What is SDLC?

- SDLC refers to Software Development Life Cycle.
- SDLC is a process of software development planned to design, develop and test high-quality products that meet user expectations.
- SDLC is teamwork aimed at a common goal. At each stage, another teammate takes the role.
- Coordination is achieved through meetings held together and project follow-up programs.
- Designers bring their quality for an attractive application.
- Developers bring high-quality technical skills to build an application.
- Testers understand the business requirement and cover all missing parts to make sure they are in the requirements.

Benefits of SDLC

- It helps to follow and control a project.
- It gives a chance for employers and investors to see the whole planning and process.
- Planning and meetings increase the speed of project creation and development.
- It strengthens the communication of the whole team.
- It reduces the risks of the project.
- It simplifies management and documentation.
- It helps employees to understand what to do and why need to do it.
- It helps all parties agree on the project in advance and sees a clear plan to achieve that goal
- It enables all parties to see the required costs and resources.

SDLC Phases

1. Planning and Requirement Analysis

- It’s a fundamental and the most important phase of SDLC.
- Experts take into consideration of thoughts of the customer.
- The thoughts of the customer are used to plan the basic project planning.

- Quality assurance requirements and identifying the risks associated with the project are also done during the planning phase.
- To successfully implement the project with minimum risks, technical approaches are planned in this phase.

2. Defining Requirements

- The next step after requirement analysis is to clearly define and document the product requirements.
- **Business Requirements Document (BRD**) describes a company's *high-level* goals they're trying to achieve or needs they're trying to fulfil by creating a service or a product. A BRD is always prepared by the business analyst.
- **The Functional Requirements Document (FRD)** is a formal statement of an application's functional requirements. It serves the same purpose as a contract. Here, the developers agree to provide the capabilities specified. The client agrees to find the product satisfactory if it provides the capabilities specified in the FRD.
- **Software Requirement Document/Specifications (SRD/SRS)** A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfil all stakeholders (business, users) needs.

3. Designing The Product Architecture

- The **design phase** can be referred to as the transformation phase because this is when an idea is transformed into a real working system.
- Business Requirements Document (BRD) is the reference for designers to come up with the best design for the product they will develop.
- Based on the requirements set out in the BRD, more than one design approach is often outlined for the product architecture.
- DDS (Design Document Specification): a detailed document that provides a list of points about a product or process. DDS is also called System Design Specification (SDS). DDS is reviewed by all the important stakeholders and based on various parameters such as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

4. **Building or Developing the Product**

- The development stage is the part where developers actually write code and build the application according to the earlier design documents and outlined specifications.
- Developers will follow any coding guidelines as defined by the organization.
- Developers will choose the right programming code to use based on the project specifications and requirements.

5. **Testing the Product**

- This is the stage where product defects (bugs) are reported, monitored, corrected and retested until the product reaches the quality standards defined in the BRD.
- The product must also meet business requirements (requirement specifications)

6. **Deployment in the Market and Maintenance**

- After the product is tested and approved, it is released.
- Once the product is released, it is maintained for the existing customer base.
- With Customer (End User) feedback and Technological Developments needs are redefined and the cycle is restarted

SDLC TEAM MEMBERS

1. **The Product Owner (PO):** The product owner is a business representative or a voice for the stakeholders. The product owner makes sure that development is done in accordance with the project requirements.

2. **The Project Manager (PM):** The project manager can be considered a team representative. The PM's task is to coordinate the work of the participants and organize meetings and negotiations.
   - Responsible for the development of the project plan
   - Establishes a close relationship with the project owners (Stakeholders)
   - Provides communication within the team
   - Manages project risk
   - Prepares the project schedule

- Manages the project budget
- Prevents conflicts that may arise in the project (responsible for crisis management)
- Manages the distribution of tasks.

3. **The Business Analyst (BA):** Business analysts are responsible for evaluating the business processes of companies, predicting requirements, identifying areas for improvement, and developing solutions.
   - BA determines the needs of a project or program and communicates with the managers and partners**.**
   - Business Analyst (BA) must determine exactly what the customer needs.
   - Business Analysts are the bridge between business stakeholders and the development team to ensure everyone is on the same page with the business needs.
   - BA creates BRD, FRD, use cases and spreadsheets.
   - Use-case= A use case is a written description of how users will perform tasks on your website or application.

4. **The Development Team :** development team starts to build the entire system by writing code using the chosen programming language.

- The development team creates the required application and program based on the software requirement document (Software Requirements Specification).
- An SRS gives you a complete picture of your entire project. It provides a single source of truth that every team involved in development will follow. It is your plan of action and keeps all your teams — from development to maintenance — on the same page.
- They write High-Quality code that will meet the expectations and requirements (customer requirements).

5. **Quality Analyst (QA, Tester) :** ensures that the software meets planned functionality, is bug-free and works flawlessly under various conditions.
   - Helping by testing to create a high-quality product before the product is released to the end user.

SOFTWARE DEVELOPMENT METHODOLOGİES

1. WATERFALL MODEL

- Waterfall is the most traditional and ”old school” method.
- A new phase can’t begin until the previous phase has been completed.
- When a phase is finished and a new phase begins, the previous phase cannot be returned.
- Project owner can see the product after the project is completed.

Phases of Waterfall Model

1. **Requirements Analysis :** The aim of this phase is to understand the exact requirements of the customer and to document them properly. Software Requirement Specification (SRS) document is created in this phase.
2. **System Design:** aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. Software Design Document (SDD) is created in this phase.
3. **Implementation:** design is implemented. If the SDD is complete, the implementation or coding phase proceeds properly, because all the information needed by software developers is contained in the SDD.
4. **Testing:** The testing team tests the complete application and identifies any defects in the application.
5. **Deployement:** The product or application is deemed fully functional and is deployed to a live environment.
6. **Maintenance:** improve, update and enhance the final product if needed.

Advantages of Waterfall Model

- It is easy to use and manage.
- The requirements are well understood.
- It is easier to transfer project information.
- Better for small projects
- Roles remain as stable as possible
- Comprehensive documents are created.

- Testing phase is quick because of well defined requirement documents.
- Less amount of defect.

Disadvantages of Waterfall Model

- Change and innovation are difficult.
- Customer feedbacks and suggestions are ignored.
- There is no working product until the project is completed.
- Unable to handle unexpected risks.
- Documentation occupies a lot of time of developers and testers
- Small changes or errors that arise in the completed software may cause a lot of problems
- Preparing the comprehensive documents cost too much time.
- If the encountered defect is found to be caused by the analysis, the analysis, design, and test documents must be updated, and the code must be corrected, resulting in a significant waste of time and money.

---

2. V MODEL

V-Model also referred to as the Verification and Validation Model. In this, each phase of SDLC must complete before the next phase starts. It follows a sequential design process same as the waterfall model. Testing of the device is planned in parallel with a corresponding stage of development.

Verification: It involves a static analysis method (review) done without executing code. It is the process of evaluation of the product development process to find whether specified requirements meet.

Validation: It involves dynamic analysis method (functional, non-functional), testing is done by executing code. Validation is the process to classify the software after the completion of the development process to determine whether the software meets the customer expectations and requirements.

So V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation process is joined by coding phase in V-shape. Thus it is known as V-Model.

The V model has the following advantages:

- It is simple and easy to use.
- Test activities, such as planning and test design, carried out
- before coding activities . Thus, the model saves time.
- Because each stage is tested before moving on to the next stage, the V model has a better chance of success than the Waterfall model.
- Defects can be found at an early stage. Taking defects to the next stage is avoided.

Disadvantages of the V model:

- The method is quite strict, depends on strict rules.
- The software is developed in the development phase as in waterfall, so no early prototypes of the software are produced.
- If there is a change in the requirements at any stage, the test
- documents should be updated along with other documents.
- The V model can be used for projects where requirements are clearly defined.

---

3. The Spiral Model

Spiral Model is a risk-driven software development process model. This model is best used for large projects which involve continuous enhancements. There are specific activities that are done in one iteration (spiral) where the output is a small prototype of the large software. The same activities are then repeated for all the spirals until the entire software is built.

