# INTRODUCTION

This Bakery Shop Management System Project Report which is developed using eclipse. This is software application and is used maintain control most of the activities happening in big shop. This maintains and controls the stock details and does billing and generates various reports.

The bakery Shop Management System will allow more than one shop owner, to sell various products less than one roof. “Details of products in the stock with product code, product name, quantity” bakery Shop Management System is inventory software that designed specially to meet the requirements of small and medium sized enterprises, such as bakery stores whole sale or retail. It is easy to create invoice, do inventory control like invoice management, stock balance management, goods item management, goods category management, staff sales records management and staff permission management, backup and restore stock by Inventory’s user friendly and graphical interface and functionalities. And Inventory supports full customizable Company info, tax code and value, invoice number etc.

A bakery management system is a software platform that helps bakery owners and managers automate and streamline their business operations. This type of system can help manage inventory, track sales and expenses, create invoices, and generate reports to help bakery owners make informed business decisions.

Bakery management systems may also include features such as recipe management, production planning, and employee scheduling. With recipe management, bakery owners can easily store and organize recipes, track ingredient usage and costs, and adjust recipes as needed. Production planning can help optimize the baking process and ensure that there is enough product available to meet customer demand. Employee scheduling can help bakery owners manage staff availability, schedule shifts, and monitor employee performance.

## 1.1. Organization Profile

Mr. Gerald Arun Dass, popularly known as Mr. GAD, is a world-renowned leader who rules the hearts and minds of his coaches and clients for the past two decades. He has trained, consulted and advised millions of people globally. His sensational ideas and superlative tools have helped ordinary men become extraordinary leaders and global phenomena. He is a Strategic Scientist, Master Leadership Coach, Business Author, Keynote Speaker, Meditation Master, and Advisor to Fortune 500 companies, Transformational Coach, Mind Architect, Start-up Advisor and Peak Performance Guru. He is a man of simplicity and an epitome of humbleness. His thoughts on spirituality and happy living have transformed the way in which thousands of leaders live their lives and run their organizations**.**.

Born and grown up in the temple town of India, he tasted unbelievable achievements from a very young age. Being an award winning orator at just 11 years, President of the National Elocution Club at 13yrs, a trainer at 15yrs and a mind coach at 21yrs, life felt like a bed of roses. But the bubble blanket of appreciations and love was burst when life took an ugly turn. Paving way to life’s cruelty he lost both his parents during his mid-teens and was forced to walk alone at such a confusing age. Instead of sitting down and blaming the fate, he rather increased his pace. The journey of a self-made legend started there. Instead of starting his career as an engineer, he threw away his job offer at Infosys and walked away to become a leader in the field of HR. His search kept growing and landed him in the world’s best pharma & health care consulting company, Pharmexx (Germany) where his success journey continued. At the age of 24, he was a global leader and Board Of Director, making him the youngest global leaders in the company. Having given the best out of every opportunity, he decided to create his own kingdom. But life puts him back to yet another struggle when he went bankrupt as an entrepreneur at just 25 yrs. But his grit and strong determination made him always jump back from any challenge that life threw at him. This time with only him as the investment, he went on to start his own maiden business consulting firm called Shadows Risen.

## System Specifications

### HARDWARE CONFIGURATION

**Processor** : Pentium -IV

**Speed** : 1 GHz

**Hard Disk Capacity** : 40GB

**RAM Capacity** : 1GB RAM

**CD-ROM Drive** : 52x speed

**Keyboard** : 104 keys

**Mouse** : Logitech

**Printer** : HP3745 series DeskJet printer

### SOFTWARE SPECIFICATION

**Operating System** : Windows 7/8/10

**Front End** : JAVA

**Back End** : SQL

**Feasibility Study**

# SYSTEM STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility of a proposed solution is evaluated in teams of its components. These components are:

* + - * Economic feasibility
      * Technical feasibility

## Economic Feasibility

The economic feasibility study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development or the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

## Technical Feasibility

The technical feasibility study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The will lead to high demands on the available technical resources. This will lead to high demands being places on the client. The developed system must have modest requirements, as only minimal or null changes are required for implementing this system.

## EXISTING SYSTEM

Here the shop has been handled by manually, so it’s very hard to managing the bakery shop. Each and every purchase and sales can be handling into the paper and managing it. Here there’s a no records have been provided to the owner or manager. These kinds of issues may be happened in the existing system.

### DRAWBACKS

* Very hard to find the stock details
* No records will be provided by the system

## PROPOSED SYSTEM

Here in this system provide lot of features, like purchase, sales and stock details. Once the admin registered these records it can be followed by the system. Here we can provide the details in less minute time.

### FEATURES

* + - * We can easily identify the product details in a systematic
      * Generate billing and stock details in a system way.

# SYSTEM DESIGN AND DEVELOPMENT

Design is concerned with identifying software components specifying relationship Among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is Minimal clearly specified. Design will explain software components in details. This will help the implementation of the system. Moreover, this will guide the further changes in the system to satisfy the further requirements.

The design document describes how to transform, the requirement and the functional design into more technical system design specification. This design involves conceiving and planning out in the mind and making a drawing pattern of sketch of. It includes type of activities, External Design, Architectural Design and Detailed Design. The architectural design and detailed design collectively referred to as internal design.

The external design involves specifying the externally observable characteristics of a software product and the internal design involves specifying the internal structure and processing details of the system. The fundamental concept of the design include abstraction structure, information hiding Modularity, concurrency, verification and design aesthetics.

