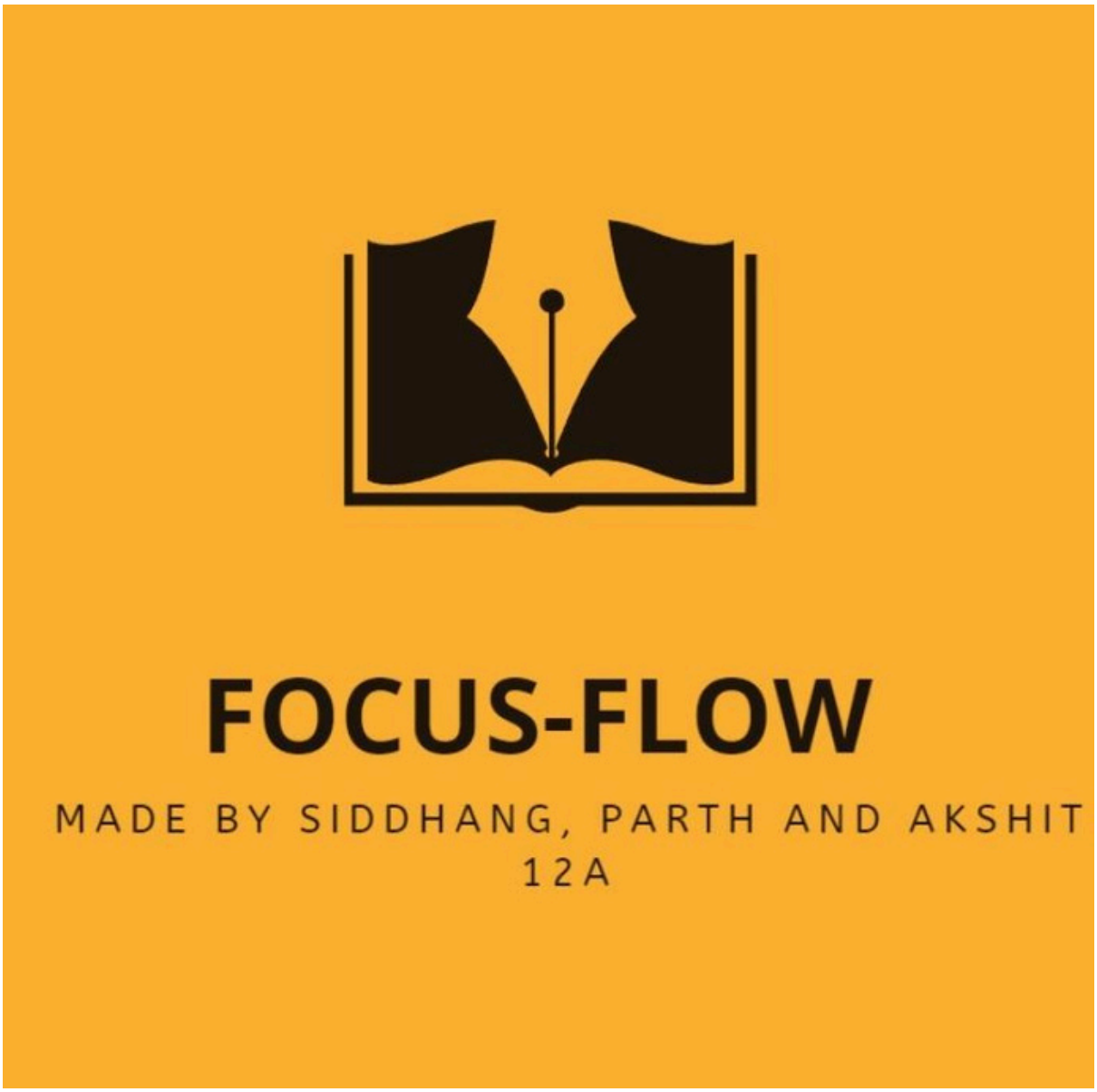
FOCUS-FLOW: TASK MANAGEMENT

COMPUTER SCIENCE PROJECT 2024-25



STUDENT NAME: AKSHIT BUTTA

CONTENTS

BONAFIDE CERTIFICATE

This is to certify that Mr/Ms _____ has successfully completed his/her project on the topic _____ in _____ and submitted for **All India Senior Secondary Practical Examination** held for the academic year 2024-25.

School : **THE SAMHITA ACADEMY, BANGALORE**

Class :XII

Roll No : _____

Date of Submission : _____

Internal Examiner External Examiner

Principal Seal

ACKNOWLEDGMENT

I am incredibly grateful to my teammates Parth Pandey and Siddhang Anil Kumar for their diligent involvement and valuable suggestions, which helped me complete the project work on time.

I sincerely thank my Computer Science teacher Mrs Neeti Goyal for her constant guidance, clear instructions, and support in completing my project and documentation.

I would also like to thank the principal, vice principal, and non-teaching staff of the computer department and other departments who helped me directly or indirectly execute this project.

Finally, yet importantly, I would like to express my heartfelt thanks to my parents for their support, my friends/classmates for their help, and wishes for the successful completion of this project.

_____

PURPOSE OF PROJECT

The purpose of this project is to provide students with a comprehensive and user-friendly task management solution tailored to meet the unique demands of both academic and non-academic responsibilities. Recognizing the challenges students face in balancing their studies, extracurricular activities, and personal commitments, this application offers an intuitive platform to plan, organize, and track their tasks effectively.

The application enables students to categorize tasks by subject, ensuring that academic tasks like assignments, exams, and study goals are well-organized and accessible. It also accommodates non-academic responsibilities, such as extracurricular activities, tutoring, or personal errands, offering a holistic approach to time management. With features like calendar integration, students can visualize their deadlines, manage their schedules, and focus on the tasks that matter most.

The project emphasizes usability by including functionalities such as creating and managing user accounts, setting task priorities, adding notes for detailed descriptions, and marking tasks as complete once finished. Overdue tasks are flagged for attention, and the system also tracks completed tasks, giving students a sense of accomplishment and progress. The color-coded subject tabs provide a visually appealing way to differentiate between academic disciplines and simplify task organization.

This task manager goes beyond traditional planners by providing students with tools to monitor their progress and make better use of their time. Its clean design, robust database integration, and features like filtering tasks by status (pending, completed, or overdue) make it an invaluable resource for students aiming to achieve academic success while maintaining a balanced lifestyle. By streamlining task tracking and prioritization, the project ultimately fosters better time management, increased productivity, and reduced stress for students navigating their busy lives.

RESOURCE REQUIREMENT

System requirement:

- MinimumRAMSize: 4MB
- MinimumHardDriveSize: 25MB
- OSRequired: Windows (64 Bit)
- MinimumProcessorType: Intel 386 or higher
- Platform: Any Python Editor
- ProgrammingLanguage: Python
- Database: SQL

PROJECT DESIGN

Important functions, classes and modules used in Project:

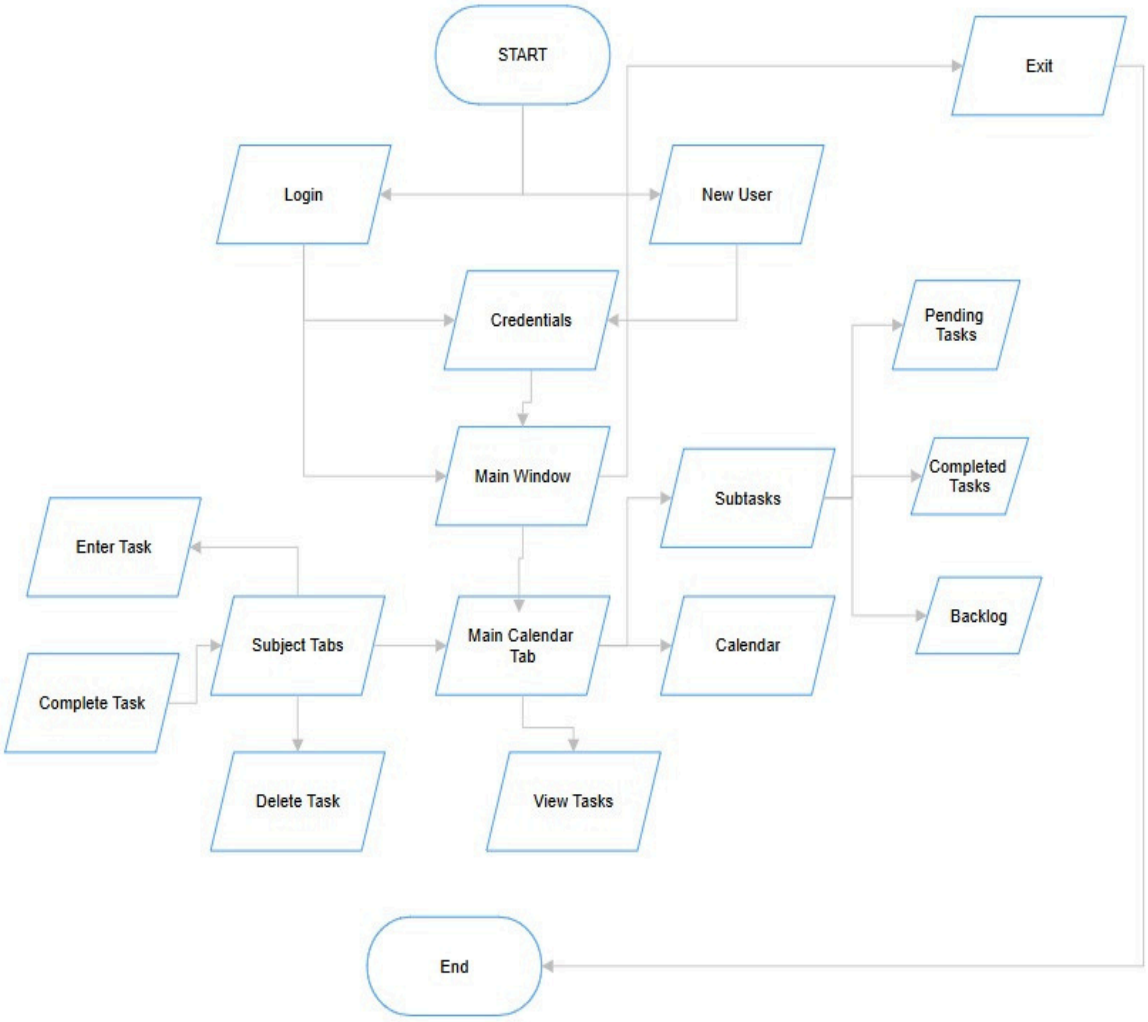
Module	ObjectName
PyQt5	**Classes:**QMainWindow,QDialog,Qwidget, QCalendarWidget, QPushButton, QLineEdit, QListWidget, QVBoxlayout(), QTextEdit(), QMessageBox(), QTabWidget(),
mysql.connector	**Functions:**connect(),cursor(),execute(), fetchall(),fetchone(), isconnected()
sys	**Functions:**exit(),execetc
Frontend	**Note:**Customloadingscreenmadein QDesginer

Note: A large amount of other Classes, Functions and Variables were also used, however in this report only the important/frequently used ones are mentioned

User Defined functions used in Project

FunctionName	FunctionPurpose
authenticate_userand create_new_user	Usedtocreateanewaccount(front-end)and toverifyaccountdetailsatthetimeoflogin (back-end)
MainWindow,LoginWindow, NewUserDialog,MainCalendar, SubjectTaskManagement	Corefront-endelementsoftheprogram,all ofthesehavetheirownscreen
load_tasks_for_selected_date, load_pending_tasks, load_completed_tasks, load_backlog_tasks, add_task, remove_task, insert_task_into_db, load_tasks, complete_task	Backendfunctionswhichinteractwiththesql database

FLOWCHART


START
Exit
Login
New User
Credentials
Pending Tasks
Main Window
Subtasks
Completed Tasks
Enter Task
Backlog
Subject Tabs
Main Calendar Tab
Calendar
Complete Task
Delete Task
View Tasks
End

