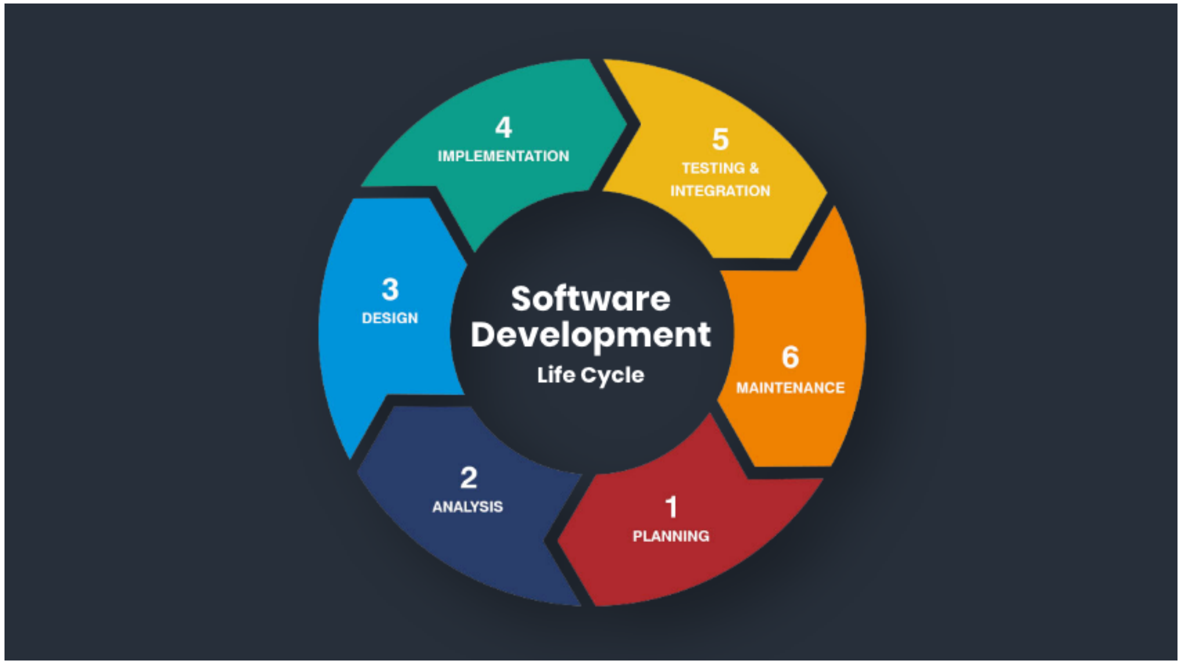
Introduction to SDLC

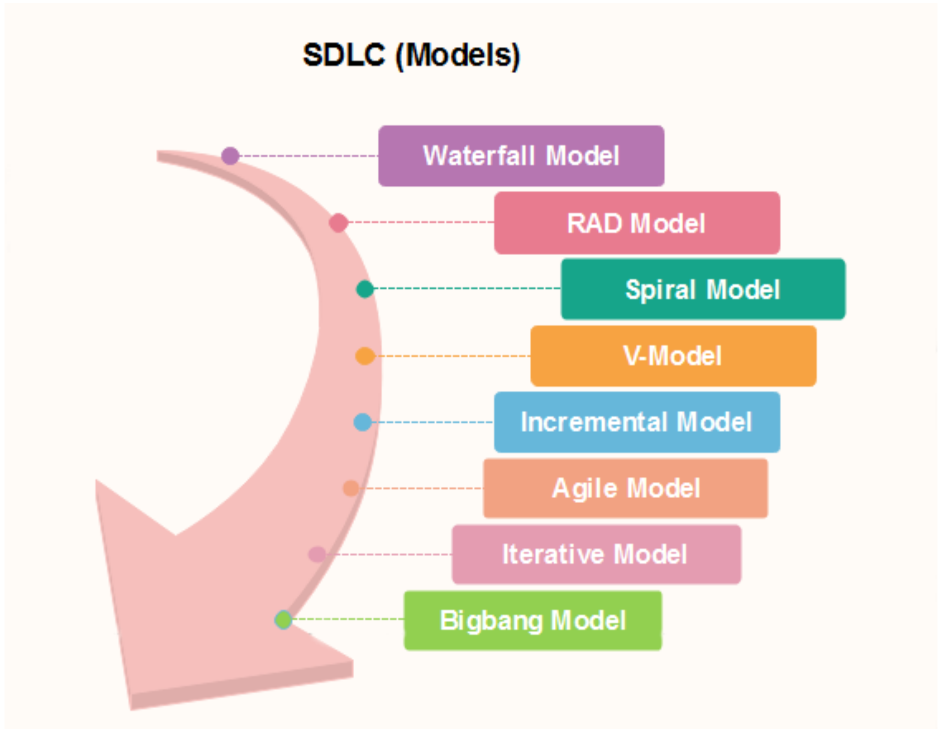
- Software Development Life Cycle (SDLC) is a structured process for developing software applications.

