1. What is Agile?

Agile software development is an umbrella term for a set of frameworks and practices based on the values and principles expressed in the Manifesto for Agile Software Development and the 12 Principles behind it.

Agile methodologies

The Agile methodology is a project management approach that involves breaking the project into phases and emphasizes continuous collaboration and improvement.

The Agile Manifesto

The Manifesto for Agile Software Development' suggests that it is only applicable to software development, the values and principles described in the Manifesto can easily be applied to the development of many types of product.

The Manifesto describes 4 values and 12 supporting principles.

The four Agile Manifesto values are:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

The 12 Agile Manifesto principles, expanding on the original manifesto, include:

1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. The project team welcomes changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. The process builds projects around motivated individuals, giving them the environment and support they need, and trusts them to get the job done.
6. A face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.
7. Working software is the most important measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should maintain a constant pace indefinitely.
9. Pay continuous attention to technical excellence, and good design enhances agility.

10. Simplicity is essential. This is the art of maximizing the amount of work not done.
11. Self-organizing teams produce the best architectures, requirements, and designs.
12. At regular intervals, the team reflects on how to become more effective and adjusts its behavior accordingly.

Types of Agile methodologies

Agile project management is not a singular framework but an umbrella term that includes a wide range of methodologies, including Scrum, Kanban, Extreme Programming (XP), and the Adaptive Project Framework (APF).

- **Scrum:** It is ideal for projects with rapidly changing requirements, using short sprints.
- **Kanban**: It visualizes project progress and is great for tasks requiring steady output.
- **Lean:** It streamlines processes, eliminating waste for customer value.
- **Extreme Programming (XP):** It enhances software quality and responsiveness to customer satisfaction.
- **Adaptive Project Framework (APF):** Works well for projects with unclear details, as it adapts to constantly evolving client needs.

Benefits Of Using Agile

- **Rapid progress:** By effectively reducing the time it takes to complete various stages of a project, teams can elicit feedback in real time and produce working prototypes or demos throughout the process
- **Customer and stakeholder alignment:** Through focusing on customer concerns and stakeholder feedback, the Agile team is well positioned to produce results that satisfy the right people
- **Continuous improvement:** As an iterative approach, Agile project management allows teams to chip away at tasks until they reach the best end result

2. Scrum

The Scrum process encourages practitioners to work with what they have and continually evaluate what is or is not working. Good communication is essential and is carried out through meetings, called "events."

Scrum events include the following:

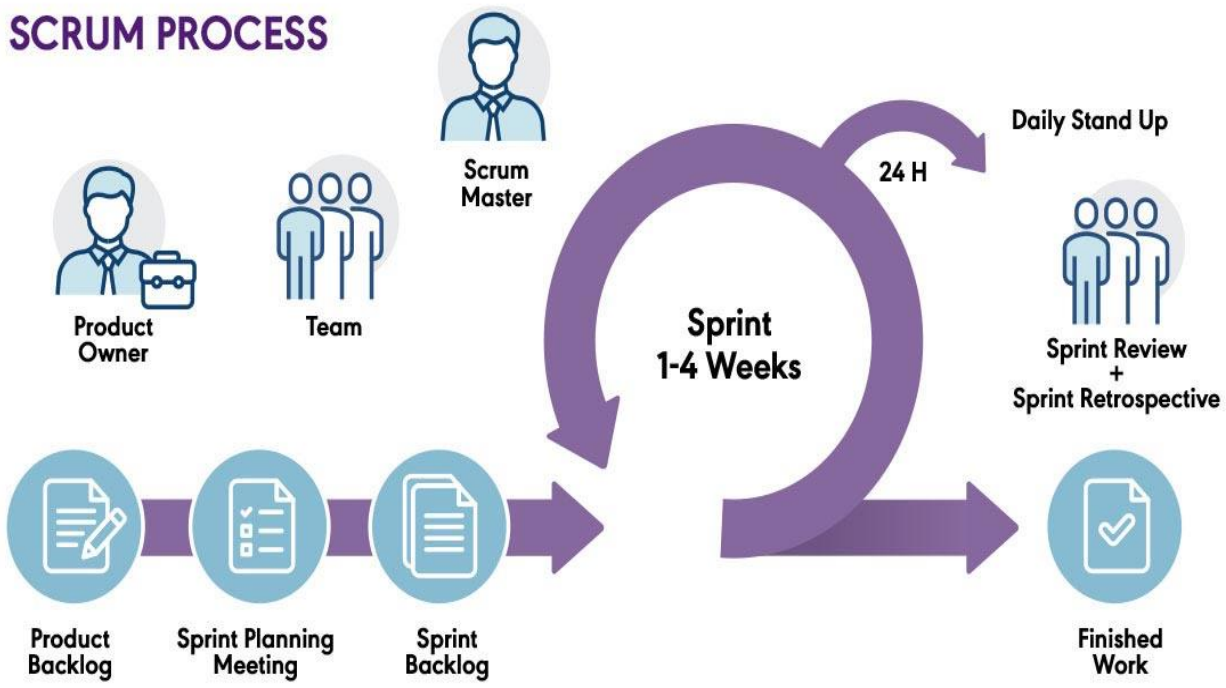
- **Daily Scrum.** This event is a short, stand-up daily meeting that takes place in the same place and time each day. In these meetings, the team reviews work accomplished the previous day and plans what will be done in the next 24 hours. This is the time when team members discuss problems that might prevent project completion.
- **Sprint.** A Sprint is the time frame in which work must be completed -- often 30 days. New Sprints start right after the end of the previous one.
- **Sprint Planning Meeting**. In these meetings, everyone participates in setting goals. At the end, at least one increment -- a usable piece of software -- should be produced.
- **Sprint Review.** This is the time to show off the increment.
- **Sprint Retrospective.** A Sprint Retrospective is a meeting held after a Sprint ends. During this meeting, everyone reflects on the process. A team-building exercise may also be offered. An important goal of this event is continuous improvement.