SOURCE CODE

```
import sys
import mysql.connector
from PyQt5.QtWidgets import QApplication, QMainWindow, QVBoxLayout,
QLineEdit, QTextEdit, QPushButton, QListWidget, QWidget, QTabWidget,
QMessageBox, QCalendarWidget, QFormLayout, QDialog
from PyQt5.QtCore import QDate
from PyQt5.QtWidgets import QApplication
import frontend
#app = QApplication([])

'''
loading_screen = LoadingScreen() #check if working in school
loading_screen.show()'''

try:
   conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="12345",
          )
   cur=conn.cursor()
   cur.execute("CREATE DATABASE IF NOT EXISTS studyscheduler")
   print("Database STUDYSCHEDULER has been created successfully")
   cur.execute("USEstudyscheduler")
   cur.execute("""
            CREATE TABLE IF NOT EXISTS users (
               user_id INT PRIMARY KEY AUTO_INCREMENT,
               username VARCHAR(255) NOT NULL UNIQUE,
               password VARCHAR(255) NOT NULL
            )
            """)
   print("Table USERS created successfully")
   cur.execute("""
            CREATE TABLE IF NOT EXISTS tasks (
               task_id INT PRIMARY KEY AUTO_INCREMENT,
               subject VARCHAR(255) NOT NULL,
               task_description TEXT NOT NULL,
```

```
            note TEXT,
            due_dateDATE,
            user_id INT,
            iscompleteTINYINT(1)DEFAULT0,
            FOREIGN KEY (user_id) REFERENCES users(user_id)
        )
        """)
    print("Table tasks created successfully")
except mysql.connector.Error as e:
    print("Error:",e)



# Database Authentication
def authenticate_user(username, password):
    try:
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="12345",
            database="studyscheduler"
        )
        cur = conn.cursor()
        cur.execute("SELECT * FROM users WHERE username=%s AND
password=%s", (username, password))
        user = cur.fetchone()
        conn.close()

        if user:
            return user[0] # Return user id
        else:
            return None
    except mysql.connector.Error as err:
        QMessageBox.critical(None, "Database Error", f"Error: {err}")
        return None

# Function to create a new user in the database
def create_new_user(username, password):
    if username and password:
        try:
```

```
            conn = mysql.connector.connect(
                host="localhost",
                user="root",
                password="12345",
                database="studyscheduler"
            )
            cur = conn.cursor()
            cur.execute("INSERT INTO users (username, password) VALUES (%s, %s)",
(username, password))
            conn.commit()
            conn.close()
            QMessageBox.information(None, "Success", "New user created
successfully!")
        except mysql.connector.Error as err:
            QMessageBox.critical(None, "Database Error", f"Error: {err}")
    else:
        QMessageBox.warning(None, "Missing Information", "Please fill all fields.")

# MainWindow for the Application
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Study  Planner")
        self.setGeometry(100, 100, 800, 600)

        self.login_ui()

    def login_ui(self):
        self.login_window = LoginWindow()
        self.login_window.show()

# Login Window to authenticate the user
class LoginWindow(QDialog):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Login")
        self.setGeometry(500, 300, 400, 200)

        layout = QFormLayout()
        self.username_input = QLineEdit(self)
```

```
        self.password_input = QLineEdit(self)
        self.password_input.setEchoMode(QLineEdit.Password)

        self.login_button = QPushButton("Login", self)
        self.login_button.clicked.connect(self.authenticate)

        # New User Button
        self.new_user_button = QPushButton("New User", self)
        self.new_user_button.clicked.connect(self.open_new_user_dialog)

        layout.addRow("Username:", self.username_input)
        layout.addRow("Password:",self.password_input)
        layout.addWidget(self.login_button)
        layout.addWidget(self.new_user_button)

        self.setLayout(layout)

    def authenticate(self):
        username = self.username_input.text()
        password = self.password_input.text()

        user_id = authenticate_user(username, password)

        if user_id:
            self.accept() #Closetheloginwindowandopenthemainwindow
            self.main_window = MainApp(user_id)
            self.main_window.showMaximized()
        else:
            QMessageBox.warning(self, "Authentication Failed", "Invalid username
or password.")

    def open_new_user_dialog(self):
        dialog = NewUserDialog()
        dialog.exec_()

# Dialog for new user registration
class NewUserDialog(QDialog):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Create New User")
```

```
        self.setGeometry(500, 300, 400, 200)

        layout = QFormLayout()
        self.username_input = QLineEdit(self)
        self.password_input = QLineEdit(self)
        self.password_input.setEchoMode(QLineEdit.Password)

        self.create_button = QPushButton("Create", self)
        self.create_button.clicked.connect(self.create_user)

        layout.addRow("New Username:", self.username_input)
        layout.addRow("New Password:", self.password_input)
        layout.addWidget(self.create_button)

        self.setLayout(layout)

    def create_user(self):
        username = self.username_input.text()
        password = self.password_input.text()
        create_new_user(username, password)
        self.accept() #Closethedialog

# Main App for managing tasks and calendars
class MainApp(QMainWindow):
    def __init__(self, user_id):
        super().__init__()
        self.setWindowTitle("Study Planner")
        self.setGeometry(100, 100, 1000, 600)
        self.user_id = user_id

        self.tabs = QTabWidget(self)
        self.setCentralWidget(self.tabs)

        self.tabs.addTab(MainCalendar(self.user_id),"MainCalendar")
        self.tabs.addTab(SubjectTaskManagement(self.user_id,"Physics"),
"Physics")
        self.tabs.addTab(SubjectTaskManagement(self.user_id,"Chemistry"),
"Chemistry")
        self.tabs.addTab(SubjectTaskManagement(self.user_id,"Math"),"Math")
```

```
        self.tabs.addTab(SubjectTaskManagement(self.user_id,"Computer
Science"), "Computer Science")
        self.tabs.addTab(SubjectTaskManagement(self.user_id,"English"),
"English")
        self.tabs.addTab(SubjectTaskManagement(self.user_id,"Tutions"),
"Tutions")
        self.tabs.addTab(SubjectTaskManagement(self.user_id,"Other"),"Other")

        self.showMaximized()

# Calendar for the main page
# Calendar for the main page
# Update the MainCalendar class to include the tab widget with "Pending
Tasks", "Completed Tasks", and "Backlog" tabs.
class MainCalendar(QWidget):
def __init__(self, user_id):
        super().__init__()
        self.user_id = user_id

        layout = QVBoxLayout()

        # Main Tab Widget for Pending, Completed, and Backlog Tasks
        self.tab_widget = QTabWidget(self)

        # Pending Tasks Tab
        self.pending_tasks_tab = QWidget()
        self.pending_tasks_list = QListWidget(self.pending_tasks_tab)
        self.load_pending_tasks()
        pending_layout = QVBoxLayout()
        pending_layout.addWidget(self.pending_tasks_list)
        self.pending_tasks_tab.setLayout(pending_layout)
        self.tab_widget.addTab(self.pending_tasks_tab, "Pending Tasks")

        # Completed Tasks Tab
        self.completed_tasks_tab = QWidget()
        self.completed_tasks_list = QListWidget(self.completed_tasks_tab)
        self.load_completed_tasks()
        completed_layout = QVBoxLayout()
        completed_layout.addWidget(self.completed_tasks_list)
        self.completed_tasks_tab.setLayout(completed_layout)
```

```
        self.tab_widget.addTab(self.completed_tasks_tab,"CompletedTasks")

        # Backlog Tab
        self.backlog_tab = QWidget()
        self.backlog_list = QListWidget(self.backlog_tab)
        self.load_backlog_tasks()
        backlog_layout = QVBoxLayout()
        backlog_layout.addWidget(self.backlog_list)
        self.backlog_tab.setLayout(backlog_layout)
        self.tab_widget.addTab(self.backlog_tab,"Backlog")

        # Add the tab widget to the main layout
        layout.addWidget(self.tab_widget)

        # Calendar widget
        self.calendar = QCalendarWidget(self)
        self.calendar.setGridVisible(True)
        self.calendar.selectionChanged.connect(self.load_tasks_for_selected_date)

self.calendar.setVerticalHeaderFormat(QCalendarWidget.NoVerticalHeader)#re
moves week number
        layout.addWidget(self.calendar)

        # Task list below the calendar
        self.task_list = QListWidget(self)
        layout.addWidget(self.task_list)

        self.setLayout(layout)

        # Load tasks for today's date by default
        self.load_tasks_for_selected_date()

    def load_tasks_for_selected_date(self):
        """Load tasks for the date selected in the calendar, including notes."""
        selected_date = self.calendar.selectedDate().toString("yyyy-MM-dd")

        # Clear the task list
        self.task_list.clear()

        try:
```

```
            conn = mysql.connector.connect(
                host="localhost",
                user="root",
                password="12345",
                database="studyscheduler"
            )
            cur = conn.cursor()
            cur.execute("""
                SELECT subject, task_description, note FROM tasks
                WHERE user_id=%s AND due_date=%s
            """, (self.user_id, selected_date))
            tasks = cur.fetchall()
            conn.close()

            # Display each task in the task list with notes
            if tasks:
                for subject, task_description, note in tasks:
                    self.task_list.addItem(f"{subject}: {task_description}\nNote: {note}")
            else:
                self.task_list.addItem("No data for selected date")

        except mysql.connector.Error as err:
            QMessageBox.critical(None, "Database Error", f"Error: {err}")

    def load_pending_tasks(self):
        """Load tasks that are pending (not completed) and display them with the
days remaining."""
        self.pending_tasks_list.clear()
        try:
            conn = mysql.connector.connect(
                host="localhost",
                user="root",
                password="12345",
                database="studyscheduler"
            )
            cur = conn.cursor()
            cur.execute("""
                SELECT subject, task_description, due_date FROM tasks
                WHERE user_id=%s AND iscomplete = FALSE
            """, (self.user_id,))
```

```
            print('works')
            tasks = cur.fetchall()
            print(tasks)
            conn.close()

            today = QDate.currentDate()
            for subject, task_description, due_date in tasks:
                due_date_obj = QDate.fromString(due_date.strftime("%Y-%m-%d"),
"yyyy-MM-dd")

                days_left = today.daysTo(due_date_obj)
                self.pending_tasks_list.addItem(f"{subject}: {task_description} - Due in
{days_left} days ({due_date})")

        except mysql.connector.Error as err:
            QMessageBox.critical(None, "Database Error", f"Error: {err}")

    def load_completed_tasks(self):
        """Load tasks that have been marked as completed."""
        self.completed_tasks_list.clear()
        try:
            conn = mysql.connector.connect(
                host="localhost",
                user="root",
                password="12345",
                database="studyscheduler"
            )
            cur = conn.cursor()
            cur.execute("""
                SELECT subject, task_description, due_date FROM tasks
                WHERE user_id=%s AND iscomplete= TRUE
            """, (self.user_id,))
            tasks = cur.fetchall()
            conn.close()

            for subject, task_description, due_date in tasks:
                self.completed_tasks_list.addItem(f"{subject}: {task_description} -
Completed on {due_date}")

        except mysql.connector.Error as err:
```

```
            QMessageBox.critical(None, "Database Error", f"Error: {err}")

    def load_backlog_tasks(self):
        """Load tasks that are overdue and not completed, displaying them with
'NOT DONE!'."""
        self.backlog_list.clear()
        try:
            conn = mysql.connector.connect(
                host="localhost",
                user="root",
                password="12345",
                database="studyscheduler"
            )
            cur = conn.cursor()
            cur.execute("""
                SELECT subject, task_description, due_date FROM tasks
                WHERE user_id=%s AND iscomplete= FALSE AND due_date <
CURDATE()
            """, (self.user_id,))
            tasks = cur.fetchall()
            conn.close()

            for subject, task_description, due_date in tasks:
                self.backlog_list.addItem(f"{subject}: {task_description} - Due on
{due_date} - NOT DONE!")

        except mysql.connector.Error as err:
            QMessageBox.critical(None, "Database Error", f"Error: {err}")


# Task Management for each Subject
class SubjectTaskManagement(QWidget):
    def __init__(self, user_id, subject):
        super().__init__()
        self.user_id = user_id
        self.subject = subject
        self.init_ui()

    def init_ui(self):
        layout = QVBoxLayout()
```

```
        # Input fields for adding tasks
        self.task_input = QLineEdit(self)
        self.task_input.setPlaceholderText(f"Entertaskfor{self.subject}")

        self.note_input = QTextEdit(self)
        self.note_input.setPlaceholderText("Enter notes for the task")

        self.due_date_input = QLineEdit(self)
        self.due_date_input.setPlaceholderText("Enter due date (YYYY-MM-DD)")

        self.add_button = QPushButton(f"Add {self.subject} Task", self)
        self.add_button.clicked.connect(self.add_task)

        self.complete_button = QPushButton("Complete Task", self)
        self.complete_button.clicked.connect(self.complete_task)

        self.remove_button = QPushButton(f"Remove {self.subject} Task", self)
        self.remove_button.clicked.connect(self.remove_task)



        self.task_list = QListWidget(self)

        layout.addWidget(self.task_input)
        layout.addWidget(self.note_input)
        layout.addWidget(self.due_date_input)
        layout.addWidget(self.add_button)
        layout.addWidget(self.remove_button)
        layout.addWidget(self.task_list)
        layout.addWidget(self.complete_button)


        self.setLayout(layout)

        self.load_tasks()

    def add_task(self):
        task_description = self.task_input.text()
        note = self.note_input.toPlainText()
        due_date = self.due_date_input.text()
```

