register-with-eureka= true
* Eureka.client.fetch-registry= true

## **11.4 Drug-Inventory Microservice**

* Server.port = 8084
* Spring.application.name = Drug-Inventory-service
* management.endpoint.web.exposure.include = hystrix
* Spring.data.mongodb.uri= uri of mongodb database
* Spring.mvc.pathmatch.matching-strategy = ant-path-matcher
* Eureka.client.register-with-eureka= true
* Eureka.client.fetch-registry= true

## **11.5 Supplier Microservice**

* Server.port = 8085
* Spring.application.name = Suppplier-service
* management.endpoint.web.exposure.include = hystrix
* Spring.data.mongodb.uri= uri of mongodb database
* Spring.mvc.pathmatch.matching-strategy = ant-path-matcher
* Eureka.client.register-with-eureka= true
* Eureka.client.fetch-registry= true

## **11.6 Eureka-Server Microservice**

* Server.port = 8761
* Spring.application.name = Eureka-server
* Eureka.client.register-with-eureka= false
* Eureka.client.fetch-registry= false

## **11.7 Zuul-gateway Microservice**

* Server.port = 9090
* Spring.application.name = zuul-gateway
* Eureka.client.register-with-eureka= true
* Eureka.client.fetch-registry= true
* eureka.client.serviceUrl.defaultZone= <uri>
* eureka.client.instance.preferIpAddressIpdress = true
* zuul.routes.user-service.path = /admin/\*\*
* zuul.routes.user-service.serviceId = Admin-service
* zuul.routes.user-service.stripPrefix=true
* zuul.routes.user-service.path = /doctor /\*\*
* zuul.routes.user-service.serviceId = Doctor-service
* zuul.routes.user-service.stripPrefix=true
* zuul.routes.user-service.path = /order /\*\*
* zuul.routes.user-service.serviceId = Order-service
* zuul.routes.user-service.stripPrefix=true
* zuul.routes.user-service.path = /drug /\*\*
* zuul.routes.user-service.serviceId = Drug-Inventory-service
* zuul.routes.user-service.stripPrefix=true
* zuul.routes.user-service.path = /supp /\*\*
* zuul.routes.user-service.serviceId = Suppplier-service
* zuul.routes.user-service.stripPrefix=true

## **11.8 Payment-Gateway Microservice**

* Server.port = 8086
* paytm.payment.sandbox.merchantId:your merchant ID
* paytm.payment.sandbox.merchantKey: your merchant key
* paytm.payment.sandbox.channelId:WEB
* paytm.payment.sandbox.industryTypeId:Retail
* paytm.payment.sandbox.website:WEBSTAGING
* paytm.payment.sandbox.paytmUrl:https://securegw-
* stage.paytm.in/order/process
* paytm.payment.sandbox.callbackUrl:<uri>
* esponse paytm.payment.sandbox.details.MID:
* ${paytm.payment.sandbox.merchantId}
* paytm.payment.sandbox.details.CHANNEL\_ID:
* ${paytm.payment.sandbox.channelId}
* paytm.payment.sandbox.details.INDUSTRY\_TYPE\_ID:
* ${paytm.payment.sandbox.industryTypeId}
* paytm.payment.sandbox.details.WEBSITE:
* ${paytm.payment.sandbox.website}
* paytm.payment.sandbox.details.CALLBACK\_URL:
* ${paytm.payment.sandbox.callbackUrl} paytm.mobile = your
* paytm registered mobile number paytm.email = your email address
* Eureka.client.register-with-eureka= true
* Eureka.client.fetch-registry= true

# **12. Integration**

## **12.1 RestTemplate**

* Integration between microservices can be done by using restemplate.
* Microservice can access specified fuctionality of other microservice using resttemplate.

## **12.2 Integration of Paytm as payment gateway with spring boot Application:**

* By creating account in Paytm business, we will get Paytm properties such as Merchant id, Merchant key, etc.
* By configuring “application.properties” file with properties like “paytm.payment.sandbox” and configuring classes in microservice , we can integrate Paytm as payment gateway with spring boot application.

