1.INTRODUCTION

1.1 OVER VIEW OF THE PROJECT

The project entitled as "**CARGO MANAGEMENT SYSTEM**" is a supply chain management component that is used to meet customer demands through planning, control and implementation of the effective movement and storage of related information, goods and services from origin to destination. Cargos service providers face many problems when they do work manually. Some of the drawbacks are customer services become inefficient, transportation cost may increase. Hence, cargo management system application solves all the entire problem and provides many features like, warehouse management, fleet management, processing orders, inventory control and order tracking.

The project contains two type of interaction like Admin and Customer. Admin has authority to add or remove goods in the system and also maintain records of the goods available in the warehouse. The admin will update the goods details if required. Whereas the customer can book the cargo by entering source and destination place.

The main advantages of the system are it excludes the use of paper work by managing all the records electronically. The administrator need not keep a manual track of the goods and can keep updating the system by filling in the details of the new goods arrived in warehouses and their availability. Hence it saves the human efforts and resources

Cargo tracking system application solves all the entire problem and provide many features like, warehouse management, fleet management, processing orders, inventory control and order tracking.

1.2 SYSTEM SPECIFICATION

System requirements specification also known as software requirements specification is a document or set of documentation that describes the features and behavior of a software application.

1.2.1 HARDWARE REQUIREMENTS

- Processor : P 4 700 GHz.
- RAM : 4 GB RAM
- Hard Disk Drive : 180 GB
- LED Monitor : 22 inches
- Keyboard : Multimedia keyboard
- Mouse : Optical Mouse

1.2.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 7/8/10
- Front End : JAVA
- Back End : SQL

1.3 LANGUAGE DESCRIPTION

WINDOWS OS

Windows is a graphical operating system developed by Microsoft. It allows users to view and store files, run the software, play games, watch videos, and provides a way to connect to the internet. It was released for both home computing and professional works.

Microsoft introduced the first version as 1.0

It was released for both home computing and professional functions of Windows on 10 November 1983. Later, it was released on many versions of Windows as well as the current version, Windows 10.

In 1993, the first business-oriented version of Windows was released, which is known as Windows NT 3.1. Then it introduced the next versions, Windows 3.5, 4/0, and Windows 2000. When the XP Windows was released by Microsoft in 2001, the company designed its various versions for a personal and business environment. It was designed based on standard x86 hardware, like Intel and AMD processor. Accordingly, it can run on different brands of hardware, such as HP, Dell, and Sony computers, including home-built PCs.
Play Video

Editions of Windows
Microsoft has produced several editions of Windows, starting with Windows XP. These versions have the same core operating system, but some versions included advance features with an additional cost. There are two most common editions of Windows:

- Windows Home
- Windows Professional

Windows Home is basic edition of Windows. It offers all the fundamental functions of Windows, such as browsing the web, connecting to the Internet, playing video games, using office software, watching videos. Furthermore, it is less expensive and comes pre-installed with many new computers.

JAVA

Java is a high-level programming language developed by Sun Microsystems. It was originally designed for developing programs for set-top boxes and handheld devices, but later became a popular choice for creating web applications.

The Java syntax is similar to C++, but is strictly an object-oriented programming language. For example, most Java programs contain classes, which are used to define objects, and methods, which are assigned to individual classes. Java is also known for being stricter than C++, meaning variables and functions must be explicitly defined. This means Java source code may produce errors or "exceptions" more easily than other languages, but it also limits other types of errors that may be caused by undefined variables or unassigned types.

Unlike Windows executable (.EXE files) or Macintosh applications (.APP files), Java programs are not run directly by the operating system. Instead, Java programs are interpreted by the Java Virtual Machine, or JVM, which runs on multiple platforms. This means all Java programs are multiplatform and can run on different platforms, including Macintosh, Windows, and Unix computers. However, the JVM must be installed for Java applications or applets to run at all. Fortunately, the JVM is included as part of the Java Runtime Environment (JRE),

SQL

Structured query language (SQL) is a programming language for storing and processing information in a relational database. A relational database stores information in tabular form, with rows and columns representing different data attributes and the various relationships between the data values. You can use SQL statements to store, update, remove, search, and retrieve information from the database. You can also use SQL to maintain and optimize database performance.

Relational database management systems use structured query language (SQL) to store and manage data. The system stores multiple database tables that relate to each other. MS SQL Server, MySQL, or MS Access are examples of relational database management systems. The following are the components of such a system.

A SQL table is the basic element of a relational database. The SQL database table consists of rows and columns. Database engineers create relationships between multiple database tables to optimize data storage space.

SQL statements, or SQL queries, are valid instructions that relational database management systems understand. Software developers build SQL statements by using different SQL language elements. SQL language elements are components such as identifiers, variables, and search conditions that form a correct SQL statement.

2.SYSTEM STUDY

2.1 EXISTING SYSTEM

In existing system there will be order from (or) letter which is registerd or updated by the employee of an organization.

The customer has to visit the office whenever they have any booking regarding the services, which is time consuming process. The registered booking may be forwarded to the workers in the specified department.

2.1.1 DRAWBACKS

- The process is done manually.
- Due to that they face problems like data loss.
- The customer does not know current location of the product.
- It consumer more time to process the bookings

2.2 PROPOSED SYSTEM

In Cargo tracking system they were done manual. Now this site to make the things computerized. The customer can submit the orders through the website. The orders can be tracked by the users. The system is a web enabled centralized information and management application to meet customer requirements in building best relation with a customer.

The website will collect more information from the user and give a better view to the booking of cargoes.

2.2.1 FEATURES

- Data is maintained in a structured format.
- Reliable customer order process.
- User can view the location of orders
- Easy time keeping and Tracking all of your customers
- Reduce the cost by lowering the shipping
- Improvements in customers service level

3.SYSTEM DESIGN AND DEVELOPMENT

3.1 FILE DESIGN

The selection of the file system design approach is done according to the needs of the developers what are the needed requirements and specifications for the new design. It allowed us to identify where our proposal fitted in with relation to current and past file system development. Our experience with file system development is limited so the research served to identify the different techniques that can be used. The variety of file systems encountered show what an active area of research file system development is. The file systems may be from one of the two fundamental categories. In one category, the file system is developed in user space and runs as a user process. Another file system may be developed in the kernel space and runs as a privileged process. Another one is the mixed approach in which we can take the advantages of both aforesaid approaches. Each development option has its own pros and cons. In this article, these design approaches are discussed.

A file system is the data structure designed to support the abstraction of the data blocks as an archive and collection of files. This data structure is unique because it is stored on secondary storage (usually the disk), which is a very slow device.

The file system structure is the most basic level of organization in an operating system. Almost all of the ways an operating system interacts with its users, applications, and security model are dependent upon the way it organizes files on storage devices.

File Design Information systems in business are file and database oriented. Data are accumulated into files that are processed or maintained by the system. The systems analyst is responsible for designing files, determining their contents and selecting a method for organizing the data.

The most important purpose of a file system is to manage user data. This includes storing, retrieving and updating data. Some file systems accept data for storage as a stream of bytes which are collected and stored in a manner efficient for the media.

3.2 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:'

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

3.3 OUTPUT DESIGN

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

External Outputs

Manufacturers create and design external outputs for printers. External outputs enable the system to leave the trigger actions on the part of their recipients or confirm actions to their recipients.

Some of the external outputs are designed as turnaround outputs, which are implemented as a form and re-enter the system as an input.

Internal outputs

Internal outputs are present inside the system, and used by end-users and managers. They support the management in decision making and reporting.

Output Integrity Controls

Output integrity controls include routing codes to identify the receiving system, and verification messages to confirm successful receipt of messages that are handled by network protocol.

Printed or screen-format reports should include a date/time for report printing and the data. Multipage reports contain report title or description, and pagination. Pre-printed forms usually include a version number and effective date.

3.4 DATABASE DESIGN

Today's businesses depend on their databases to provide information essential for day-to-day operations, especially in case of electronic commerce businesses who has a definite advantage with up-to-date database access. Good design forms the foundation of any database, and experienced hands are required in the automation process to design for optimum and stable performance.

Software Solutions have been constantly working on these platforms and have attained a level of expertise. We apply proven methodologies to design, develop, integrate and implement database systems to attain its optimum level of performance and maximize security to meet the client's business model.

