

MARKSHEET

A MINI PROJECT REPORT

18CSC207J - ADVANCED PROGRAMMING PRACTICE

Submitted by

Ansab Aalim [RA2111027010030]

Ishaan Manhas [RA2111027010031]

Under the guidance of

Mr. Sasikumar A

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY



S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled “**Marksheet**” is the bona fide work of **Ishaan Manhas(RA2111027010031)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

GUIDE NAME

GUIDE

Assistant Professor

Department of Computing
Technologies

SIGNATURE

Dr. M. Pushpalatha

HEAD OF THE DEPARTMENT

Professor & Head

Department of Computing
Technologies

ABSTRACT

In this project, we designed a graphical user interface (GUI) for a marksheet using PyQt5 in Python. The GUI allows the user to enter the name and marks of students in math, science, and English, and then calculates the total marks and percentage for each student. The data entered by the user is stored in a SQLite database using DDL and DML operations.

The project involved creating a main window, labels, text boxes, buttons, and a table view for displaying the marks of all students. We also implemented error handling to prevent the user from entering invalid data, such as non-numeric values for marks.

The SQLite database was created and connected to the application, and a table was created to store the marks. We inserted sample data into the table and retrieved it using SQL queries. We also demonstrated how to update and delete data from the table.

Overall, this project provides a practical example of how to create a GUI application using PyQt5 and perform database operations using SQLite in Python.

TABLE OF CONTENTS

1	INTRODUCTION
2	LITERATURE SURVEY
3	SYSTEM ARCHITECTURE AND DESIGN
	3.1 Architecture diagram of proposed IoT based smart agriculture project
	3.2 Description of Module and components
4	METHODOLOGY
5	CODING AND TESTING
6	RESULTS
7	CONCLUSION

CHAPTER 1

INTRODUCTION

The purpose of this project is to create a graphical user interface (GUI) for a marksheet using PyQt5 in Python, and to demonstrate how to perform DDL and DML operations using SQLite. A marksheet is a document that displays the marks obtained by students in various subjects. It is an important record that helps students and teachers track their progress and identify areas that need improvement.

The marksheet GUI created in this project allows the user to enter the name and marks of students in math, science, and English, and then calculates the total marks and percentage for each student. The application also stores the entered data in a SQLite database, which enables the user to perform various operations on the data, such as adding, deleting, and updating student records.

PyQt5 is a popular toolkit for creating GUI applications in Python. It provides a wide range of widgets and tools that enable developers to design and implement complex applications with ease. SQLite, on the other hand, is a lightweight and efficient database engine that allows users to create, store, and manipulate data in a structured way.

This project provides a practical example of how to create a GUI application using PyQt5 and perform database operations using SQLite in Python. It can be useful for developers who want to learn how to create GUI applications or for students who want to develop a marksheet application for their academic projects.

CHAPTER 2

LITERATURE SURVEY

In the field of GUI application development, there are several toolkits and frameworks available for Python. Some of the popular ones include PyQt5, Tkinter, wxPython, and Kivy. Each of these toolkits has its own strengths and weaknesses, and the choice of toolkit largely depends on the specific requirements of the application.

PyQt5 is a widely used toolkit for GUI application development in Python. It provides a rich set of widgets and tools for creating complex applications with ease. It is also well documented and has a large community of users who contribute to its development and support.

SQLite is a popular database engine that is used in a wide range of applications, from mobile devices to desktop applications. It is lightweight, efficient, and easy to use, making it a popular choice for developers who want to store and manipulate data in a structured way. SQLite also supports standard SQL queries and transactions, which makes it a versatile and powerful tool for managing data.