# **13. HTTP Status Code**

201 – Customer Registered

200 - Request succeeded

400 – Inputs are invalid

404 – Customer Not found

# **15. Unit Testing**

|  |  |
| --- | --- |
| Project Name | Pharmacy Management System |
| Date of Creation | 18/04/22 |
| Date of review |  |

### **15.1 Admin Registration/login**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | Actual Result |
| TC\_o1 | Admin Registration  **(Valid details)** | Enter the valid data to get registered | Admin needs to enter all the valid details | **A**. Enter  1)Admin firstname: group  2)Admin lastname:  seven  3)email:group7@gmail.com  4)password:  123@Group7  5)Confirm password:  123@Group7  6)Phone:9123456789  7)address: India  8)gender:male  **B**. Enter Submit | <Valid Details> | Successful registration | Successful registration |
| TC\_o2 | Admin registration  **(Wrong Phone Number)** | Entered the invalid data for registration, | Admin entered the phone number with a invalid digits. | **A**. Enter  Admin firstname: group  2)Admin lastname:  seven  3)email:group7@gmail.com  4)password:  123@Group7  5)Confirm password:  123@Group7  6)Phone:9123  7)address: India  8)gender:male  **B**. Enter Submit | <invalid phone number> | Phone number should contain 10 digits. | Phone number should contain 10 digits. |
| TC\_o3 | Admin registration  **(Wrong Email)** | Enter the correct email address for registration. | Customer must enter the valid email address. | **A**. Enter  Admin FirstName: group  2)Admin LastName:  seven  3)email:group121  4)password:  123@Group7  5)Confirm password:  123@Group7  6)Phone:9123456789  7)address: India  8)gender:male  **B**. Enter Submit | <invalid Email Address> | Please enter the valid Email Address | Please enter the valid Email Address |
| TC\_o4 | Admin registration  **(Wrong Password)** | Enter the correct password format. | Enter atleast 8 characters with one special character and and 1 digit. | **A**. Enter  Admin FirstName: group  2)Admin LastName:  seven  3)email: group7@gmail.com  4)password:  abcdef  5)Confirm password:  abcdef  6)Phone:9123456789  7)address: India  8)gender:male  **B**. Enter Submit | <invalid password> | Please enter the valid password | Please enter the valid password |
| TC\_o5 | Admin registration  **(Missed Attributes)** | Do not Enter all the required fields to get registered. | If admin misses one of the fields during registration. | **A**. Enter  Admin firstname: group  2)Admin lastname:  seven  3)email:  4)password:  123@Group7  5)Confirm password:  123@Group7  6)Phone:9123456789  7)address: India  8)gender:male  **B**. Enter Submit | <email is missing> | Missing the fields | Missing the fields |
| TC\_o6 | Admin Login  **(Wrong Credentials)** | Invalid credentials | Either the Email address or the password is wrong, | **A.** Enter  Email: group10@gmail.com  Password: 123@G  **B.** Enter login | <Wrong Credentials> | Enter the correct Credential | Enter the correct Credentials |
| TC\_o7 | Admin Login  **(Correct Credentials )** | Enter the correct Credentials | The Email and the password are correct. | **A**. Enter 1)email:group7@gmail.com  2)password:  123@Group7  **B.** Enter login | < correct Credentials> | Successfully logged in | Successfully logged in |

### 

### **15.2 Admin Features**

1. **To Edit Drugs**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | Actual Result | Status |
| TC\_o1 | Admin Features | Price accepts only numerical element. | If the price doesn’t contain only numerical values. | 1)Enter  Previous price: 100  Updated price: gehde  2) Save | <invalid format of the price> | Price should be entered only in the numerical values. | Price should be entered only in the numerical values. | pass |
| TC\_o2 | Admin Features | Enters the numerical element | If the price contains only numerical values. | 1)Enter  Previous price: 100  Updated price: 150  2) Save | <valid format of the price> | The price is successfully updated | The price is successfully updated | pass |

1. **To Add Drugs**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | Actual Result | Status |
| TC\_o1 | Admin Features  **(Missed Attributes)** | Did not Entered all the required fields to add the drug. | If admin misses one of the fields during adding the drugs.. | **A)** Enter  1. id:  2. name: vita D  3. price: 100 | <id is missing> | Missing the fields | Missing the fields | pass |
| TC\_o2 | Admin Features | Enters all the required fields to add the drug. | Entered all the fields during adding the drugs.. | **A)** Enter  1. id:01  2. name: vita D  3. price: 100 | <valid data> | Successfully added the drugs. | Successfully added the drugs. | pass |

1. **To Delete Drugs**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | Actual Result |
| TC\_o1 | Admin Features  **(Wrong ID/Name)** | Enters the wrong id/name of the drug, which is not in the database. | That id/name of the drug is not present in the database. | 1) Enters invalid Id/Name  2) Enter Submit | <Invalid Id> | Check the id/name of the drug you want to delete | Check the id/name of the drug you want to delete |
| TC\_o2 | Admin Features | Enters the valid id/name of the drug, which is present in the database. | That id/name of the drug is present in the database. | 1) Enters valid Id/Name  2) Enter Submit | <valid Id> | The drug is deleted successfully. | The drug is deleted successfully. |