- Phases:
- 1. Planning
- 2. Analysis
- 3. Design
- 4. Implementation
- 5. Testing
- 6. Deployment
- 7. Maintenance



Types of SDLC Models

- There are different models in SDLC, each with its unique approach to development:
- - Waterfall Model
- - V-Model
- - Iterative Model
- - Spiral Model
- - Agile Model

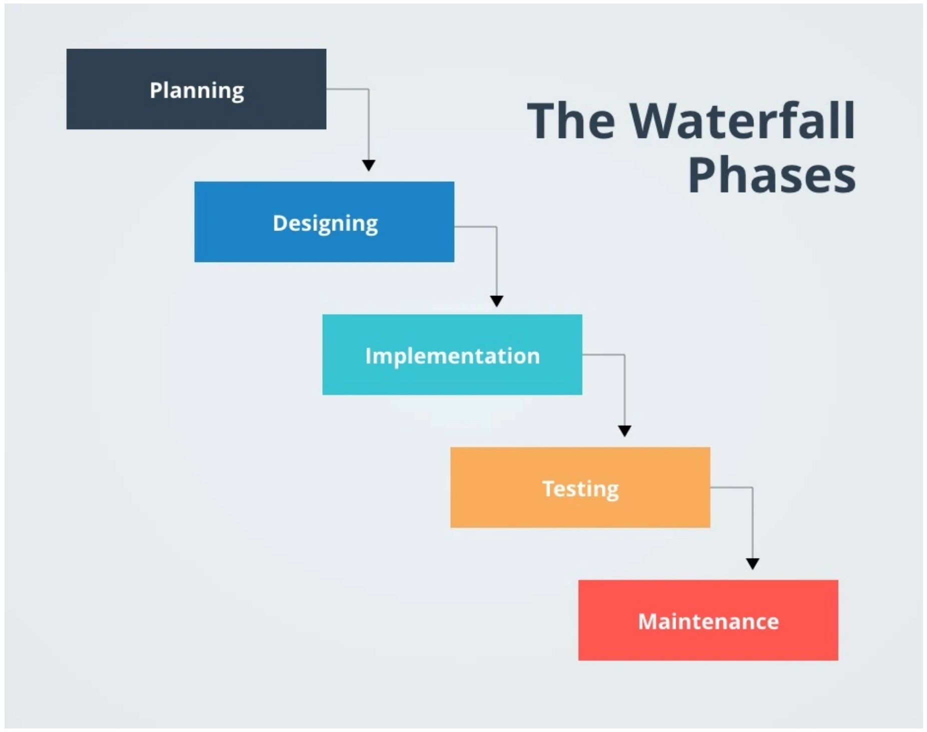


Waterfall Model

- The **Waterfall Model** is a **linear and sequential** approach to software development, where each phase must be completed before moving to the next. It follows a top-down approach, meaning there is no going back once a phase is finished. This model is best suited for projects with well-defined requirements and minimal expected changes.

- Example: Imagine building a **house** using the Waterfall Model:

1. You first plan the house design.
2. Get approvals and create blueprints.
3. Start construction from foundation to roof.
4. Conduct safety inspections and final touches.
5. Finally, you move in and maintain the house over time.



Waterfall Model - Example

Requirement Gathering & Analysis

- All project requirements are collected and documented.
- Example: A bank wants a secure online banking system with specific features.

System Design

- The software architecture and technical specifications are created.
- Example: Designing the database schema, UI layouts, and APIs.

Implementation (Coding)

- Developers write the actual code based on the design.
- Example: Developers build login functionality for the online banking system.

Testing

- The system is tested to identify and fix bugs.
- Example: Checking if transactions are processed correctly in the banking system.

Deployment

- The software is released for end-users.
- Example: The online banking portal is made available to customers.

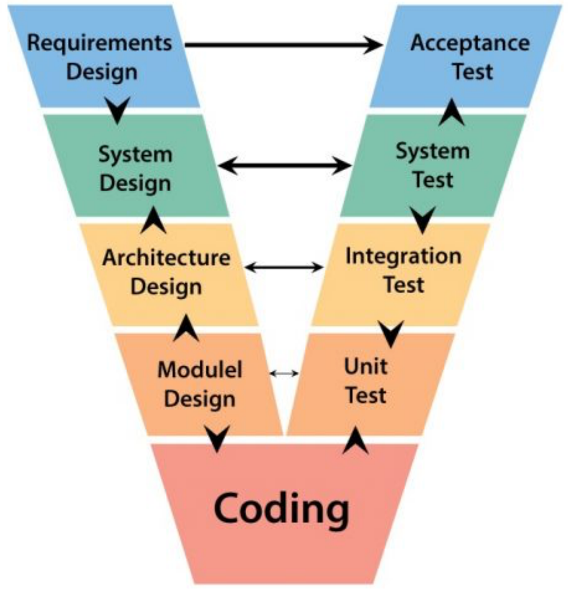
Maintenance

- Issues reported by users are fixed, and updates are provided.
- Example: Adding a two-factor authentication feature after feedback.

V-Model (Verification & Validation)

The **V-Model**, also called the **Verification and Validation Model**, is an extension of the **Waterfall Model** where each development phase is directly linked to a corresponding **testing phase**. It follows a **V-shaped** structure, ensuring that testing is planned alongside development.

Unlike the Waterfall Model, where testing happens at the end, the V-Model integrates **testing at every stage**, making it more reliable for **high-quality software development**



V-Model - Example

1. Verification (Left Side of the "V")

This part ensures that the system is **designed correctly** before development begins.

1. **Requirement Analysis → Acceptance Testing**
   - Gather all user and business requirements.
   - Example: A hospital wants an appointment scheduling system.
   - **Test Plan**: Acceptance tests will check if all features meet customer needs.
2. **System Design → System Testing**
   - Define the software architecture, components, and interfaces.
   - Example: Designing database structures and user access roles.
   - **Test Plan**: System tests will verify overall performance and security.
3. **High-Level Design (HLD) → Integration Testing**
   - Identify main software modules and their interactions.
   - Example: The appointment booking system should connect with the doctor's calendar.
   - **Test Plan**: Integration tests will check if different modules work together.
4. **Low-Level Design (LLD) → Unit Testing**
   - Break down modules into smaller components.
   - Example: Implementing functions for date selection in the booking system.
   - **Test Plan**: Unit tests will check if each function works correctly.