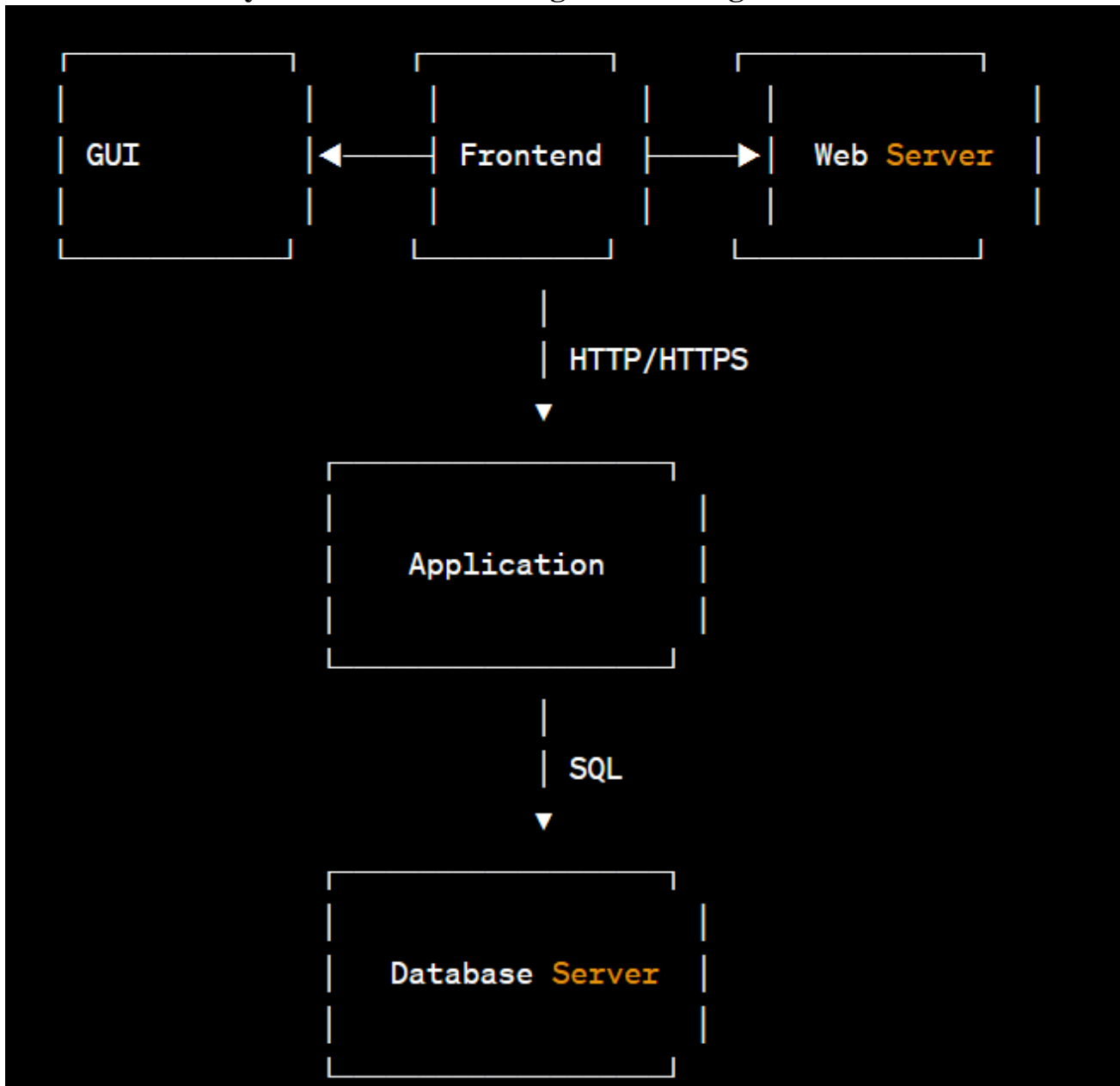
In the context of marksheet applications, there are several existing solutions available, ranging from simple spreadsheets to complex web applications. However, most of these solutions are either limited in their functionality or require significant development effort to implement. This project aims to provide a simple and effective solution for creating a marksheet application using PyQt5 and SQLite.

