

Grocery Management System

A Major Project Report submitted to

Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal

in partial fulfillment of the requirements for the award of

Degree of

Bachelor of Engineering

in

Information Technology

BY

AYUSH VIJAYVARGIYA (0832IT141017)

ABUZER PATHAN (0832IT141004)

ANOOB SINGH PARMAR (0832IT141013)

YASH VARDHAN SINGH CHOUHAN (0832IT141054)

)

UNDER THE GUIDANCE OF

Prof. Radheshyam Acholiya

(Designation)



Session: 2017-18

Department of Information Technology

**Chameli Devi Group of Institutions, Indore
452 020 (Madhya Pradesh)**

DECLARATION

We certify that the work contained in this report is original and has been done by us under the guidance of my supervisor(s).

- a. The work has not been submitted to any other Institute for any degree or diploma.
- b. We have followed the guidelines provided by the Institute in preparing the report.
- c. We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
- d. Whenever we have used materials (data, theoretical analysis, figures, and text) from other sources, we have given due credit to them by citing them in the text of the report and giving their details in the references.

NAME AND SIGNATURE OF PROJECT TEAM MEMBERS:

Sr. No.	Enrollment No.	Name of students	Signature of students
1.	0832IT141017	Ayush Vijayvargiya	
2.	0832IT141004	Abuzar Pathan	
3.	0832IT141013	Anoop Singh Parmar	
4.	0832IT141054	Yash Vardhan Singh Chouhan	

CHAMELI DEVI GROUP OF INSTITUTIONS, INDORE



CERTIFICATE

Certified that the project report entitled **TD Racer** is a bonafide work done under my guidance by Ayush Vijayvargiya, Abuzar Pathan, Anoop Singh Parmar and Yash Vardhan Singh Chouhan in partial fulfillment of the requirements for the award of degree of Bachelor of Engineering in Information Technology Engineering.

Date:

(Mr. Radheshyam Acholiya)

Guide

(Mr. Jasvant Mandloi)

Head of the Department

(Dr. K.S. Jairaj)

(Dean, CDGI)

(Internal Examiner)

(External Examiner)

CHAMELI DEVI GROUP OF INSTITUTIONS

INDORE

ACKNOWLEDGEMENT

We have immense pleasure in expressing our sincerest and deepest sense of gratitude towards our guide Mr. Radheshyam Acholiya for the assistance, valuable guidance and co-operation in carrying out this Project successfully. We have developed this project with the help of Faculty members of our institute and we are extremely grateful to all of them. We also take this opportunity to thank Head of the Department Mr. Jasvant Mandloi, and Dean of Chameli Devi Group of Institutions, Dr. K.S. Jairaj, for providing the required facilities in completing this project. We are greatly thankful to our parents, friends and faculty members for their motivation, guidance and help whenever needed.

NAME AND SIGNATURE OF TEAM MEMBERS:

1.
2.
3.
4.

TABLE OF CONTENTS

S.No	CONTENTS	No.
1	Title Page	i
2	Certificate by the Supervisor	ii
3	Declaration	v
4	Acknowledgement	vi
5	List of Figures	ix
6	<ul style="list-style-type: none"> 1) Introduction of Project study objectives 2) Technology used 3) Database Used 4) Scripting languages used 5) Tools, Software used in project 6) Related Work and Study 7) Practical application of the Project 8) Limitation and Future Enhancement 9) UML Diagrams 10) Database Connectivity code 11) Minimum Hardware and Software requirements 	
7	Conclusions	
8	References	

Fig.No.	List of Figures	Page No.
1.1	Problem Definition	1
1.2	Scope of Project	1
1.3	Need of Project	1
1.4	Report Organization	1
2	Technology Used	2
2.1	Python	2
2.2	Tkinter	5
3	Database used	7
3.1	Sqlite3	7
4	Scripting Method used	9
4.1	Random	9
4.2	Time	10
5	Tools used in the project	12
5.1	Notepad++	12
5.1.1	Notepad++ Features	12
6	Related Work and Study	14
7	Practical Application of the Project	15
8	Limitation and Future Enhancemment	16
8.1	Limitation	16
8.2	Future Enhancement	16
9	Diagram	17
9.1	Use case Diagram	17
9.2	Component Diagram	18
9.3	Deployment Diagram	19

9.4	Activity Diagram	19
9.4.1	Add stock item activity diagram	19
9.4.2	Delete stock item activity diagram	20
9.4.3	Update stock item activity diagram	20
9.5	Screenshots	21
10	Database Connectivity Code	24
11	Hardware and Software Specification	25
11.1	Hardware Specification	25
11.2	Software Specification	25
12	Conclusion and References	26
12.1	Conclusion	26
12.2	References	27

Chapter 1 –

1 Introduction

1.1 Problem Definition:

At present the Grocery Management System was done manually. It is difficult to manage different file system and it is slow and steady process. This phase of management system also lead to false information. The management use manually to keeping they record of the store, but the system is difficult to maintain due to the cost of material, human errors, less data integrity, difficulty in searching and retrieving product and feasible loss of records and retrieving files.

