VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"JNANA SANGAMA", BELAGAVI - 590 018

A MINI PROJECT REPORT

on

"CASH AND CARRY MANAGEMENT SYSTEM"

Submitted by

Swathi S Nayak	4SF19IS115
Vishnu N V	4SF19IS120

In partial fulfillment of the requirements for the V semester

DBMS LABORATORY WITH MINI PROJECT

of

BACHELOR OF ENGINEERING

in

INFORMATION SCIENCE & ENGINEERING

Under the Guidance of

Ms. Jayapadmini Kanchan

Assistant Professor, Department of ISE

at

SAHYADRI

College of Engineering & Management
Adyar, Mangaluru - 575 007
2021 - 22

SAHYADRI

College of Engineering & Management

Adyar, Mangaluru - 575 007

Department of Information Science & Engineering

CERTIFICATE

This is to certify that the **Mini Project** entitled **“Cash and Carry Management System”** has been carried out by **Swathi S Nayak (4SF19IS115)** and **Vishnu N V (4SF19IS120)**, the bonafide students of Sahyadri College of Engineering & Management in partial fulfillment of the requirements for the V semester **DBMS Laboratory with Mini Project (18CSL58)** of **Bachelor of Engineering in Information Science & Engineering** of Visvesvaraya Technological University, Belagavi during the year 2021 - 22. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work.

Ms. Jayapadmini Kanchan
Assistant Professor
Dept. of ISE, SCEM

Mr. Ganaraj K
Assistant Professor
Dept. of ISE, SCEM

Dr. Shamanth Rai
HOD & Associate Professor
Dept. of ISE, SCEM

External Practical Examination:

Examiner’s Name Signature with Date

1. ....

2. ....

SAHYADRI

College of Engineering & Management

Adyar, Mangaluru - 575 007

Department of Information Science & Engineering

DECLARATION

We hereby declare that the entire work embodied in this Mini Project Report titled **"Cash and Carry Management System"** has been carried out by us at Sahyadri College of Engineering and Management, Mangaluru under the supervision of **Ms. Jayapadmini Kanchan** as the part of the V semester **DBMS Laboratory with Mini Project (18CSL58)** of **Bachelor of Engineering** in **Information Science & Engineering**. This report has not been submitted to this or any other University.

Swathi S Nayak (4SF19IS115)
Vishnu N V (4SF19IS120)
SCEM, Mangaluru

Abstract

This application is used to store and validate the stock information from individual stores in a supermarket. At regular intervals of time, the database is updated with accurate stock purchases. By studying the stock being sold by a set of shops, the dependency factor of each manufacturer can be estimated. This gives the supermarket a measure of how much they've profited from a certain manufacturer. The database provides a way to calculate the revenue generated by each shop and the profit made through customers. The end product has the potential to assist the supermarket on a substantial scale. The purpose of this project is to display details of purchases made from each shop together with the ability to add, delete, and modify the quantities of stock each time a purchase is made. In the aspect of software, this project uses Python language and SQLAlchemy as the background database. Various configurations in computer including input and output capacity, internal memory and external memory capacity are met the requirements of users.

Acknowledgement

It is with great satisfaction and euphoria that we are submitting the Mini Project Report on **"Cash and Carry Management System"**. We have completed it as a part of the V semester **DBMS Laboratory with Mini Project (18CSL58)** of **Bachelor of Engineering** in **Information Science & Engineering** of Visvesvaraya Technological University, Belagavi.

We are profoundly indebted to our guide, **Ms. Jayapadmini Kanchan**, Assistant Professor, Department of Information Science & Engineering for innumerable acts of timely advice, encouragement and We sincerely express our gratitude.

We express our sincere gratitude to **Dr. Shamanth Rai**, Head & Associate Professor, Department of Information Science & Engineering for his invaluable support and guidance.

We sincerely thank **Dr. Rajesha S**, Principal, Sahyadri College of Engineering & Management and **Dr. D. L. Prabhakara**, Director, Sahyadri Educational Institutions,who have always been a great source of inspiration.

Finally, yet importantly, We express our heartfelt thanks to our family & friends for their wishes and encouragement throughout the work.

Swathi S Nayak
4SF19IS115
V Sem, B.E., ISE
SCEM, Mangaluru

Vishnu N V
4SF19IS120
V Sem, B.E., ISE
SCEM, Mangaluru

Table of Contents

List of Figures

Chapter 1

Introduction

This application is to store and validate stock information from individual stores in a supermarket. At regular intervals of time, the database is updated with accurate stock purchases. By studying the stock being sold by a set of shops, the dependency factor of each manufacturer can be estimated. This gives the supermarket a measure of how much they've profited from a particular manufacturer. The database provides a way to calculate the revenue generated by each shop and the profit made through customers. The end product has the potential to assist the supermarket on a substantial scale.

1.1 Purpose

This project is concerned with managing the activities that occur in a supermarket. This allows us to check the profitability of each shop from a particular manufacturer within the supermarket. By doing so, the supermarket can determine how to expand its geographical reach to new people. This serves as an advantage in maintaining accurate stock information whenever a purchase is made. It minimizes the need to calculate sales activities. This facilitates the storage of large amounts of data in the database, reducing clumsiness.

1.2 Scope

This project is aimed at providing supermarkets with an efficient cash management system. As a result, the current stock information, employee details, newly added items, transactions, and reports can be viewed. The research work deals with stock control and seeks to correct anomalies in the supermarket industry. As a result, it analyzes the opening and updating of newly issued stocks, as well as the viewing of existing stocks. By eliminating manual procedures and automating them, it provides a quick and efficient way to operate. The project assists in computerizing the item transaction, keeping track of sales activity, and maintaining stock levels.