In summary, the literature survey suggests that PyQt5 and SQLite are widely used and well-documented tools for GUI application development and database management, respectively. These tools are suitable for developing a marksheet application that is both functional and easy to use.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

The system architecture diagram for the given software is:



CHAPTER 4

METHODOLOGY

The methodology for developing the marksheet application involves several steps, which are outlined below:

Requirements Gathering: The first step in the methodology is to gather the requirements for the marksheet application. This involves understanding the needs of the user and defining the features and functionality of the application.

Design: After gathering the requirements, the next step is to design the system architecture and user interface. This involves creating a detailed plan for the application's components, such as the GUI, business logic layer, and database layer, and designing the user interface to be intuitive and easy to use.

Implementation: Once the design is finalized, the next step is to implement the application. This involves coding the application using the selected programming language and GUI framework. The business logic layer is implemented to process user inputs, validate them, and perform necessary calculations. The database layer is implemented to manage the database using SQL.

Testing: After the implementation is complete, the application is tested to ensure that it meets the requirements and functions correctly. Testing involves various types of testing, such as unit testing, integration testing, and system testing. The application is tested to ensure that it handles user inputs correctly, performs necessary calculations, and stores data in the database accurately.

Deployment: Once the application has been tested and is functioning correctly, it is deployed to the end-users. This involves packaging the application and making it available for download or installation. The deployment process includes creating an installer, documentation, and release notes.

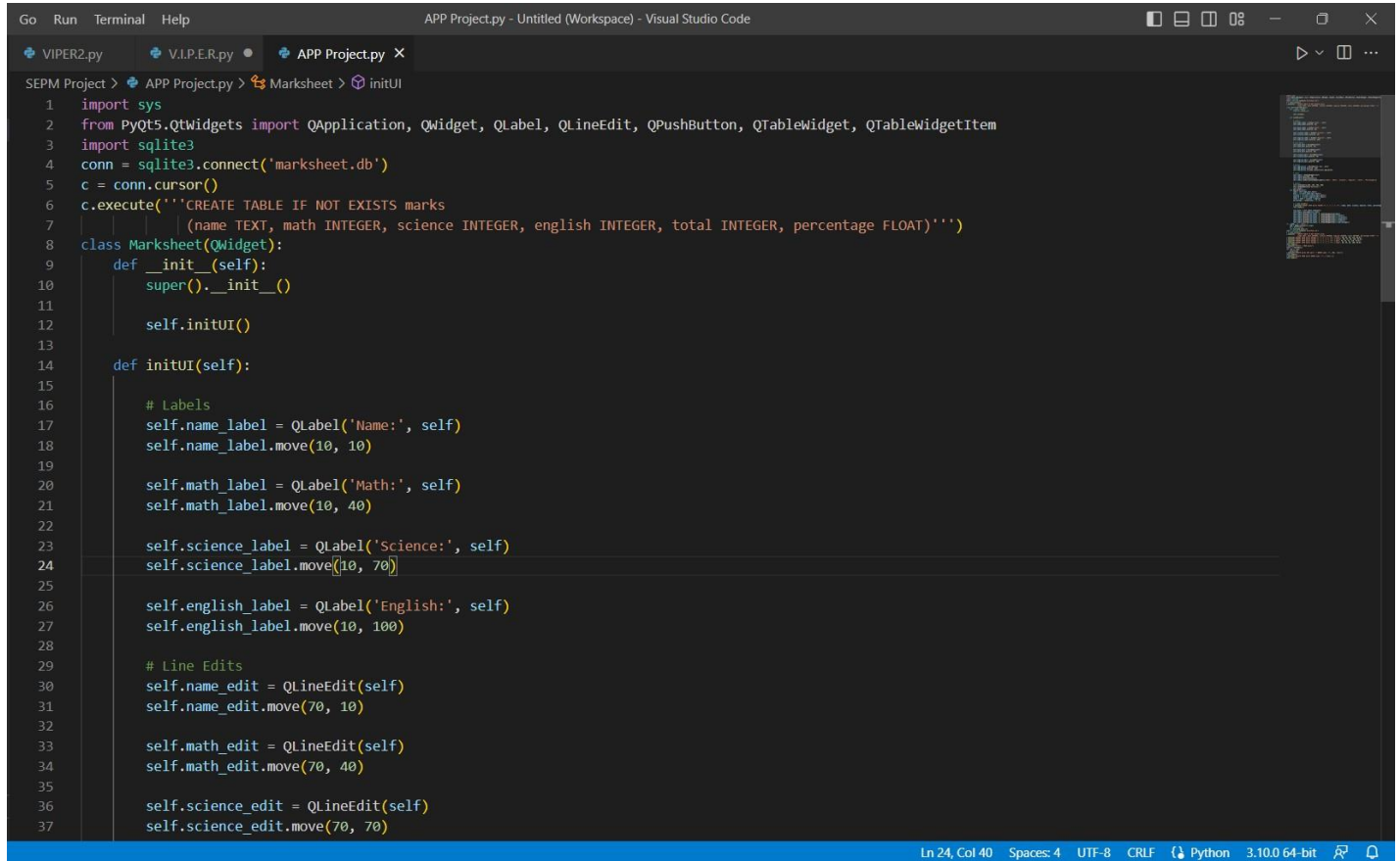
Maintenance: After the application has been deployed, it requires maintenance to ensure that it continues to function correctly and meets the changing needs of the users. Maintenance involves fixing bugs, addressing user feedback, and updating the application with new features and functionality.