Business needs addressed:

- Determine the basic objects about which the information is stored
- Determine the relationships between these groups of information and the objects
- Effectively manage data and create intelligent information
- Remote database administration or on site administrative support
- Database creation, management, and maintenance
- Information retrieval efficiency, remove data redundancy and ensure data security

The most important consideration in designing the database is how the information will be used. The main objective of designing a database is Data Integration, Data Integrity and Data Independence.

Data Integration In a database, information from several files is coordinated, accessed and operated upon as through it is in a single file. Logically, the information is centralized, physically; the data may be located on different devices, connected through data

communication facilities.

Data Integrity

Data integrity means storing all data in one place only and how each application accesses it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications. This leads to less data redundancy that is data items need not be duplicated.

Data Independence

Data in dependence is the insulation of application programs from changing aspects of physical data organization. This objective seeks to allow changes in the content and organization of physical data without reprogramming of application and allow modifications to application programs without reorganizing the physical data.

TABLE DESCRIPTION

Admin

Admin has the access to use the this application ,before access they should do the login and Access the application

Customer Registration.

This is collects all the information about the customer like name, mobile number , and address details etc..,.

Ship Registration

Admin register the shipping details and register the ship, once ship has been registerd can make an order under the ship.

Order Registration

After the customer registration admin can able to register an order, this will collect all the information about the order details.

Order details

This is shown the order details tab. We can see the order current status as well.

3.5 SYSTEM DEVELOPMENT

Systems development is the process of defining, designing, testing, and implementing a new software application or program. It could include the internal development of customized systems, the creation of database systems, or the acquisition of third party developed software.

Systems development life cycle phases include planning, system analysis, system design, development, implementation, integration and testing, and operations and maintenance.

3.5.1 DESCRIPTION OF MODULES

Admin Login

Admin has the access to use this application, before access they should do the login and access this application. This application can refer the admin table to validate the login credentials.

Customer Registration

This module collects all the information about the customer like name, mobile number and address details etc.… these details has been stored into the customer details. When we need to collect the customer details can get here.

Ship Registration

Admin register the shipping details and register the ship, once ship has been registered can make an order under the ship. It also contains shipping details.

Order Registration

After the customer registration admin can able to register an order, this will collect all the information about the order details. Which collect the order product name, branch, from, to and weight also.

Order Details

This module had shown the order details in this order details tab. We can see the order current status as well, also able to display the amount of the order.

4. TESTING AND IMPLEMENTATION

TESTING METHODOLOGIES

System testing is state of implementation, which is aimed at ensuring that the system works accurately and efficiently as expect before live operation commences. It certifies that the whole set of programs hang together.

System testing requires a test plan that consists of several key activities and step for run program, string, system and user acceptance testing. The implementation of newly designed package is important in adopting a successful new system

Testing is the important stage in software development. the system test in implementation stage in software development process. The system testing implementation should be confirmation that all is correct and an opportunity to show the users that the system works as expected. It accounts the largest percentage of technical effort in the software development process.

Testing phase in the development cycle validates the code against the functional specification testing is vital to achievement of the system goals. The objective of the testing is to discover errors to fulfill this objective a series of test step unit, integration. Validation and system tests were planned and executed the test steps are:

SYSTEM TESTING

Testing is an integral part of any system development life cycle. Insufficient and untested applications may tend to crash and the result is loss of economic and manpower investment besides user's dissatisfaction and downfall of reputation. Software testing can be looked upon as one among many processes, an organization performs, and that provides the lost opportunity to correct any flaws in the developed system. Software testing includes selecting test data that have more probability of giving errors.

The first step in system testing is to develop a plan that tests all aspects of the system. Completeness, correctness, reliability and maintainability of the software are to be tested for the best quality assurance that the system meets the specification and requirements for its intended use and performance. System testing is the most useful practical process of executing

a program with the implicit intention of finding errors that make the program fails. System testing is done in three phases.

- Unit Testing
- Integration Testing
- Validation Testing

UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software the module. Using the detailed design and the process specification testing is done to registration by the user with in the boundary of the Login module. The login form receives the username and password details and validates the value with the database. If valid, the home page is displayed.

INTEGRATION TESTING

Integration Testing is the process of this activity can be considered as testing the design and hence module interaction. The primary objective of integration testing is to discover errors in the interfaces between the components. Login form and registration form are integrated and tested together. If the user is newly registered, the received details will be stored in the registration table. While logging in, the application will check for valid user name and password in the registration table and if valid the user is prompted for submitting complaints.

Data can be lost across an interface, one module can have adverse effect on another sub function when combined it may not produce the desired major functions. Integration testing is a systematic testing for constructing test to uncover errors associated within an interface.

The objectives taken from unit tested modules and a program structure is built for integrated testing. All the modules are combined and the test is made.

A correction made in this testing is difficult because the vast expenses of the entire program complicated the isolation of causes. In this integration testing step, all the errors are corrected for next testing process.

VALIDATION TESTING

Validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it fulfills its in purpose the actual result from the expected result for the complaint process. Select the complaint category of the complaint by user. The input given to various forms fields are validated effectively. Each module is tested independently. It is tested that the complaint module fields receive the correct input for the necessary details such as complaint category, complaint id, reference name, complaint description, and email for further process.s

After the completion of the integrated testing, software is completely assembled as a package; interfacing error has been uncovered and corrected and a final series of software test validation begins.

Validation testing can be defined in many ways but a simple definition is that validation succeeds when the software function in a manner that can be reasonably expected by the customer. After validation test has been conducted, one of two possible conditions exists.

OUTPUT TESTING

The next process of validation testing, is output testing of the proposed system, since no system could be successful if it does not produce the required output in the specified format. Asking the user about the format required, list the output to be generated or displayed by the system under considerations.

Output testing is a different test whose primary purpose is to fully exercise the computer based system although each test has a different purpose all the work should verify that all system elements have been properly integrated and perform allocated functions.

The output format on the screen is found to be corrected as the format was designed in the system design phase according to the user needs for the hard copy also; the output testing has not resulted in any correction in the system.

SYSTEM IMPLEMENTATION

When the initial design was done for the system, the client was consulted for the acceptance of the design so that further proceedings of the system development can be carried on. After the development of the system a demonstration was given to them about the working of the system. The aim of the system illustration was to identify any malfunction of the system.

After the management of the system was approved the system implemented in the concern, initially the system was run parallel with existing manual system. The system has been tested with live data and has proved to be error free and user friendly.

Implementation is the process of converting a new or revised system design into an operational one when the initial design was done by the system; a demonstration was given to the end user about the working system.

This process is uses to verify and identify any logical mess working of the system by feeding various combinations of test data. After the approval of the system by both end user and management the system was implemented.

System implementation is made up of many activities. The six major activities are as follows.

CODING

Coding is the process of whereby the physical design specifications created by the analysis team turned into working computer code by the programming team. A design code may be a tool which helps ensure that the aspiration for quality and quantity for customers and their requirements, particularly for large scale projects, sought by the water agency Design pattern are documented tried and tested solutions for recurring problems in a given context. So basically you have a problem context and the proposed solution for the same.

INSTALLATION

Installation is the process during which the current system is replaced by the new system. This includes conversion of existing data, software, and documentation and work procedures to those consistent with the new system.

DOCUMENTATION

Documentation is descriptive information that describes the use and operation of the system. The user guide is provided to the end user as the student and administrator. The documentation part contains the details as follows,

User requirement and water agency details administration has been made online. Any customer can request their water requirement details through online and also use of documentation, they can view the purpose of each purpose, The admin could verify the authentication of the users, users requirements and need to take delivery process, thus the documentation is made of full view of project thus it gives the guideline to study the project and how to execute also.

USER TRAINING AND SUPPORT

The software is installed at the deployment environment, the developer will give training to the end user of the regional transport officer and police admin officer in that software. The goal of an end user training program is to produce a motivated user who has the skills needed to apply what has been to apply what has been learned to perform the job related task. The following are the instruction which is specified the handling and un-handling events in the application,

- The authenticated user of admin and office workers only login in the application with authorized username and password.
- Don’t make user waste their time to come straight to the water agency or make a phone call.
- It can easily track through online by the user.
- Very user friendliness software

IMPLEMENTATION PROCEDURES

Implementation includes all the activities that take place to convert the old system to the new one. Proper implementation is essential to provide a reliable system to meet the organization requirements. Implementation is the stage in the project where the theoretical design is turned into a working system. The most crucial stage is achieving a successful new system & giving the user confidence in that the new system will work efficiently & effectively in the implementation state.

