Name :- Harsh Kumar
Class :- CSB 1
CRN :- 2315085
URN :- 2302543

Software Engineering (SE)

Assignment :- 1

Q1. Given a software project with tight deadlines, unclear initial requirements and a high risk of changing specification with SDLC model. which would you choose and why? Analyze the strengths and weakness of atleast two other models in comparison to your choice

Ans For software projects with tight deadlines, unclear initial requirements and a high risk of changing specification, the spiral Model is the best choice. This model combines iterative development with risk management, allowing flexibility for changing requirements

Reasons

1.] Risk management :- Since the model has risk assesment at each phase, it is suited for project like this with changing specification

2.) Flexibility :- The iterative nature allows for change in project to accomodate for evolving requirements, making it a good choice for this Project

3) Easy Prototyping :- Helps in clarifying unclear requirements through early Prototyping

Comparisons

1.] waterfall Model :-

(a) Strengths :- → well structured and easy to manage
→ clearly defined phases with Outputs

(b) weakness :- → Rigid structure, backtracking is difficult
→ Poor adaptobility to changing Requirements
→ late testing phase increases risk of undiscovered major issues

(c) Comparison :- The waterfall model is unsuitable for unclear initial requirement and changing specification unlike spiral model

2.) RAD Model

(a) Strength :- → Enables quick prototyping and feed back
→ Reduced development time suitable for tight deadlines

(b) Weakness :-
→ Requires highly skilled developer
→ low risk analysis
→ Scalability issues for complex projects

(c) Comparision :- while RAD model is good for tight deadlines and evolving requirement it lacks the risk management and scalability of spiral model

Q.2. A Banking application require a secure login system. while the functional requirement is to implement MFA, what possible non-functional requirement should be considered? How could ignoring them impact the performed and user experience

Ans While MFA is a functional requirement, several non functional requirements must be considered to ensure, security, performance and user-experience

Key- Non Functional Requirements :-

1. Security : → Encryption :- Credentials, authentication tokens, and session data must be encrypted using strong algorithm
→ Access Control :- Implement position based access control to restrict operations
→ Compliance :- Ensures adherence to industry standards

2.) Performance :- → Response time :- Authentication should be completed within 2-3 sec
→ Concurrent users :- The system should handle thousands of simultaneous login requests
→ Load Balancing :- Use distributed authentication servers to prevent bottleneck

3.) Availability :- → uptime Guarante :- Achieve 99% availability to ensure customers can access their accounts anytime
→ Redundancy :- Implement failsafe mechanisms to avoid downtime in case of server failure

4.) Usability and Accessibility :-

→ User Friendly MFA : offer multiple MFA options (SMS, e-mail) for convenience

→ Error handling :- Provide clear messages instead of generic "Login failed" response

→ Accessibility compliance :- Ensure compatibility with screen readers for difficulty-abled users.

Impact of ignoring Non-functional Requirements

1.) Security breaches :- Lack of encryption or weak authentication mechanisms could lead to data leaks, identify theft or fraud.

2.) Performance Issues :- An unstable system could crash under high traffic, especially during peak hours.

3.) Downtime :- Repeated downtime can lead to loss of customer trust and damage reputation.

4.) Poor User Experience :- Confusing errors messages or complicated MFA processes may make authentication frustating for customers

Q.3. "Software Development Project estimation is often laborious and time Consuming" Justify and explain different types of estimation techniques.

Ans Estimation of software development project is laborous and time consuming due to many factors :-

→ Unclear Requirements : Projects often begin with incomplete or evolving specifications.

→ Changing Scope : Frequent modifications impact cost and time techniques

→ Technology :- The availability of skilled resources and Technology choices affect estimation accuracy

→ Uncertainity :- Unexpected challenges, such as integration issues or security concerns, increase complexity

Due to these challenges organizations use different estimation techniques to improve accuracy and manage risks, a few of them are :.

1.) Size metrics :- It measures the magnitude or complexity of a software project. These metrics are used as a base for effort and cost estimation

(a) Common Size Metrics :-

(i) Lines of Code :- Measures total lines of source code

(ii) Function points :- Measures the system's functionality based on user interaction, inputs and outputs.

2.) Empirical estimation :- uses historical project data and mathematical models to predict effort, cost and ~~t~~ time. It relies on past experiences.

(a) Examples :-
(a) Constructive Cost Model :- Uses historical data and Project characteristi to estimate effort

3.) Heuristic Estimation :- uses rule of thumb or expert judgement to estimate project size, effort and cost. It is based on experience and intution rather than strict formulas

(a) Examples :-
(i) expert Judgement : Senior developers estimate based on past project
(ii) Delphi Technique :- A group of experts give independent estimates and Concensus is reached after multiple rounds.

