MOBILE SHOP MANAGEMENT SYSTEM

Submitted in partial fulfillment of requirement for the award of the Degree

Bachelor of Computer Science

In the faculty of Computer Science of Bharathiar University, Coimbatore

Submitted by

R.DHANYA

(Reg.No.2022K0123)

Under the guidance of

DR.P.KOKILA Msc., M.Phil., Ph.D

Guest lecturer , Department of Computer Science

Department of Computer Science

L.R.G GOVERNMENT ARTS COLLEGE FOR WOMEN

(Affiliated To Bharathiar University)

TIRUPUR

APRIL-2023

CERTIFICATE

This is to certify that the project work entitled **“ MOBILE SHOP MANAGEMENT SYSTEM”** Submitted to Bharathiar University in partial fulfilled of the requirement for the award of the Degree of Bachelor of computer science is a record of the original work done by **Ms.R.DHANYA (Reg.No.2022K0123)** Under my supervisor and that project work has not formed the basis for the any Degree /Diploma /Association /Fellowship or similar title to any candidate of any university.

Internal Guide	**Head of the Department**
DR.P.KOKILA MSc.,M.Phil.,Ph.D	**DR.R.PARIMALA MSc.,M.Phil.,Ph.D**

Viva-voce examination is held on________________________L.R.G Government Arts College for Women, Tirupur-641604.

Internal Examiner	**External Examiner**

DECLARATION

I hereby declare that the project work submitted to the **Department of the Computer Science, L.R.G. Government Arts College for Women, Tirupur** , affiliated to Bharathiar University, Coimbatore in the partial fulfillment of the required for the award of Bachelor of Computer Science is an original work done by me during the sixth semester.

Place: **Signature of the Candidate**

Date: **(R.DHANYA)**

(Reg.No:2022K0123

ACKNOWLEDGEMENT

As first and foremost I would like to external my thankfulness to the almighty for blessing the work of my hands, I am grateful to my parents for continued encouragement that they had given to me.

I would like to external my profound gratitude and sincere thanks to **DR.M.R.YEZHILI MA.,M.Phil.,Ph.D** Principal, L.R.G Government Arts College For Women , for the encouragement rendered to me during thisproject.

It’s my privilege thank **DR.R.PARIMALA MSc.,M.Phil.,Ph.D** In-charge Head of the department of computer science for her valuable guidance and support throughout the project development work.

I express my deep sense of gratitude and sincere thanks to my guide **DR.P.KOKILA MSc.,M.Phil.,Ph.D** Guest Lecturer, Department of computer science, for providing all sorts of facilities to complete my project work successfully.

I also extend my sincere thanks to all the other faculty member ofthe department and technical assistance for their co-operation and valuable guidance.

I thank our college library for providing me with many informativebooks that help me to enrich my knowledge to bring out the project successfully.

SYNOPSIS

A Mobile shop management system, this project has only one user, the admin is an owner of the shop, login into the application and using all the features. For example, the admin has several shops the single man can’t be handle the all the store in same time but this application can do it. Because this application connects all the store into a single application. He can see all the store details in a single hand. Admin can create a store and login into the application they can entries purchase and sales entry, based on this we can generate billing details, which provide all the details of the application. Which is very important to customize all these features in single application but admin can do this.

The main goal of this project is an connect all the store into a single application and managing all the lifecycle of the project. The project is aimed to develop by Python as Front end and MYSQL as Back end.

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

Information technology has been growing rapidly, with the course of time, as technology and human knowledge coverage to a twilight, the need for the betterment of their own sustaining life also increases. Rapid need of high speed data access and faster delivery is one of the major requirement is one of the major requirement of a product. Technology has given many gifts to the human race to lead a sophisticated life. One of the fabulous gifts was Mobile phone which was invented by Joel Engel and Richard Frenkiel in 1983.

As the havoc increase in the platform of cellular phone, there must be well organized software which makes it possible for a management of a mobile store to maintain a healthy business. Thus getting interested on this matter we started thinking to develop a software name

"MOBILE SHOP MANAGEMENT SYSTEM"

It has a well define hardcore database of Microsoft SQL server 2005 were the details Of the specific data are safely kept in that Database and can also be retrieved as per need

1.1 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 SPECIFICATION

	Processor	: P 4 700 GHz.
➢	RAM	: 4 GB RAM
➢	Hard Disk Drive	: 180 GB

1.2.2 SOFTWARE SPECIFICATION

➢ Operating System	: Windows 7/8/10
➢ Front End	: PYTHON
➢ Back End	: MYSQL

1.2.3 SOFTWARE DESCRIPTION

WINDOW OS

Windows is a graphical operating system developed by Microsoft. It allows users to view and store files, run the software. 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 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.

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.

INTRODUCTION TO FRONT END

PYTHON

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of themost-used programming languages today. A survey conducted by industry analyst firmRedMonk found that it was the second-most popular programming language among developers in 2021.

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

"Writing programs is a very creative and rewarding activity," says University of Michigan and Courser instructor Charles R Severance in his book Python for Everybody. "You can write programs for many reasons, ranging from making your living to solving a difficult data analysis problem to having fun to helping someone else solve a problem."

INTRODUCTION TO BACKEND

MY-SQL 4.0

MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web applications and online publishing.

MySQL is an important component of an open source enterprise stack called LAMP. LAMP is a web development platform that uses Linux as the operating system, Apache as the web server, MySQL as the relational database management system and PHP as the object-oriented scripting language. (Sometimes Perl or Python is used instead of PHP.)

Originally conceived by the Swedish company MySQL AB, MySQL was acquired by SunMicrosystems in 2008 and then by Oracle when it bought Sun in 2010. Developers can useMySQL under the GNU General Public License (GPL), but enterprises must obtain a commercial license from Oracle.

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 the existing system all transaction, dealing of products, purchasing of products were done manually which is time consuming. To buy any product user has to collect information about it either by visiting the shop or ask people which are the better one.

2.1.1 DRAWBACKS

- Very difficult to manage store
- It takes more time to calculate daily reports
- Too hard to find missing accessories and mobile phones

2.2 PROPOSED SYSTEM

In this mobile store management system can easily solved this kind of issue. This is very help to manage mobile shop. Manager can easily find the stock of mobiles and mobile accessories. This is very help to avoid unwanted issues.

2.2.1FEATURES

- Maintain the stock details
- Managing the customer details

Managing the purchase and sales details

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 filesystem 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.

OBJECTIVES

- Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
- It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
- When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user
- Will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

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.

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 MODULE

CUSTOMER REGISTRATION

This module collects all the information about the customer like, mobile number, name and address etc… these fields are very help to getting further details about the customer. We can easily find out the regular customers.

EMPLOYEE REGISTRATION

The module will help to store the information about the employees, it collects all the information and store into the employee table. This module collect salary, mobile number etc…

PURSHASE MOBILES AND ACCESSORIES PRODUCT

In this module owner will added purchased mobile phones and accessories. Admin should register before selling that product. Each and every product should enter in this module, then only calculate the stock details if some items are missing to register, we can't get the exact stock report.

SALES PRODUCT

This module has used to register the product details before selling. So, we can identify the stock of product and billing details. Each and every sale will be store in the billing module, when enter the sales that item have reduce in the stock module.

BILLING DETAILS

We can calculate billing details based on sales details. This module will generate billing details based on its sales. Billing details has shown the sales product details, it will be showing overall sales details report

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 isdisplayed

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 displayedby 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.

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 MAINTENTANCE

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.

PERVENTIVE 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

Mobile shop management systems are essential tools for managing and organizing mobile phone retail businesses. These systems help to automate several aspects of mobile phone sales, including inventory management, sales tracking, and customer relationship management.

Mobile shop management systems offer several benefits to mobile phone retailers, including improved efficiency, increased sales, and enhanced customer satisfaction. By automating various aspects of sales and inventory management, retailers can focus on providing better customer service, making sales, and growing their business.

These systems also offer real-time sales tracking, which enables retailers to monitor sales performance, identify trends, and make informed business decisions. They can use this data to adjust their pricing strategy, optimize their inventory, and improve their customer ser vice.

Moreover, mobile shop management systems offer customer relationship management tools, enabling retailers to provide personalized and targeted customer service. By tracking customer purchase history and preferences, retailers can offer tailored recommendations and promotions, leading to increased customer satisfaction and loyalty.

In conclusion, mobile shop management systems are powerful tools that help mobile phone retailers automate and optimize their sales and inventory management processes. These systems provide several benefits, including improved efficiency, increased sales, enhanced customer satisfaction, and real-time data insights. Mobile phone retailers can leverage these benefits to grow their businesses and provide better service to their customers.

FUTURE ENHANCEMENTS

The scope of the project includes that what all future enhancements can be done.

In this system to make it more feasible to us:-

Databases for different products range and strong can be provided.

Multilingual support can be provided so that it can be understandable by person.

More graphics can be added to make it more user-friendly and understandable.

Manage & backup versions of documents online.

BIBLIOGRAPHY

Textual Reference:

☐ Beazley, David M. Python Essential Reference. Addison-Wesley Professional, 2009.

☐ Brownlee, Jason. Machine Learning Mastery with Python. Machine Learning Mastery, 2016.

☐ Lutz, Mark. Learning Python, 5th Edition. O'Reilly Media, 2013

Online Reference:

☐ Python.org. "Official Python Documentation." Python.org, https://www.python.org/doc/.