1. **To Add Supplier:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | Actual Result | Status |
| TC\_o1 | Admin Features  **(Missed Attributes)** | Did not Entered all the required fields to add the supplier. | If admin misses one of the fields during adding the drugs.. | **A)** Enter  1. Supplier id:  2. Supplier name:  Pharma industry  3. Address: India  **B)** Enter Submit | <id is missing> | Missing the fields | Missing the fields | pass |
| TC\_o2 | Admin Features | Enters all the required fields to add the supplier. | Entered all the fields during adding the supplier. | **A)** Enter  1. Supplier id:01  2. Supplier name:  Pharma industry  3. Address: India  **B)** Enter Submit | <valid data> | Successfully added the Supplier. | Successfully added the Supplier. | pass |

1. **To Delete Supplier:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | Actual Result |
| TC\_o1 | Admin Features  **(Wrong ID/Name)** | Enters the wrong id/name of the Supplier, which is not in the database. | That id/name of the Supplier is not present in the database. | 1) Enters invalid Id/Name  2) Enter Submit | <Invalid Id> | Check the id/name of the Supplier you want to delete | Check the id/name of the supplier you want to delete |
| TC\_o2 | Admin Features | Enters the valid id/name of the supplier, which is present in the database. | That id/name of the supplier is present in the database. | 1) Enters valid Id/Name  2) Enter Submit | <valid Id> | The supplier is deleted successfully. | The supplier is deleted successfully. |

### **15.3 Doctor Registration/login**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | Actual Result |
| TC\_o1 | Doctor Registration  **(Valid details)** | Enter the valid data to get registered | Doctor needs to enter all the valid details | **A**. Enter  Doctor firstname: group  2)Doctor lastname:  seven  3)Email:group7@gmail.com  4)password:  123@Group7  5)Confirm password:  123@Group7  6)Phone:9123456789  7)address: India  8)gender:male  **B**. Enter Submit | <Valid Details> | Successful registration | Successful registration |
| TC\_o2 | Doctor registration  **(Wrong Phone Number)** | Entered the invalid data for registration, | Doctor entered the phone number with a invalid digits. | **A**. Enter  Doctor firstname: group  2)Doctor lastname:  seven  3)email:group7@gmail.com  4)password:  123@Group7  5)Confirm password:  123@Group7  6)Phone:9123  7)address: India  8)gender:male  **B**. Enter Submit | <invalid phone number> | Phone number should contain 10 digits. | Phone number should contain 10 digits. |
| TC\_o3 | Doctor registration  **(Wrong Email)** | Enter the correct email address for registration. | Customer must enter the valid email address. | **A**. Enter  Doctor FirstName: group  2)Doctor LastName:  seven  3)email:group121  4)password:  123@Group7  5)Confirm password:  123@Group7  6)Phone:9123456789  7)address: India  8)gender:male  **B**. Enter Submit | <invalid Email Address> | Please enter the valid Email Address | Please enter the valid Email Address |
| TC\_o4 | Doctor registration  **(Wrong Password)** | Enter the correct password format. | Enter atleast 8 characters with one special character and and 1 digit. | **A**. Enter  Doctor FirstName: group  2)Doctor LastName:  seven  3)email: group7@gmail.com  4)password:  abcdef  5)Confirm password:  abcdef  6)Phone:9123456789  7)address: India  8)gender:male  **B**. Enter Submit | <invalid password> | Please enter the valid password | Please enter the valid password |
| TC\_o5 | Doctor registration  **(Missed Attributes)** | Enter all the required fields to get registered. | If Doctor misses one of the fields during registration. | **A**. Enter  Doctor firstname: group  2)Doctor lastname:  seven  3)email:  4)password:  123@Group7  5)Confirm password:  123@Group7  6)Phone:9123456789  7)address: India  8)gender:male  **B**. Enter Submit | <email is missing> | Missing the fields | Missing the fields |
| TC\_o6 | Doctor Login  **(wrong Credentials)** | Invalid credentials | Either the Email address or the password is wrong, | **A.** Enter  Email: group10@gmail.com  Password: 123@G  **B.** Enter login | <Wrong Credentials> | Enter the correct Credential | Enter the correct Credentials |
| TC\_o7 | Doctor Login  **(Correct Credentials )** | Enter the correct Credentials | The Email and the password are correct. | **A**. Enter 1)email:group7@gmail.com  2)password:  123@Group7  **B.** Enter login | < correct Credentials> | Successfully logged in | Successfully logged in |