2. Validation (Right Side of the "V")

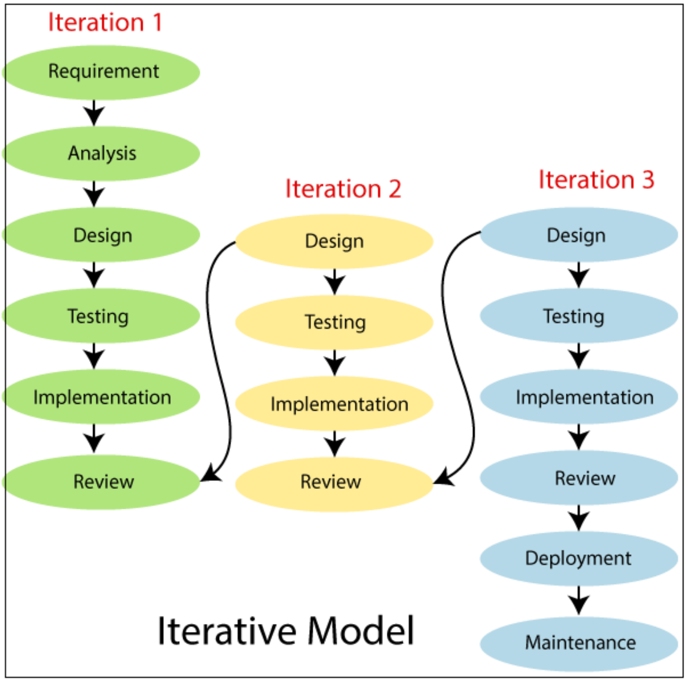
This part ensures that the system **works correctly** after development.

1. **Unit Testing**
    - Each function and module is tested individually.
    - Example: Verifying if the booking confirmation email is sent correctly.
2. **Integration Testing**
    - Modules are tested together to ensure smooth interactions.
    - Example: Checking if patient details are correctly stored in the database.
3. **System Testing**
    - The entire system is tested for functionality, performance, and security.
    - Example: Ensuring that thousands of users can book appointments without crashing.
4. **Acceptance Testing**
    - The client tests the system to confirm it meets business requirements.
    - Example: A hospital staff member tests the system before launching it.
    -

Iterative Model

The **Iterative Model** is a software development approach where the system is **developed and improved step by step** through multiple iterations (repetitions). Instead of building the entire system at once, **small parts** of the software are developed, tested, and refined **in cycles**.

Each iteration results in a **working prototype**, which is reviewed and improved in the next cycle until the final product is complete. This model is especially useful when **requirements evolve** over time.



Phases of the Iterative Model

1. **Planning & Requirement Analysis**

- Initial project requirements are gathered, but they do **not need to be fully defined** at the start.
- Example: A company wants an **e-commerce website** but is unsure of all features.

2. **Design & Development (First Iteration)**

- A **basic version** of the system is designed and developed.
- Example: The e-commerce website starts with only a **homepage and product listing**.

3. **Testing & Feedback**

- The first version is **tested**, and user feedback is collected.
- Example: Users request a **shopping cart feature**.

4. **Refinement (Next Iteration)**

- Based on feedback, **new features** are added in the next iteration.
- Example: A **shopping cart and checkout system** are introduced.

5. **Repeat Steps Until Completion**

- The process continues **iteratively**, improving the system in each cycle.
- Example: **Payment gateway, order tracking, and customer reviews** are added over several iterations.

Real-Life Example

Imagine **building a mobile app** for food delivery.

1. **First Iteration**: The app starts with just **restaurant listings**.
2. **Second Iteration**: A **menu and ordering system** are added based on feedback.
3. **Third Iteration**: A **payment gateway** is integrated.
4. **Final Iteration**: The app is fully refined with **real-time order tracking and reviews**.

Instead of waiting months for a **complete product**, the app is usable and **gradually improves** with each iteration.

Spiral Model

The **Spiral Model** is a **risk-driven** software development process that combines elements of both the **Iterative Model**and the **Waterfall Model**. It is best suited for **large, complex, and high-risk projects** where requirements are unclear and may change frequently.

Instead of following a strict linear approach, the **Spiral Model progresses in loops (spirals)**, with each loop representing a phase of development. Each spiral cycle goes through four main phases:

1. **Planning**
2. **Risk Analysis**
3. **Development & Testing**
4. **Evaluation & Review**

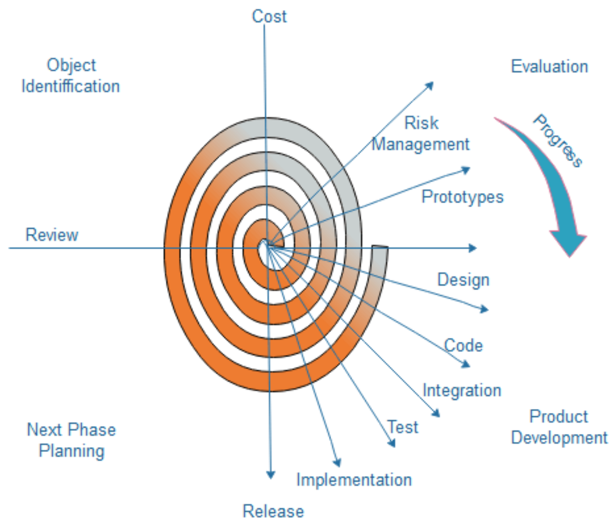


Fig. Spiral Model

How the Spiral Model Works (Example with a Banking App)

1. **First Spiral (Basic Functionality)**
    - Develop a **basic banking app** with login and balance checking.
    - Identify security risks and test login functionality.
2. **Second Spiral (Feature Expansion)**
    - Add features like **fund transfer** and **transaction history**.
    - Analyze potential fraud risks and implement security measures.
3. **Third Spiral (Enhancements & Refinements)**
    - Introduce a **mobile payment system** and **loan application** feature.
    - Test user experience and collect feedback.
4. **Final Spiral (Deployment & Maintenance)**
    - Launch the **full-featured** banking app and perform ongoing updates.