Scrum roles

- **Product owner** The product owner has a core role in the Scrum workflow. They guide agile team discussions about product backlog items and features. In addition, product owners guide quality assurance to make sure deliverables are up to par.
- **Scrum Master** The Scrum Master will closely follow the principles in the agile manifesto to support sprint planning. Scrum masters guide development teams through agile methods to add value for stakeholders.
- **Software development team** Development teams are skilful and cross-functional. Teams that work in agile software development environments will typically include designers, developers, testers, and others to prevent the need for external assistance.

Steps of the Scrum workflow

The Scrum framework consists of several key steps or events that guide the development process. Here are the main steps in the Scrum workflow:



1. **Backlog development**

   A product roadmap guides team members in creating user stories and product requirements, which make up the sprint backlog. In the backlog, teams propose a list of features or user stories that the team must deliver. Product owners decide which features will make up the backlog.

2. **Backlog release**

   Produce owner and team collaboration now decide which user stories will make it into each backlog release. Each backlog release is the completion of a smaller set of activities which eventually make up a sprint release. After completing this planning and setting timeframes for each action item, team members choose specific features for each sprint.

3. **Sprint work**

   In a sprint, team members complete a set of backlog tasks within predetermined timeframes (usually 14-28 days). During this time, the agile team builds the product features from a specific sprint backlog.

   Scrum or sprint meeting

   Teams also hold Scrum or sprint meetings. During sprint meetings, the team sets a sprint goal (usually work on a specific feature). They agree on which product backlog items to complete in

order to complete this product iteration. The team will prioritize, plan, and estimate the time needed to complete each task within the sprint.

Daily stand-ups

Agile teams use these daily standup meetings to track their agile workflow towards meeting sprint goals. Daily standup meetings are typically held — naturally — standing up, as they should last no more than 15 minutes. Standup meetings help teams discuss solutions to daily work issues.

4. The burndown chart

Team members can use software to create their burndown charts. Burndown charts show original time estimates compared to real-time activities, which shows where expectations or team resources need to be adjusted.

5. Testing

During testing, the team demonstrates product functionalities for stakeholders. Feedback from product testing guides any needed changes.

6. Sprint retrospective and follow-up planning

The final phase of the workflow is to hold a sprint retrospective. Sprint retrospectives are post-mortems on the previous workflow. At this stage, agile teams question what they did well, what didn't go as they hoped, and what changes they should make in the next sprint. Groups hold these sprint retrospectives to concentrate on better value deliverables through continuous improvement.

3. Scrum Work Flow for Student Budgeting Mobile Application

Implementing Scrum for a student budgeting system mobile application involves breaking down the development process into iterative cycles called sprints. Here's a simplified Scrum workflow for your project:

1. Backlog Development

The project team collects the user requirements, list the features and activities of the app

Features of Student Budgeting App

- Student registration
- Storing the student information
- Allowing the student to enter the expenses
- Categorizing the expenses as money spent on food, Stationary and other expenses
- Calculating the total expenses and display savings at the end of month.

