An Enterprise Resource Planning System (ERP) For Pharmaceutical Company Distributor

By

Rajan Mahida (CE-063)
Tarang Parikh (CE-083)
Udit Parmar (CE-085)
Khushi Desai (CE-025)

A project submitted
In

Partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
In Computer Engineering

Internal Guide
Prof. P. M. Jadav
Associate Professor
Dept. of Comp. Engg.

External Guide
Mr. Ujjawal Aggrawal
Managing Director
Croods Consolidates Pvt. Ltd.

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University
April 2017

CERTIFICATE

This is to certify that the project work titled

An Enterprise Resource Planning System (ERP) For Pharmaceutical Company Distributor

Is the bonafide work of

Rajan Mahida	**CE-063**	**ID No. 14CEUON115**
Tarang Parikh	**CE-083**	**ID No. 14MHUOS122**
Udit Parmar	**CE-085**	**ID No. 14CEUBS047**
Khushi Desai	**CE-025**	**ID No. 14ECUOG037**

Carried out in the partial fulfillment of the degree of Bachelor of Technology in Computer Engineering at Dharmsinh Desai University in the academic session **December 2016 to April 2017.**

Prof. P. M. Jadav
Associate Professor
Dept. of Comp. Engg.

Dr. C. K. Bhensdadia
Professor and Head of Department
Dept. of Comp. Engg.

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University
April 2017

Croods Engineering

CROODS
ENGINEERING.PVT.LTD

TO WHOMSOEVER IT MAY CONCERN

This is to certify that the following students of Dharmsinh Desai University have carried out his/her field project work at Croods Consolidates, Vadodara:

Name of Student(s):	Rajan Mahida Udit Parmar Tarang Parikh Khushi Desai
Branch /Discipline:	Computer Engineering B.Tech
College /Institute /University:	Dharmsinh Desai University, Nadiad, Gujarat, India, 387001
Project Title:	An Enterprise Resource Planning System (ERP) For Pharmaceutical Company Distributor
Project Guide:	Mr. Ujjawal Aggrawal
Project Duration:	18th Dec 2017 to 25th March 2018
Project Remarks:	**Fully Completed**

Mr. Ujjawal Aggrawal
Managing Director
Croods Consolidates, Vadodara

Date: 03/04/2018

Phone: +918780174844 Email: info@croodstech.com

Acknowledgements

This project work appears in its current form due to the assistance, guidance and support of several people. It is our pleasure to offer sincere thanks to all of them.

We apologize in advance for any omissions. Some code and information has been omitted due to Non-Disclosure Agreement with the Company.
We feel honored to express our sincere gratitude to our Head of Computer Department Dr. C.K. Bhensdadia for providing us an opportunity to work as Interns at Croods Consolidates Pvt. Ltd., Vadodara for our final semester training.

We take immense pleasure in thanking Prof. P. M. Jadav of Computer Department at Dharmsinh Desai University, and our Managing Director Mr. Ujjawal Aggrawal with Project assistance of Mr. Bhavin Pandya at Croods Consolidates Pvt. Ltd. for helping us in any way possible to carry out this project work.

Index

Tables

Images

Chapter 1

Introduction

1.1 Purpose

The purpose of the system is to provide automation for the process of pharmaceuticals management for the distributor. Pharma ERP captures activities performed by different roles in the pharmaceuticals distributor segment on a day-to-day basis and provides enhanced techniques for providing the required information. The document intends to shed light on the features and requirements of this system and provides detailed guidelines for its usage.

1.2 Intended Audience and Reading Suggestions

The document is intended for developers, testers, project managers and all those eligible for using this software in the distributor segment. Section I of this SRS contains a general introduction of the system. Section II, which contains the overall description of the product will be beneficial to all readers. Section III and Section IV contains External Interface Requirements and System Features that will be helpful to developers and testers. Non-functional requirements and other requirements from Section V and VI will be highly useful for all the company staff.

1.3 Product Scope

The scope of this project is limited to the activities performed by a small-scale distributor. The system is limited to a three-tier hierarchy for the functioning of the company, which includes the Chairman, CEO and others in the first tier, senior managers in the second tier and junior managers in the third tier. Professionals of the first tier will have complete access to the entire company’s activities; the privileges for the second tier will be limited to the management of their respective departments and the third tier members will have extremely limited privileges.

Developers at Croods Consolidates Pvt. Ltd. will have administrator access.

1.4 Product Advantages

In the absence of an ERP system, a large manufacturer may find itself with many software applications that cannot communicate or interface effectively with one another. Tasks that need to interface with one another may involve:

1. Integration among different functional areas to ensure proper communication, productivity and efficiency Design engineering (how to best make the product).
2. Order tracking, from acceptance through fulfillment.
3. The revenue cycle, from invoice through cash receipt. Managing inter-dependencies of complex processes bill of materials.
4. Tracking the three-way match between purchase orders (what was ordered), inventory receipts (what arrived), and costing (what the vendor invoiced).
5. The accounting for all of these tasks: tracking the revenue, cost and profit at a granular level.

ERP Systems centralize the data in one place. Benefits of this include:

1. Eliminates the problem of synchronizing changes between multiple systems.
2. Permits control of business processes that cross-functional boundaries.
3. Provides top-down view of the enterprise (no "islands of information").
4. Reduces the risk of loss of sensitive data by consolidating multiple permissions and security models into a single structure.

Some security features are included within an ERP system to protect against both outsider crime, such as industrial espionage, and insider crime, such as embezzlement. A data-tampering scenario, for example, might involve a

disgruntled employee intentionally modifying prices to below-the-breakeven point in order to attempt to interfere with the company's profit or other sabotage. ERP systems typically provide functionality for implementing internal controls to prevent actions of this kind. ERP vendors are also moving toward better integration with other kinds of information security tools.

Chapter 2

About The System

2.1 Overall Description

2.1.1 Product Perspective

The system is an automated version of the distributor management system for the pharmaceuticals industry, which is traditionally done offline, and is intended to obliterate manual tasks, provide quality services and easy access to the user wherever they are, thus saving time and increasing customer satisfaction. The system will also contribute to saving paper and better maintenance of records by complete digitization of the user database.

2.1.2 Product Functions

1. Tier1 professionals

- Members can avail subscription for the company's account
- Members can add, remove or edit employee records; add a subscription account and control privileges for all employee accounts.
- Activities including personnel administration, leaves and attendance and payroll can be handled.
- Individual or group messages can be sent to employees.
- Task and deadline reminders can be set.
- Members can add or remove drug suppliers to the existing list of records.
- Members can place or approve pending requests for drug orders.
- New outlets can be created or existing outlet details can be viewed.
- Inventory stock management can be undertaken.
- Outlet-wise performance can be mapped.
- Members can add or remove new clients to the existing list of records.
- Members can pass orders for clients and track their delivery status.
- Cash inflow and outflow can be tracked and subsequent statistics be generated.

- Invoice for transactions can be generated in PDF format.
- Supplier-wise Drug details can be managed.
- Can make bill payments.
- Pricing can be coordinated.
- GST Calculations can be made.

2. Human Resource Professionals (Tier-2)
    - Members can add, remove or modify records of employees and grant UserID for access to account, as well as control their privileges.
    - Members can assign tasks and set deadlines for peers.
    - One-to-one or group messaging facility included.
    - Members can administer personnel, monitor attendance & leaves and payroll for each employee.

3. Outlet Managers (Tier-2)
    - Can manage inventory stocks of outlets.
    - Generate Invoices of orders.
    - View stats regarding in and out of stocks.
    - Members can add or remove drug suppliers to the existing list of records.
    - Members can place or approve requests for drug orders.
    - Members can add or remove new clients to existing list of records.
    - One-to-one or group messaging facility included.

4. Finance managers (Tier-2)
    - Manage cash flow of the company.
    - Pricing coordination and handling of taxes.
    - Members can pay outstanding bills.
    - Generate invoices of orders.

- Profit management.
- Members can view stats regarding finances of the company.
- One-to-one or group messaging facility can be availed.

5. Tier-3 Professionals

- All tier three members will have only view and report privileges in their respective departments.
- One-to-one or group messaging facility can be availed.

6. Product Administrator

- Can register companies or revoke their access to accounts.
- Can manage subscription stats.
- Can view usage stats.
- Can communicate with subscribers.

2.1.3 User Classes and Characteristics

The various user classes that will use this product are – Product Administrators, Company Heads which are Tier I users, Tier II users which are Senior managers in Human Resource, Finance or Outlet Departments, and Tier III users which comprise the junior managers. The Company Heads will have the widest scope amongst all subscribers, as the system will automate their daily operational requirements. Tier II managers will have full privileges of only their own departments and will be oblivious to all the other activities in the system. Tier III users' scope will be limited to their own departments and will not be able to perform administrative functions – which will be solely reserved for professionals of the highest tier. The Product Administrator will be a part of the software development team and will be able to overlook the usage of all subscribers. She

will also be able to revoke the rights of a particular user or group.

2.1.4 Operating Environment

The software is optimally developed for Desktops. It is set to work correctly with browsers such as Google Chrome, Mozilla Firefox, Opera and Microsoft Edge. For optimum results, use any of these with cookies enabled. For usage on mobile devices, desktop version of this site will be available on any of these..

2.1.5 Design and Implementation Constraints

The various design and implementation constraints are –

1. Standards Compliance

- Report format:
  The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specification.
- Requires at-least 1GB on-board memory.
- Based completely on Windows functionality platform.

The software should be portable and must be inaccessible to unauthorized users.

2. Regulatory Policies

- Copyright will be as per systems. All Rights Reserved. Except as permitted under the Indian jurisdiction copyright act. No part of this software may be reproduced or distributed in any form or by any means, without the prior written permission of the developing organization.
- Some ancillaries including documentation except the user manual will not be available to the customer until a prior execution of the application.
- All the other issues/disputes regarding the terms and conditions shall be liable to the Indian Jurisdiction.

2.1.6 User Documentation

The Software Requirements Specifications manual would be sufficient for total understanding of the system.

2.1.7 Assumptions and Dependencies

1. Assumptions:
   - The application would be robust enough to handle heavy traffic without crashing.
   - The application will be secure.
2. Dependencies:
   - Power Source
   - Systems (User Systems/servers)
   - Communication Mediums (Wired/Wireless)
   - Internet Connection

2.2 External Interface Requirements

2.2.1 User Interfaces

The Interface must be simple and sleek. The user interface includes:

1. Screen Formats/Organization:

   The introductory screen will the default page of the systems. Users will be able to login/register via this page.
2. Data Format:

   The data entered by all users will be alphanumeric.
3. End Messages:

   Appropriate error messages will be displayed in accordance to situations.
4. Download Facility:

   Users will be able to download certain documents through available options.

2.2.2 Hardware Interfaces

The system must support certain input and output devices. Their descriptions are as follows:

1. Monitor
   - Source of Output.
   - To display results.
2. Keyboard
   - Source of Input.
   - To accept data from the user.

2.2.3 Software Interfaces

The main control panels and the operating system, which hosts the algorithms for

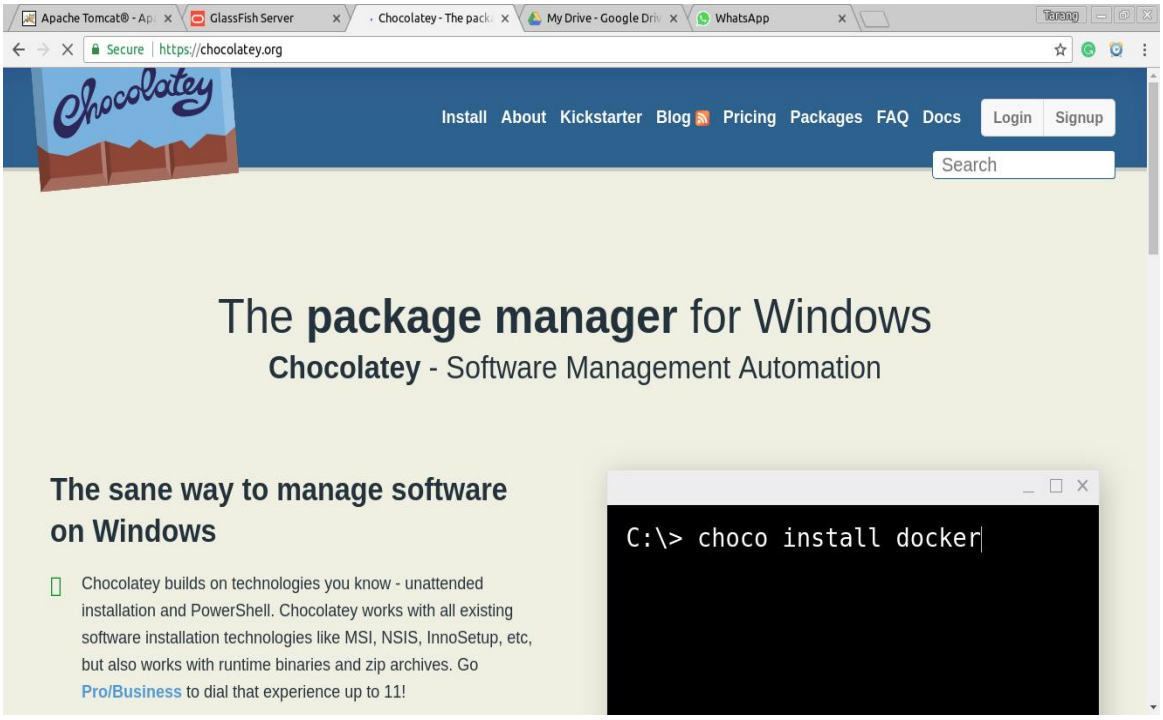
calculating distributed travel and wait time information, support software interface. Additionally, the algorithms define and export system commands for main control panels, and communication mediums. For testing purposes, the software shall be capable of interfacing with software simulators on a PC computer using GUI applications of webpages. The various softwares that have been used for the development of the product are –

IntelliJ IDE with GlassFish server, Gradle, Chocolaty Package Manager as Development Environment. JDK 8.1 is used for developing Servlets and JSPs using the Model-View-Controller Framework. Persistence Layer has been created using Hibernate and MySQL. Front End has been created using HTML, CSS and Bootstrap. Jasper Reports and Apache PDF has been used for generating reports.

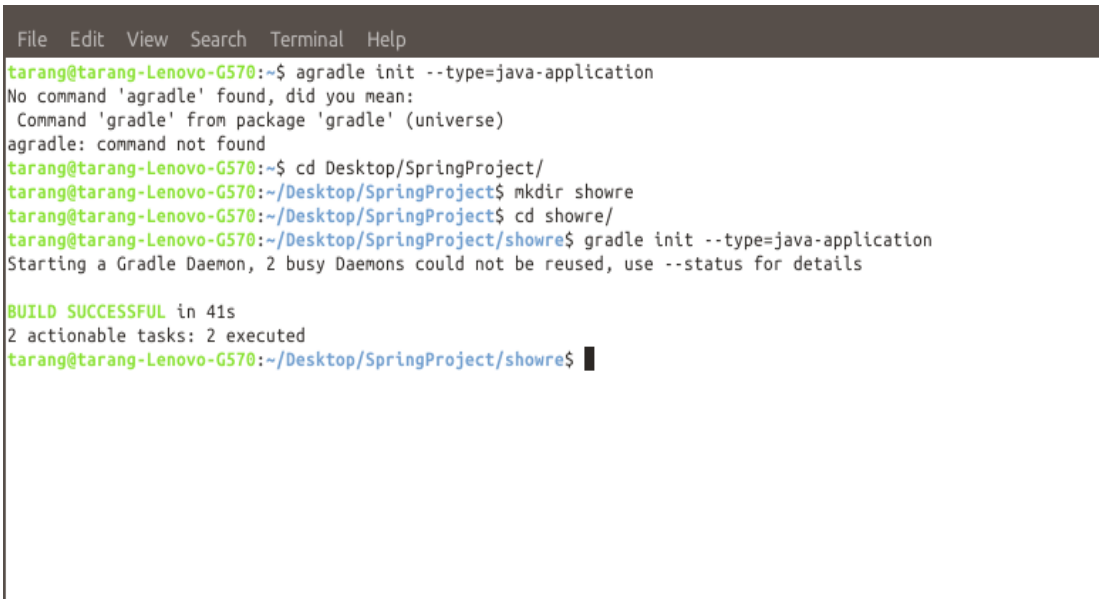
Softwares Used		
SL.	**Name**	**Purpose**
1.	Javax.servlet-api (3.1.0)	For the purpose of creating controllers.
2.	Hibernate-core (5.1.12)	Interacting with MySQL database and his dependency gives us the benefit of not interacting with MySQL via conventional queries.
3.	Javax.servlet.jsp.jstl-api (1.2.1)	Enhancing he rendering of jsp page via tags provided in java standard tag library.
4.	Mysql-connector-java (6.0.6)	Hibernate uses MySQL connecter for interaction with MySQL database.
5.	Jsoup (1.11.2)	Jsoup provides the functionality of he html parsing which can heavily be used for gathering information from various websites
6.	Gson (2.8.2)	Gson is he popular json library for production of json objects from conventional object in java, Jackson library in jersey was replaced by this gson library for json purpose.

