PROJECT REPORT

JEE/NEET Strategic Study Planner

- ## Introduction:

  In India, there are millions of students preparing for competitive exams like JEE and NEET. These exams define their future careers in engineering and medical fields. In these exams the hurdle is not always the difficulty of the subjects of subjects – Physics, Chemistry, Mathematics and biology but also the vast syllabus.

  Success in these exams requires more than just hard work, it demands consistency and smart planning. Many students every year struggle to create a study planner. They start their studies with high energy by making over ambitious plans but burn out within a week.

  This **JEE/NEET Strategic Study Planner** is my attempt to solve this problem using python. It gives all the tips and tricks as well as a well organised planner simply by entering exam date, the program generates a day-by-day roadmap ensuring that student can cover every chapter and perform well in examination.

- ## Problem Statement:

  Students preparing for these exams faced many problems but among them main problem is "**Planning Paralysis**."

  Key problems are:

  1. **Time Management**: It's difficult to calculate that how many hours should you study in a day according to your exam date.
  2. **Syllabus coverage**: This is also a big confusion and problem faced by the aspirants that how to divide chapters evenly so that everything is finished before time of revision.
  3. **Resource confusion**: Uncertainty about which books or strategies to prioritize for specific subjects.

  This project aims to solve these issues. The students do not need to calculate dates; the program uses an algorithm to distribute chapters across the remaining days and provide a strategic study planner.

- ## Functional Requirements:

  This planner provides the following functionalities:

  1. **User Customization**: Firstly, it accepts the input from user for the target exam (JEE or NEET) and the scheduled exam date.
  2. **Time Calculation**: It automatically calculates the number of days remaining until the exam.
  3. **Study Hour Allocation**: It determines the required daily study hours based on the exam date.
  4. **Generate Dailly Routines**: It provides user a perfect schedule including break time which helps to focus on studies well.
  5. **Resources Recommendation**: Provides subject wise tips and recommend books for preparation.

- ## Non – Functional Requirements:

  1. **Usability**: This program is made up of CLI (Command Line Interface) which features this program lightweight.
  2. **Efficiency**: The algorithms execute instantly (complexity relative to syllabus size).
  3. **Reliability**: This program handles invalid sate formats and incorrect choices.
  4. **Scalability**: Dictionary based data structure feature allows for easy addition of new subjects or exams.

- ## System Architecture:

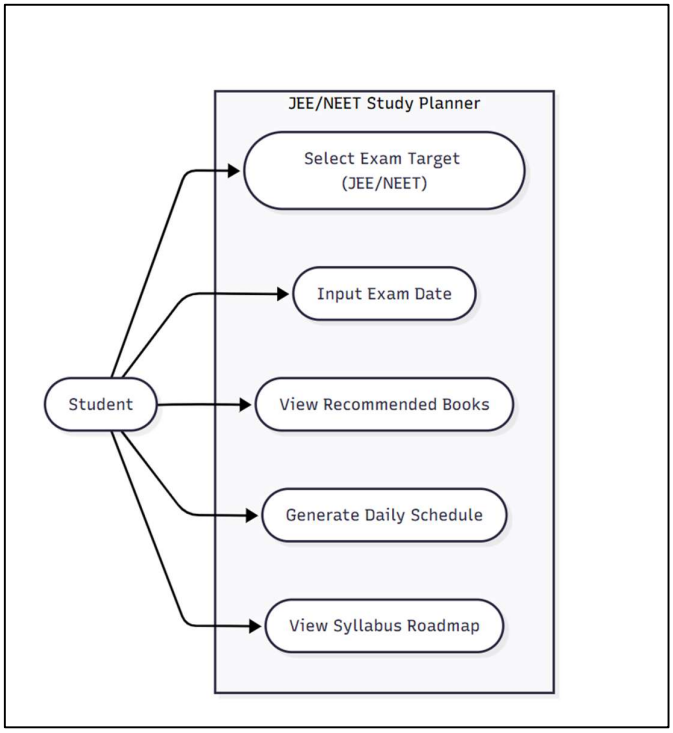
  The systems follow a modular procedural architecture. It utilizes python's standard libraries to process inputs and generate output.