4.) Analytic Estimation :- Uses mathematical and logical models to break down software estimation into structured components. It is based well defined formulas and step-by-step calculations

(a) Example :-
(i) Function Point Analysis :- Breaks software into functional components and assignes complexity points

(ii) Algorithmic Models :- uses formula to calculate effort based on inputs like project size and complexity

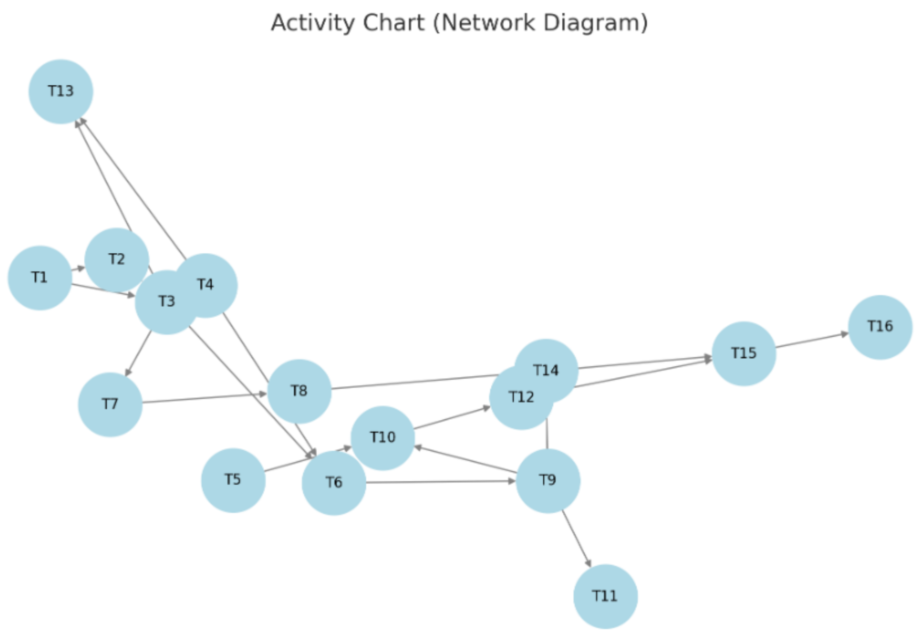
Q4. For the following tasks, their durations and dependencies make an activity chart and a GANT chart showing the Project schedule

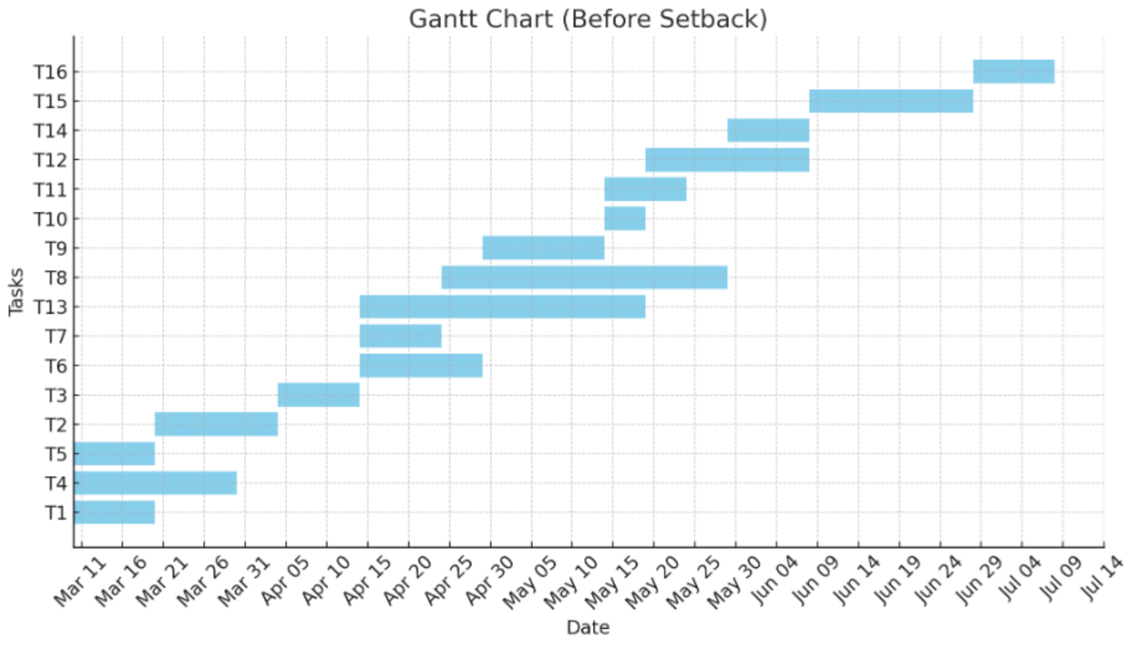
Task	Duration (days)	Dependencies
T1	10	—
T2	15	T1
T3	10	T1, T2
T4	20	—
T5	10	—
T6	15	T3, T4
T7	10	T3
T8	35	T7
T9	15	T6
T10	5	T5, T9
T11	10	T9
T12	20	T10
T13	35	T3, T4
T14	10	T8, T9
T15	20	T12, T14
T16	10	T15