## FILE DESIGN

In computing, a file design (or file system) is used to control how data is stored and retrieved. Without a file system, information placed in a storage area would be one large body of data with no way to tell where one piece of information stops and the next begins. By separating the data into individual pieces, and giving each piece a name, the information is easily separated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file". The structure and logic rules used to manage the groups of information and their names are called a "file system".

Some file systems are used on local data storage devices; others provide file access via a network protocol. Some file systems are "virtual", in that the "files" supplied are computed on request or are merely a mapping into a different file system used as a backing store. The file system manages access to both the content of files and the metadata about those files. It is responsible for arranging storage space; reliability, efficiency, and tuning with regard to the physical storage medium are important design considerations.

## INPUT DESIGN

The input design is the process of entering data to the system. The input design goal is to enter to the computer as accurate as possible. Here inputs are designed effectively so that errors made by the operations are minimized.

The inputs to the system have been designed in such a way that manual forms and the inputs are coordinated where the data elements are common to the source document and to the input. The input is acceptable and understandable by the users who are using it.

Input design is the process of converting user-originated inputs to a computer-based format input data are collected and organized into group of similar data. Once identified, appropriate input media are selected for processing.

The input design also determines the user to interact efficiently with the system. Input design is a part of overall system design that requires special attention because it is the common source for data processing error. The goal of designing input data is to make entry easy and free from errors.

Input design is the process of connecting the user-originated inputs into a computer to used format.

The goal of the input design is to make the data entry logical & free from errors.

## OUTPUT DESIGN

Output design is the process of converting computer data into hard copy that is understood by all. The various outputs have been designed in such a way that they represent the same format that the office and management used to.

Computer output is the most important and direct source of information to the user. Efficient, intelligible output design should improve the systems relationships with the user and help in decision making. A major form of output is the hardcopy from the printer.

Output requirements are designed during system analysis. A good starting point for the output design is the Data Flow Diagram (DFD). Human factors educe issues for design involves addressing internal controls to ensure readability.

The output form in the system is either by screen or by hard copies. Output design aims at communicating the results of the processing of the users. The reports are generated to suit the needs of the users. The reports have to be generated with appropriate levels.

All reports are output formats, maintained details can be reported over crystal reports, this project sustain following reports

## DATABASE DESIGN

The most important consideration in designing the database is how information will be used.

The main objectives of designing a database are:

### Data Integration

In a database, information from several files are coordinated, accessed and operated upon as through it is in a single file. Logically, the information are 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 to access it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications, which use it. This leads to less data redundancy; data items need not be duplicated; a reduction in the direct access storage requirement.

### Data Independence

Data independence 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 applications and to allow modifications to application programs without reorganizing the physical data.

The tables needed for each module were designed and the specification of each and every column was given based on the records and details collected during record specification of the system study.

## SYSTEM DEVELOPMENT

The key to control maintenance costs is to design systems that are easy to change, so the link between development and maintenance is very strong. Many of the analysis and design methodologies, tools, and techniques employed during system development can be applied to system maintenance, but there are significant differences between development and maintenance. Maintainability is the ease with which software can be understood, corrected, adopted and enhanced.

### DESCRIPTION OF MODULES

To develop this project several step should be followed. There are various modules in this proposed system they are listed below.

* + - * Customer Registration
      * Product Registration
      * Purchase
      * Sales
      * Stock
      * Billing

### Customer Registration:

This module will be collecting the all the information about the customer, when customer visit and purchase any product admin has collect the customer details and register here.

### Product Registration:

This module admin registers the product details, what are the products are available into the bakery shop, once it’s added it will be stored into the product table

### Purchase:

When owner purchasing the product for a shop, it should manage by this module. each and every purchase detail are should be managing here, then only we can maintain the stock details.

### Sales:

This module is act as an when customer purchasing the any product admin collect the purchase details and sales it. Then it will show the total cost rupees in a sales screen.

### Stock:

This module helps to shown the what are the stocks are available in the shop, we can easily identify the overall stock in the single screen. Also, can able to find the what are the stocks will be close find then can purchase the order.

### Billing:

This module has to shown the billing details for the bakery shop. We can see the billing detail as a date wise and customer wise. This can provide the report to the shop.

# TESTING AND IMPLEMENTATION

System testing is the process of exercising software with the intent of finding and ultimately correcting errors. This fundamental philosophy does not change for web applications, because Web-based systems and application reside on a network and interoperate with many different operating system, browsers, hardware platforms, and communication protocols; the search for errors represents a significant challenge for web application.

The distributed nature of client\server environments, the performance issues associated with transaction processing, the potential presence of a number of different hardware platforms, the complexities of network communication, the need to serve multiple clients from a centralized database and the requirements imposed on the server all combine to make testing of client\server architectures.

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer based system. System testing is the state of implementation that is aimed at assuring that the system works accurately and efficiently. Testing is the vital to the success of the system. System testing makes the logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

### 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

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

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.

**IMPLEMENTATION**

System implementation is the stage of the project that the theoretical design is turned into a working system. If the implementation stage is not properly planned and controlled, it can cause error. Thus it can be considered to be the most crucial stage in achieving a successful new system and in giving the user confidence that the new system will work and be effective.

Normally this stage involves setting up a coordinating committee, which will act as a sounding board for ideas; complaints and problem. The first task is implementation planning; i.e., deciding on the methods and time scale to be adopted. Apart from planning two major task of preparing for implementation are, education takes place much earlier in the project; at the implementation stage the emphasis must be on training in new skills to give staff confidence they can use the system. Once staff has been trained, the system can be tested.