In this model, we create the application module by module and handed over to the customer so that they can start using the application at a very early stage. And we prepare this model only when the module is dependent on each other. In this model, we develop the application in the stages because sometimes the client gives the requirements in between the process.

The different phases of the spiral model are as follows:

Determine objectives and find alternate solutions – This phase includes requirement gathering and analysis. Based on the requirements, objectives are defined and different alternate solutions are proposed.

Risk Analysis and resolving – all the proposed solutions are analyzed and any potential risk is identified, analyzed, and resolved.

Develop and test: This phase includes the actual implementation of the different features. All the implemented features are then verified with thorough testing.

Review and planning of the next phase – In this phase, the software is evaluated by the customer. It also includes risk identification and monitoring like cost overrun or schedule slippage and after that planning of the next phase is started.

Spiral Model Advantages

- Perfect for projects that are large and complex.
- Because of its risk handling ability, the model is best suited for projects which are very critical.
- This model supports the client feedback and implementation of change requests (CRs)
- Since customer gets to see a prototype in each phase, so there are higher chances of customer satisfaction.

Spiral Model Disadvantages

- Because of the prototype development and risk analysis in each phase, it is very expensive and time taking.
- Risk analysis requires high expertise
- It is **not** suitable for a simpler and smaller project because of multiple phases.
- It requires more documentation as compared to other models.
- Project deadlines can be missed since the number of phases is unknown in the beginning.

4. Prototype Model

This model is used when the customers do not know the exact project requirements beforehand. In this model, a prototype of the end product is first developed, tested and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.

Phases of Prototype Model

1. Requirement Identification
2. Design Stage
3. Build the Initial Prototype
4. Review of the Prototype
5. Enhancement of Prototype

5. Agile Methodologies

The Agile methodology is a method of project management that divides a project into phases. It involves ongoing collaboration with stakeholders as well as continuous improvement at each stage. When the work begins, teams go through a cycle of planning, execution, and evaluation. Collaboration is essential, both with team members and project stakeholders.

Agile project management is based on the idea that a project can be continuously improved upon throughout its life cycle, with changes made quickly and responsively.

Due to its flexibility, adaptability to change, and high level of customer input, agile is one of the most popular approaches to project management. Customer input refers to suggestions, enhancement requests, recommendations or other feedback provided by customer.

Agile project management is not a singular framework — rather, it can be used as an umbrella term to include many different frameworks. Agile project management can refer to terms including Scrum, Kanban, Extreme Programming (XP), and Adaptive Project Framework (APF).

1. Scrum Methodology

Scrum is an agile development methodology used in the development of Software based on an iterative and incremental processes. Scrum is adaptable, fast, flexible and effective agile framework that is designed to deliver value to the customer throughout the development of the project. The primary objective of Scrum is to satisfy the customer's need through an environment of transparency in communication, collective responsibility and continuous progress. The development starts from a general idea of what needs to be built, elaborating a list of characteristics ordered by priority (product backlog) that the owner of the product wants to obtain.

What is Product Backlog?

It is a list of requirements for the project. The product owner receives requirements from customers and organizes them in priority order. According to changing needs, the product owner can add or remove items from the product backlog. As a result, the change can be easily integrated into the project at any stage.

Product Backlog Item (PBI): The name given to each requirement in the product backlog.

Backlog Grooming/Refinement(arıtmak, saflaştırmak) : Product Backlog grooming (also known as backlog refinement) is a recurring event or meeting where product managers, product owners, and the rest of the team review and re-prioritize backlog items. The primary goal of product backlog grooming is to keep the backlog up to date and to ensure that backlog items are ready for future sprints. Regular product backlog grooming sessions ensure that the right stories are prioritized and that the product backlog does not turn into a black hole.

What is Sprint Backlog?

The product backlog is divided into sprint backlogs. The Sprint Backlog is a list of tasks the team plans to complete in a Sprint. During the Sprint Planning meeting, team members collaborate to create the list, which is obtained from the Product Backlog.

While determining the user stories to be included in the Sprint Backlog, the order of priority in the product backlog is considered. They do not put the low priority ones in the first sprints.

The amount of user stories to be included in the Sprint Backlog varies according to the points given by the team.

What is Sprint?

Scrum teams work in short timeframes known as sprints. Sprints can be as short as one week or as long as one month. Sprints are held one after the other with no breaks in between to maintain a consistent project timing.

Each sprint begins with a plan (sprint planning) and concludes with a review of the work completed (sprint review) and an additional look back at how the team worked together (sprint retrospective).

During each sprint, the entire scrum team collaborates to complete one or more increments of a larger overall product or project. Each completed increment must be potentially releasable, which means it could theoretically be delivered as-is. In other words, each increment must be thoroughly tested and approved.

Scrum Ceremonies

- **Sprint Planning:** The project manager gathers the team to determine how much work can be completed in one sprint. It is critical that there is enough work to fill the time frame, but not too much. Inadequate work planning can derail a project and result in budget and timeline overruns. Too much planning can result in team burnout and missed deadlines.
- Sprint planning is the first activity in each Sprint.
- The Product Owner and Development team will meet here to decide which Product Backlog Items (PBI) will be included in the Sprint.
- While the Product Owner has the authority to prioritize any PBI for inclusion in the Sprint, the Development team is encouraged to respond, raise issues, and provide feedback as needed.
- After learning about the speed, resources, and factors that can affect the time and resources available, the development team estimates how many PBIs they can add in the Sprint.
- The Sprint Planning Meeting results in a realistic and achievable Sprint Goal and Sprint Backlog that everyone agrees on.
- Sprint Planning answers the following questions;

  1) What will we do?

  2) How will we do?
- If there are any actions that affect the Sprint Planning Meeting among the decisions made at the previous Sprint Retrospective Meeting, they must be carried out.
- A good plan is the foundation of everything. If you start well, you have a good chance of finishing it successfully. A good Scrum Master ensures that the Scrum Team plans well.
- **Daily Scrum:** Daily Scrum meetings, also known as stand-up meetings, ensure that sprints run on time and that all team members are kept informed when problems arise. Sprint stand-ups are typically 15 minutes long and require each team member to discuss what

they've accomplished since the last meeting, what they'll work on before the next meeting, and if any obstacles are in the way.

- Daily stand-up meetings should be brief. More in-depth meetings should be scheduled separately from stand-ups.
- What did we do yesterday? What will we work on today? Have we encountered any difficulties or obstacles? are discussed.
- Answers to questions should be brief and clear.
- If we consider a Development Team of seven people, each member will have 128 seconds to speak. Each question requires 43 seconds to answer.
- **Sprint Review/Demo:** After completing a sprint, the project manager holds a sprint review meeting with all team members and stakeholders to demonstrate sprint outputs, determine what was accomplished and what wasn't, and review project forecasts. Untested or incomplete work is not displayed, but is instead saved for the planning round of the next sprint.
- It is usually held on the last day of the Sprint and gives Stakeholders (customers, management and everyone involved) an opportunity to demonstrate the work done.
- In addition to showing the features of work produced during the sprint, you also get helpful feedback where you can add the Product Backlog that can help guide work for future sprints.
- The owner of the SprintReview Meeting is the Product Owner.
- It is necessary to emphasize to the stakeholders that "the product is their product." It is to inform them that it will be determined in this meeting whether what you will tell them and what you intend to show corresponds to (örtüşmek) what they expect from you.
- Stakeholders can see the working product and tell what new features they want to see added to it, as well as what quality, visuality, and competence the features should have.
- **Sprint Retrospective/Retro:** The final step in the sprint management process is the sprint retrospective. This takes place after the sprint review and before the next sprint planning session. This collaborative session allows team members to discuss accomplishments and challenges during the previous sprint so that processes can be altered, if needed. The goal is to fix one thing at a time and make small, incremental changes from sprint to sprint.

- Retrospective meetings are held to review the previous Sprint in order to improve business processes in the next Sprint and to determine "how can we perform better?" The development team, scrum master, and product owner all attend these meetings.
- It takes place right after the "sprint review" meeting and is the sprint's final meeting.
- The Scrum team discusses what can be improved for future Sprints and how they should be performed.
- They evaluate the difficulties they encountered, as well as the ideas and updates that contributed to their improvement.