PILOT RUNNING

Processing the current data by only one user at a time called the pilot running process. When one user is accessing the data at one system, the system is sets to be engaged and connected in network. This process is useful only in system where more than one user is restricted.

PARALLEL RUNNING:

Processing the current data by more than one user at a time simultaneously is said to be parallel running process. This same system can be viewed and accessed by more than one user at the time. Hence the implementation method used in the system is a pilot type of implementation.

Implementation is the stage in the project where the theoretical design is turned into a working system. The most crucial stage is achieving a successful new system & giving the user confidence in that the new system will work efficiently & effectively in the implementation state.

The stage consists of,

- Testing the developed program with sample data.
- Detection's and correction of error.
- Creating whether the system meets user requirements.
- Making necessary changes as desired by the user.
- Training user personnel.

USER TRAINING

User Training is designed to prepare the user for testing &consenting the system. .

- User Manual.
- Help Screens.
- Training Demonstration.

USER MANUAL

The summary of important functions about the system and software can be provided as a document to the user.

HELP SCREENS

This features now available in every software package, especially when it is used with a menu. The user selects the "Help" option from the menu. The system accesses the necessary description or information for user reference.

TRAINING DEMONSTRATION:

Another User Training element is a Training Demonstration. Live demonstrations with personal contact are extremely effective for Training Users.

SYSTEM MAINTENANCE

Maintenance is actually the implementation of the review plan. As important as it is, many programmers and analysts are to perform or identify themselves with the maintenance effort. There are psychological, personality and professional reasons for this. Analysts and programmers spend far more time maintaining programs than they do writing them. Maintenance accounts for 50-80 percent of total system development

Maintenance is expensive. One way to reduce the maintenance costs are through maintenance management and software modification audits**.**

- Maintenance is not as rewarding as exciting as developing systems. It is perceived as requiring neither skill not experience.
- Users are not fully cognizant of the maintenance problem or its high cost.
- Few tools and techniques are available for maintenance.
- A good test plan is lacking.
- Standards, procedures, and guidelines are poorly defined and enforced.
- Programs are often maintained without care for structure and documentation.
- There are minimal standards for maintenance.
- Programmers expect that they will not be in their current commitment by time their programs go into the maintenance cycle.

Corrective Maintenance

It means repairing, processing or performance failure or making changes because of previously uncovered problems or false assumptions. Task performed to identify, isolate, and rectify a fault so that the failed equipment, machine, or system can be restored to an operational condition within the tolerances or limits established for in-service operations.

Corrective maintenance can be subdivided into "immediate corrective maintenance" (in which work starts immediately after a failure) and "deferred corrective maintenance" (in which work is delayed in conformance to a given set of maintenance rules).

Perfective Maintenance

It means changes made to a system to add new features or to improve performance. Preventive maintenance is predetermined work performed to a schedule with the aim of preventing the wear and tear or sudden failure of equipment components.

Process or control equipment failure can have adverse results in both human and economic terms. In addition to down time and the costs involved to repair and/or replace equipment parts or components, there is the risk of injury to operators, and of acute exposures to chemical and/or physical agents.

Time-based or run-based Periodically inspecting, servicing, cleaning, or replacing parts to prevent sudden failure .On-line monitoring of equipment in order to use important/expensive parts to the limit of their serviceable life. Preventive maintenance involves changes made to a system to reduce the chance of future system failure.

An example of preventive maintenance might be to increase the number of records that a system can process far beyond what is currently needed or to generalize how a system sends report information to a printer so that so that the system can adapt to changes in printer technology.

Preventive Maintenance

Changes made to a system to avoid possible future problems Perfective maintenance involves making enhancements to improve processing performance, interface usability, or to add desired, but not necessarily required, system features. The objective of perfective maintenance is to improve response time, system efficiency, reliability, or maintainability.

During system operation, changes in user activity or data pattern can cause a decline in efficiency, and perfective maintenance might be needed to restore performance. Usually, the perfective maintenance work is initiated by the IT department, while the corrective and adaptive maintenance work is normally requested by users.

5. CONCLUSION

A cargo management system is a powerful tool that helps logistics companies and cargo carriers manage the transportation of goods efficiently and effectively. These systems provide several benefits, including increased efficiency, enhanced visibility, and improved customer satisfaction.

Cargo management systems ensure increased efficiency by automating various processes, such as scheduling, tracking, and reporting. This reduces the time and effort required to complete these tasks and ensures that cargo is transported efficiently and on time. This can lead to increased productivity and profitability for logistics companies and cargo carriers.

Moreover, these systems enhance visibility by providing real-time updates on cargo status, location, and delivery. This helps to promote transparency and trust, and ensures that customers are informed and up-to-date on the status of their cargo. This can lead to improved customer satisfaction and loyalty.

In conclusion, cargo management systems are powerful tools that help logistics companies and cargo carriers manage the transportation of goods efficiently and effectively. They offer several benefits, including increased efficiency, enhanced visibility, and improved customer satisfaction. By leveraging these benefits, logistics companies and cargo carriers can improve their operations, attract more customers, and increase their profitability.

6. FUTURE ENCHANCEMENT

The scope of the project includes that what all future enchancements can be done in this system to make it more feasible to us:-

- Database for different products range and storage can be provided.
- Multilingual support can be provided so that it can be provided.
- More graphics can be added to make it more user friendly and understandable.
- Manage & backup version of documents online.

BIBLIOGRAPHY

Book Reference:

- Bloch, Joshua. Effective Java: Programming Language Guide. Addison-Wesley, 2017.
- Eckel, Bruce. Thinking in Java. Prentice Hall, 2006.
- Horstmann, Cay S. Core Java Volume I--Fundamentals. Prentice Hall, 2018.
- Lea, Doug. Concurrent Programming in Java: Design Principles and Patterns. Addison-Wesley, 2000.
- Shildt, Herbert. Java: A Beginner's Guide. McGraw-Hill Education, 2018.
- Walrath, Kathy, et al. The Java Tutorial: A Short Course on the Basics. Addison-Wesley Professional, 2018.

Website:

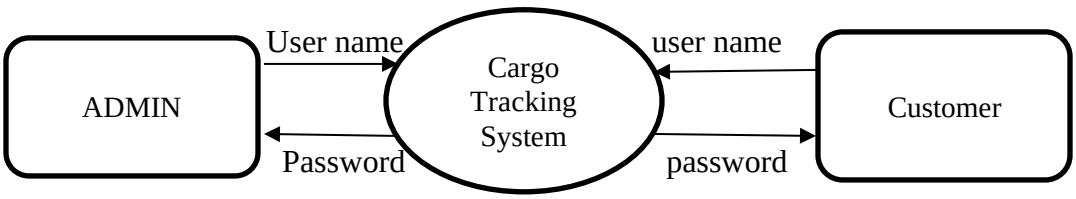
- Baeldung. "Java Tutorials and Articles." Baeldung, 2023, https://www.baeldung.com/java-tutorials.
- GeeksforGeeks. "Java Programming Language." GeeksforGeeks, 2023, https://www.geeksforgeeks.org/java/.
- Stack Overflow. "Questions tagged [java]." Stack Overflow, https://stackoverflow.com/questions/tagged/java.
- Tutorials Point. "Java Tutorial." Tutorials Point, 2023, https://www.tutorialspoint.com/java/index.htm.
- Vogella. "Java Tutorials." Vogella, 2023, https://www.vogella.com/tutorials/java.html.

Spring Framework. "Spring Boot Documentation." Spring Framework, https://docs.spring.io/spring-boot/docs/current/reference/html/.