After the implementation phase is completed and the user staff is adjusted to the changes created by the candidate system, evaluation and maintenance is to bring the new system to standards.

# CONCLUSION

In conclusion, a bakery management system is a crucial tool for managing various aspects of a bakery business, including inventory management, sales tracking, employee scheduling, and financial reporting. By implementing a bakery management system, bakery owners can streamline their operations, improve efficiency, reduce waste, and increase profitability.

The bakery management system can also help to provide better customer service by facilitating faster checkout times, allowing for easy ordering and delivery, and enabling personalized marketing campaigns. In addition, the system can generate detailed reports and analytics that can help bakery owners make informed business decisions.

Overall, a bakery management system is an essential investment for any bakery business that wants to stay competitive and thrive in today's market. With the right bakery management system in place, bakery owners can focus on their passion for baking and providing excellent customer service while leaving the business's administrative tasks to the technology..

## BIBLIOGRAPY

**Books Referred:**

* Walrath, Kathy, et al. The Java Tutorial: A Short Course on the Basics. Addison-Wesley Professional, 2018.
* Bloch, Joshua. Effective Java: Programming Language Guide. Addison-Wesley, 2017.
* Eckel, Bruce. Thinking in Java. Prentice Hall, 2006.
* Freeman, Eric, and Elisabeth Robson. Head First Java. O'Reilly Media, 2005.
* Horstmann, Cay S. Core Java Volume I--Fundamentals. Prentice Hall, 2018.
* Lea, Doug. Concurrent Programming in Java: Design Principles and Patterns. Addison-Wesley, 2000.

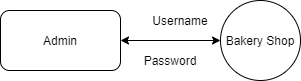
## Websites:

* Baeldung. "Java Tutorials and Articles." Baeldung, 2023, https://www.baeldung.com/java-tutorials.
* GeeksforGeeks. "Java Programming Language." GeeksforGeeks, 2023, <https://www.geeksforgeeks.org/java/>.
* Tutorials Point. "Java Tutorial." Tutorials Point, 2023, https://www.tutorialspoint.com/java/index.htm.
* Vogella. "Java Tutorials." Vogella, 2023,

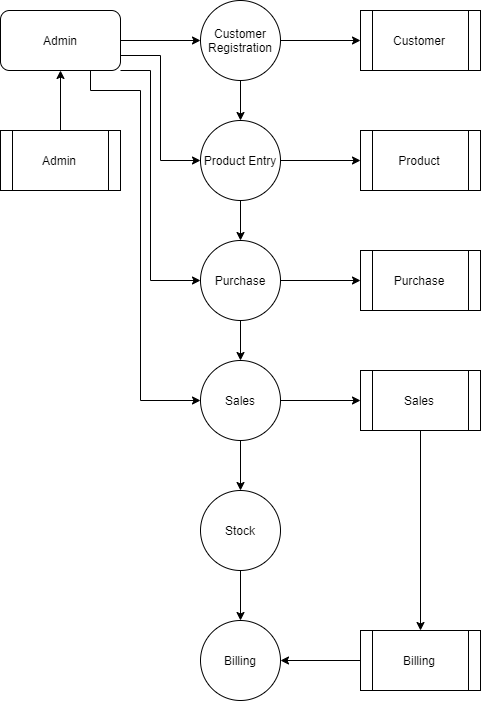
# APPENDICES

## Data Flow Diagram

**Level 0**



**Level 1**

****

## TABLE STRUCTURE

**Table Name :** Admin

**Primary Key :** Admin\_id

**Table Description :** This table is used to maintain the details about admin

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Admin\_id | Int | 8 | Primary Key |
| Username | Varchar | 25 | Not null |
| Address | Varchar | 50 | Not null |
| Phone\_no | Varchar | 15 | Not null |
| Mobile\_No | Varchar | 15 | Not null |

**Table Name :** Customer

**Primary Key :** Customer\_id

**Table Description :** This table is used to maintain the details about Customer

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Customer\_id | Int | 8 | Primary Key |
| Name | Varchar | 25 | Not null |
| Mobile | Varchar | 10 | Not null |
| Alternate | Varchar | 10 | Not null |
| Address | Varchar | 50 | Not null |
| Gender | Varchar | 10 | Not null |

**Table Name :** Product

**Primary Key :** product\_id

**Table Description :** This table is used to maintain the details about Product

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Product\_id | Int | 8 | Primary Key |
| Company | Varchar | 25 | Not null |
| Model | Varchar | 25 | Not null |
| Price | Int | 10 | Not null |

**Table Name :** Purchase

**Primary Key :** purchase\_id

**Table Description :** This table is used to maintain the details about Purchase

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Purchase\_id | Int | 8 | Primary Key |
| Product\_id | Int | 8 | Foreign Key |
| Quantity | Int | 8 | Not null |
| Details | Varchar | 50 | Not null |
| Date | Date | 10 | Not null |

**Table Name :** Sales

**Primary Key :** sales\_id

**Table Description :** This table is used to maintain the details about Purchase

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Sales\_id | Int | 8 | Primary Key |
| Customer\_id | Int | 8 | Foreign Key |
| Product\_id | Int | 8 | Foregin Key |
| Quantity | Int | 8 | Not null |

