1.INTRODUCTION

**1.1 INTRODUCTION OF THE SYSTEM**

**1.1.1 Title of the Project:**

Havmor With Us

**1.1.2 PROJECT CATEGORY:**

Web Application using python-django

**1.1.3 OVERVIEW:**

A havmor with us is a software application designed to help ice-cream distributor. The system typically includes features such as customer management, inventory management, order management, billing, analytics, and more. Here customer can purchase products online.

**1.2 OBJECTIVES OF THE SYSTEM:**

The main objectives of this project is to help the distributor and customer. It is used to help the distributor to check all the details and transaction. The objective of the project is to maintain the transaction with the people for the services and satisfying the people with the final bills. This system stores all the records about the services.

**1.3 SCOPE OF THE SYSTEM**

* Mobile application can be developed.
* Website can be developed in multiple languages like Hindi, Kannada, etc..
* Online payment can be added

**1.4 STRUCTURE OF THE SYSTEM:**

**Module Description**

**1.4.1 Login MODULE:**

This module allows the admin and customer to login to the system by entering valid email and password.

**1.4.2 ADMIN MODULE:**

This module is for administrator. Admin has full authority of the website. Admin can do all configurations of the website. Admin will manage existing customers and product details.

**1.4.3 PURCHASE MODULE:**

This module allow the customer to purchase the product from distributor and check the product status.

**1.4.4 BILL MODULE:**

In this module bill will be generated after the purchase of the product.

**1.4.5 PAYMENT MODULE:**

In this module customer make payment of the product through online methods.

**1.4.6 FEEDBACK MODULE:**

This module include feedback of the customer about products sold by the distributor.

**1.5 END USERS:**

The end users here are customer. The customer can buy the products in this platform.

**1.6 SOFTWARE/HARDWARE ARE USED FOR THE DEVELOPMENT /IMPLEMENTATION:**

**1.6.1 SOFTWARE REQUIREMENTS:**

Operating System: Windows 10

Front End : HTML, CSS

Back End : Python-Django

**1.6.2 HARDWARE REQUIREMENTS:**

Processor : Intel Core i3

RAM :4GB

Hard Disk :100GB

2.SOFTWARE

REQUIREMENT

SPECIFICATION

**2.1 INTRODUCTION :**

SRS is a complete description of the behaviour of the development. It includes a set of use cases that describe all the interactions the user will have with the software. It deals with the requirement of the purposed system. It describes what the system should do without describing how the system will do it. An SRS provide a reference for the validation of the final product i.e., the SRS helps the client to determine if the system meets the requirements.

**2.2 OVERALL DESCRIPTION:**

This section of the SRS describes all general factor of the products and requirements.

2.2.1 PRODUCT PERSPECTIVE:

The proposed web application contains easy graphical interface to the user. Here the administrator can view the registered customers on this website. It allow customers to buy products online.

2.2.2 PRODUCT FUNCTION:

• The distributor can sell their products through online and customer can purchase those products online.

• Maintains the details about products.

• The user in this website can require login id and password to login to the system

2.2.3 USER CHARACTERISTICS:

This web application is built in such a way that person having little knowledge about the internet can use this site easily. It is basically based on clicking or entering the details.

2.2.4 GENERAL CONSTRAINTS:

• It is necessary to fill all mandatory field with proper information.

• The system requires internet connectivity.

• Only authorized personnel can use the system.

2.2.5 ASSUMPTIONS:

A basic assumption is made about end user that he/she has some knowledge about computer. The project depends on the user’s ability to understand the features of the software application and able to use the best of his/her use.

**2.3 SPECIAL REQUIREMENT:**

**2.3.1 EXTERNAL INTERFACE REQUIREMENT:**

All the interactions of the website with the user, hardware and software are specification here.

**1 USER INTERFACE:**

The user interface is designed in Python-Django. A validation is provided to login the page. On successful validation, the permission to use the website is provided.

**2 HARDWARE INTERFACE:**

The system should have this minimum hardware requirement:

• Processor: Intel core i3

• RAM: 8GB

• Hard disk: 256 GB

3 SOFTWARE INTERFACE:

• Front End: Microsoft Visual Studio 2022

• Back End: Python-Django

4 COMMUNICATION INTERFACE:

• HTML,CSS

**2.4 FUNCTIONAL REQUIREMENTS:**

**2.4.1 Login MODULE:**

This module allows the admin and customer to login to the system by entering valid email and password.