Scrum Team

Chicken Roles: They are people who are not actively involved in the operation of Scrum. Like customers, vendors.

Pig Roles: Those who are involved in the Scrum process, that is, the people who do the main work in the project. These;

1) Product Owner (PO)

2) Scrum Master

3) Development Team

Scrum Team Roles

Scrum has three roles: product owner, scrum master and the development team members.

1. Product Owner:

- provides communication between the development team and the customer.
- represents the stakeholders in the team.
- defines the features of the project.
- creates a product backlog according to the priorities of the project.
- The Product Backlog should be accessible and understandable by everyone.
- In each Sprint, PO decides which user stories will be included in the sprint.
- organizes sprint review meetings. He is the owner of sprint review meetings.

  PO is not;
- Not a manager of the Development Team and Scrum

- In technical matters, the Development Team is self-managed.
- Although the PO may have a technical background, this does not give the PO the authority to decide who does what job.
- PO cannot assign jobs or tasks!
- PO cannot interfere(müdahale etmek) with how the job is performed!

2. Scrum Master

- The Scrum master knows the rules, theories and practices of Scrum well and is the person responsible for the team's implementation of these rules. He is not the manager of the team. The Scrum master removes the situations that disturb the Team and prevent them from working efficiently.
- Scrum master is a member of the scrum team, is collaborative, protective, helpful, problem solver, determined, available and knowledgeable.
- Scrum master guides and coaches the team, helping them overcome the obstacles encountered.
- Scrum master strives to increase the harmony and communication between the team members.
- Scrum master facilitates scrum rituals and meetings such as sprint planning, sprint retrospective, daily scrum, and sprint review.
- Scrum master should ensure that the team can work in a safe and trouble-free environment, and should provide many services, including individual coaching, to team members when necessary.
- The Scrum master manages the relationship with the product owner. The Scrum master makes sure that the work set by the product owner is understood by everyone on the team, assists the product owner by finding techniques to organize the backlog effectively.

3. Development Team

It is a self-organizing, cross-functional team of people who are at the core of the Scrum development team structure. It is the team that is responsible for building the actual product increment and meeting the sprint goal.

Self-organizing: *No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality.*

Cross-functional: *they should possess all the skills necessary to complete the sprint goal. They shouldn't need help from people outside the Scrum team to finish their sprint work.*

- Responsibility for technical issues belongs to the development team.
- The product owner and scrum master can determine the priority of the work to be done by the development team, but they do not interfere with what they will do.
- Scrum recognizes no titles for Development Team members, regardless of the work being performed by the person.
- Scrum recognizes no titles for Development Team members, regardless of the work being performed by the person.
- Regardless of the roles and abilities of the people within the team, the entire team is responsible for the completion of a job to the outside.
- The development team is obliged to put a product backlog item that it pulled from the product backlog as a working code snippet in front of the product owner.
- The development team should be "Self Organized". Doesn't need a job description or a job tracker to get the job done.

2. Kanban Methodology

Kanban methodology is an agile method that aims at continuous improvement, flexibility in task management, and enhanced workflow. With this illustrative approach, the progress of the whole project can be easily understood in a glance.

Kanban method revolves around the kanban board. It is a tool that visualizes the entire project to track the flow of their project. Through this graphical approach of Kanban boards, a new member or an external entity can understand what's happening right now, tasks completed and future tasks.

Principles of Kanban

3. **Start With What You Are Doing Now**

Kanban must be applied directly to current workflow. Any changes needed can occur gradually over a period of time at a pace the team is comfortable with.

4. **Agree To Pursue Incremental, Evolutionary Change**

Kanban encourages you to make small incremental changes rather than making radical changes that might lead to resistance within the team and organization.

5. **Respect Current Roles, Responsibilities, And Job-Titles**

Kanban does not impose any organizational changes by itself. So, it is not necessary to make changes to your existing roles and functions which may be performing well.

6. **Encourage Acts Of Leadership At All Levels**

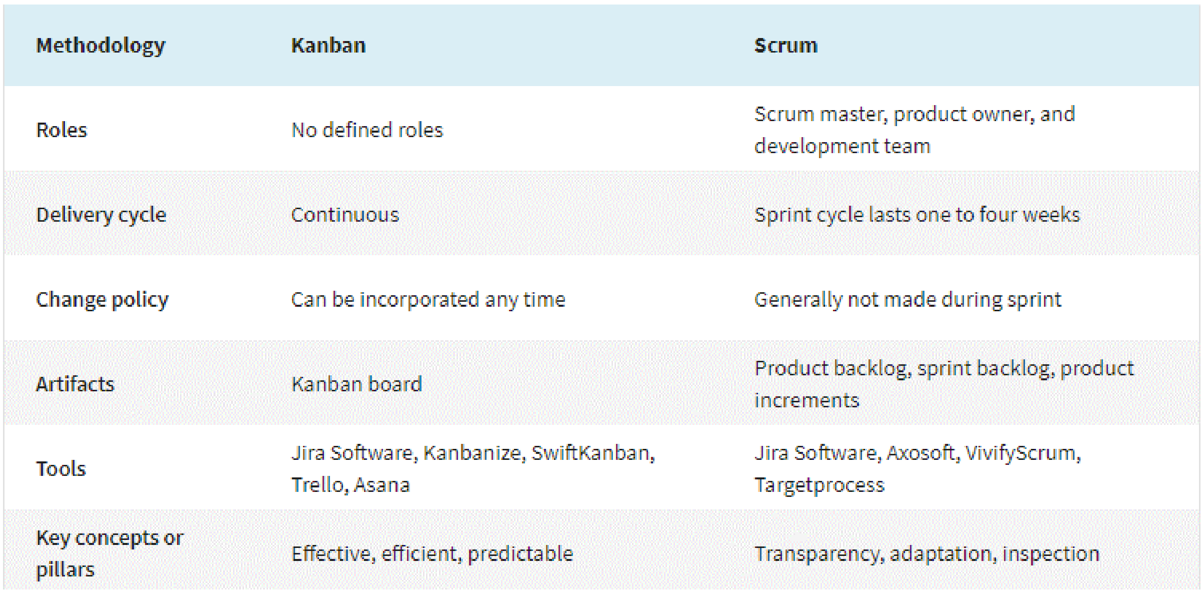
Kanban encourages continuous improvement at all the levels of the organization and it says that leadership acts don't have to originate from senior managers only.

Kanban Practices

1. **Visualize the workflow**: In software development, work is quite abstract, for it does not have a physical form or it can be touched. A Kanban board is used to visualize the work using a physical board and post-it notes. It is an intuitive way to see how works flow from one stage to another.
2. **Limit Work-in-Progress (WIP):** WiP stands for Work in Progress. It is a key term in the Kanban method. Usually, people would think the number of WiP is the bigger, the better. However, any work in progress means that it is not completed. Therefore, it is a waste of time and effort. The final output is much valued, so to limit the work in progress is to make sure the work actually flows from "upstream" to "downstream" in a team and no blockage to affect the team productivity. Balance, a value of Kanban, is closely related to this practice.
3. **Manage flow :**Lead-time is everything for managing the flow of work. It is the time from when the work is started till the end of the work done. Flow is the measurement of work is done in a steady and predictable mode. The staff and work are allocated reasonably.
4. **Make policies explicit:** Transparency is valued in the Kanban method, so making policies simple and clear is a good way to improve team collaboration. As Kanban is a pull system, the team has to set the pull-criteria. Only when the work downstream is done, the upstream work can be started. It is like a pull system, rather than a push system. Then, fix the WiP limit which is immutable afterward. Other process rules can be set accordingly.

5. **Improve feedback loops:** On a Kanban board, feedback is not displayed. Feedback is done via meetings which are cadences of Kanban. A brief introduction of Kanban cadences can be found HERE.
6. **Improve collaboratively, evolve experimentally:** The first part of this practice, improve collaboratively, is how to drive the change in your team, while the second part, evolve experimentally, is how you implement the change. PDCA is often used by the Kanban community. It is a four-step iterative way to improve continuously. PDCA stands for plan-do-check-act. The team needs to do implement change together and cooperate.