## B. Sample Coding

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.response.GetBillingResponse;

import com.example.demo.response.GetCustomerResponse;

import com.example.demo.response.GetProductResponse;

import com.example.demo.response.GetStockResponse;

import com.example.demo.service.ApiService;

@RestController

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

public class ApiController {

@Autowired

ApiService service;

@GetMapping("/login/{username}/{password}")

public Boolean login(@PathVariable String username,@PathVariable String password) {

return service.login(username,password);

}

@PostMapping("/add\_customer/{name}/{mobile}/{alternate}/{address}/{gender}")

public String add\_customer(@PathVariable String name,

@PathVariable String mobile,

@PathVariable String alternate,

@PathVariable String address,

@PathVariable String gender) {

service.add\_customer(name,mobile,alternate,address,gender);

return "Customer Saved Sucessfully";

}

@GetMapping("/get\_customer")

public ResponseEntity<List<GetCustomerResponse>> get\_customer() {

return ResponseEntity.ok().body(service.get\_customer());

}

@PostMapping("/add\_product/{company}/{model}/{price}")

public String add\_product(@PathVariable String company,

@PathVariable String model,

@PathVariable Integer price) {

service.add\_product(company,model,price);

return "Product Saved Sucessfully";

}

@GetMapping("/get\_product")

public ResponseEntity<List<GetProductResponse>> get\_product() {

return ResponseEntity.ok().body(service.get\_product());

}

@PostMapping("/add\_purchase/{product\_id}/{quantity}/{details}")

public String add\_purchase(@PathVariable Integer product\_id,

@PathVariable Integer quantity,@PathVariable String details) {

service.add\_purchase(product\_id,quantity,details);

return "Purchase Saved Sucessfully";

}

@PostMapping("/add\_sales/{customer\_id}/{product\_id}/{quantity}")

public String add\_sales(@PathVariable Integer customer\_id,@PathVariable Integer product\_id,

@PathVariable Integer quantity) {

service.add\_sales(customer\_id,product\_id,quantity);

return "Sales Saved Sucessfully";

}

@GetMapping("/get\_customer/{mobile}")

public Integer get\_mobile(@PathVariable String mobile) {

return service.get\_mobile(mobile);

}

@GetMapping("/get\_stock")

public ResponseEntity<List<GetStockResponse>> get\_stock() {

return ResponseEntity.ok().body(service.get\_stock());

}

@GetMapping("/get\_billing")

public ResponseEntity<List<GetBillingResponse>> get\_billing() {

return ResponseEntity.ok().body(service.get\_billing());

}

}

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.configuration;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@Configuration

@EnableWebMvc

public class WebConfig extends WebMvcConfigurerAdapter {

@Override

public void addCorsMappings(CorsRegistry registry) {

registry.addMapping("/\*\*");

}

}

package com.example.demo.dao;

import java.math.BigInteger;

import java.util.List;

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

public class ApiDao {

@Autowired

SessionFactory sf;

public void add\_customer(String name, String mobile, String alternate,String address, String gender) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `customer` (`id`, `name`, `mobile`,`alternate`, `address`, `gender`) VALUES (NULL, '"

+ name + "', '" + mobile + "', '" + alternate + "', '" + address + "', '" + gender + "');";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_customer() {

Session session = sf.getCurrentSession();

String sql = "Select \* from customer";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public void add\_product(String company, String model, Integer price) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `product` (`id`, `company`, `model`, `price`) VALUES (NULL, '" + company + "', '"

+ model + "', '" + price + "');";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_product() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "Select \* from product";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public void add\_purchase(Integer product\_id, Integer quantity, String details) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `purchase` (`id`, `product\_id`, `quantity`, `details`, `date`) VALUES (NULL, '" + product\_id

+ "', '" + quantity + "', '" + details + "', current\_timestamp());";

session.createSQLQuery(sql).executeUpdate();

}

public void add\_sales(Integer customer\_id, Integer product\_id, Integer quantity) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `sales` (`id`, `customer\_id`, `product\_id`, `quantity`) VALUES (NULL, '" + customer\_id

+ "', '" + product\_id + "', '" + quantity + "');";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_stock() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select p.company,p.model,COALESCE(sum(pqty),0) - COALESCE(sum(sqty),0) qty from product p \r\n"

+ "LEFT JOIN (select product\_id,COALESCE(SUM(quantity),0) pqty from purchase GROUP by product\_id) as a on a.product\_id = p.id\r\n"

+ "LEFT JOIN (select product\_id,COALESCE(SUM(quantity),0) sqty from sales GROUP by product\_id) as b on b.product\_id = p.id\r\n"

+ "GROUP BY p.company,p.model";

System.out.println(sql);

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public Integer get\_mobile(String mobile) {

Session session = sf.getCurrentSession();

String sql = "Select id,name from customer where mobile='" + mobile + "'";

NativeQuery nq = session.createNativeQuery(sql);

List<Object[]> list = nq.getResultList();

if (list.size() != 0) {

return (Integer) list.get(0)[0];

} else {

return null;

}

}

public List<Object[]> get\_billing() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select customer.name,customer.mobile,product.company,product.model,sales.quantity,product.price from sales LEFT JOIN customer on(customer.id=sales.customer\_id) LEFT JOIN product on(product.id=sales.product\_id) ";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public Boolean 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 true;

} else {

return false;