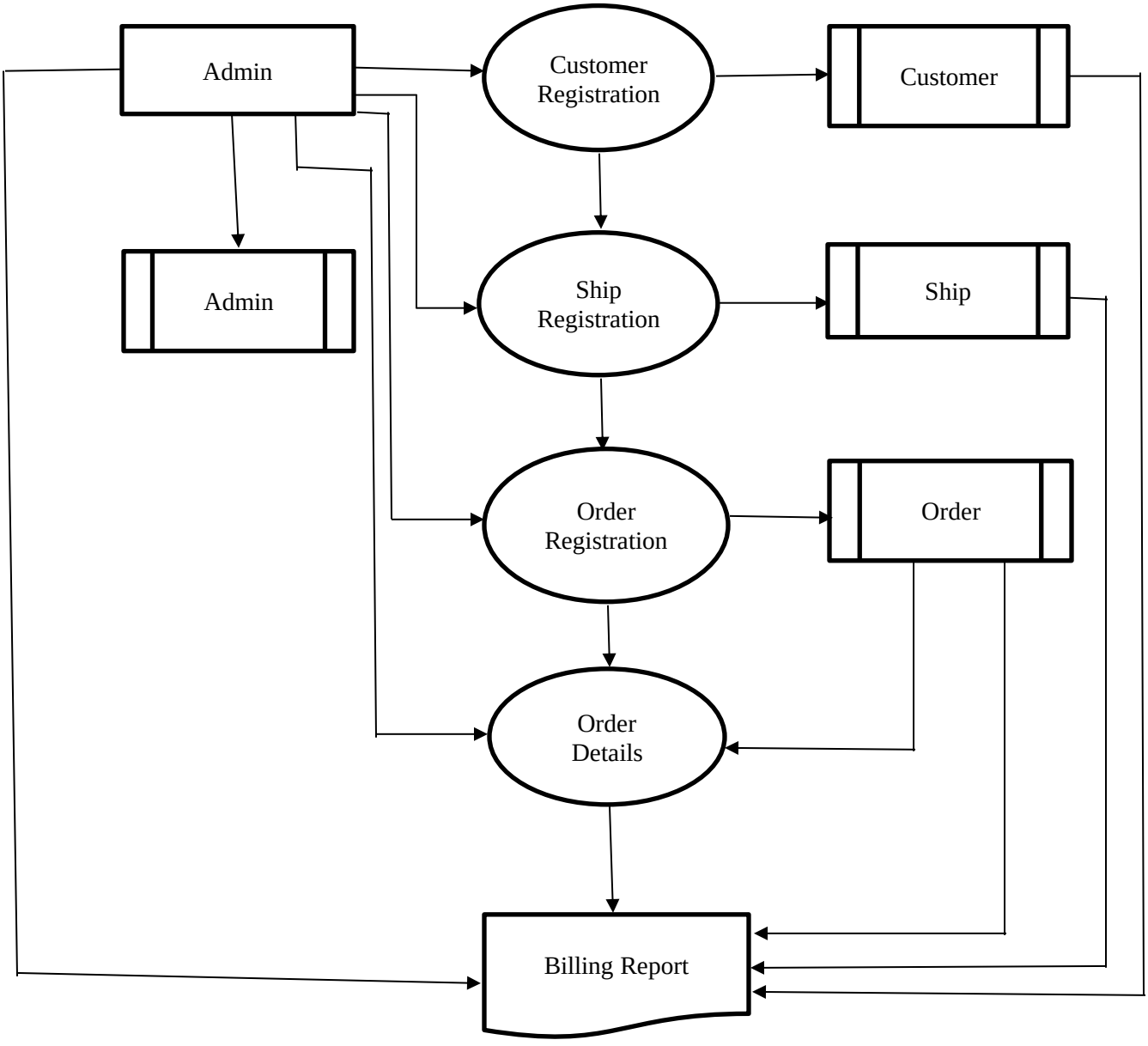
APPENDICES

A. DATA FLOW DIAGRAM

LEVEL 0:



LEVEL 1:


Admin
Customer Registration
Customer
Admin
Ship Registration
Ship
Order Registration
Order
Order Details
Billing Report

B. TABLE STRUCTURE

TABLE NAME: ADMIN

PRIMARY KEY : Admin_ id

Field Name	Data Type	Size	Constraints
Admin _id	Int	10	Primary key
Name	Varchar	25	Not null
Email	Varchar	25	Not null
Phone	Int	10	Not null
Address	Varchar	25	Not null
Zip_ code	Int	6	Not null
Username	Varchar	25	Not null
Password	Varchar	12	Not null

TABLE NAME: SHIP

PRIMARY KEY : Ship_ id

Field Name	Data Type	Size	Constraints
Ship id	Int	10	Primary key

Vehicle_number	Varchar	12	Not null
Vehicle_type	Varchar	15	Not null
Vehicle_model	Varchar	15	Not null

TABLE NAME: ORDER

PRIMARY KEY : Order id

Field Name	**Data Type**	**Size**	**Constraints**
Order id	Int	10	Primary key
Name	Varchar	20	Not null
Address	Varchar	25	Not null
Pickup date	Date	10	Not null
Delivery date	Date	10	Not null
Weight	Numbers	25	Not null
Packing_type	Varchar	25	Not null
Company_name	Varchar	20	Not null
Company_address	Varchar	25	Not null
Mobile number	Number	12	Not null
Picklocation	Varchar	15	Not null
Drop location	Varchar	15	Not null
Customer_id	Int	10	Primary key

TABLE NAME: CUSTOMER

PRIMARY KEY : Customer id

Field Name	Data Type	Size	Constraints
Customer id	Int	10	Primary key
Name	Varchar	25	Not null
Email	Varchar	25	Not null
Phone	Number	10	Not null
Address	Varchar	25	Not null
Zip_code	Int	6	Not null