1.3 Overview

As technology has evolved a lot in today's world, people are also upgrading themselves. The objective of this project is to store as much data as possible in a database so that it can be accessed whenever needed, unlike paper work which is considered to be a tedious process to store information on large amounts of stock. This displays details of purchases made from each shop. This allows adding, deleting, and modifying the quantities of stock each time a purchase is made.

Chapter 2

Hardware and Software Details

2.1 Hardware Details

- Processor : Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz
- RAM : 8GB
- Hard Disk : 1TB
- Input Device : Standard Keyboard and Mouse
- Output Device : Monitor

2.2 Software Details

- Database : MySQL
- Programming Language : Python
- IDE : Visual Studio Code
- Operating System : Microsoft Windows 10 and above

Chapter 3

System Design

3.1 ER Diagram

The project starts by creating a Employee entity with attributes like E_id, Eame, Eaddress, Ephone and then Customer, Stock, Shop, Manufacturer and Branch entities are created. The Employee, Customers, Shop, Stock, Manufacturer are the strong entity types and Branch as a weak entity.The supermarket has customer who visits the shop. The role of an employee is to do the entries of the products the customer wants to purchase. Each shop has their products which will be ordered from the manufacturers associated with the shop.

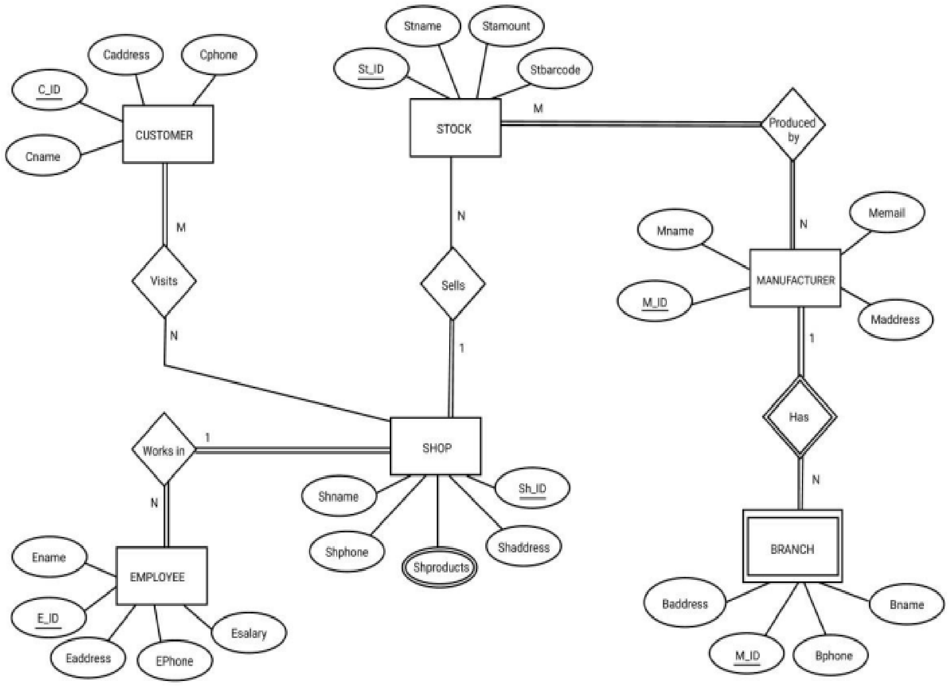


Figure 3.1: ER Diagram for Cash and Carry Management System

3.2 Mapping From ER Diagram to Schema Diagram

The schema diagram for mapping is illustarted as shown in the Figure 3.2. **1.Mapping of Regular Entities**:This step involves mapping all the regular entity types to tabular format by identifying their primary keys.

2.Mapping of 1:1 Relation: In this step foreign keys are assigned using foreign key approach.The primary key of the participating relation R or S is added as primary key to second entity types by looking at the participating constraints.

3.Mapping of 1:N Relation: Foreign key approach is used to add one sided primary key to the n sided entity at foreign key.

4.Mapping of M:N Relation: Here we use the cross reference approach where the relationship is converted to a new relation within attributes on primary keys of both participating relation.

5.Mapping of Weak Entity: When mapping weak entity types along with other attributes the partial key and primary key of parent entity together will form their primary key of the new relation.

6.Mapping of N-ary Relation: For mapping N array relationship we create a new relation with a relationship name in its attribute and primary keys of all participating entity types.

7.Mapping of Multivalued Relation: For multivalued attributes a separate relation has to be created along with primary key of parent relation.

In our database we have the following mappings:

Step – 1 : Mapping of Regular Entities.

From the ER diagram we identify all the strong entities E and create a relation R that includes all it's simple attributes and primary keys.

The following are the strong entities from our schema diagram :

1.CUSTOMER(C_ID, Cname, Caddress, Cphone)

2.EMPLOYEE(E_ID, Ename, Eaddress, EPhone, Esalary)

3.MANUFACTURER(M_ID, Mname, Maddress, Memail)

4.SHOP(Sh_ID, Shname, Shproducts, Shaddress, Shphone)

5.STOCKS(St_ID, Stname, Stamount, Stbarcode)

Step – 2 : Mapping of binary 1:1 Relation Types.

In relational database design, a one-to-one (1:1) relationship exists when zero or one instance of entity A can be associated with zero or one instance of entity B, and zero or one instance of entity B can be associated with zero or one instance of entity A.
Unfortunately, we don't have any 1:1 relation existing within our database design.

Step – 3 : Mapping of binary 1:N Relation Types.

The SHOP and the STOCK entities are participating in the 1:N relation type. Since STOCK is on the nth side of the relation we include the primary key of SHOP entity as the Foreign key in STOCK entity.