}

}

}

<html>

<link href="./css/bootstrap.css" rel="stylesheet" id="bootstrap-css">

<script src="./js/bootstrap.js"></script>

<script src="./js/jquery.js"></script>

<script src="./js/home.js"></script>

<!------ Include the above in your HEAD tag ---------->

<body>

<nav style="background-color:red!important" class="navbar navbar-icon-top navbar-expand-lg navbar-dark bg-warning">

<a class="navbar-brand" href="#">Bakery Shop</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="customer-registration.html">

<i class="fa fa-home"></i>

Customer Registration

<span class="sr-only">(current)</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="product.html">

<i class="fa fa-home"></i>

Product

<span class="sr-only">(current)</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="purchase.html">

<i class="fa fa-home"></i>

Purchase

<span class="sr-only">(current)</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="sales.html">

<i class="fa fa-home"></i>

## D. Sample Input

## User Login page

## 

## User login page with user input

## 

## Customer Registration Page

## 

## Product Registration Page

## 

## Product Purchase Page

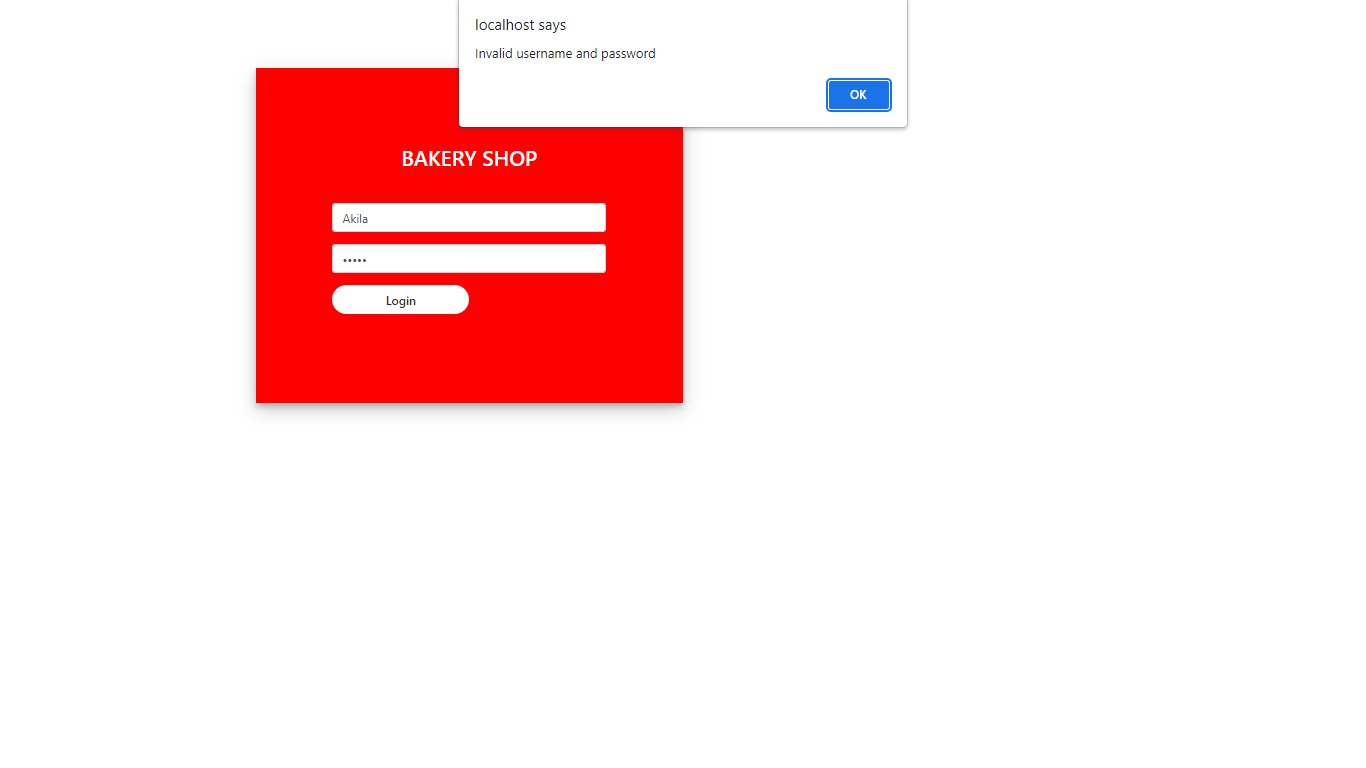
## 

## Product Sales Page

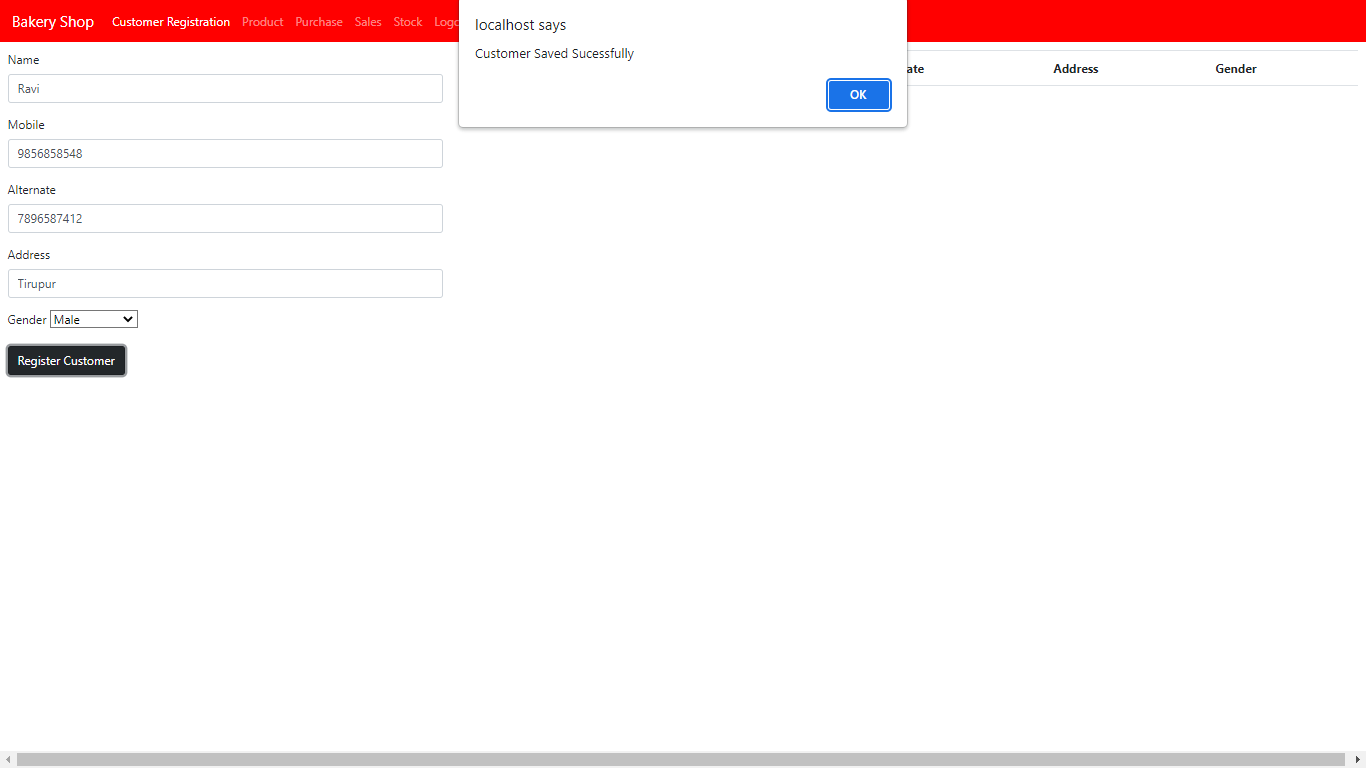
## 

## E. Sample Output

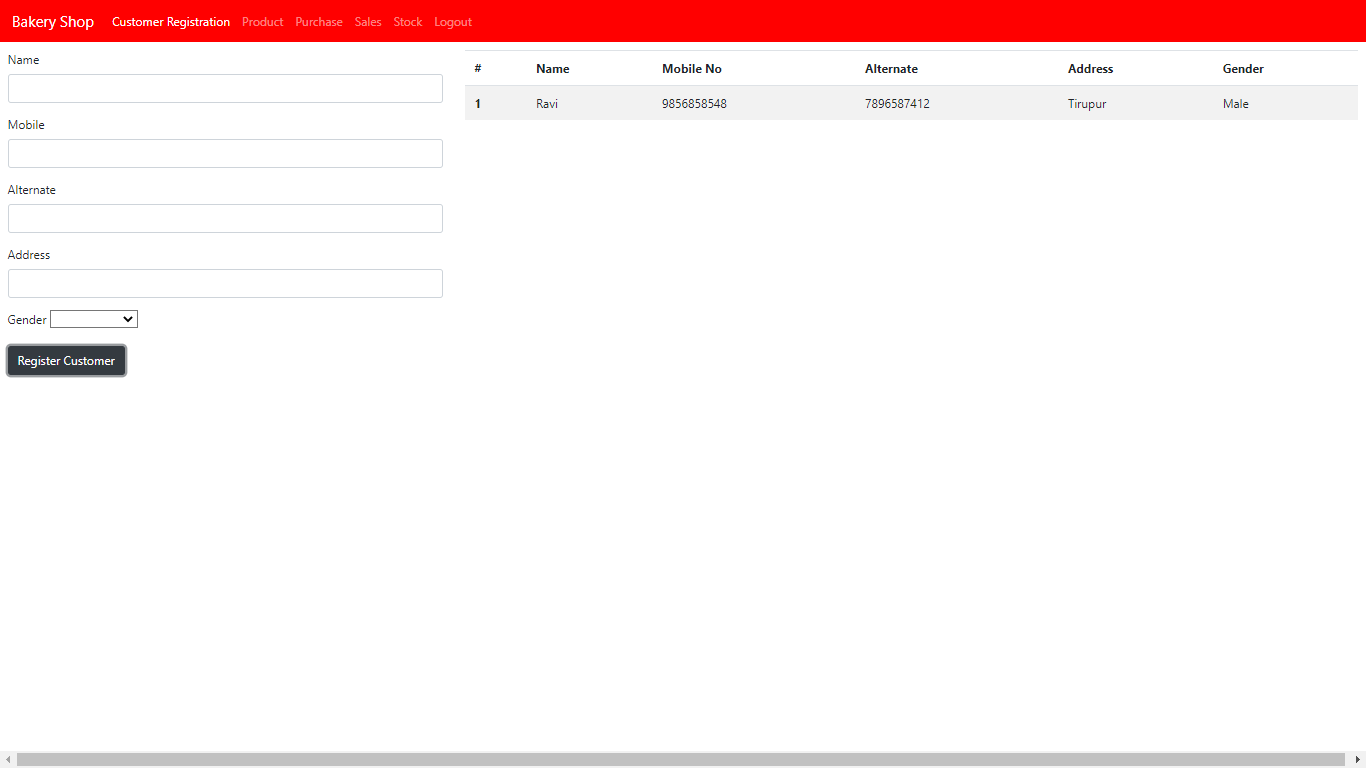
**Output of Invalid username or password**