C. SAMPLE CODEING

```
package com.example.demo.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.example.demo.dao.ApiDao;

import com.example.demo.response.GetCitizenResponse;

import com.example.demo.response.GetComplaintResponse;

import com.example.demo.service.ApiService;

@RestController

@RequestMapping(value = { "/api" })

public class ApiController

@Autowired
```

```
ApiService service;
@Autowired
ApiDao dao;
@GetMapping("/login/{username}/{password}")
public String login(@PathVariable String username,@PathVariable String password) {
return dao.login(username,password);
}
@GetMapping("/add_customer/{name}/{email}/{phone}/{address}/{pincode}")
public String add_customer(@PathVariable String name,
@PathVariable String email,
@PathVariable String  phone,
@PathVariable String  address,
@PathVariable String  pincode
) {
dao.add_customer(name,email,phone,address,pincode);
return "Customer Sucessfully Register";
}
@GetMapping("/update_location/{id}/{location}")
public String update_location(@PathVariable Integer id,
@PathVariable String location
) {
dao.update_location(id,location);
return "Location Sucessfully Updated";
}
@GetMapping("/add_ship/{type}/{number}/{model}")
public String add_ship(@PathVariable String type,
@PathVariable String number,
@PathVariable String  model
) {
dao.add_ship(type,number,model);
```

```
return "Ship Register Sucessfully";
}
@GetMapping("/add_order/{shipid}/{cusid}/{address}/{picdate}/{deliverydate}/{weight}/{packing}
/{comname}/{comaddress}/{mobile}/{picklocation}/{droplocation}/{amount}")
public String add_order(@PathVariable Integer shipid,@PathVariable Integer cusid,
@PathVariable String address,
@PathVariable String  picdate,
@PathVariable String  deliverydate,
@PathVariable String  weight,
@PathVariable String packing,
@PathVariable String  comname,
@PathVariable String  comaddress,
@PathVariable String  mobile,
@PathVariable String picklocation,
@PathVariable String  droplocation,
@PathVariable String  amount
) {
dao.add_order(shipid,cusid,address,picdate,deliverydate,weight,packing,comname,comaddress,mobile
,picklocation,droplocation,amount);
return "Order Register Sucessfully";
}
@GetMapping("/get_customers")
public List<Object[]> get_customers() {
return dao.get_customers();
}
@GetMapping("/get_ships")
public List<Object[]> get_ships() {
return dao.get_ships();
}
@GetMapping("/get_orders")
```

```
public List<Object[]> get_orders() {
return dao.get_orders();
}
}
package com.example.demo.dao;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import javax.transaction.Transactional;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.NativeQuery;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
@Repository
@Transactional
public class ApiDao {
@Autowired
SessionFactory sf;
public String login(String username, String password) {
// TODO Auto-generated method stub
Session session = sf.getCurrentSession();
String sql = "select * from admin where username='"+username+"' and password='"+password+"'";;
NativeQuery nq = session.createNativeQuery(sql);
if (nq.list().size() != 0) {
return "admin";
}else {
return "Invalid";
}
```

```
}
public void add_customer(String name, String email, String phone, String address, String pincode) {
// TODO Auto-generated method stub
// TODO Auto-generated method stub
Session session = sf.getCurrentSession();
String sql = "INSERT INTO `customer` (`id`, `name`, `email`, `phone`, `address`, `pincode`) VALUES
"
+ "(NULL, '"+name+"', '"+email+"', '"+phone+"', '"+address+"', '"+pincode+"');";
session.createSQLQuery(sql).executeUpdate();
}
public void add_ship(String type, String number, String model) {
// TODO Auto-generated method stub
Session session = sf.getCurrentSession();
String sql = "INSERT INTO `ship` (`id`, `type`, `number`, `model`) VALUES "
+ "(NULL, '"+type+"', '"+number+"', '"+model+"');";
session.createSQLQuery(sql).executeUpdate();
}
public void add_order(Integer shipid,Integer cusid, String address, String picdate, String deliverydate,
String weight, String packing,
                String comname, String comaddress, String mobile, String picklocation, String
droplocation, String amount) {
        // TODO Auto-generated method stub
        Session session = sf.getCurrentSession();
        String sql = "INSERT INTO `orders` (`id`,`ship_id`, `customer_id`, `address`,
`pickdate`, `deliverydate`, `weight`, `packingtype`, `comname`, `comaddress`, `mobile`, `pickloc`,
`droploc`, `amount`) VALUES "
                + "(NULL, "+shipid+","+cusid+", '"+address+"', '"+picdate+"',
'"+deliverydate+"', '"+weight+"', '"+packing+"', "
                + "'"+comname+"', '"+comaddress+"', '"+mobile+"',
'"+picklocation+"', '"+droplocation+"', '"+amount+"');";
```

```
                session.createSQLQuery(sql).executeUpdate();
        }
        public List<Object[]> get_customers() {
                // TODO Auto-generated method stub
                Session session = sf.getCurrentSession();
                String sql = "select * from customer";
                NativeQuery nq = session.createNativeQuery(sql);
                return nq.list();
        }
   package com.example.demo.dao;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import javax.transaction.Transactional;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.NativeQuery;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
@Repository
@Transactional
public class ApiDao {
        @Autowired
        SessionFactory sf;
        public String login(String username, String password) {
                // TODO Auto-generated method stub
                Session session = sf.getCurrentSession();
                String sql = "select * from admin where username='"+username+"' and
password='"+password+"'";;
```

```
		NativeQuery nq = session.createNativeQuery(sql);
		if (nq.list().size() != 0) {
			return "admin";
		}else {
			return "Invalid";
		}
	}
	public void add_customer(String name, String email, String phone, String address, String pincode) {
		// TODO Auto-generated method stub
		// TODO Auto-generated method stub
				Session session = sf.getCurrentSession();
				String sql = "INSERT INTO `customer` (`id`, `name`, `email`, `phone`, `address`, `pincode`) VALUES "
						+ "(NULL, '"+name+"', '"+email+"', '"+phone+"', '"+address+"', '"+pincode+"');";
				session.createSQLQuery(sql).executeUpdate();
	}
	public void add_ship(String type, String number, String model) {
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "INSERT INTO `ship` (`id`, `type`, `number`, `model`) VALUES "
				+ "(NULL, '"+type+"', '"+number+"', '"+model+"');";
		session.createSQLQuery(sql).executeUpdate();
	}
	public void add_order(Integer shipid,Integer cusid, String address, String picdate, String deliverydate, String weight, String packing,
			String comname, String comaddress, String mobile, String picklocation, String droplocation, String amount) {
		// TODO Auto-generated method stub
```

```
		Session session = sf.getCurrentSession();
		String sql = "INSERT INTO `orders` (`id`,`ship_id`, `customer_id`, `address`, `pickdate`, `deliverydate`, `weight`, `packingtype`, `comname`, `comaddress`, `mobile`, `pickloc`, `droploc`, `amount`) VALUES "
				+ "(NULL, "+shipid+","+cusid+", '"+address+"', '"+picdate+"', '"+deliverydate+"', '"+weight+"', '"+packing+"', "
				+ "'"+comname+"', '"+comaddress+"', '"+mobile+"', '"+picklocation+"', '"+droplocation+"', '"+amount+"');";
		session.createSQLQuery(sql).executeUpdate();
	}
	public List<Object[]> get_customers() {
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "select * from customer";
		NativeQuery nq = session.createNativeQuery(sql);
		return nq.list();
	}
	public List<Object[]> get_ships() {
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "select * from ship";
		NativeQuery nq = session.createNativeQuery(sql);
		return nq.list();
	}
	public List<Object[]> get_orders() {
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "SELECT orders.id,customer.name,customer.phone,orders.pickloc,orders.droploc,orders.weight,orders.deliverydate,orders.amount,orders.current FROM `orders` LEFT JOIN customer
```

```
on(customer.id=orders.customer_id) ";
                NativeQuery nq = session.createNativeQuery(sql);
                return nq.list();
        }
        public void update_location(Integer id, String location) {
                // TODO Auto-generated method stub
                Session session = sf.getCurrentSession();
                String sql = "UPDATE `orders` SET `current` = '"+location+"' WHERE `orders`.`id` =
"+id;
                session.createSQLQuery(sql).executeUpdate();
        }
}
        public List<Object[]> get_ships() {
                // TODO Auto-generated method stub
                Session session = sf.getCurrentSession();
                String sql = "select * from ship";
                NativeQuery nq = session.createNativeQuery(sql);
                return nq.list();
        }
        public List<Object[]> get_orders() {
                // TODO Auto-generated method stub
                Session session = sf.getCurrentSession();
                String sql = "SELECT
orders.id,customer.name,customer.phone,orders.pickloc,orders.droploc,orders.weight,orders.deliveryd
ate,orders.amount,orders.current FROM `orders` LEFT JOIN customer
on(customer.id=orders.customer_id) ";
                NativeQuery nq = session.createNativeQuery(sql);
                return nq.list();
        }
        public void update_location(Integer id, String location) {
```

```
                // TODO Auto-generated method stub
                Session session = sf.getCurrentSession();
                String sql = "UPDATE `orders` SET `current` = '"+location+"' WHERE `orders`.`id` =
"+id;
                session.createSQLQuery(sql).executeUpdate();
        }
}
package com.example.demo.configuration;
import java.util.Properties;
import javax.sql.DataSource;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.transaction.annotation.EnableTransactionManagement;
@Configuration
@EnableTransactionManagement
public class HibernateConfiguration {
   @Value("${db.driver}")
   private String DB_DRIVER;
   @Value("${db.password}")
   private String DB_PASSWORD;
   @Value("${db.url}")
   private String DB_URL;
   @Value("${db.username}")
   private String DB_USERNAME;
   @Value("${hibernate.dialect}")
   private String HIBERNATE_DIALECT;
```

```
    @Value("${hibernate.show_sql}")
    private String HIBERNATE_SHOW_SQL;
//    @Value("${hibernate.hbm2ddl.auto}")
    private String HIBERNATE_HBM2DDL_AUTO;
    @Value("${entitymanager.packagesToScan}")
    private String ENTITYMANAGER_PACKAGES_TO_SCAN;
    @Bean
    public LocalSessionFactoryBean sessionFactory() {
        LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
        sessionFactory.setDataSource(dataSource());
        sessionFactory.setPackagesToScan(ENTITYMANAGER_PACKAGES_TO_SCAN);
        Properties hibernateProperties = new Properties();
        hibernateProperties.put("hibernate.dialect", HIBERNATE_DIALECT);
        hibernateProperties.put("hibernate.show_sql", HIBERNATE_SHOW_SQL);
//        hibernateProperties.put("hibernate.hbm2ddl.auto", HIBERNATE_HBM2DDL_AUTO);
        sessionFactory.setHibernateProperties(hibernateProperties);
        return sessionFactory;
    }
    @Bean
    public DataSource dataSource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setDriverClassName(DB_DRIVER);
        dataSource.setUrl(DB_URL);
        dataSource.setUsername(DB_USERNAME);
        dataSource.setPassword(DB_PASSWORD);
        return dataSource;
    }
    @Bean
    public HibernateTransactionManager transactionManager() {
        HibernateTransactionManager txManager = new HibernateTransactionManager();
```