**2.4.2 ADMIN MODULE:**

This module is for administrator. Admin has full authority of the website. Admin can do all configurations of the website. Admin will manage existing customers and product details.

**2.4.3 PURCHASE MODULE:**

This module allow the customer to purchase the product from distributor and check the product status.

**2.4.4 BILL MODULE:**

In this module bill will be generated after the purchase of the product.

**2.4.5 PAYMENT MODULE:**

In this module customer make payment of the product through online methods.

**2.4.6 FEEDBACK MODULE:**

This module include feedback of the customer about products sold by the distributor.

**2.5 DESIGN CONSTRAINTS:**

The system must be designed to allow web usability. the system must be designed in such a way that it can be easy to use and visible on most of the browser.

**2.6 SYSTEM ATTRIBUTES:**

The quality of the websites is such a way that it can be very user friendly to all the user of the website.

• Reliability: We can assure a fully security to the user that we will not provide their personal information to other or any other third parties.

• Maintainability: During maintenance stage, the SRS can be referred for the validation.

• Portability: This system can be run in any operating system and browser.

• Compatibility: This system will be compatible with almost all the web site.

• Flexibility: The system keeps on updating the data according to the changes the changes that takes place

**2.7 OTHER REQUIREMENT:**

**2.7.1 SAFETY REQUIREMENT:**

•In case the user forgets or loses password, the repair functionality helps by chossing “Forgot Password option in the main login window.

•Data will not be shared with others.

• Authorization: Checking for the entity and provide features for them.

**2.7.2 SECURITY REQUIREMENT:**

The proposed website is secured. It means the administrator can modify and delete the data. All other only have the rights to retrieve the information from the website that is concerned to them.

3. SYSTEM DESIGN

**3.1 INTRODUCTION:**

The purpose of the design phase is to plan a solution of the problem specified by the requirements of the document. This is the phase in moving from problem specified by the requirement of the document. The design process is the set of iterative steps that enable the designer to describe all aspect the software to be built. Design is essentially the bridge between the requirement specification and final solution from satisfying the requirement. The design of the system essentially for blueprint or plan for the system.

The design process is often divided into separate phases:

* System Design
* Detailed Design

In system design, the focus is on deciding which modules are needed for the system, the specification of those modules, and how the modules should be interconnected. It is also called top-level design. The system design controls the major structural characteristics of the system. It has a major impact on the testability and modifiability of a system design, and it impacts its efficiency. At the end of the system design all the major data structures, file formats and major modules in the system and their specification are decided.

**3.2 ASSUMPTIONS AND CONSTRAINTS:**

An Assumptions are things that we believe to be true and which we therefore build into the project plan. It is something that we take for granted.

Constraints are things that we know to be true and which must be accounted for in the plan so that we can work around them.

**3.3 FUNCTIONAL DECOMPOSITION:**

The software is decomposed into several modules for the conveniences of the users. Functional decomposition is used to describe a set of steps in which the overall function of a device, system o process are broken into smaller parts. This is used to understand the large and complex process and also helps to solve problems and aids in development of several operations.

**3.3.1 SYSTEM SOFTWARE ARCHITECTURE**

This refers to the fundamental structures of software system and discipline of creating such structures and systems. Each structure comprises software elements, relationship among them and properties if both elements and relations.

**3.3.2 SYSTEM TECHNICAL ARCHITECTURE** It is a form of IT architecture that is used to design computer systems. It involves the development of technical blueprint with regards to the arrangement, interaction and independence of all elements so that system relevant requirements are met.

**3.3.3 SYSTEM HARDWARE ARCHITECTURE**

This is primarily concerned with internal electrical interfaces among the systems component or subsystems and the interfaces between the system and its external environment.

**3.3.4 EXTERNAL INTERFACES**

External interfaces are typically a products lifeline to the outside world. Such interfaces may be used for the number of purposes including connecting to peripherals, field programming or testing during product manufacturing.

**3.4 DESCRIPTION OF PROGRAMS**

There are 4 kinds of system components:

1.Process

2.External entity

3.Data flow

4.Data stores

|  |  |  |
| --- | --- | --- |
| SYMBOLS | NAME | DESCRIPTION |
|  | Process | The circle is used to distinguish between data input and comes from a process. It represents functions or process. |
|  | Entity | The rectangle represents a simple source or termination of the diagram by mapping real world entities. |
|  | Data flow | The arrow represents the bidirectional lines and depicts the flow of data from one location to another. |
|  | Data storage | The parallel lines represented here displays the location where the data is stored in the system. |

**3.4.1 CONTEXT FLOW DIAGRAM**

A context diagram, sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. The entire software system is shown as a single process.