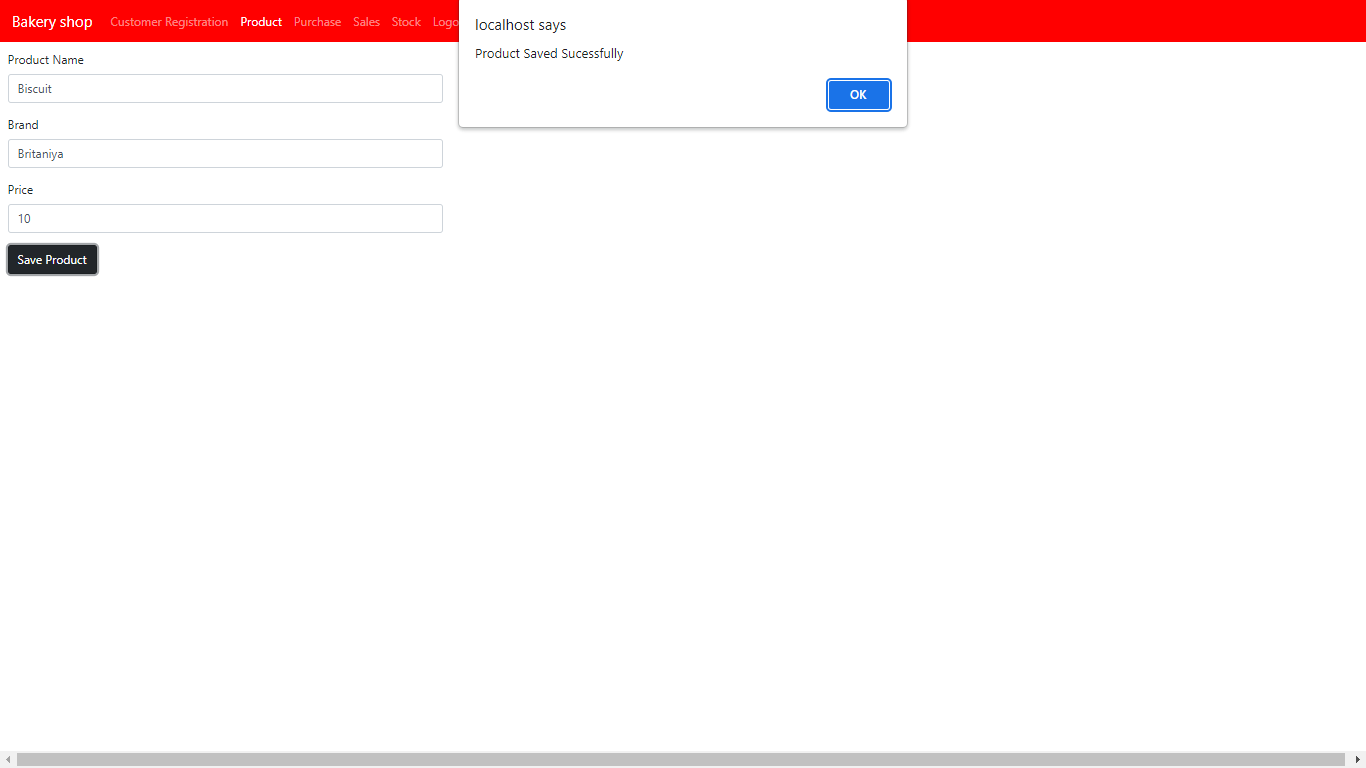
Output of customer Input Registration



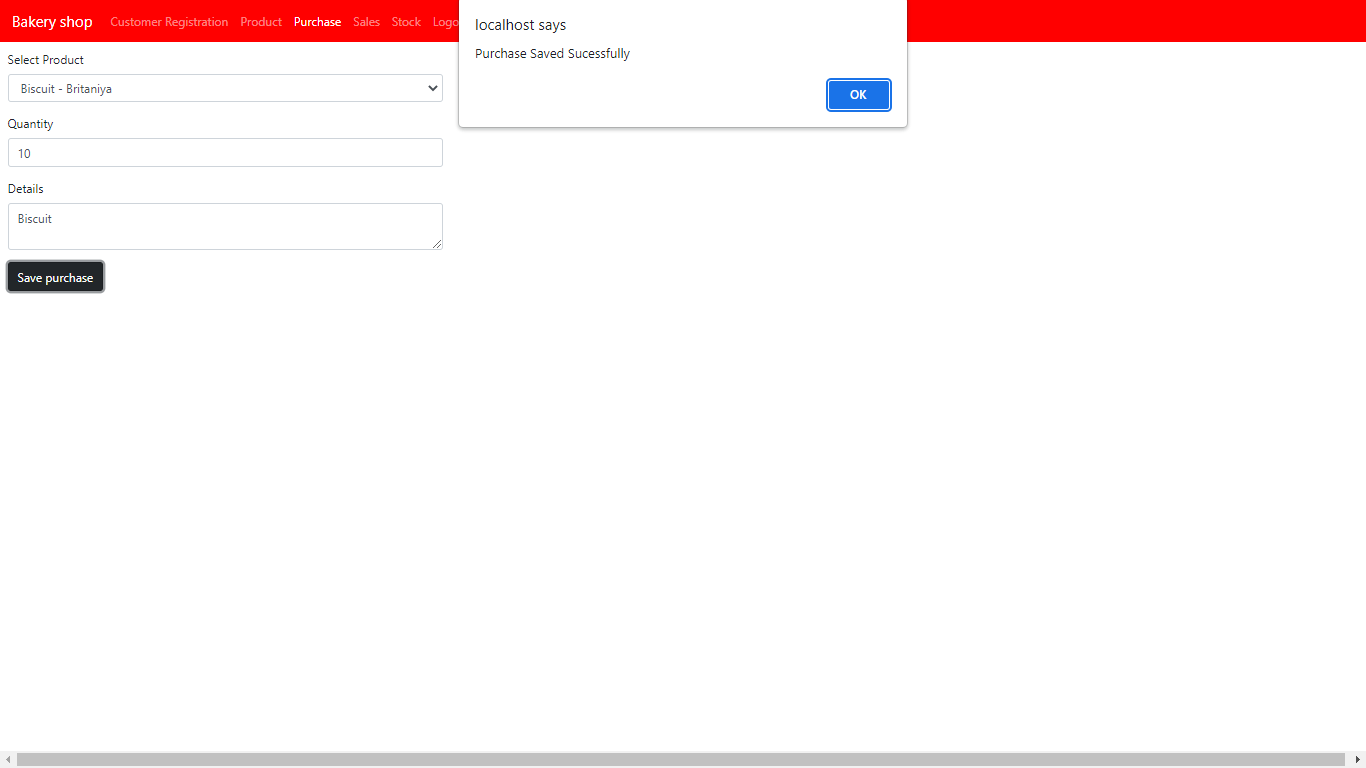
Output Of Customer Details