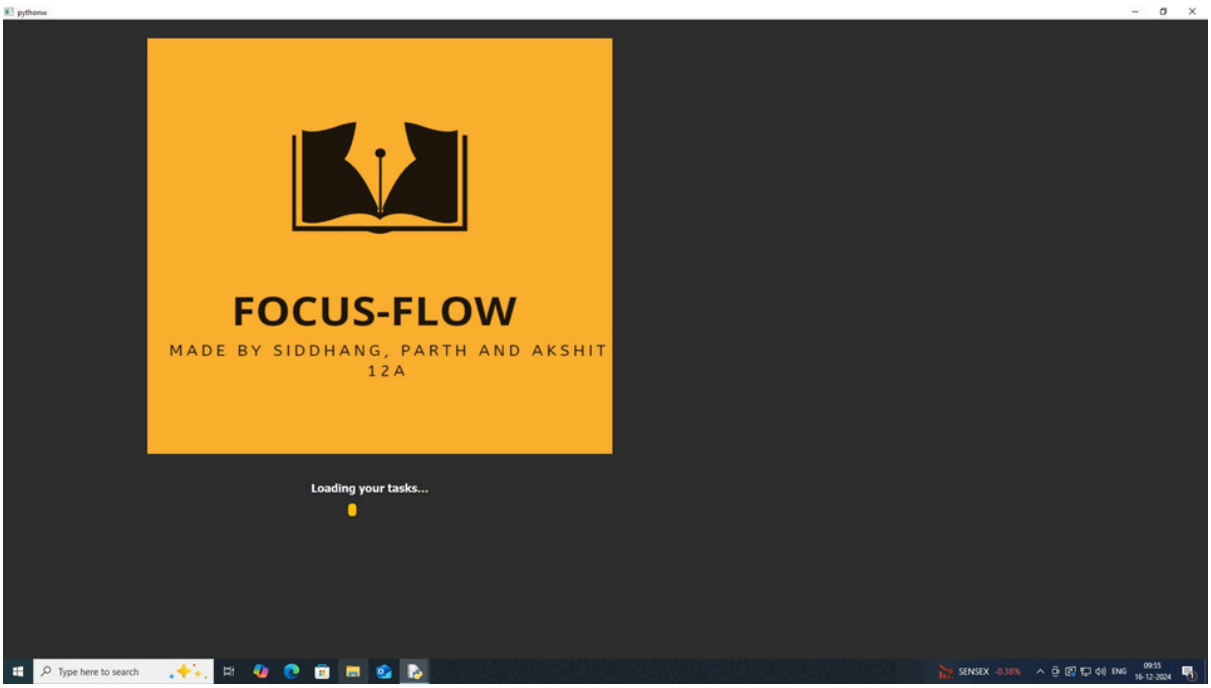
```
        if task_description and due_date:
            self.insert_task_into_db(task_description, note, due_date)
            self.load_tasks()
        else:
            QMessageBox.warning(self, "Missing Information", "Please fill all
fields.")

    def remove_task(self):
        selected_task = self.task_list.currentItem()
        if selected_task:
            task_description = selected_task.text()
            self.delete_task_from_db(task_description)
            self.load_tasks()
        else:
            QMessageBox.warning(self, "No Task Selected", "Please select a task to
remove.")

    def insert_task_into_db(self, task_description, note, due_date):
        try:
            conn = mysql.connector.connect(
                host="localhost",
                user="root",
                password="12345",
                database="studyscheduler"
            )
            cur = conn.cursor()
            cur.execute("""
                INSERT INTO tasks (subject, task_description, note, due_date, user_id)
                VALUES (%s, %s, %s, %s, %s)
            """, (self.subject, task_description, note, due_date, self.user_id))
            conn.commit()
            conn.close()
        except mysql.connector.Error as err:
            QMessageBox.critical(None, "Database Error", f"Error: {err}")

    def delete_task_from_db(self, task_description):
        try:
            conn = mysql.connector.connect(
                host="localhost",
```

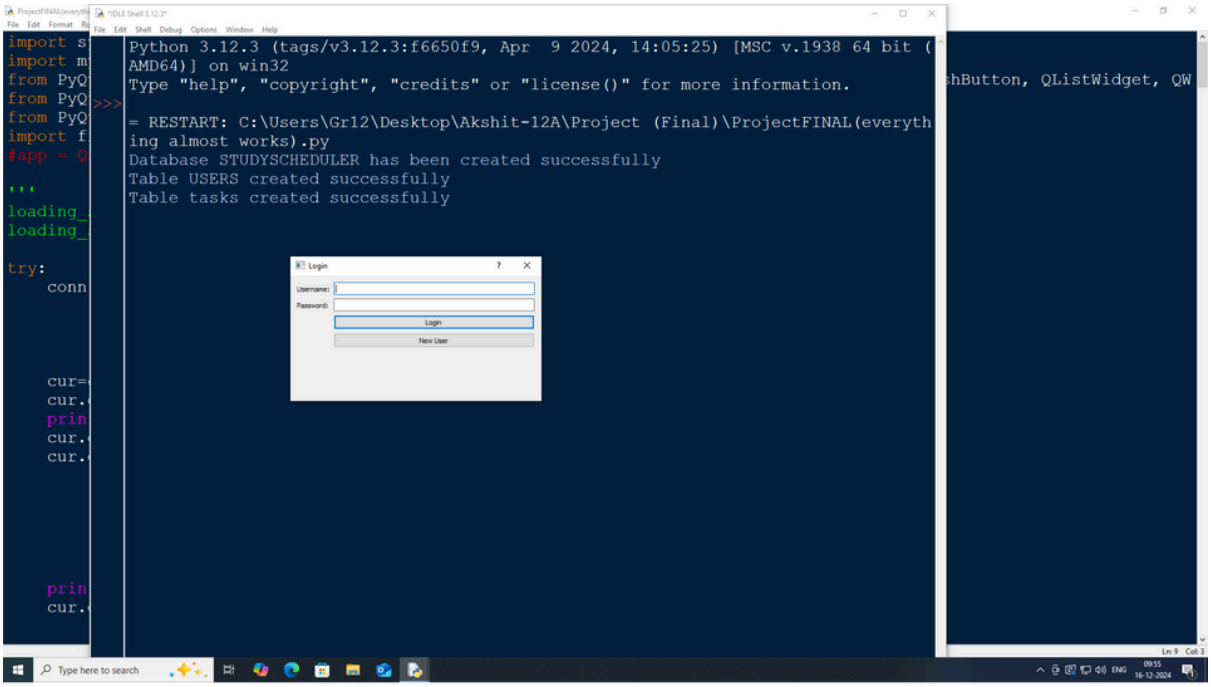
```
            user="root",
            password="12345",
            database="studyscheduler"
        )
        cur = conn.cursor()
        cur.execute("""
            DELETE FROM tasks WHERE subject=%s AND task_description=%s
AND user_id=%s
        """, (self.subject, task_description, self.user_id))
        conn.commit()
        conn.close()
    except mysql.connector.Error as err:
        QMessageBox.critical(None, "Database Error", f"Error: {err}")

def load_tasks(self):
    self.task_list.clear()
    try:
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="12345",
            database="studyscheduler"
        )
        cur = conn.cursor()
        cur.execute("""
            SELECT task_description FROM tasks
            WHERE subject=%s AND user_id=%s and iscomplete = FALSE
        """, (self.subject, self.user_id))
        tasks = cur.fetchall()
        conn.close()

        for task in tasks:
            self.task_list.addItem(task[0])

    except mysql.connector.Error as err:
        QMessageBox.critical(None, "Database Error", f"Error: {err}")

def complete_task(self):
# Get the selected task from the list
    selected_task = self.task_list.currentItem()
```

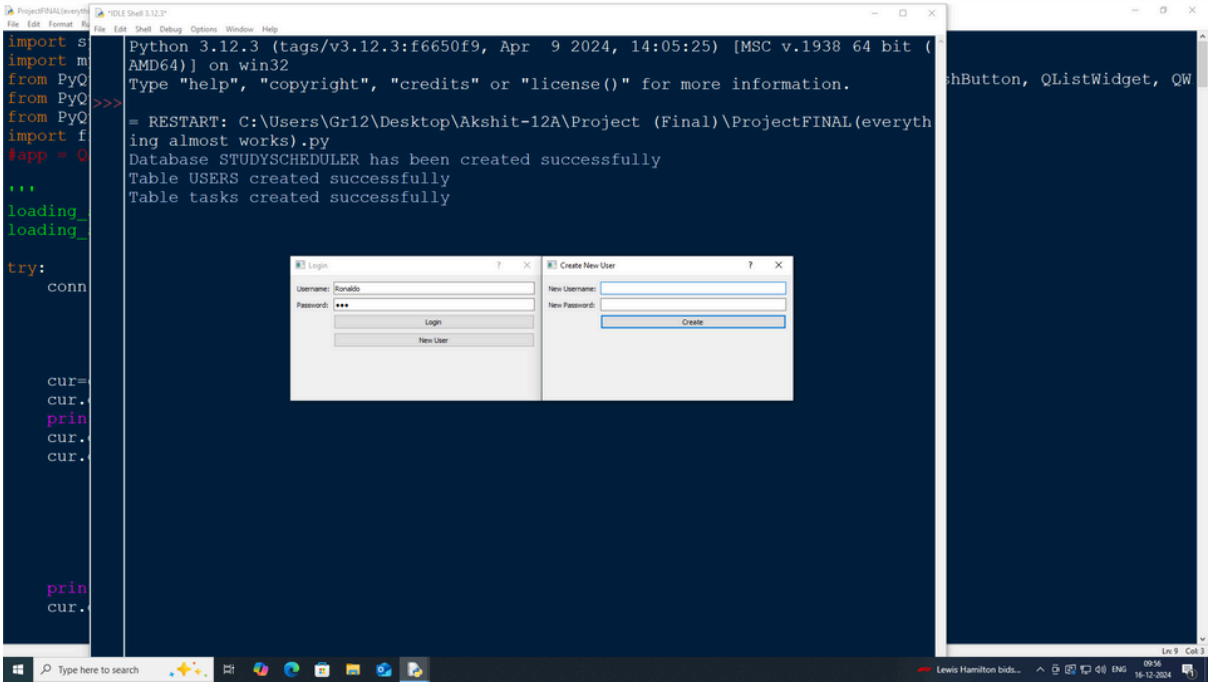
```
        if selected_task:
            description = selected_task.text() # Get the task description from the
selected item

            try:
                # Establish connection to the database
                conn = mysql.connector.connect(
                    host="localhost",
                    user="root",
                    password="12345",
                    database="studyscheduler"
                )
                cur = conn.cursor()

                # Update the task to mark it as complete
                cur.execute("""
                    UPDATE tasks
                    SET iscomplete = TRUE
                    WHERE subject = %s AND task_description = %s AND user_id = %s
AND iscomplete = FALSE
                """, (self.subject, description, self.user_id))



                # Check if any row was updated
                if cur.rowcount > 0:
                conn.commit()
                QMessageBox.information(None, "Task Completed", "The task has
been marked as complete. Please restart the app to see changes",)
                else:
                    QMessageBox.warning(None, "Task Not Found", "No incomplete
task matching the criteria was found.")

                conn.close()

            except mysql.connector.Error as err:
                # Handle database connection or query errors
                QMessageBox.critical(None, "Database Error", f"Error: {err}")
        else:
```

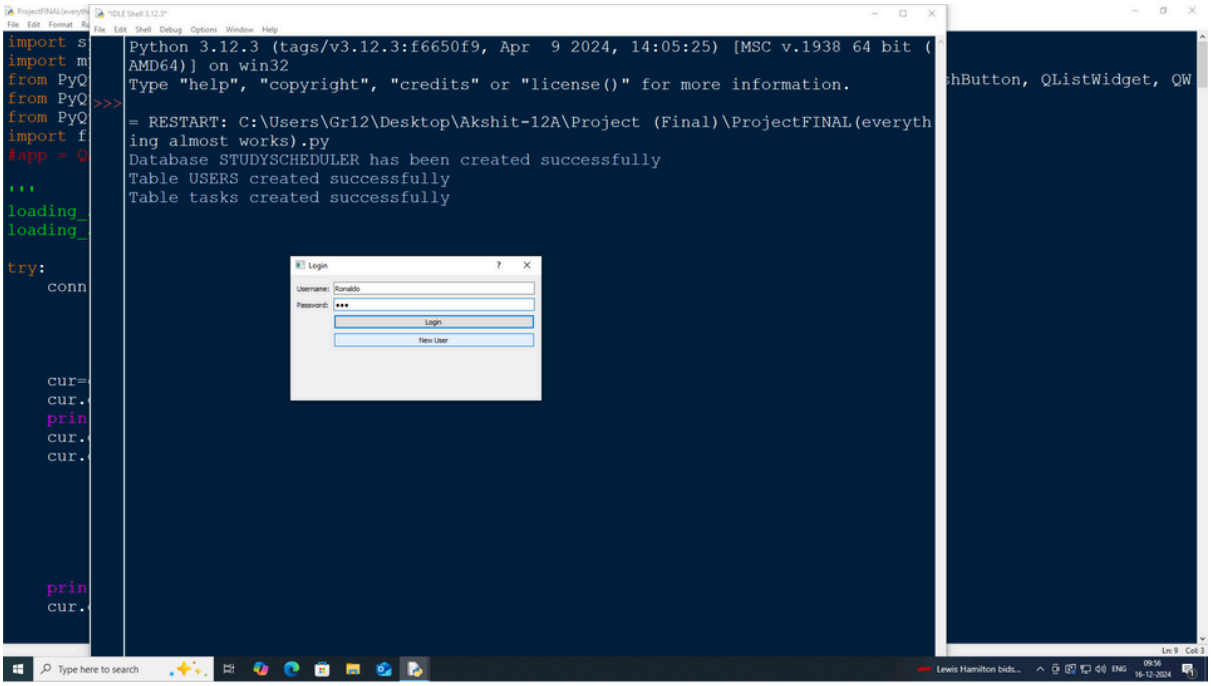
```
        QMessageBox.warning(None, "No Task Selected", "Please select a task
to mark as complete.")

# Main application execution
app = QApplication(sys.argv)
window = MainWindow()
#window.showMaximized()
sys.exit(app.exec_())
```