7.	Javax.servlet.jsp.jstl (1.2.4)	Subsequent dependency for the jstl functionality.
8.	Jersey-media-multipart (2.25.1)	Uploading file in multi-part file to server via rest api created via jersey.
9.	Jersery-container-servlet (2.25.1)	Jersey functionality can be accessed via his dependency along with servlet api.
10.	Jersery-meda-json-jackson (2.25.1)	Chaining dependency for the purpose of multi file format
Diagram and Reporting Tools Used		
1.	Umlet	Used for UML design
2.	MS Office 2013	Used for reports and documents preparations

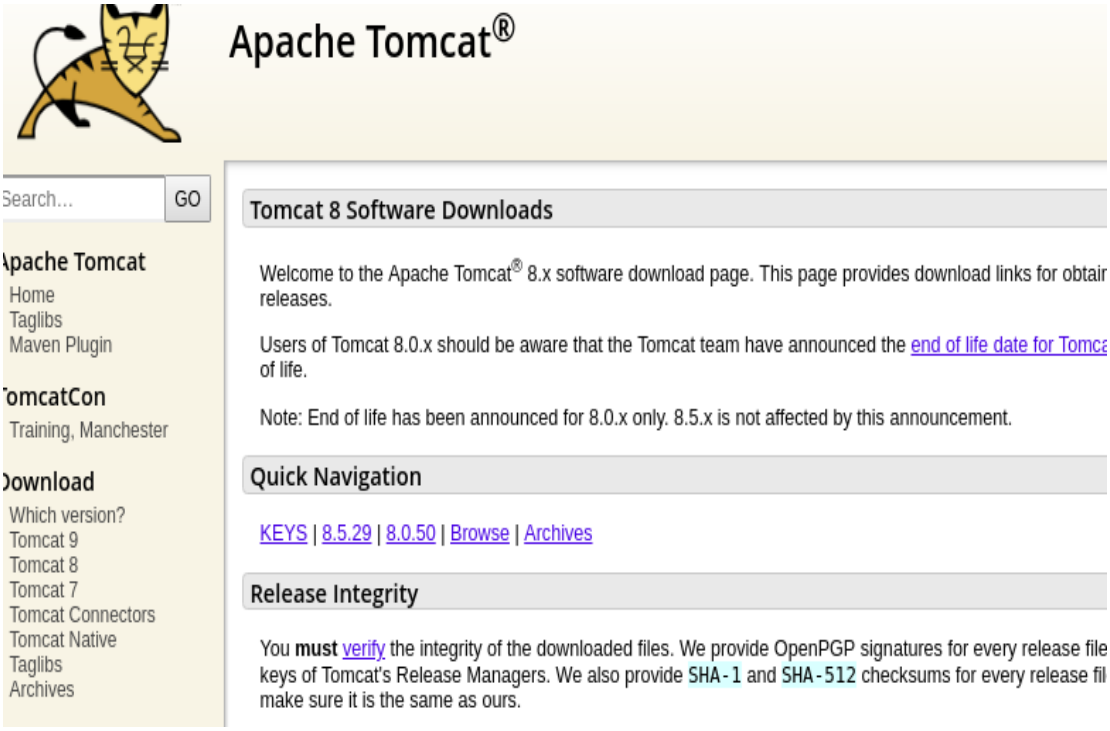
Table 2.1 Software technologies used



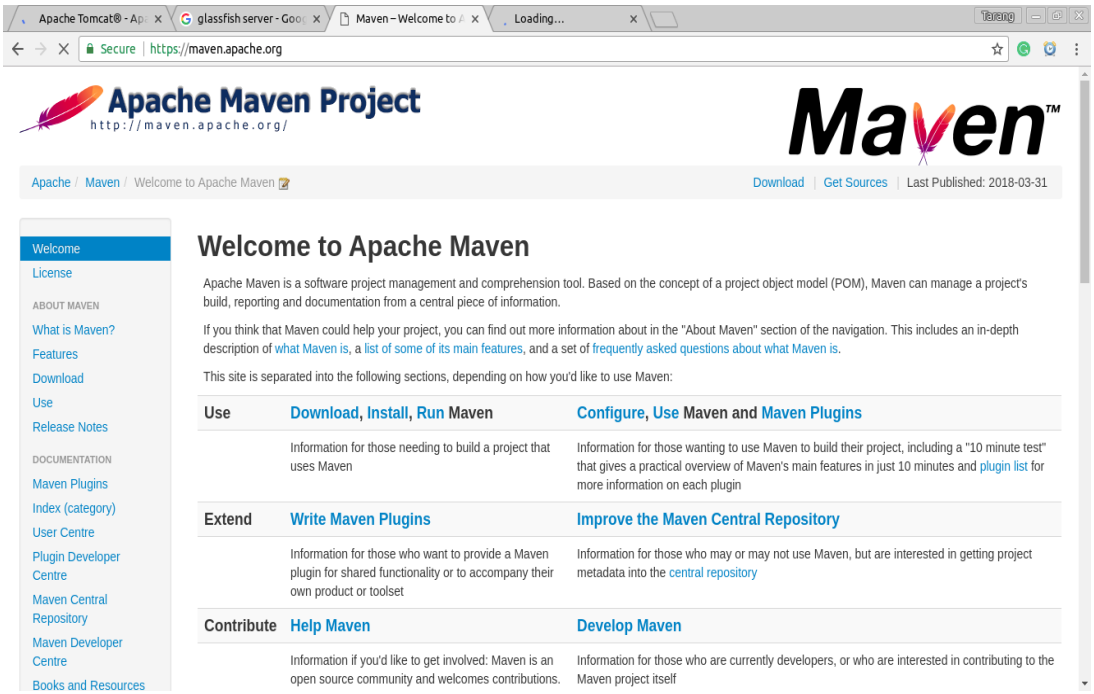
2.1 Chocolatey Package Manager



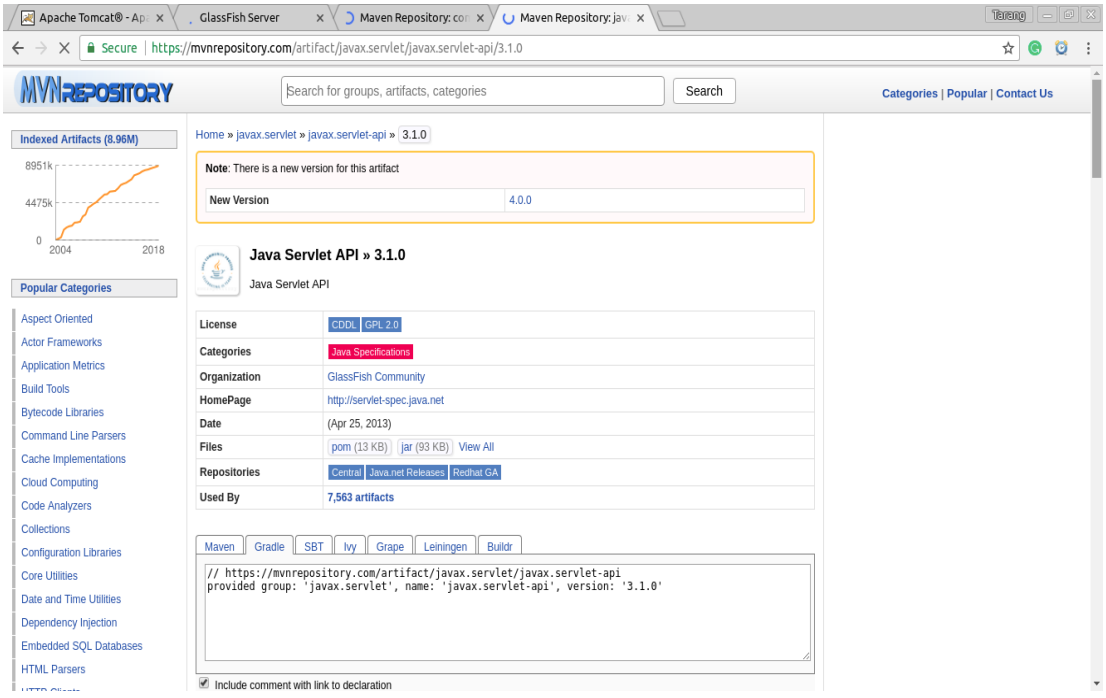
2.2 Build System on Command Prompt



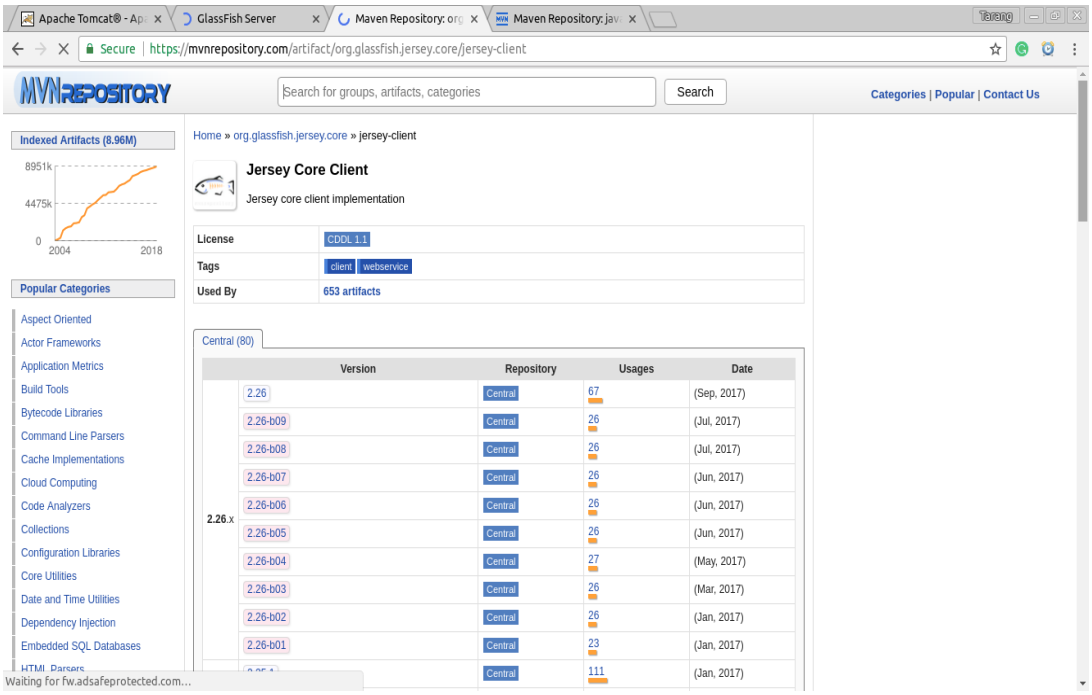
2.3 Apache Tomcat



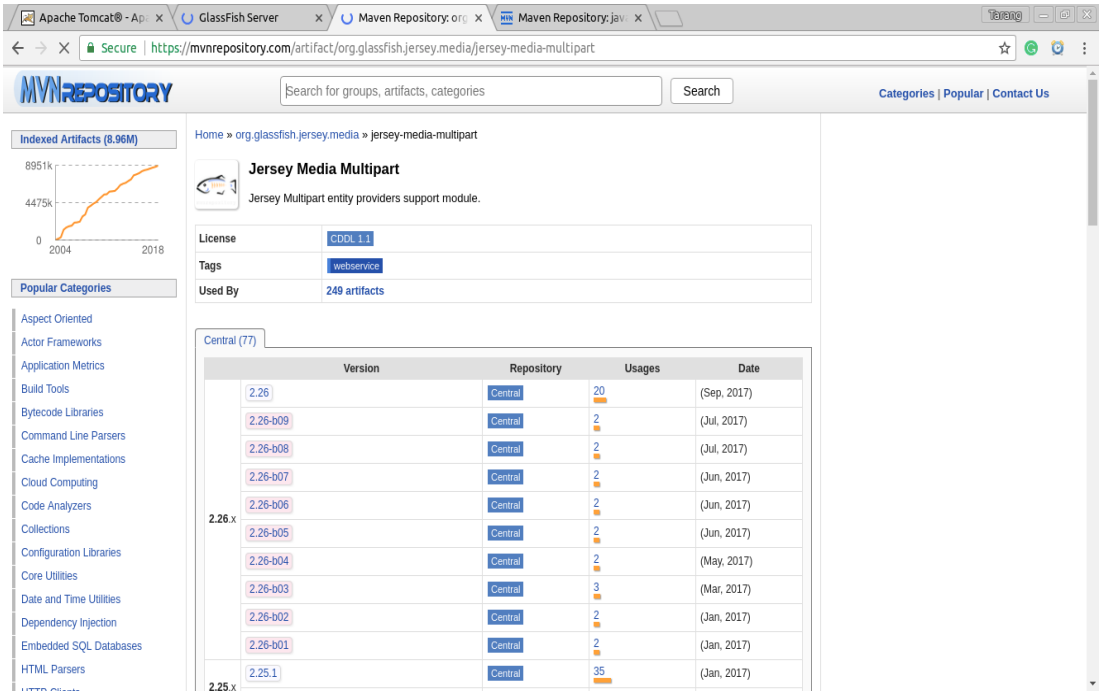
2.4 Apache Maven



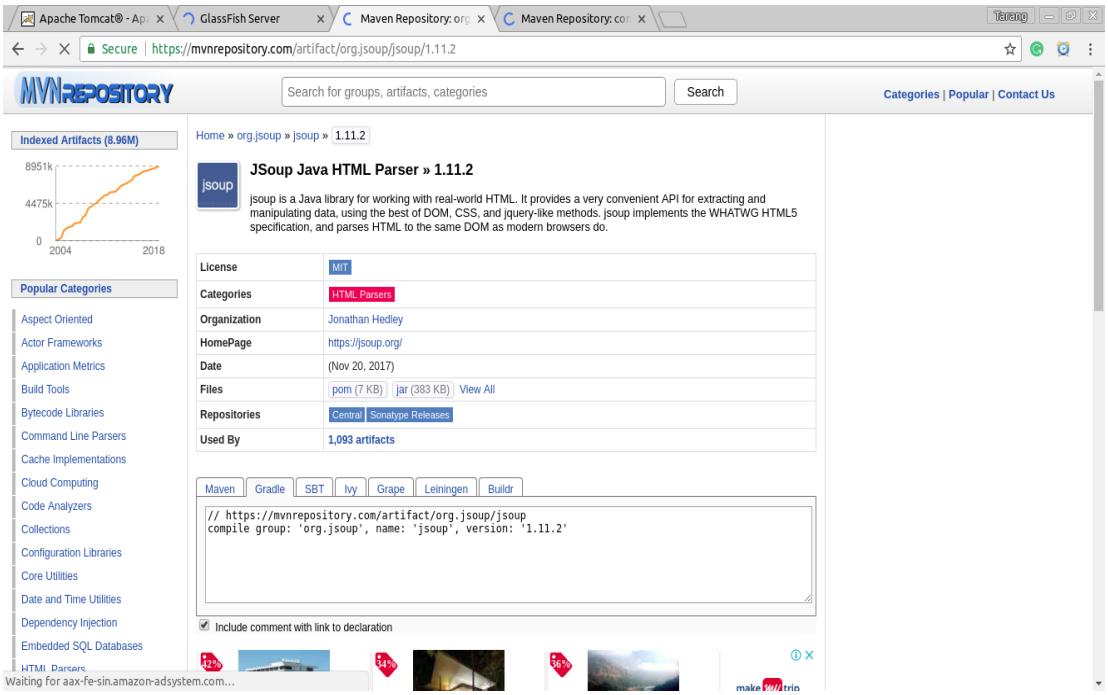
2.5 Maven Repository



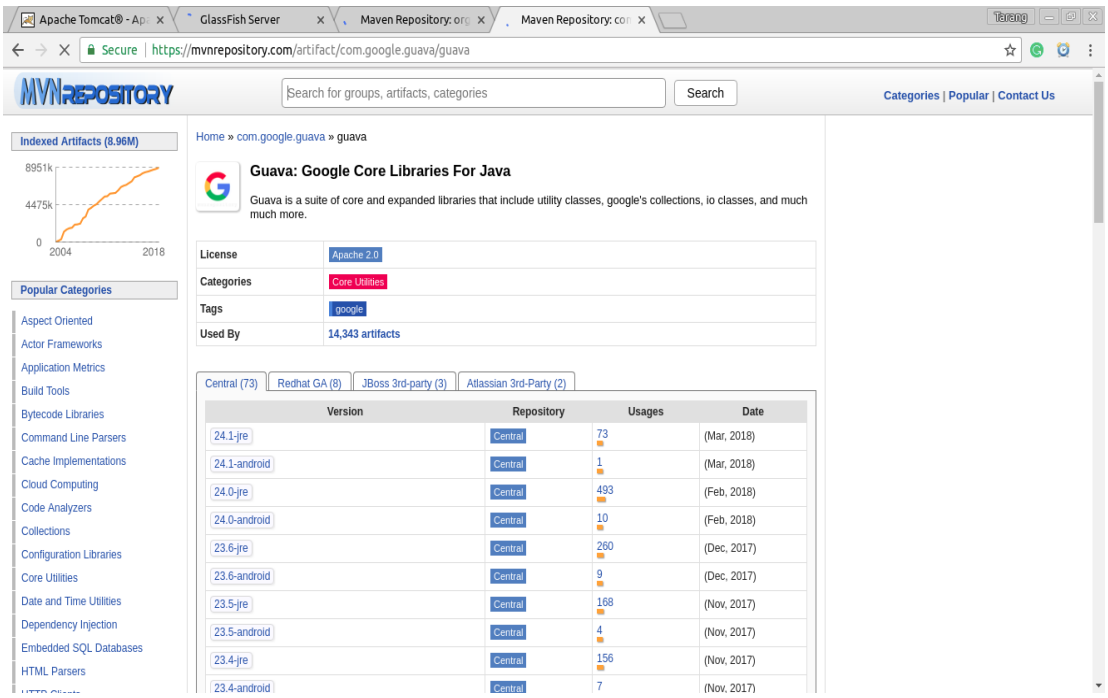
2.6 Jersey Core Client



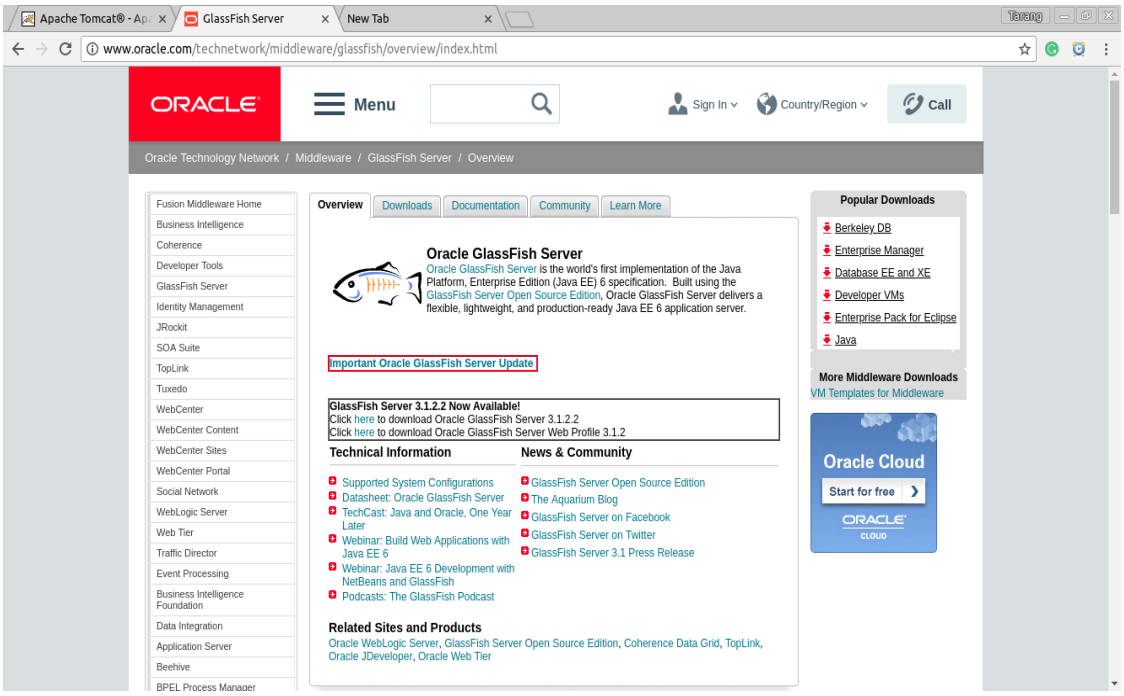
2.7 Jersey Media Multipart



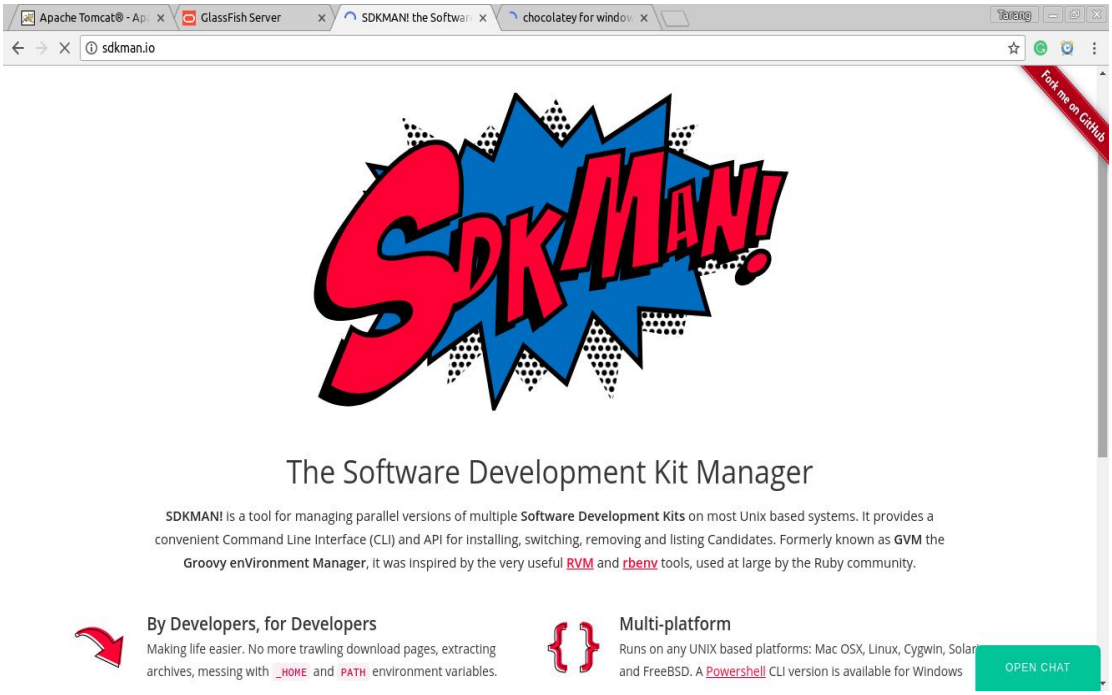
2.8 Jsoup Java HTML Parser



2.9 Guava Library for Java



2.10 Oracle GlassFish Server



2.11 SDK Manager

2.2.4 Communications Interfaces

All system interfaces communicate in order to activate ordered requests. The communication mediums (wired or wireless) are the external interface that communicates with the control panel of the Pharma ERP system. This communication allows for failure messages, and requests to be sent and received by the main system.

2.3 System Features

2.3.1 Registration and Login

2.3.1.1 Description and Priority

The system administrator has the right to register a new company as a subscriber. Once the company account is created, the administrator will create an account for the company head, who will be a Tier I manager and hence will have comprehensive rights regarding the management of his company's activities. The company head can then add employees into the system. Adding employees to the company database doesn't automatically guarantee a subscription account with the system. To give employees access to the system, the company head or the senior Human Resource Manager can add a subscription account of that particular employee in the system. He will also have to set privileges for that employee. Upon creation of the account, the employee will be able to change her password. A facility to recover a user's password is also provided in the event that he forgets her password.

2.3.1.2 Stimulus/Response Sequences

For new registration, company head will have to go to the register page and give in her company's details and then her personal details. They will also require to choose a password for their account. Account verification will occur via security questions that will require user input. In the end of the registration process, users will be able to log into the system using their UserID and password, which will be verified by the system. If the account is inactive for more than 10 minutes, the session will expire and the user will be automatically logged out.

2.3.1.3 Functional Requirements

- REQ-1: Company Details and Company Head's Credentials

  Input: Company Details, Company Head's Details.
  Output: Company ID, UserID & Password for Company Head.
  Processing: Company Head will have to feed in the Company Details and his own details the first time he registers the company on the site.

- REQ-2: UserID

  Input (For Registration): Employee Details .
  Output (For Registration): UserID and Password.
  Processing: Company Head or HR manager can create accounts for others.

  Input (For Login): UserID and Password
  Output (For Login): Successful login
  Processing: Valid UserID and Password will enable the user to access their account. In case the password is lost, it can be retrieved via the Forget Password link after a series of verification steps.

2.3.2 Place order to Supplier

2.3.2.1 Description and priority

Company Head or Outlet manager can place or approve an order for restocking inventory items. If supplier does not exist in the Company's records, then the

authority will have to create a new supplier record and after proper sifting through its existing inventory, will have to choose all the required items. Once the stock is delivered, payment will be automatically sanctioned and deductions will be made from the Company's account post calculation of all the necessary taxes. Any junior employee can also request an order, but it will only be passed after approval from Company Head or Outlet Manager.

2.3.2.2 Stimulus/Response Sequences

User will have to request a place new order form and select appropriate supplier for the same. Once supplier info is available, User can select items from the manufacturer's inventory. The order will be directly placed if it is by a Company Head or Outlet Manager, else will be queued until further approval. Once stocks are delivered, amount will be deducted and finance manager will be notified.

2.3.2.3 Functional Requirements

- REQ-1: Supplier Details

  Input: Supplier Name
  Output: Supplier ID and its inventory
  Processing: Based on the input, the system will search for matching results in the database and display the results on the screen. Users will be able to retrieve further information by clicking on one of the search result.

- REQ-2: Stocks

  Input: Item type and Quantity
  Output: Selection
  Processing: The input will be matched with the supplier records and if it exists,

appropriate details will be prepared.

2.3.3 Pass Client Order

2.3.3.1 Description and priority

Clients, which include Retailers or Medical Representatives will be able to request for stocks according to their requirements. The Company Head, after due examination of existing stocks and their expiry information will pass the order. Once the order is passed, Inventory & Company Account information will be updated and Finance manager will be notified. Taxes and profit margins will also be duly calculated. If the products reach expiry, they will be removed from current stocks.

2.3.3.2 Stimulus/Response Sequences

Users will have to click on Add Client tab and enter client details. After that, users will have to click on Create Order tab and select the client from the list of available clients. Inventory Items and their respective quantities will have to be added. Upon passage of order, an invoice will be generated in PDF format. After Payment by client is done, Company accounts will be updated and Finance Manager will be notified.

2.3.3.4 Functional Requirements

- REQ-1: Client Details