What is the Agile Model?

The **Agile Model** is a **flexible, iterative, and customer-centric** approach to software development. It emphasizes **continuous delivery**, **collaboration**, and **quick adaptation to changes** rather than following a rigid, step-by-step process like the **Waterfall Model**.

Key Features of Agile

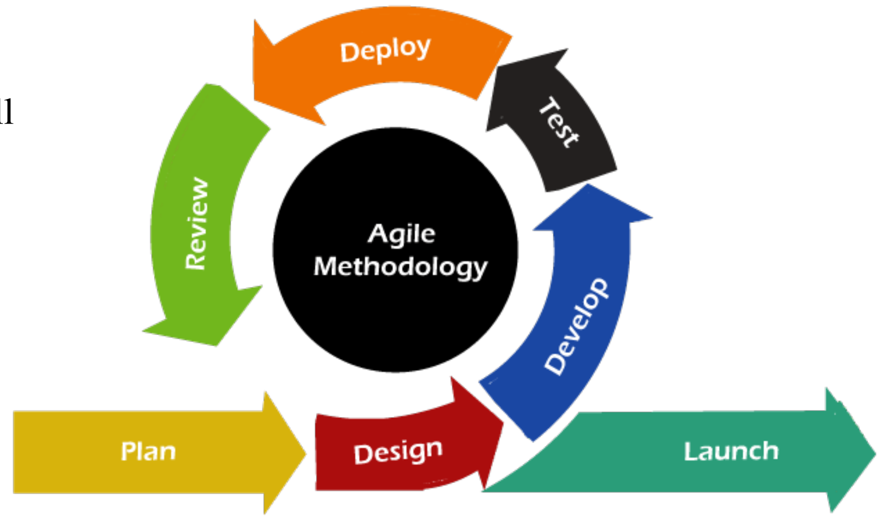
✅ **Iterative Development** – Software is developed in small cycles called **iterations (sprints)**.

✅ **Customer Involvement** – Continuous feedback ensures the product meets user needs.

✅ **Flexibility** – Requirements can evolve throughout the project.

✅ **Faster Delivery** – Working software is released frequently instead of waiting for a final product.

✅ **Collaboration** – Developers, testers, and stakeholders work together.



How Agile Works?

1. Divide Work into Sprints

- A project is broken down into **small, manageable tasks**.
- Each task is completed in a **sprint** (a time-boxed development cycle, usually 2-4 weeks).
- **Example:** A company wants a new **food delivery app**.

2. Prioritize Requirements

- The most important features are developed **first**.
- Features are based on **user stories** (descriptions of what the user needs).
- **Example:** The first sprint focuses on **restaurant listings and user registration**.

3. Develop, Test, and Review

- After each sprint, the software is tested and demonstrated to stakeholders.
- Feedback is collected, and improvements are made.
- **Example:** After launching the **restaurant listing feature**, users request a **search filter**.

4. Release a Working Product Frequently

- A **working version** of the software is released every few weeks.
- New features are added in future sprints.
- **Example:** After multiple sprints, the app now has **ordering, payment, and tracking** features.

5. Continuous Improvement

- The team reviews what worked well and what needs improvement.
- This process repeats until the final product is fully developed.
- **Example:** The team realizes users want **real-time delivery tracking**, so they plan it for the next sprint.
-

Agile Frameworks (Popular Agile Methods)

1. Scrum

- Uses **sprints** (fixed development cycles).
- A **Scrum Master** manages the process.
- **Example:** A team developing a mobile game plans work in **2-week sprints**.

2. Kanban

- Uses a **visual board** to track progress.
- Tasks move through different stages (**To Do → In Progress → Done**).
- **Example:** A marketing team tracks their campaign tasks using a **Kanban board**.

3. Extreme Programming (XP)

- Focuses on **frequent releases** and **continuous feedback**.
- **Example:** A financial software company releases **updates every week** to fix issues.

Real-Life Example of Agile Development

Example: Developing an Online Shopping Website

1. **Sprint 1:** Basic website with product listings.
2. **Sprint 2:** Add shopping cart functionality.
3. **Sprint 3:** Implement payment system.
4. **Sprint 4:** Introduce customer reviews and ratings.

Instead of waiting **months for a full product**, customers can start using **early features** while developers improve the system over time.

Traditional Methodologies

Traditional methodologies like **Waterfall, V-Model, Iterative, and Spiral** follow a structured, step-by-step approach.

Characteristics:

✔ **Well-defined stages** – Each phase must be completed before the next starts.
✔ **Emphasis on documentation** – Detailed project plans, requirement documents, and design documents.
✔ **Minimal flexibility** – Difficult to incorporate changes once development starts.
✔ **Late testing** – Bugs are identified and fixed at the end of the process.

Example Scenario:

- A **banking application** where security and compliance are crucial, and the project requirements remain fixed.

Agile Methodologies

Agile methodologies like **Scrum, Kanban, and Extreme Programming (XP)** focus on iterative development, delivering small functional parts of the software frequently.

Characteristics:

✔ **Iterative & Incremental** – The project is divided into small cycles (Sprints).
✔ **Customer Collaboration** – Continuous feedback is received throughout the development process.
✔ **Flexibility** – Changes can be made at any stage.
✔ **Early & Continuous Testing** – Bugs are detected and fixed early.

Example Scenario:

- **E-commerce platforms** like Amazon or Flipkart, where new features are frequently added based on customer feedback.

Agile vs. Traditional – When to Use Which?