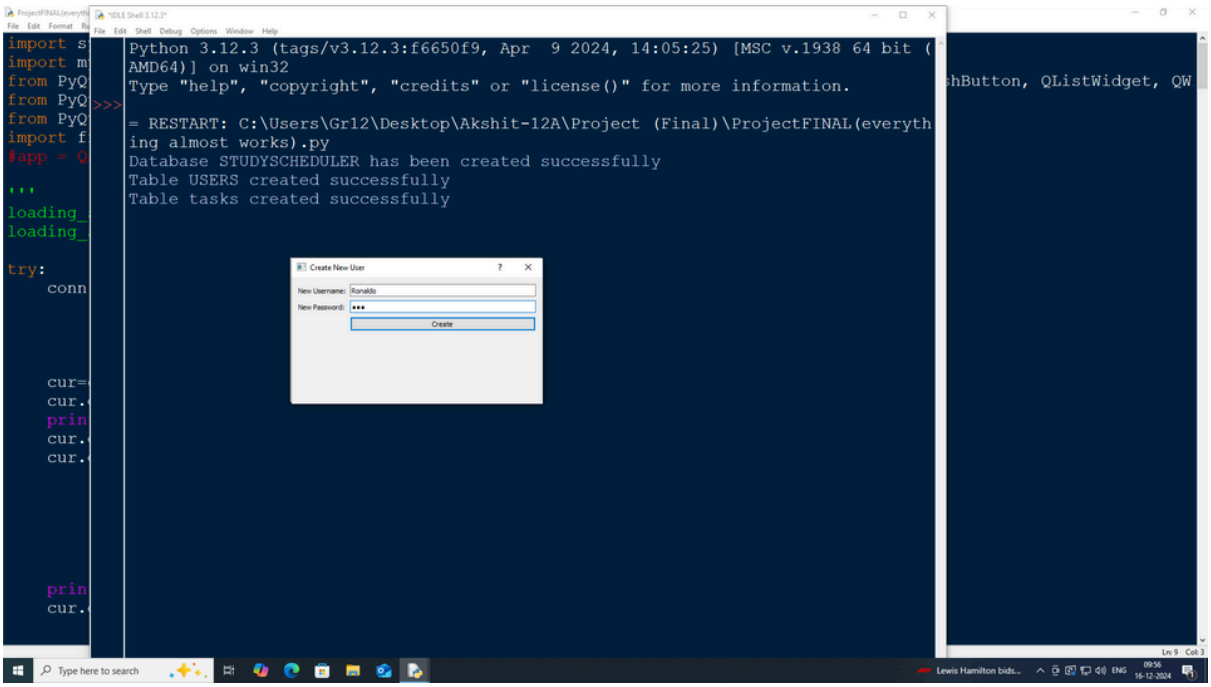
OUTPUT

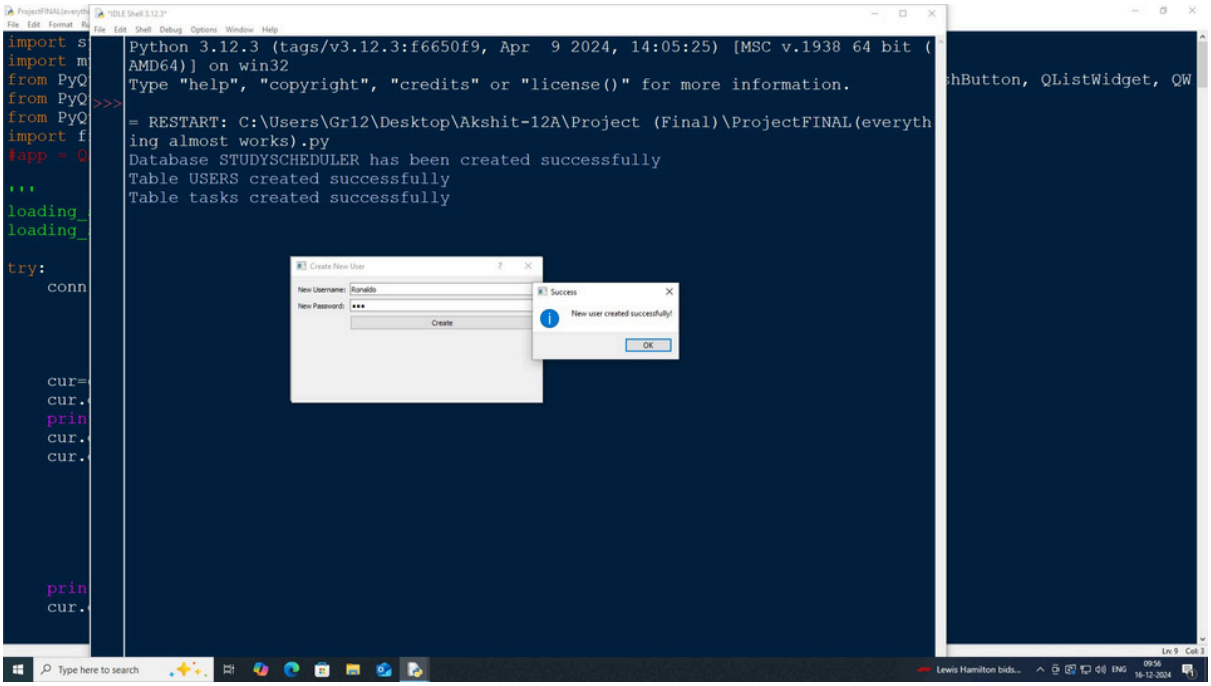

pythonw
FOCUS-FLOW
MADE BY SIDDHANG, PARTH AND AKSHIT
12A
Loading your tasks...
Type here to search
SENSEX -0.38%
ENG
09:55
16-12-2024

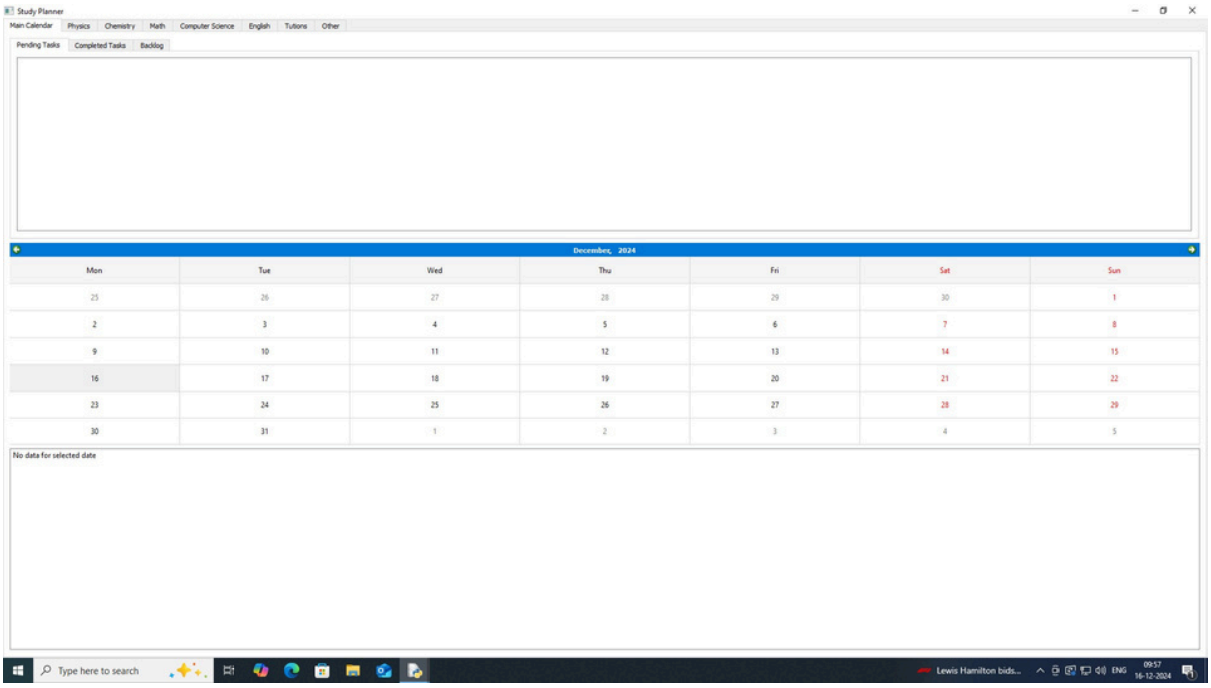

Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Gr12\Desktop\Akshit-12A\Project (Final)\ProjectFINAL(everything almost works).py
Database STUDYSCHEDULER has been created successfully
Table USERS created successfully
Table tasks created successfully
Login
Username:
Password:
Login
New User

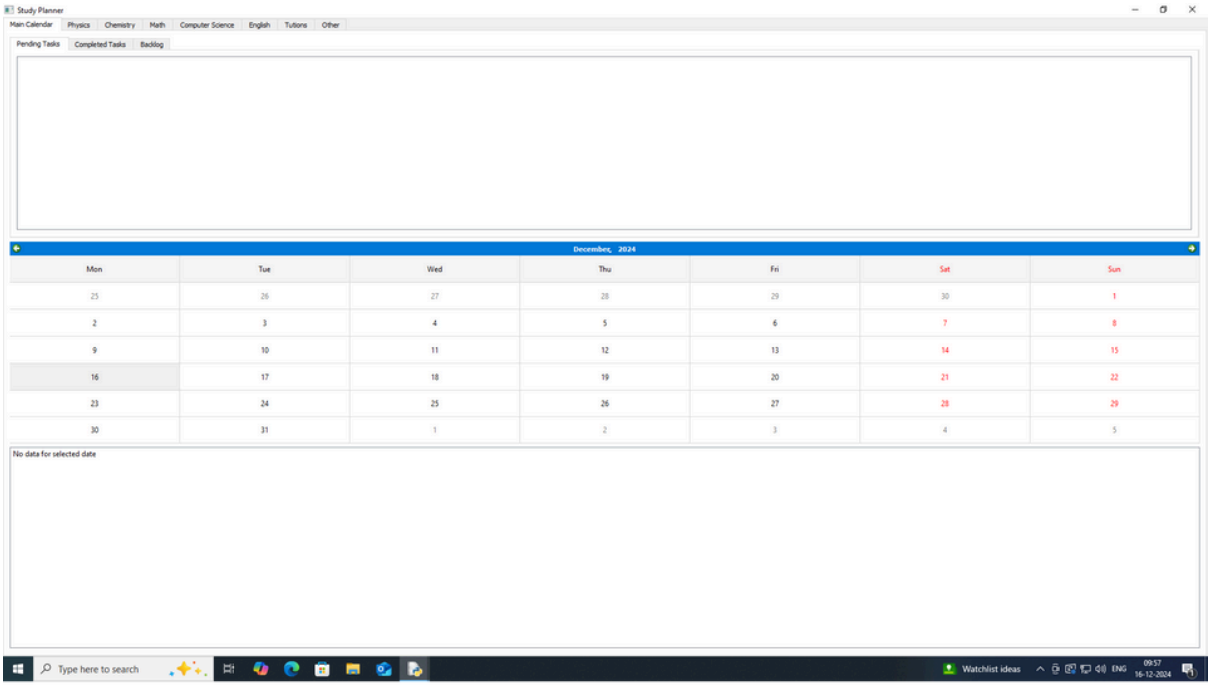

Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Gr12\Desktop\Akshit-12A\Project (Final)\ProjectFINAL(everything almost works).py
Database STUDYSCHEDULER has been created successfully
Table USERS created successfully
Table tasks created successfully
Login
Username: Ronaldo
Password:
Login
New User
Create New User
New Username:
New Password:
Create