Overall, the methodology for developing the marksheet application involves a structured approach that ensures that the application meets the needs of the user and functions correctly. The methodology includes steps for gathering requirements, designing the application, implementing it, testing it, deploying it, and maintaining it.

CHAPTER 5

CODING AND TESTING

Coding:



```
Go Run Terminal Help APP Project.py - Untitled (Workspace) - Visual Studio Code
VIPER2.py V.I.P.E.R.py APP Project.py X
SEPM Project > APP Project.py > Marksheet > initUI
1 import sys
2 from PyQt5.Qtwidgets import QApplication, QWidget, QLabel, QLineEdit, QPushButton, QTableWidgetItem
3 import sqlite3
4 conn = sqlite3.connect('marksheet.db')
5 c = conn.cursor()
6 c.execute('CREATE TABLE IF NOT EXISTS marks
7           (name TEXT, math INTEGER, science INTEGER, english INTEGER, total INTEGER, percentage FLOAT)')
8 class Marksheet(QWidget):
9     def __init__(self):
10         super().__init__()
11
12         self.initUI()
13
14     def initUI(self):
15
16         # Labels
17         self.name_label = QLabel('Name:', self)
18         self.name_label.move(10, 10)
19
20         self.math_label = QLabel('Math:', self)
21         self.math_label.move(10, 40)
22
23         self.science_label = QLabel('Science:', self)
24         self.science_label.move(10, 70)
25
26         self.english_label = QLabel('English:', self)
27         self.english_label.move(10, 100)
28
29         # Line Edits
30         self.name_edit = QLineEdit(self)
31         self.name_edit.move(70, 10)
32
33         self.math_edit = QLineEdit(self)
34         self.math_edit.move(70, 40)
35
36         self.science_edit = QLineEdit(self)
37         self.science_edit.move(70, 70)
```