Factor	Traditional (Waterfall, V-Model, Spiral)	Agile (Scrum, Kanban, XP)
Project Size	Large, complex projects	Small to medium projects
Requirement Stability	Fixed and well-defined	Frequently changing
Customer Availability	Low	High
Time Sensitivity	Less important	Crucial
Testing Approach	After development	Continuous
Delivery Speed	Slow (Long-term delivery)	Fast (Frequent releases)

🔷 **Use Traditional Methodologies** when requirements are **fixed**, and documentation is a priority (e.g., Banking, Healthcare, Government Projects).

🔷 **Use Agile** when the project requires **frequent updates**, customer collaboration, and faster releases (e.g., Startups, Mobile Apps, SaaS).

Scrum Framework in Agile

Scrum is one of the most widely used **Agile frameworks** that focuses on **iterative and incremental development**. It helps teams deliver high-quality products quickly by working in **short, time-boxed cycles** called **Sprints**.

1. Key Elements of Scrum

1️⃣ **Product Owner** – Represents the customer and defines the product requirements.
2️⃣ **Scrum Master** – Facilitates the Scrum process and removes obstacles for the team.
3️⃣ **Development Team** – A self-organizing team that builds the product incrementally.
4️⃣ **Product Backlog** – A prioritized list of features and tasks to be completed.
5️⃣ **Sprint** – A time-boxed iteration (typically 1-4 weeks) where a set of tasks is completed.
6️⃣ **Daily Scrum (Stand-up Meeting)** – A short 15-minute meeting where team members discuss progress, challenges, and next steps.
7️⃣ **Sprint Planning** – A meeting to decide what work will be completed in the Sprint.
8️⃣ **Sprint Review** – A meeting at the end of the Sprint where the completed work is demonstrated.
9️⃣ **Sprint Retrospective** – A meeting where the team reflects on what went well and what can be improved.

2. Scrum Workflow

1️⃣ **Product Backlog Creation** – The Product Owner creates a list of features & requirements.
2️⃣ **Sprint Planning** – The team selects items from the Product Backlog for the Sprint.
3️⃣ **Sprint Execution** – The team works on tasks during the Sprint, with daily stand-ups.
4️⃣ **Sprint Review** – The completed product increment is demonstrated to stakeholders.
5️⃣ **Sprint Retrospective** – The team discusses lessons learned and improvements.
6️⃣ **Next Sprint Begins** – The cycle repeats until the product is complete.

3. Example Scenario

📌 **E-commerce Website Development**:

- The **Product Owner** defines features like Login, Search, Cart, and Checkout.
- The team picks **Login and Search** for the first **Sprint (2 weeks)**.
- Daily Stand-ups track progress, and at the end of the Sprint, Login and Search are reviewed.
- The team improves the process in the **Sprint Retrospective** and moves to the next Sprint for the **Cart and Checkout** features.

Benefits of Scrum

✅ **Faster Delivery** – Working features are delivered in every Sprint.

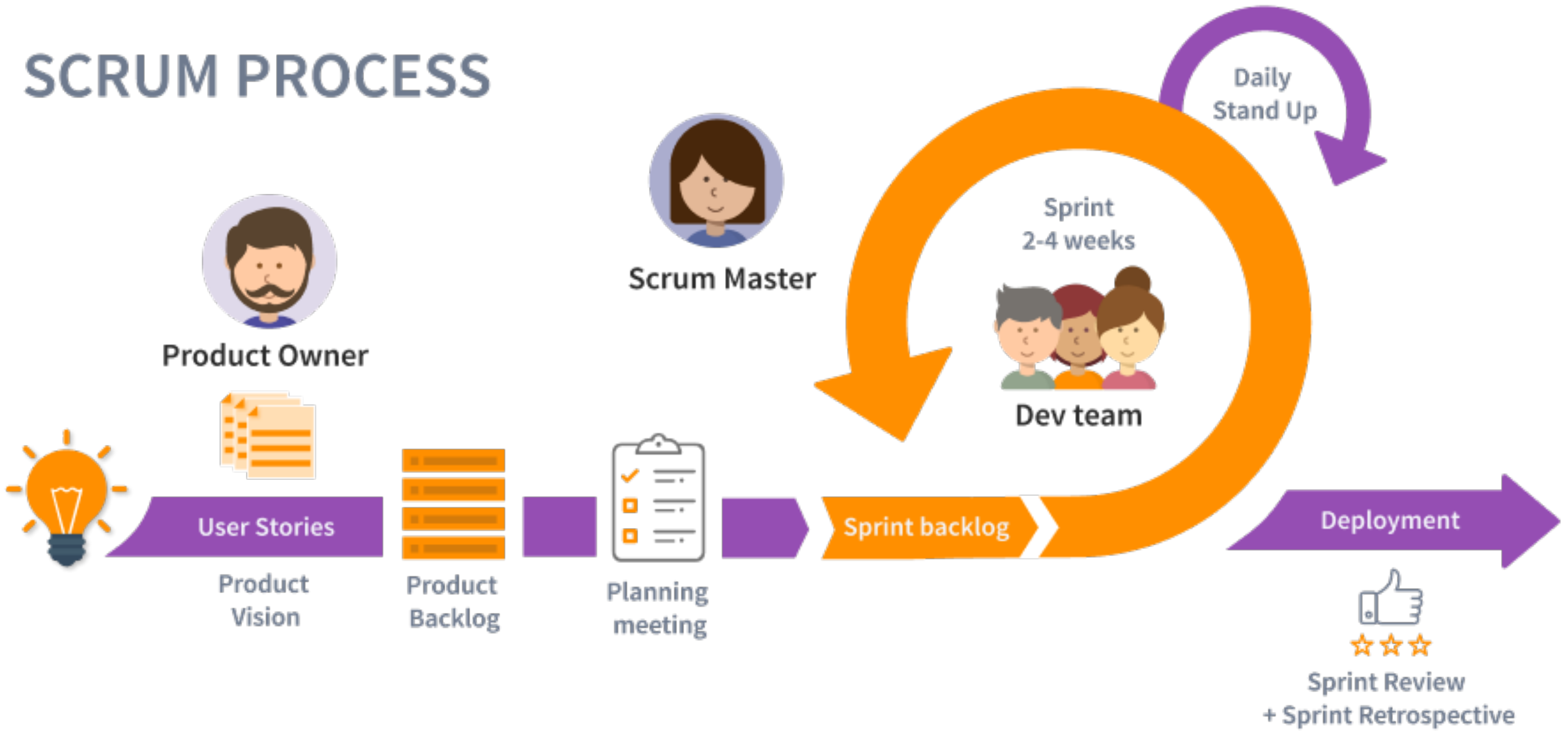
✅ **Flexibility** – Changes can be made based on customer feedback.