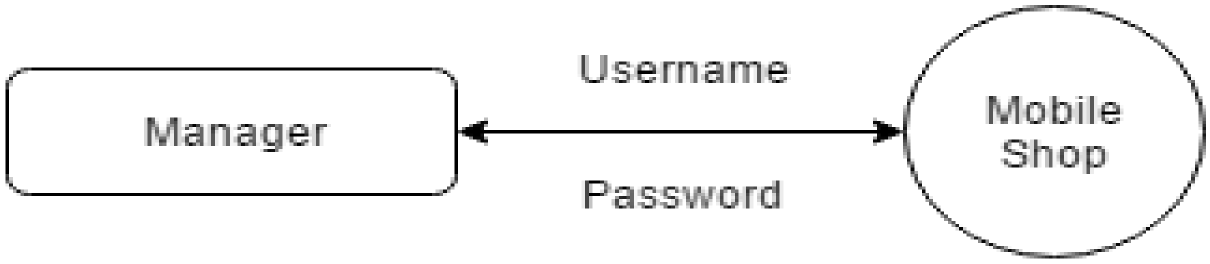
☐ Real Python. "Python Tutorials, Articles and News." Real Python, https://realpython.com/.

APPENDICES

A. DATA FLOW DIAGRAM

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or system. The DFD also provides information about the outputs and inputs of each entity and process itself. A data-flow diagram is a part of structured-analysis modeling tools.

LEVEL 0:



LEVEL 1:

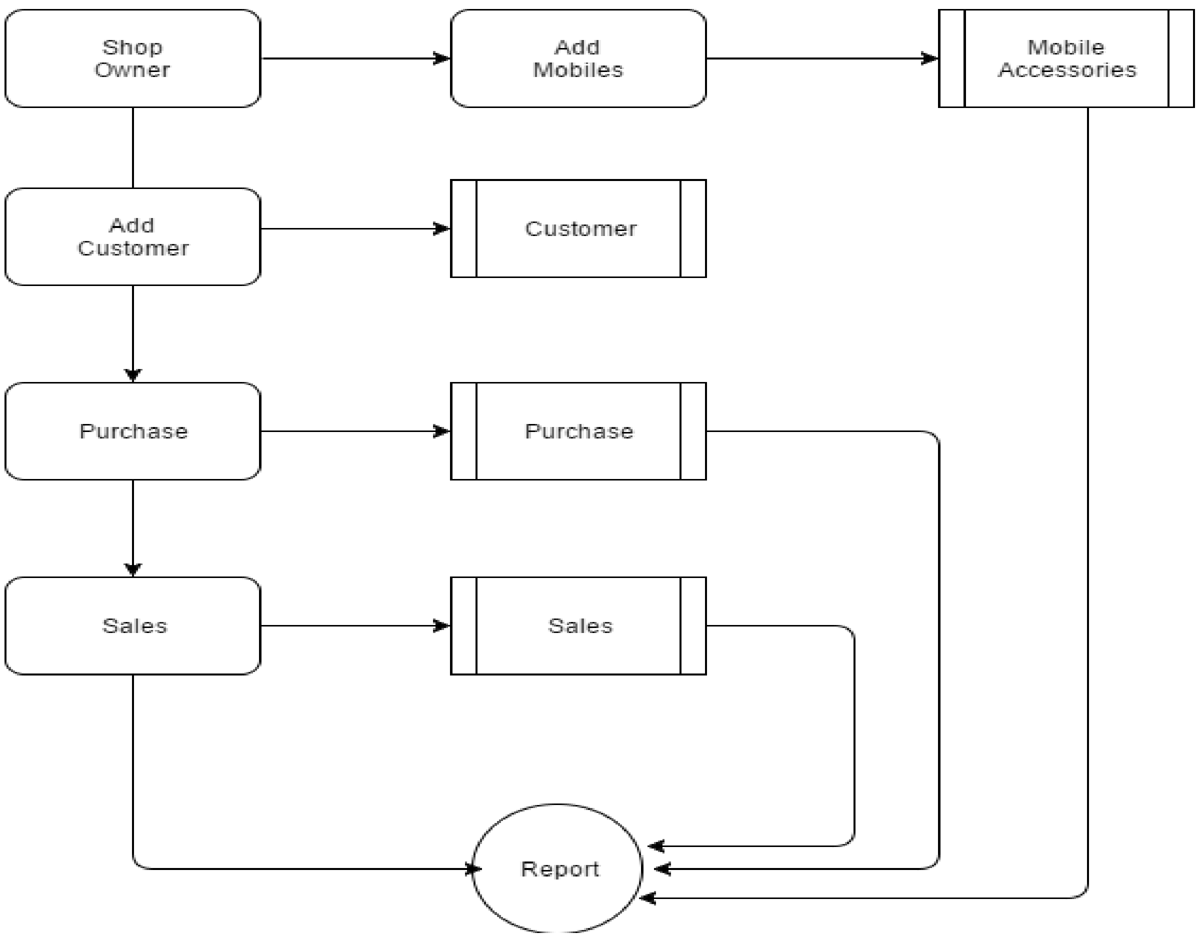

Shop Owner
Add Mobiles
Mobile Accessories
Add Customer
Customer
Purchase
Purchase
Sales
Sales
Report

TABLE STRUCTURE

The table needed for each module was 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.

TABLE NAME: ADMIN

FIELD	DATA TYPE	SIZE	CONSTRAINT
Admin id	INT	10	Primary key
Username	Varchar	20	Not null
Password	Varchar	20	Not null

TABLE NAME: EMPLOYEE

FIELD	DATA TYPE	SIZE	CONSTRAINT
Employee id	Int	10	Primary key
Name	Varchar	20	Not null
Mobile	Int	10	Not null
Age	Int	5	Not null
Gender	Varchar	6	Not null
Salary	Int	5	Not null

TABLE NAME: CUSTOMER

FIELD	DATA TYPE	SIZE	CONSTRAINT
Id	Int	10	Primary key
Name	Varchar	20	Not null
Mobile	Int	10	Not null
Alternate	Int	10	Not null
Address	Varchar	30	Not null
Gender	Varchar	10	Not null

TABLE NAME: MOBILE

FIELD	DATA TYPE	SIZE	CONSTRAINT
Mobile id	Int	10	Primary key
Mobile model	Varchar	20	Not null
Mobile brand	Varchar	20	Not null
Price	Int	5	Not null
RAM	Int	5	Not null
CAMERA PX	Int	5	Not null
STORAGE	Varchar	10	Not null

TABLE NAME: PURCHASE

FIELD	DATA TYPE	SIZE	CONSTRAINT
Purchase id	Int	10	Primary key
Product id	Int	10	Foreign key
Quantity	Int	5	Not null
Details	Varchar	30	Not null
Date	Date	10	Not null

TABLE NAME: SALES

FIELD	DATA TYPE	SIZE	CONSTRAINT
Sales id	Int	10	Primary key
Customer id	Int	10	Foreign key
Product id	Int	10	Foreign key
Quantity	Int	5	Not null