USER

ADMIN

**3.4.2 DATA FLOW DIAGRAM**

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

ADMIN LEVEL DFD:

login

Order

Report

Item

Status

Customer

Status

Logout

Reports

Database

V

iew Orders

and update status

V

iew item details

Edit/add new item

V

iew customer details

V

iew transaction reports

Valid

Invalid

Username

Password

Admin

USER LEVEL DFD:

Retailers

Customer

Login

Password

Username

Invalid

Order

Product

Place

Order

Orders

Logout

Database

V

iew order product

Update stock

Placing order

Order history

**3.5 DESCRIPTION OF COMPONENTS**

* **Login module**

Input : User name, password.

Process: Admin details will be stored.

Output: Allows viewing the main page.

* **User module**

Input : Email, password.

Process: User details will be stored.

Output: Allows to order the products.

4.DATABASE DESIGN

**4.1 INTRODUCTION:**

A database is a collection of interrelated data stored with minimum redundancy to serve many and efficiently. The general objective of database design is to make the data easy. pensive and flexible to the user. Table list of information organized into fields.

**4.2 PURPOSE AND SCOPE:**

A good database design is, therefore, one that Divides your information into subject based ales to reduce redundant data. Provides Access with the information it requires to join the information in the tables together as needed. Helps support and ensure the accuracy and integrity of your information. Information about the specific subject range, format, or de range a particular specialized database covers is called its scope. A specialized database may be or broad in scope, depending on whether it, for instance, contains materials on one of many subject areas:

**4.3 DATABASE IDENTIFICATION:**

The database object name is referred to as its identifier. Everything in Microsoft SQL Server can have an identifier. Servers, databases and database objects, such tables, views, columns, indexes, triggers, procedures, constraints, and rules can have identifiers.

**4.4 SCHEMA INFORMATION:**

Information schema is an ANSI-standard set of read-only views 9/16 information about all of tables, views, columns, and procedures i provides access to database metadata, information about de MySQL server name of a database or table, the data type of a column, or access privileges. Other terms that are sometimes used for this information are data dictionary and system catalog.

**4.5 TABLE DEFINITION:**

Tables are database objects that contain all the data in a database. In tables, data is logically organized in a row-and-column format similar to a spreadsheet. Each now represents a unique record, d each column represents a field in the record.

**Brand List:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Constraints | Description |
| Id | int(30) | Primary key | Stores id of the brand |
| Name | text | Not null | Stores name of brand |
| image\_path | text | Not null | Stores image path |
| delete\_flag | tintint(1) | Not null | Used to delete brands |
| Status | tinyint(1) | Not null | To check status of brand |
| date\_created | datetime | Not null | Stores brand created date |
| date­\_updated | datetime | null | Stores updated date |

**Cart list:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Constraints | Description |
| id | int(30) | Primary key | Stores id of the cart products |
| client\_id | int(30) | Not null | Stores id of the client |
| product\_id | int(30) | Not null | Stores id of the product |
| quantity | Float | Not null | Stores quantity of the product |
| date\_added | datetime | Not null | stores product added date |

**Categories:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Constraints | Description |
| id | int(30) | Primary key | Stores id of category |
| category | varchar(250) | Not null | Stored type of category |
| status | tinyint(1) | Not null | Stores status of the category |
| delete\_flag | tinyint(1) | Not null | used to dalete |
| date\_created | datetime | Not null | stored categories created date |

**Client list:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Constraints | Description |
| id | int(30) | Primary key | Stores id of client |
| firstname | text | Not null | Stores the first name of client |
| middlename | text | Not null | Stores the middle name of client |
| lastname | text | Not null | Stores the lastname of client |
| gender | text | Not null | Stores the gender |
| contact | text | Not null | Stores the contact |
| address | text | Not null | Stores the address |
| email | text | Not null | Stores the email of client |
| password | text | Not null | Stores the password of client |
| status | tinyint(1) | Not null | Stores the status of client |
| delete\_flag | tinyint(1) | Not null | Used to delete |
| date\_create | datetime | Not null | Stores the date when client login |
| date\_added | datetime | Not null | Stores client added date |