  Input: Required details of Client.
  Output: Successful add acknowledgement.
  Processing:

The system will have to check the validity of all the input details before further processing.

- REQ-2: Item Details and Quantity

Input: Required details of Item and its quantity.
Output: Successful add acknowledgement.
Processing:
The system will have to check the validity of all the input details before further processing.

2.3.4 Messaging and Task Reminders

2.3.4.1 Description and priority

Users can engage in one-to-one and one-to-many messaging. Each user will have a unique UserID, which will be in an email format. Higher Tier managers can send Task Reminders to Lower Tier Managers and can specify deadlines in them. Reminders can be auto-generated as well, depending on the scenario, such as imminent expiry date of drug or other such examples.

2.3.4.2 Stimulus/Response Sequences

Users can directly send messages to one or more users from their accounts. Auto-Generated Reminders will be sent in case of approaching deadlines or dates.

2.3.4.3 Functional Requirements

- REQ-1: Message support

Input: Required message, UserID.
Output: Message will be sent to the intended user(s).
Processing: User will need to fill the above-mentioned details and the e-mail will be sent through mail server. Mail API will be used to build the service.

2.3.5 Generate Invoice/Record and Reports

2.3.5.1 Description and priority

Invoices will be generated when stocks are delivered to intended clients post the approval of their order. Subsequent updation of Company Accounts will also be carried out when payment is made. Records are generated when a supplier delivers stocks to the company or when outstanding bills are paid. Timely reports can also be generated to view inflow and outflow of goods and cash.

2.3.5.2 Stimulus/Response Sequences

Users can directly use the Generate Report tab to generate timely reports. Invoices can be generated from Transactions tab.

2.3.5.3 Functional Requirements

- REQ-1: Generate Report

  Input: Time Frame
  Output: Downloadable report in PDF format.
  Processing: Details will be fetched from the database and be used for generating the report.

2.3.6 Make Outstanding Payments

2.3.6.1 Description and priority

Company Head or Finance Head will be able to make outstanding payments incurred to the company in the form of Maintenance Costs, Transportation Costs and many others. Reminders will be set for payments. Records will be directly generated on successful payment of costs. Company Accounts will be updated post payment. This facility is not available to Tier III members.

2.3.6.2 Stimulus/Response Sequences

Users can directly go to the make payments tab and enter the required details to complete the task.

2.3.6.3 Functional Requirements

- REQ-1: Payment Details

  Input: Payment Details and Amount.
  Output: Downloadable record in PDF Format.
  Processing: User will need to fill the above-mentioned details and record will be generated post money deductions.

2.3.7 View Statistics

2.3.7.1 Description and priority

Usage statistics will be visible to System Administrator and Company Head on their dashboards. Statistics will be different for both users, and it will gauge the overall activity of the system's users in a particular time frame. Only Tier I and Tier II users will get this facility.

2.3.7.2 Stimulus/Response Sequences

Statistics will be available automatically on the User's dashboard.

2.4 Other Non-Functional Requirements

2.4.1 Performance Requirements

The Pharma ERP System shall be built upon an internet connection of server. The processor must be capable of handling real-time functionality activated by the defined users and communication medium. In addition, the system must be safety-critical. All failures reported by the communication medium must be handled instantaneously to allow for user and system safety. The software shall have a response time variable of 5 seconds, based on signal or web based inputs, which if exceeded, the software shall recognize an error and take corrective action. Application shall show no visible deterioration in response time as the number of users increases.

2.4.2 Safety Requirements

The product is 100% safe from the design point-of-view. However, mishandling of Unique ID and password may put the user in jeopardy.

2.4.3 Security Requirements

- It must be ensured that access will be provided to the authorized persons through Email ID and password.
- Network security will be provided by the use of firewalls.
- Checks can be performed at regular intervals to ensure data integrity.

2.4.4 Software Quality Attributes

1. Reliability

- Application shall be available 24 hours a day, 7 days a week
- Application shall always provide real time information about User/Admin/Application Itself
- Application shall be robust enough to have a high degree of fault tolerance. The system should not crash in case of invalid input and shall identify the invalid input and produce a suitable error message.
- Application shall be able to recover from hardware failures, power failures and other natural catastrophes and rollback the databases to their most recent valid state.

2. Usability

- Application shall provide an easy-to-use graphical interface similar to other existing systems so that the users do not have to learn a new style of interaction.
- The web interface should be intuitive and easily navigable. Users should be able to understand the menu and options provided.
- Any notification or error messages generated by application shall be clear, succinct, polite and free of jargon.

3. Availability

When in normal operating conditions, request by a user for a service shall be handled within seconds. Immediate feedback of the systems activities shall be communicated to the user by link page clicked. At peak system load, individual users at either the server in the security office, at the links or inside the banking system shall not experience any delay in the service response to their commands in a very short time. The system is available

100% for the user and is used 365 days round the clock.

4. Integrity

- Only system administer has the right to change system parameters. The system should be secure and must use encryption to protect the databases.
- Users need to be authenticated before having access to any personal data.

5. Maintainability

- There shall be design documents describing the internal works of the software. There shall be an access on the control panel and servers for the purpose of upgrading the software or flashing any firmware.

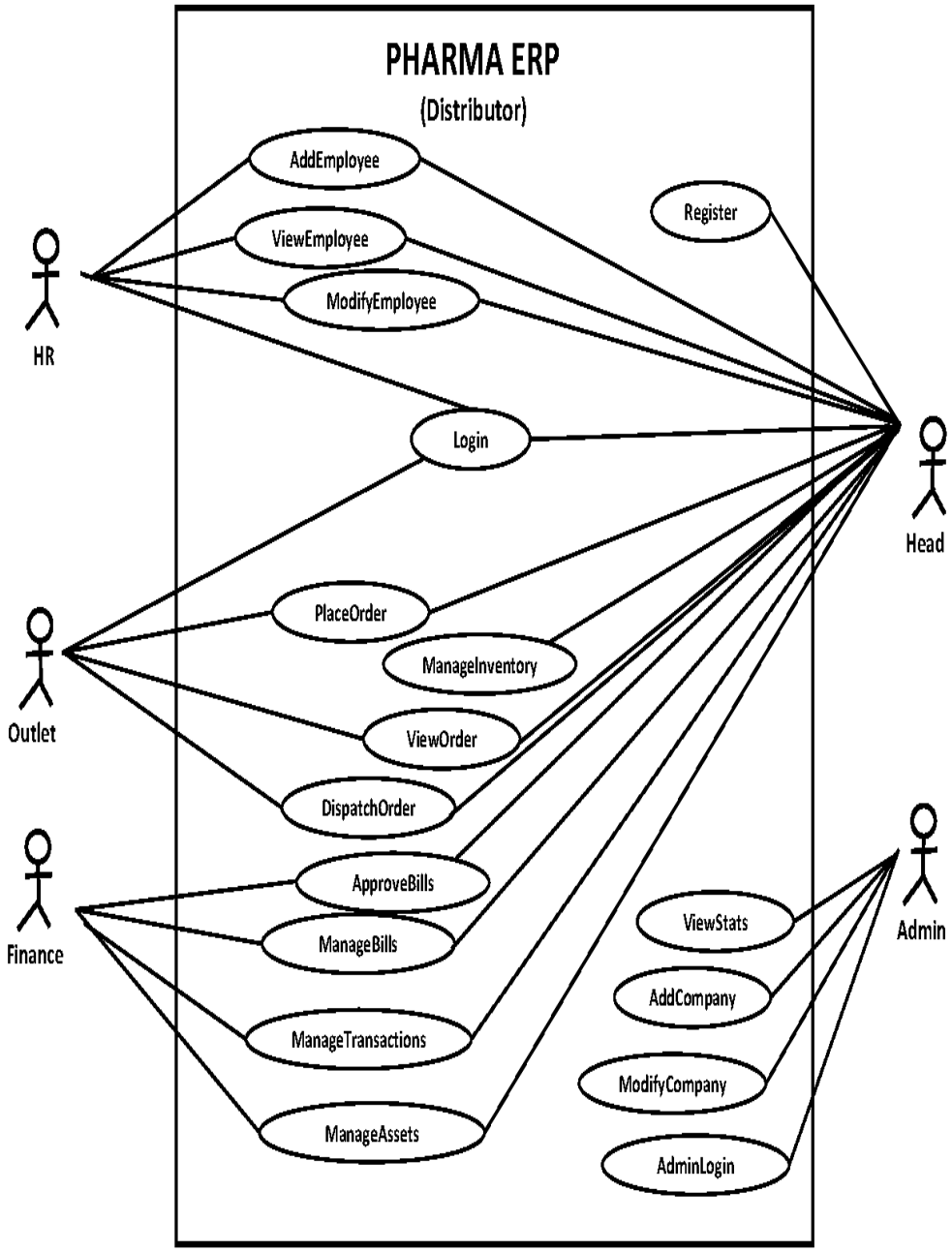
6. Portability

- There are no portability requirements.