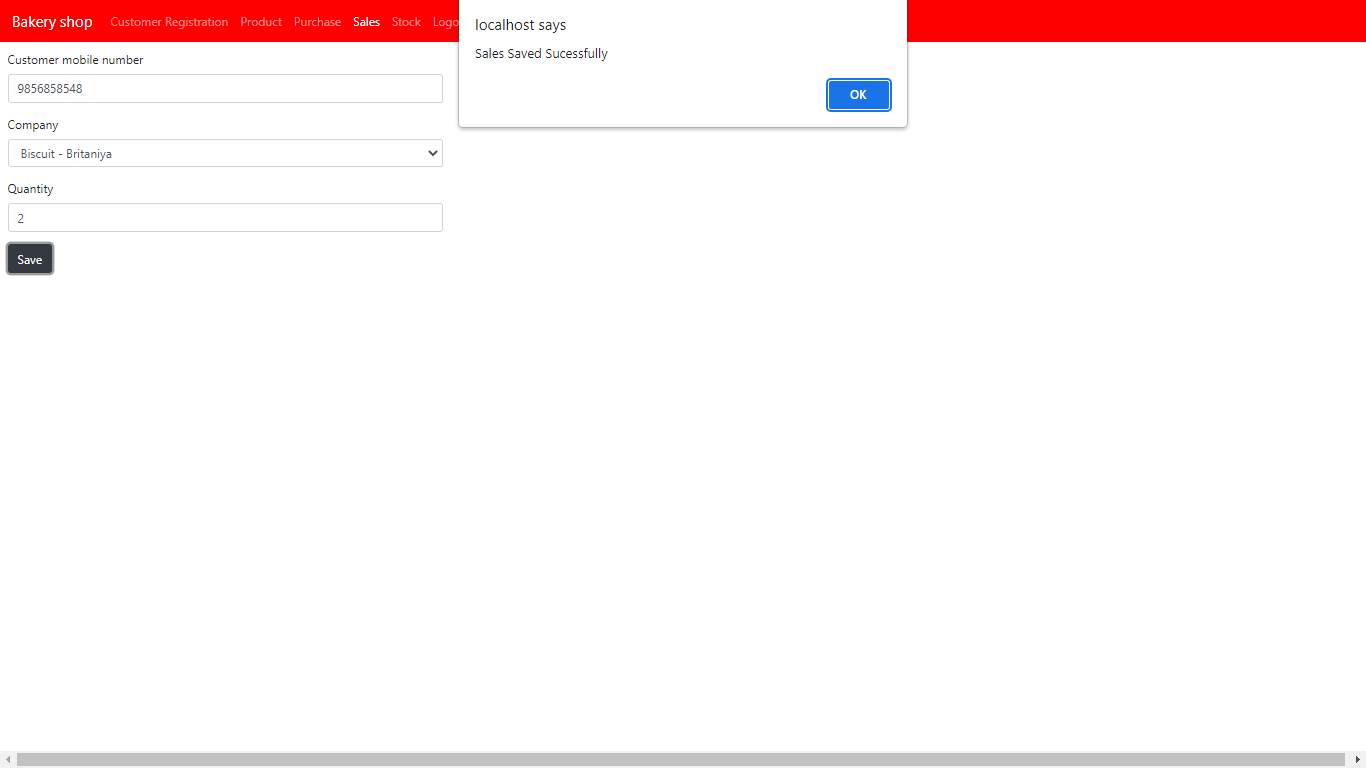
Output of product Input



Output of purchase Input



Output of sales input



Output of stock details