B. SAMPLE CODEING

```
#!/usr/bin/env python
# package:
com.example.demo.controllerimport
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.GetEmployeeResponse

import

com.example.demo.response.GetProductResponse

import

com.example.demo.response.GetStockResponse

import com.example.demo.service.ApiService


@RequestMapping(value="/api")
class ApiController(object):
    """ generated source for class
    ApiController """service = ApiService()

    @GetMapping("/login/{username}/{passwor
    d}")def login(self, username, password):
```

```
    """ generated source for method login """
    return self.service.login(username,
    password)

@PostMapping("/add_employee/{name}/{mobile}/{address}/{gender}/{salary}/{
age}") def add_employee(self, name, mobile, address, gender, salary, age):

    """ generated source for method add_employee """
    self.service.add_employee(name, mobile, address, gender, salary,
    age)return "Employee Saved Sucessfully"

@GetMapping("/get_employe
e")def get_employee(self):

    """ generated source for method get_employee """
    return ResponseEntity.ok().body(self.service.get_employee())


@PostMapping("/add_customer/{name}/{mobile}/{address}/{gender}/{ema
il}") def add_customer(self, name, mobile, address, gender, email):

    """ generated source for method add_customer """
    self.service.add_customer(name, mobile, address, gender,
    email)return "Customer Saved Sucessfully"

@GetMapping("/get_customer
")def get_customer(self):

    """ generated source for method get_customer """
    return ResponseEntity.ok().body(self.service.get_customer())


@PostMapping("/add_product/{company}/{model}/{pri
ce}") def add_product(self, company, model, price):

    """ generated source for method add_product
    """          self.service.add_product(company,
    model,    price)   return    "Product    Saved
    Sucessfully"
```

```
@GetMapping("/get_product")
def get_product(self):

   """ generated source for method get_product """
   return ResponseEntity.ok().body(self.service.get_product())


@PostMapping("/add_purchase/{product_id}/{quantit
y}")def add_purchase(self, product_id, quantity):

   """ generated source for method add_purchase
   """        self.service.add_purchase(product_id,
   quantity)
```

```
        return "Purchase Saved Sucessfully"


    @PostMapping("/add_sales/{customer_id}/{product_id}/{quant
    ity}")def add_sales(self, customer_id, product_id, quantity):

        """ generated source for method add_sales """
        self.service.add_sales(customer_id, product_id, quantity)
        return "Sales Saved Sucessfully"

    @GetMapping("/get_customer/{mobile
    }")def get_mobile(self, mobile):

        """ generated source for method
        get_mobile """return
        self.service.get_mobile(mobile)

    @GetMapping("/get_stock
    ")def get_stock(self):

        """ generated source for method get_stock """
        return ResponseEntity.ok().body(self.service.get_stock())


    @GetMapping("/get_billin
    g")def get_billing(self):

        """ generated source for method get_billing """
        return ResponseEntity.ok().body(self.service.get_billing())


        #!/usr/bin/env python
# package:
com.example.demo.daoimport
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

class ApiDao(object):
    """ generated source for class ApiDao
    """sf = SessionFactory()

    def add_employee(self, name, mobile, address, gender, salary,
        age):""" generated source for method add_employee """
        session = self.sf.getCurrentSession()
        sql = "INSERT INTO `employee` (`id`, `name`, `mobile`, `address`, `gender`, `salary`,
`age`) VALUES (NULL, '" + name + "', '" + mobile + "', '" + address + "', '" + gender +
"', '" +salary + "', '" + age + "');"
        session.createSQLQuery(sql).executeUpdate()


    def add_customer(self, name, mobile, address, gender,
        email):""" generated source for method add_customer
        """
        # TODO Auto-generated method stub
        session = self.sf.getCurrentSession()
        sql = "INSERT INTO `customer` (`id`, `name`, `mobile`, `address`, `gender`,
`email`)VALUES (NULL, '" + name + "', '" + mobile + "', '" + address + "', '" + gender +
"', '" + email
+ "');"
        session.createSQLQuery(sql).executeUpdate()


    def get_employee(self):
        """ generated source for method
        get_employee """#  TODO Auto-generated
        method stub
        session =
        self.sf.getCurrentSession()sql =
        "Select * from employee"
        nq =
        session.createNativeQuery(sql)
        return nq.list_()
```

```
def get_customer(self):
    """ generated source for method
    get_customer """session =
    self.sf.getCurrentSession()
    sql = "Select * from customer"
    nq =
    session.createNativeQuery(sql)
    return nq.list_()

def add_product(self, company, model, price):
    """ generated source for method
    add_product """#  TODO Auto-generated
    method stub
    session = self.sf.getCurrentSession()
    sql = "INSERT INTO `product` (`id`, `company`, `model`, `price`) VALUES (NULL, '"
+ company + "', '" + model + "', '" + price + "');"
        session.createSQLQuery(sql).executeUpdate()


    def get_product(self):
        """ generated source for method
        get_product """#  TODO Auto-generated
        method stub

        session =
        self.sf.getCurrentSession()sql =
        "Select * from product"

        nq =
        session.createNativeQuery(sql)
        return nq.list_()


    def add_purchase(self, product_id, quantity):
        """ generated source for method
        add_purchase """#  TODO Auto-generated
        method stub

        session = self.sf.getCurrentSession()
        sql = "INSERT INTO `purchase` (`id`, `product_id`, `quantity`, `date`)
VALUES(NULL, '" + product_id + "', '" + quantity + "', current_timestamp());"

        session.createSQLQuery(sql).executeUpdate()


    def add_sales(self, customer_id, product_id,
        quantity):""" generated source for method
        add_sales """

        # TODO Auto-generated method stub
        session = self.sf.getCurrentSession()

        sql = "INSERT INTO `sales` (`id`, `customer_id`, `product_id`, `quantity`)
VALUES(NULL, '" + customer_id + "', '" + product_id + "', '" + quantity + "');"

        session.createSQLQuery(sql).executeUpdate()


    def get_stock(self):
```

```
    """ generated source for method
    get_stock """#  TODO Auto-generated
    method stub session =
    self.sf.getCurrentSession()

    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"

    nq =
    session.createNativeQuery(sql)
    return nq.list_()

  def get_mobile(self, mobile):
    """ generated source for method
    get_mobile """session =
    self.sf.getCurrentSession()

    sql = "Select id,name from customer where mobile='" + mobile
    + "'"nq = session.createNativeQuery(sql)

    list_ =
    nq.getResultList()if
    len(list_) != 0:

      return
    int(list_.get(0)[0])else:

      return None


  def get_billing(self):
    """ generated source for method
    get_billing """#  TODO Auto-generated
    method stub

    session = self.sf.getCurrentSession()
    sql                                 =
                                        "select
customer.name,customer.mobile,product.company,product.model,sales.quantity,produ
ct.price from sales LEFT JOIN customer on(customer.id=sales.customer_id) LEFT
JOIN producton(product.id=sales.product_id) "

    nq =
    session.createNativeQuery(sql)
    return nq.list_()


  def login(self, username, password):
    """ generated source for method login
    """# TODO Auto-generated method
    stub session =
    self.sf.getCurrentSession()

    sql = "select * from admin where username='" + username + "' and
password='" +password + "'"
```

```
    nq =
    session.createNativeQuery(sql)if
    nq.list_().size() != 0:

       return
    Trueelse:

       return False


    #!/usr/bin/env python
# package:
com.example.demo.configurationimport
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

@Value("${db.driver}")
@Value("${db.password}")
@Value("${db.url}")
@Value("${db.username}")
@Value("${hibernate.dialect}")
@Value("${hibernate.show_sql}")
@Value("${hibernate.hbm2ddl.auto}")
@Value("${entitymanager.packagesToSca
n}")class HibernateConfiguration(object):
   """ generated source for class
   HibernateConfiguration """DB_DRIVER = str()
   DB_PASSWORD = str()
   DB_URL =  str()
   DB_USERNAME = str()
   HIBERNATE_DIALECT =
   str()
   HIBERNATE_SHOW_SQL =
   str()
```

```
HIBERNATE_HBM2DDL_AUTO = str()
ENTITYMANAGER_PACKAGES_TO_SCAN =
str()

def sessionFactory(self):
   """ generated source for method
   sessionFactory """sessionFactory =
   LocalSessionFactoryBean()
   sessionFactory.setDataSource(dataSource())

   sessionFactory.setPackagesToScan(self.ENTITYMANAGER_PACKAGES_TO_SC
   AN) hibernateProperties = Properties()

   hibernateProperties.put("hibernate.dialect", self.HIBERNATE_DIALECT)
   hibernateProperties.put("hibernate.show_sql", self.HIBERNATE_SHOW_SQL)
```

```
        hibernateProperties.put("hibernate.hbm2ddl.auto",
self.HIBERNATE_HBM2DDL_AUTO)
        sessionFactory.setHibernateProperties(hibernatePropertie
        s) return sessionFactory

    def dataSource(self):
        """ generated source for method dataSource """
        dataSource = DriverManagerDataSource()
        dataSource.setDriverClassName(self.DB_DRIV
        ER)dataSource.setUrl(self.DB_URL)
        dataSource.setUsername(self.DB_USERNAME
        )
        dataSource.setPassword(self.DB_PASSWORD)
        return dataSource

    def transactionManager(self):
        """ generated source for method transactionManager """
        txManager = HibernateTransactionManager()
        txManager.setSessionFactory(self.sessionFactory().getObject
        ()) return txManager

    #!/usr/bin/env python
# package:
com.example.demo.serviceimport
java.math.BigDecimal

import

java.util.ArrayList

import java.util.List

import javax.transaction.Transactional
```

```
import

org.springframework.beans.factory.annotation.Autowired

import org.springframework.stereotype.Service

import com.example.demo.dao.ApiDao

import

com.example.demo.response.GetBillingResponse

import

com.example.demo.response.GetCustomerResponse
```

```
import

com.example.demo.response.GetEmployeeResponse

import

com.example.demo.response.GetProductResponse

import

com.example.demo.response.GetStockResponse


class ApiService(object):
    """ generated source for class ApiService
    """dao = ApiDao()

    def add_employee(self, name, mobile, address, gender, salary,
        age): """ generated source for method add_employee """
        self.dao.add_employee(name, mobile, address, gender, salary,
        age)

    def add_customer(self, name, mobile, address, gender,
        email):""" generated source for method add_customer
        """

        # TODO Auto-generated method stub
        self.dao.add_customer(name, mobile, address, gender,
        email)

    def get_employee(self):
        """ generated source for method
        get_employee """#  TODO Auto-generated
        method stub

        result =
        self.dao.get_employee()
        response = ArrayList()
```

```
# TODO Auto-generated method
stub # TODO Auto-generated
method stub i = 0

while i < len(result):
    # TODO Auto-generated method
    stub# TODO Auto-generated
    method stubobj.setId(int(row[0]))
    obj.setName(str(row[1]))
    obj.setMobile(str(row[2]))
    obj.setAddress(str(row[3]))
    obj.setGender(str(row[4]))
    obj.setSalary(int(row[5]))
    obj.setAge(int(row[6]))
    response.add(obj)

    i += 1
return response
```

```
def get_customer(self):
    """ generated source for method
    get_customer """#  TODO Auto-generated
    method stub

    result =
    self.dao.get_customer()
    response = ArrayList()

    #  TODO  Auto-generated  method
    stub    #    TODO    Auto-generated
    method    stub    #    TODO    Auto-
    generated method stub i = 0

    while i < len(result):
        # TODO Auto-generated method
        stub# TODO Auto-generated
        method stub# TODO Auto-
        generated method stub
        obj.setId(int(row[0]))
        obj.setName(str(row[1]))
        obj.setMobile(str(row[2]))
        obj.setAddress(str(row[3]))
        obj.setGender(str(row[4]))
        obj.setEmail(str(row[5]))
        response.add(obj)

        i += 1
    return response


def add_product(self, company, model, price):
    """ generated source for method
    add_product """# TODO Auto-generated
    method stub self.dao.add_product(company,
    model, price)


def get_product(self):
```

```
""" generated source for method
get_product """#  TODO Auto-generated
method stub

result =
self.dao.get_product()
response = ArrayList()

# TODO Auto-generated method
stub # TODO Auto-generated
method stub # TODO Auto-
generated method stub # TODO
Auto-generated method stub #
TODO Auto-generated method
stub i = 0
```

```
    while i < len(result):
        # TODO Auto-generated method
        stub# TODO Auto-generated
        method stub# TODO Auto-
        generated method stub# TODO
        Auto-generated method stub#
        TODO Auto-generated method
        stubobj.setId(int(row[0]))
        obj.setCompany(str(row[1]))
        obj.setModel(str(row[2]))
        obj.setPrice(int(row[3]))
        response.add(obj)

        i += 1
    return response


def add_purchase(self, product_id, quantity):
    """ generated source for method
    add_purchase """# TODO Auto-generated
    method stub
    self.dao.add_purchase(product_id, quantity)


def add_sales(self, customer_id, product_id,
    quantity):""" generated source for method
    add_sales """

    # TODO Auto-generated method stub
    self.dao.add_sales(customer_id, product_id,
    quantity)


def get_stock(self):
    """ generated source for method get_stock
    """result = self.dao.get_stock()

    response = ArrayList()
    # TODO Auto-generated method
    stub # TODO Auto-generated
    method stub # TODO Auto-
```

```
generated method stub # TODO
Auto-generated method stub #
TODO Auto-generated method
stub # TODO Auto-generated
method stub # TODO Auto-
generated method stub i = 0

while i < len(result):
    # TODO Auto-generated method
    stub # TODO Auto-generated
    method stub # TODO Auto-
    generated method stub
```