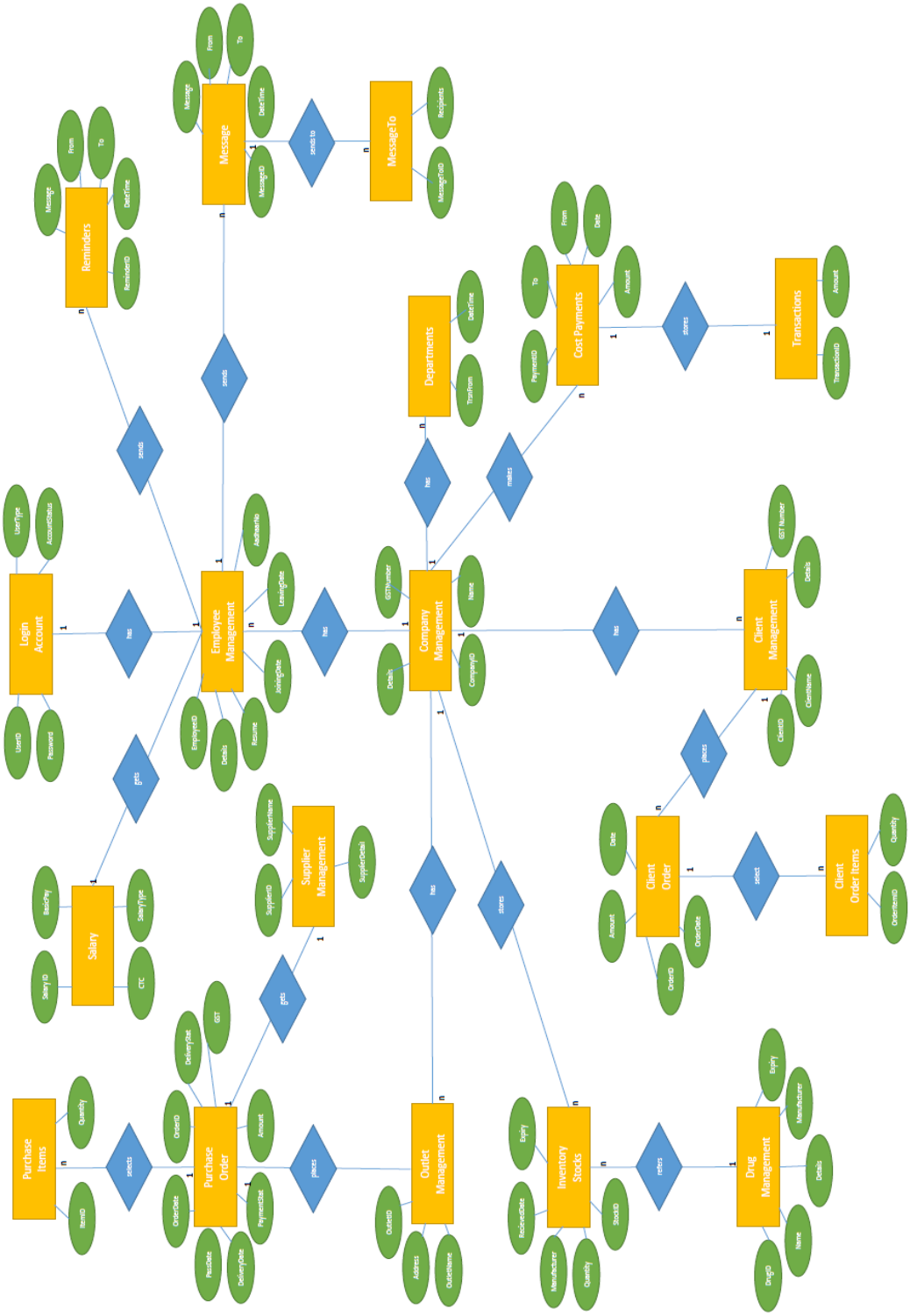
Chapter 3

Analysis

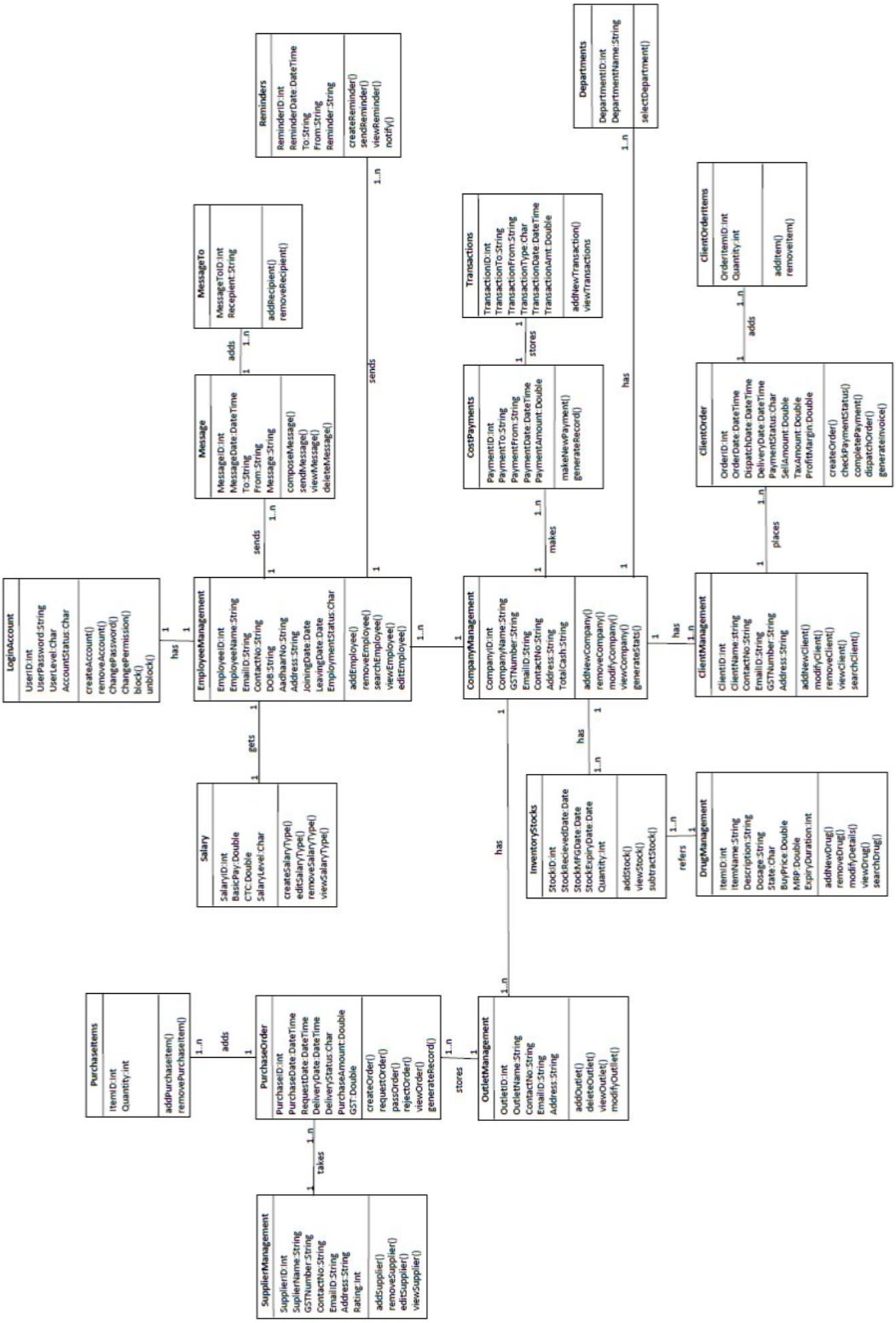
3.1 Use Case Diagram



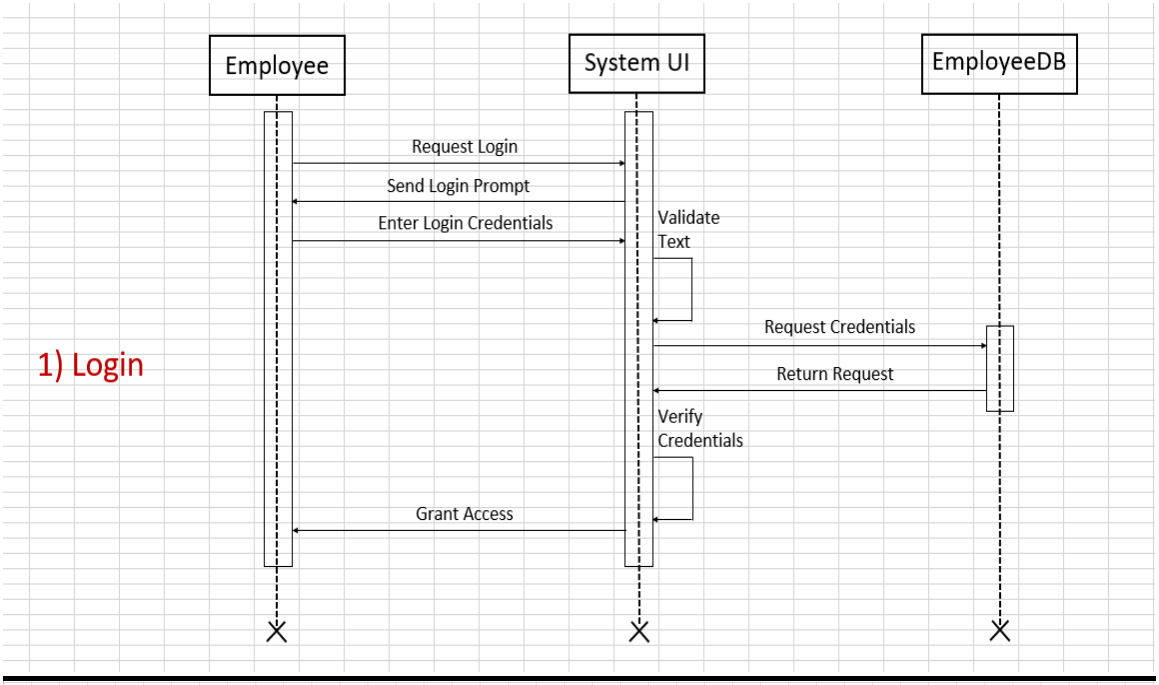
3.2 ER Diagram

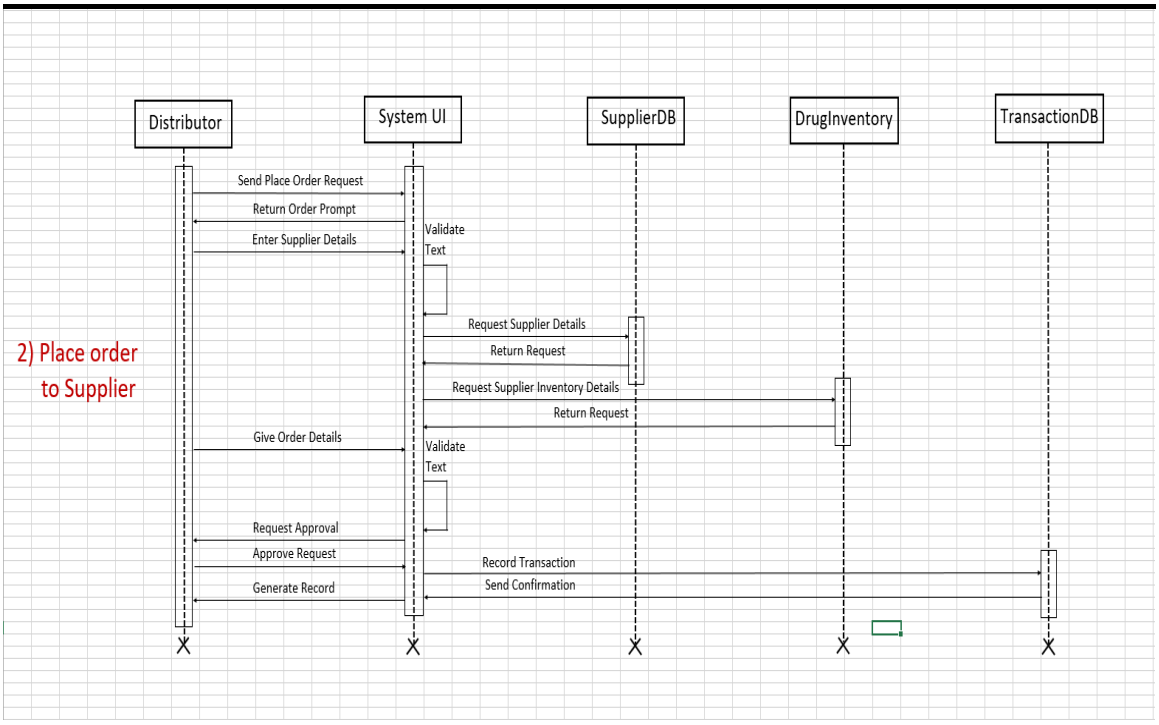


3.3 Class Diagram

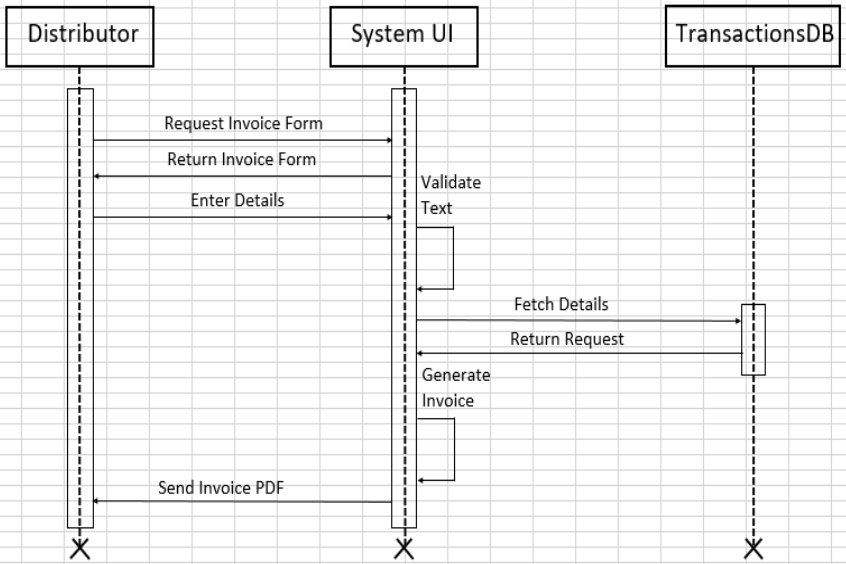


3.4 Sequence Diagram

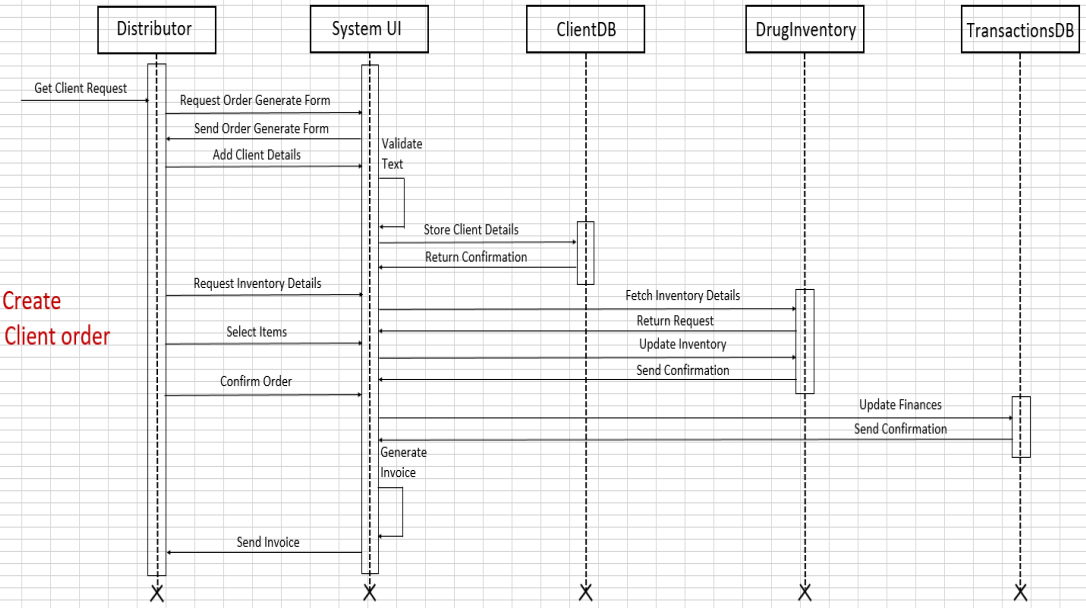


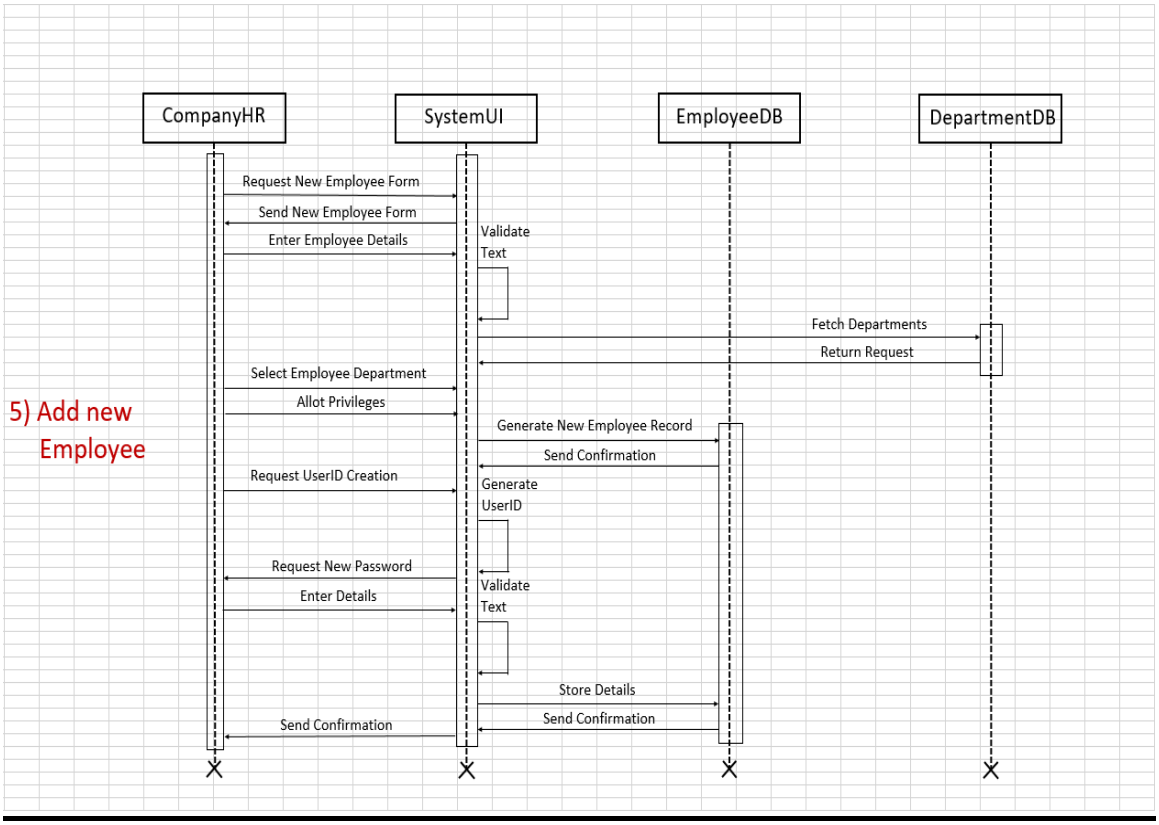


3)Generate Invoice



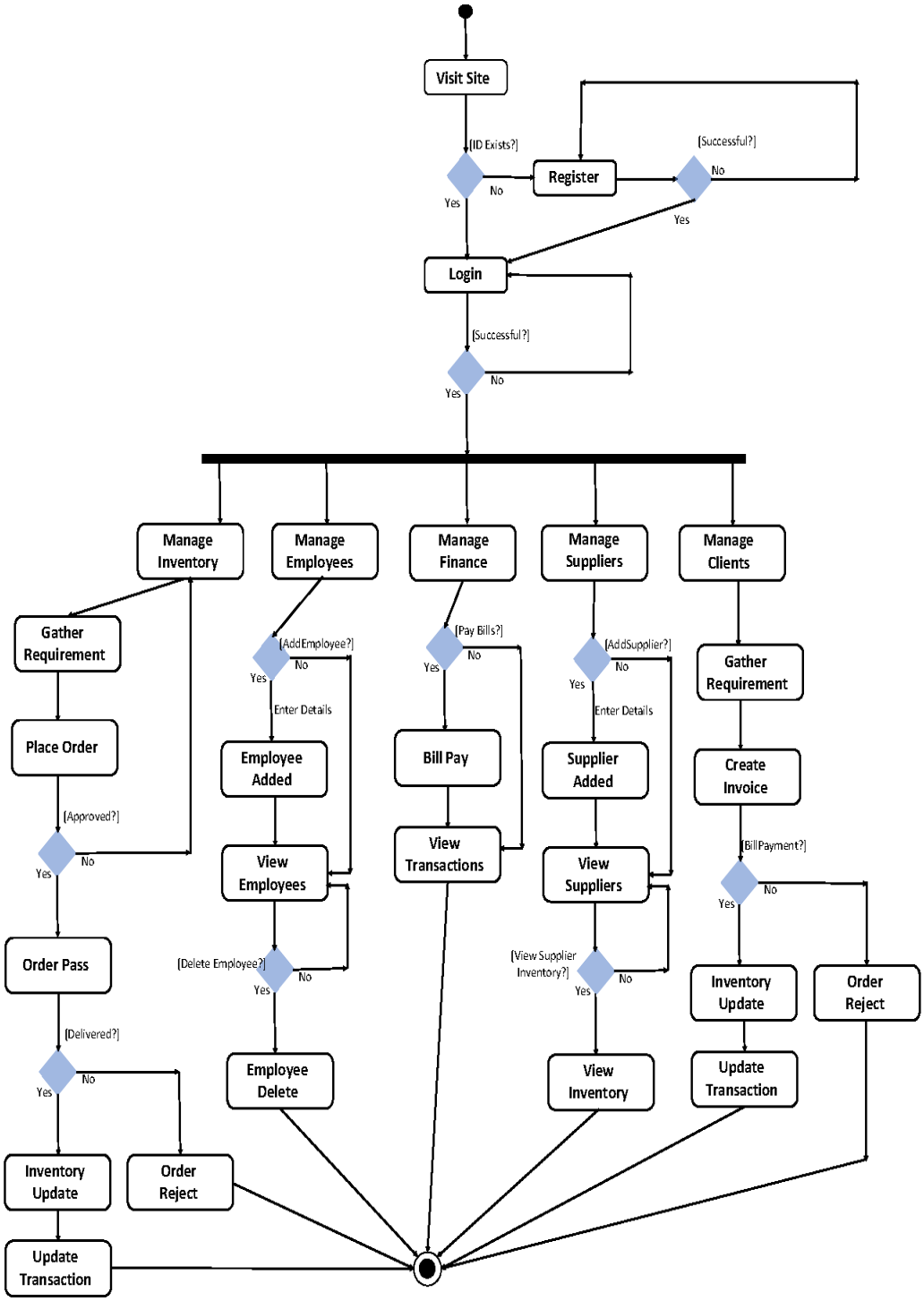
4) Create Client order





5) Add new Employee

3.5 Activity Diagram



Chapter 4

Design

SL.	Field Name	Data Type	Constraint (PK, FK, Unique, Null/Not NULL)
1	Company Id	Int	Primary Key, Foreign key
2	Company name	Varchar	
3	Estd Year	Int	
4	GST Number	Varchar	
5	Contact No. 1	Varchar	
6	Contact No. 2	Varchar	
7	Email ID	Varchar	Unique, Not Null
8	Total Cash	Double	
9	Outlet ID	Int	Primary key, Foreign Key
10	Outlet name	Varchar	