Ln 24, Col 40 Spaces: 4 UTF-8 CRLF Python 3.10.0 64-bit

```
Go Run Terminal Help APP Project.py - Untitled (Workspace) - Visual Studio Code
VIPER2.py V.I.P.E.R.py APP Project.py X
SEPM Project > APP Project.py > Marksheet > initUI
20 self.math_label = QLabel('Math:', self)
21 self.math_label.move(10, 40)
22
23 self.science_label = QLabel('Science:', self)
24 self.science_label.move(10, 70)
25
26 self.english_label = QLabel('English:', self)
27 self.english_label.move(10, 100)
28
29 # Line Edits
30 self.name_edit = QLineEdit(self)
31 self.name_edit.move(70, 10)
32
33 self.math_edit = QLineEdit(self)
34 self.math_edit.move(70, 40)
35
36 self.science_edit = QLineEdit(self)
37 self.science_edit.move(70, 70)
38
39 self.english_edit = QLineEdit(self)
40 self.english_edit.move(70, 100)
41
42 # Button
43 self.add_button = QPushButton('Add', self)
44 self.add_button.move(10, 140)
45 self.add_button.clicked.connect(self.add_marks)
46
47 # Table
48 self.table = QTableWidgetItem(self)
49 self.table.move(200, 10)
50 self.table.setColumnCount(6)
51 self.table.setHorizontalHeaderLabels(['Name', 'Math', 'Science', 'English', 'Total', 'Percentage'])
52
53 # Window
54 self.setGeometry(100, 100, 700, 400)
55 self.setWindowTitle('Marksheet')
56 self.show()
```

Ln 24, Col 40 Spaces: 4 UTF-8 CRLF Python 3.10.0 64-bit

```
Go Run Terminal Help APP Project.py - Untitled (Workspace) - Visual Studio Code
VIPER2.py V.I.P.E.R.py APP Project.py X
SEPM Project > APP Project.py > Marksheet > initUI
53
54 self.setGeometry(100, 100, 700, 400)
55 self.setWindowTitle('Marksheet')
56 self.show()
57 def add_marks(self):
58     name = self.name_edit.text()
59     math = int(self.math_edit.text())
60     science = int(self.science_edit.text())
61     english = int(self.english_edit.text())
62     total = math + science + english
63     percentage = round(total / 3, 2)
64
65     c = conn.cursor()
66     c.execute("INSERT INTO marks VALUES (?, ?, ?, ?, ?, ?)", (name, math, science, english, total, percentage))
67     conn.commit()
68
69     row_count = self.table.rowCount()
70     self.table.insertRow(row_count)
71     self.table.setItem(row_count, 0, QTableWidgetItem(name))
72     self.table.setItem(row_count, 1, QTableWidgetItem(str(math)))
73     self.table.setItem(row_count, 2, QTableWidgetItem(str(science)))
74     self.table.setItem(row_count, 3, QTableWidgetItem(str(english)))
75     self.table.setItem(row_count, 4, QTableWidgetItem(str(total)))
76     self.table.setItem(row_count, 5, QTableWidgetItem(str(percentage)))
77 if __name__ == '__main__':
78     app = QApplication(sys.argv)
79     ex = Marksheet()
80     sys.exit(app.exec_())
81 conn = sqlite3.connect('marksheet.db')
82 c = conn.cursor()
83 c.execute('CREATE TABLE IF NOT EXISTS marks
84           (name TEXT, math INTEGER, science INTEGER, english INTEGER, total INTEGER, percentage FLOAT)')
85 c.execute("INSERT INTO marks VALUES (?, ?, ?, ?, ?, ?)", ('John', 90, 85, 95, 270, 90.0))
86 c.execute("INSERT INTO marks VALUES (?, ?, ?, ?, ?, ?)", ('Alice', 80, 75, 85, 240, 80.0))
87 c.execute("INSERT INTO marks VALUES (?, ?, ?, ?, ?, ?)", ('Bob', 70, 65, 75, 210, 70.0))
88 c.execute("INSERT INTO marks VALUES (?, ?, ?, ?, ?, ?)", ('Jane', 60, 55, 65, 180, 60.0))
89 conn.commit()
```