1.2 Scope of Project

The scope of this system is to provide user efficient working environment and more output can be generated through this. This system provides user friendly interface resulting in knowing each and every usability features of the system. This system helps in tracking records so that past records can be verified through them and one can make decisions based on the past records. This system completes the work in a very less time resulting in less time consumption and high level of efficiency.

1.3 Need of Project

The need of project is to fulfill the solution regarding of the maintenance of stock and automated billing. This software is supported to eliminate and in some cases reduce the hardships faced by this existing system. This application can be used by any grocery store to automate the process of manually maintaining the records related to the subject of maintaining the stock and liquid flows.

1.4 Report Organization

The purpose of this documentation is to describe all external requirements of software and hardware. It serves its purpose as an management product. This application can be used by any grocery store to automate the process of manually maintaining the records related to the subject of maintaining the stock and liquid flows.

Chapter 2 –

2 Technology Used –

There are several technologies used in the coding of this project Python, its libraries like tkinter, sqlite, os etc..

2.1 Python -

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

Python has a large standard library, commonly cited as one of Python's greatest strengths, providing tools suited to many tasks. This is deliberate and has been described as a "batteries included" Python philosophy. For Internet-facing applications, many standard formats and protocols (such as MIME and HTTP) are supported. Modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary precision decimals, manipulating regular expressions, and doing unit testing are also included.

Some parts of the standard library are covered by specifications (for example, the Web Server Gateway Interface (WSGI) implementation [wsgiref](#) follows PEP 333), but most modules are not. They are specified by their code, internal documentation, and test suites (if supplied). However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.

As of September 2017, the Python Package Index, the official repository containing third-party software for Python, contains over 117,000 packages offering a wide range of functionality, including:

- Graphical user interfaces, web frameworks, multimedia, databases, networking
- Test frameworks, automation and web scraping, documentation tools, system administration
- Scientific computing, text processing, image processing.

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and many language features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution.

The design of Python offers some support for functional programming in the Lisp tradition. The language has `filter()`, `map()`, and `reduce()` functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (`itertools` and `functools`) that implement functional tools borrowed from Haskell and Standard ML.

The core philosophy of the language is summarized by the document *The Zen of Python* (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

Rather than requiring all desired functionality to be built into the language's core, Python was designed to be highly extensible. Python can also be embedded in existing applications that need a programmable interface. This design of a small core language with a large standard library and an easily extensible interpreter was intended by Van Rossum from the start because of his frustrations with ABC, which espoused the opposite mindset.

While offering choice in coding methodology, the Python philosophy rejects exuberant syntax, such as in Perl, in favour of a sparser, less-cluttered grammar. As Alex Martelli put it: "To describe something as clever is *not* considered a compliment in the Python culture. "Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favour of "there should be one—and preferably only one—obvious way to do it".

Python's developers strive to avoid premature optimization, and moreover, reject patches to non-critical parts of CPython that would offer a marginal increase in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or try using PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is making it fun to use. This is reflected in the origin of the name, which comes from Monty Python, and in an occasionally playful approach to tutorials and reference materials, such as using examples that refer to spam and eggs instead of the standard foo and bar.

A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.

Python is intended to be a highly readable language. It is designed to have an uncluttered visual layout, often using English keywords where other languages use punctuation. Python does not use curly brackets to delimit blocks, and semicolons after statements are optional, in contrast

to many other programming languages. Further, Python has fewer syntactic exceptions and special cases than C or Pascal.

2.2 Tkinter –

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with the standard Microsoft Windows and Mac OS X install of Python.

The name Tkinter comes from Tk interface. Tkinter was written by Fredrik Lundh.

As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

Python 2.7 and Python 3.1 incorporate the "themed Tk" ("ttk") functionality of Tk 8.5. This allows Tk widgets to be easily themed to look like the native desktop environment in which the application is running, thereby addressing a long-standing criticism of Tk (and hence of Tkinter).

There are several popular GUI library alternatives available, such as wxPython, PyQt (PySide), Pygame, Pyglet, and PyGTK.

Tkinter is free software released under a Python license.

Some definitions

Window

This term has different meanings in different contexts, but in general it refers to a rectangular area somewhere on the user's display screen.

Top Level Window

A window that exists independently on the screen. It will be decorated with the standard frame and controls for the desktop manager. It can be moved around the desktop, and can usually be resized.

Widget

The generic term for any of the building blocks that make up an application in a graphical user interface. Examples of widgets: buttons, radiobuttons, text fields, frames, and text labels.