11	Address Line	Varchar	
12	City	Varchar	
13	State	Varchar	
14	Employee ID	Int	Primary Key, Foreign Key
15	Employee Name	Varchar	
16	Designation	Varchar	
17	Date of birth	Date	
18	Aadhar No.	Varchar	Unique, Not Null
18	Joining Date	Date	
19	Leaving Date	Date	
20	Current Status	Char	

21	Salary ID	Int	Primary Key, Foreign key, Unique, Not Null
22	Basic Pay	Double	
23	CTC	Double	
24	Salary Level	Char	
25	Department ID	Int	Primary Key, Foreign key
26	Department Name	Varchar	
27	User ID	Int	Unique, Not Null
28	Password	Varchar	Unique, Null
29	User Level	Char	
30	Account Status	Char	
41	Supplier ID	Int	Primary Key, Foreign Key, Unique, Not Null

42	Supplier Name	Varchar	
43	Owner Name	Varchar	
44	Rating	Int	
45	Purchase ID	Int	Primary Key, Foreign Key, Unique, Not Null
46	Purchase Date	Date	
47	Purchase Request	Char	
48	Request date	Date	
49	Delivery Status	Char	
50	Delivery Date	Date	
51	Purchase Amount	Double	

52	GST	Double	
53	Item ID	Int	Primary Key, Unique, Not Null
54	Quantity	Int	
55	Drug ID	Int	Primary Key, Foreign Key, Unique, Not Null
56	Item Name	Int	
57	Item Description	Varchar	
58	Dosage	Varchar	
59	Drug Type	Char	
60	Buy Price	Double	
61	MRP	Double	
62	Expiry Duration	Int	

63	Stock ID	Int	Primary Key, Unique
64	Stock Received date	Date	
65	Stock mfg date	Date	
66	Stock Expiry date	Date	
67	Stock Quantity	Int	
68	Client ID	Int	Primary Key, Foreign Key
69	Client Name	Varchar	
70	Order ID	Int	Primary Key, Foreign Key, Unique, Not Null
71	Order Date	Date	
72	Dispatch Date	Date	

73	Delivery Date	Date	
74	Payment Status	Char	
75	Sell Amount	Double	
76	Tax Amount	Double	
77	Profit margin	Double	
78	Order Item ID	Int	Primary Key, Unique, Not Null
79	Quantity	Int	
80	Payment ID	Int	Primary Key
81	Payment Name	Varchar	
82	Payment To	Varchar	
83	Payment	Double	

	Amount		
84	Payment Date	Date	
85	Transaction ID	ID	Primary Key, Unique, Not Null
86	Payment To	Varchar	
87	Payment From	Varchar	
88	Payment Type	Char	
89	Payment Amount	Double	
90	Payment Date and Time	Date and Time	
91	Reminder ID	Int	Primary Key
92	Reminder Date	Date	

93	Reminder Time	Time	
94	Reminder To	Varchar	
95	Reminder From	Varchar	
96	Reminder Message	Varchar	
97	Message ID	Int	Primary Key, Foreign Key, Unique, Not Null
98	Message Date	Date	
99	Message Time	Time	
100	Message To	Varchar	
101	Message From	Varchar	
102	Message	Varchar	
103	Message To ID	Int	Primary Key, Unique, Not Null

104	Recipient	Varchar	

Table 4.1 Data Dictionary

4.1 Database Schema

Database schema		
CompanyManagement		**Foreign Key**
CompanyID	Int (Primary Key)	
CompanyName	Varchar	
EstdYear	Int	
GSTNumber	Varchar	
ContactNo1	Varchar	
ContactNo2	Varchar	
EmailID	Varchar	
TotalCash	Double	
OutletManagement		**Foreign Key**
OutletID	Int (Primary Key)	CompanyID
OutletName	Varchar	
ContactNo1	Varchar	
ContactNo2	Varchar	
AddressLine	Varchar	
City	Varchar	
State	Varchar	
EmployeeManagement		**Foreign Key**
EmployeeID	Int (Primary Key)	CompanyID
EmployeeName	Varchar	DepartmentID
Designation	Varchar	OutletID
DOB	Date	SalaryID
EmailID	Varchar	
ContactNo1	Varchar	
ContactNo2	Varchar	
AddressLine1	Varchar	**Directory Data**
City	Varchar	
State	Varchar	Resume
AadhaarNo	Varchar	
JoiningDate	Date	
LeavingDate	Date	
CurrentStatus	Char (Current/Ex)	
Salary		**Foreign Key**
SalaryID	Int (Primary Key)	CompanyID
BasicPay	Double	DepartmentID
CTC	Double	
SalaryLevel	Char (Head, Senior, Junior)	
Departments		**Foreign Key**
DepartmentID	Int (PrimaryKey)	CompanyID
DepartmentName	Varchar (Management, HR, Finance, Outlet)	

LoginAccount		Foreign Key
UserID	Int (Primary Key)	EmployeeID
Password	Varchar	
UserLevel	Char (Head, Senior, Junior)	
AccountStatus	Char (Active, Blocked)	
SupplierManagement		**Foreign Key**
SupplierID	Int (Primary Key)	CompanyID
SupplierName	Varchar	
GSTNumber	Varchar	
OwnerName	Varchar	
EmailID	Varchar	
ContactNo1	Varchar	
ContacNo2	Varchar	
AddressLine	Varchar	
City	Varchar	
State	Varchar	
Rating	Int (1-10)	
PurchaseOrder		**Foreign Key**
PurchaseID	Int (Primary Key)	OutletID
PurchaseDate	Date	EmployeeID
PurchaseRequest	Char (Awaiting, Passed, Rejected)	
RequestDate	Date	
DeliveryStatus	Char (Delivered, Waiting)	
DeliveryDate	Date	
PurchaseAmount	Double	
GST	Double	
PurchaseItems		**Foreign Key**
ItemID	Int (Primary Key)	PurchaseID
Quantity	Int	DrugID
DrugManagement		**Foreign Key**
DrugID	Int (Primary Key)	SupplierID
ItemName	Int	
ItemDescription	Varchar	
Dosage	Varchar	
DrugType	Char (Tablet, Capsule)	
State	Char (Solid, Liquid)	
BuyPrice	Double	
MRP	Double	
ExpiryDuration	Int (Months)	
InventoryStocks		**Foreign Key**
StockID	Int (Primary Key)	DrugID
StockRecievedDate	Date	OutletID
StockMfgDate	Date	
StockExpiryDate	Date	
StockQuantity	Int	

ClientManagement		Foreign Key
ClientID	Int (Primary Key)	CompanyID
ClientName	Varchar	
GSTNumber	Varchar	
ContactNo1	Varchar	
ContactNo2	Varchar	
EmailID	Varchar	
AddressLine	Varchar	
City	Varchar	
State	Varchar	
ClientOrder		**Foreign Key**
OrderID	Int (Primary Key)	ClientID
OrderDate	Date	CompanyID
DispatchDate	Date	
DeliveryDate	Date	
PaymentStatus	Char (Done, Awaiting)	
SellAmount	Double	
TaxAmount	Double	
ProfitMargin	Double	
ClientOrderItems		**Foreign Key**
OrderItemID	Int (Primary Key)	OrderID
Quantity	Int	DrugID
CostPayments		**Foreign Key**
PaymentID	Int (Primary Key)	CompanyID
PaymentName	Varchar (store ID)	EmployeeID
PaymentTo	Varchar (store ID)	
PaymentAmount	Double	
PaymentDate	Date	
Transactions		**Foreign Key**
TransactionID	Int (Primary Key)	CompanyID
PaymentTo	Varchar (store ID)	
PaymentFrom	Varchar (store ID)	
PaymentType	Char (CostPayment, Supplier, Client)	
PaymentAmount	Double	
PaymentDateTime	DateTime	
Reminders		**Foreign Key**
ReminderID	Int (Primary Key)	CompanyID
ReminderDate	Date	
ReminderTime	Time	
To	Varchar (Store ID)	
From	Varchar (Store ID)	
Message	Varchar	

Message		Foreign Key
MessageID	Int (Primary Key)	CompanyID
MessageDate	Date	
MessageTime	Time	
To	Varchar (Store ID)	
From	Varchar (Store ID)	
Message	Varchar	
MessageTo		**Foreign Key**
MessageToID	Int (Primary Key)	MessageID
Recepient	Varchar (Store ID)	

4.2 Code Navigation

4.2.1 Class Hierarchy

- java.lang.Object
  - javax.ws.rs.core.Application
    - REST.**Application**
  - VO.**BreadCrumbVO**
  - DAO.**ClientDAO**
  - DAO.**ClientItemDAO**
  - DAO.**ClientOrderDAO**
  - REST.**ClientOrderItemREST**
  - VO.**ClientOrderItemsVO** (implements java.io.Serializable)
  - REST.**ClientOrderREST**
  - VO.**ClientOrderVO** (implements java.io.Serializable)
  - REST.**ClientREST**
  - VO.**ClientVO** (implements java.io.Serializable)
  - VO.**CommonVO**
  - DAO.**CompanyDAO**
  - REST.**CompanyREST**
  - VO.**CompanyVO** (implements java.io.Serializable)
  - DAO.**CostPaymentsDAO**
  - REST.**CostPaymentsREST**
  - VO.**CostPaymentsVO** (implements java.io.Serializable)
  - REST.**CrosFilter** (implements javax.ws.rs.container.ContainerResponseFilter)
  - DAO.**DepartmentDAO**
  - REST.**DepartmentREST**
  - VO.**DepartmentVO** (implements java.io.Serializable)

- DAO.**DocumentDAO**
- REST.**DocumentREST**
- VO.**DocumentVO**
- DAO.**DrugDAO**
- REST.**DrugREST**
- VO.**DrugVO** (implements java.io.Serializable)
- DAO.**EmployeeDAO**
- REST.**EmployeeREST**
- VO.**EmployeeVO** (implements java.io.Serializable)
- javax.servlet.GenericServlet (implements java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig)
    - javax.servlet.http.HttpServlet
        - Controller.ConfirmMail.**ConfirmMail**
        - Controller.Dashboard.**Dashboard**
        - Controller.Employee.Portal.**Employee**
        - Controller.Employee.Registration.**EmployeeRegistration**
        - Controller.Finance.**Finance**
        - Controller.Home.**Home**
        - Controller.Inventory.**Inventory**
        - Controller.LockScreen.**LockScreen**
        - Controller.Login.**Login**
        - Controller.Logout.**Logout**
        - Controller.Message.**Message**
        - Controller.Message.**MessageCompose**
        - Controller.Message.**MessageInbox**
        - Controller.Message.**MessageRead**
        - Controller.Message.**MessageSent**
        - Controller.Outlets.**Outlets**
        - Controller.RecoverPassword.**RecoverPassword**

    - Controller.Register.**Register**
    - Controller.Staff.**Staff**
    - Controller.Supplier.**Supplier**
    - Controller.**Tester**
- DAO.**InventoryStocksDAO**
- REST.**InventoryStocksREST**
- VO.**InventoryStocksVO** (implements java.io.Serializable)
- Listener.**Listener** (implements javax.servlet.http.HttpSessionAttributeListener, javax.servlet.http.HttpSessionListener, javax.servlet.ServletContextListener)
- DAO.**MessageDAO**
- REST.**MessageREST**
- DAO.**MessageToDAO**
- REST.**MessageToREST**
- VO.**MessageToVO** (implements java.io.Serializable)
- VO.**MessageVO** (implements java.io.Serializable)
- DAO.**OutletDAO**
- REST.**OutletREST**
- VO.**OutletVO** (implements java.io.Serializable)
- DAO.Global.**PersistenceDAO**
    - DAO.Global.**DBOperationDAO**
- DAO.**PurchaseItemDAO**
- REST.**PurchaseItemREST**
- VO.**PurchaseItemVO** (implements java.io.Serializable)
- DAO.**PurchaseOrderDAO**
- REST.**PurchaseOrderREST**
- VO.**PurchaseOrderVO** (implements java.io.Serializable)
- DAO.**ReminderDAO**

- REST.**ReminderREST**
- VO.**ReminderVO** (implements java.io.Serializable)
- DAO.**SalaryDAO**
- REST.**SalaryREST**
- VO.**SalaryVO** (implements java.io.Serializable)
- VO.**StateCity**
- DAO.**SupplierDAO**
- REST.**SupplierREST**
- VO.**SupplierVO** (implements java.io.Serializable)
- java.lang.Throwable (implements java.io.Serializable)
  - java.lang.Exception
    - Exception.UserException.**PassWordContainsSpace**
    - Exception.UserException.**PasswordInvalidException**
    - Exception.UserException.**PassWordInvalidLength**
    - Exception.UserException.**PasswordNullException**
    - Exception.UserException.**UserEmptyException**
    - Exception.UserException.**UserNameAlreadyTaken**
    - Exception.UserException.**UserNameContainsInvalidCharacter**
    - Exception.UserException.**UserNameContainsOnlyNumber**
    - Exception.UserException.**UserNameContainsSpace**
    - Exception.UserException.**UserNameInvalidLength**
    - Exception.UserException.**UserNameNullException**
    - Exception.UserException.**UserNotFoundException**
- DAO.**TransactionDAO**
- REST.**TransactionREST**
- VO.**TransactionVO** (implements java.io.Serializable)
- REST.**UploadREST**
- DAO.**UserDAO**
- REST.**UserREST**

- Validation.**UserValidation**
- VO.**UserVO** (implements java.io.Serializable)

4.2.1 Enum Hierarchy

- java.lang.Object
  - java.lang.Enum<E> (implements java.lang.Comparable<T>, java.io.Serializable)
    - VO.**PurchaseOrderVO.PurchaseRequest**
    - VO.**PurchaseOrderVO.DeliveryStatus**
    - VO.**DrugVO.DrugType**
    - VO.**DrugVO.State**
    - VO.**EmployeeVO.WorkingStatus**
    - VO.**EmployeeVO.Designation**
    - VO.**UserVO.UserLevel**
    - VO.**UserVO.AccountStatus**
    - VO.**SalaryVO.Position**
    - VO.**ClientOrderVO.PaymentStatus**

4.2.2 Classes

1. Class DBOperationDAO

Java.lang.object

DAO.Global.PersistenceDAO

DAO.Global.DBOperationDAO

public class DBOperationDAO extends PersistenceDAO

Constructors

DBOperationDAO()

Method Summary

void	closeCurrentSession()
void	closeCurrentSessionWithTransaction()
void	delete(java.lang.String _query)
boolean	deleteById(java.lang.Class<?> _class_type, int _object_id)
org.hibernate.Session	getCurrentSession()
org.hibernate.Transaction	getCurrentTransaction()
java.util.List	getList(java.lang.String _query)
void	insert(java.lang.Object object)
<T> T	load(java.lang.Class<T> type, int _id)
org.hibernate.Session	openCurrentSession()
org.hibernate.Session	openCurrentSessionWithTransaction()
void	setCurrentSession(org.hibernate.Session currentSession)
void	setCurrentTransaction(org.hibernate.Transaction currentTransaction)
void	update(java.lang.Object object)

2. Class PersistenceDAO

java.lang.Object

DAO.Global.PersistenceDAO

Direct Known Subclasses:

DBOperationDAO

public class PersistenceDAO

extends java.lang.Object

Constructors

Constructor and Description

PersistenceDAO()

Method Summary

static org.hibernate.SessionFactory getSessionFactory()

static void shutdown()

3. Class EmployeeREST

java.lang.Object

REST.EmployeeREST

@Path(value="/employee")

public class EmployeeREST

extends java.lang.Object

Constructors

EmployeeREST()