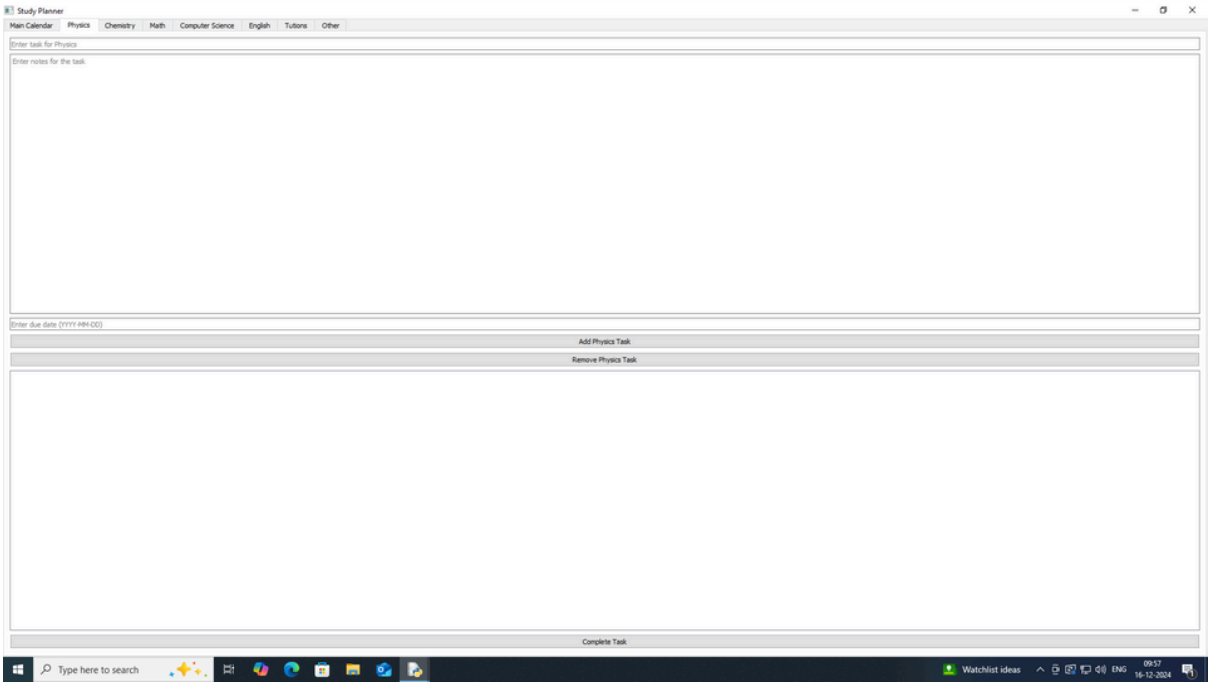

Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Gr12\Desktop\Akshit-12A\Project (Final)\ProjectFINAL(everyth
ing almost works).py
Database STUDYSCHEDULER has been created successfully
Table USERS created successfully
Table tasks created successfully
Login
Username:
Ronaldo
Password:
Login
New User

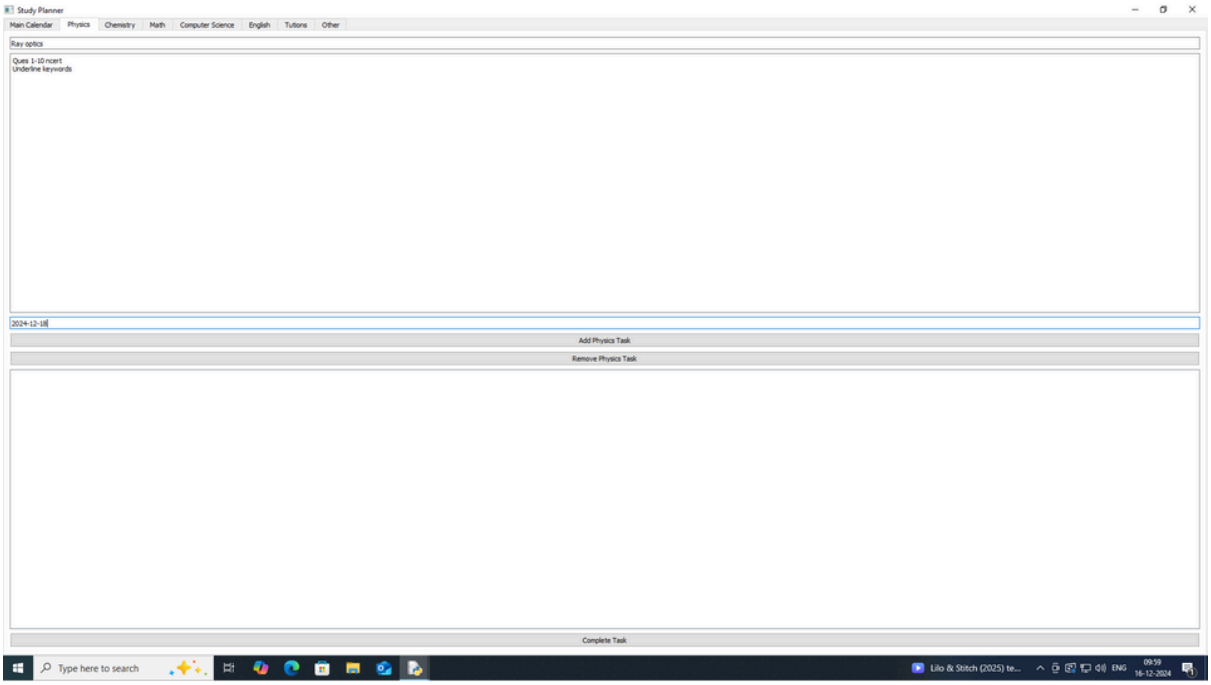

Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Gr12\Desktop\Akshit-12A\Project (Final)\ProjectFINAL(everyth
ing almost works).py
Database STUDYSCHEDULER has been created successfully
Table USERS created successfully
Table tasks created successfully
Create New User
New Username:
Ronaldo
New Password:
Create

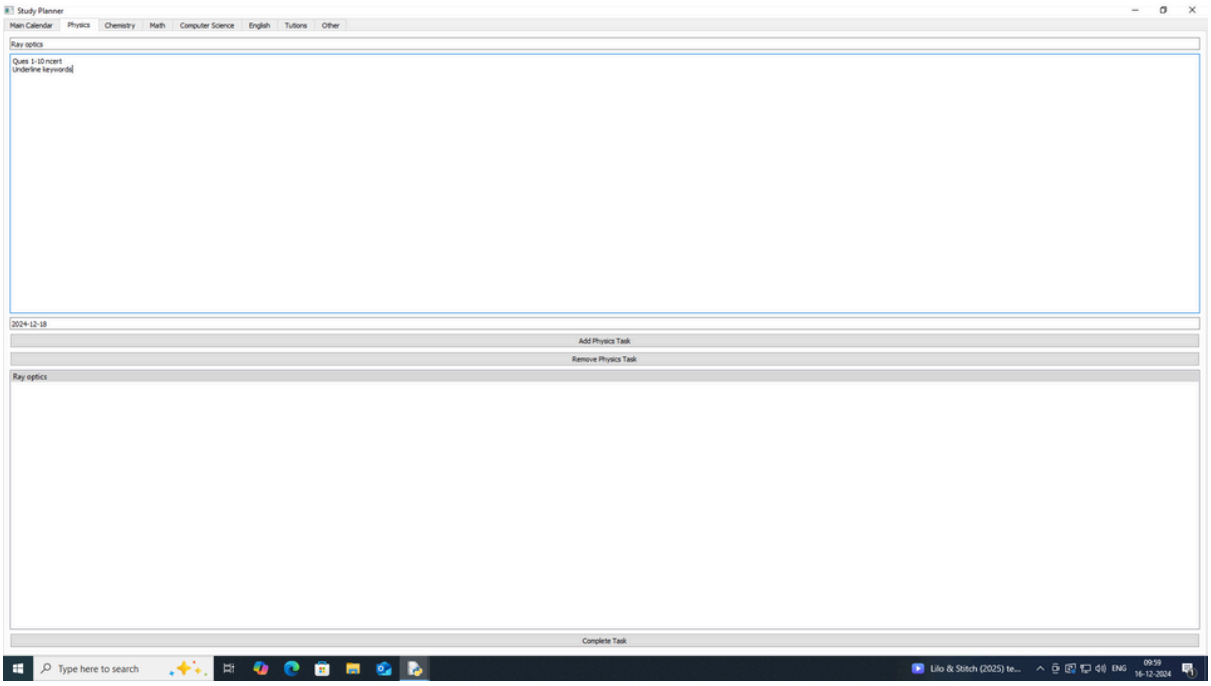

Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Gr12\Desktop\Akshit-12A\Project (Final)\ProjectFINAL(everyth
ing almost works).py
Database STUDYSCHEDULER has been created successfully
Table USERS created successfully
Table tasks created successfully
Create New User
New Username:
Ronaldo
New Password:
Create
Success
New user created successfully!
OK

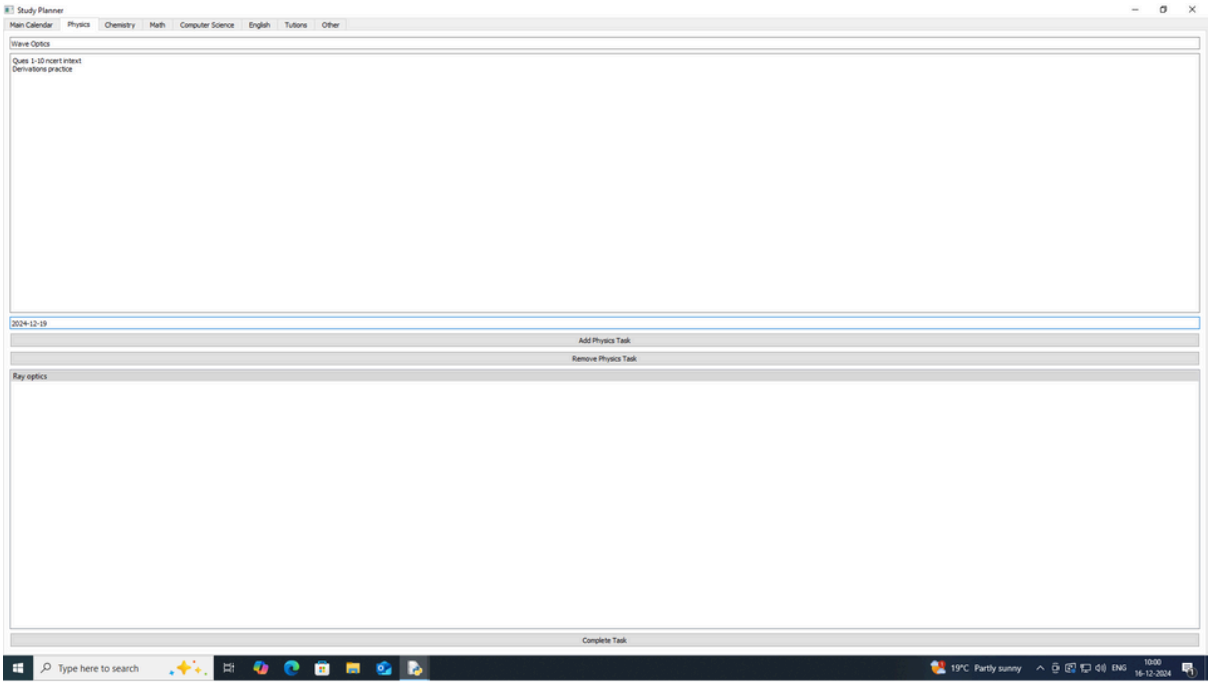

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Pending Tasks
Completed Tasks
Backlog
December, 2024
Mon
Tue
Wed
Thu
Fri
Sat
Sun
No data for selected date