# **16. Request**

**Admin**

/admin/login

{

“username”:”group7@gmail.com”,

“password”:”123@Group7”,

}

/admin/drug/add

{

“drug id”: 01 ,

“drugname”:”name of the drug”,

“drugprice”: ,

}

/admin/drug/edit

{

“drugid”: ,

“drugprice”: ,

}

/admin/drug/delete

{

“drug id”: ,

“drugname”:”name of the drug”,

}

/admin/supp/add

{

“id”: ,

“name”:”name of the supplier”,

“address”:”address”,

“phonenumber”: ,

}

/admin/supp/delete

{

“id”: ,

“name”:”name of the drug”,

}

**Doctor**

/Doctor/signup

{

“id”: ,

“name”:”name of the doctor”,

“emailid”:”Doctor@gmail.com”,

“password”: password,

}

/Doctor/login

{

“username”:”username”,

“password”: password,

}

/Doctor/order

{

“id”: ,

“drugname”: “drug name”,

}

**Order**

/order/add

{ drugname:”name of the drug”,}

**Supplier Inventory**

/supplier inventory /drug/add

{

“id”: ,

“Drugname”:”Name of the drug”,

“Price”: ,

}

Supplier inventory/admin/drug/Delete

{

“id”: ,

“Drugname”:”Name of the drug”,

}

Supplier inventory/admin/drug/edit

{

“id”: ,

“drugprice”: ,

}

Supplier inventory/admin/supp/add

{

“id”: ,

“name”:”name of the supplier”,

“address”:”address”,

“phonenumber”: ,

}

Supplier inventory/admin/supp/delete

{

“id”: ,

“name”:”name of the drug”,

}

**17.0 Response:**

**Admin Registration**

**If valid details**

{

"message": "Admin registered successfully"

}

status code: 201

*status code 201=created*

**If invalid details**

{

“message”: “Inputs are not valid”

}, status code: 400

*status code 400=client error*

**If server encounters unexpected error**

{

“message”:” Internal server error”

}, status code: 500

*status code 500=internal server error*

**Admin login**

**If the credentials are correct**

{

"message": "Admin logged in Successfully"

}

status code: 200

*status code 200=Request Successful*

**If the credentials are incorrect**

{

“message”: “Inputs are not valid”

}, status code: 400

*status code 400=client error*

**If server encounters unexpected error**

{

“message”:” Internal server error”

}, status code: 500

*status code 500=internal server error*

**Admin Features**

1. **To Edit drugs**

**If valid ID**

{

“Price of the drug updated successfully”

}

status code: 200

*status code 200=Request Successful*

**If invalid ID**

{

“message”: “Drug not found”

}, status code:404

*status code 404=Not Found*

**If server encounters unexpected error**

{

“message” :” Internal server error ”

}, status code: 500

*status code 500=internal server error*

1. **To Add Drugs**

**If the details are valid**

{

“The Drug has been successfully added”

}

status code: 201

*status code 201=Created*

**If the details are invalid**

{

“message”: “Invalid details”

}, status code:400

*status code 401=Bad Request*

**If server encounters unexpected error**

{

“message”:” Internal server error”

}, status code: 500

*status code 500=internal server error*

1. **To Delete drugs**

**If valid ID**

{

“Drug deleted successfully”

}

status code: 200

*status code 200=Request Successful*

**If invalid ID**

{

“message”: “Drug not found”

}, status code:404

*status code 404=Not Found*

**If server encounters unexpected error**

{

“message”:” Internal server error”

}, status code: 500

*status code 500=internal server error*

1. **To Add Supplier**

**If the details are valid**

{

“The Supplier has been successfully added”

}

status code: 201

*status code 201=Created*

**If the details are invalid**

{

“message”: “Invalid details”

}, status code:400

status code 401=Bad Request

**If server encounters unexpected error**

{

“message” :” Internal server error ”

}, status code: 500

*status code 500=internal server error*

**5. To Delete Supplier**

**If valid ID**

{

“Supplier deleted successfully”

}

status code: 200

*status code 200=Request Successful*

**If invalid ID**

{

“message”: “Supplier not found”

}, status code:404

*status code 404=Not Found*

**If server encounters unexpected error**

{

“message” :” Internal server error ”

}, status code: 500

*status code 500=internal server error*

**Doctor Registration**

**If valid details**

{

"message": "Doctor registered successfully"

}

status code: 201

*status code 201=created*

**If invalid details**

{

“message”: “Inputs are not valid”

}, status code: 400

*status code 400=client error*

**If server encounters unexpected error**

{

“message” :” Internal server error ”

}, status code: 500

*status code 500=internal server error*

**Doctor login**

**If the credentials are correct**

{

"message": "Doctor logged in Successfully"

}

status code: 201

*status code 200=Request Successful*

**If the credentials are incorrect**

{

“message”: “Inputs are not valid”

}, status code: 400

*status code 400=client error*

**If the Doctor tries to login as an Admin**

{

“message”: “You are not Authorized”

}, status code: 401

*status code 401=Unauthorized*

**If server encounters unexpected error**

{

“message” :” Internal server error ”

}, status code: 500

*status code 500=internal server error*