Method Summary

javax.ws.rs.core.Response deleteEmployee(int _employee_id)

javax.ws.rs.core.Response getEmployeeByContactAndCompany(int _company_id, java.lang.String contact_number)

javax.ws.rs.core.Response getEmployeeById(int _employee_id)

javax.ws.rs.core.Response getEmployeeList()

javax.ws.rs.core.Response getEmployeeListByCompanyId(int

_company_id)

javax.ws.rs.core.Response getEmployeeListByDepartment(int _company_id, int _department_id)

javax.ws.rs.core.Response getEmployeeListByDesignation(EmployeeVO.Designation _designation)

javax.ws.rs.core.Response getEmployeeListByJoiningDate(int _company_id, java.lang.String _date_input)

javax.ws.rs.core.Response getEmployeeListByName(java.lang.String _employee_name)

javax.ws.rs.core.Response getEmployeeListByOutlet(int _company_id, int _outlet_id)

javax.ws.rs.core.Response getEmployeeListByWorkingStatus(int _company_id, EmployeeVO.WorkingStatus _working_status)

javax.ws.rs.core.Response getEmployeeQuantity(int _company_id)

javax.ws.rs.core.Response insertEmployee(java.lang.String employeeData)

javax.ws.rs.core.Response insertEmployeeInCompany(int _company_id, java.lang.String employeeData)

javax.ws.rs.core.Response insertEmployeeList(java.lang.String employeeDataList)

javax.ws.rs.core.Response updateUser(java.lang.String _employee_data)

4. Class UploadREST

java.lang.Object

REST.UploadREST

@Path(value="/upload")

public class UploadREST

extends java.lang.Object

Constructors

UploadREST()

Method Summary

javax.ws.rs.core.Response uploadFile(java.io.InputStream uploadedInputStream, org.glassfish.jersey.media.multipart.FormDataContentDisposition fileDetail)

5. Class DepartmentREST

java.lang.Object

REST.DepartmentREST

@Path(value="/department")

public class DepartmentREST

extends java.lang.Object

Constructors

DepartmentREST()

Method Summary

javax.ws.rs.core.Response deleteDepartment(int _department_id)

javax.ws.rs.core.Response getDepartmentList()

javax.ws.rs.core.Response getDepartmentListByCompany(int _company_id)

javax.ws.rs.core.Response getDepartmentListById(int _department_id)

javax.ws.rs.core.Response getDepartmentListByName(java.lang.String _name)

javax.ws.rs.core.Response insertDepartemntList(java.lang.String _inputdata)

javax.ws.rs.core.Response insertDepartment(java.lang.String _inputdata)

javax.ws.rs.core.Response updateUser(java.lang.String _department_data)

6. Class CompanyREST

java.lang.Object

REST.CompanyREST

@Path(value="/company")

public class CompanyREST

extends java.lang.Object

Constructors

CompanyREST()

Method Summary

javax.ws.rs.core.Response deleteCompany(int _company_id)

javax.ws.rs.core.Response getCompanyList()

javax.ws.rs.core.Response getCompanyListByContactNumber(java.lang.String _company_contact)

javax.ws.rs.core.Response getCompanyListByEmail(java.lang.String _company_email)

javax.ws.rs.core.Response getCompanyListByGst(java.lang.String _gst_number)

javax.ws.rs.core.Response getCompanyListById(int _company_id)

javax.ws.rs.core.Response getCompanyListByName(java.lang.String _company_name)

javax.ws.rs.core.Response insertCompany(java.lang.String _inputdata)
javax.ws.rs.core.Response insertCompanyList(java.lang.String _inputdata)
javax.ws.rs.core.Response updateUser(java.lang.String _company_data)

7. Class CompanyVO

java.lang.Object
VO.CompanyVO
All Implemented Interfaces:
java.io.Serializable

@Entity
public class CompanyVO
extends java.lang.Object
implements java.io.Serializable

Constructors
CompanyVO()
CompanyVO(java.lang.String companyName, java.util.Date establishedDate, java.lang.String gstNumber, java.lang.String contactNumberOne, java.lang.String contactNumberTwo, java.lang.String emailId, double totalCash)

Method Summary
java.lang.String getCompanyName()
java.lang.String getContactNumberOne()
java.lang.String getContactNumberTwo()
java.lang.String getEmailId()
java.util.Date getEstablishedDate()
java.lang.String getGstNumber()

int getId()

double getTotalCash()

void setCompanyName(java.lang.String companyName)

void setContactNumberOne(java.lang.String contactNumberOne)

void setContactNumberTwo(java.lang.String contactNumberTwo)

void setEmailId(java.lang.String emailId)

void setEstablishedDate(java.util.Date establishedDate)

void setGstNumber(java.lang.String gstNumber)

void setId(int id)

void setTotalCash(double totalCash)

java.lang.String toString()

8. Class EmployeeVO

java.lang.Object

VO.EmployeeVO

All Implemented Interfaces:

java.io.Serializable

@Entity

public class EmployeeVO

extends java.lang.Object

implements java.io.Serializable

Nested Class Summary

static class EmployeeVO.Designation

static class EmployeeVO.WorkingStatus

Constructors

EmployeeVO()

EmployeeVO(java.lang.String employeeName, EmployeeVO.Designation designation, java.util.Date date, java.lang.String contactNumberOne, java.lang.String contactNumberTwo, java.lang.String address, java.lang.String city, java.lang.String state, java.lang.String aadhaarNo, java.util.Date joiningDate, java.util.Date leavingDate, EmployeeVO.WorkingStatus currentStatus, OutletVO outletVO, CompanyVO companyVO, java.util.List<DepartmentVO> departmentVO, SalaryVO salaryVO)

Method Summary

java.lang.String getAadhaarNo()

java.lang.String getAddress()

java.lang.String getCity()

CompanyVO getCompanyVO()

java.lang.String getContactNumberOne()

java.lang.String getContactNumberTwo()

EmployeeVO.WorkingStatus getCurrentStatus()

java.util.Date getDate()

java.util.List<DepartmentVO> getDepartmentVO()

EmployeeVO.Designation getDesignation()

java.lang.String getEmployeeName()

int getId()

java.util.Date getJoiningDate()

java.util.Date getLeavingDate()

OutletVO getOutletVO()

SalaryVO getSalaryVO()

java.lang.String getState()

void setAadhaarNo(java.lang.String aadhaarNo)

void setAddress(java.lang.String address)

void setCity(java.lang.String city)
void setCompanyVO(CompanyVO companyVO)
void setContactNumberOne(java.lang.String contactNumberOne)
void setContactNumberTwo(java.lang.String contactNumberTwo)
void setCurrentStatus(EmployeeVO.WorkingStatus currentStatus)
void setDate(java.util.Date date)
void setDepartmentVO(java.util.List<DepartmentVO> departmentVO)
void setDesignation(EmployeeVO.Designation designation)
void setEmployeeName(java.lang.String employeeName)
void setId(int id)
void setJoiningDate(java.util.Date joiningDate)
void setLeavingDate(java.util.Date leavingDate)
void setOutletVO(OutletVO outletVO)
void setSalaryVO(SalaryVO salaryVO)
void setState(java.lang.String state)
java.lang.String toString()

Chapter 5

Module Description

5.1 Employee Module

The Employee module takes care of all the Subscribers of the system.
First, the Company's Head, which is usually designated by the CEO's post, registers his Company on the system. The Database table *CompanyManagement* keeps track of all the companies that are registered with the ERP System. Next, the Head adds key managers to their Company's account, and from then onwards the Company Head or the Human Resource Head adds further employees to the system and tracks their activities. The Human Resource Manager also takes care of all the salary payments of employees. He can send task reminders to his employees and also set deadlines for them. One to one, or group communication facility is provided, through which all employees can effectively communicate with one another. All the database tables associated with the Employee module are-

1. *CompanyManagement* (keeps tracks pf companies registered with the system)
2. *EmployeeManagement* (keeps track of employees that are employed by each company)
3. *Salary* (keeps track of salaries)
4. *Departments* (stores metadata of each department)
5. *LoginAccount* (grants a user access to the system)
6. *Reminders* (task reminders can be given to employees with set deadline)
7. *Message* (One-to-one messages can be sent by employees to one another)
8. *MessageTo* (Stores IDs of recipients to whom the message is to be sent to)

8.2 Client Module

The Client Module keeps track of all the clients of the company. Since this ERP is tailor-made for Pharmaceutical Distribution companies, each company registered with the ERP has several clients to which they supply their drugs to. These clients could be brick and mortar retailers, Pharmacy retail chains, Ecommerce firms or Sales Representatives. The Client needs to first get itself registered in the Company's database to place an order. Once it has been registered, it can first check all the stock available with the supplier (the functionality of which is provided in the Client division of Pharma ERP), then place an order according to his needs. Once the order is placed, it will be ratified by the Head or the Outlet Manager of a specific branch of the company. An invoice will be generated, which will be available to the company for future reference. Once the order has been done, all the payment activities will be managed by the finance Head and the company's record books will be updated. All the database tables associated with the Client Module are –

1. *ClientManagement* (keeps track of all the clients associated with a company)
2. *ClientOrder* (stores the details of the order placed by the client)
3. *ClientOrderItems* (stores details about all items ordered by the client)
4. *Transactions* (post-delivery, transaction of the order will be stored)

5.3 Outlet Module

The outlet module takes care of the Company's outlets and manages their Inventory. An Outlet manager is assigned to each outlet who will take care of all the activities happening inside the outlet. Any employee working in a particular

outlet can place an order to the supplier regarding any required stock. But, it can only be sanctioned by the Head or that Outlet's manager. Once the order has been sanctioned and placed, the stock will be supplied to the inventory in due time and all the finances will be updated. The stocks will be added or reduced as per the orders placed to the supplier or by the Client. All the information regarding the stocks and their usage will also be available. All database tables associated with the Outlet Module are –

1. *OutletManagement* (keeps track of all outlets)
2. *DrugManagement* (detailed description of all drugs available at the outlet)
3. *InventoryStocks* (details regarding available stock bundles)
4. *PurchaseOrder* (stores details of order placed to the supplier)
5. *PurchaseItems* (contains all the items placed in the order)
6. *Transactions* (post delivery, transaction of the order will be stored)

5.4 Finance Module

The Finance module takes care of all the company's finances. The Company Head or the Finance Manager will be able to review all of the company's financial records. The actions that fall under the purview of the Finance Manager are tasks like Sanctioning bill payments, client payments, salary payments, report reviewing, checking bill cycle etc. A facility to generate reports is present which will generate timely reports as and when required by the user. Cost calculations keeping the rates of GST in mind will be automatically done. All transactions occurring in the system will have an effect on the Company's liquid assets and the Company Head and the Finance manager will be able to actively keep track of all such activities through various statistical models. All database tables associated with the module are –

1. *Salary* (keeps track of salaries)
2. *PurchaseOrder* (stores details of order placed to the supplier)
3. *PurchaseItems* (contains all the items placed in the order)
4. *ClientOrder* (stores the details of the order placed by the client)
5. *ClientOrderItems* (stores details about all items ordered by the client)
6. *CostPayments* (contains records of all the transactions such as bill payments etc.)
7. *Transactions* (post-delivery, transaction of the order will be stored)

8.5 Supplier Module

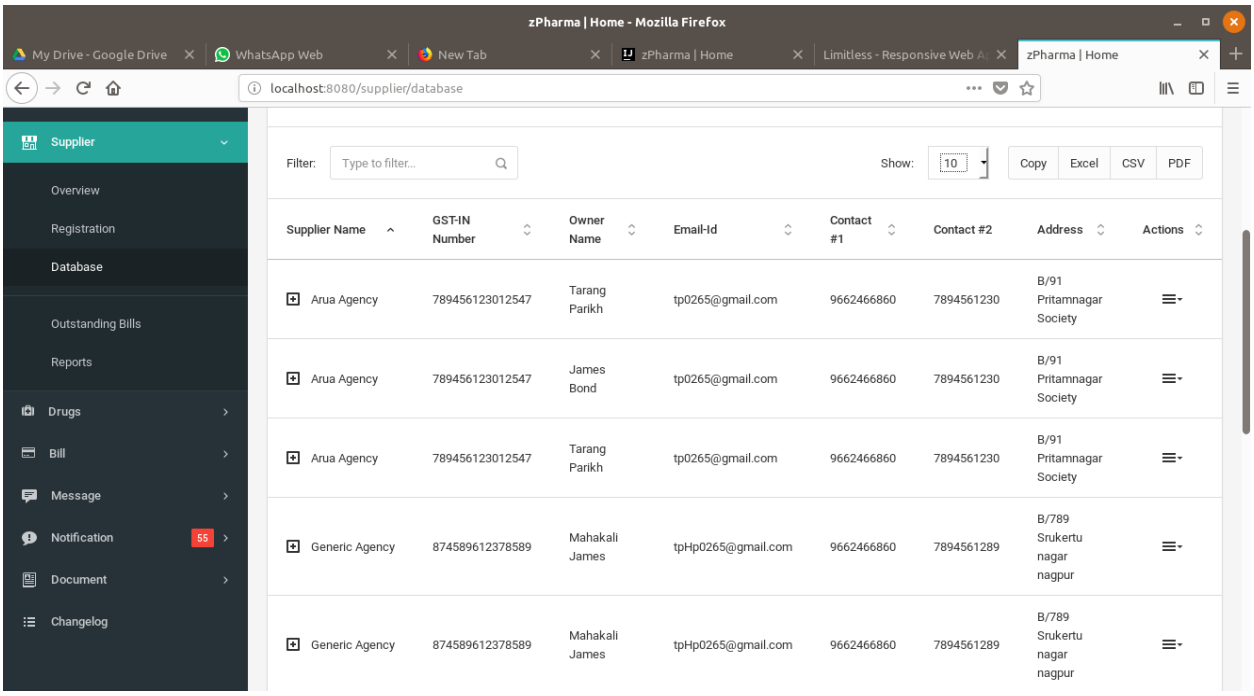
The supplier module takes care of all the suppliers that supply stocks to the various outlets of the distributor. First, whenever needed any employee at a particular outlet will place an order to the supplier, which will then be ratified by the Head or the Outlet Manager, upon delivery of the stocks, all necessary databases as explained above will be updated. But before that, the company needs to add a particular Supplier and all of its necessary details including the GSTN number in its database. The distributor will also have an access to the supplier's inventory, from which he can choose items as per his need and place the order. A company's supplier can be a Pharmaceutical Company, which manufactures drugs and then sells it to various wholesalers and distributor. All companies registered under this system are distributors. The Supplier Module and the Outlet module work very closely to carry out the basic underlying task of maintaining the company's inventory. All the database tables associated with the module are –

1. *SupplierManagement* (keeps track of all suppliers registered with the distributor company)

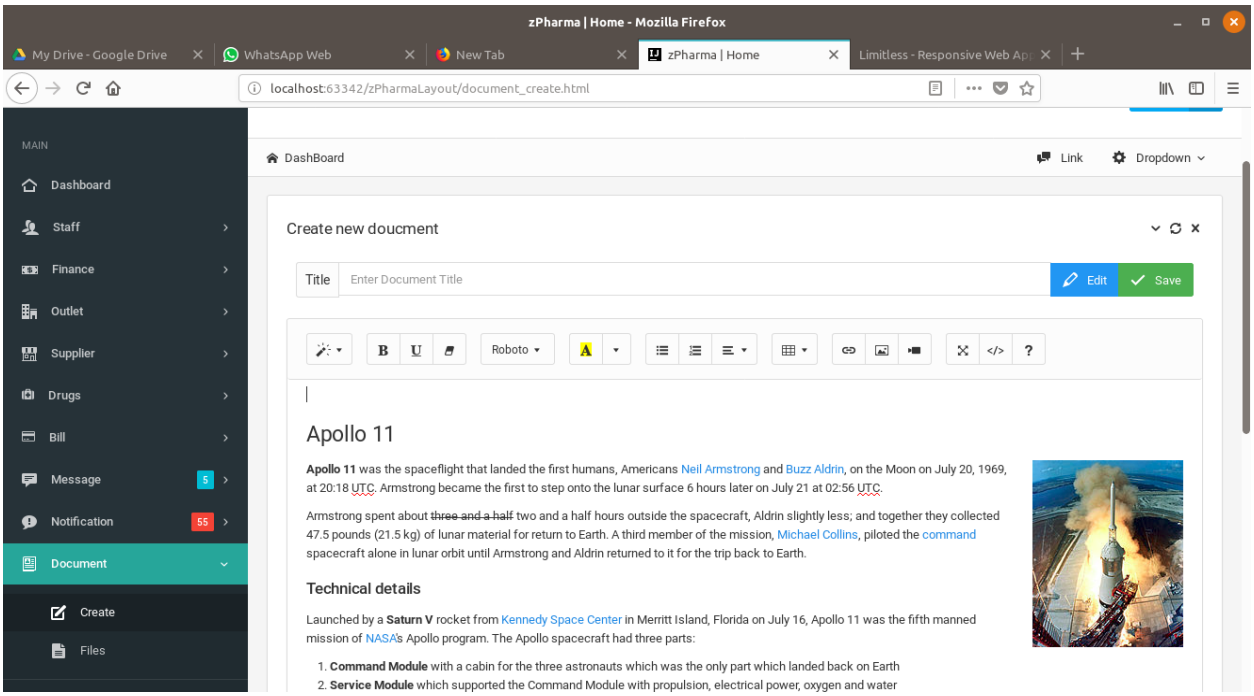
2. *PurchaseOrder* (stores details of order placed to the supplier)
3. *PurchaseOrderItems (*contains all the items placed in the order)
4. *DrugManagement* (detailed description of all drugs available with the supplier)
5. *Transactions* (post-delivery, transaction of the order will be stored)