**Mechanic list:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Constraints | Description |
| id | int(30) | Primary key | Stores the id of mechanic |
| name | text | Not null | Stores the mechanic name |
| contact | varchar(50) | Not null | Stores the contact number |
| email | varchar(200) | Not null | Stores the email of mechanic |
| status | tinyint(1) | Not null | Stores the status of mechanic |
| date\_create | datetime | Not null | Stores the mechanic added date |

**Order items:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Constraints | Description |
| id | int(30) | Primary key | Stores the id of item |
| order\_id | int(30) | Foreign key | Stores the od |
| product\_id | int(30) | Foreign key | Stores the product id |
| quantity | float | Not null | Stores the quantity |
| date\_added | datetime | Not null | Stores order added date |

**Order list:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Constraints | Description |
| id | int(30) | Primary key | Stores the order id |
| ref\_code | varchar(100) | Not null | Stores the reference id |
| client\_id | int(30) | Foreign key | Stores the client id |
| total\_amount | float | Not null | Stores the total amount |
| delivery\_address | text | Not null | Stores the delivery address |
| status | tinyint(1) | Not null | Stores the status of order |
| date\_created | datetime | Not null | Stores the order added date |
| date\_updated | datetime | Not null | Stores order updated date |

**Product list:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Constraints | Description |
| id | int(30) | Primary key | Stores the product id |
| brand\_id | int(30) | Foreign key | Stores brand id |
| category\_id | int(30) | Foreign key | Stores category id |
| name | text | Not null | Stores the name |
| models | text | Not null | Stores the models |
| descripton | text | Not null | Stores the description of product |
| price | float | Not null | Stores the price |
| status | tinyint(1) | Not null | Stores the status of product |
| image\_path | text | Not null | Stores the image |
| delete\_flag | tinyint(1) | Not null | Used to deleted |
| date\_created | datetime | Not null | Stores the product added date |
| date\_updated | datetime | Not null | Stores product updated date |

**Request meta:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Constraints | Description |
| request\_id | int(30) | Foreign key | Stores the request id |
| meta\_field | text | Not null | Stores the meta detail |
| meta\_value | text | Not null | Stores the meta value |

**Service list:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Constraints | Description |
| id | int(30) | Primary key | Stores the id of service |
| service | text | Not null | Stores the service |
| description | text | Not null | Stores the description |
| status | tinyint(4) | Not null | Stores the status |
| delete\_flag | tinyint(1) | Not null | Used to delete |
| date\_created | datetime | Not null | Stores the service added date |

**Service request:**

|  |  |  |  |
| --- | --- | --- | --- |
| Filed name | Data type | Constraints | Description |
| id | int(30) | Primary key | Stores the id of service request |
| client\_id | int(30) | Not null | Stores the client of service request |
| service\_type | text | Not null | Stores the service type |
| mechanic\_id | int(30) | Not null | Stores the mechanic id |
| status | tinyint(1) | Not null | Stores the status |
| date\_created | datetime | Not null | Stores the service added date |

**Stock list:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Constraints | Description |
| id | int(30) | Primary key | Stores the id of stock |
| product\_id | int(30) | Foreign key | Stores the product id |
| quantity | float | Not null | Stores the quantity |
| type | tinyint(1) | Not null | Stores the type of stock |
| date\_created | datetime | Not null | Stores the stock created date |

**4.6 PHYSICAL DATA:**

The physical data model is a representation of a data design as implemented, or intended to be implemented, in a database management system. In the lifecycle of a project it typically derives from a logical data model, though it may be reverse-engineered from a given database implementation.

**4.7 DATA DICTIONARY:**

A Data Dictionary is a collection of names, definitions, and attributes about data elements that are being used or captured in a database, information system, or part of a research project A data dictionary is a centralized repository of metadata.

**4.8 ENTITY RELATIONSHIP DIAGRAM(ER):**

The basic object that the ER model represents in an entity which is a thing in a real world an pendent existence" An Entity is an object with a physical existence of with a conceptual existence Each entity has attributes particular properties that describes it Whenever an attribute of entity type refers to another entity sons relationships exists. The degree of relationship type is the number of participate in entity type. The cardinality ratio for binary relationship specifics a number of relationship exist like that an entities.

**Entity Relationship Diagram (ER):**

|  |  |
| --- | --- |
| Symbol | Description |
|  | Entity |
|  | Attribute |
|  | Key Attribute |
|  | Relationship |
|  | Weak Entity |
|  | Card Ratio |

An Entity Relationship Model (ERM), in software engineering is an abstract And conceptual representation of data. Entity relationship schema database modelling method, used to produce a type conceptual schema or schematic data model of a system, often relational database, and it is represented in a top-down fashion.