  ### Core modules:

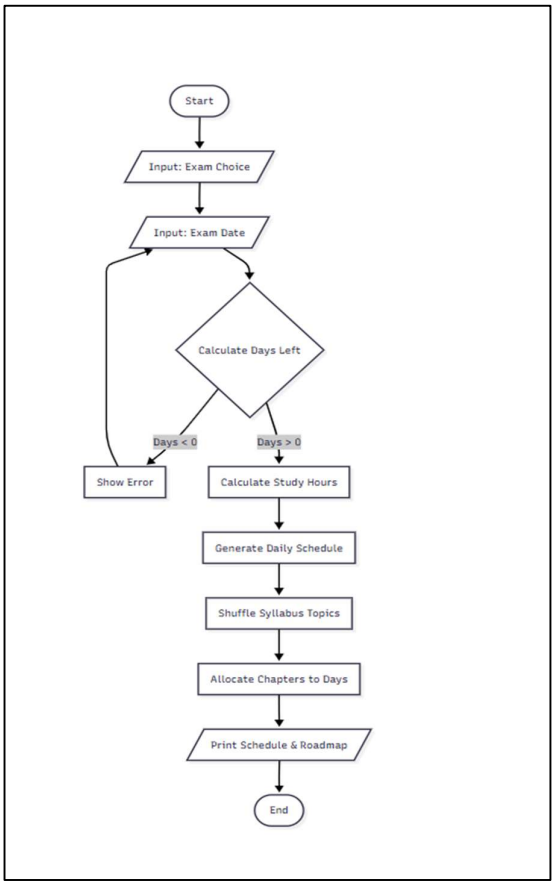
  1. **Data Store**: This program used dictionaries storing Syllabus, Exam date, Book lists, Tips.
  2. **Logic Layer:** This program includes functions for time calculation (timetable, scheduling (schedule), and chapter mapping(Days).
  3. **Presentation layer**: This function includes the main () function that handles user I/O.

- **Design Decisions and Rationale:**

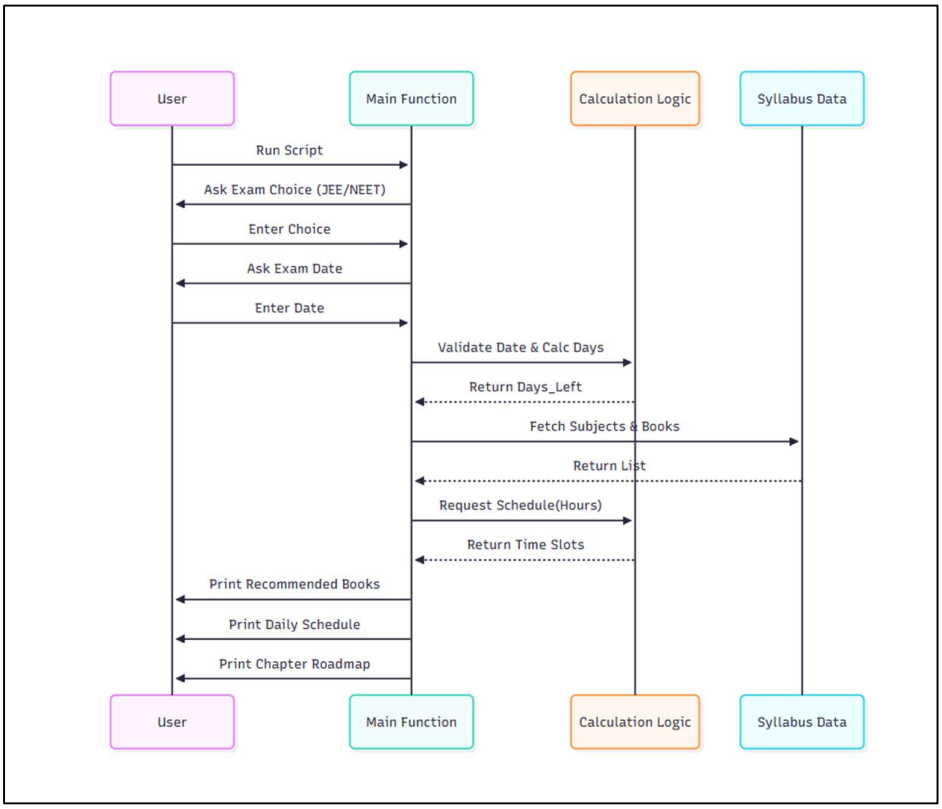
1. **Use case Diagram**:



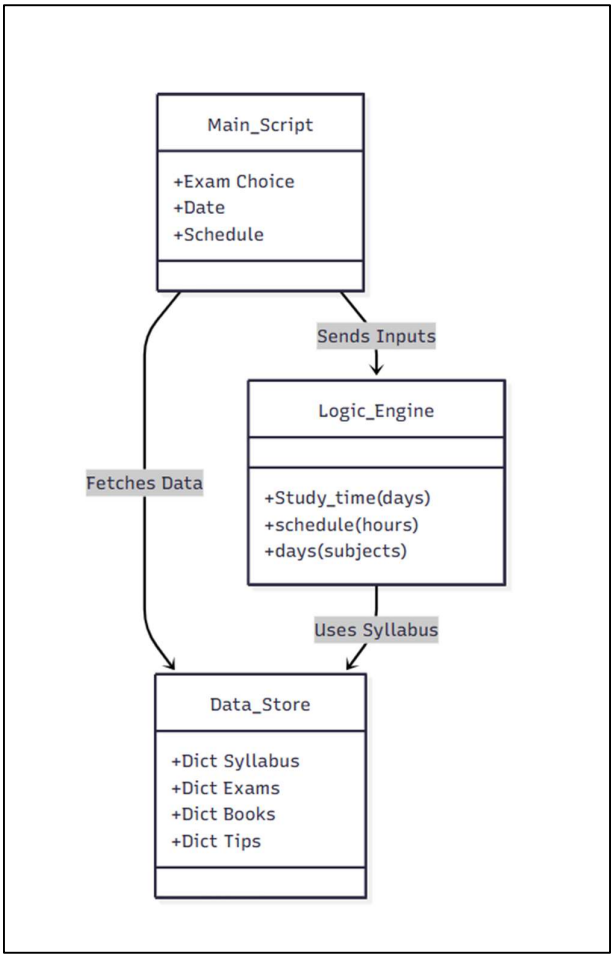
2. **Workflow diagram:**



3. Sequence diagram:



4. Class/Component Diagram:



- ## Designs Decisions and Rationale:

  1. **Language Selection:**
     I chose python for its readability and powerful standard modules like datetime and random which simply manipulate the date and shuffle the syllabus.
  2. **Data Structures (Dictionaries):** Using nested dictionaries for Syllabus and Books allows for structured data and updates without changing the code logic.
  3. **Algorithm for Chapter Allocation:** The logic calculates base_days per chapter using integer division:

$$\textbf{Base Days} = \frac{\textbf{Total Days-31}}{\textbf{Total Chapters}}$$

- This reserves the last 31 days for revision and mock tests, ensuring the syllabus is finished a month prior.

- ## Implementation Details:

  This project is implemented in a single pyhton script utilizing the following key components:

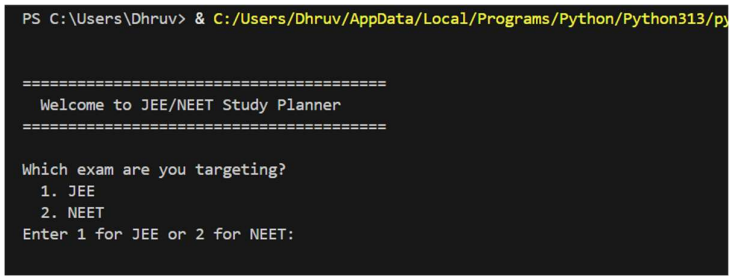
  ### Libraries used:

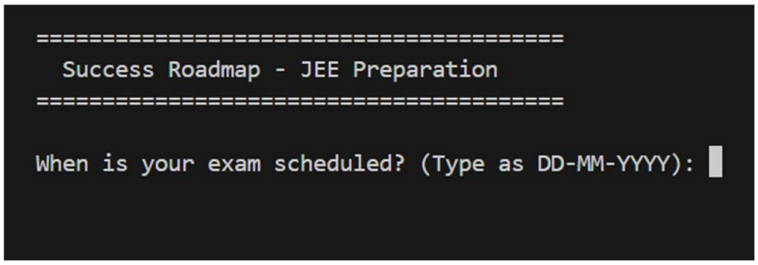
  1. **datetime:** It allow user inputs dates and calculate and calculate the remaining days.
  2. **math:** Used for math. ceil () to round up study sessions.
  3. **random:** It is used random. shuffle () to randomize the order of chapters to prevent monotony (e.g. not studying mechanics only for 2 weeks straight).