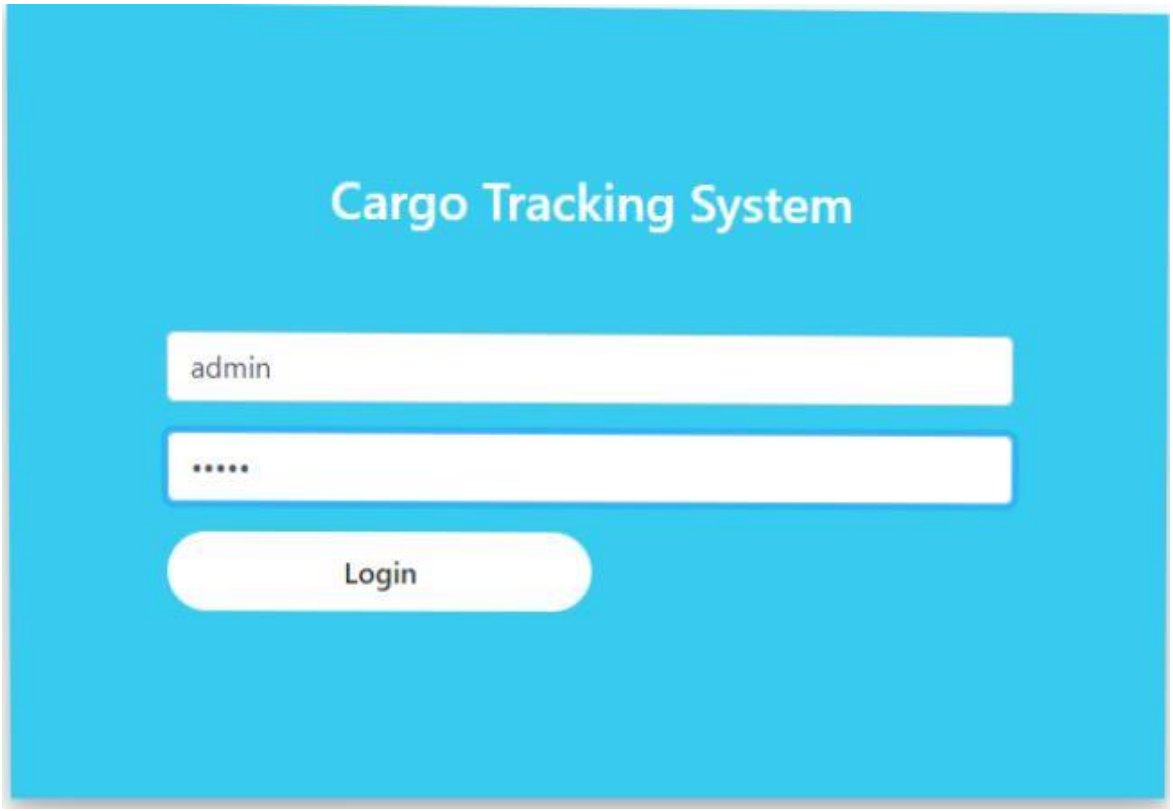
```
        txManager.setSessionFactory(sessionFactory().getObject());
        return txManager;
    }
}
package com.example.demo.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.example.demo.dao.ApiDao;
import com.example.demo.response.GetCitizenResponse;
import com.example.demo.response.GetComplaintResponse;
import com.example.demo.service.ApiService;
@RestController
@RequestMapping(value = { "/api" })
public class ApiController {
        @Autowired
        ApiService service;
        @Autowired
        ApiDao dao;
        @GetMapping("/login/{username}/{password}")
        public String login(@PathVariable String username,@PathVariable String password) {
                return dao.login(username,password);
        }
        @GetMapping("/add_customer/{name}/{email}/{phone}/{address}/{pincode}")
        public String add_customer(@PathVariable String name,
```

```
                    @PathVariable String email,
                    @PathVariable String  phone,
                    @PathVariable String  address,
                    @PathVariable String  pincode
                    ) {
            dao.add_customer(name,email,phone,address,pincode);
            return "Customer Sucessfully Register";
        }
        @GetMapping("/update_location/{id}/{location}")
        public String update_location(@PathVariable Integer id,
                    @PathVariable String location
                    ) {
            dao.update_location(id,location);
            return "Location Sucessfully Updated";
        }
        @GetMapping("/add_ship/{type}/{number}/{model}")
        public String add_ship(@PathVariable String type,
                    @PathVariable String number,
                    @PathVariable String  model
                    ) {
            dao.add_ship(type,number,model);
            return "Ship Register Sucessfully";
        }
        @GetMapping("/add_order/{shipid}/{cusid}/{address}/{picdate}/{deliverydate}/{weight}/{
packing}/{comname}/{comaddress}/{mobile}/{picklocation}/{droplocation}/{amount}")
        public String add_order(@PathVariable Integer shipid,@PathVariable Integer cusid,
                    @PathVariable String address,
                    @PathVariable String  picdate,
                    @PathVariable String  deliverydate,
                    @PathVariable String  weight,
```

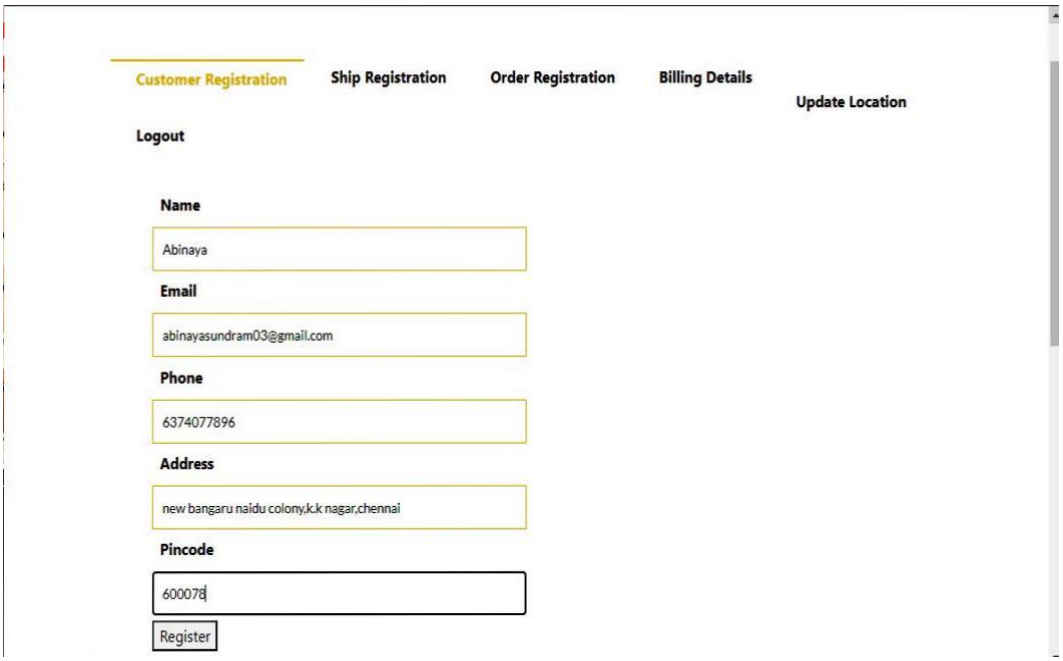
```
                        @PathVariable String packing,
                        @PathVariable String  comname,
                        @PathVariable String  comaddress,
                        @PathVariable String  mobile,
                        @PathVariable String picklocation,
                        @PathVariable String  droplocation,
                        @PathVariable String  amount
                        ) {

        dao.add_order(shipid,cusid,address,picdate,deliverydate,weight,packing,comname,comaddres
s,mobile,picklocation,droplocation,amount);
                return "Order Register Sucessfully";
        }
        @GetMapping("/get_customers")
        public List<Object[]> get_customers() {
                return dao.get_customers();
        }
        @GetMapping("/get_ships")
        public List<Object[]> get_ships() {
                return dao.get_ships();
        }
        @GetMapping("/get_orders")
        public List<Object[]> get_orders() {
                return dao.get_orders();
        }
}
```