The MANUFACTURER and the BRANCH entities are participating in the 1:N relation type. Since BRANCH is on the nth side of the relation we include the primary key of MANUFACTURER entity as the Foreign key in BRANCH entity.

The EMPLOYEE and the SHOP entities are participating in the 1:N relation type. Since EMPLOYEE is on the nth side of the relation we include the primary key of EMPLOYEE entity as the Foreign key in SHOP entity.

Step – 4 : Mapping of binary M:N Relation Types.

The relationship between the CUSTOMERS and the SHOP is M:N .So we create a new relation VISITS which includes the primary key of CUSTOMERS and SHOP entity. The combination of the two primary keys will form the primary key of the VISITS relation.

The relationship between the STOCKS and the MANUFACTURERS is M:N. So we create a new relation HAS which includes the primary key of STOCKS and MANUFACTURERS entity. The combination of the two primary keys will form the primary key of the HAS relation.

Step – 5 : Mapping of Multivalued Relation Types.

The relationship between the SHOP has Shproducts as its Multivalued attribute .So we create a new relation SHPRODUCTS which includes the primary key of SHOP entity and the multivalued attribute Shproducts.

SHPRODUCTS(Sh_ID, Shproducts)

Step – 6 : Mapping of Weak Entity.

From the ER diagram we identify all the weak entities E and create a relation R that includes all it's simple attributes and partial keys.

The following are the weak entities from our schema diagram :

BRANCH(M_ID, Bname, Baddress, Bphone)

3.3 Assumptions

- There are multiple shops in the supermarket where one shop can many employees to work in them. As there are many employees who work in a shop, only one employee can work in one shop.
- Multiple stocks that are produced can be sold from a particular shop in the supermarket.
- There are many stocks that are produced by a manufacturers. One stock can be manufactured by many manufcaturers.
- There are many manufacturers who work under diffeerent branch. One manufacturer can work under many branches.
- There are customers who visits to shops for purchase. Many customers can visit many shops and one shop can be visited by many customers.

3.4 Schema Diagram

A schema is a pictorial representation of the relationship between the database tables in the database that is created. The database schema of a database system is its structure described in a formal language sup ported by the database management system (DBMS). The term ”schema” refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases).

The formal definition of a database schema is a set of formulas (sentences) called integrity constraints imposed on a database. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language.The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real-world entities are modelled in the database.