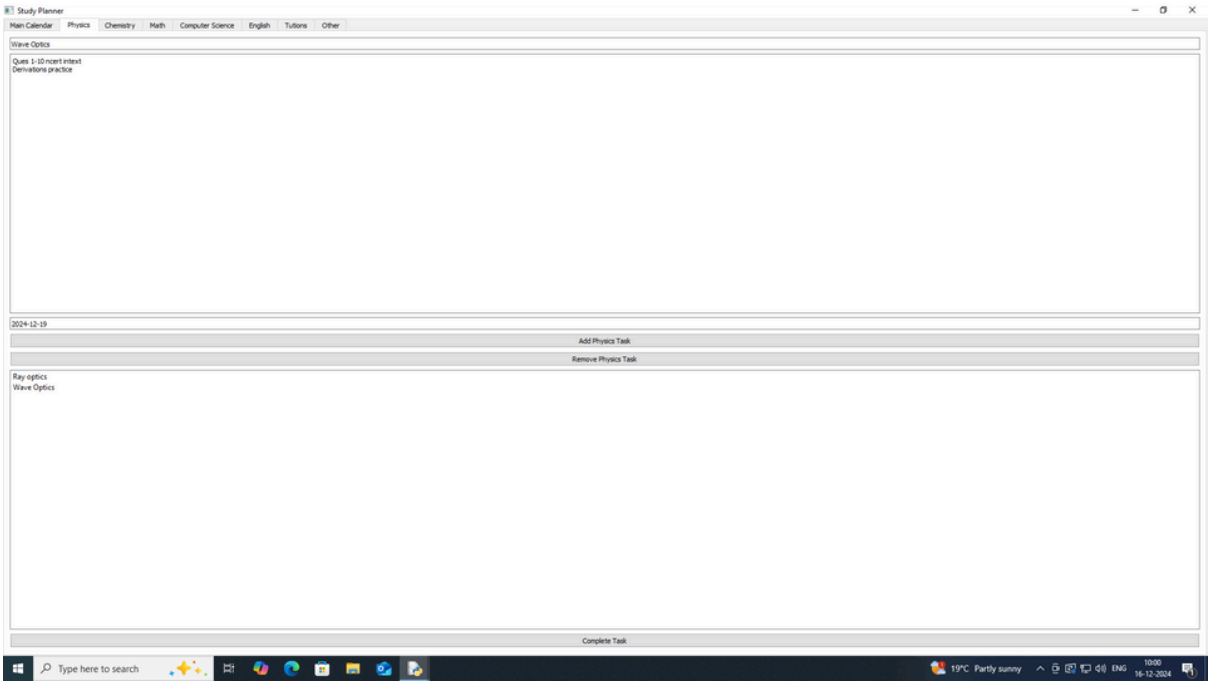

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Pending Tasks
Completed Tasks
Backlog
December, 2024
Mon
Tue
Wed
Thu
Fri
Sat
Sun
No data for selected date
Type here to search
Watchlist ideas
ENG
09:57
16-12-2024

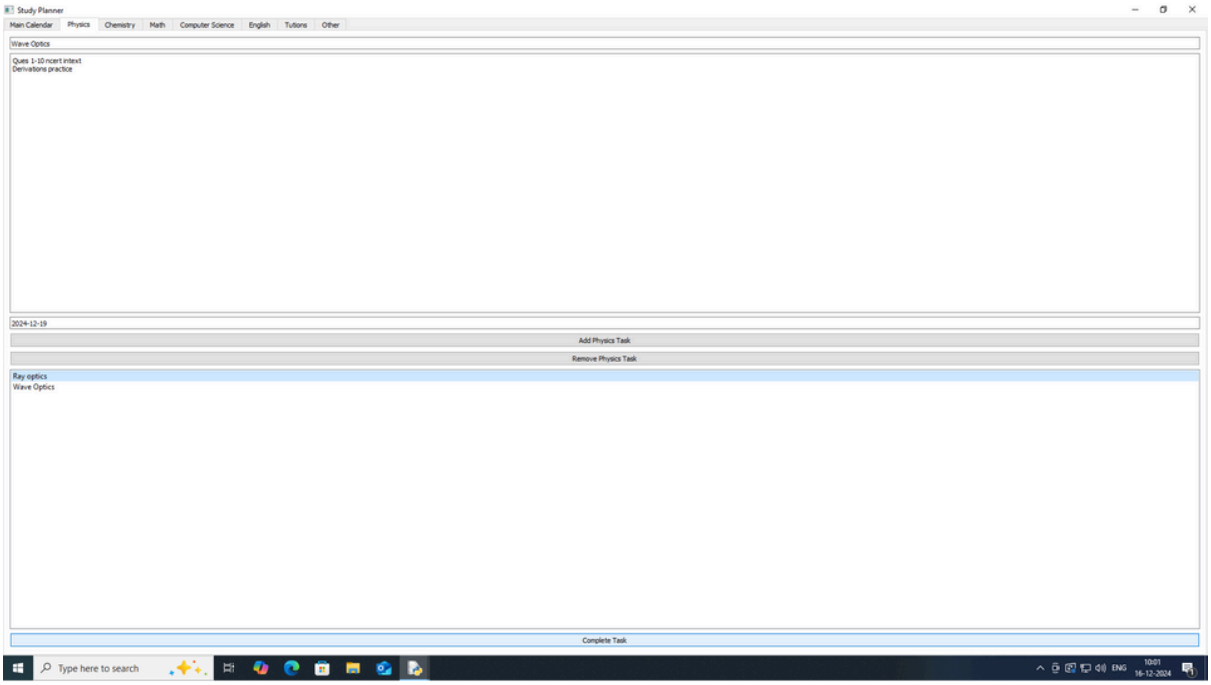

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Enter task for Physics
Enter notes for the task
Enter due date (YYYY-MM-DD)
Add Physics Task
Remove Physics Task
Complete Task
Type here to search
Watchlist ideas
ENG
09:57
16-12-2024

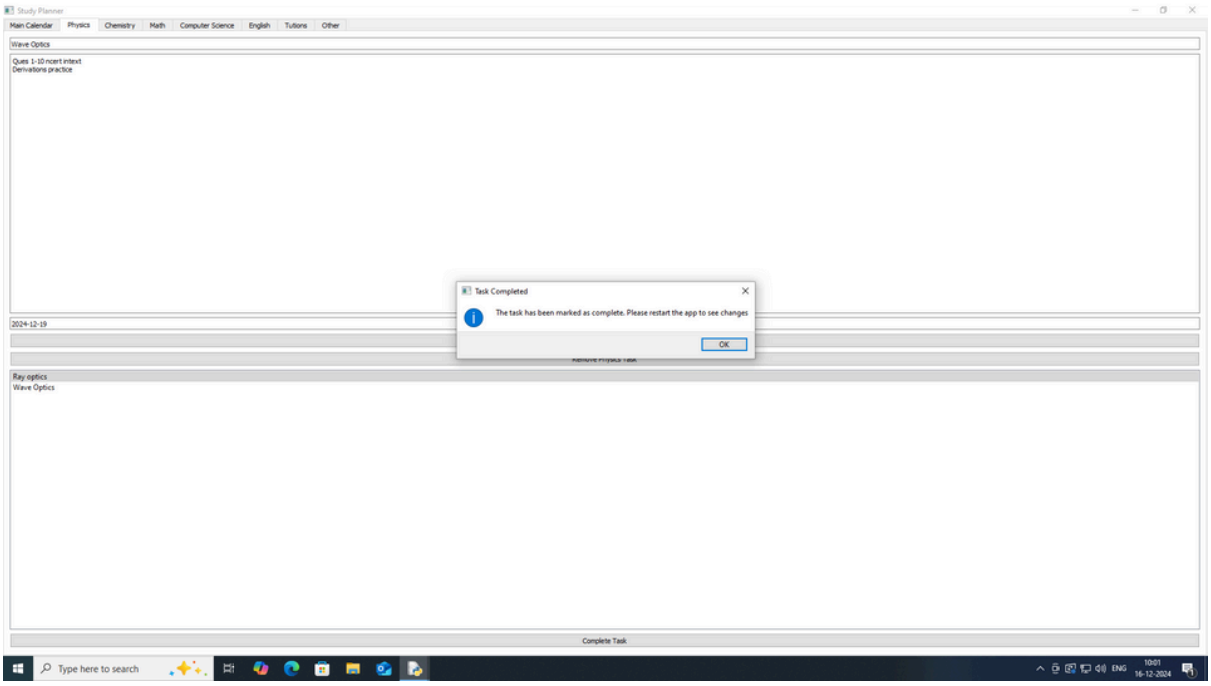

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Ray optics
Ques 1-10 ncert
Underline keywords
2024-12-18
Add Physics Task
Remove Physics Task
Complete Task
Type here to search
Lilo & Stitch (2025) te...
ENG
09:59
16-12-2024

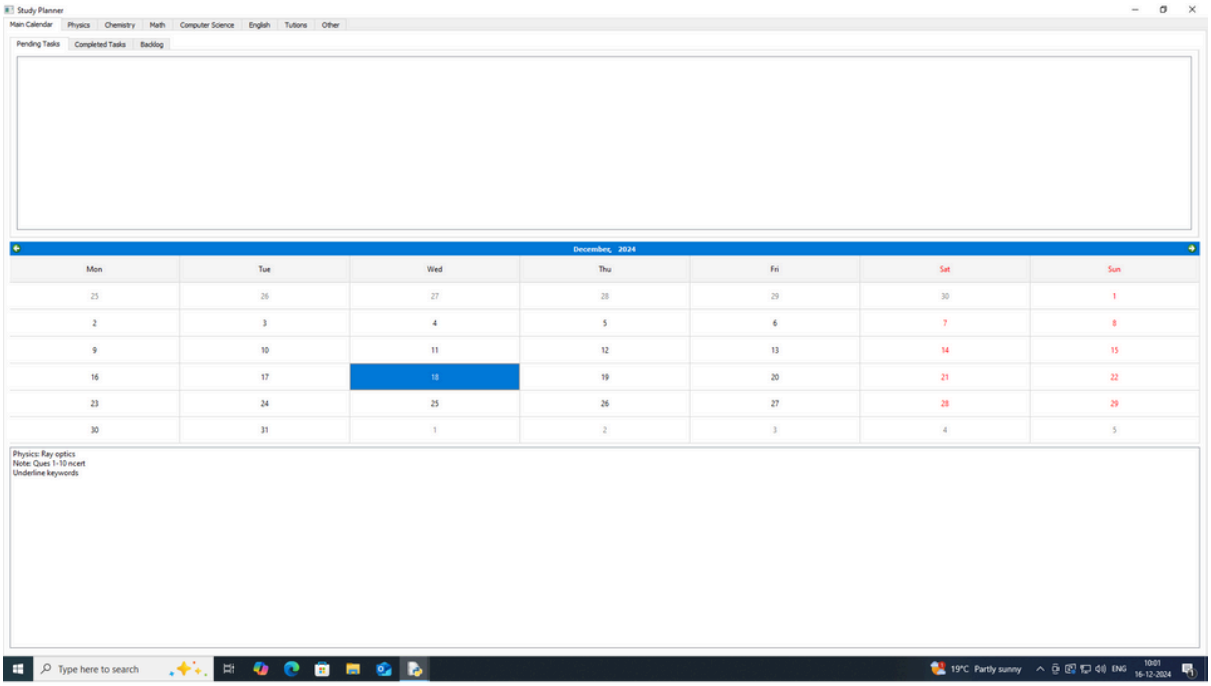

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Ray optics
Ques 1-10 ncert
Underline keywords
2024-12-18
Add Physics Task
Remove Physics Task
Ray optics
Complete Task
Type here to search
Lilo & Stitch (2025) te...
ENG
09:59
16-12-2024

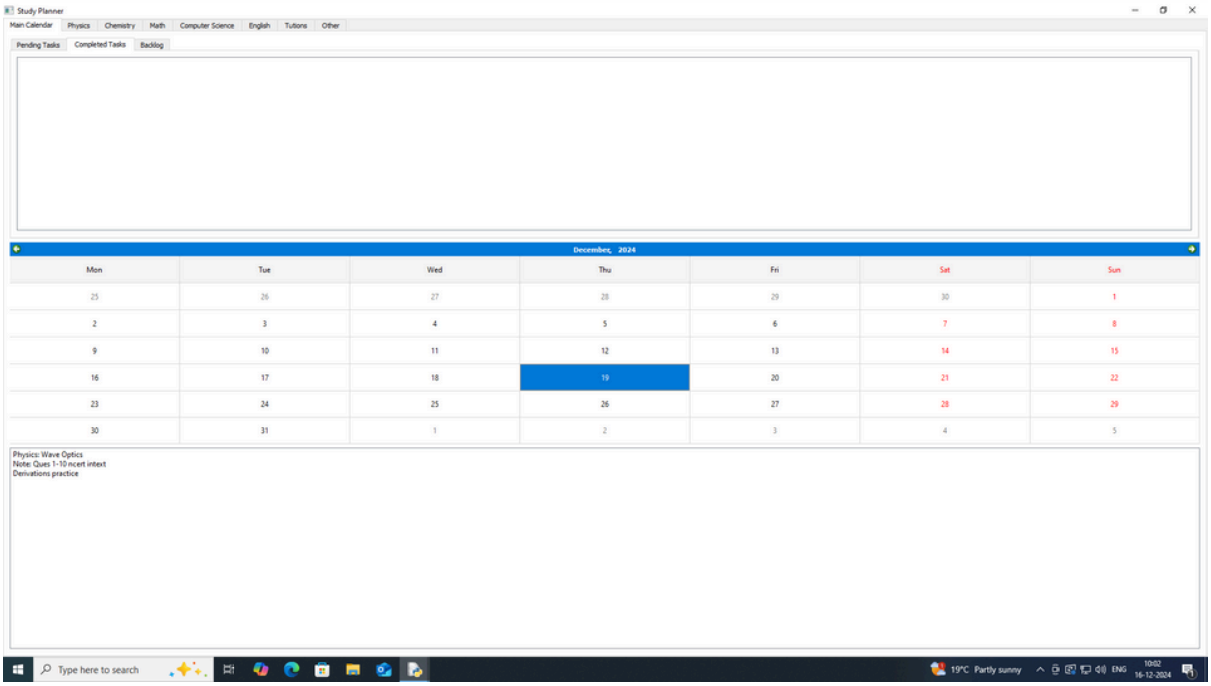

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Wave Optics
Ques 1-10 ncert intext
Derivations practice
2024-12-19
Add Physics Task
Remove Physics Task
Ray optics
Complete Task
Type here to search
19°C Partly sunny
ENG
10:00
16-12-2024