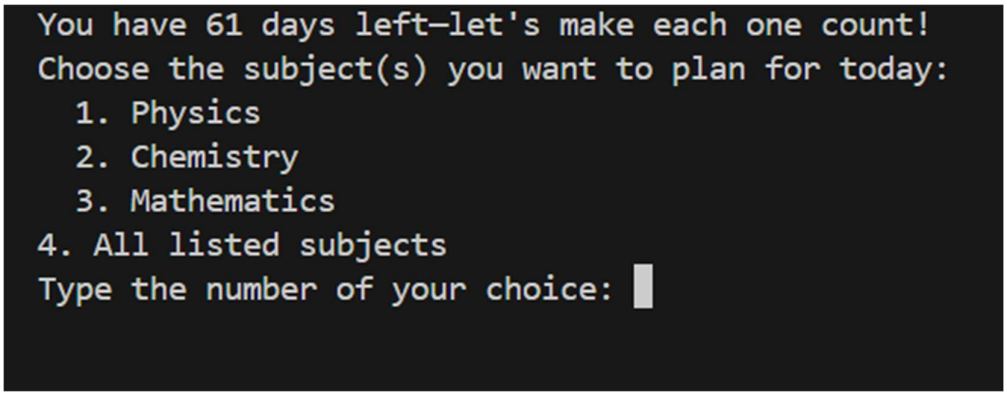
  ### Key Functions:

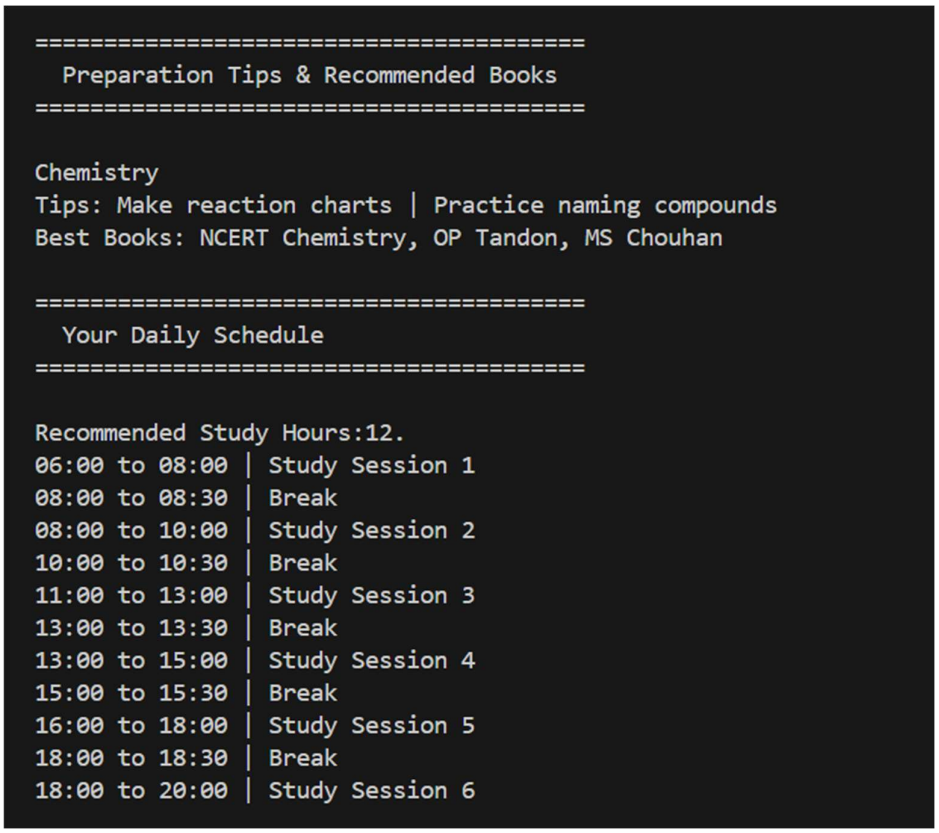
  1. timetable(days_left): It returns study hours (6,8,10 or 12) which is based on how close the exam is from present time.
  2. schedule(hours): It can generate string representing 2-hour study slots separated by 30 minutes breaks.
  3. days (subjects, total_days): This program can distribute chapters across the available timeline.