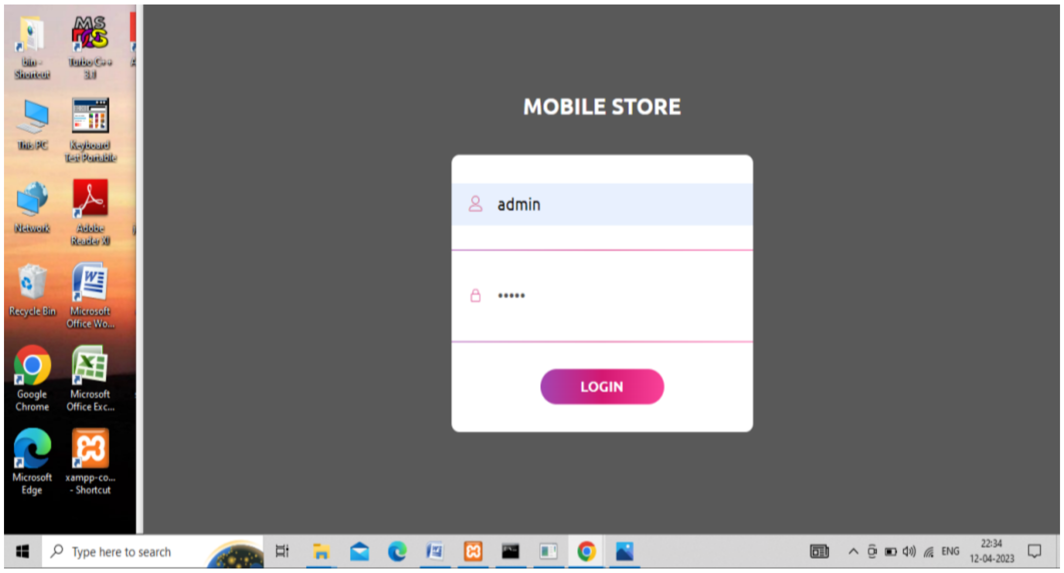
```
            # TODO  Auto-generated  method
            stub   #   TODO   Auto-generated
            method   stub   #   TODO   Auto-
            generated  method  stub  #   TODO
            Auto-generated method stub

            obj.setCompany_name(str(row[0]) + "-" +
            str(row[1]))obj.setQuantity(BigDecimal(row[2]))
            response.add(obj)

            i += 1
        return response


    def get_mobile(self, mobile):
        """ generated source for method
        get_mobile """#  TODO Auto-generated
        method stub

        return self.dao.get_mobile(mobile)


    def get_billing(self):
        """ generated source for method get_billing
        """result = self.dao.get_billing()

        response = ArrayList()
        #  TODO  Auto-generated  method
        stub   #   TODO   Auto-generated
        method   stub   #   TODO   Auto-
        generated  method  stub  #  TODO
        Auto-generated   method   stub   #
        TODO    Auto-generated    method
        stub   #   TODO   Auto-generated
        method   stub   #   TODO   Auto-
        generated  method  stub  #  TODO
        Auto-generated method stub i = 0

        while i < len(result):
            # TODO Auto-generated method
            stub# TODO Auto-generated
```

```
method stub# TODO Auto-
generated method stub# TODO
Auto-generated method stub#
TODO Auto-generated method
stub# TODO Auto-generated
method stub# TODO Auto-
generated method stub# TODO
Auto-generated method stub
obj.setCustomer_name(str(row[0]))
obj.setMobile(str(row[1]))
obj.setCompany(str(row[2]))
```

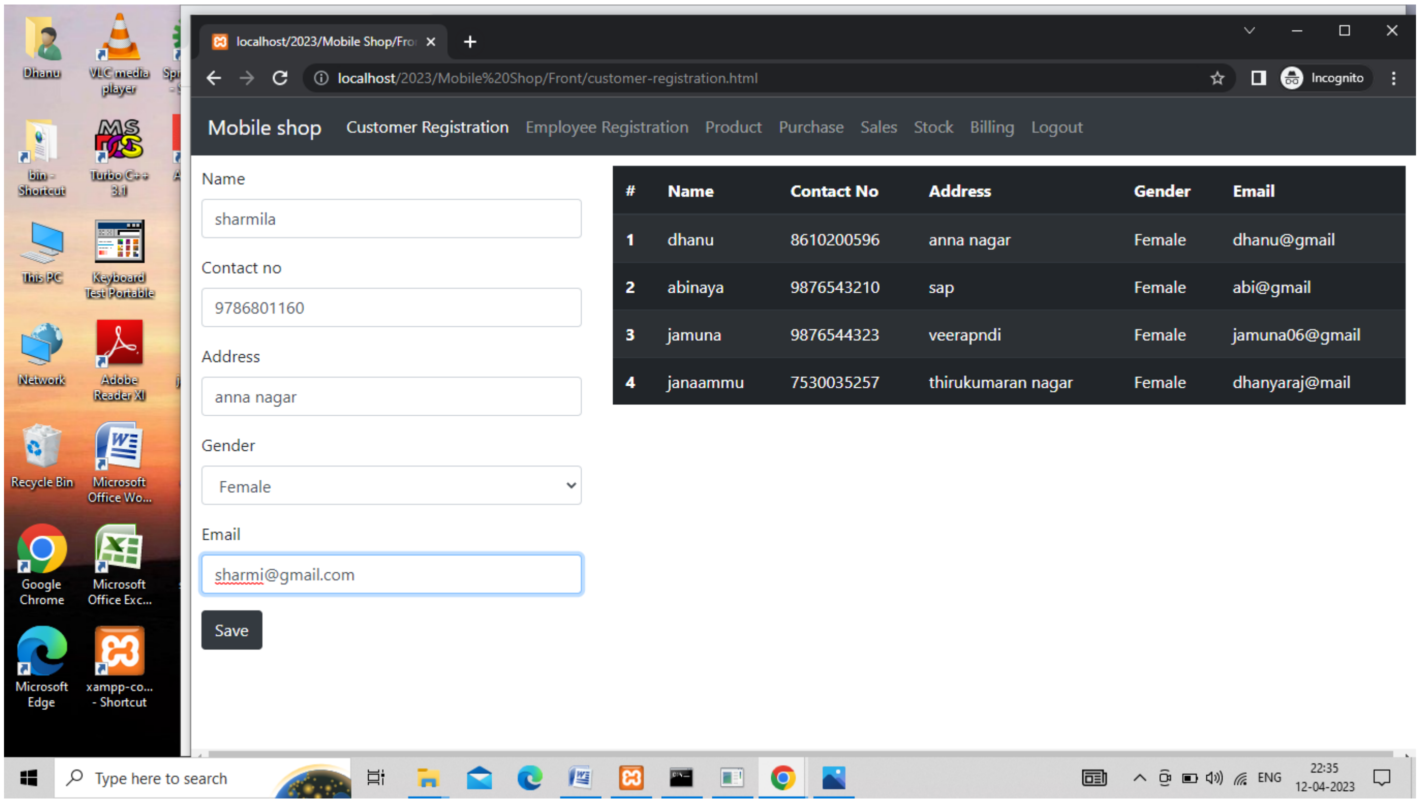
```
        obj.setModel(str(r
        ow[3]))
        obj.setQuantity(in
        t(row[4]))
        obj.setPrice(int(ro
        w[5]))
        response.add(obj)

        i += 1
    return response


def login(self, username, password):
    """ generated source for
    method login """return
    self.dao.login(username,
    password)
```