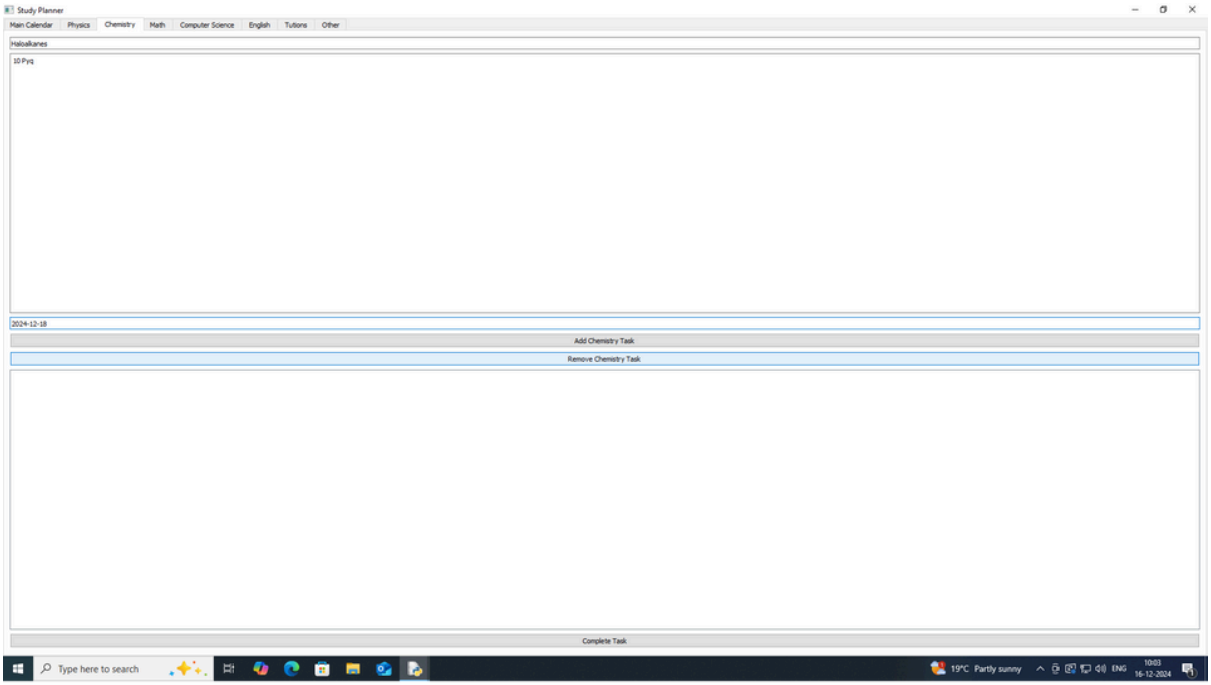

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Wave Optics
Ques 1-10 ncert intext
Derivations practice
2024-12-19
Add Physics Task
Remove Physics Task
Ray optics
Wave Optics
Complete Task
Type here to search
19°C Partly sunny
ENG
10:00
16-12-2024

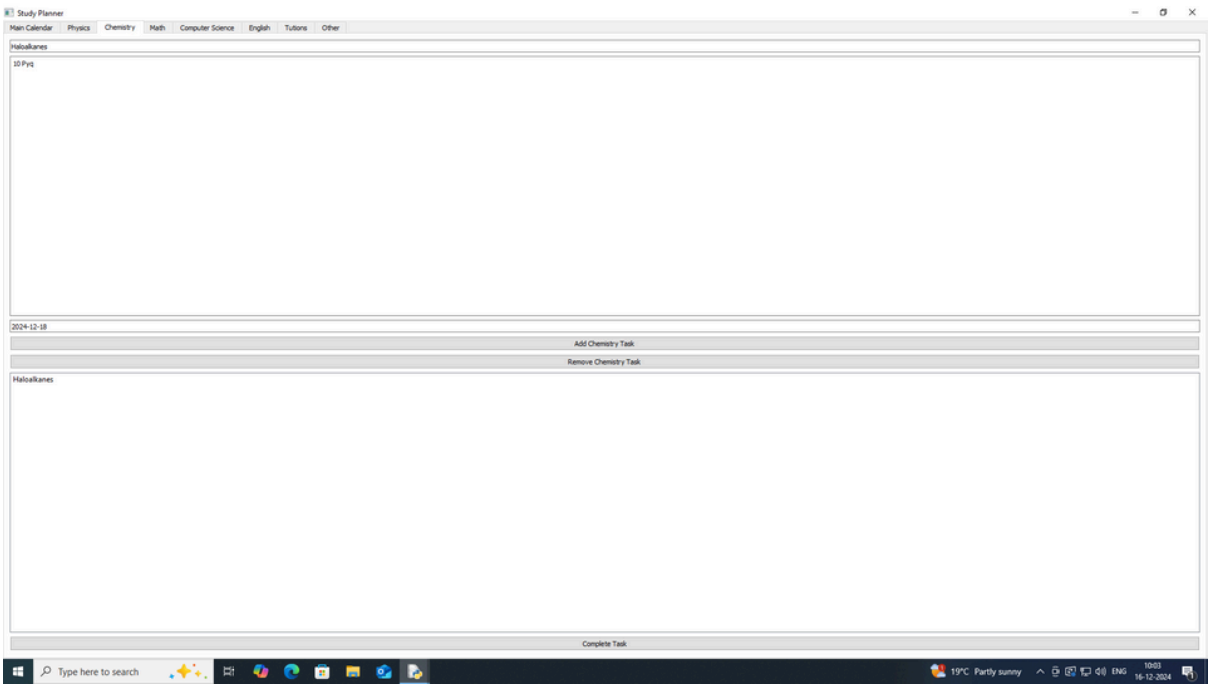

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Wave Optics
Ques 1-10 ncert intext
Derivations practice
2024-12-19
Add Physics Task
Remove Physics Task
Ray optics
Wave Optics
Complete Task
Type here to search
ENG
10:01
16-12-2024

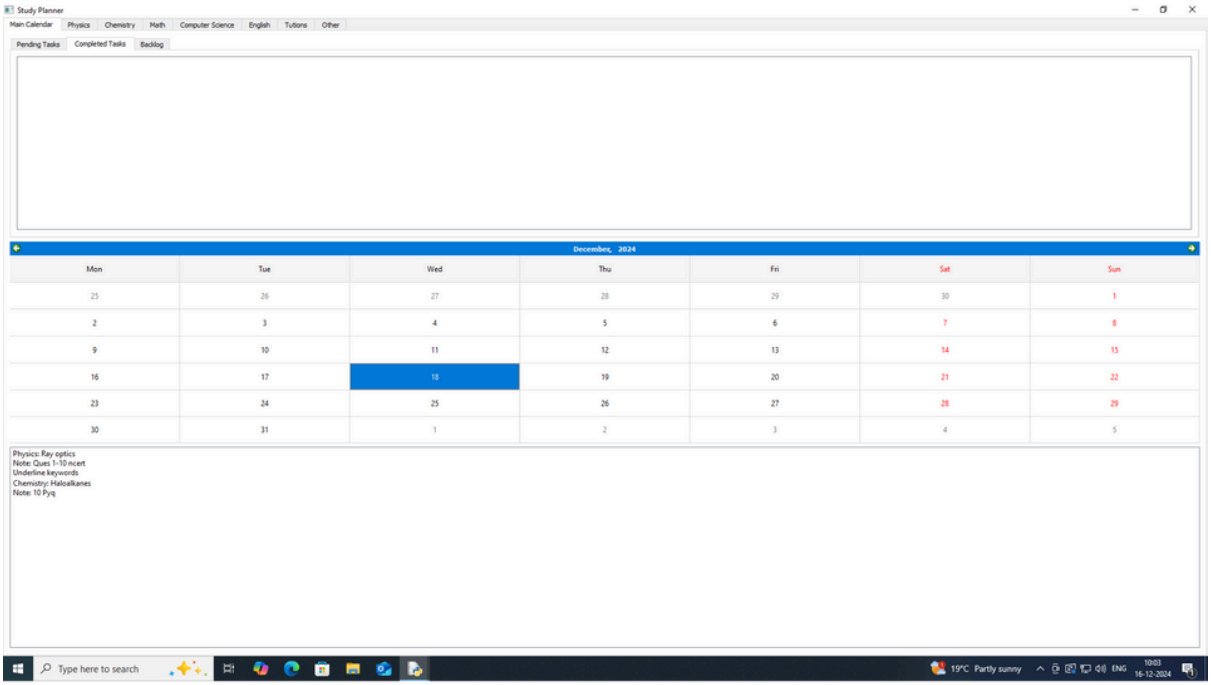

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Wave Optics
Ques 1-10 ncert intext
Derivations practice
Task Completed
The task has been marked as complete. Please restart the app to see changes
OK
2024-12-19
Ray optics
Wave Optics
Complete Task
Type here to search
ENG
10:01
16-12-2024

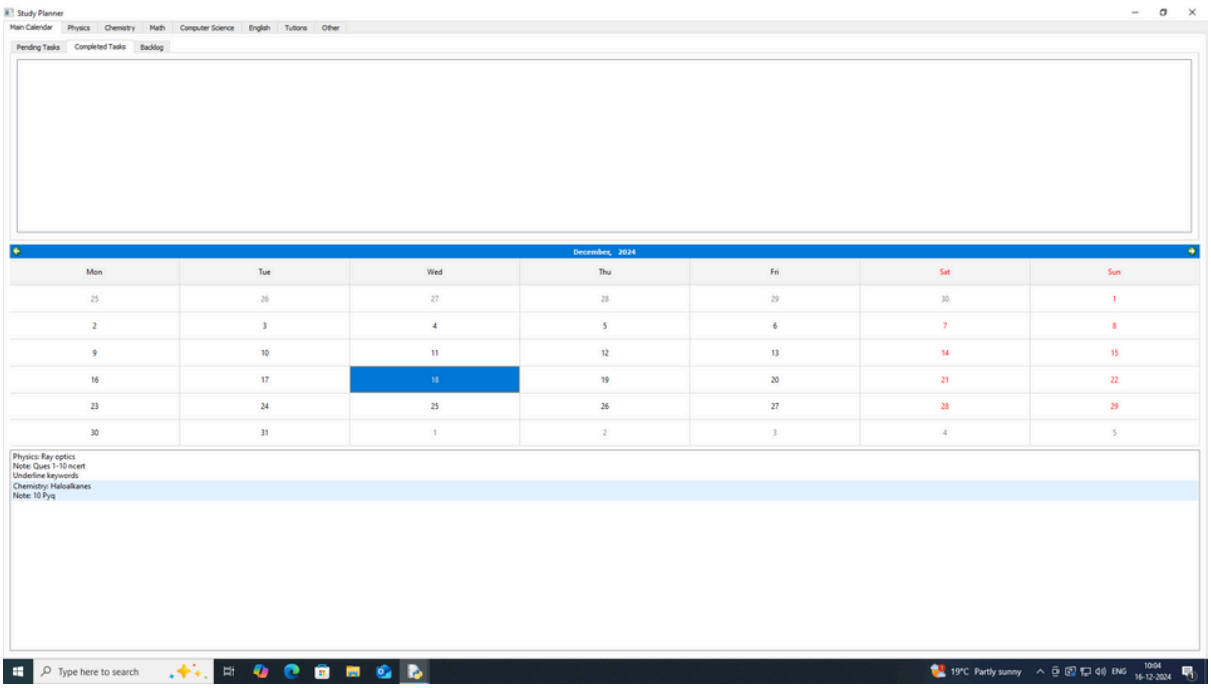

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Pending Tasks
Completed Tasks
Backlog
December, 2024
Mon
Tue
Wed
Thu
Fri
Sat
Sun
25
26
27
28
29
30
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
1
2
3
4
5
Physics: Ray optics
Note: Ques 1-10 ncert
Underline keywords
Type here to search
19°C Partly sunny
ENG
10:01
16-12-2024