D. SAMPLE INPUT DESIGN

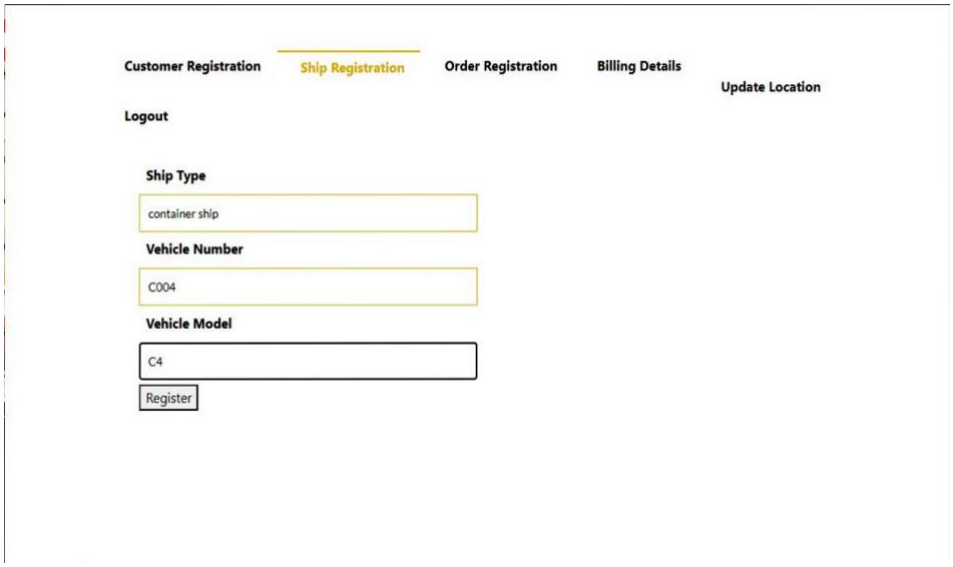
Admin login



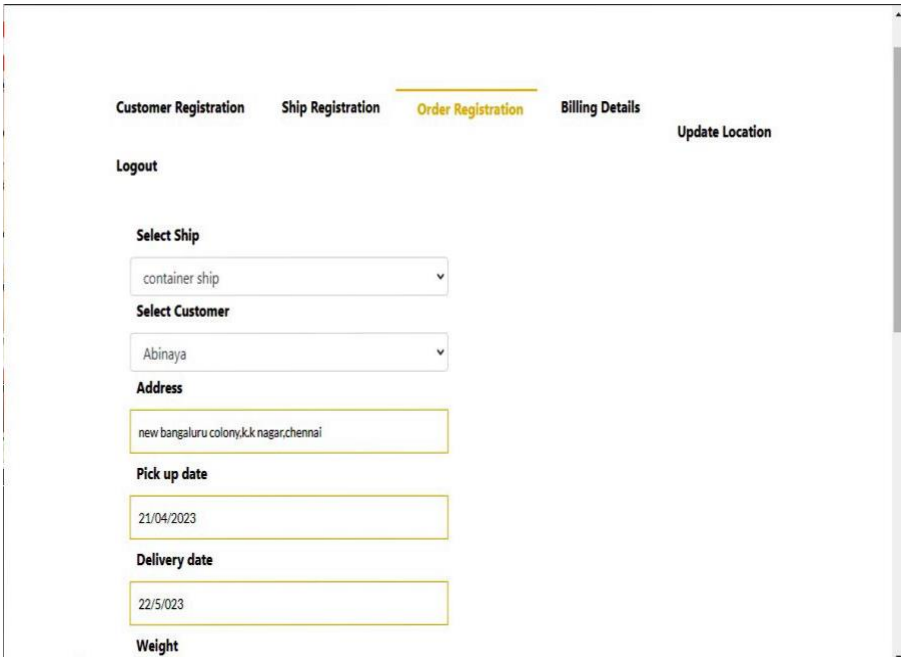
Customer Registration

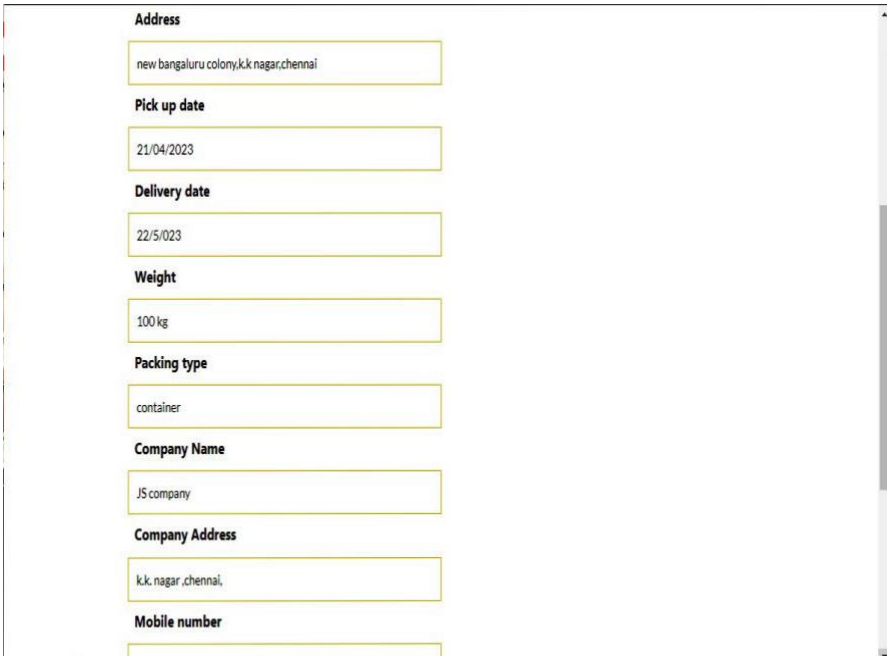


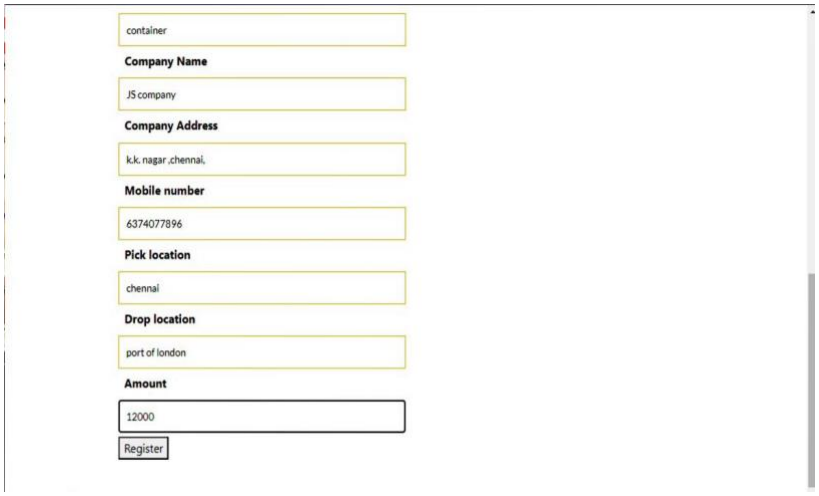
Ship Registration



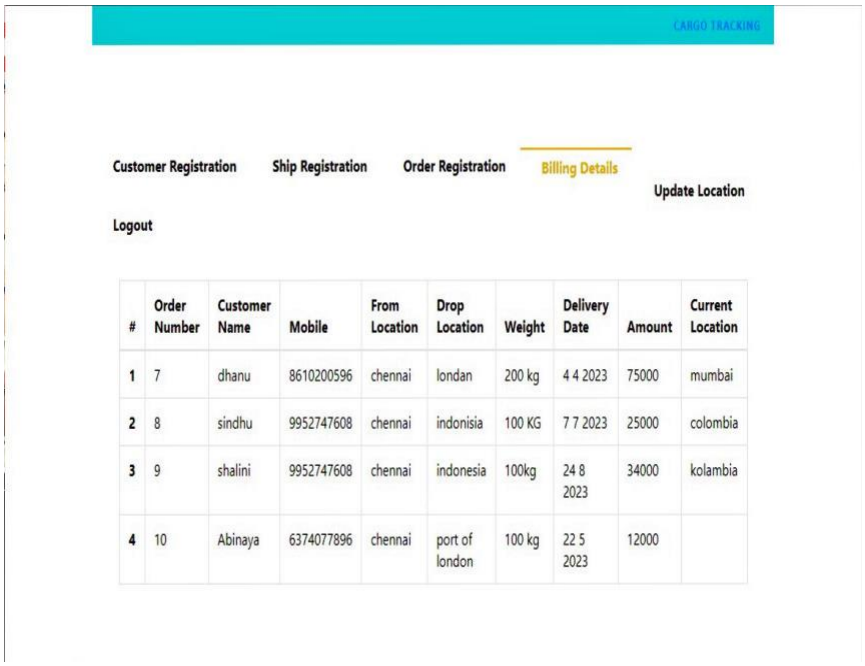
Order Registered




Address
new bangaluru colony,k.k nagar,chennai
Pick up date
21/04/2023
Delivery date
22/5/023
Weight
100 kg
Packing type
container
Company Name
JS company
Company Address
k.k. nagar ,chennai,
Mobile number