SAMPLE INPUT & OUTPUT DESIGN

LOGIN PAGE



CUSTOMER REGISTRATION



Name
sharmila

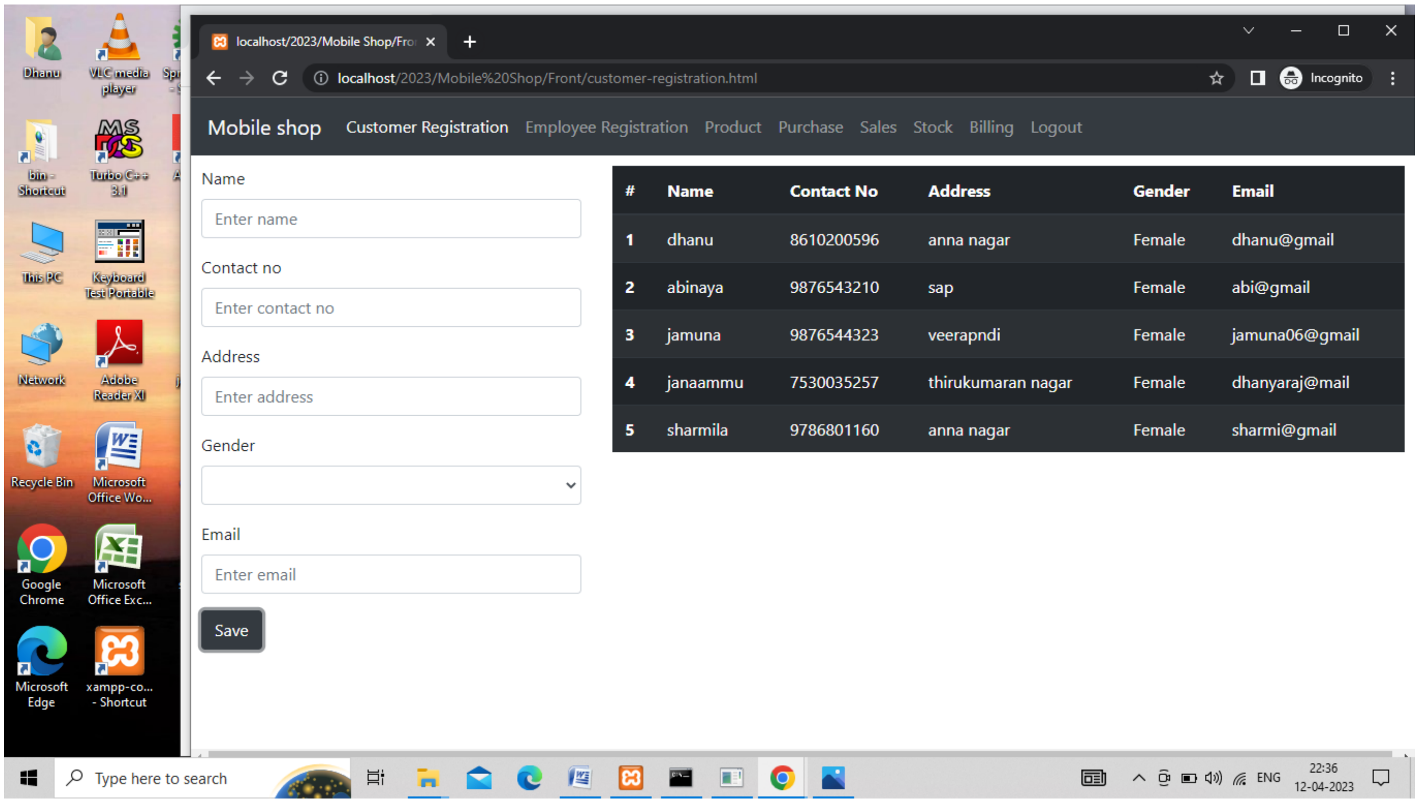
Contact no
9786801160

Address
anna nagar

Gender
Female

Email
sharmi@gmail.com

#	Name	Contact No	Address	Gender	Email
1	dhanu	8610200596	anna nagar	Female	dhanu@gmail
2	abinaya	9876543210	sap	Female	abi@gmail
3	jamuna	9876544323	veerapndi	Female	jamuna06@gmail
4	janaammu	7530035257	thirukumaran nagar	Female	dhanyaraj@mail



Name
Enter name

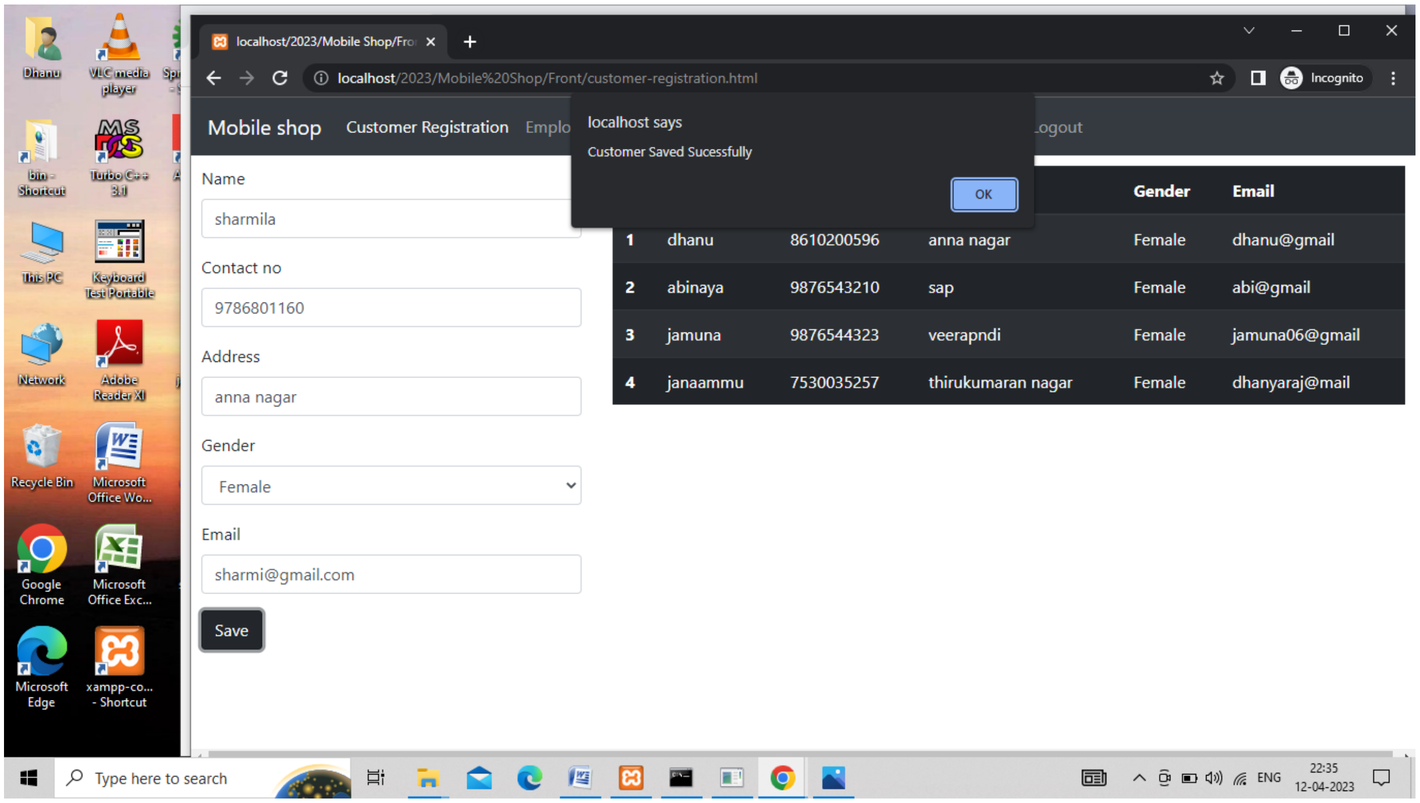
Contact no
Enter contact no

Address
Enter address

Gender

Email
Enter email

#	Name	Contact No	Address	Gender	Email
1	dhanu	8610200596	anna nagar	Female	dhanu@gmail
2	abinaya	9876543210	sap	Female	abi@gmail
3	jamuna	9876544323	veerapndi	Female	jamuna06@gmail
4	janaammu	7530035257	thirukumaran nagar	Female	dhanyaraj@mail
5	sharmila	9786801160	anna nagar	Female	sharmi@gmail



localhost/2023/Mobile Shop/Front/customer-registration.html

Mobile shop Customer Registration Emplo Logout

localhost says
Customer Saved Sucessfully
OK

Name
sharmila

Contact no
9786801160

Address
anna nagar

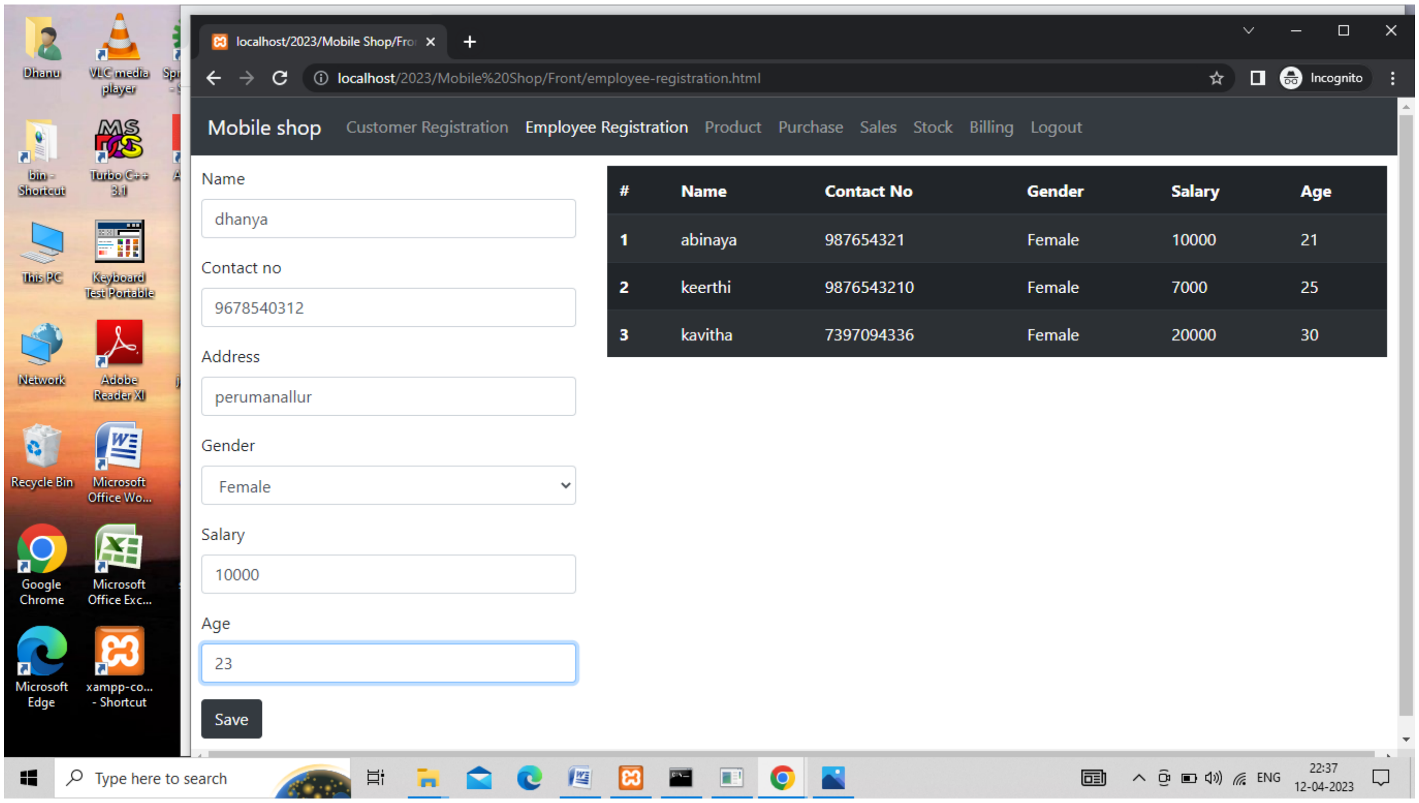
Gender
Female

Email
sharmi@gmail.com

Save

				Gender	Email
1	dhanu	8610200596	anna nagar	Female	dhanu@gmail
2	abinaya	9876543210	sap	Female	abi@gmail
3	jamuna	9876544323	veerapndi	Female	jamuna06@gmail
4	janaammu	7530035257	thirukumaran nagar	Female	dhanyaraj@mail