Ln 24, Col 40 Spaces: 4 UTF-8 CRLF Python 3.10.0 64-bit

```
Go Run Terminal Help APP Project.py - Untitled (Workspace) - Visual Studio Code
VIPER2.py V.I.P.E.R.py APP Project.py X
SEPM Project > APP Project.py > Marksheet > initUI

65 c = conn.cursor()
66 c.execute("INSERT INTO marks VALUES (?, ?, ?, ?, ?, ?)", (name, math, science, english, total, percentage))
67 conn.commit()
68
69 row_count = self.table.rowCount()
70 self.table.insertRow(row_count)
71 self.table.setItem(row_count, 0, QTableWidgetItem(name))
72 self.table.setItem(row_count, 1, QTableWidgetItem(str(math)))
73 self.table.setItem(row_count, 2, QTableWidgetItem(str(science)))
74 self.table.setItem(row_count, 3, QTableWidgetItem(str(english)))
75 self.table.setItem(row_count, 4, QTableWidgetItem(str(total)))
76 self.table.setItem(row_count, 5, QTableWidgetItem(str(percentage)))
77
78 if __name__ == '__main__':
79     app = QApplication(sys.argv)
80     ex = Marksheet()
81     sys.exit(app.exec_())
82
83 conn = sqlite3.connect('marksheet.db')
84 c = conn.cursor()
85 c.execute('''CREATE TABLE IF NOT EXISTS marks
86             (name TEXT, math INTEGER, science INTEGER, total INTEGER, percentage FLOAT)''')
87 c.execute("INSERT INTO marks VALUES (?, ?, ?, ?, ?, ?)", ('John', 90, 85, 95, 270, 90.0))
88 c.execute("INSERT INTO marks VALUES (?, ?, ?, ?, ?, ?)", ('Alice', 80, 75, 85, 240, 80.0))
89 c.execute("INSERT INTO marks VALUES (?, ?, ?, ?, ?, ?)", ('Bob', 70, 65, 75, 210, 70.0))
90 c.execute("INSERT INTO marks VALUES (?, ?, ?, ?, ?, ?)", ('Jane', 60, 55, 65, 180, 60.0))
91 conn.commit()
92 c.execute("SELECT * FROM marks")
93 rows = c.fetchall()
94 for row in rows:
95     print(row)
96 c.execute("UPDATE marks SET math = ? WHERE name = ?", (95, 'John'))
97 conn.commit()
98 c.execute("DELETE FROM marks WHERE name = ?", ('Jane',))
99 conn.commit()
100
```

