**i.**

The academic lifecycle refers to the processes undertaken by an educational institution from the point of accepting students to the point of actually interacting with the graduates. It includes admission, enrollment, creation of class schedules, course offering, evaluation, monitoring, graduation and alumni interaction. The goal is to facilitate the student’s life cycle in relation to both the academic and administrative requirements.

**ii.**

1. **Admissions**: Handling applications, reviewing student credentials, and admitting new students.
2. **Course Management**: Organizing courses, assigning faculty, and ensuring the course schedule is up-to-date.
3. **Grading**: Recording students' performance, computing grades, and maintaining academic records.
4. **Scheduling**: Creating class schedules, managing room assignments, and accommodating teacher availability
5. **Alumni Management**: Keeping in touch with graduates, supporting alumni networks, and tracking their career progress.

**iii.**

Software process model allows to work on projects in a systematic way. They ensure better communication, better project requirements gathering and result in an effective quality project. Models like Waterfall, Agile, and Spiral help plan, develop, test, and deploy software efficiently.

**iv.**

Developing an Academic ERP system involves managing many interconnected processes, like admissions, course registration, and grading. Software process models offer a structured approach to handle these complexities. They help organize tasks, manage dependencies between modules, and ensure development follows a clear plan. This approach also helps identify and address potential issues early in the development process.

**v.**

1. **Waterfall Model**: A very simple linear approach , it is useful when all the requirements are clear. It does not allow changes afterward/
2. **Agile Model**: It is an Iterative and flexible model , focuses on small, incremental releases. It allows continuous feedback in each iteration and suitable for projects where requirements are not very certain and changes over time.
3. **Spiral Model**: An iterative approach which also allows risk management. It’s effective for complex projects as it emphasizes risk assessment and can handle changes well.
4. **Hybrid Model**: Combines elements of Waterfall and Agile to offer both structure and flexibility. It can be suitable when initial requirements are stable but may need refinements.

**vi.**

For an Academic ERP system, the **Spiral Model** is a good choice. Its iterative approach allows the team to assess risks at each step, which is essential for a complex system like ERP. This model combines the structured nature of Waterfall with the flexibility of Agile, making it suitable for the large scope and evolving requirements typical of Academic ERP systems. Its focus on risk management helps address potential issues early in the project.

**vii. Provide a detailed explanation of how the chosen software process model is applied in the development of the Academic ERP.**

In the **Spiral Model**:

1. **Planning**: In each iteration, requirements are gathered for different modules, such as admissions, scheduling, and grading.
2. **Risk Analysis**: Potential risks for each module, like data security in student records, are identified and addressed before moving forward.
3. **Development and Testing**: Each module is developed and tested individually, allowing issues to be fixed immediately without impacting the entire system.
4. **Evaluation**: Stakeholders review the progress and provide feedback. This feedback loop allows the development team to make adjustments as needed.

**viii.**

1. **Requirements Gathering**: Identify functional and non-functional requirements for each module (e.g., admissions, grading). This phase involves discussions with institution stakeholders to ensure the system meets all academic needs.
2. **Design**: Create detailed design documents, including database design and interface mockups for each module. For example, the course management module’s interface will include features for adding, removing, or updating courses.
3. **Implementation**: Develop the code for each module. For instance, coding the grading module to calculate and store grades securely. Each module is implemented in stages based on the plan.
4. **Testing**: Each module undergoes rigorous testing for functionality, performance, and security. For example, testing the scheduling module to ensure it doesn’t allow scheduling conflicts.
5. **Deployment**: After successful testing, the Academic ERP system is deployed in stages to ensure a smooth transition. Users receive training on the system, and live data is gradually integrated.