Scrum vs Kanban



Methodology	Kanban	Scrum
Roles	No defined roles	Scrum master, product owner, and development team
Delivery cycle	Continuous	Sprint cycle lasts one to four weeks
Change policy	Can be incorporated any time	Generally not made during sprint
Artifacts	Kanban board	Product backlog, sprint backlog, product increments
Tools	Jira Software, Kanbanize, SwiftKanban, Trello, Asana	Jira Software, Axosoft, VivifyScrum, Targetprocess
Key concepts or pillars	Effective, efficient, predictable	Transparency, adaptation, inspection

What is the difference between a theme, an epic, a feature, and a user story?

User Stories are short requirements or requests written from the perspective of an end user. A user story (or just a “story”) is a specific task within an epic. For instance, we may have such user stories as “Sign Up with Email”, “Sign Up with Facebook or Google”, “Log In with Email”, “Log In with Facebook or Google”, “Forgot Password”, and “Log out” in the broad epic that can be called “Registration & Authentication”.

User stories should be short and clearly defined.

User Story consists of 3 parts

Value statement*:* As a (user role) ***(Who)***, I want to (activity) ***(What)***, so that (business value)**(Why)**

Acceptance criteria: Given (context), when (action performed), then (observable consequences)

Definition of done: The Development Team prepares a checklist like a quality control process before presenting the work expected from them to the Product Owner. This list is common for all User Stories and all finished jobs must go through this procedure. This is called “DoD Definition of Done” for short. This list usually includes the following controls;

Unit tests

QA tests

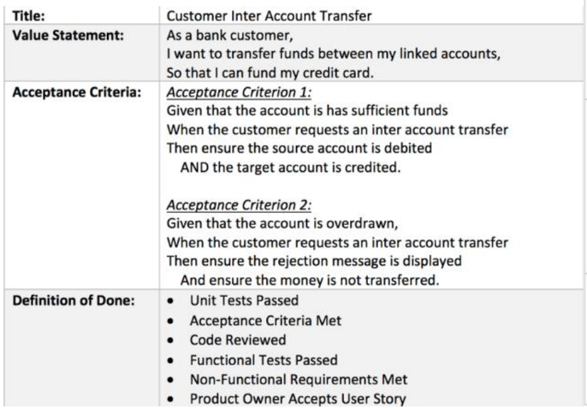
Updating all documents in User Story

Reviewing the code

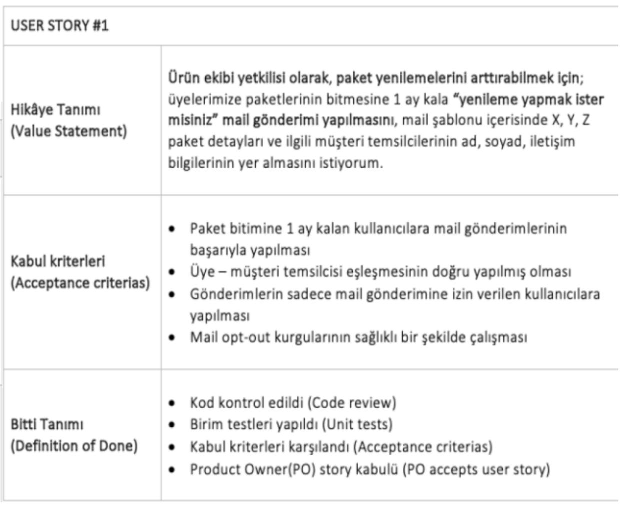
Meeting the User Story acceptance criteria

Product Owner approval

The DoD checklist can also be prepared to be used to check whether certain completion criteria have been met at the end of the Sprint or Release.



Title:	Customer Inter Account Transfer
Value Statement:	As a bank customer, I want to transfer funds between my linked accounts, So that I can fund my credit card.
Acceptance Criteria:	*Acceptance Criterion 1:* Given that the account is has sufficient funds When the customer requests an inter account transfer Then ensure the source account is debited AND the target account is credited. *Acceptance Criterion 2:* Given that the account is overdrawn, When the customer requests an inter account transfer Then ensure the rejection message is displayed And ensure the money is not transferred.
Definition of Done:	• Unit Tests Passed • Acceptance Criteria Met • Code Reviewed • Functional Tests Passed • Non-Functional Requirements Met • Product Owner Accepts User Story



USER STORY #1	
Hikâye Tanımı (Value Statement)	**Ürün ekibi yetkilisi olarak, paket yenilemelerini arttırabilmek için;** üyelerimize paketlerinin bitmesine 1 ay kala **“yenileme yapmak ister misiniz” mail gönderimi yapılmasını**, mail şablonu içerisinde X, Y, Z paket detayları ve ilgili müşteri temsilcilerinin ad, soyad, iletişim bilgilerinin yer almasını istiyorum.
Kabul kriterleri (Acceptance criterias)	• Paket bitimine 1 ay kalan kullanıcılara mail gönderimlerinin başarıyla yapılması • Üye – müşteri temsilcisi eşleşmesinin doğru yapılmış olması • Gönderimlerin sadece mail gönderimine izin verilen kullanıcılara yapılması • Mail opt-out kurgularının sağlıklı bir şekilde çalışması
Bitti Tanımı (Definition of Done)	• Kod kontrol edildi (Code review) • Birim testleri yapıldı (Unit tests) • Kabul kriterleri karşılandı (Acceptance criterias) • Product Owner(PO) story kabulü (PO accepts user story)

User Story Scoring

User stories are scored by the team while they are included in the sprint backlog. At the sprint backlog meeting or refinement meeting, the User Story is read and all members of the development team give points to the user story. The points given are shown by writing on the cards. Points are given according to the numbers in the Fibonacci Series; 1 2 3 5 8 13 21 34

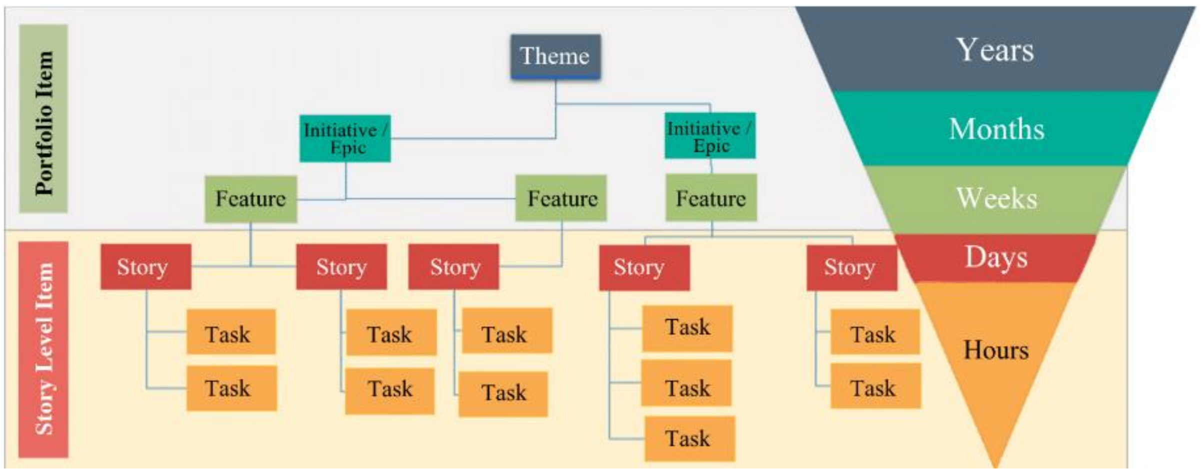
Each member explains why he gave this point and the point of the user story is determined by the joint decision of the team. Team capacity is determined according to the number of developers and QA in the team. 1 point is 1 day's work, that is 8 hours.

Epics are large bodies of work that can be broken down into a number of smaller tasks (called user stories). Epics is a grouping of one or more new functions or features that are planned to be made. Basically, an epic is one big piece of product functionality. Usually, it is too big to be completed in one sprint and should be split up into smaller bodies of work. For instance, an epic may constitute the pieces of code that are responsible for registration & authorization, authentication, user profile, etc. As simple as that.

Feature: These are modules chosen by the PO from Epics and added to the Product Backlog, each containing more than one user story and expected to be completed in more than one week. It is delivered to the end user as a finished product. Features could be defined as characteristics or events of the product or service.