Frame

In Tkinter, the Frame widget is the basic unit of organization for complex layouts. A frame is a rectangular area that can contain other widgets.

Child and parent

When any widget is created, a parent-child relationship is created. For example, if you place a text label inside a frame, the frame is the parent of the label.

Chapter 3 –

3 Database used –

3.1 Sqlite3 -

SQLite3 can be integrated with Python using `sqlite3` module, which was written by Gerhard Haring. It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249. You do not need to install this module separately because it is shipped by default along with Python version 2.5.x onwards.

To use `sqlite3` module, you must first create a connection object that represents the database and then optionally you can create a cursor object, which will help you in executing all the SQL statements. Following are important `sqlite3` module routines, which can suffice your requirement to work with SQLite database from your Python program. If you are looking for a more sophisticated application, then you can look into Python `sqlite3` module's official documentation.

`sqlite3.connect(database [,timeout ,other optional arguments])`

This API opens a connection to the SQLite database file. You can use `":memory:"` to open a database connection to a database that resides in RAM instead of on disk. If database is opened successfully, it returns a connection object.

When a database is accessed by multiple connections, and one of the processes modifies the database, the SQLite database is locked until that transaction is committed. The timeout parameter specifies how long the connection should wait for the lock to go away until raising an exception. The default for the timeout parameter is 5.0 (five seconds).

If the given database name does not exist then this call will create the database. You can specify filename with the required path as well if you want to create a database anywhere else except in the current directory.

`connection.cursor([cursorClass])`

This routine creates a **cursor** which will be used throughout of your database programming with Python. This method accepts a single optional parameter `cursorClass`. If supplied, this must be a custom cursor class that extends `sqlite3.Cursor`.

`cursor.execute(sql [, optional parameters])`

This routine executes an SQL statement. The SQL statement may be parameterized (i. e. placeholders instead of SQL literals). The sqlite3 module supports two kinds of placeholders: question marks and named placeholders (named style).

For example – `cursor.execute("insert into people values (?, ?)", (who, age))`

`connection.execute(sql [, optional parameters])`

This routine is a shortcut of the above execute method provided by the cursor object and it creates an intermediate cursor object by calling the cursor method, then calls the cursor's execute method with the parameters given.

`connection.commit()`

This method commits the current transaction. If you don't call this method, anything you did since the last call to commit() is not visible from other database connections.

`connection.close()`

This method closes the database connection. Note that this does not automatically call commit(). If you just close your database connection without calling commit() first, your changes will be lost!

Chapter 4 –

4 Scripting Method used –

4.1 Random –

This module implements pseudo-random number generators for various distributions.

For integers, uniform selection from a range. For sequences, uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement.

On the real line, there are functions to compute uniform, normal (Gaussian), lognormal, negative exponential, gamma, and beta distributions. For generating distributions of angles, the von Mises distribution is available.

Almost all module functions depend on the basic function `random()`, which generates a random float uniformly in the semi-open range `[0.0, 1.0)`. Python uses the Mersenne Twister as the core generator. It produces 53-bit precision floats and has a period of $2^{19937}-1$. The underlying implementation in C is both fast and threadsafe. The Mersenne Twister is one of the most extensively tested random number generators in existence. However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes.

The functions supplied by this module are actually bound methods of a hidden instance of the `random.Random` class. You can instantiate your own instances of `Random` to get generators that don't share state. This is especially useful for multi-threaded programs, creating a different instance of `Random` for each thread, and using the `jumpahead()` method to make it likely that the generated sequences seen by each thread don't overlap.

Class `Random` can also be subclassed if you want to use a different basic generator of your own devising: in that case, override the `random()`, `seed()`, `getstate()`, `setstate()` and `jumpahead()` methods. Optionally, a new generator can supply a `getrandbits()` method — this allows `randrange()` to produce selections over an arbitrarily large range. New in version 2.4: the `getrandbits()` method.

As an example of subclassing, the `random` module provides the `WichmannHill` class that implements an alternative generator in pure Python. The class provides a backward compatible way to reproduce results from earlier versions of Python, which used the Wichmann-Hill algorithm as the core generator. Note that this Wichmann-Hill generator can no longer be recommended: its period is too short by contemporary standards, and the sequence generated is known to fail some stringent randomness tests. See the references below for a recent variant that repairs these flaws.

Changed in version 2.3: `MersenneTwister` replaced `WichmannHill` as the default generator.

The `random` module also provides the `SystemRandom` class which uses the system function `os.urandom()` to generate random numbers from sources provided by the operating system.

4.2 Time –

This module provides various time-related functions. For related functionality, see also the `datetime` and `calendar` modules.

Although this module is always available, not all functions are available on all platforms. Most of the functions defined in this module call platform C library functions with the same name. It may sometimes be helpful to consult the platform documentation, because the semantics of these functions varies among platforms.

An explanation of some terminology and conventions is in order.

