**Student Class Planner Application Initial Prompt**

1. **Background**: The application will serve as a centralized platform for faculty members to input, update, and manage graduate students' course plans, aiding in efficient class planning and resource allocation.
2. **Basic Application Design**:
   * **Main Menu**: Upon launch, users are presented with a menu offering options to add, update, delete student programs, and generate enrollment reports.
   * **Add Student Program of Study**:
     1. **Heading**: Contains fields for student first-name, last-name, emphasis area (dropdown with three options), and student ID.
     2. **Body**: Allows input for up to 12 classes and their corresponding semesters.
     3. **Buttons**: 'Submit' to save data, 'Cancel' to return to the main menu, and 'Load Default' to auto-fill based on the selected emphasis area.
   * **Update Student Program of Study**:
     1. Users input a student ID.
     2. If found, the program displays the student's data, allowing edits. If not, a "not found" message appears.
     3. 'Cancel' button to return without saving changes.
   * **Delete Program of Study**:
     1. Users input a student ID.
     2. If found, a confirmation prompt appears before deletion. If not, a "not found" message is displayed.
   * **Generate Report**:
     1. Users input a desired semester.
     2. The application displays projected enrollments for each class during that semester.
3. **Modular Program Design**: Each function (add, update, delete, report) is developed as a separate module, ensuring maintainability and scalability.
4. **Database Structure**:
   * A MySQL database with tables for students, courses, and enrollments.
   * Relations to be established between student IDs and courses.
5. **GUI Framework**: Utilizes Java's Swing framework, ensuring a consistent and responsive user experience.
6. **Error Handling**: Basic error messages for common issues like "Student Not Found" or "Invalid Input".
7. **Authentication**: Given the internal nature of the application, no authentication layer is required.
8. **Concurrency**: Designed for individual use, ensuring data integrity without the need for concurrent access handling.
9. **External Libraries**: While the core functionality relies on Java's standard libraries, external libraries can be integrated for enhanced database connectivity or specialized GUI components.
10. **Deployment**: Packaged as a standalone application, it's deployable on both local VMWare Virtual Machines or cloud platforms for broader accessibility.
11. **User Roles and Permissions:** Faculty, Teacher Assistant.
12. **Data Validation and Constraints:** Validate where appropriate. The date fields should be numeric.
13. **Backup and Recovery:** After the application is developed, backup and recovery classes will be developed.
14. **Performance Requirements:** No specific requirements.
15. **Integration with Other Systems:** Not applicable.
16. **User Experience (UX) and User Interface (UI) Design:** Use Metal, but no specific design guidelines, color schemes, or branding elements are provided – use your judgement.
17. **Logging and Auditing:** Not applicable.
18. **Feedback Mechanism:** Not applicable.
19. **Training and Documentation:** This should be undertaken after the application's development is complete, ensuring that the materials accurately reflect the final version of the software and its functionalities.
20. **Scalability and Future Expansion:** Not applicable.