**4.8.1 Entity:**

Entity is the thing which we want to store the information. It is an elementary basic building block of storing information about business process. An entity represents an object defined within the information system about which we want to store information. Entities are distinct things in the enterprise.

**4.8.2 Relationship:**

A Relationship is a named collection or association between entities or used to relate Two or more entities with some common attributes or meaningful interaction between the objects.

**4.8.3 Attributes:**

Attributes are the properties of the entities and relationship, descriptor of the entity Attributes are elementary pieces of inform attached to an entity.

**4.8.4 Cardinality:**

Specifies how many instances of an entity relate to one instance of another entity attribute are elementary pieces of information attached to an entity.

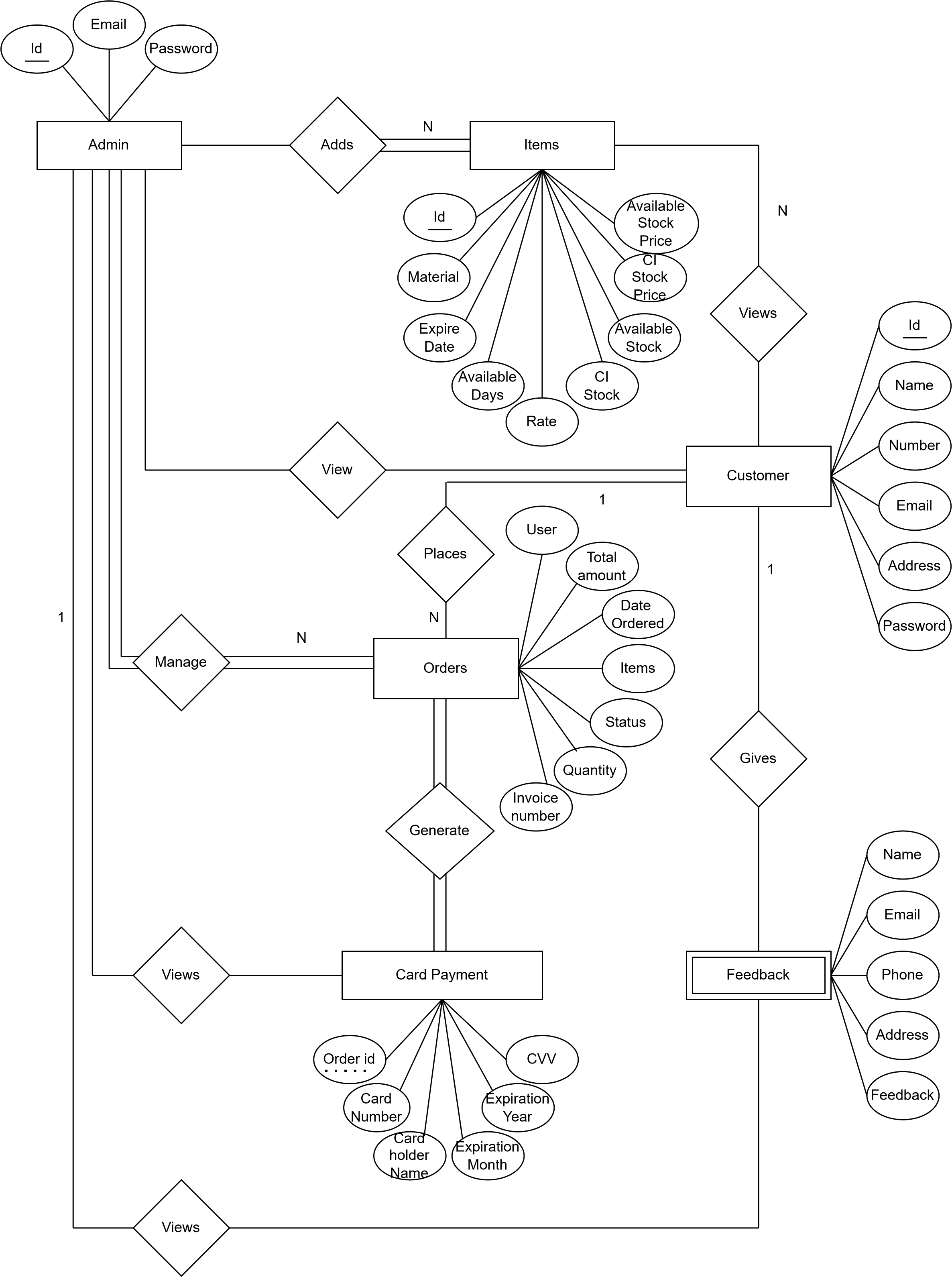
**4.8.5 Weak Entity:**

A weak entity does not have a primary key attribute and depends on other entity via a foreign key attribute.