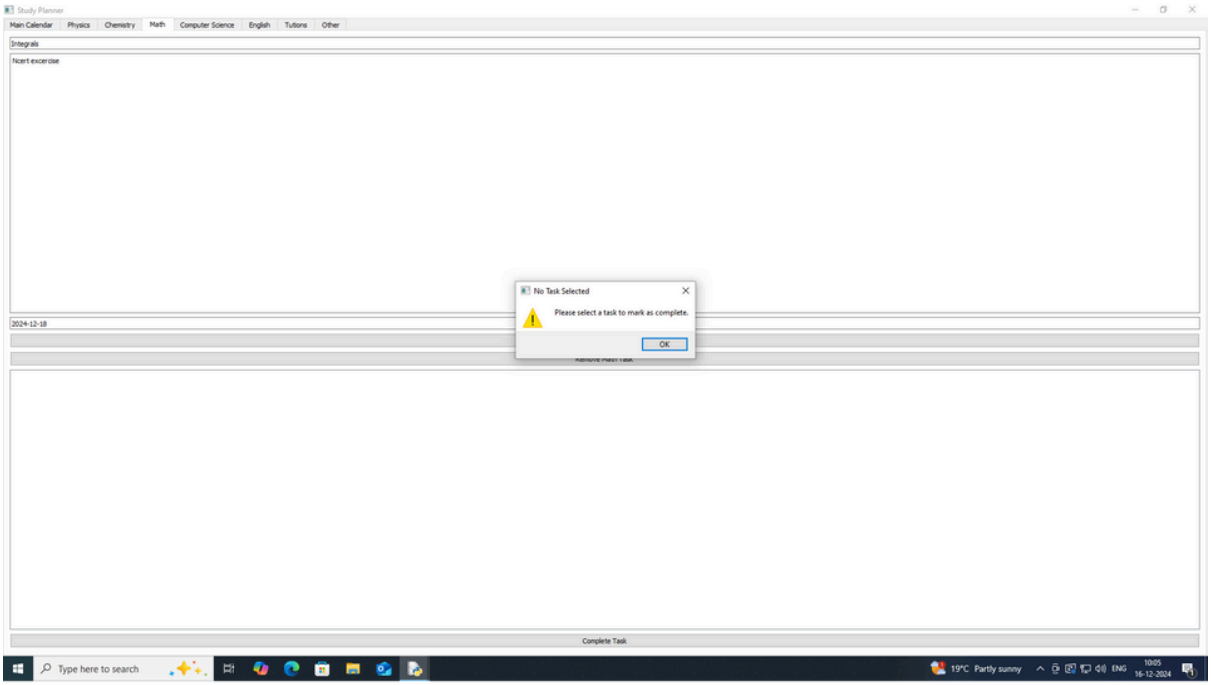

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Pending Tasks
Completed Tasks
Backlog
December, 2024
Mon
Tue
Wed
Thu
Fri
Sat
Sun
25
26
27
28
29
30
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
1
2
3
4
5
Physics: Wave Optics
Note: Ques 1-10 ncert intext
Derivations practice
Type here to search
19°C Partly sunny
ENG
10:02
16-12-2024

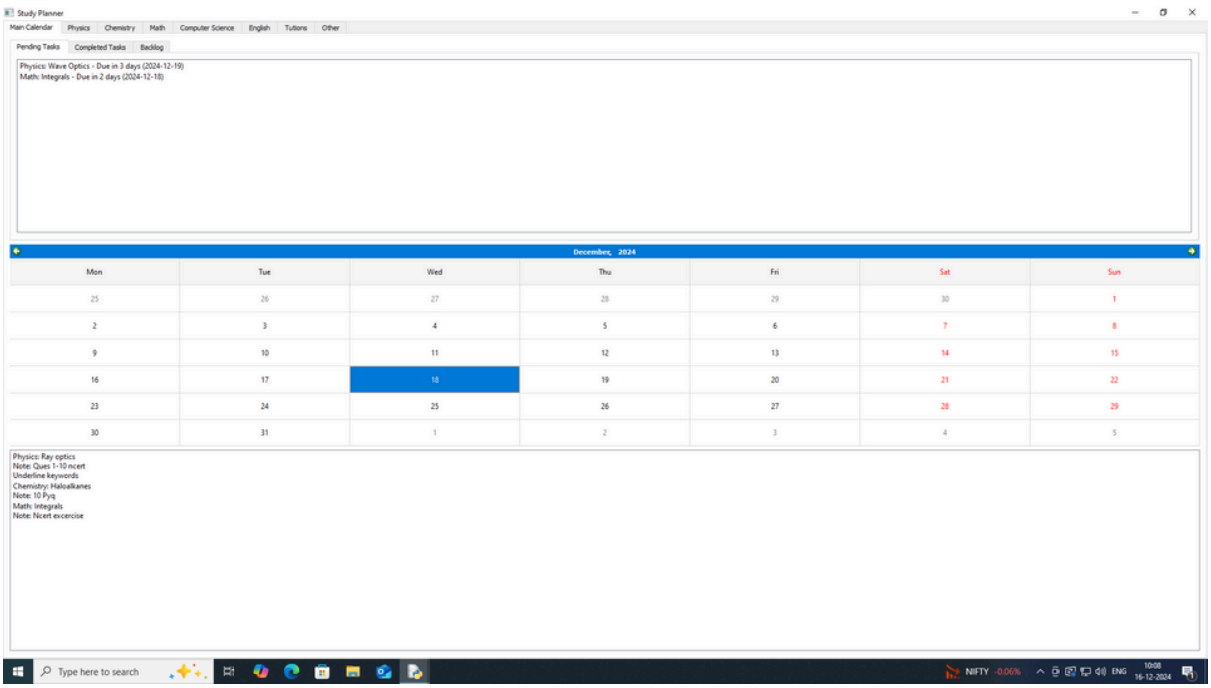

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Haloalkanes
10 Pyq
2024-12-18
Add Chemistry Task
Remove Chemistry Task
Complete Task
Type here to search
19°C Partly sunny
ENG
10:03
16-12-2024

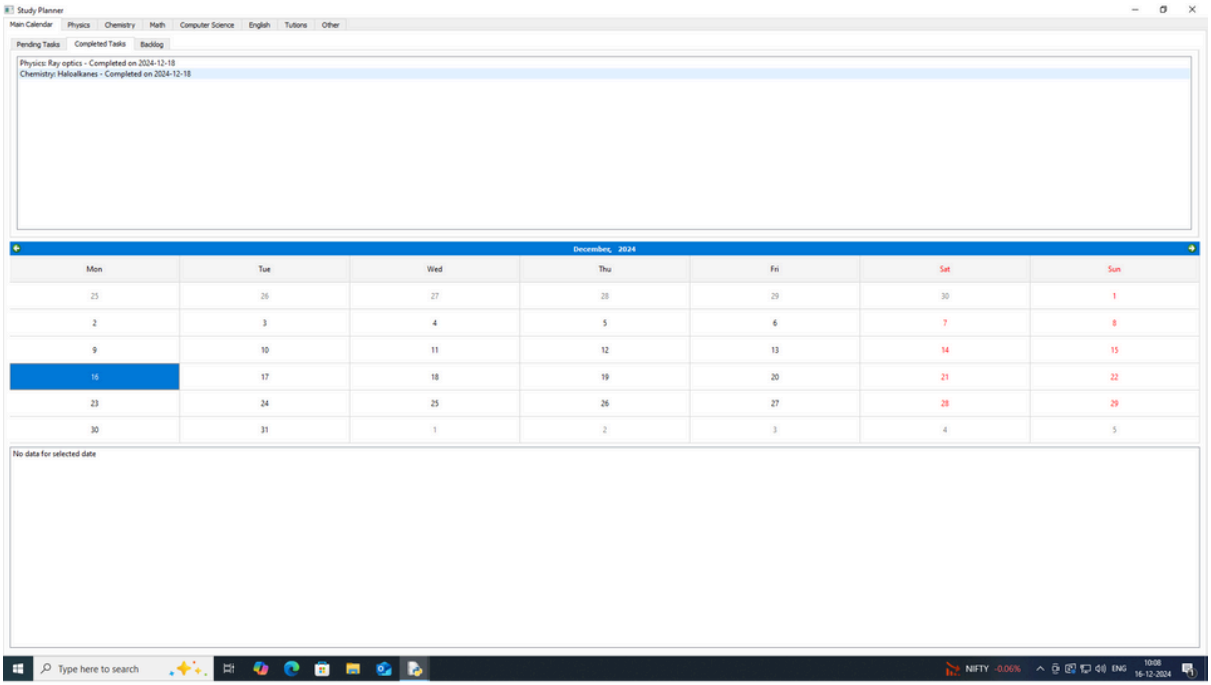

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Haloalkanes
10 Pyq
2024-12-18
Add Chemistry Task
Remove Chemistry Task
Haloalkanes
Complete Task
Type here to search
19°C Partly sunny
ENG
10:03
16-12-2024

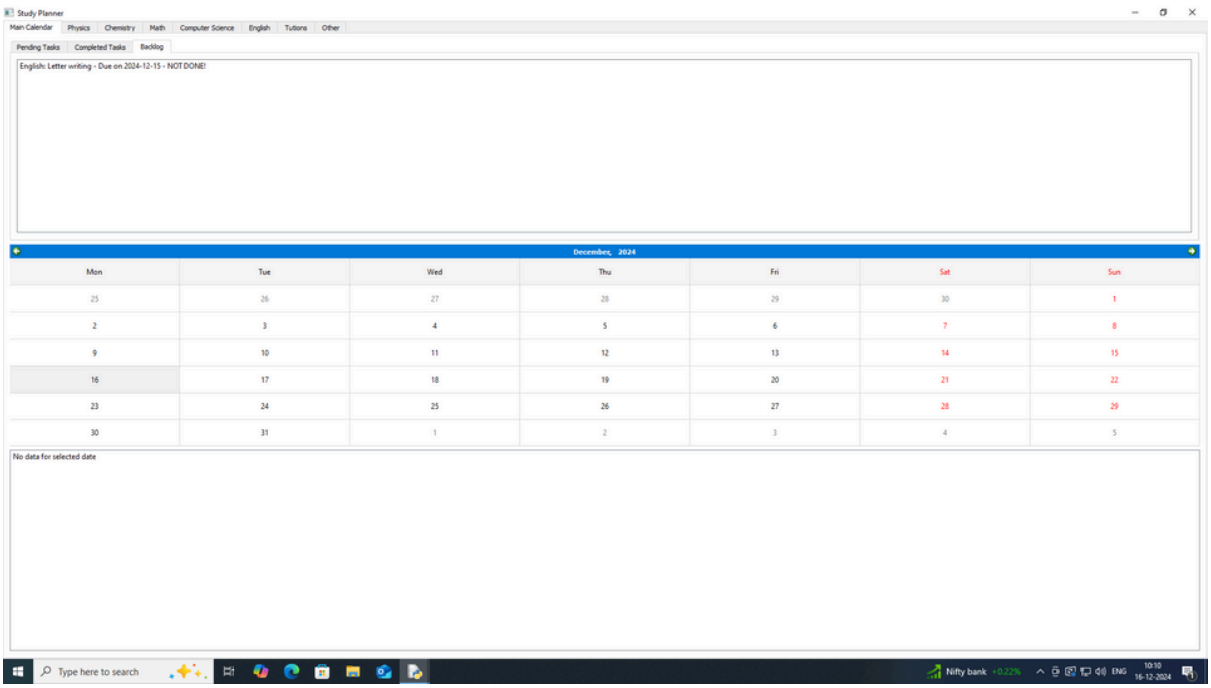

Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Pending Tasks
Completed Tasks
Backlog
December, 2024
Mon
Tue
Wed
Thu
Fri
Sat
Sun
25
26
27
28
29
30
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
1
2
3
4
5
Physics: Ray optics
Note: Ques 1-10 ncert
Underline keywords
Chemistry: Haloalkanes
Note: 10 Pyq
Type here to search
19°C Partly sunny
ENG
10:03
16-12-2024


Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Pending Tasks
Completed Tasks
Backlog
December, 2024
Mon
Tue
Wed
Thu
Fri
Sat
Sun
25
26
27
28
29
30
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
1
2
3
4
5
Physics: Ray optics
Note: Ques 1-10 ncert
Underline keywords
Chemistry: Haloalkanes
Note: 10 Pyq
Type here to search
19°C Partly sunny
ENG
10:04
16-12-2024


Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Integrals
Ncert excercise
No Task Selected
Please select a task to mark as complete.
OK
2024-12-18
Complete Task
Type here to search
19°C Partly sunny
ENG
10:05
16-12-2024


Study Planner
Main Calendar
Physics
Chemistry
Math
Computer Science
English
Tutions
Other
Pending Tasks
Completed Tasks
Backlog
Physics: Wave Optics - Due in 3 days (2024-12-19)
Math: Integrals - Due in 2 days (2024-12-18)
December, 2024
Mon
Tue
Wed
Thu
Fri
Sat
Sun
Physics: Ray optics
Note: Ques 1-10 ncert
Underline keywords
Chemistry: Haloalkanes
Note: 10 Pyq
Math: Integrals
Note: Nicert excercise
Type here to search
NIFTY -0.06%
ENG
10:08
16-12-2024





BIBLIOGRAPHY

- Computer Science With Python by Sumita Arora for Class 12
- https://www.w3schools.com/
- https://www.geeksforgeeks.org/
- https://stackoverflow.com/