✅ **Higher Collaboration** – Regular communication improves team efficiency.

✅ **Early Bug Detection** – Continuous testing ensures fewer defects.

Conclusion

Scrum is ideal for **dynamic projects** where requirements evolve over time, such as **mobile apps, SaaS products, and startup projects**.



Kanban in Agile

Kanban is an **Agile framework** that focuses on **visualizing workflow, limiting work in progress (WIP), and optimizing efficiency**. Unlike Scrum, Kanban does not use fixed iterations (Sprints); instead, it ensures **continuous delivery** by managing work as it flows through different stages.

1. Key Principles of Kanban

✔ **Visualize the Workflow** – Represent tasks on a **Kanban Board** with columns like "To Do," "In Progress," and "Done."
✔ **Limit Work in Progress (WIP)** – Restrict the number of tasks in each phase to avoid overloading the team.
✔ **Manage Flow** – Focus on reducing bottlenecks to maintain a steady workflow.
✔ **Make Process Policies Explicit** – Clearly define rules for moving tasks between stages.
✔ **Implement Feedback Loops** – Regularly review performance and improve the process.
✔ **Improve Continuously** – Adapt and refine the workflow based on experience.

A **Kanban Board** is a **visual tool** used to track the progress of tasks. It typically includes the following columns:

🔷 **To Do** – List of tasks yet to be started.

🔷 **In Progress** – Tasks currently being worked on.

🔷 **Testing** – Tasks being verified before completion.

🔷 **Done** – Completed tasks ready for delivery.

Kanban Workflow Example

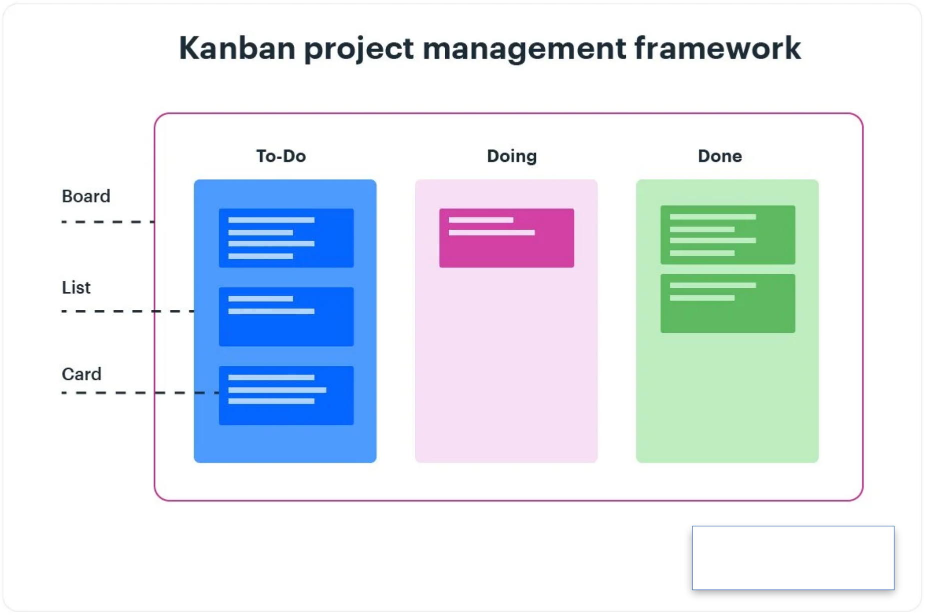
Imagine a **Software Development Team** working on a website update:

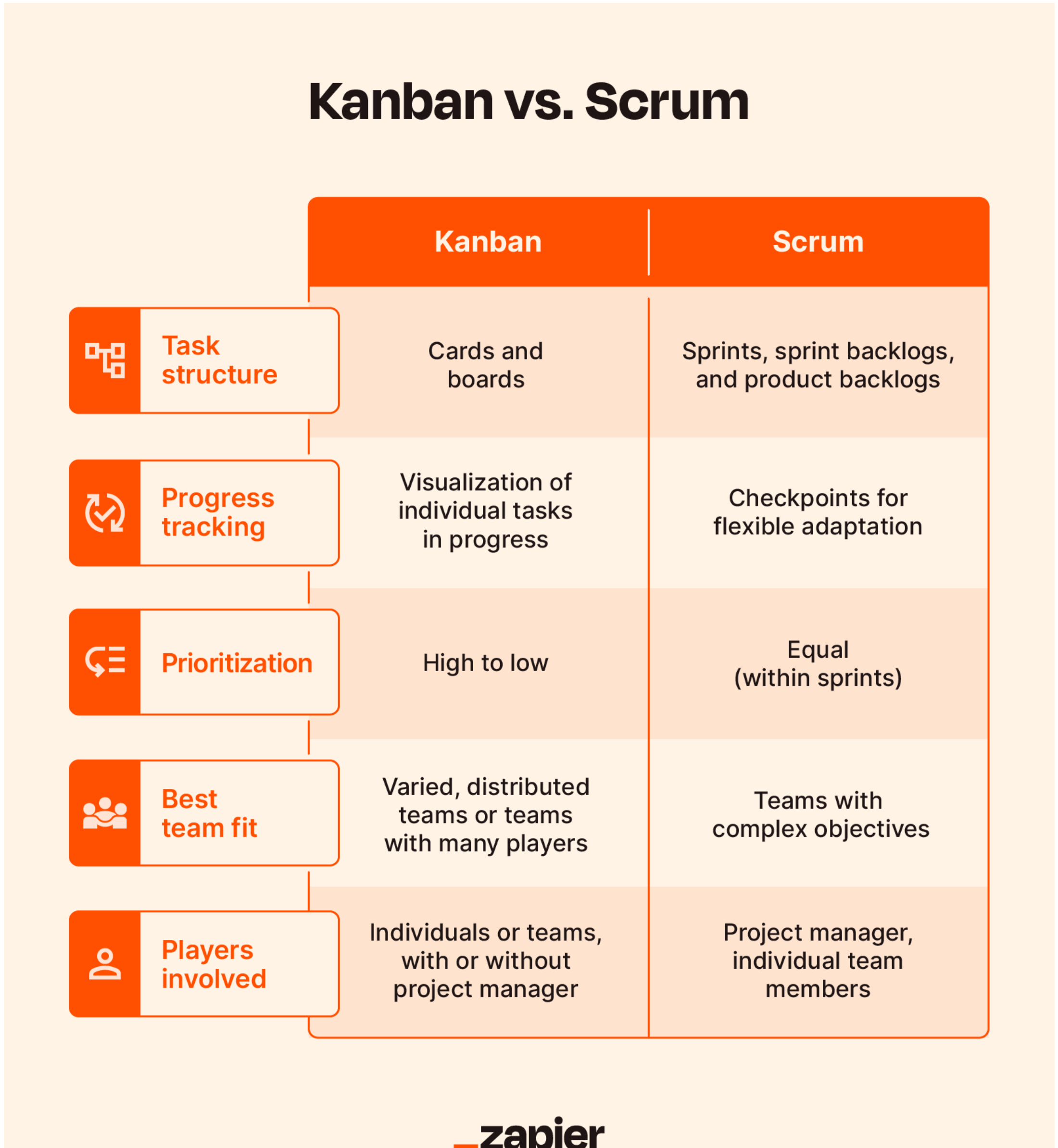
1️⃣ The Product Owner adds tasks to the **To Do** column (e.g., "Fix login issue," "Add new banner").
2️⃣ Developers pick up a task and move it to **In Progress** when they start working on it.
3️⃣ Once a task is completed, it moves to **Testing** for quality checks.
4️⃣ After testing, the task moves to **Done**, and the update is deployed.

4. Example Scenario: Customer Support Team

📌 A **customer support team** handling tickets can use Kanban:

- New **customer issues** go into the **To Do** column.
- Agents take tickets and move them to **In Progress** when working on them.
- If an issue requires validation, it moves to **Review**.
- Once resolved, it moves to **Done**.





Kanban vs. Scrum

	Kanban	Scrum
Task structure	Cards and boards	Sprints, sprint backlogs, and product backlogs
Progress tracking	Visualization of individual tasks in progress	Checkpoints for flexible adaptation
Prioritization	High to low	Equal (within sprints)
Best team fit	Varied, distributed teams or teams with many players	Teams with complex objectives
Players involved	Individuals or teams, with or without project manager	Project manager, individual team members

1. What is Trello?

Trello is a **visual project management tool** that helps teams **organize tasks, track progress, and collaborate efficiently**. It is based on the **Kanban methodology**, where tasks move through different stages using a **Trello Board**.

Trello allows users to create **Boards, Lists, and Cards** to manage projects in a simple and interactive way. It is widely used in **software development, content creation, event planning, and team collaboration**.

2. Key Components of Trello

🔷 **Board** – Represents a project or a workspace (e.g., "Website Development").

🔷 **Lists** – Represent stages of work (e.g., "To Do," "In Progress," "Done").

🔷 **Cards** – Represent tasks (e.g., "Create homepage design").

🔷 **Labels** – Used for categorization (e.g., "High Priority").

🔷 **Checklists** – Used to break down a task into smaller steps.

🔷 **Attachments** – Used to add files, images, or documents to tasks.

🔷 **Comments** – Used for team discussions and updates.

Real-Life Scenario: Managing a Marketing Campaign

Imagine a **marketing team** launching a new product. They can use Trello to track tasks efficiently:

🔶 **Trello Board**: "Product Launch Campaign"

🔶 **Lists**:

✅ To Do → Tasks that need to be done.

✅ In Progress → Tasks currently being worked on.

✅ Review → Tasks that need approval.

✅ Done → Completed tasks.

🔶 **Cards (Tasks)**:

📌 "Design promotional posters"

📌 "Create a social media strategy"

📌 "Launch ad campaign"

📢 The team members update the progress by moving **cards from 'To Do' to 'Done'**, ensuring a smooth workflow.

Scenario-Based Example: Software Development Team

A **tech startup** developing a mobile app can use Trello to track development tasks:

🔷 **Trello Board**: "Mobile App Development"

🔷 **Lists**:

- **Backlog** → Features to be developed (e.g., "User Login").
- **In Progress** → Features being coded.
- **Testing** → Features being tested.
- **Completed** → Features that are ready.