**4.8.6 Primary Key:**

A primary key is either an existing table column or a column that is specifically generated by the database according to a defined seque

**Entity Relationship Diagram:**



**4.9 DATABASE ADMINISTRATION:**

Database Administrator (DBA) in Database Management System (DBMS) is an IT who works on creating, maintaining. querying, and tuning the database of the organisation. They are also responsible for maintaining data security and integrity.

**4.9.1 System information:**

A database management system (or DBMS) is essentially nothing more than a computerized data-keeping system. Users of the system are given facilities to perform several kinds of operations on such a system for either manipulation of the data in the database or the management of the database structure itself.

**4.9.2 DBMS Configuration:**

DBMS Server Configuration. The Ingres client server architecture allows multiple users access to databases through connections to one or more DBMS Server processes. The DBMS Server (iidbms) is a multi-threaded daemon process that performs asynchronous disk input and output.

**4.9.3 Support software required:**

**1. Solar Wind Database Performance Analyzer:**

Solar Wind Database Performance Analyzer pinpoints performance issues while providing expert advice with tuning advisor.

**2. Db Visualizer:**

Db Visualizer is a database management solution that helps businesses monitor and analyse the performance of multiple databases through various operating systems including Windows, macOS and Linux.

**3. Manage Engine Application Manager:**

Manage Engine Applications Manager is an enterprise-ready, easy-to use, and affordable application management solution that helps enterprises monitor their mission- critical applications and data canters effectively.

**4. Oracle RDBMS:**

Oracle Database is an RDBMS. An RDBMS that implements object-oriented features such as user-defined types, inheritance, and polymorphism is called an object relational database management system (ORDBMS).

**4.9.4 Storage Requirement:**

Primary Storage: The memory storage that is directly accessible to the CPU comes under this category backup.

Secondary Storage: Secondary storage devices are used to store data for future use or as backup

Tertiary Storage: Tertiary storage is used to store huge volumes of data. Since such storage devices are external to the computer system, they are slowest in speed. These storage devices are mostly used to take the backup of an entire system. Optical disks and magnetic tapes are widely used as teritary storage.

**4.9.5 Backup and recovery:**

It enables the creation of a duplicate instance or copy of a database in case the primary database crashes, is corrupted or is lost. Recovery is the process of restoring the database to a correct (consistent) state in the event of a failure. In other words, it is the process of restoring the database to the most recent consistent state that existed shortly before the time of system failure.

5.DETAILED DESIGN

**5.1 INTRODUCTION**

The detail design phase involves completing the product’s design. The design team works toward completion of the specifications for the product and its subassemblies, product elements, and manufacturing processes. Like the other phases of product development, detail design is an iterative process.

**5.2 STRUCTURE OF THE SOFTWARE PACKAGE**

A structure package bundles a set of packages at the top level of the package hierarchy. Structure packages do not contain any repository objects except for its own package interfaces and subpackages

**5.3 MODULAR DECOMPOSITION AND COMPONENTS**

* **LOGIN MODULE**

Input : Email, password.

Process: The admin and customer are provide with login form where they can update

email id and password, if it is correct then it goes to main page otherwise it

display error message.

Output: Allows to order the products.

* **ADMIN MODULE**

Input : Admin will add product details.

Process: Added product details are stored in database and it can be edited, updated or

deleted.

Output: Added product details are updated with any information and information can

viewed.

* **EMPLOYEE MODULE**

Input : Admin will adds the Employee details

Process: Employee details will be stored.

Output:

* **SERVICE MODULE**

Input : Admin can add the services.

Process: Service details will be stored.

Output: Allows viewing in the main page.

* **SALE MODULE**

Input : Sale of the products

Process: Sale details will be stored.

Output: Admin can view.

* **CATEGORY MODULE`**

Input : Admin can store the product category.

Process: Product details will be stored.

Output: Admin and user can view.

* **BILLING MODULE**

Input : Admin can generate bill for the services.

Process: Billing details will be stored.

Output: Allows viewing to the admin.

6.PROGRAM CODE LISTING

7.USER INTERFACE