CHAPTER 6

SCREENSHOTS AND RESULTS

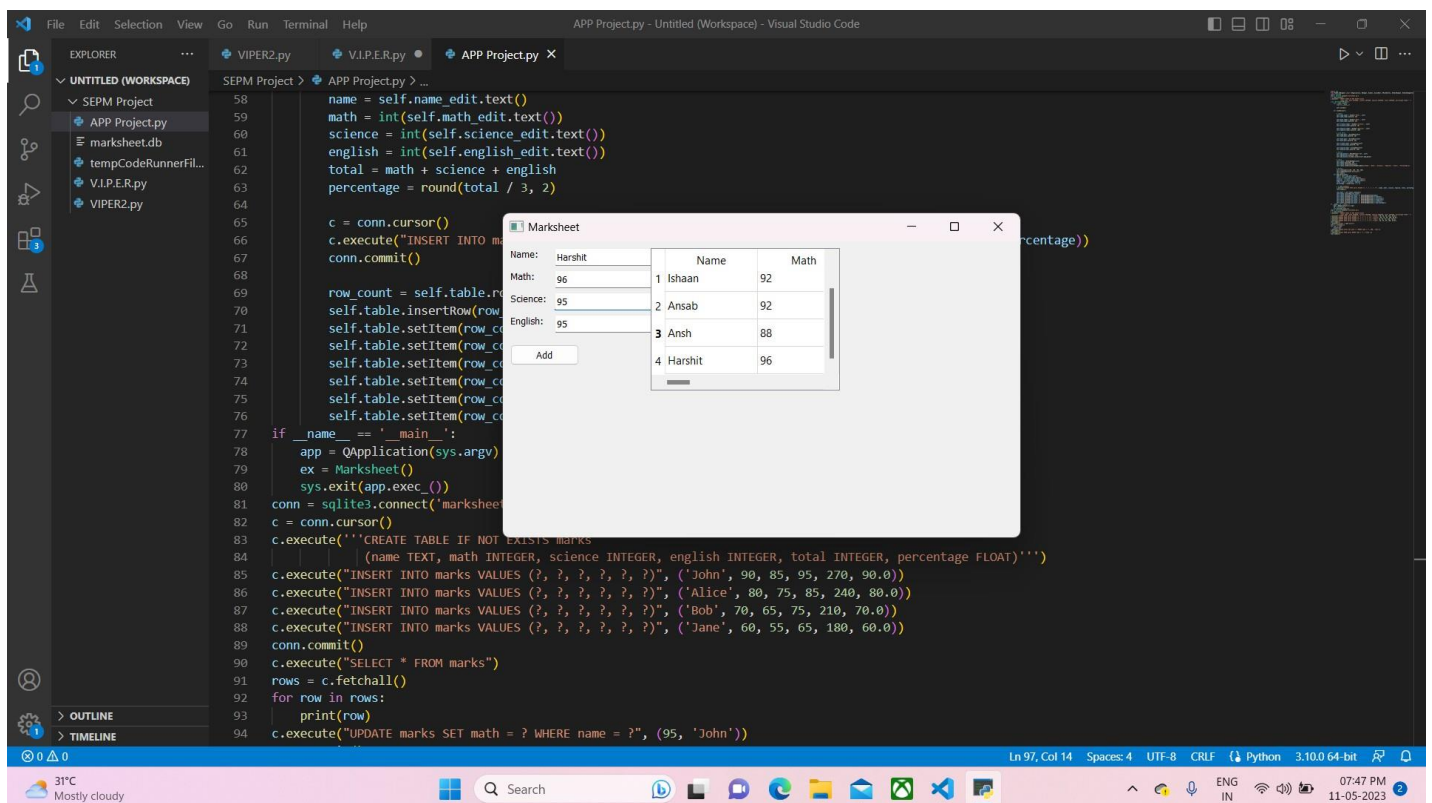
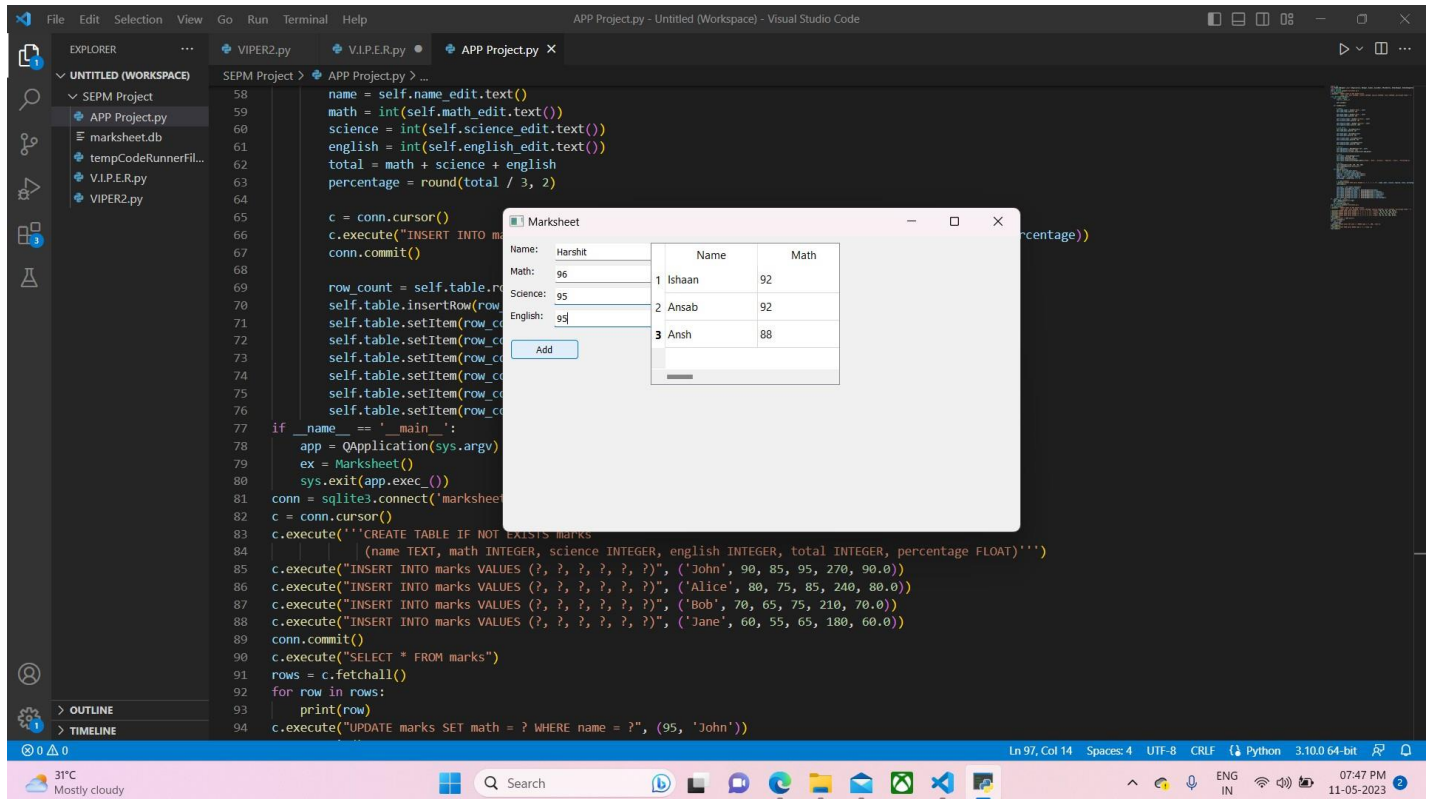
The screenshot displays the Visual Studio Code editor with the Python code for the Marksheet application. The code includes database setup, data insertion, and a main loop for adding and updating records. A screenshot of the application window is overlaid on the code.