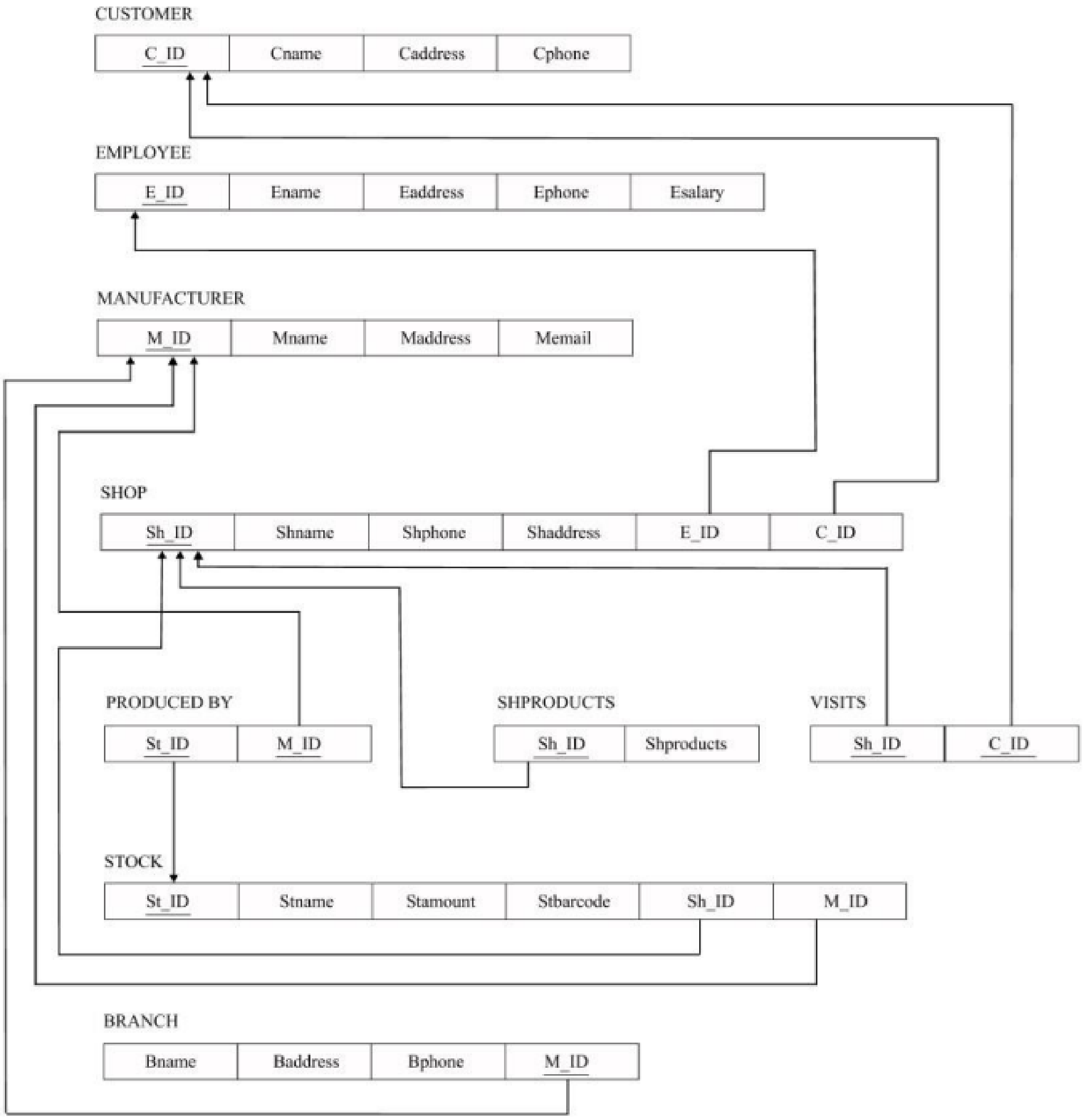


Figure 3.2: Schema Diagram for Cash and Carry Management System

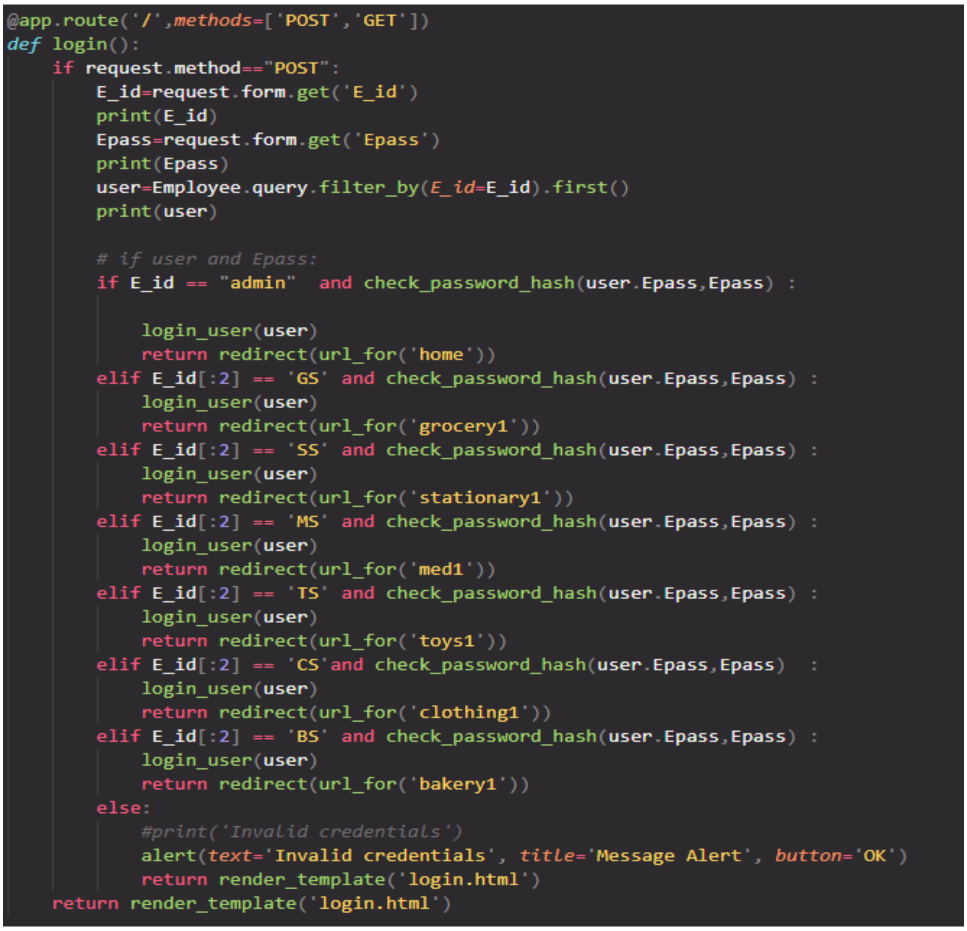
Chapter 4

Implementation

4.1 Pseudo-Codes

Pseudo-code for Login Operation

An employee is not allowed to access a store until he has logged in. When the employee logins in with the valid credentials, it will direct him to a particular store.

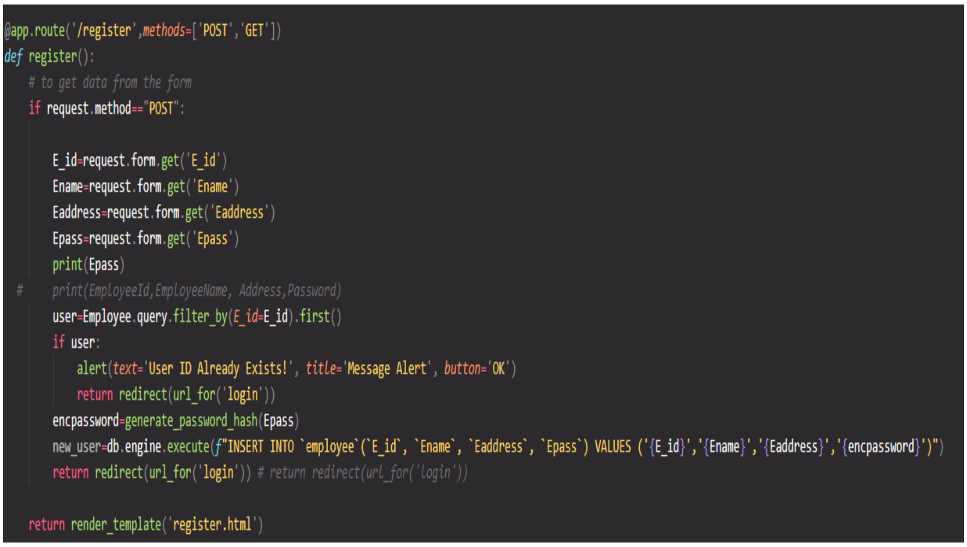


```
@app.route('/',methods=['POST','GET'])
def login():
    if request.method=="POST":
        E_id=request.form.get('E_id')
        print(E_id)
        Epass=request.form.get('Epass')
        print(Epass)
        user=Employee.query.filter_by(E_id=E_id).first()
        print(user)

        # if user and Epass:
        if E_id == "admin"  and check_password_hash(user.Epass,Epass) :

            login_user(user)
            return redirect(url_for('home'))
        elif E_id[:2] == 'GS' and check_password_hash(user.Epass,Epass) :
            login_user(user)
            return redirect(url_for('grocery1'))
        elif E_id[:2] == 'SS' and check_password_hash(user.Epass,Epass) :
            login_user(user)
            return redirect(url_for('stationary1'))
        elif E_id[:2] == 'MS' and check_password_hash(user.Epass,Epass) :
            login_user(user)
            return redirect(url_for('med1'))
        elif E_id[:2] == 'TS' and check_password_hash(user.Epass,Epass) :
            login_user(user)
            return redirect(url_for('toys1'))
        elif E_id[:2] == 'CS'and check_password_hash(user.Epass,Epass)  :
            login_user(user)
            return redirect(url_for('clothing1'))
        elif E_id[:2] == 'BS' and check_password_hash(user.Epass,Epass) :
            login_user(user)
            return redirect(url_for('bakery1'))
        else:
            #print('Invalid credentials')
            alert(text='Invalid credentials', title='Message Alert', button='OK')
            return render_template('login.html')
    return render_template('login.html')
```