- **Screenshots/Results:**









```
========================================
  Your Chapter-by-Chapter Path
========================================

CHEMISTRY
----------------------------------
Day 1-5: Equilibrium (Chemical equilibrium, Ionic equilibrium, Buffer solutions, Solubility product, Common ion effect)
Day 6-10: Electrochemistry (Electrolytes, Conductivity, EMF, Electrochemical cells, Corrosion)
Day 11-14: Coordination Compounds (Nomenclature, Isomerism, Bonding theories, Applications)
Day 15-18: Chemical Bonding (Types of bonding, Molecular orbital theory, VSEPR theory, Hybridization, Bond parameters)
Day 19-22: Physical Chemistry (Atomic structure, Chemical bonding, Equilibrium, Thermodynamics, Electrochemistry, Solutions, Solid state, Surface chemistry)
Spend extra time on challenging topics today. Attempt more practice MCQs. Summarize concepts in your own words. Revise errors from last mock test.
Day 23-26: Inorganic Chemistry (Periodic table, s/p/d/f-block elements, Coordination compounds, Metallurgy, Qualitative analysis, Hydrogen, Group chemistry)
Day 27-30: Organic Chemistry (General organic chemistry, Hydrocarbons, Haloalkanes, Alcohols, Phenols, Ethers, Carbonyls, Carboxylic acids, Amines, Polymers, Biomo
lecules, Environmental chemistry)
Spend extra time on challenging topics today. Attempt more practice MCQs. Summarize concepts in your own words. Revise errors from last mock test.

Your 3-Day Countdown Strategy
Day 31: Rapid revision from notes and flashcards
Day 60: Take a mock test, then calmly review your mistakes
Day 61: EXAM DAY! Light review and bring your confidence!

========================================
  Stay Inspired - Your Success Mantras
========================================

['Small steps done consistently lead to greatness.', 'Keep your mind healthy—take mindful breaks.', 'Discuss doubts; never hesitate to ask questions.', 'Avoid last
-minute cramming. Trust your prep.']

========================================
  All the Best!
========================================

You have 61 days to reach your dream. Trust yourself, and let each day's effort bring you closer!
Remember, success comes from dedication, not pressure. You're on a path to greatness!
```

- **Testing Approach:**

Unit Testing:

- **Test Case 1**: Inputting a past date.
  Results: Systems prompt “Exam date should be upcoming.”
- **Test Case 2:** Inputting non-existent exam choice (e.g. 5).
  Results: invalid input.

Boundary Testing:

- It can be tested with an exam date exactly 1 day away to ensure the logic holds or not.

Format Testing:

- This can be tested by entering the date as 2025/05/10 instead of 10-05-2025. System caught the exception and asked to re-enter the date.

- **Challenges Faced:**

1. **Date Format Errors**: Users often enter dates in different formats. I implemented a try-except block to handle Value error during date analysis.
2. **Even Distribution of Chapters:** The total number of Chapters rarely divides perfectly into the remaining days.

Solution: I have used modulo operator to calculate “extra days” and distributed them to the first few chapters in the list.

- **Learnings & Key Takeaways:**
    1. Gained proficiency in Python’s Datetime module for real-world time calculations.
    2. I have learned how to structure data using Nested Dictionaries like structures.
    3. I have understood the logic behind allocating resources (time) over a fix days, which is fundamental concept in algorithm design.

- **Future Enhancement:**
    1. **GUI Implementation**: We can migrate from CLI to Graphical User Interface using Tkinter for better user experience.
    2. **Save Functionality**: We can add a feature to export the generated schedule to a pdf.
    3. **Progress Tracking**: Implement a feature to mark chapters as “complete” or “incomplete”.
    4. **Google Calendar Integration**: We can also use APIs to automatically add the schedule to the user’s Google Calendar.

- **References:**

    1. Official JEE/NEET syllabus from NTA Official Website.
    2. VITYARTHI COURSE , VIT BHOPAL.