Initiatives are collections of epics. Initiatives in agile management are parts of a theme. In other words, a theme is a set of initiatives united by a common goal. In the above example, under the theme of strength and safety, an initiative could be reducing the bridge vibration by 50% or reducing the building costs.

Themes are large focus areas that span the organization. Themes can be understood as larger focus areas that the whole organization identifies with. In our example, the larger objective of the project is to build a bridge. So the theme could be “strength and safety”. Themes in agile can also be defined as the high-level goals of the team as a whole.



Product backlog: a list of requirements provided by the customer in the form of Epic, Features, Stories, etc. generally captured in an excel sheet format.

Epic: Large level requirements/ needs of the business.

Features: Divisions of Epic, precisely defines the needs of the customer.

Stories: It refers to the detailed definition of each requirement/ Feature.

Tasks: Defines the solution of each Story. They are the actions taken on "Story".

Portfolio Backlog: The Portfolio Backlog is an ordered list of all strategic themes. A portfolio backlog is similar to a product backlog; however, while a product backlog contains items relevant to only one product, a portfolio backlog describes multiple products, programs, or projects for which development has been approved but not yet begun.

What is software testing?

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect/Bug free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements before the software is installed and goes live.

Why software testing is iportant?

Identifies defects early : Software testing is imperative, as it identifies any issues and defects with the written code so they can be fixed before the software product is delivered.

Cost-Effective : Software development consists of many stages and if bugs are caught in the earlier stages it costs much less to fix them.

Security : There have been many situations where user information has been stolen or hackers have gotten to it and used it for their benefit.

Product quality : In order to make your product vision come to life, it has to work as planned. Following product requirements is imperative, to an extent, because it helps you get the wanted end results.

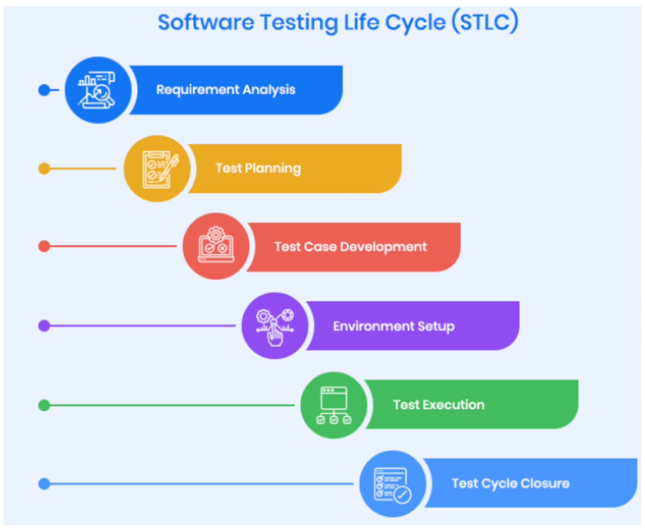
Customer satisfaction/trust : The ultimate goal for a product owner is to give the best customer satisfaction. Reasons why apps and software should be tested is to bring the best user experience possible. Software testing makes sure that the software is user-friendly. That makes it capable of being used by the customers it is intended for.

Software adaptability : Given that there are many devices, operating systems, and browsers available today, it is necessary for software to be compatible with all platforms in order to offer users a smooth user experience.

Helps with scalability(ölçeklenirlik) : A type of nonfunctional software testing process, scalability testing is done to gauge(ölçmek) how well an application scales with increasing workloads, such as user traffic, data volume and transaction counts. It can also identify the point where an application might stop functioning and the reasons behind it, which may include meeting or exceeding a certain threshold, such as the total number of concurrent app users.

What is Software Testing Life Cycle (STLC)?

Software Testing Life Cycle refers to a testing process that has specific steps to be executed in a definite sequence to ensure that the quality goals have been met. In the STLC process, each activity is carried out in a planned and systematic way. There are different phases in STLC which are given below.



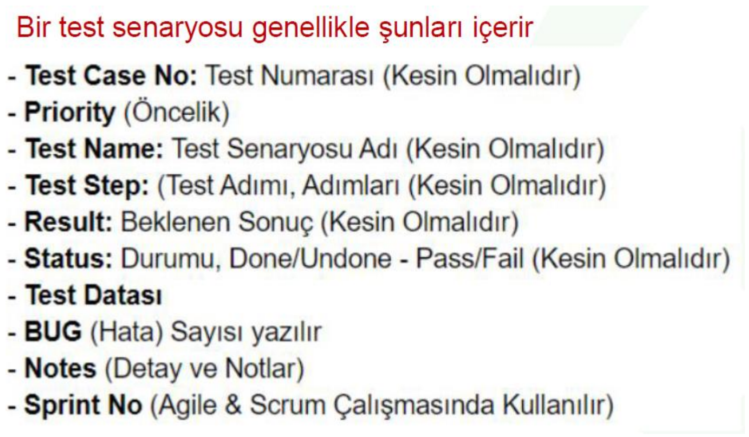
1. **Requirement Analysis:** Requirement Analysis is the first step of Software Testing Life Cycle (STLC). In this phase quality assurance team understands the requirements like what is to be tested. If anything is missing or not understandable then quality assurance team meets with the stakeholders to better understand the detail knowledge of requirement.
2. **Test Planning:** What to test, how the test needs to be done, and who's going to test it… these are the things determined during the test planning phase. Once the requirements have been reviewed, it's time to plan the testing project at a high level. A test plan document is created at this point. This strategy includes tools needed, testing steps, and roles and responsibilities. Part of determining this strategy is a risk and cost analysis and an estimated timeline for testing.

   Test Plan Document (TPD): is a document detailing the objectives, resources, and processes for a specific test for a software or hardware product. The plan typically contains a detailed understanding of the eventual workflow. The Test Plan document is usually prepared by the Test lead or Test Manager, and the focus of the document is to explain what to test, how to test, when to test, and who will do which test.

   Software Requirement Specification (SRS): is a document that *describes what the software will do and how it will be expected to perform.*

   Use Case Documents: a use case is a *list of action or event* steps, typically defining the interactions between a role and a system.
3. **Test Case Development:** The goal of this phase is to determine in detail "how" to test. Test cases should be written to guide the tester through each test. If old test cases are being used, make sure they are up to date. A number of tests might require test data, so

prepare to run tests with the necessary data during this phase so you don’t have to spend time doing this during the tests.

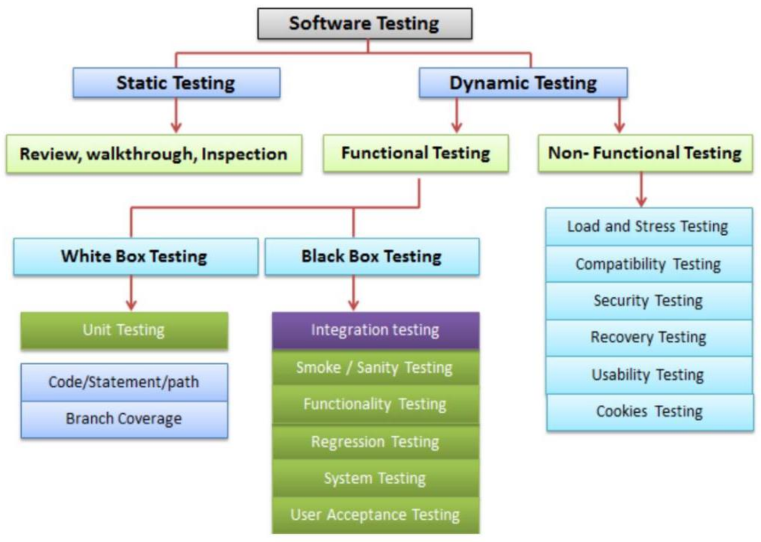


Bir test senaryosu genellikle şunları içerir

- **Test Case No:** Test Numarası (Kesin Olmalıdır)
- **Priority** (Öncelik)
- **Test Name:** Test Senaryosu Adı (Kesin Olmalıdır)
- **Test Step:** (Test Adımı, Adımları (Kesin Olmalıdır)
- **Result:** Beklenen Sonuç (Kesin Olmalıdır)
- **Status:** Durumu, Done/Undone - Pass/Fail (Kesin Olmalıdır)
- **Test Datası**
- **BUG** (Hata) Sayısı yazılır
- **Notes** (Detay ve Notlar)
- **Sprint No** (Agile & Scrum Çalışmasında Kullanılır)

4. **Environment Setup:** The Test Environment Setup phase helps decide the software and hardware conditions under which a product is tested. It’s one of the critical aspects of the testing process and can be done in parallel with the Test Case Development Phase. The test team may not be involved in this activity if the development team provides the test environment, but the test team is required to do a readiness check (smoke testing) of the given environment.
5. **Test Execution:** The Test Execution phase is carried out by the testers in which testing of the software build is done based on test plans and test cases prepared. The process consists of test script execution, test script maintenance and bug reporting. If bugs are reported, then it is reverted back to the development team for correction and retesting will be performed.
6. **Test Cycle Closure:** The Test Cycle Closure phase is the completion of test execution which involves several activities like test completion reporting, collection of test completion matrices, and test results. Testing team members meet, discuss, and analyze testing artifacts to identify strategies that have to be implemented in the future, taking lessons from current test cycles.