EMPLOYEE REGISTRATION



localhost/2023/Mobile%20Shop/Front/employee-registration.html

Name: dhanya

Contact no: 9678540312

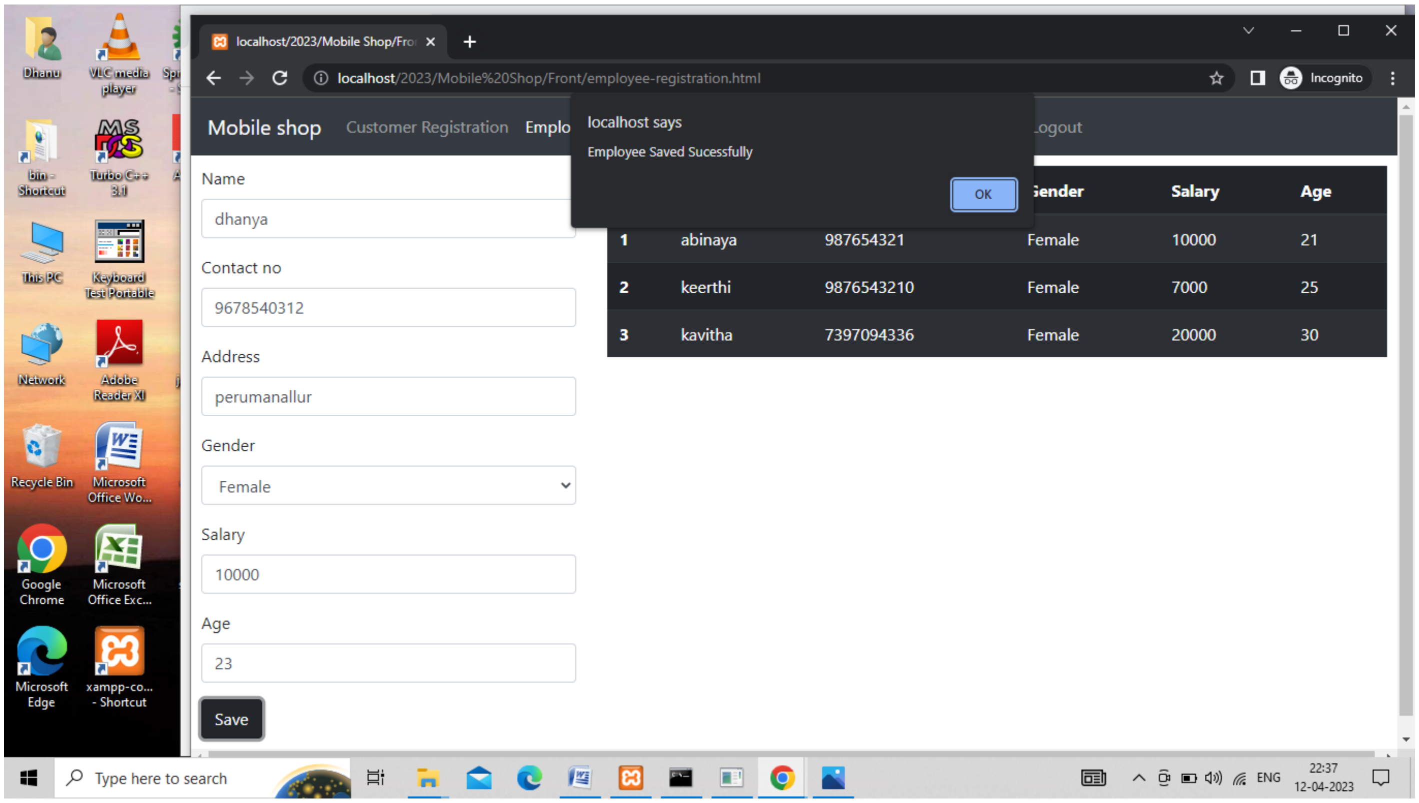
Address: perumanallur

Gender: Female

Salary: 10000

Age: 23

#	Name	Contact No	Gender	Salary	Age
1	abinaya	987654321	Female	10000	21
2	keerthi	9876543210	Female	7000	25
3	kavitha	7397094336	Female	20000	30



localhost says

Employee Saved Sucessfully

Name: dhanya

Contact no: 9678540312

Address: perumanallur

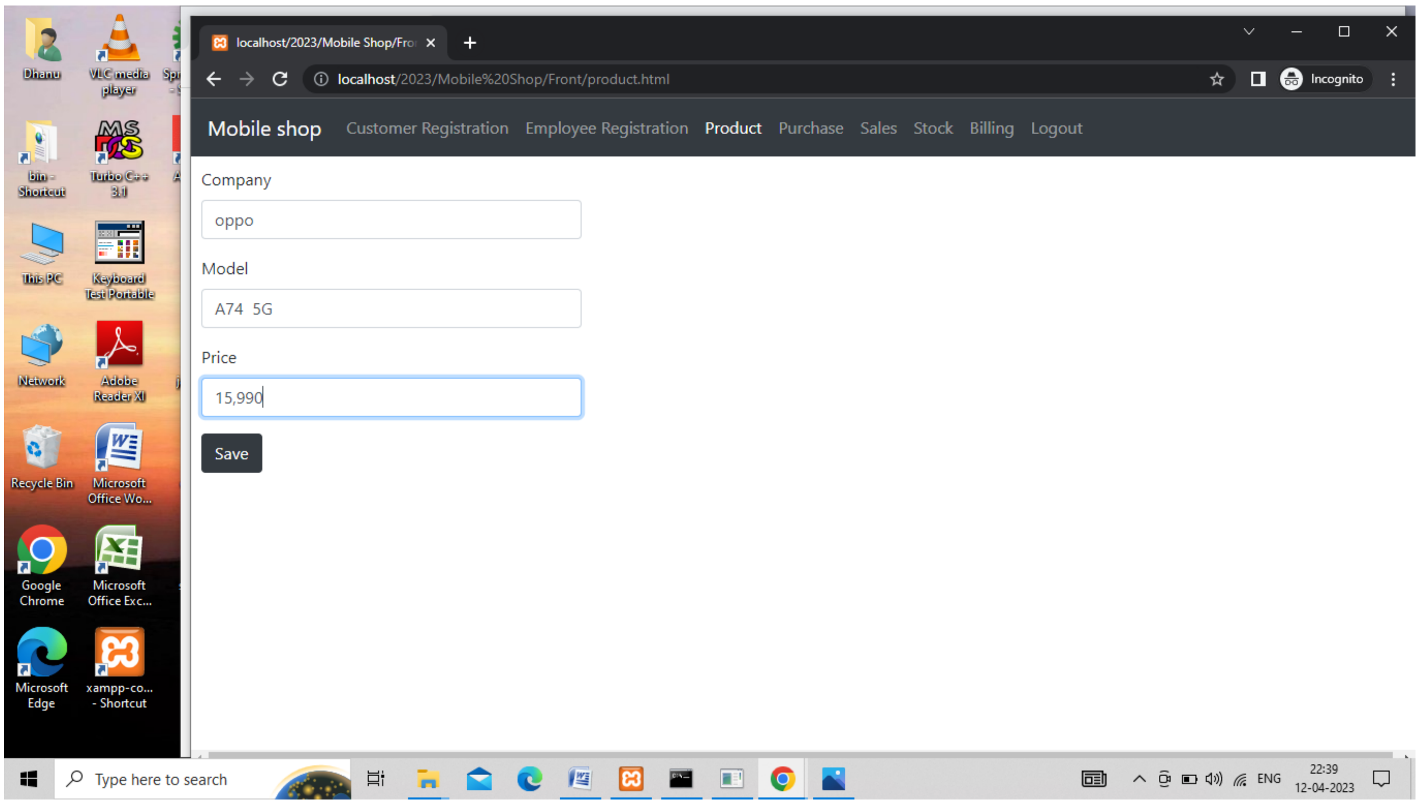
Gender: Female

Salary: 10000

Age: 23

#	Name	Contact No	Gender	Salary	Age
1	abinaya	987654321	Female	10000	21
2	keerthi	9876543210	Female	7000	25
3	kavitha	7397094336	Female	20000	30