- The epoch is the point where the time starts, and is platform dependent. For Unix, the epoch is January 1, 1970, 00:00:00 (UTC). To find out what the epoch is on a given platform, look at `time.gmtime(0)`.
- The term seconds since the epoch refers to the total number of elapsed seconds since the epoch, typically excluding leap seconds. Leap seconds are excluded from this total on all POSIX-compliant platforms.

- The functions in this module may not handle dates and times before the epoch or far in the future. The cut-off point in the future is determined by the C library; for 32-bit systems, it is typically in 2038.
- UTC is Coordinated Universal Time (formerly known as Greenwich Mean Time, or GMT). The acronym UTC is not a mistake but a compromise between English and French.
- The precision of the various real-time functions may be less than suggested by the units in which their value or argument is expressed. E.g. on most Unix systems, the clock “ticks” only 50 or 100 times a second.
- On the other hand, the precision of `time()` and `sleep()` is better than their Unix equivalents: times are expressed as floating point numbers, `time()` returns the most accurate time available (using Unix `gettimeofday()` where available), and `sleep()` will accept a time with a nonzero fraction (Unix `select()` is used to implement this, where available).
- The time value as returned by `gmtime()`, `localtime()`, and `strptime()`, and accepted by `asctime()`, `mktime()` and `strftime()`, is a sequence of 9 integers. The return values of `gmtime()`, `localtime()`, and `strptime()` also offer attribute names for individual fields.

`time.localtime([secs])`

Like `gmtime()` but converts to local time. If `secs` is not provided or `None`, the current time as returned by `time()` is used. The `dst` flag is set to 1 when DST applies to the given time.

Chapter 5 –

5 Tools used in the project –

5.1 Notepad++ -

The main tool that is used in the development of this project is Notepad+.

Notepad++ is a highly functional, free, open-source, editor for MS Windows that can recognize (i.e., highlight syntax for) several different programming languages from Assembly to XML, and many others in between, including, of course, Python.

Besides syntax highlighting, Notepad++ has some features that are particularly useful to coders. It will allow you to create shortcuts to program calls, such as a *Run Python* menu item that will invoke *python.exe* to execute your Python code without having to switch over to another window running a Python shell, such as IPython.

Another very convenient feature is that it will group sections of code and make them collapsible so that you can hide blocks of code to make the page/window more readable.

Notepad++ provides indentation guides, particularly useful for Python which relies not on braces to define functional code blocks, but rather on indentation levels.

5.1.1 Notepad++ Features –

- Syntax Highlighting and Syntax Folding
- User Defined Syntax Highlighting and Folding: screenshot 1, screenshot 2, screenshot 3 and screenshot 4
- PCRE (Perl Compatible Regular Expression) Search/Replace
- GUI entirely customizable: minimalist, tab with close button, multi-line tab, vertical tab and vertical document list
- Document Map

- Auto-completion: Word completion, Function completion and Function parameters hint
- Multi-Document (Tab interface)
- Multi-View
- WYSIWYG (Printing)
- Zoom in and zoom out
- Multi-Language environment supported
- Bookmark
- Macro recording and playback
- Launch with different arguments

Chapter 6 –

6 Related Work and Study –

Most of the study done to make the project was offline. I found this by one of the instance of my life. At my native place my relative uncle had a grocery shop and he got a lot of trouble in inventory control. I influenced terrific by this circumstance so that I want to reduce headache by making a desktop application which will maintain inventory control as well as maintenance of a shop. Moreover, through this app we will also create a bill. It will help in burgeoning an efficiency of service furthermore it will save a lot of time. So I collect all the information which are required to make this system through my uncle. Information includes which type of grocery they used, how to maintain the stocks, billing formate etc.. After all this requirement gathering I need to decide on which technology this system should I make after searching on Internet I found that there is a easiest way in python to make GUI application and it can also handles other tasks easily So I chooses python for making this system. There is a way in python to convert the python program into the desktop window execution(.exe) file using pyinstaller which is a great tool for converting .py to .exe.

Chapter 7 –

7 Practical Application of the Project –

This application can be used by any grocery store to automate the process of manually maintaining the records related to the subject of maintaining the stock and liquid flows. As this is generic software it can be used by a wide variety of outlets (Retailers and Wholesalers) to automate the process of manually maintaining the records related to the subject of maintaining the stock and cash flows.

The main application of the application is to automate the existing system of manually maintaining the records of the groceries, counter sales, purchases, bills and customer monetary positions and other related transactions made at the counter. The grocery records of a generic shop which deals in Wheat, rice, pulses, oil, cosmetics, etc.