Types of Software Testing



A. **Static Testing :** Static testing is the manual review of code or other project documents without executing the code. Static tests should be done before moving on to dynamic tests. Defects identified through review early in the project are less costly to fix than to find them later in the project. Reviews, walkthroughs and inspections are the three types of Static Testing.

B. **Dynamic Testing:** Dynamic testing technique is the type of testing that validates the functionality of an application when the code is executed / by executing the code. In simple terms dynamic testing is performed by actually using the application and seeing if a functionality works the way it is expected to. Dynamic Testing comprises of two major groups of testing (functional testing and non-functional testing).

1. **Functional Testing:** Functional Testing is a type of dynamic test that verifies whether each function of the application we are testing is working in accordance with the given requirements. Functional testing is not related to the source code of the application. The focus when performing this test is always the user-friendliness of the main functions of the application.

2. **Non-Functional Testing:** It is a type of test in which non-functional features of the application are tested. The goal is to determine whether the system is ready or not. The quality characteristics of the components or the system are tested. It is just as important as functional testing in ensuring the quality and proper operation of the software. For

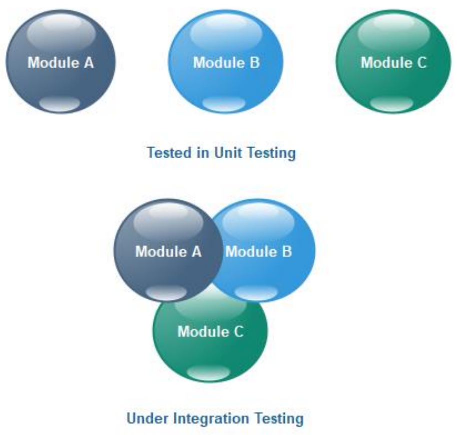
example, how many users can access the system at the same time? Is the system secure enough? The system is being tested to find answers to such questions.

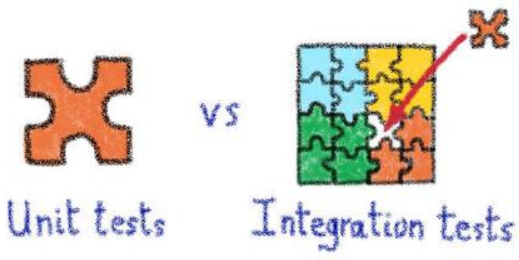
Functional Testing is divided in two;

a) **White-Box Testing:** White-box testing tests a software's internal structures or workings instead of its functionality. The accuracy and quality of the code are tested in white box testing. This test type requires code access. Code structure and design tests are performed. For example, an unnecessary block of code can be detected, as can situations aimed at improving code readability. Errors discovered early in the code also make Black Box testing easier. White box tests are mostly implemented by developers.

   i. **Unit Test:** Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

   ii. **Integration Test:** is a level of software testing where individual units / components are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Integration testing can be either black box or white box testing.





b) **Black-Box Testing:** Black-box testing verifies the system from an end-user perspective without giving any consideration to implementation details. It is not concerned with the structure, design, and implementation of the code. In black box tests, how the system works is tested according to the input and output changes.

i. **Smoke Test:** Smoke Testing is a software testing process that determines whether the deployed software build is stable or not. In simple terms, smoke tests means verifying the important features are working and there are no showstoppers in the build that is under testing. It is a mini and rapid regression test of major functionality. It is a simple test that shows the product is ready for testing. This helps determine if the build is flawed as to make any further testing a waste of time and resources.

ii. **Sanity Test:** is made after a minor bug is fixed or minor changes on the application. Sanity testing focuses on one or two functions of the application during the testing process, while smoke testing is done to ensure that all key functions are working. Thanks to this test, errors are discovered at an earlier stage.

iii. **Regression Test:** This testing is done to ensure that new code changes do not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

Partial Regression is done to verify that the code works well even when changes are made to the code and that unit is integrated with unchanged or already existing code.

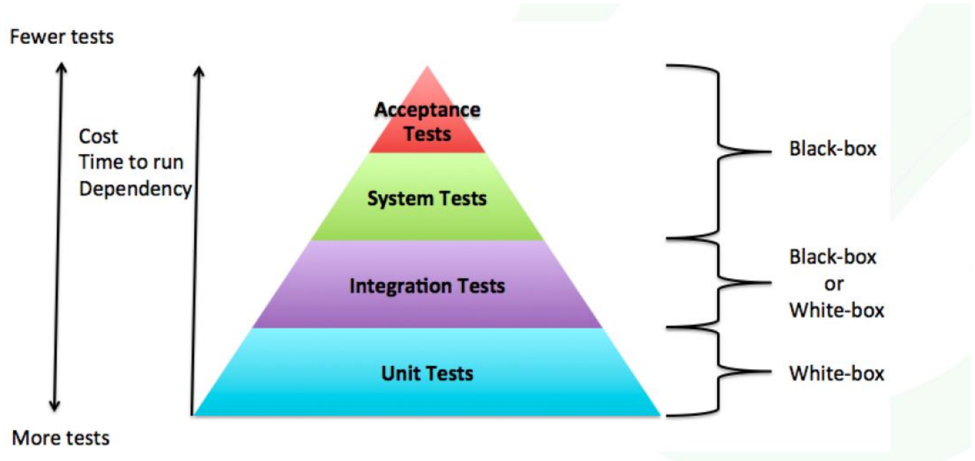
Full Regression is done when a change is made to a set of modules in the code and also when the effect of a change in any other module is uncertain. The product is tested as a whole to check for any changes due to changed code.

iv. **System Test (E2E Test):** System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested. System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. To accomplish this, the QA team might utilize a variety of test types, including performance, usability, load testing and functional tests.

v. **Ad-hoc Testing (Monkey Test):** Adhoc testing is a type of software testing which is performed informally and randomly to find out possible defects or errors at an early possible stage. For this reason, it is also known as Random testing or Monkey testing. Adhoc testing is not performed in an structured way so it is not based on any methodological approach. That's why Adhoc testing is a type of Unstructured Software Testing.

vi. **User Acceptance Test (UAT):** User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done. Also called User Acceptability Testing, **Beta**, Application, or End-User Testing.

Test Pyramid/Hierarchy



Black-Box Test Techniques

1. Boundary Value Analysis

Boundary value analysis tests the software with test cases with extreme values of test data. BVA is used to identify the flaws or errors that arise due to the limits of input data.

For example: Taking inputs for a test case data for an age section should accept a valid data of anything between 1-100. According to BVP analysis, the software will be tested against four test data as -1, 1, 100, and 101 to check the system's response using the boundary values.