Figure 4.1: Pseudo-code for Login Operation

Pseudo-code for Inserting values into the database:

If an employee is not registered for a store, he has to register first and then only he can login. Figure 4.2 illustrates the Register page for insertion operation.



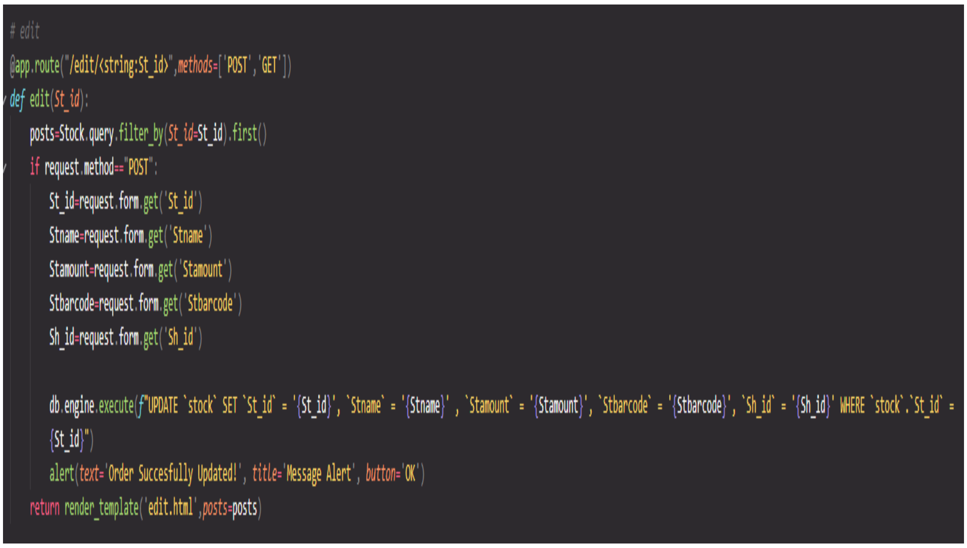
```
@app.route('/register',methods=['POST','GET'])
def register():
    # to get data from the form
    if request.method=="POST":

        E_id=request.form.get('E_id')
        Ename=request.form.get('Ename')
        Eaddress=request.form.get('Eaddress')
        Epass=request.form.get('Epass')
        print(Epass)
#       print(EmployeeId,EmployeeName, Address,Password)
        user=Employee.query.filter_by(E_id=E_id).first()
        if user:
            alert(text='User ID Already Exists!', title='Message Alert', button='OK')
            return redirect(url_for('login'))
        encpassword=generate_password_hash(Epass)
        new_user=db.engine.execute(f"INSERT INTO `employee`(`E_id`, `Ename`, `Eaddress`, `Epass`) VALUES ('{E_id}','{Ename}','{Eaddress}','{encpassword}')")
        return redirect(url_for('login')) # return redirect(url_for('login'))

    return render_template('register.html')
```

Figure 4.2: Inserting values into the database

Pseudo-code for Updating of values into the database

An employee must be logged in to update the stock details. The changes can be made accordingly which will be saved in the database for further operations.

Figure 4.3 illustrates the Updating of Stocks .