PRODUCT DETAILS



Company

oppo

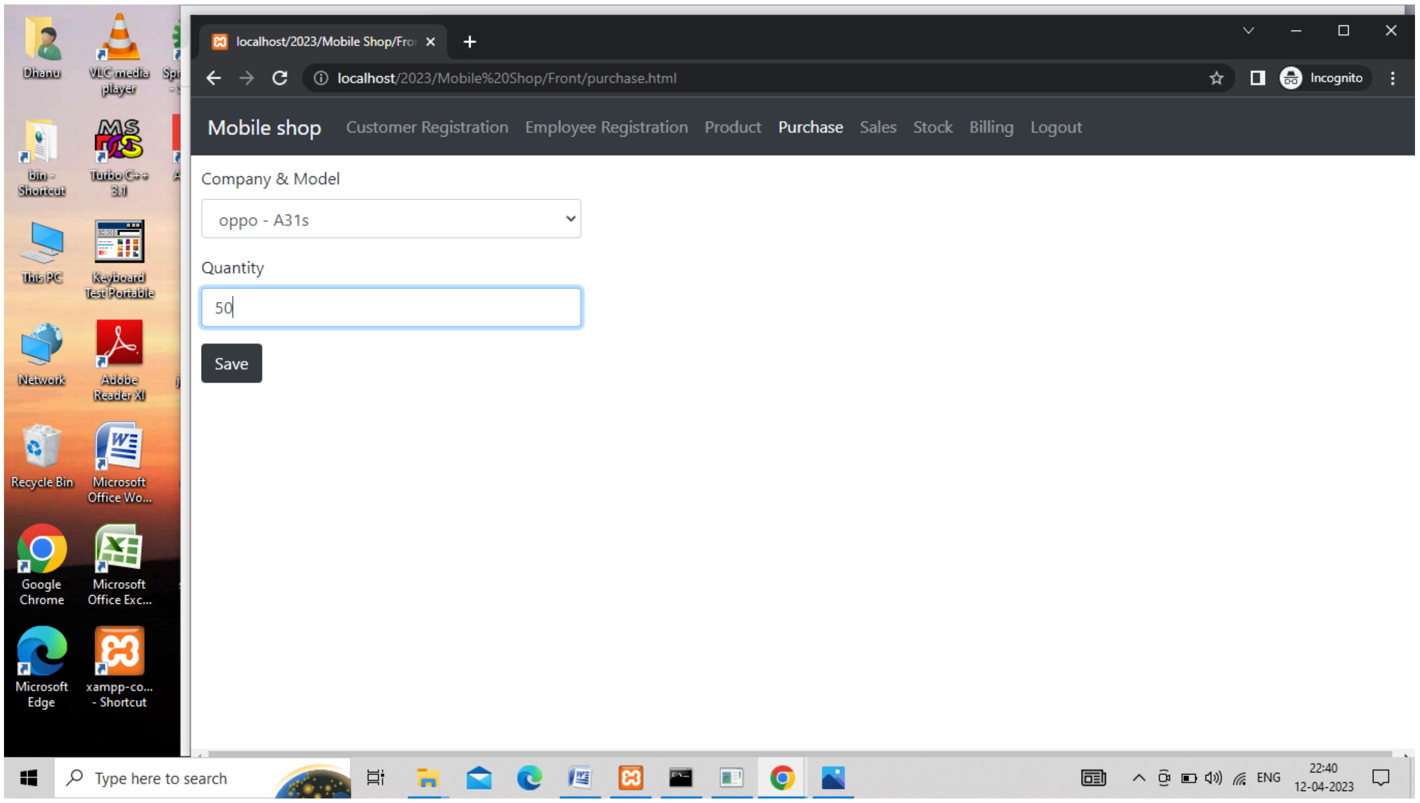
Model

A74 5G

Price

15,990

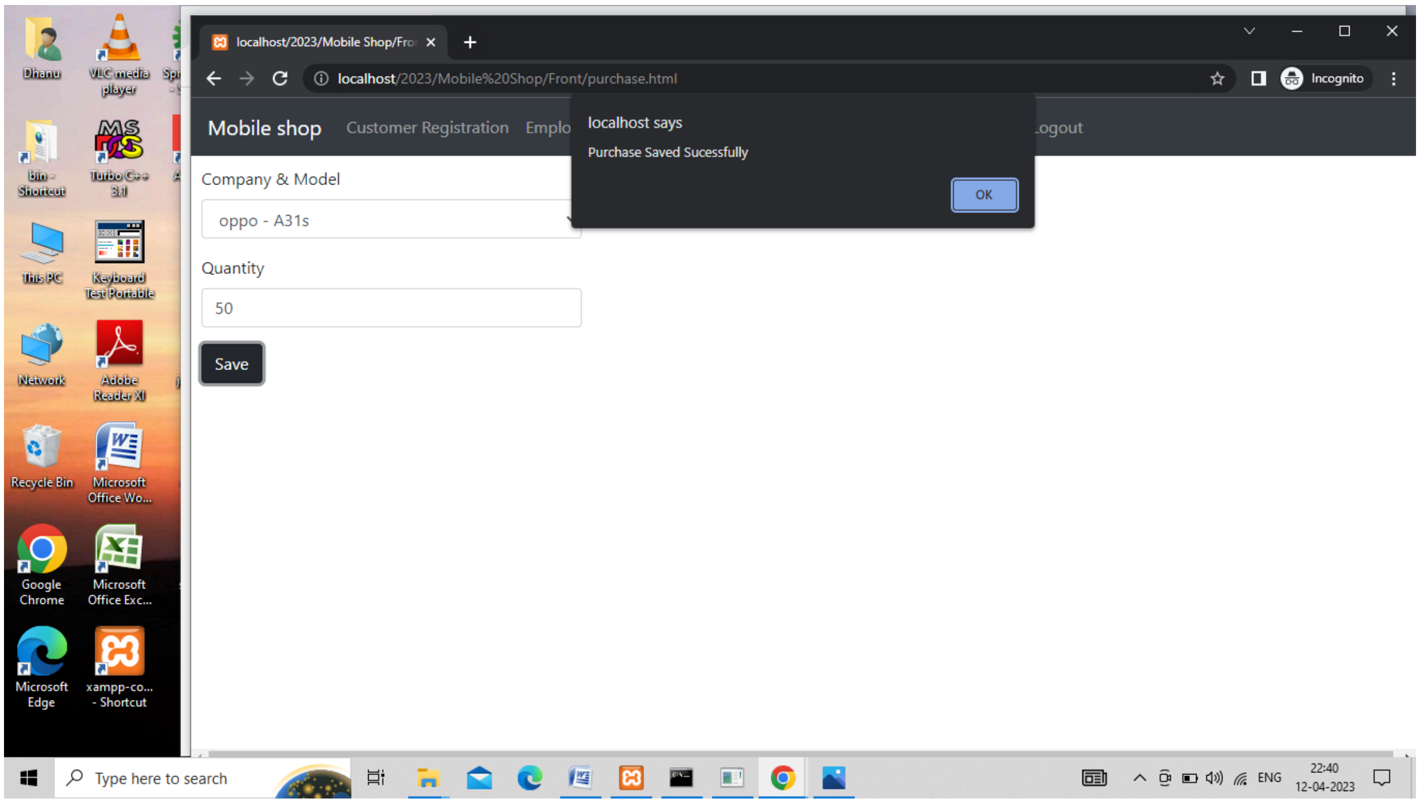
PURCHASE DETAILS


Mobile shop — Customer Registration, Employee Registration, Product, Purchase, Sales, Stock, Billing, Logout