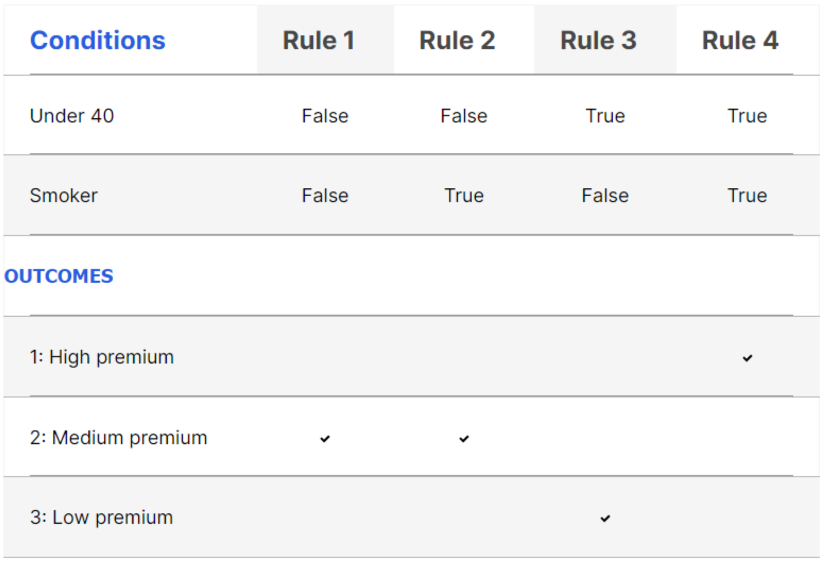
2. Equivalence Partitioning

Testers can divide possible inputs into groups or "partitions", and test only one example input from each group. For example, if a system requires a user's birth date and provides the same response for all users under the age of 18, and a different response for users over 18, it is sufficient for testers to check one birth date in the "under 18" group and one date in the "over 18" group.

3. Decision Table Testing

This approach creates test cases based on various possibilities. It considers multiple test cases in a decision table format where each condition is checked and fulfilled, to pass the test and provide accurate output. It is preferred in case of various input combinations and multiple possibilities.

For example, a health insurance company may provide different premium based on the age of the insured person (under 40 or over 40) and whether they are a smoker or not. This generates a decision table with four rules and up to four outcomes—below is an example with three possible outcomes.



Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Under 40	False	False	True	True
Smoker	False	True	False	True
OUTCOMES				
1: High premium				✓
2: Medium premium	✓	✓		
3: Low premium			✓	

4. State Transition Test

It tests the software application for a sequence of transitions of test inputs. It checks the system behavior changes depending upon what events have occurred/ or what input value is given. Events set off states which become scenarios that the testers test.

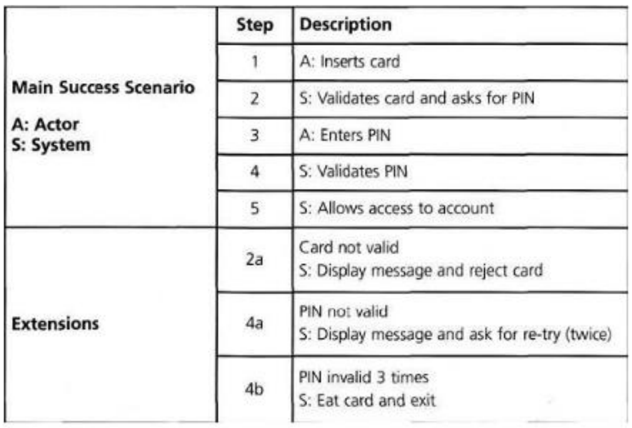
For example, A login page will let you input username and password until three attempts. Each incorrect password will be sent the user to the login page. After the third attempt, the user will be sent to an error page. This state transition method considers the various states of the system and the inputs to pass only the right sequence of the testing.

5. Use Case Testing

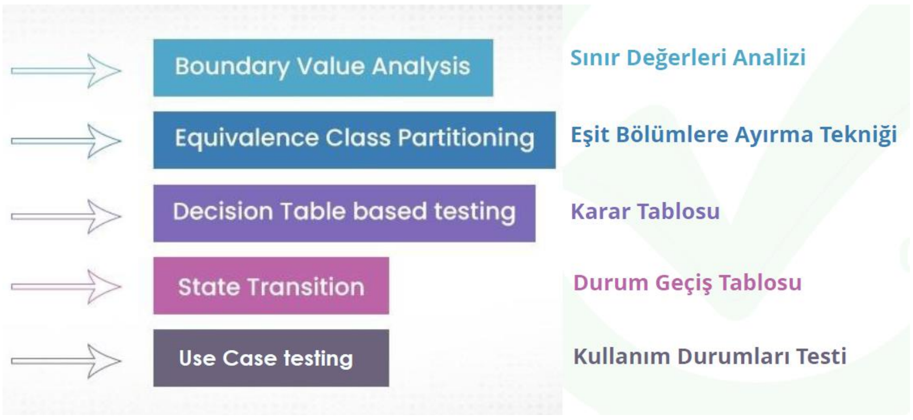
Use Case Testing is a functional black box testing technique that helps testers to identify test scenarios that exercise the whole system on each transaction basis from start to finish.

System requirements can also be specified as a set of use cases. This approach can make it easier to involve the users in the requirements gathering and definition process.

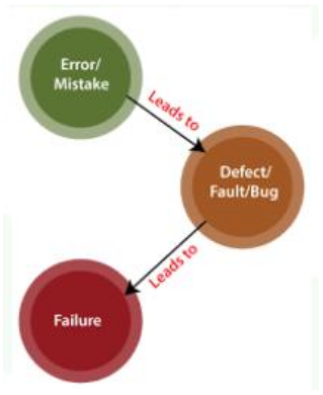
Example:



	Step	**Description**
Main Success Scenario **A: Actor** **S: System**	1	A: Inserts card
	2	S: Validates card and asks for PIN
	3	A: Enters PIN
	4	S: Validates PIN
	5	S: Allows access to account
Extensions	2a	Card not valid S: Display message and reject card
	4a	PIN not valid S: Display message and ask for re-try (twice)
	4b	PIN invalid 3 times S: Eat card and exit



BUG LIFE CYCLE



Error: Problem in Code leads to errors; this means that an error can occur due to developer coding error, either because the developer misunderstood the requirement or the requirement was not defined correctly. Developers use the term error . As a result of the error, the concept of bug and defect emerges.

Defect: an error that we encounter during the development phase of the application. Defect can be found by developer or tester. The important thing is at what stage the defet was found.

Bugs: These are errors found after the application is completed. These errors can be found by testers or by the end user. Such errors are called bugs.

Failure: is defined as the result of an error. A failure is the inability of a software system or component to perform its required functions within specified performance requirements.

What should we do when we find a bug as a tester?

- When we find a bug, we first need to **make sure it's a bug**.
- For this, we should **repeat our test**, if necessary, it **would be useful to get another tester's opinion** if there is. **Information can be given to the Test Lead**.
- If you are working in the same environment with developers, the **developer can be informed about the situation**. The developer may state that it is not a bug, it is a bug, or it is a trivial matter.
- We can **act according to the developer's return**.

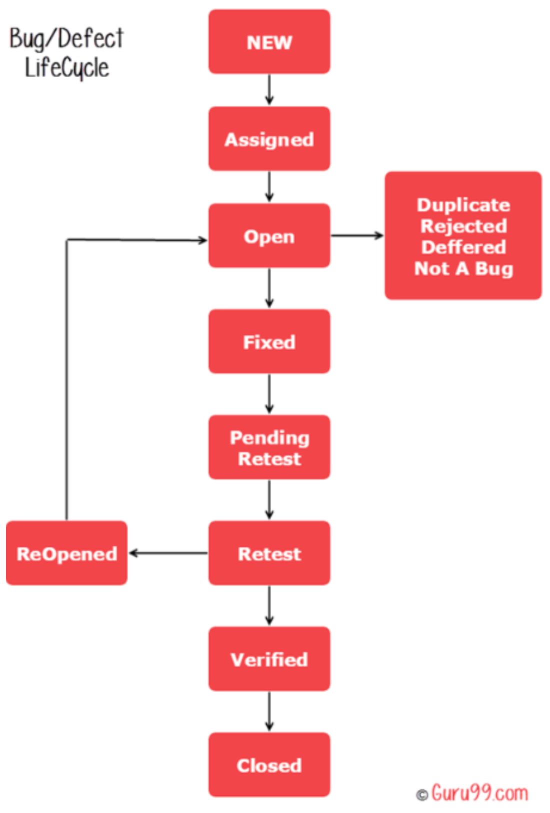
Bug/Defect Life Cycle Steps

New: When a new defect is logged and posted for the first time. It is assigned a status as NEW.