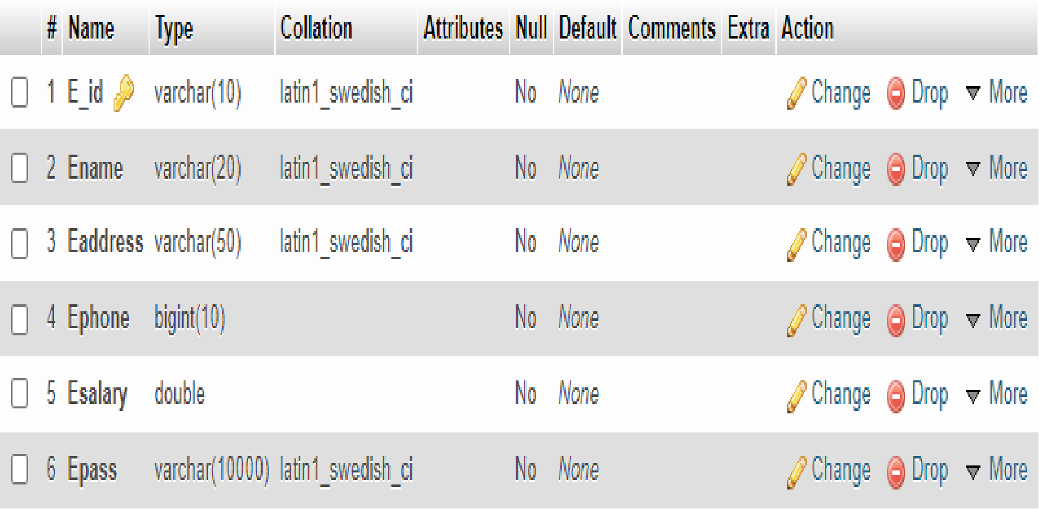
```
# edit
@app.route("/edit/<string:St_id>",methods=['POST','GET'])
def edit(St_id):
    posts=Stock.query.filter_by(St_id=St_id).first()
    if request.method=="POST":
        St_id=request.form.get('St_id')
        Stname=request.form.get('Stname')
        Stamount=request.form.get('Stamount')
        Stbarcode=request.form.get('Stbarcode')
        Sh_id=request.form.get('Sh_id')

        db.engine.execute(f"UPDATE `stock` SET `St_id` = '{St_id}', `Stname` = '{Stname}' , `Stamount` = '{Stamount}', `Stbarcode` = '{Stbarcode}', `Sh_id` = '{Sh_id}' WHERE `stock`.`St_id` =
        {St_id}")
        alert(text='Order Succesfully Updated!', title='Message Alert', button='OK')
    return render_template('edit.html',posts=posts)
```

Figure 4.3: Pseudo-code for Updating of values into the database

4.2 Tables used

Employee Table :

The Employee table contains the attributes E_id, Ename, Eaddress, Ephone, Esalary and Epass. Here, the attribute E_id is the primary key.

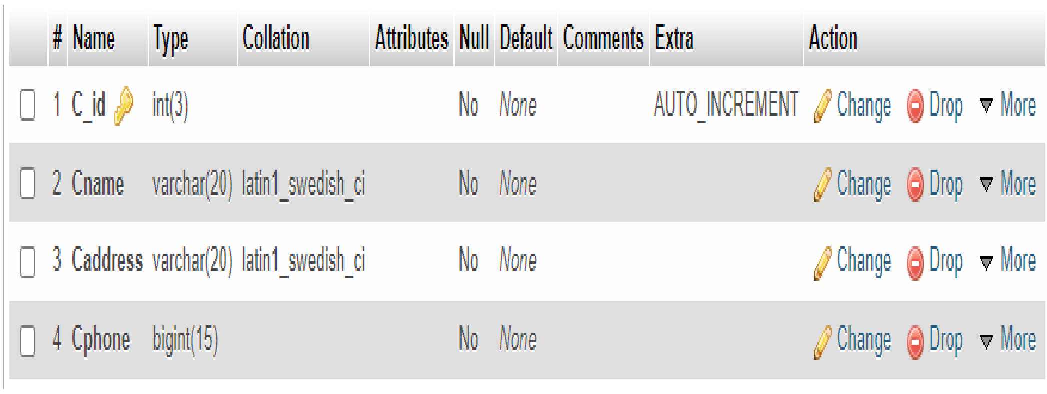


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	E_id 🔑	varchar(10)	latin1_swedish_ci		No	*None*			Change Drop ▼ More
□	2	Ename	varchar(20)	latin1_swedish_ci		No	*None*			Change Drop ▼ More
□	3	Eaddress	varchar(50)	latin1_swedish_ci		No	*None*			Change Drop ▼ More
□	4	Ephone	bigint(10)			No	*None*			Change Drop ▼ More
□	5	Esalary	double			No	*None*			Change Drop ▼ More
□	6	Epass	varchar(10000)	latin1_swedish_ci		No	*None*			Change Drop ▼ More

Figure 4.4: Structure of Employee table

Customer Table :

The Customer table contains the attributes C_id, Cname, Caddress, Cphone. Here, the attribute C_id is the primary key.

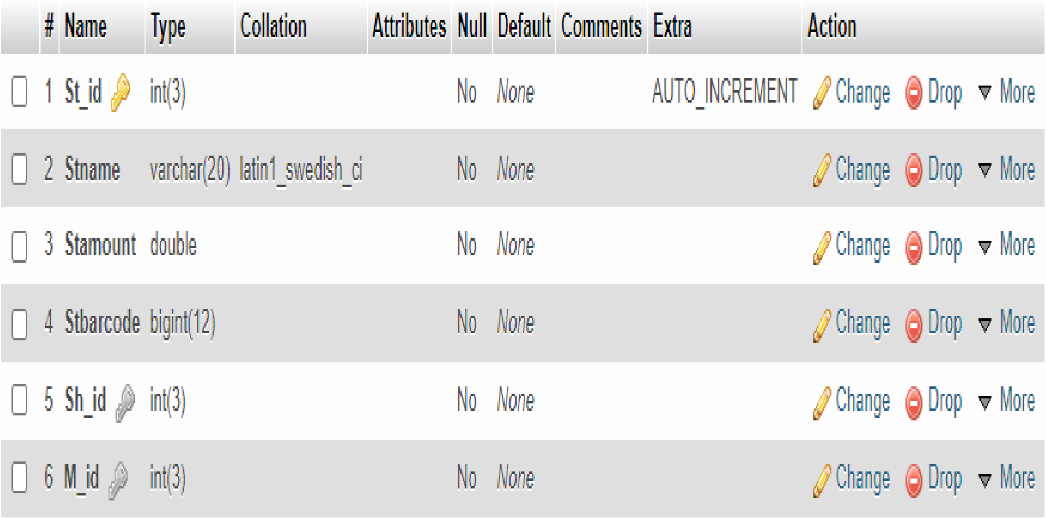


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	C_id	int(3)			No	*None*		AUTO_INCREMENT	Change Drop ▼ More
□	2	Cname	varchar(20)	latin1_swedish_ci		No	*None*			Change Drop ▼ More
□	3	Caddress	varchar(20)	latin1_swedish_ci		No	*None*			Change Drop ▼ More
□	4	Cphone	bigint(15)			No	*None*			Change Drop ▼ More

Figure 4.5: Structure of Employee table

Stock Table :

The Stock table contains the attributes St_id, Stname, Stamount, Stbarcode, Sh_id, M_id. Here, the attribute St_id is the primary key and Sh_id and M_id are the foreign keys.