Chapter 6

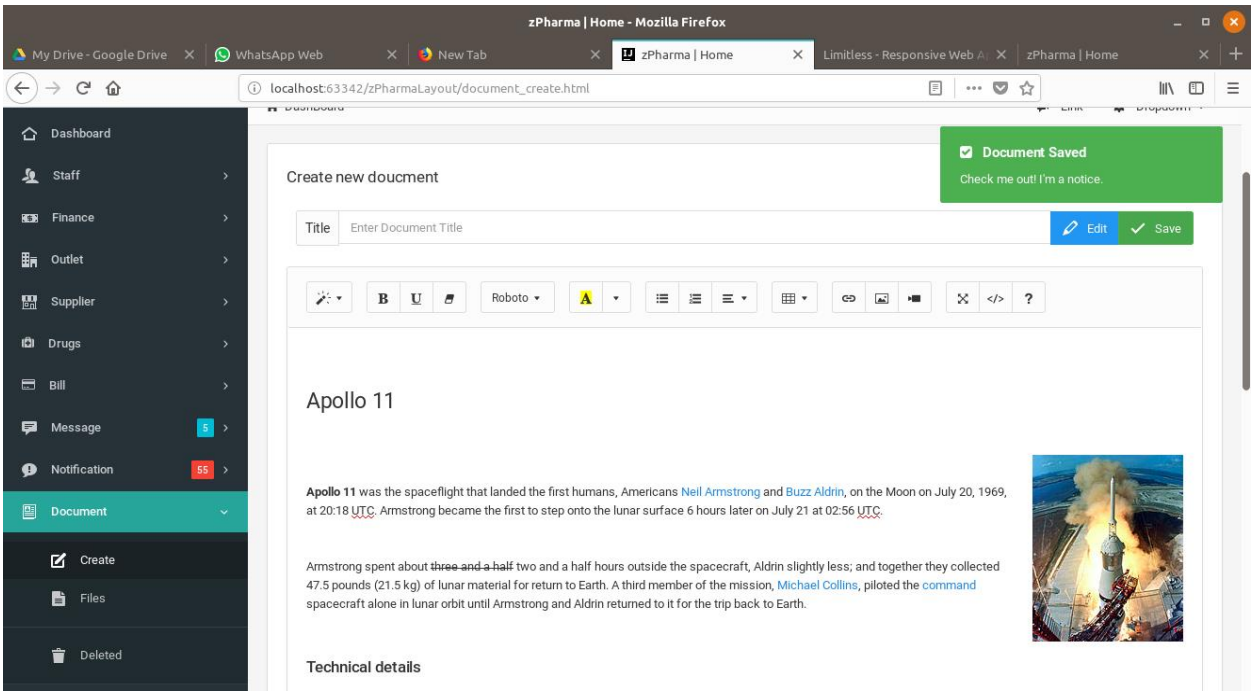
Implementation



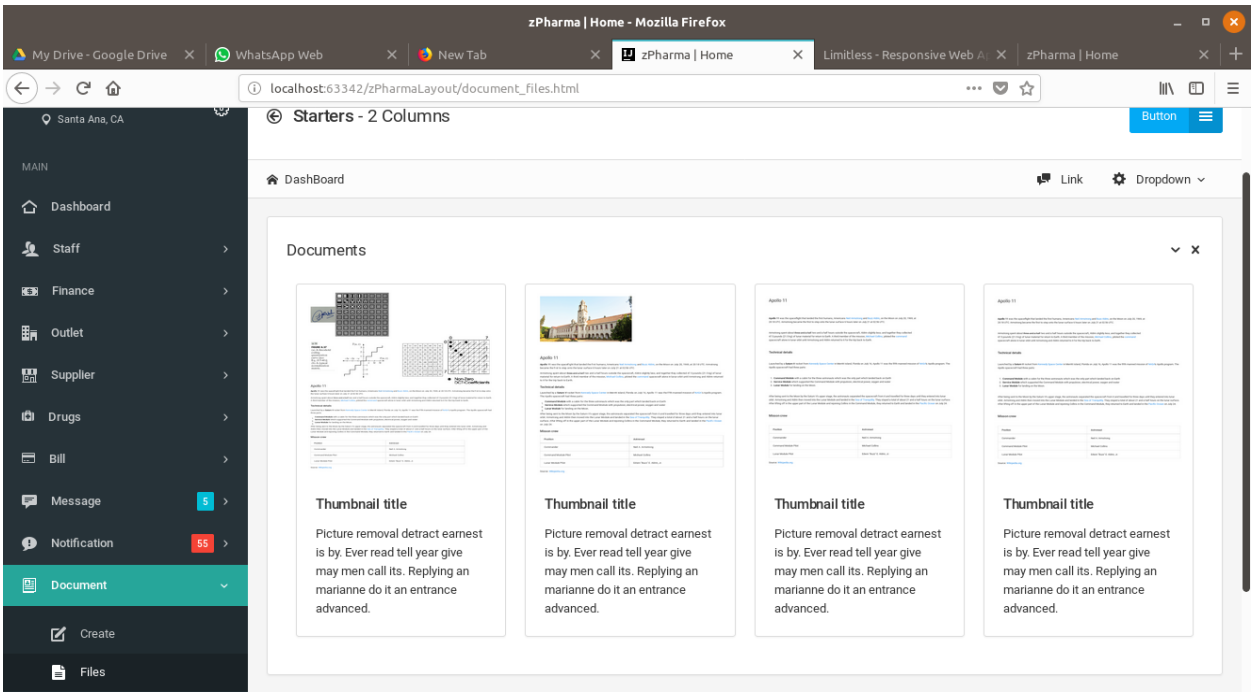
6.1 Supplier Interface



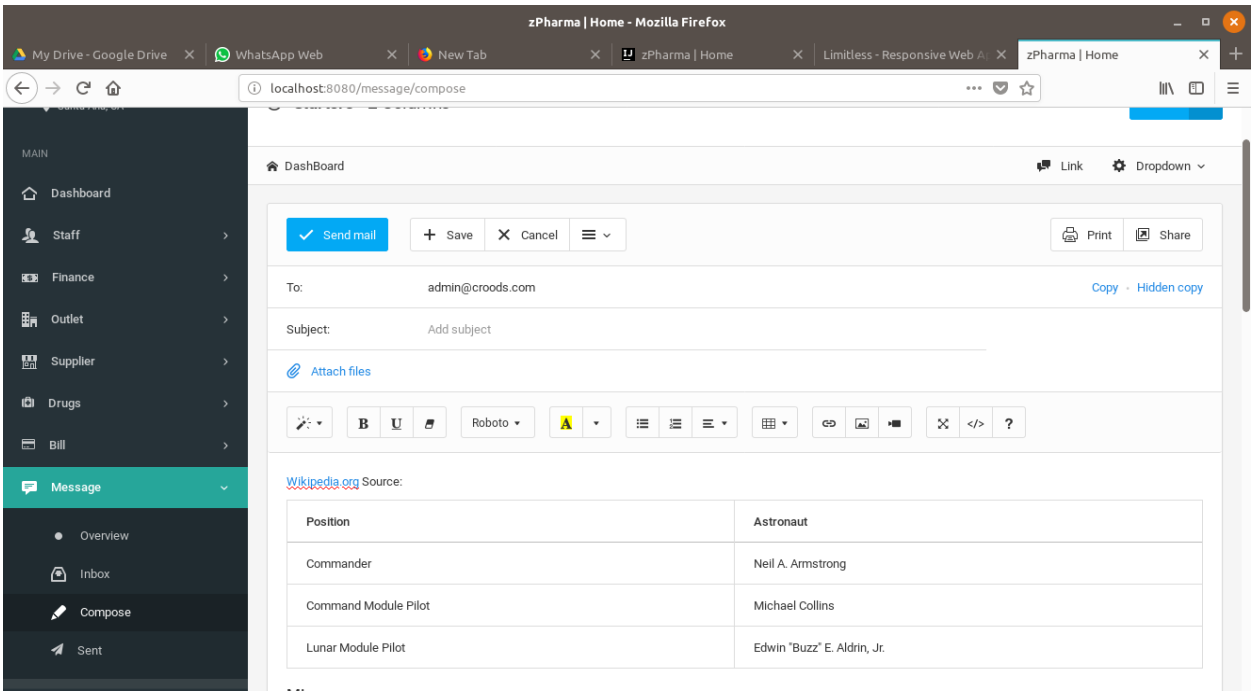
6.2 Document Reporting and Handling



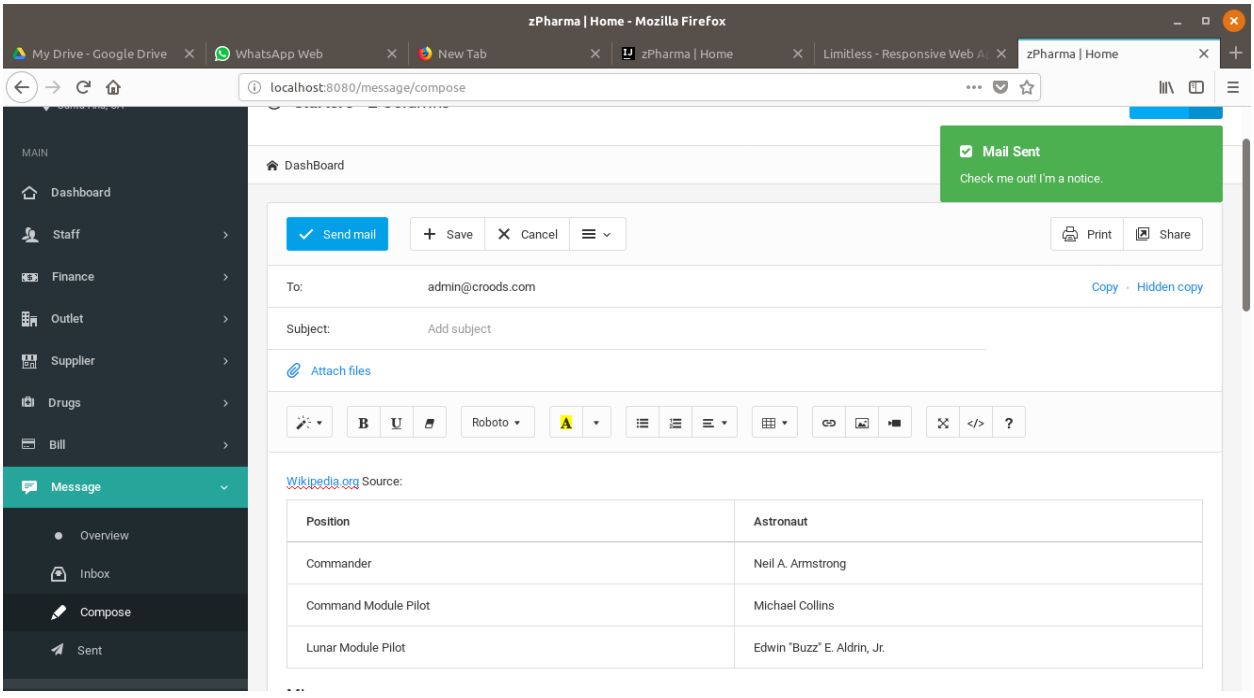
6.3 Creating Documents and Reports



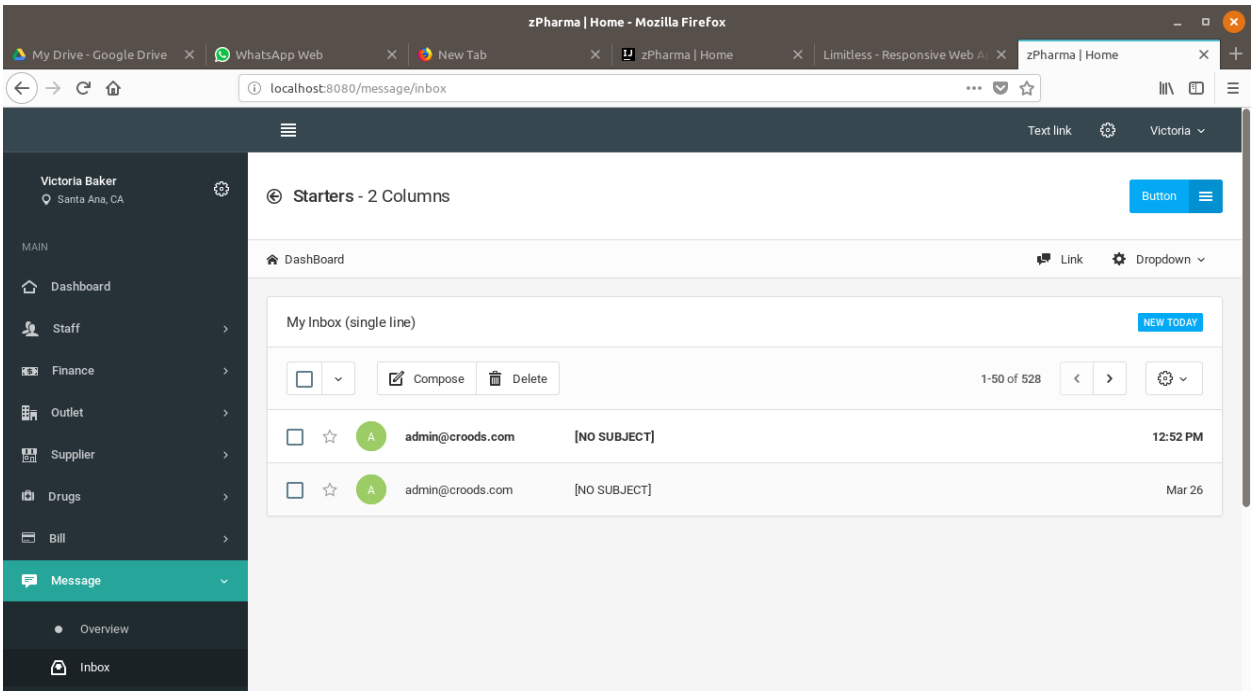
6.4 Document and Report display with Notifications



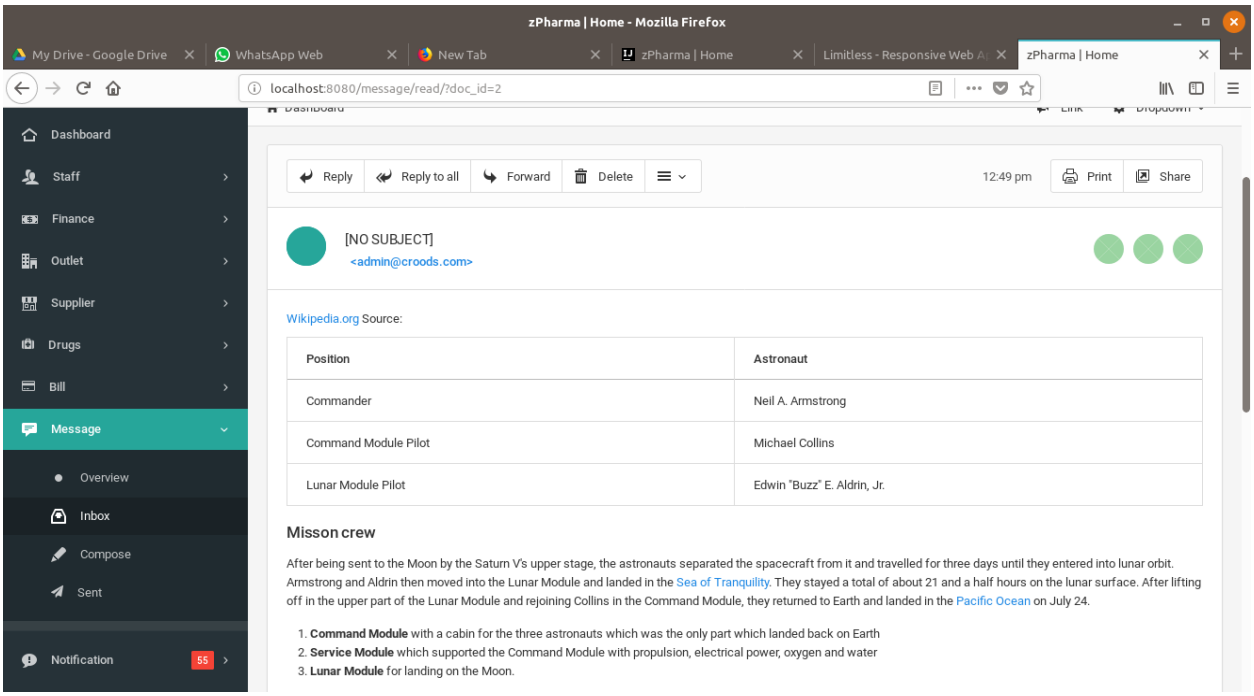
6.5 Messaging Interface



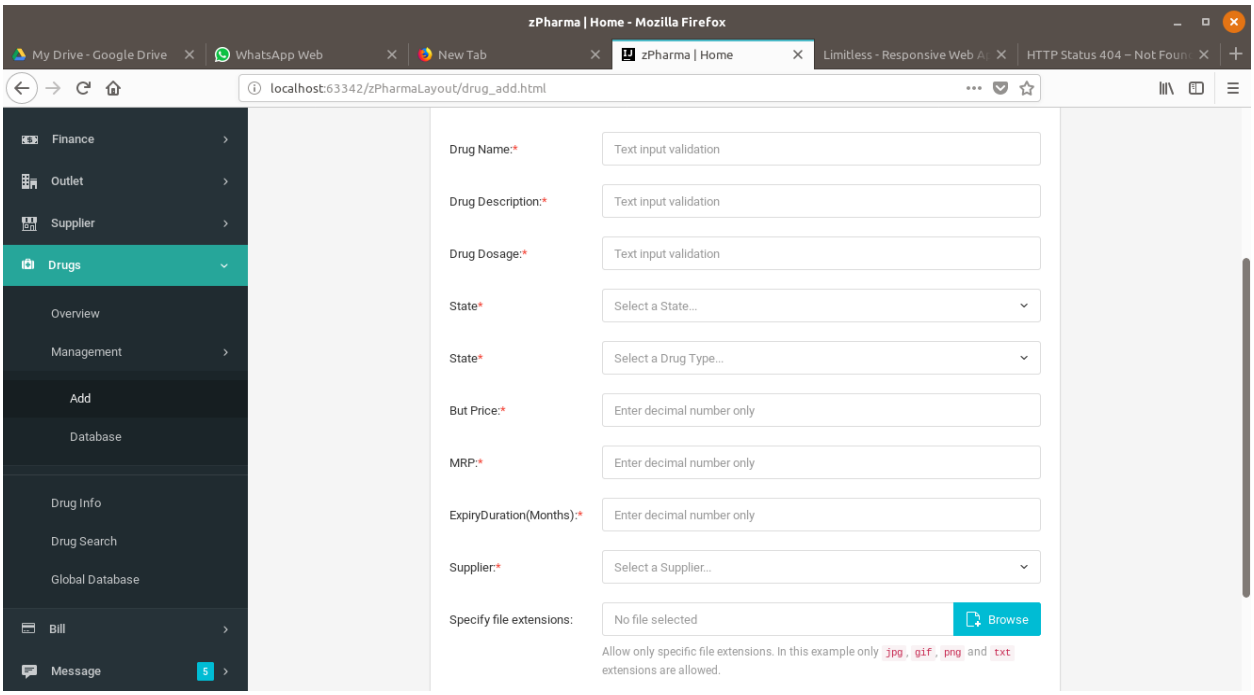
6.6 Composing Message with Attachment Feature



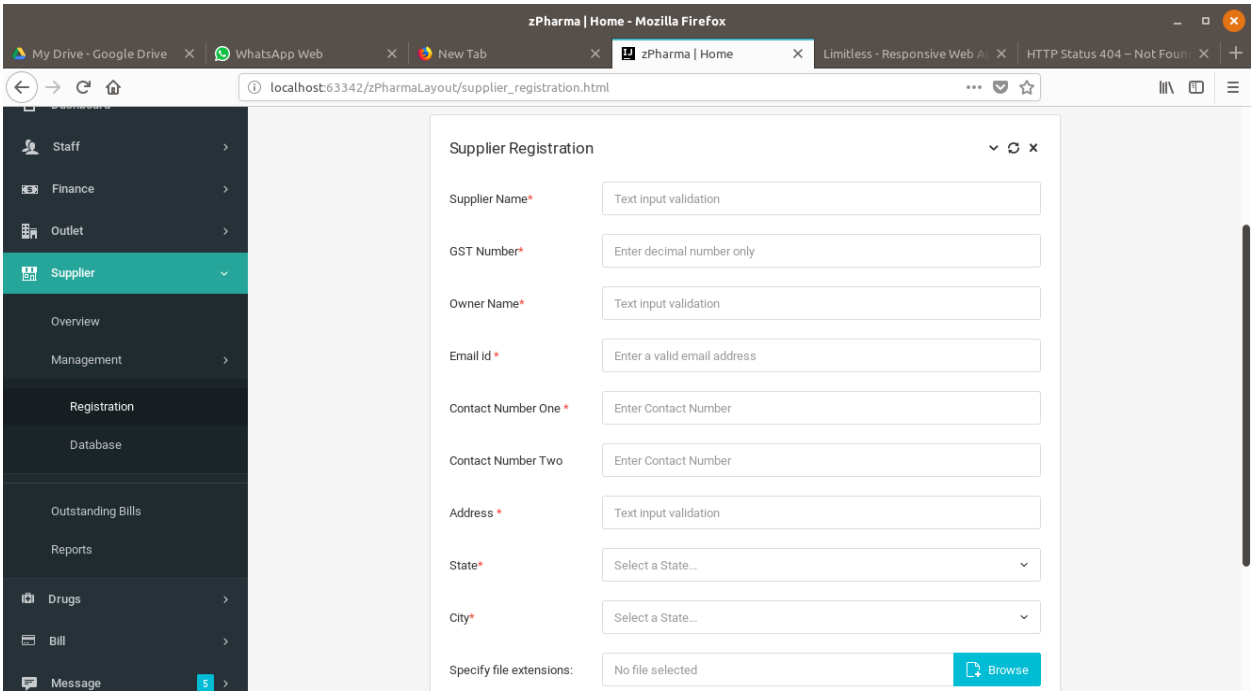
6.7 Messaging - Inbox



6.8 Reading Messages



6.9 Drugs and Medicine



6.10 Supplier Registration

Chapter 7

Test Case Design

7.1 Testing Plane

The testing is a technique that is going to be used in the project is black box testing.
The expected inputs to the system are applied and only the outputs are checked.