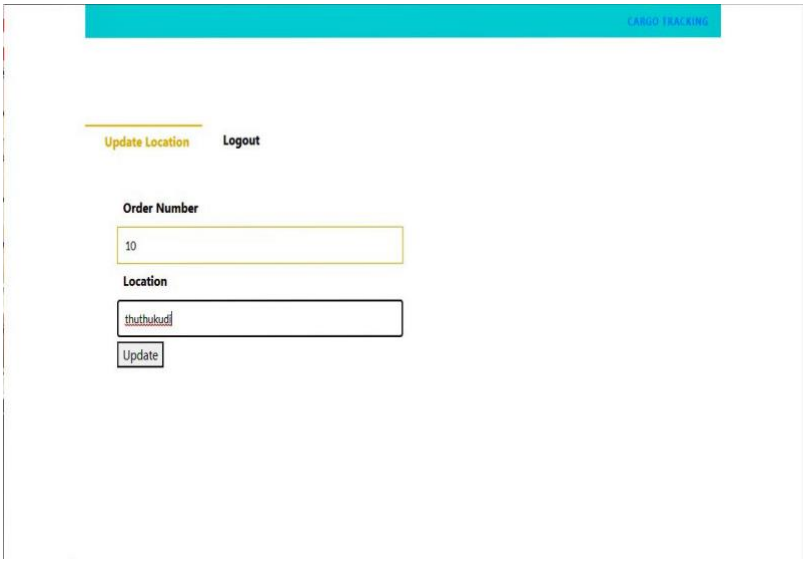

container
Company Name
JS company
Company Address
k.k. nagar ,chennai,
Mobile number
6374077896
Pick location
chennai
Drop location
port of london
Amount
12000
Register

Billing details



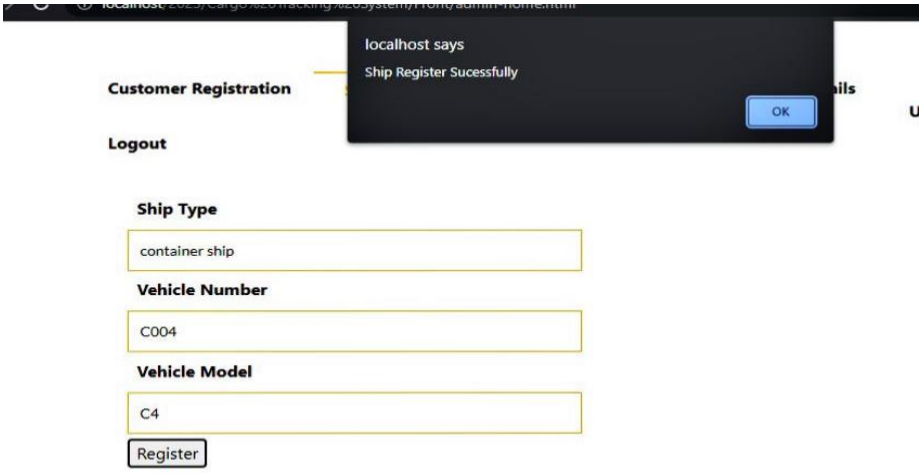
#	Order Number	Customer Name	Mobile	From Location	Drop Location	Weight	Delivery Date	Amount	Current Location
1	7	dhanu	8610200596	chennai	londan	200 kg	4 4 2023	75000	mumbai
2	8	sindhu	9952747608	chennai	indonisia	100 KG	7 7 2023	25000	colombia
3	9	shalini	9952747608	chennai	indonesia	100kg	24 8 2023	34000	kolambia
4	10	Abinaya	6374077896	chennai	port of london	100 kg	22 5 2023	12000	

Update location

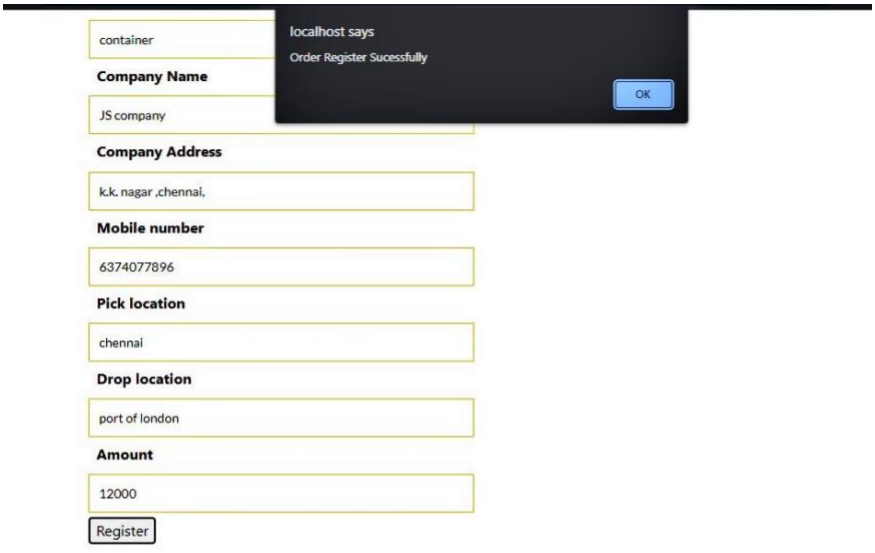


SAMPLE OUTPUT DESIGN

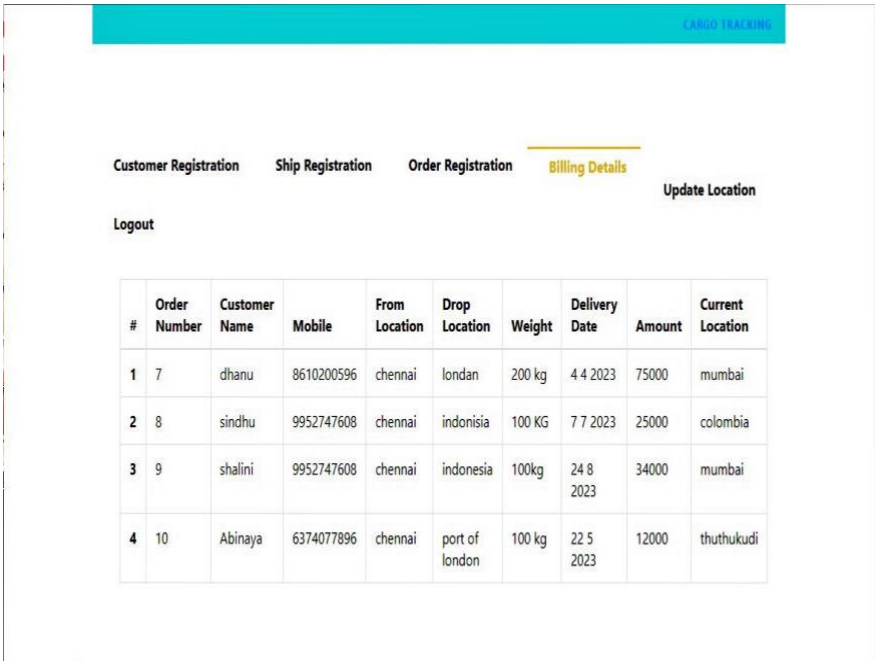
Ship Registered



Order register



Location updated



#	Order Number	Customer Name	Mobile	From Location	Drop Location	Weight	Delivery Date	Amount	Current Location
1	7	dhanu	8610200596	chennai	londan	200 kg	4 4 2023	75000	mumbai
2	8	sindhu	9952747608	chennai	indonisia	100 KG	7 7 2023	25000	colombia
3	9	shalini	9952747608	chennai	indonesia	100kg	24 8 2023	34000	mumbai
4	10	Abinaya	6374077896	chennai	port of london	100 kg	22 5 2023	12000	thuthukudi