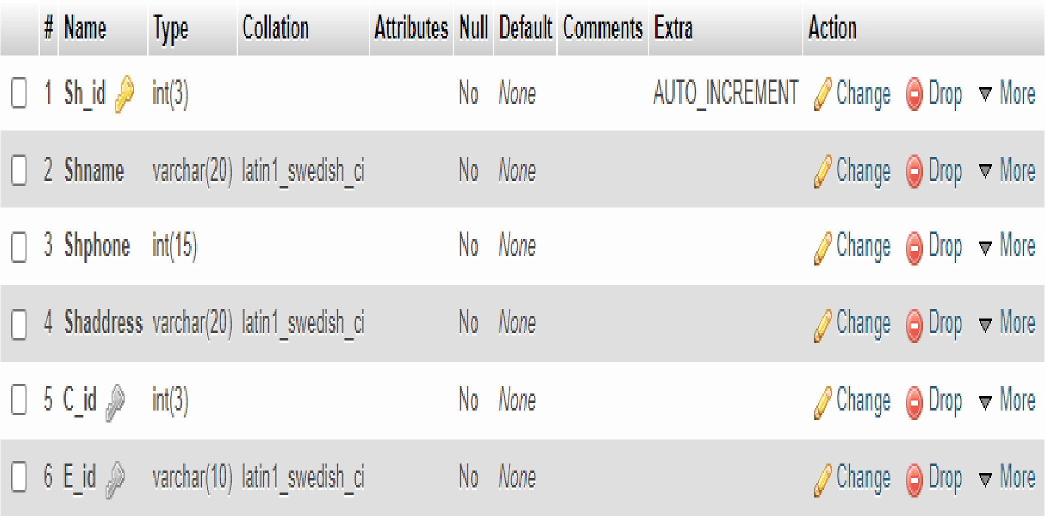


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	St_id	int(3)			No	*None*		AUTO_INCREMENT	Change Drop ▼ More
□	2	Stname	varchar(20)	latin1_swedish_ci		No	*None*			Change Drop ▼ More
□	3	Stamount	double			No	*None*			Change Drop ▼ More
□	4	Stbarcode	bigint(12)			No	*None*			Change Drop ▼ More
□	5	Sh_id	int(3)			No	*None*			Change Drop ▼ More
□	6	M_id	int(3)			No	*None*			Change Drop ▼ More

Figure 4.6: Structure of Stock table

Shop Table :

The Shop table contains the attributes Sh_id, Shname, Shphone, Shaddress, C_id, E_id. Here, the attribute Sh_id is the primary key and C_id and E_id are the foreign keys.



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	Sh_id	int(3)			No	*None*		AUTO_INCREMENT	Change Drop ▾ More
□	2	Shname	varchar(20)	latin1_swedish_ci		No	*None*			Change Drop ▾ More
□	3	Shphone	int(15)			No	*None*			Change Drop ▾ More
□	4	Shaddress	varchar(20)	latin1_swedish_ci		No	*None*			Change Drop ▾ More
□	5	C_id	int(3)			No	*None*			Change Drop ▾ More
□	6	E_id	varchar(10)	latin1_swedish_ci		No	*None*			Change Drop ▾ More

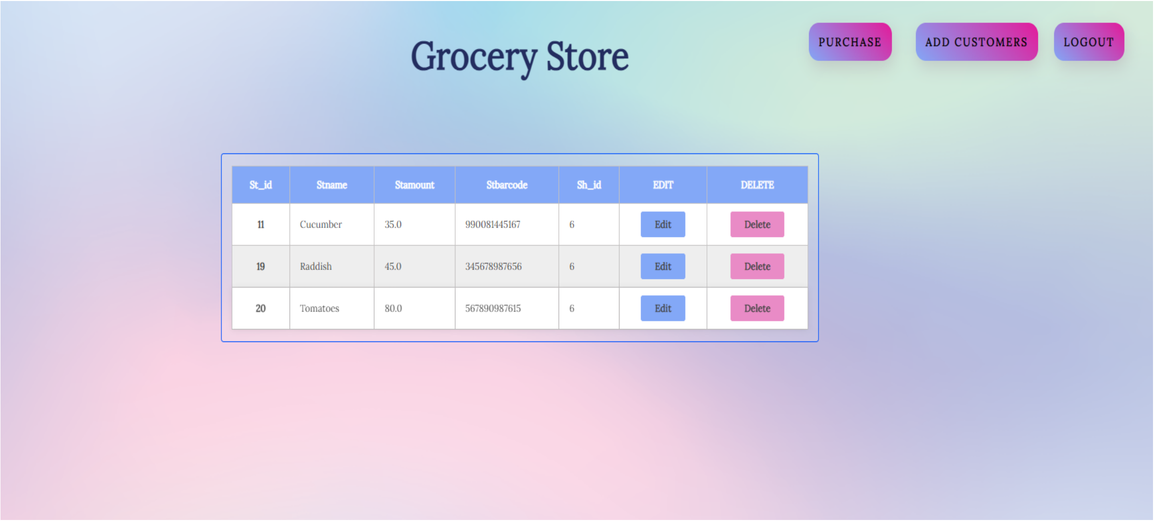
Figure 4.7: Structure of Shop table

Chapter 5

Results and Discussion

Shop Page :

Once the employee logs in, he is redirected to the shop page where the details of the stocks in that shop is seen with the facility to edit and delete a particular stock whenever needed.