Company & Model: oppo - A31s

Quantity: 50

Save


localhost says

Purchase Saved Sucessfully

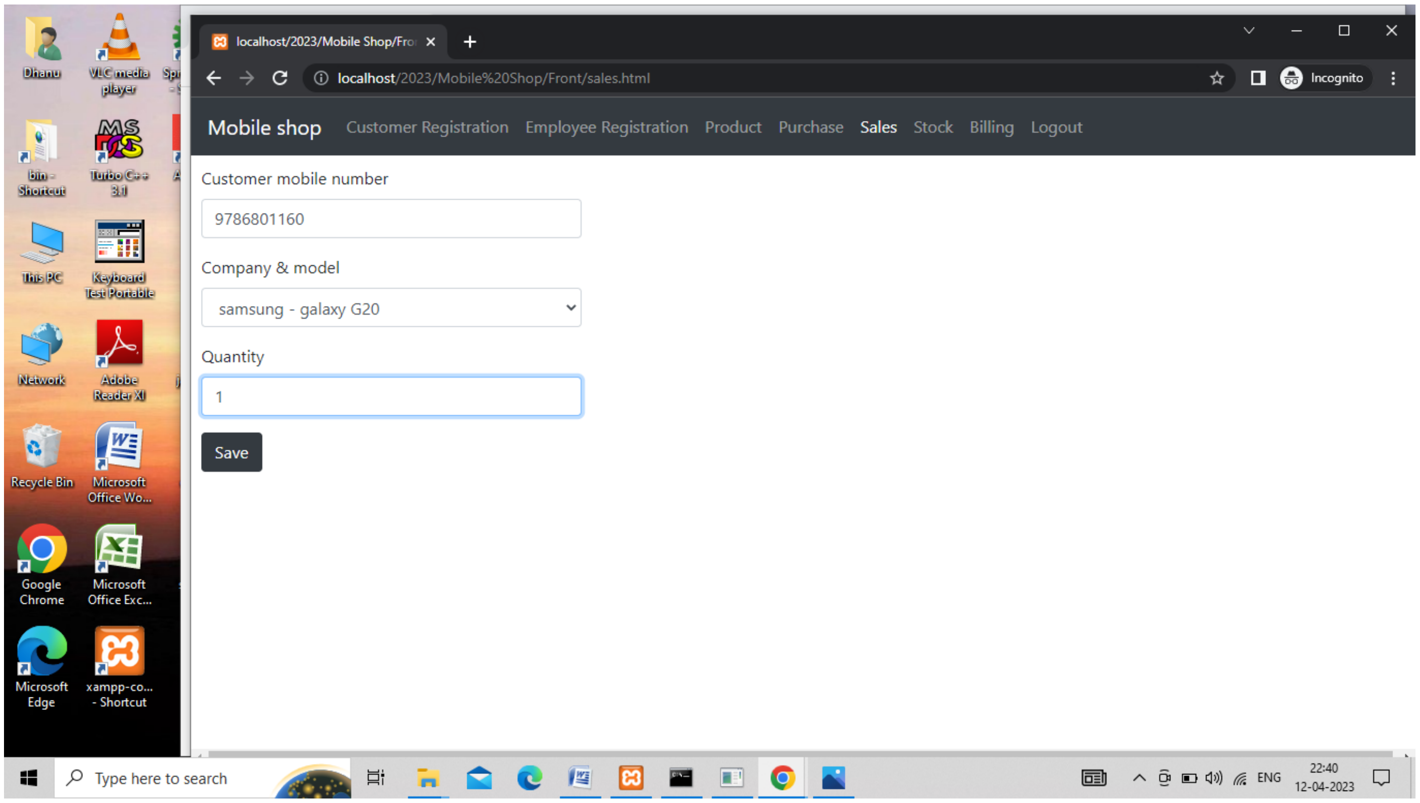
OK

Company & Model: oppo - A31s

Quantity: 50

Save

SALES DETAILS



localhost/2023/Mobile%20Shop/Front/sales.html

Customer mobile number

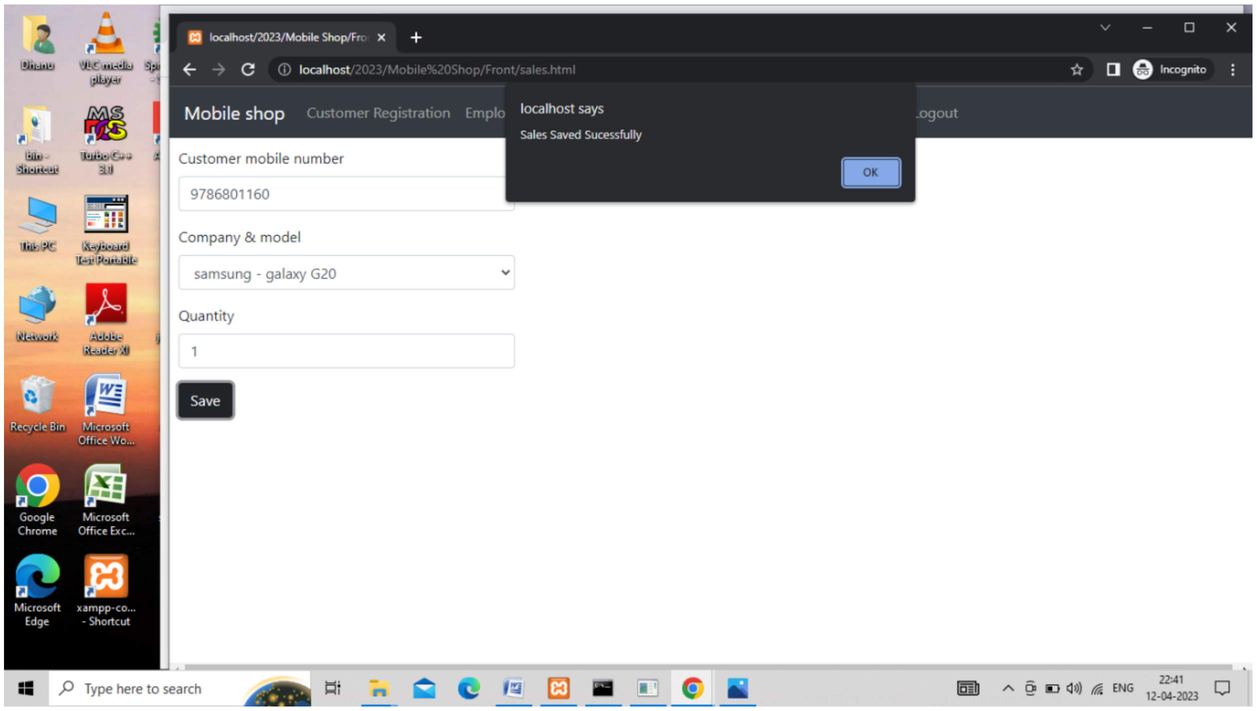
9786801160

Company & model

samsung - galaxy G20

Quantity

1



localhost says

Sales Saved Sucessfully

Customer mobile number

9786801160

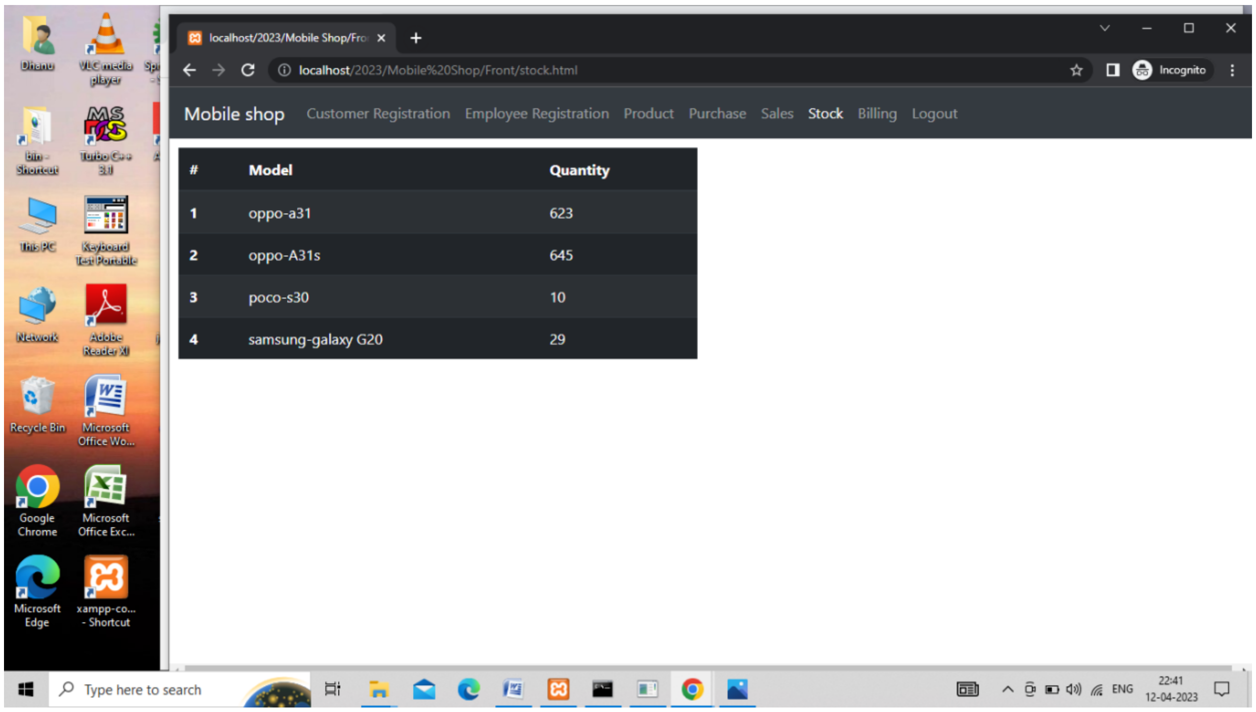
Company & model

samsung - galaxy G20

Quantity

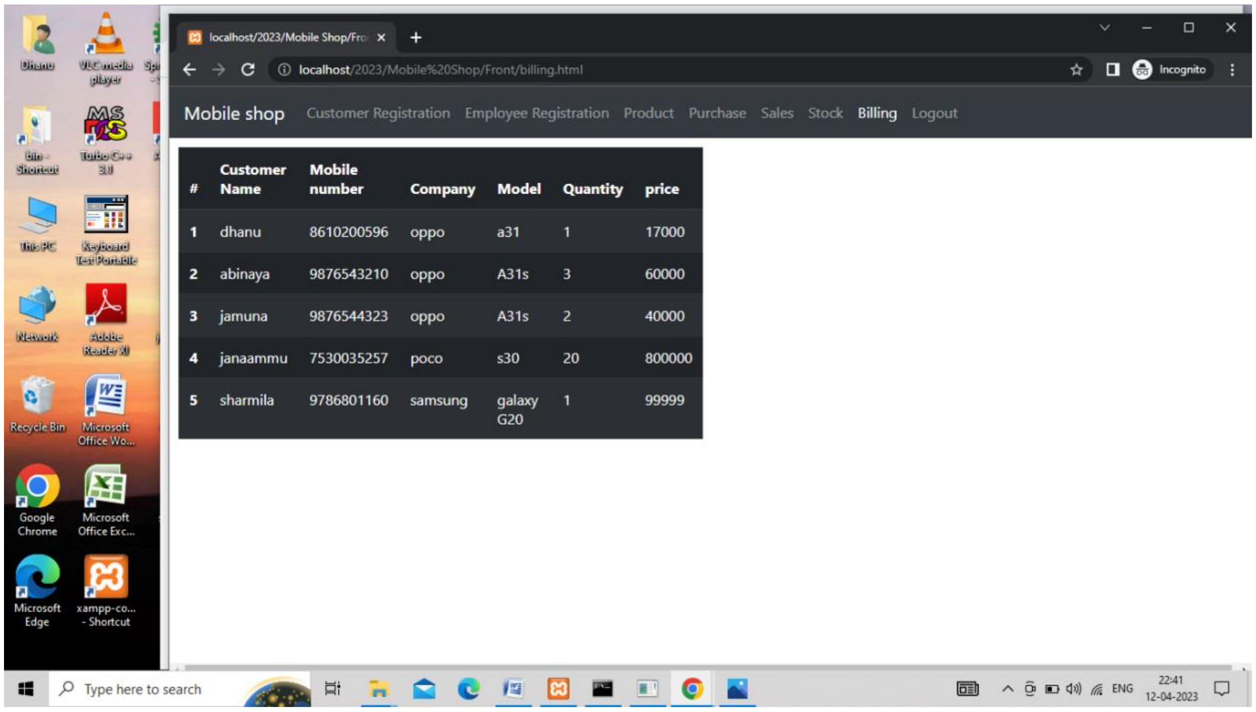
1

STOCK DETAILS



#	Model	Quantity
1	oppo-a31	623
2	oppo-A31s	645
3	poco-s30	10
4	samsung-galaxy G20	29

BILLING DETAILS



#	Customer Name	Mobile number	Company	Model	Quantity	price
1	dhanu	8610200596	oppo	a31	1	17000
2	abinaya	9876543210	oppo	A31s	3	60000
3	jamuna	9876544323	oppo	A31s	2	40000
4	janaammu	7530035257	poco	s30	20	800000
5	sharmila	9786801160	samsung	galaxy G20	1	99999