- Stock maintenance
- Bill generation
- To keep accounts of purchase and sales

Chapter 8 –

8 Limitation and Future Enhancement –

8.1 Limitation –

The scope of this system is to provide user efficient working environment and more output can be generated through this. This system provides user friendly interface resulting in knowing each and every usability features of the system. Specifically, in sales, it involves computation of the daily sales, weekly, monthly and yearly, it shows inventory transaction of purchase, it can add new stock products and it secures the storage of records and its maintainability.

The system will not cover the entire enterprises needs, the said system will help to ease only some of the difficulties of the enterprises. The system generates only reports, calculate and adding stock and securing the data which always been lost during the manual report.

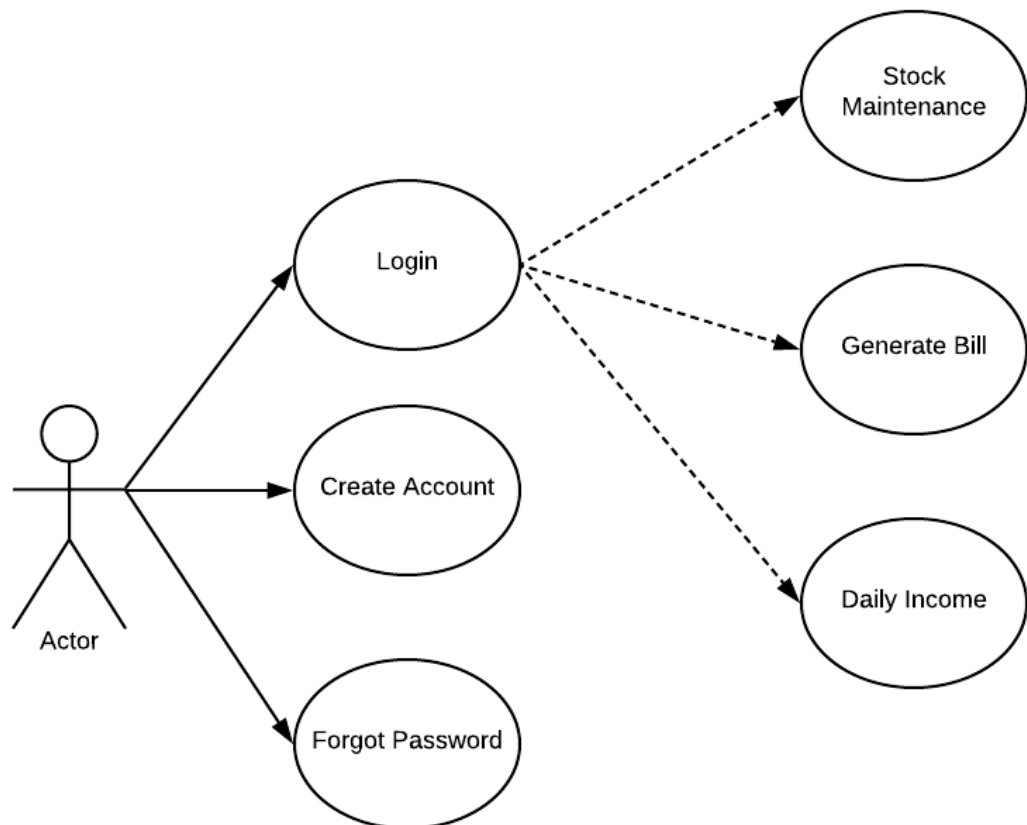
8.2 Future Enhancement –

The system can't generate a report through email or cannot add another system to another computer, in other words can't connect to another computer to review the data. So we can add this new feature in our system. One of the most import feature of printing of generated bill can be added in future. So there are some new features are in mind which can enhance current grocery management system in future.