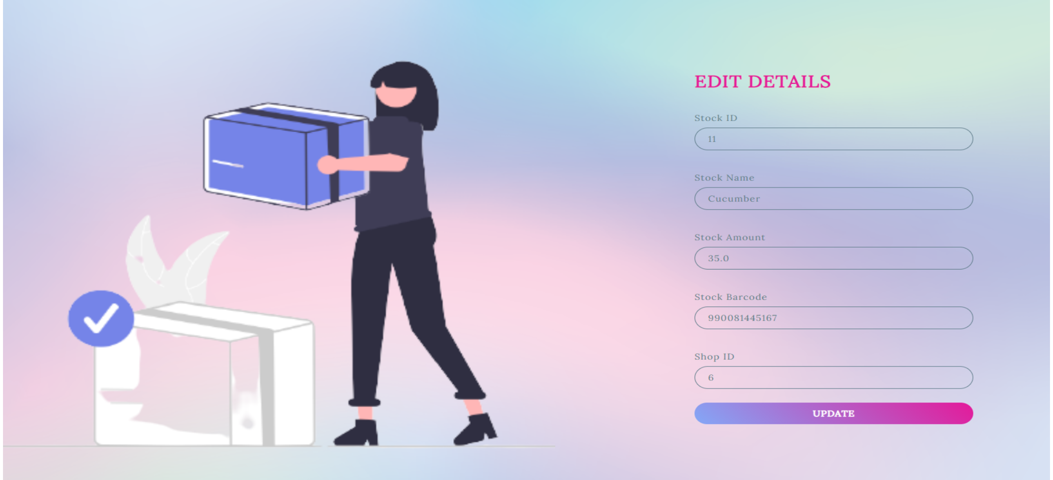


St_id	Stname	Stamount	Stbarcode	Sh_id	EDIT	DELETE
11	Cucumber	35.0	990081445167	6	Edit	Delete
19	Raddish	45.0	345678987656	6	Edit	Delete
20	Tomatoes	80.0	567890987615	6	Edit	Delete

Figure 5.1: Shop Page

Update Page :

When edit option in the shop page is clicked, it will redirect him to update page where the details of stock can be updated. The changes made are stored in the database.



EDIT DETAILS

Stock ID: 11

Stock Name: Cucumber

Stock Amount: 35.0

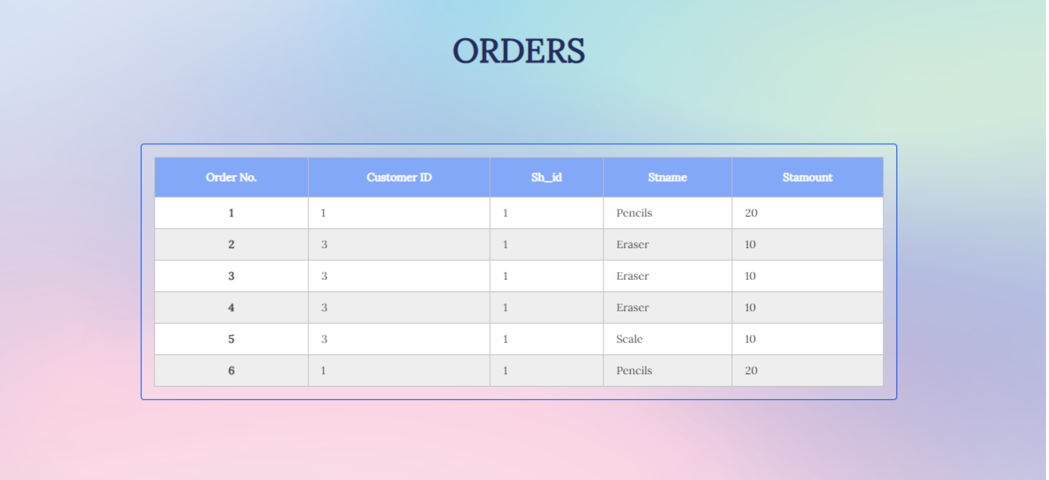
Stock Barcode: 990081445167

Shop ID: 6

Figure 5.2: Update Page

Orders Page :

When a customers orders the stock he wants by entering the details of the stock he wants, it will be visible in this page.



ORDERS

Order No.	Customer ID	Sh_id	Stname	Stamount
1	1	1	Pencils	20
2	3	1	Eraser	10
3	3	1	Eraser	10
4	3	1	Eraser	10
5	3	1	Scale	10
6	1	1	Pencils	20

Figure 5.3: Orders Page

Chapter 6

Conclusion and Future work

The Cash and Carry Management System is using Python-Flask SQLAlchemy Toolkit to develop and realize procurement of the stocks in the supermarket. With the continuous improvement of science and technology, the computer's powerful function has been known and used. As a future scope, the auto generation of the stock details will be done when a particular stock id is entered for a purchase for improving the efficiency thereby reducing time. By keeping track of when the employee was registered or deleted, giving a warning when the shop is unavailable or the stocks quantity goes low, the supermarket will analyze how to proceed further.

References

[1] Database systems Models, Languages, Design and Application Programming, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, Pearson.

[2] Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill.

[3] Silberschatz Korth and Sudharshan: Database System Concepts, 6th Edition, McGraw Hill, 2013.

[4] Coronel, Morris, and Rob, Database Principles Fundamentals of Design, Implementation and Management, Cengage Learning 2012.