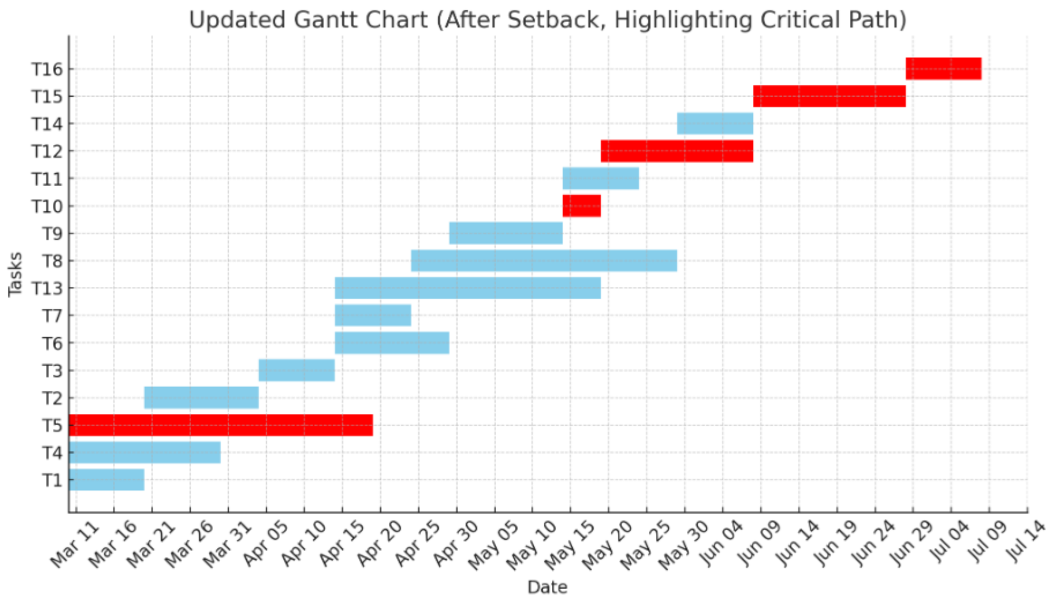
Assume that a serious, unanticipated setback occurs and instead of taking 10 days, task T5 takes 40 days. Revise the activity chart accordingly, highlighting the new critical path. Draw up the new bar charts showing how the project might be organized.

Ans

- Activity chart (Network Diagram)
- Gantt chart (Before setback)
- Updated Gantt Chart (After setback, Highlighting Critical Path)


Activity Chart (Network Diagram)
T13
T2
T1
T3
T4
T7
T8
T14
T12
T15
T16
T10
T5
T6
T9
T11


Gantt Chart (Before Setback)
Tasks
T16
T15
T14
T12
T11
T10
T9
T8
T13
T7
T6
T3
T2
T5
T4
T1
Date
Mar 11
Mar 16
Mar 21
Mar 26
Mar 31
Apr 05
Apr 10
Apr 15
Apr 20
Apr 25
Apr 30
May 05
May 10
May 15
May 20
May 25
May 30
Jun 04
Jun 09
Jun 14
Jun 19
Jun 24
Jun 29
Jul 04
Jul 09
Jul 14


Updated Gantt Chart (After Setback, Highlighting Critical Path)
Tasks
T16
T15
T14
T12
T11
T10
T9
T8
T13
T7
T6
T3
T2
T5
T4
T1
Date
Mar 11
Mar 16
Mar 21
Mar 26
Mar 31
Apr 05
Apr 10
Apr 15
Apr 20
Apr 25
Apr 30
May 05
May 10
May 15
May 20
May 25
May 30
Jun 04
Jun 09
Jun 14
Jun 19
Jun 24
Jun 29
Jul 04
Jul 09
Jul 14