Software Engineering - Assignment - 1 Harshit - 2301010210

Ans 1: 1) Causes of Software Crisis

- Scope Creep: No control over changing requirements → delays & complexity.
- Poor User Interface: Ignored user needs → low adoption.
- Integration Issues: weak planning with legacy systems → failure in data exchange.

2) Model Evaluation

- Waterfall: Rigid, poor for changing needs & late error discovery.
- Spiral: Iterative, risk-focused, allows prototype & user feedback.

Spiral is more suitable for handling evolving requirement, risks, and integration.

3) Improvements:

- Apply Agile principles → small iterations & user feedback.
- Use early prototyping & usability testing.
- use TSP/PSP for disciplined tracking & quality.
- Plan continous integration with legacy systems.

Ans 2) 1) Prototype vs Evolutionary Model:

→ Prototype Model:

- Builds quick mock-ups to capture user needs.
- useful for clarifying unclear requirements
- weak in handling large, complex systems and long term compliance.

→ Evolutionary Model:

- Develops the system in increments, each delivering a working version.

- Allows continous user feedback and regularly checks.
- Better suited for integration of multiple modules (Patient Records, Billing Pharmacy).

2) Justification:

- User Feedback: Evolutionary model ensures ongoing feedback after each release, not just during prototypes.
- Compliance: Regulatory standards (like data privacy) can be validated at every phase.
- Phased Delivery: Supports gradual rollout of modules, reduced risks.

→ Evolutionary model is better since it balances user involvement, compliance, and modular delivery.

3) Requirement Elicitation & Risk Handling

- Early Requirements Elicitation &: Begin with core requirements (Patient Records) and gather feedback via small releases. Prototypes may be used within increments for clarity.
- Risk Handling: Each iteration identifies risks (eg, data security, module integration) early, allowing corrections before full-scale roll out.

Ans 3) Elicitation Techniques

- Municipalities: Interviews, document analysis → rules & fees.
- Drivers: Surveys, focus groups → usability & features
- Traffic Police: Workshops, observation → violation alerts.

2) Context-level DFD

- Drivers ↔ App (requests, slot info, payments).
- Municipality ↔ App (rules, reports).
- Traffic Police ↔ App (violations, alerts).

Requirements

Functional:

1) Show real-time slot availability
2) Allow reservation
3) Enable secure payments
4) Notify police of violations
5) Generate reports for municipalities.

Non-functional:

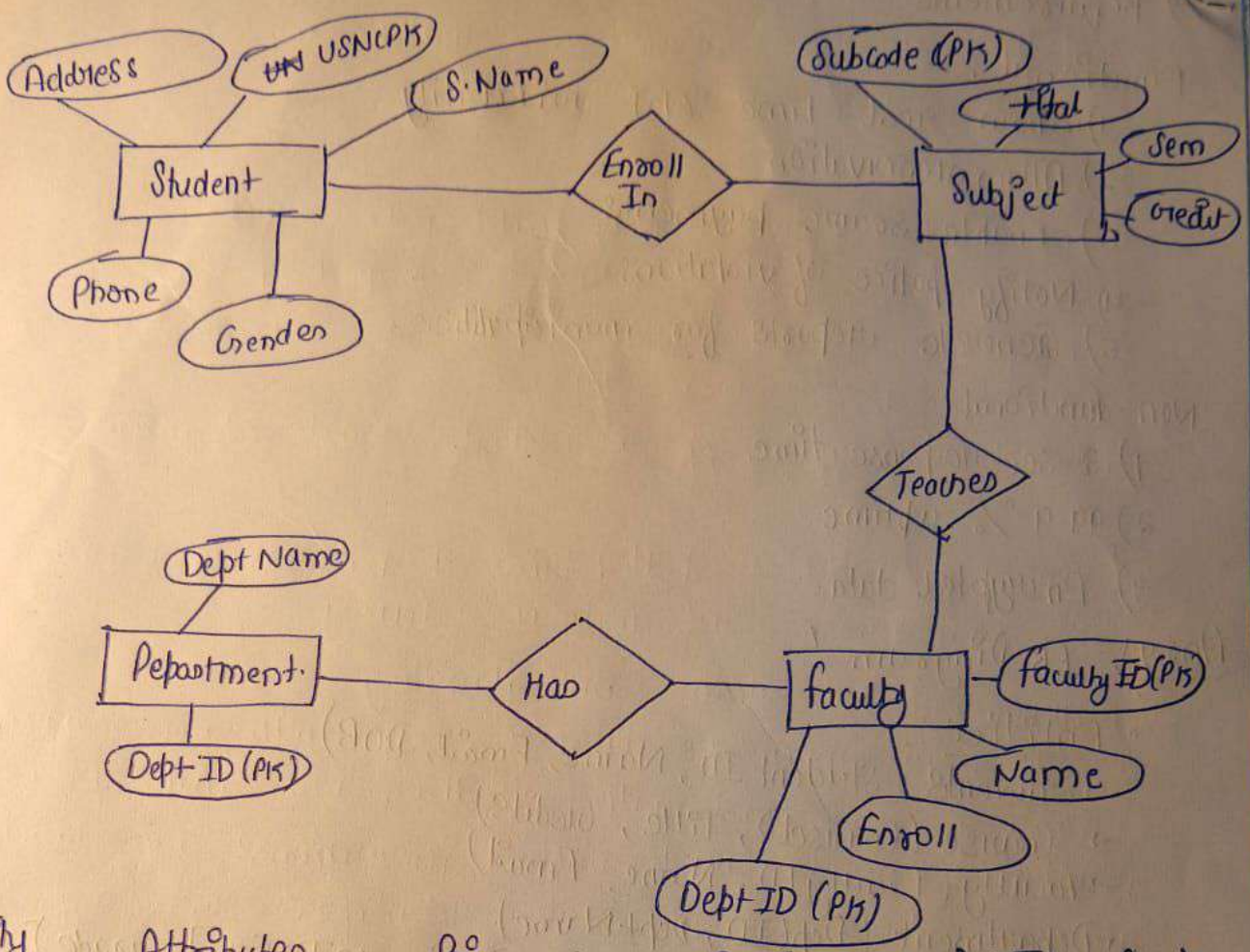
1) 2-sec response time
2) 99.9% uptime
3) Encrypted data.

Ans 4) ER Diagram:

- Entities:
  - → Students (Student ID, Name, Email, DOB)
  - → Course (CourseID, Title, Credits)
  - → faculty (FacultyID, Name, Email)
  - → Department (DeptID, DeptName)
  - → Enrollment (EnrollID, StudentID, CourseID [FK], Grade)
- Relationships:
  - → Student ↔ Course (M:N) via Enrollment.
  - → Course ↔ Faculty (Many courses taught by 1 faculty)
  - → faculty ↔ Department (Many faculty belong to 1 department)

2) Data Dictionary

Entity	Attributes	PK / FK	Data Type
• Student	StudentID, Name, Email, DOB	PK: StudentID	INT, VARCHAR, DATE
• Course	CourseID, Title, Credits	PK: CourseID	INT, VARCHAR, INT
• faculty	facultyID, Name, Email, DeptID	PK: facultyID, FK: DeptID	INT, VARCHAR, VARCHAR
• Enrollment	EnrollID, StudentID, CourseID, Grade	PK: EnrollID, FK: StudentID, CourseID.	INT, INT, INT, CHAR.



Entity	Attributes	Primary key	foreign key	Data Types (Ex).
Student	StudentID, Name, DOB, Email, Phone	StudentID	StudentID	INT, VARCHAR, DATE
Course	CourseID, Title, Credits, ~~faculty ID~~	CourseID	CourseID	INT, VARCHAR, INT
faculty	facultyID, Name, Email, Dept ID.	facultyID	Dept. ~~Faculty~~ ID	

30/09.