Marksheet Application Window:

The application window has a title bar "Marsheet". It contains a form with input fields for "Name", "Math", "Science", and "English", and a button labeled "Add". Below the form is a table with two columns: "Name" and "Math".

	Name	Math
1	Ishaan	92
2	Ansab	92
3	Ansh	88

We have the given table where we added the details of students in the marksheet already. Now we will enter the details of another student named Harshit. To do so, we will give his details in columns on the left and click on “Add” button.



CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, we have developed a GUI application using Python and SQLite to manage student marks. The application allows the user to enter student details and marks in different subjects, calculates the total and percentage, and stores the data in an SQLite database. The application is user-friendly, easy to use, and provides an efficient way to manage and track student performance.

The use of SQLite in this project allowed for the creation of a simple, lightweight database that could be easily integrated into the application. SQLite is a robust and reliable database engine that is easy to set up and use, making it an excellent choice for small-scale applications like this.

Future enhancements to this project could include adding more features such as viewing, updating, and deleting student records, generating reports, and graphs, adding validation to ensure that the data entered is valid, and improving the user interface. Additionally, we could integrate this application with other systems such as an SMS or email notification system to notify parents or guardians of their children's performance.

Overall, this project demonstrates how Python and SQLite can be used together to create a simple yet effective database management system. With further development and improvements, this application could be a valuable tool for teachers, administrators, and parents to manage and track student performance.