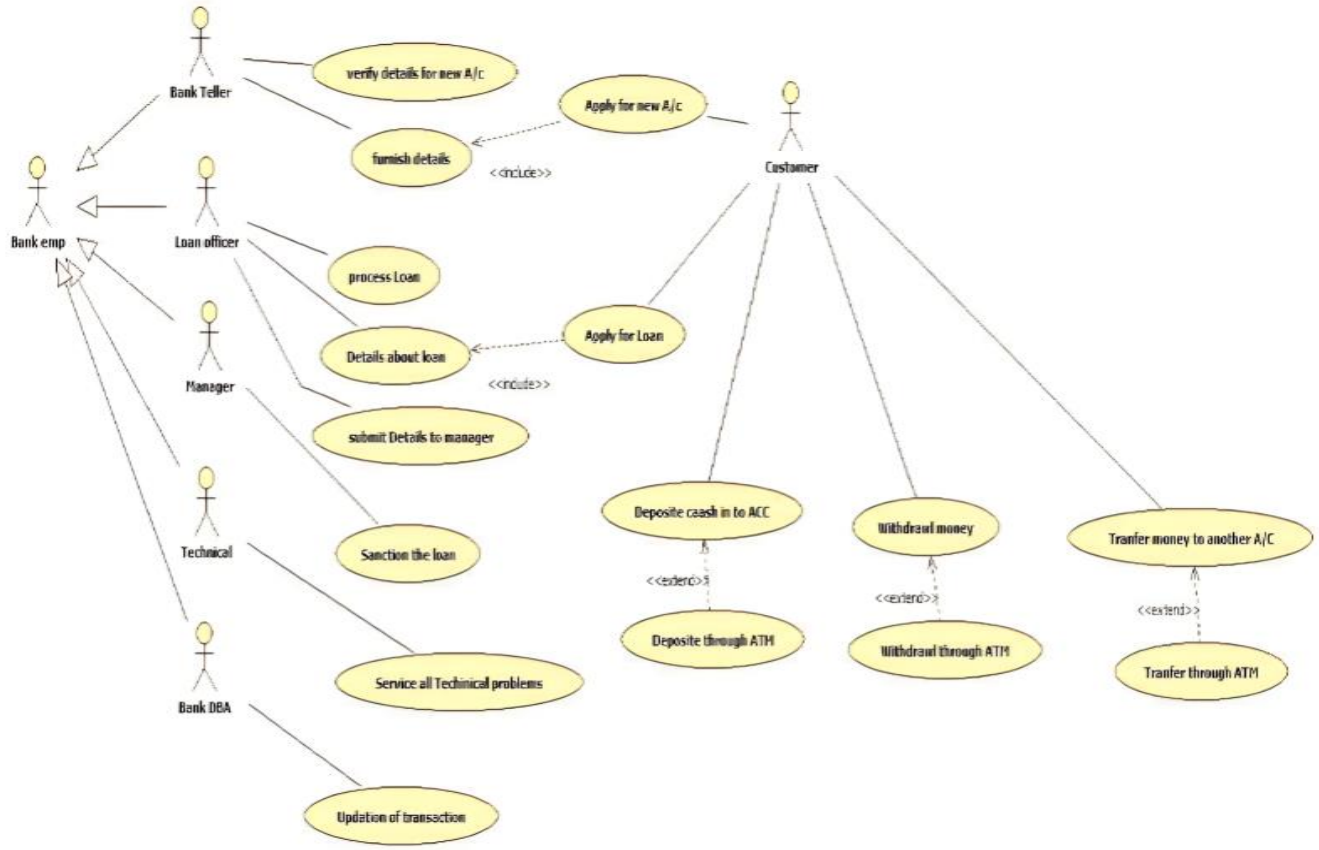
SOFTWARE ENGINEERING

1. Draw a use case diagram for banking system.



2. List the important principles behind Agile model. Compare its characteristics with RAD model.

Agile development is based on iterative and incremental processes. The key principles that drive the Agile model include:

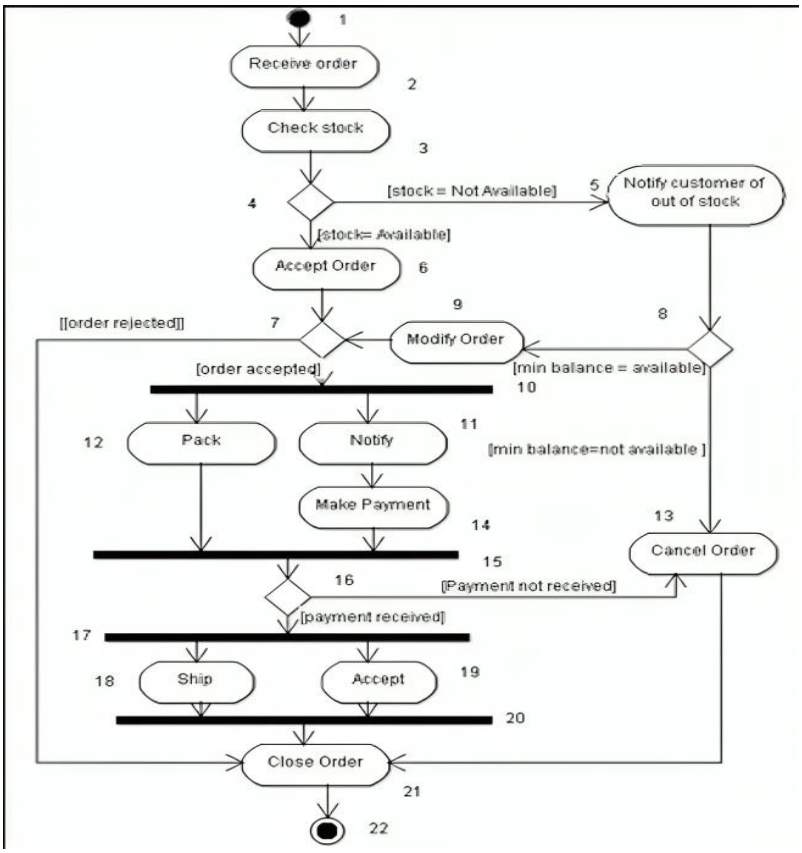
1. Agile emphasizes close cooperation with customers throughout the project lifecycle to ensure their needs are met.
2. Agile values face-to-face communication, frequent collaboration, and team autonomy.
3. Agile prioritizes delivering functional software in frequent iterations rather than producing extensive documentation.
4. Agile encourages flexibility, welcoming changes even in late stages of development to meet the customer's needs.
5. Deliver software in short cycles (often every 1-4 weeks) to obtain continuous feedback and improve.
6. Maximizing the amount of work not done is essential. Agile teams strive to develop just enough to satisfy immediate requirements.
7. Agile focuses on maintaining a constant pace to ensure long-term productivity without burnout.

8. Teams should have the autonomy to make decisions and organize themselves to achieve the best results.
9. Agile encourages regular retrospectives to reflect on the team's processes and improve efficiency.

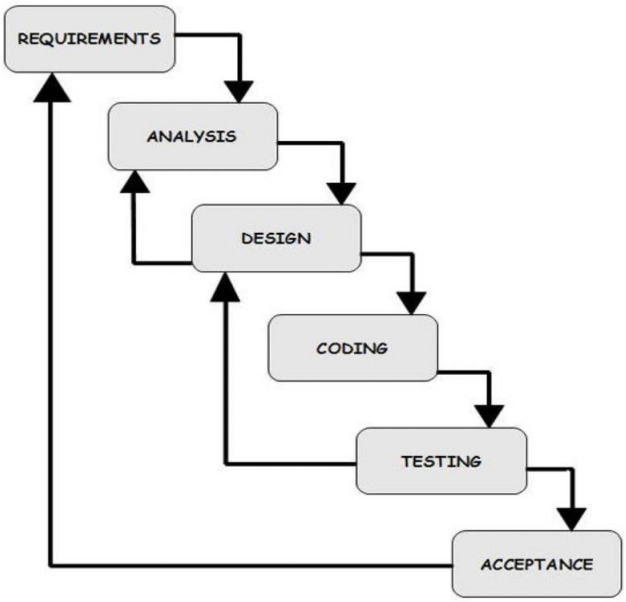
Comparing Agile Model with RAD (Rapid Application Development) Model:

Aspect	Agile Model	RAD Model
Focus	Continuous collaboration, flexibility, and iterative delivery	Fast development through prototyping and quick iterations
Customer Involvement	High involvement throughout the process	High involvement, especially during the initial stages
Development Time	Iterative development with frequent releases (short sprints)	Very fast, through use of prototypes and iterative delivery
Flexibility	Highly adaptable to changes, even late in the project	Adaptable, but less so than Agile; focuses on user feedback
Team Size	Small, cross-functional, and self-organizing teams	Typically smaller, but relies more on specialized teams
Documentation	Minimal, focus is on working software	Minimal documentation; functional prototypes are preferred
Iteration Length	Short (usually 1-4 weeks)	Short, but often not as structured as Agile's time-boxed sprints
Emphasis on Working Product	Working product after each sprint	Functional prototypes during early stages, with incremental refinement
Change Handling	Embraces changes at any stage	Can handle changes, but late changes may slow the prototyping cycle
Testing	Continuous testing throughout the cycle (integrated with development)	Testing primarily focused after prototypes are developed
Key Practices	User stories, Scrum, XP, Kanban, continuous feedback	Prototyping, user design sessions, iterative refinement

3. List the important principles behind Agile model. Compare its characteristics with RAD model. How you select a suitable life cycle model for a specific report?



4. Sketch out the waterfall model and compare its usage with the spiral model.



Aspect	Waterfall Model	Spiral Model
Complexity	The Waterfall model is simple and easy.	The spiral model is a lot more complex.
Development Method	The waterfall model works in a sequential method.	While the spiral model works in the evolutionary method.
Risk Management	In the waterfall model errors or risks are identified and rectified after the completion of stages.	In the spiral model errors or risks are identified and rectified earlier.

Project Size Suitability	The waterfall model is applicable for small projects.	While the Spiral model is used for large projects.
Planning	In waterfall model requirements and early-stage planning are necessary.	While in spiral model requirements and early-stage planning are necessary if required.
Flexibility to Change	Flexibility to change in waterfall model is Difficult.	Flexibility to change in spiral model is not Difficult.
Risk Level	There is high amount risk in waterfall model.	There is low amount risk in spiral model.
Cost	Waterfall model is comparatively inexpensive.	While cost of spiral model is very expensive.
Maintenance	It requires least maintenance.	It requires typical maintenance.
Framework Type	It is based on linear framework type.	It is based on linear and iterative framework type.
Testing	Testing is done after the coding phase in the development life cycle.	Testing is done after the engineering phase in the development cycle.
Reusability	Reusability is extremely unlikely.	To a certain extent, reusability is possible.

5. Given a small software team using XP practices, demonstrate how pair programming and continuous integration would improve the development of real-time messaging.

Pair programming involves two developers working together on a single piece of code. One developer writes the code while the other reviews the code in real-time. They frequently switch roles, ensuring continuous collaboration and oversight. **Pair Programming Improves Real-Time Messaging Development by:**

- **Faster Problem-Solving:** Pair programming ensures that any bugs or design flaws in the messaging system are spotted and resolved immediately, reducing the time spent on debugging.
- **Code Quality and Reduced Defects:** Real-time applications are sensitive to performance bottlenecks, concurrency, and security vulnerabilities. It helps write cleaner, more efficient, and robust code that handles high-volume, real-time message exchanges.
- **Knowledge Sharing:** The messaging app's components often require expertise in multiple areas. Pair programming enables team members to share knowledge and ensure that no single person becomes a bottleneck in understanding the system.

- **Improved Design:** While one developer is writing code for message delivery or handling user status updates, the other can think critically about the design and suggest improvements resulting in better design choices.

Continuous Integration is a practice where developers frequently merge code changes into the main codebase, followed by automated builds and tests to ensure that the application remains functional. **CI Improves Real-Time Messaging Development by:**

- **Frequent Integration Reduces Conflicts:** In a real-time messaging application, different developers might work on various features. CI ensures that these features are integrated frequently, reducing integration issues or merging conflicts that could arise when combining different pieces of code.

- **Automated Testing:** Real-time messaging apps need to be responsive and reliable. CI pipelines can include automated tests for message delivery, concurrency and security.

- **Faster Feedback Loop:** When a developer commits code that breaks the real-time message flow, CI catches these issues early. Developers get immediate feedback, allowing them to fix issues quickly before they affect the entire application.

- **Continuous Deployment:** In real-time messaging apps, user expectations for reliability and uptime are high. CI allows for frequent, automated deployments with minimal downtime.

NAME: TANISHTA

REGISTRATION NUMBER: 225811120 (ROLL NUMBER: 09)

DEPARTMENT: INFORMATION TECHNOLOGY (IT-B)