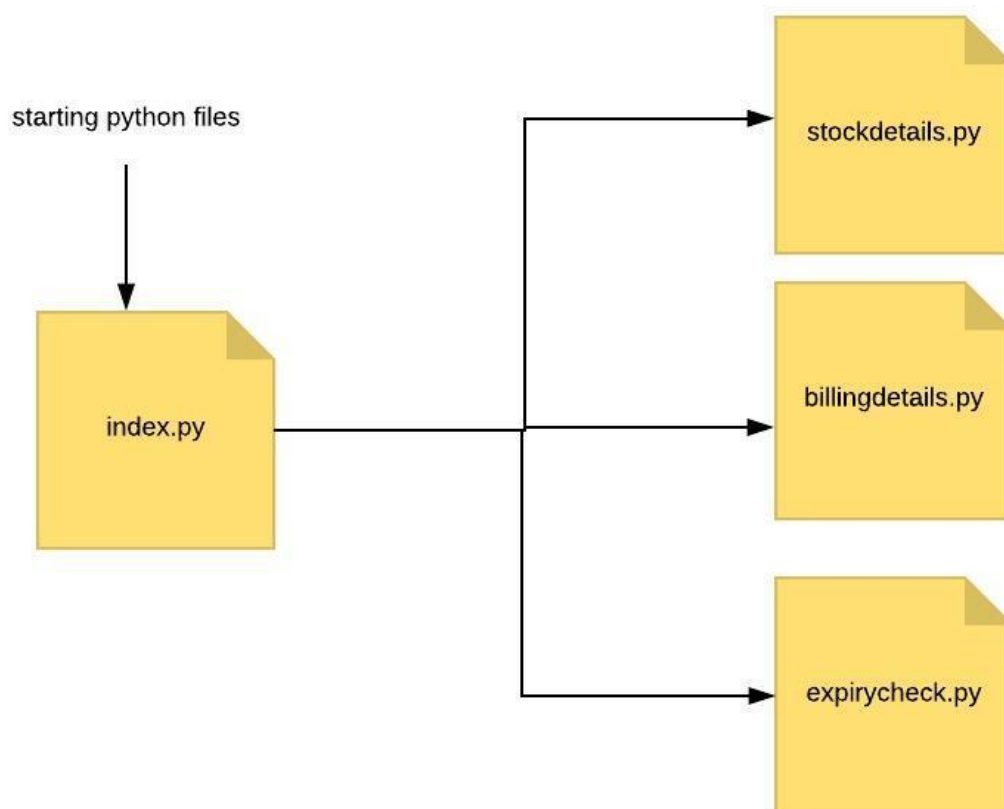
Chapter 9 –

9 Diagrams –

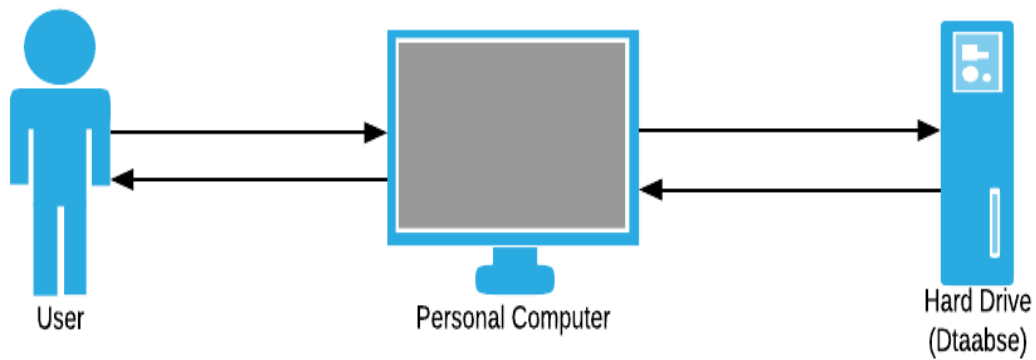
9.1 Use Case Diagram –



9.2 Component Diagram –

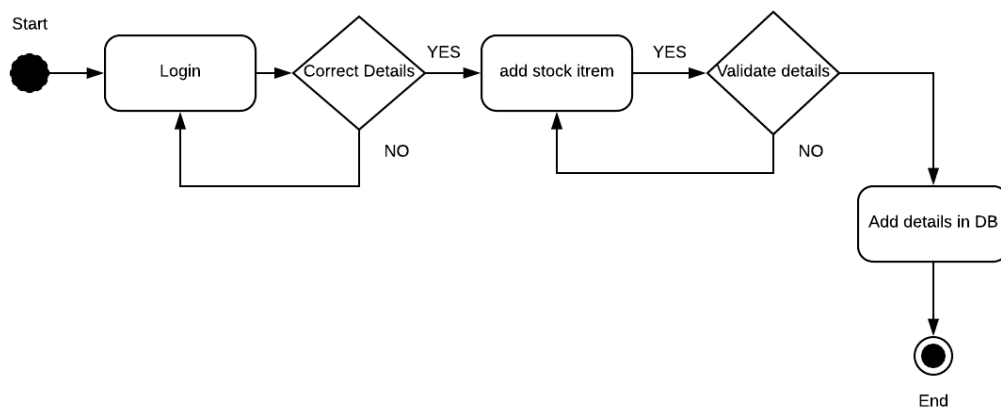


9.3 Deployment Diagram –

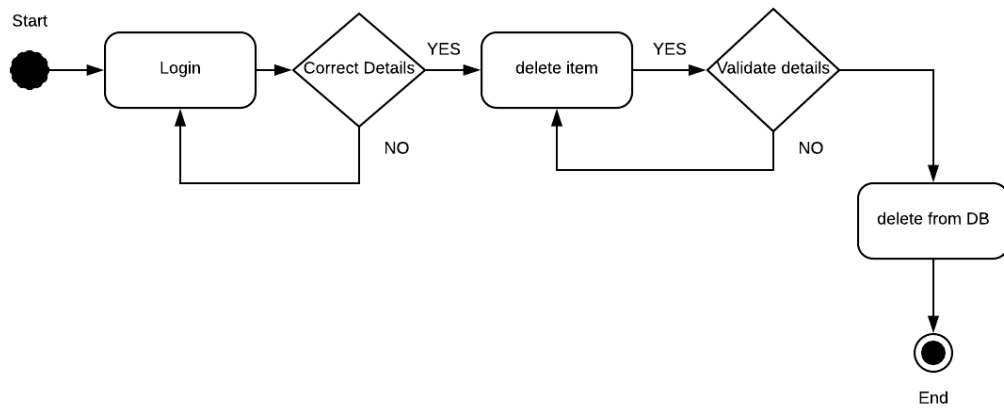


9.4 Activity Diagram –

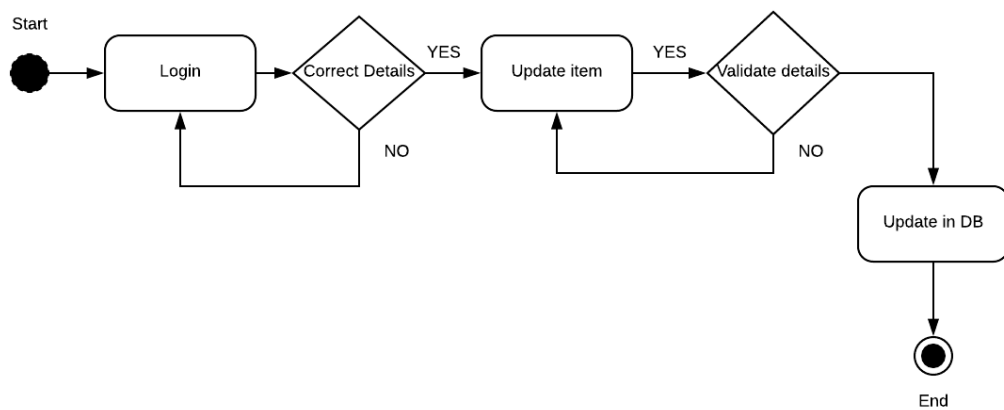
9.4.1 Add stock item activity –



9.4.2 Delete stock item activity –

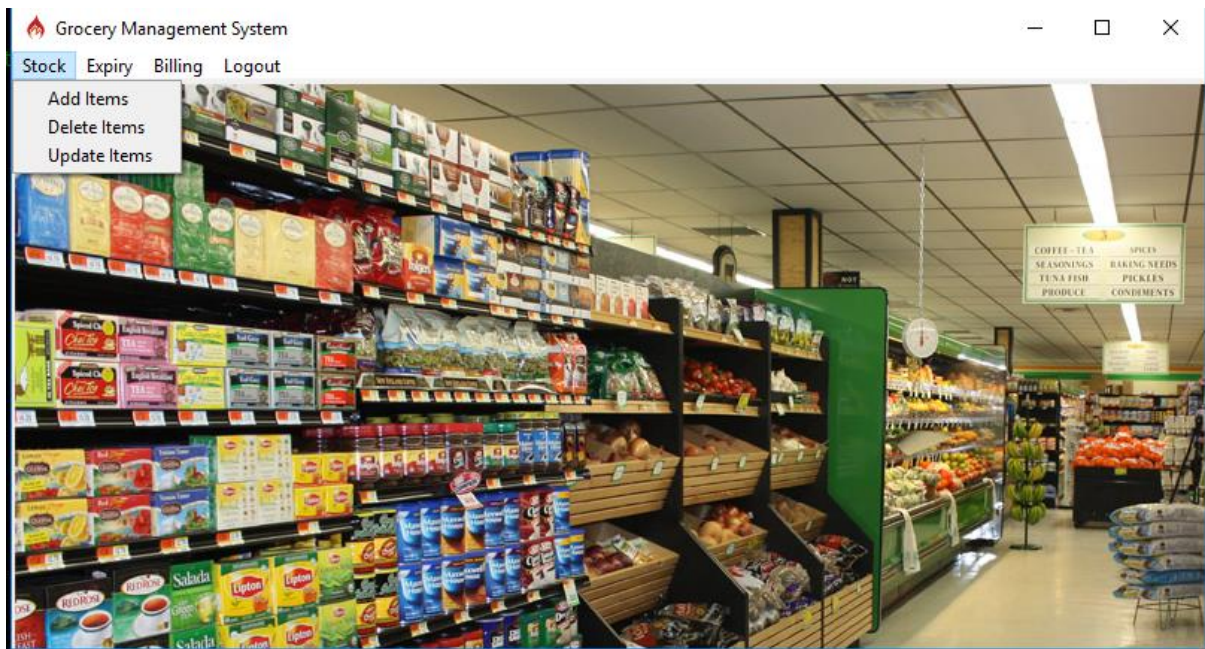


9.4.3 Update stock item activity –



9.5 Screenshots –





Add Stock

Enter a New Item to the Grocery Stock

Item_No: 3

Item_Name: Arial

Item_Type: detergent


Quantity_Remain: 30

Item_Cost: 10

Expiry_Date: 12/05/2018


Manufactured_By: Aria

Item_Name	Item_Type	Quantity_Remain	Item_Cost	Expiry_Date	Manufactured_By
1. Chai	Drink	100	20	12/05/2018	tata
2. meggi	snacks	50	10	12/05/2018	nestle
3. Arial	detergent	30	10	12/05/2018	Arial

 Check Expiry of the Items


Today: 8/4/2018
Its Illegal to sell expired items

Chai 12/05/2018

 Update grocery item fr...


