Go to repo browser – create new branch with create folder option

Remove target folder before commit.

Show unversioned files – show files present in local not on repo browser.

**Q:** You updated your working copy in Eclipse, but there’s a conflict in a file. How would you resolve it using Eclipse?

1. Right-click on the project → **Team → Update**
2. If a conflict appears, the file will show a red marker.
3. Open the conflicted file – Eclipse will show conflict markers:
4. Manually edit the file to merge changes.
5. Right-click the file → **Team → Mark Resolved**
6. Commit the changes:
   * **Team → Commit** → Add a commit message → **Finish**

**Q:** You committed the wrong file to SVN. How would you undo the commit in Eclipse?

1. Open **History View**:
   * Right-click project → **Team → Show History**
2. Right-click the wrong revision → **Revert Changes from Revision**
3. Eclipse will roll back to the previous state.
4. Commit the reverted state:
   * **Team → Commit**

**Q:** Two developers modified the same file in different branches. How would you merge changes and resolve conflicts using Eclipse?

1. Switch to the branch you want to merge into: **Team → Switch to Branch**
2. Start the merge: **Team → Merge** → Select the branch → **Finish**
3. If conflicts appear:
   * Right-click the conflicted file → **Edit Conflicts**
   * Manually resolve conflicts
   * Right-click the file → **Mark Resolved**
4. Commit the changes: **Team → Commit**

**Q:** You are working on a new feature. How would you create a new branch in Eclipse using SVN?

1. Right-click the project → **Team → Branch/Tag**
2. Enter the branch name and repository URL:

https://your-svn-repo/branches/feature-branch

1. Click **OK**
2. Switch to the new branch:
   * **Team → Switch to Branch**

**Q:** You modified a file but haven't committed it yet. How would you undo the local changes using Eclipse?

1. Right-click the file → **Team → Revert**
2. Confirm the revert action
3. File will be restored to the last committed state

**Q:** How would you check the history of a specific file in Eclipse?

1. Right-click the file → **Team → Show History**
2. Eclipse will display: Revision Number Author Date Commit Message

**Q:** How would you compare two revisions of a file in Eclipse?

**Team → Show History**

Select two revisions → **Compare with Each Other**

**Q:** How would you apply a patch created by another developer in Eclipse?

1. Right-click the project → **Team → Apply Patch**
2. Select the .patch file
3. Resolve any conflicts
4. Commit the changes:
   * **Team → Commit**

Flow –

BA team gathers the requirement from the officers.

Scrum Master conducts a **Sprint Planning** meeting.

Assign tasks to developers using:

* **Jira - Development Timeline**

We are Understanding business and technical requirements provided by the BA team and Lead. Clarify the doubts if any.

Then we are creating a new branch from the main repository. **Create a feature branch from the development branch.**

Implement it do the manual testing and then commit. Commit to feature branch. ~~Unit test~~

Create a Merge Request (MR) or Pull Request (PR). Lead will check the code review it

Incorporate feedback from the reviewer (Senior Developer/Lead).

 Fix any issues raised during the code review.

 Fix integration issues if the code fails after merging into the development branch.

 Support in resolving any dependency conflicts.

 Sprint Planning: Scrum Master creates a 2-week sprint.

 Development:

* Junior Developer creates a feature branch from the main branch.
* Develops and tests the code.

 Code Review: Code is reviewed and merged into the main branch.

 Sprint Review: Scrum Master organizes a review to demonstrate the feature.

 Retrospective: Team discusses what went well and what can be improved.

 Next Sprint: Start working on the next feature.

A **Story** is a high-level requirement or feature request written from the end-user's perspective.

A **Task** is a specific piece of work required to complete a story.

**Story:** "As a user, I want to log in using OTP..."  
**Tasks:**

* Create OTP generation service → **(3 hours)**
* Create UI for OTP input → **(2 hours)**
* Write unit tests → **(2 hours)**

**Unit testing - how each method or function is working**

**Integration testing - combing all method and then test. Check different parts of application working together.**

**Git –** Version control system is a tool helps to track changes in code.

Advantages –

1. To track the history of code.
2. To collabrate

**Difference Between Agile and SDLC:**

* **SDLC** → Defines the overall process of software development.
* **Agile** → A methodology that can be used within SDLC for faster and flexible development.

**What is Agile?**

**Agile is a software development method that delivers work in small fast interactions with continuous feedback**

**Agile** is a **software development methodology** that focuses on:

* **Incremental and iterative** development.
* Delivering software in **small, workable units**.
* Encourages **continuous feedback** and **improvement**.
* Promotes **collaboration** between developers, testers, and stakeholders.
* Adapts to changing requirements quickly.

**Key Principles of Agile:**

* Customer satisfaction through early and continuous delivery.
* Welcome changing requirements even late in development.
* Frequent delivery of working software.
* Close collaboration between business and developers.
* Self-organizing teams.
* Continuous improvement.

**Example Agile Frameworks:**

* **Scrum** – Work divided into sprints (2–4 weeks).
* **Kanban** – Visual board to manage work progress.

**What is SDLC (Software Development Life Cycle)?**

**SDLC** is a structured process for developing software in **phases** to ensure high quality and efficiency.  
It defines the **end-to-end development process** from planning to maintenance.

**Phases of SDLC:**

1. **Planning** – Define project scope, goals, and resources.
2. **Requirement Gathering** – Collect functional and non-functional requirements.
3. **Design** – Create architecture, UI/UX, and system design.
4. **Development** – Write code based on design.
5. **Testing** – Identify and fix bugs (unit, integration, system testing).
6. **Deployment** – Release to production.
7. **Maintenance** – Fix issues and improve performance after release.