Assigned: Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team

Open: The developer starts analyzing and works on the defect fix

Fixed: When a developer makes a necessary code change and verifies the change, he or she can make bug status as "Fixed."



Pending retest: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is “pending retest.”

Retest: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”

Verified: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “verified.”

Reopen: If the bug persists even after the developer has fixed the bug, the tester changes the status to “reopened”. Once again the bug goes through the life cycle.

Closed: If the bug is no longer exists then tester assigns the status “Closed.”

Duplicate: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate.”

Rejected: If the developer feels the defect is not a genuine defect then it changes the defect to “rejected.”

Deferred: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs

Not a bug: If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”.

---Bug Report---

While error reporting can be performed on the project management tool, it can be created in a document consisting of the following titles. Thanks to this document, the developer can understand the error more easily.

Title: The error title is written to distinguish it from other errors. The error title should be short and concise. For example, it gives an error when the save button is pressed on the user identification screen."

Assigned To: To which developer or team leader will be assigned should be selected in this field.

Priority: In the Software Test Plan, bugs should be ranked according to their priority. The priority values of the errors found should be entered correctly according to the plan.

Severity: Errors found should be grouped according to their severity. The developer involved should resolve the bugs in order of severity.

Root Cause: The source of the found error must be specified. This information can be used for measurement and analysis as a result of testing activities.

Repro Steps: The error must be written step by step so that the found error can be simulated again by the relevant developer. For this, we can give the following example:

a. Go to the user identification page.

b. All mandatory fields on the screen are filled.

c. The save command is given.

Expected Result: The message "User identification completed successfully" should appear.

Actual Result: NullPointerException error message received.

Attachments: Taking a screenshot or video recording of the found error and uploading it to this area will enable the test team to transfer the error to the development team more easily.

TEST PLAN

Test Planning: What to test, how the test needs to be done, and who's going to test it… these are the things determined during the test planning phase. Once the requirements have been reviewed, it's time to plan the testing project at a high level. A test plan document is created at this point. This strategy includes tools needed, testing steps, and roles and responsibilities. Part of determining this strategy is a risk and cost analysis and an estimated timeline for testing.

***Test Plan Document (TPD)*:** is a document detailing the objectives, resources, and processes for a specific test for a software or hardware product. The plan typically contains a detailed understanding of the eventual workflow. The Test Plan document is usually prepared by the Test lead or Test Manager, and the focus of the document is to explain what to test, how to test, when to test, and who will do which test.

What should be considered while preparing a test plan?

Following the preparation of the test plan, the most important consideration is the detection of errors that may occur during the test and their adaptation to the plan. Otherwise, managing the errors and solutions that may occur in accordance with the plan becomes extremely difficult. Test plans are primarily used to collect ideas, manage updates, and communicate. As a result, a good test strategy should include:

- Which items are covered by the test and which are not,
- What are the test objectives?
- What are the risks associated with the project and the product?
- Limitations are the most important aspect of the project and product.
- What is available for testing,
- Software/hardware required for testing,
- It should include information such as a schedule.
- It should be short and focused.

A Software Test Plan document should consist of the following main topics;

Test Strategy and Components to Test: Modules to be tested, test levels to be used and test techniques are defined in this section. The sample text is as follows.

During the software testing activities in the X project, unit, integration and system tests will be carried out. The following software modules will be subjected to the tests specified in this plan. (Module A, Module B, Module C)

Entry and Exit Criteria: The conditions under which the tests specified in the plan will be started and when they will be completed should be defined in detail in this section. Entry and Exit criteria are determined together with the project manager.

Error Management: Rating criteria are determined according to error severity. It is stated in which project and job management tool (TFS, JIRA, etc.) that the found errors will be recorded. The sample text is as follows. The following grading criteria will be used for errors found within the scope of project X:

1st Degree Errors: These are the errors that directly affect the overall operation of the system.

2nd Degree Errors: These are the errors that do not directly affect the general operation of the system, but prevent some parts from working functionally.

3rd Degree Errors: These are the errors that do not prevent the operation of the system and occur visually.

Errors found within the scope of project X will be registered under JIRA. These error logs

will be marked in order of importance and priority.

Risk management

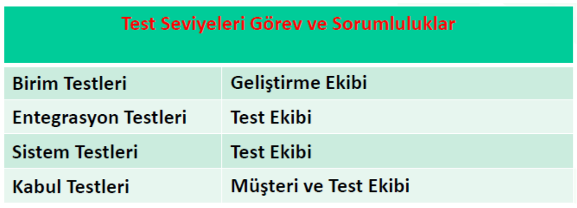
Any risk that may occur during the project should be recorded with the Project Risk Database. In the Test Plan document, any risk that will affect the start or completion of the test activities should be specified in this section. Risks that may hinder the progress of testing activities may include:

- Writing test steps and test cases can be delayed when requirements are delayed.
- Test activities may be disrupted if the test team works in different projects.
- Test activities will not start as a result of not completing the modules to be developed within the scope of the project on time.

Duties and Responsibilities

Proje kapsamında test seviyelerine göre görev dağılımı ve sorumluluklar bu kısımda

tanımlanacaktır.



Test Seviyeleri Görev ve Sorumluluklar	
Birim Testleri	**Geliştirme Ekibi**
Entegrasyon Testleri	**Test Ekibi**
Sistem Testleri	**Test Ekibi**
Kabul Testleri	**Müşteri ve Test Ekibi**

Test Environment and Test Tools

The hardware and software features of the computers / servers to be used for the execution of the test activities will be written in detail in this section. If test automation tools are to be used, information about them (such as version) should be added. The platforms (Windows, Linux, MacOS, etc.) on which the test will be carried out should be specified in detail. If possible, by drawing a simple and understandable diagram describing the test environment, other stakeholders are provided with information about the test environment. The Test Environment and Test Tools section can be written as follows.

Integration and system testing activities within the scope of the X project will be carried out on a server with an IP address of 192.168.1.254. The features of the server are as follows:

CPU: Intel Xeon 3.3GHz

RAM: 8GB DDR4 2133MHz

HDD: 100GB

OS: Windows Server 2012 R2

Programs to Install: Java 8 Update 211 x86

Aim of Using Test Tools Version

JUnit 4.0, JUnit library will be used by the development team for Unit tests.

Selenium WebDriver 3.141.0, SeleniumWebDriver test automation tool will be used by the test team for system tests.

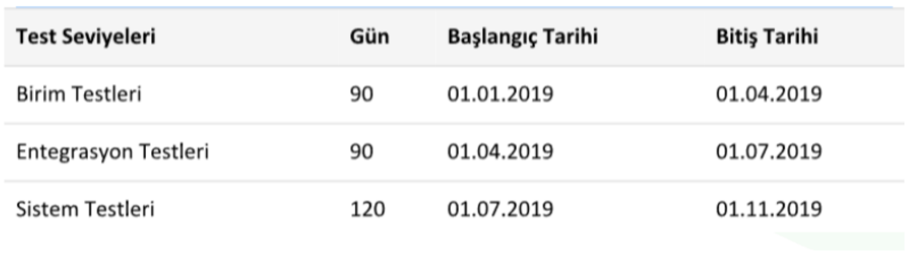
Apache JMeter 4.0, Apache JMeter tool will be used for non-functional tests (Performance, Load and Stress).

JIRA 2013, Errors found as a result of testing activities will be opened on JIRA as an error log to the relevant developer.

Test Schedule

According to the test levels, the start, end dates and estimated time information of the test activities should be specified. The testing schedule can also be included in the Project

Management Plan, in which case you can reference the Project Management plan. If you don't want to show it as a reference, you can show the test schedule you set with your project manager as in the table below.



Test Seviyeleri	Gün	Başlangıç Tarihi	Bitiş Tarihi
Birim Testleri	90	01.01.2019	01.04.2019
Entegrasyon Testleri	90	01.04.2019	01.07.2019
Sistem Testleri	120	01.07.2019	01.11.2019