Enter the Item to update to the Grocery Stock

Item_Name	Chai
Quantity_Remain	100
Cost	20
Expiry_Date	12/05/2018

 Delete grocery item from Stock

Enter the Item No to Delete

Item	Qty Remain	Cost	Expiry Date
1) Chai	100	20	12/05/2018
2) meggi	50	10	12/05/2018
3) Arial	30	10	12/05/2018

 BILLING

Enter Name:

Enter Address:

Main Menu

Refresh Stock

Reset Bill

Save Bill

Select Item	Qty_Remain	Cost	Expiry Date	QUANTITY
1 Chai	100	20	12/05/2018	
2 meggi	50	10	12/05/2018	
3 Arial	30	10	12/05/2018	

Add to bill

Chapter 10 –

10 Database Connectivity Code –

In grocery management system we create a database using sqlite3 and the database file is named as grocery.sqlite. The connection of our system with database is created using the python library sqlite3. For importing sqlite3 module the following syntax is used

```
from sqlite3 import dbapi2 as sqlite
```

Here dbapi2 is the database api of the sqlite3 module in python.

For making connection with database file “grocery.sqlite” the following syntax is used

```
login = sqlite.connect("grocery.sqlite")
```

This API opens a connection to the SQLite database file. You can use ":memory:" to open a database connection to a database that resides in RAM instead of on disk. If database is opened successfully, it returns a connection object.

```
l = login.cursor()
```

This routine creates a **cursor** which will be used throughout of your database programming with Python. This method accepts a single optional parameter cursorClass. If supplied, this must be a custom cursor class that extends sqlite3.Cursor.

So for making connection with database the following three syntax are used

```
from sqlite3 import dbapi2 as sqlite
```

```
login=sqlite.connect("grocery.sqlite")
```

```
l=login.cursor()
```


Chapter 11 –

11 Hardware and Software specifications –

11.1 Hardware Specification –

- System : Dual core 2.4 GHz.
- Hard Disk : 500 GB.
- Ram : 2 GB.
- Keyboard : Standard key board
- Mouse : 2 button mouse
- Clock speed : 500 MHZ
- System bus : 32 bits

11.2 Software Specification –

- Operating system : Windows 7 or Above.
- Coding Language : Python.
- Data Base : SQLite v3

Chapter 12 –

12 Conclusion and References –

12.1 Conclusion –

In conclusion, an Grocery Management System is a perfect solution for the keeping the record of the store. The system can be used as a tool to maintain due to the cost of material, human errors, less data integrity, difficulty in searching and retrieving product and feasible loss of records and retrieving files. We were following SDLC phases such as planning, analysis, design and implementation of system to come out with a full documentation. Then, we also discussed the advantages of programming languages as you see in the documentation.

The objective of this project was to build a program for maintaining the details of all stocks and sales .The system developed is able to meet all the basic requirements. It will provide the facility to the shopkeeper so that they can keep tracks of all the items in stocks. The management of the Inventory will be also benefited by the proposed system, as it will automate the whole supply procedure, which will reduce the workload. The security of the system is also one of the prime concerns.

There is always a room for improvement in any software, however efficient the system may be. The important thing is that the system should be flexible enough for future modifications. The system has been factored into different modules to make system adapt to the further changes. Every effort has been made to cover all user requirements and make it user friendly.

- **Goal achieved:** The System is able provide the interface to the user so that he can replicate his desired data.
- **User friendliness:** Though the most part of the system is supposed to act in the background, efforts have been made to make the foreground interaction with user as smooth as possible. Also the integration of the system with Grocery Management project has been kept in mind throughout the development phase.

12.2 References –

Martelli, Alex; Ravenscroft, Anna; Ascher, David (2005). Python Cookbook, 2nd Edition. O'Reilly Media. p. 230. ISBN 978-0-596-00797-3.

Python Official Docs, 2017, <https://docs.python.org/>

Graphical User Interfaces with Tk - Python3.6.5 documentation, 2017, <https://docs.python.org/3/library/tk.html>

sqlite3 — DB-API 2.0 interface for SQLite databases — Python3.6.5, 2017, <https://docs.python.org/3/library/sqlite3.html>