🔷 **Cards**:

📌 "Develop login functionality"

📌 "Add payment gateway"

📌 "Fix UI bugs"

Developers pick tasks, update status, and ensure **efficient workflow management**.

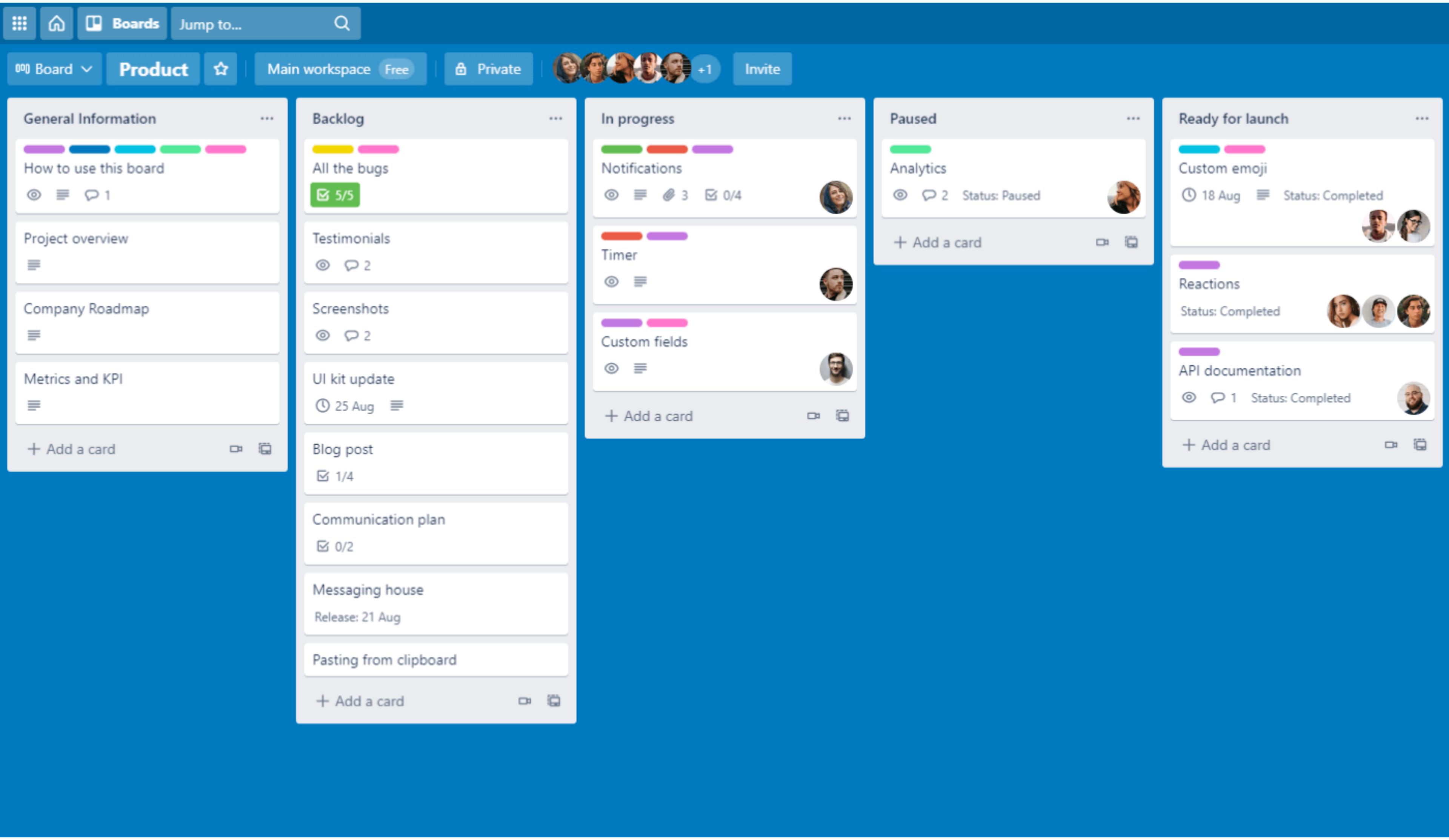
Why Use Trello?

✅ **Simple & Visual** – Drag-and-drop interface makes task management easy.

✅ **Collaboration-Friendly** – Teams can assign tasks, add comments, and track progress.

✅ **Flexible & Customizable** – Works for various projects like event planning, education, or software development.

✅ **Integrations** – Connects with tools like Google Drive, Slack, and Dropbox.


Boards Jump to...

Board Product Main workspace Free Private +1 Invite

General Information
- How to use this board — 1
- Project overview
- Company Roadmap
- Metrics and KPI
- Add a card

Backlog
- All the bugs — 5/5
- Testimonials — 2
- Screenshots — 2
- UI kit update — 25 Aug
- Blog post — 1/4
- Communication plan — 0/2
- Messaging house — Release: 21 Aug
- Pasting from clipboard
- Add a card

In progress
- Notifications — 3 — 0/4
- Timer
- Custom fields
- Add a card

Paused
- Analytics — 2 — Status: Paused
- Add a card

Ready for launch
- Custom emoji — 18 Aug — Status: Completed
- Reactions — Status: Completed
- API documentation — 1 — Status: Completed
- Add a card

Conclusion

Conclusion: Which One is Better?

- **If your project requires flexibility, quick changes, and customer feedback → Agile is better.**
- **If your project has fixed requirements, strict regulations, and heavy documentation → Traditional SDLC is better.**
- **For most modern software projects, Agile is preferred** as it allows faster time-to-market, better adaptability, and continuous improvements.

🔷 **Final Verdict: Agile is generally better for modern projects**, but SDLC models like Waterfall still have their place in industries like **healthcare, defense, and banking**, where strict regulations apply.