2. Backlog Release

The project team finalize the features and activities of the app are:

- Student registration
- Storing the student information
- Allowing the student to enter the expenses
- Categorizing the expenses as money spent on food, Stationary and other expenses
- Calculating the total expenses and display savings at the end of month.

3. Sprint Work

Sprint 1: Initial Setup and User Input

- Planning and Setup

  Tasks:
  - Set up the project structure and version control.
  - Define data models for storing student information.
  - Create a simple user interface for data input.
- User Information Input

  Tasks:
  - Implement a form for users to input their personal information (name, student email).
  - Validate and store the entered information in the database.
- Expense Entry

  Tasks:
  - Design a form for entering daily expenses.
  - Allow users to add details like expense category, amount, and description.

- Store the entered expense data in the database.

Sprint 2: Expense Categorization and Total Expenses Calculation

- Expense Categorization

  Tasks:

  - Enhance the expense entry form to include a category selection (food, stationary, etc.).
  - Implement logic to categorize and store expenses accordingly.
- Total Expenses Calculation

  Tasks:

  - Create a function to calculate total expenses based on the entered data.
  - Display the total expenses on the user interface.
- Savings Calculation

  Tasks:

  - Implement a function to calculate savings (income - total expenses).
  - Display the calculated savings on the user interface.
- UI Refinements and Testing

  Tasks:

  - Refine the user interface for a better user experience.
  - Conduct thorough testing to identify and fix any bugs or issues.

4. Burndown Chart

A burndown chart is a graphical representation of work completed versus time. In the student budgeting app, the burndown chart can help the team and owner to understand how well they are progressing towards completing the tasks planned for the Sprint.

5. Testing

Integrating each activity like combing all the categorize of expenses and calculating the total expenses and displaying the savings. Testing the App whether it meets all user requirements or not.

6. Sprint retrospective and follow-up planning

After completing a Sprint, the team gathers for a Sprint Retrospective meeting. This meeting is an opportunity for the team to reflect on the recently concluded Sprint and discuss various aspects of their work.

The key components of a Sprint Retrospective for the student budgeting app could include:

- **What Went Well:** Identify and discuss the aspects of the Sprint that went well. This could include successful implementations, efficient collaboration, or positive outcomes. Implanting the all activities of student budgeting system.

- **Areas for Improvement:** Discuss the challenges and areas where the team could have performed better. This might involve addressing obstacles, refining processes, or enhancing communication.
- **Action Items:** Identify specific action or improvements that the team can implement in the next Sprint. These action items should be actionable, measurable, and time-bound.
- **Celebrations:** Acknowledge achievements and celebrate successes, fostering a positive team culture.

Follow-up Planning:

- **Implementing Changes:** In response to the insights gained from the Sprint Retrospective, the team is committed to implementing changes that will enhance the efficiency and effectiveness of the development process. This could involve refining the workflow, optimizing collaboration methods, or addressing specific challenges encountered during the previous Sprint. If the expense entry process proved to be cumbersome, the team might streamline the user interface for better user experience.
- **Adjusting the Backlog**: The retrospective findings are used to reassess and adjust the Product Backlog for the student budgeting app. Tasks are reprioritized, and new items may be added to accommodate lessons learned from the previous Sprint.
- **Communication:** Transparency is pivotal for continuous improvement. The insights derived from the retrospective are communicated openly and thoroughly to all team members. This includes sharing what went well, areas identified for improvement, and the specific action items to be implemented. Open communication fosters a collaborative environment, ensuring everyone is aligned with the proposed changes and improvements.
- **Training and Skill Development:** If the retrospective reveals areas where team members feel they need additional training or skill development, these considerations are incorporated into the plan for upcoming Sprints. For instance, if there are new technologies or tools identified during the retrospective that could enhance the development process, the team may plan training sessions to upskill team members.
- **Setting Goals for the Next Sprint:** Armed with the insights and improvements identified in the retrospective, the team collaboratively establishes clear goals and objectives for the upcoming Sprint. These goals align with the overarching project objectives and take into account the lessons learned from the retrospective. For example, if the previous Sprint highlighted the need for more accurate expense categorization, a goal for the next Sprint might involve implementing an enhanced categorization system.