7.2 Testing Strategy

The development process repeats this testing sub-process a number of lines for the following phases.

1. Unit Testing
2. Integration Testing

Unit Testing tests a unit of code after coding of that unit is completed. Integration Testing tests whether the previous programs that make up a system, interface with each other as desired. System testing ensures that the system meets its stated design specifications. Acceptance testing is testing by users to ascertain whether the system developed is a correct implementation of the software requirements specification.

Testing is carried out in such a hierarchical manner to that each component is correct and the assembly/combination of the component is correct. Merely testing a whole system at the end would most likely throw up errors in a component that would be very costly to trace and fix. We have performed both Unit Testing and System Testing to detect and fix errors.

7.3 Testing Methods

Test Performed: BlackBox Test

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well.

7.4 Test Cases

7.4.1 Rest API Testing

Test Case ID	Test Scenario	Method Type	Test Steps	Test Data	Expected Results	Actual Results
T01	Company Test	Get	URL	NIL	Return company list	As per expected
		Post	Collect Company data	Company object	Ok Status	As per expected
T02	Client Test	Get	URL	NIL	Return client list	As per expected
		Post	Collect Client data	Client object	Ok Status	As per expected
T03	Client Order	Get	URL	NIL	Return client order list	As per expected
		Post	Collect client order data	Client order object	Ok Status	As per expected

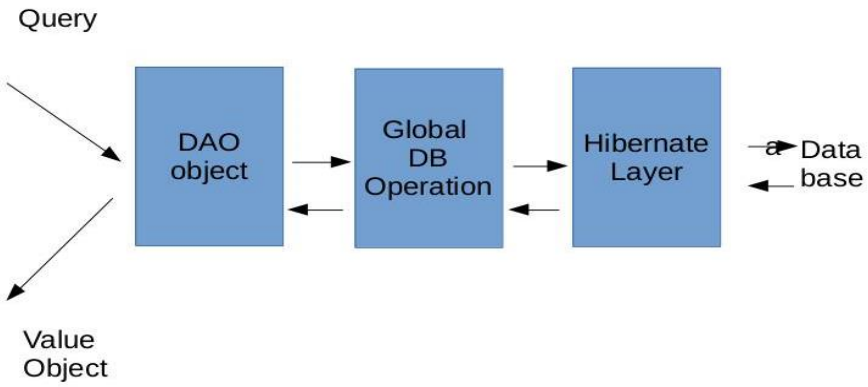
T04	Client Order Item	Get	URL	NIL	Return client order item list	As per expected
		Post	Collect client order item data	client order item object	Ok Status	As per expected
T05	Cost Payments	Get	URL	NIL	Return cost payments list	As per expected
		Post	Collect cost payments data	cost payments object	Ok Status	As per expected
T06	Department	Get	URL	NIL	Return department list	As per expected
		Post	Collect department data	departmen t object	Ok Status	As per expected
T07	Drug Document	Get	URL	NIL	Return drug document list	As per expected
		Post	Collect drug document data	drug document object	Ok Status	As per expected
T08	Drug	Get	URL	NIL	Return drug list	As per expected
		Post	Collect drug data	drug object	Ok Status	As per expected
T09	Employee	Get	URL	NIL	Return employee list	As per expected
		Post	Collect employee data	employee object	Ok Status	As per expected
T10	Inventory Stock	Get	URL	NIL	Return inventory stock list	As per expected
		Post	Collect inventory stock data	inventory stock object	Ok Status	As per expected

T11	Message	Get	URL	NIL	Return message list	As per expected
		Post	Collect message data	message object	Ok Status	As per expected
T12	Message To	Get	URL	NIL	Return message to list	As per expected
		Post	Collect message to data	message to object	Ok Status	As per expected
T13	Outlet	Get	URL	NIL	Return outlet list	As per expected
		Post	Collect outlet data	outlet object	Ok Status	As per expected
T14	Purchase Item	Get	URL	NIL	Return purchase item list	As per expected
		Post	Collect purchase item data	purchase item object	Ok Status	As per expected
T15	Purchase Order	Get	URL	NIL	Return purchase order list	As per expected
		Post	Collect purchase order data	purchase order object	Ok Status	As per expected
T16	Reminder	Get	URL	NIL	Return reminder list	As per expected
		Post	Collect reminder data	reminder object	Ok Status	As per expected
T17	Salary	Get	URL	NIL	Return salary list	As per expected
		Post	Collect salary data	salary object	Ok Status	As per expected
T18	Supplier	Get	URL	NIL	Return	As per

	Document				supplier document list	expected
		Post	Collect supplier document data	supplier document object	Ok Status	As per expected
T19	Supplier	Get	URL	NIL	Return supplier list	As per expected
		Post	Collect supplier data	supplier object	Ok Status	As per expected
T20	Transaction	Get	URL	NIL	Return transaction list	As per expected
		Post	Collect transaction data	transaction object	Ok Status	As per expected

7.4.2 Data Access Object Testing (DAO)

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results
T01	Global DB Test	Start /Stop Connection	Query	Return value object	As per expected
T02	Common test case for all DAO classes	Customized Query	NIL	Return value object	As per expected



7.4.3 Controller Testing

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results
T01	URL Mapping Test	Check URL and Browser	NIL	Navigation to jsp view	As per expected
T02	Request /Response	Check content	Object according to view	Correct rendering of data	As per expected

7.4.4 View Testing

Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Results**	**Actual Results**
T01	JSTL Tag Test	Manual checking of page	NIL	Correct rendering of data	As per expected

Chapter 8

Conclusion and Future Enhancements

8.1 Conclusion

Hereby we declare that we had performed a project by understanding all module of this project. We checked the feasibility and requirement for this system. Then we defined overall look and flow of control among modules in the paper. After this, we started the actual design of our modules of the system in Java with Hibernate framework and Gradle. For database MySQL was used, and for front-end Bootstrap framework has been applied which supports HTML, CSS, etc. All modules of the system were developed separately. Then we integrated all modules by means of control flow among all modules.

After Coding and integrating of all modules done, then we tested all modules separately this is basically Unit Testing of an all modules.by completion of Unit Testing, the whole system is then tested once again this called Integration Testing. Test cases were designed by performing black box testing.

8.2 Future Enhancements

We can change MySQL, which is a relational database to NoSQL database such as Cassandra database.
The reason behind this is Cassandra gives capability to scale user base from thousands to millions.

Chapter 9

Appendix

9.1 Java Development Kit (JDK 8.1)

The JDK is a development environment for building applications, applets, and components using the Java programming language. The JDK includes tools useful for developing and testing programs written in the Java programming language running on the Java platform.

The following is an example of a Java program –

```
import java.sql.*;

public class jdbcConn {
  public static void main(String[] args) {
    try {
      Class.forName("org.apache.derby.jdbc.ClientDriver");
    } catch(ClassNotFoundException e) {
      System.out.println("Class not found "+ e);
    }
    System.out.println("JDBC Class found");
    int no_of_rows = 0;

    try {
      Connection con = DriverManager.getConnection (
        "jdbc:derby://localhost:1527/testDb","username", "password");
      Statement stmt = con.createStatement();
      ResultSet rs = stmt.executeQuery ("SELECT * FROM employee");
      while (rs.next()) {
        no_of_rows++;
      }
```

```
            System.out.println("There are "+ no_of_rows + " record in the table");
          } catch(SQLException e){
            System.out.println("SQL exception occured" + e);
          }
        }
}
```

9.2 IntelliJ IDE with GlassFish Server

IntelliJ IDEA is a JAVA integrated development environment (IDE) for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition.

It includes features such as –
Coding Assistance, built-in tools and integration, plugin ecosystem.
Supports language such as Java, Python, and Groovy etc.
Supports frameworks such as Android, Gradle, and Maven etc.

GlassFish is an open source server project and supports Enterprise JavaBeans, Servlets etc.

9.3 Gradle

Gradle is an open-source build automation system that builds upon the concepts of Apache Maven and introduces a domain – specific language (DSL) instead of the XML form used by Apache Maven for declaring the project configuration. Its supports incremental builds by intelligently determining which parts of the build tree are up-to-date, any task dependent on those parts does not need to be re-executed.
The following is a snippet of the gradle file -

```
group 'GradleThemeDemo'
version '1.0-SNAPSHOT'

apply plugin: 'java'
apply plugin: 'war'

sourceCompatibility = 1.8

repositories {
   mavenCentral()
}

dependencies {

   // https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api
   providedCompile group: 'javax.servlet', name: 'javax.servlet-api', version:
    '3.1.0'
```

```
    // https://mvnrepository.com/artifact/org.hibernate/hibernate-core
    compile group: 'org.hibernate', name: 'hibernate-core', version: '5.1.12.Final'

    // https://mvnrepository.com/artifact/javax.servlet.jsp.jstl/javax.servlet.jsp.jstl-
      api
    compile group: 'javax.servlet.jsp.jstl', name: 'javax.servlet.jsp.jstl-api', version:
      '1.2.1'

    // https://mvnrepository.com/artifact/mysql/mysql-connector-java
    compile group: 'mysql', name: 'mysql-connector-java', version: '6.0.6'


    testCompile group: 'junit', name: 'junit', version: '4.11'
    testCompile group: 'junit', name: 'junit', version: '4.12'
}
```

9.4 Hibernate ORM

Hibernate is an object-relational mapping tool for Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database. Hibernate handles object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions.

Following is a snippet of the HibernateUtil class –

```
package com.vaannila.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;
    static {
        try {
                sessionFactory = new Configuration().configure()
                                .buildSessionFactory();
        } catch (Throwable ex) {
                System.err.println("Initial SessionFactory creation failed." + ex);
                throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
```

9.5 MySQL

It is an open-source relational database management system (RDBMS).
MySQL is fast, reliable, easy-to-use database system used on the web that runs on a server. It is ideal for both small and large applications and uses standard SQL.
MySQL is developed, distributed, and supported by Oracle Corporation.
The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.
Databases are useful for storing information categorically.

Following is a short summary of the most common MySQL commands used –

-- Database-Level

DROP DATABASE databaseName -- Delete the database (irrecoverable!)
DROP DATABASE IF EXISTS databaseName -- Delete if it exists
CREATE DATABASE databaseName -- Create a new database
CREATE DATABASE IF NOT EXISTS databaseName -- Create only if it does not exists
SHOW DATABASES -- Show all the databases in this server
USE databaseName -- Set the default (current) database
SELECT DATABASE() -- Show the default database
SHOW CREATE DATABASE databaseName -- Show the CREATE DATABASE statement

9.6 Jasper Reports

Jasper Reports is an open-source Java reporting tool that can write to a variety of targets, such as a screen, printer into PDF, HTML, Microsoft Excel, RTF, ODT, Comma-Separated Values or XML Files.
It can be used in Java Enabled applications, including Java EE or web applications, to generate dynamic content. It reads its instructions from an XML or .Jasper file.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.util.HashMap;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperExportManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
// import com.mycompany.helper.* ;
// import com.mycompany.dbi.*;
public class ReportGenerator {
 public static void main(String[] args) {
  HashMap hm = null;
  // System.out.println("Usage: ReportGenerator ....");
  try {
   System.out.println("Start ....");
   // Get jasper report
   String jrxmlFileName = "C:/reports/C1_report.jrxml";
   String jasperFileName = "C:/reports/C1_report.jasper";
   String pdfFileName = "C:/reports/C1_report.pdf";
```

```
JasperCompileManager.compileReportToFile(jrxmlFileName,
  jasperFileName);

// String dbUrl = props.getProperty("jdbc.url");
String dbUrl = "jdbc:oracle:thin:@localhost:1521:mydbname";
// String dbDriver = props.getProperty("jdbc.driver");
String dbDriver = "oracle.jdbc.driver.OracleDriver";
// String dbUname = props.getProperty("db.username");
String dbUname = "mydb";
// String dbPwd = props.getProperty("db.password");
String dbPwd = "mydbpw";
// Load the JDBC driver
Class.forName(dbDriver);
// Get the connection
Connection conn = DriverManager
  .getConnection(dbUrl, dbUname, dbPwd);
// Create arguments
// Map params = new HashMap();
hm = new HashMap();
hm.put("ID", "123");
hm.put("DATENAME", "April 2006");
// Generate jasper print
JasperPrint jprint = (JasperPrint) JasperFillManager.fillReport(jasperFileName,
  hm, conn);
// Export pdf file
JasperExportManager.exportReportToPdfFile(jprint, pdfFileName);

System.out.println("Done exporting reports to pdf");
```

```
  } catch (Exception e) {
   System.out.print("Exceptiion" + e);
  }
 }
}
```

9.7 JSoup

JSoup is a Java HTML parser. It is a Java library that is used to parse HTML document. JSoup provides API to extract and manipulate data from URL or HTML file. It uses DOM, CSS and JQuesy-like methods for extracting and manipulating file. Following is an example of showing how to extract HTML elements using JSoup–

```
String html = "<p>An <a href='http://example.com/'><b>example</b></a>
    link.</p>";
Document doc = Jsoup.parse(html);
Element link = doc.select("a").first();

String text = doc.body().text(); // "An example link"
String linkHref = link.attr("href"); // "http://example.com/"
String linkText = link.text(); // "example""

String linkOuterH = link.outerHtml();
   // "<a href="http://example.com"><b>example</b></a>"
String linkInnerH = link.html(); // "<b>example</b>"
```

9.8 HTML

Hypertext Markup Language is the standard markup language for creating web pages and web applications. With *Cascading Style Sheets (CSS)* and *JavaScript*, it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from a local storage and then render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Following is an example of a very basic HTML document –

```
<!DOCTYPE html>
<html>
<body>

<p>This text is normal.</p>

<p><b>This text is bold.</b></p>

</body>
</html>
```

9.9 Bootstrap

Bootstrap is a free front-end framework for faster and easier web development. Bootstrap includes HTML and CSS based design templates for typography, forms,

buttons, tables etc., as well as optional JavaScript plugins. It also gives you the ability to easily create responsive designs.

Following is a very basic example of Bootstrap Code –

```
<form class="form-inline" action="/action_page.php">
  <div class="form-group">
    <label for="email">Email address:</label>
    <input type="email" class="form-control" id="email">
  </div>
  <div class="form-group">
    <label for="pwd">Password:</label>
    <input type="password" class="form-control" id="pwd">
  </div>
  <div class="checkbox">
    <label><input type="checkbox"> Remember me</label>
  </div>
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

Chapter 10

Bibiliography

1. Gradle In Action. Benjamin Muschko (2014)

Alexis Leon (1999). Enterprise Resource Planning. New Delhi: Tata McGraw-Hill Publishing Company.

2. ASAP World Consultancy (2000). Administering SAP R/3 The Production Planning Module. New Delhi: Prentice Hall of India.

3. ASAP World Consultancy (1998). Administering SAP R/3 MM – Materials Management Module. New Delhi: Prentice Hall of India.

4. Ashim Raj Singla (2008). Enterprise Resource Planning. New Delhi: Cengage Learning India Pvt. Ltd.

5. Bastin Gerald, Nigel King, Dan Natchek (2002). Oracle Functional Consultants E-Business Suite Manufacturing & Supply Chain Management. New Delhi: Tata McGraw-Hill Publishing Company.

6. Imed Boughzala and other Editors.(2007). Trends in Enterprise Knowledge Management. USA: ISTE.

7. Mary Sumner (2006). Enterprise Resource Planning. Pearson Education. New Delhi.

8. Pankaj Sharma (2004). Enterprise Resource Planning. New Delhi: APH Publishing Corporation. New Delhi.

9. Seetharama L Narasimhan, Dennis W McLeavey, Peter J Billington (1995). Production Planning and Inventory Control 2nd Ed. New Delhi. Prentice Hall of India.

10. Sachin Sethi (2008). Enhancing Supplier Relationship Management Using SAP SRM. USA:SAPPRESS.

11. Srivastava R.K (2008). Enterprise Resource Planning. New Delhi: Galgotia Publications.

12. Nancy Day, Software Requirements Specification and Analysis, Lecture 3, http://www.student.cs.uwaterloo.ca/cs445/Fall2005/Schedule/l3.pdf.

12. Software Engineering Standards Committee of the IEEE Computer Society, IEEE Recommended Practice for Software Requirements Specifications, 1998.
13. Software Requirement Specification (SRS), http://se.uwaterloo.ca/dberry/ATRE/srs.pdf.
14. Wikipedia, the Free Encyclopedia, http://en.wikipedia.org/.
15. Croods Consolidates Pvt. Ltd.