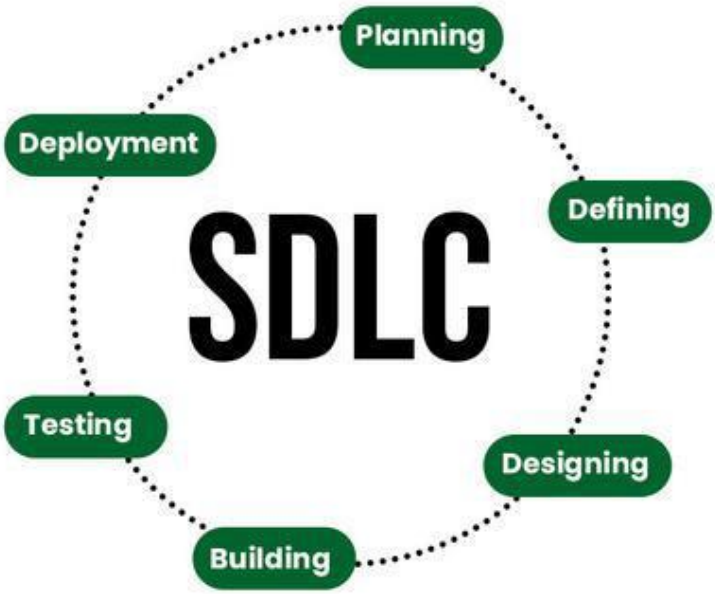
Day-4

Date: - 08-08-2024

Software Development Lifecycle & Modern Development Methodologies

Software Development Lifecycle:

Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software. SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step.



The goal of the SDLC life cycle model is to deliver high-quality, maintainable software that meets the user's requirements.

SDLC in software engineering models outlines the plan for each stage so that each stage of the software development model can perform its task efficiently to deliver the software at a low cost within a given time frame that meets users' requirements.

Assignment-3:

write an SDLC document for to develop an ecommerce application where without login you are not able to do anything after login see home page with products and advertisement. sort products by name, by brand add them into cart do checkout process with cash on delivery. SDLC for above system

Software Development Life Cycle (SDLC) Document:

1. Introduction

This document describes the Software Development Life Cycle (SDLC) for creating an eCommerce application. The application will require user login to access functionality and will include features such as product browsing, sorting, cart management, and a checkout process with cash on delivery.

2. Requirements

2.1. Requirement Analysis

2.1.1. Objectives

- Determine the user needs and system requirements.
- Identify core features and functionalities of the eCommerce application.

2.1.2. Activities

- Conduct stakeholder interviews and workshops.
- Analyze market and competitor solutions.
- Review existing documentation and use cases.

2.1.3. Deliverables

- Requirement Analysis Document.
- Use Case Diagrams.
- User Stories and Acceptance Criteria.

3. Define Requirements

3.1. Functional Requirements

- **User Authentication**: Users must log in to access the application.
- **Home Page**: After login, users will see a home page with a list of products and advertisements.
- **Product Sorting:** Users can sort products by name and by brand.
- **Cart Management**: Users can add products to a cart.
- **Checkout Process**: Users can complete purchases using cash on delivery.

3.2. Non-Functional Requirements

- **Performance:** The application must handle concurrent users efficiently.

- **Security:** User data must be protected and encrypted.
- **Usability:** The user interface must be intuitive and user-friendly.
- **Scalability:** The system should scale to accommodate growing numbers of users and products.

3.3. Deliverables

- Functional Requirements Specification Document.
- Non-Functional Requirements Specification Document.

4. Design

4.1. High-Level Design

- System Architecture: Define the overall system architecture, including server setup, database management, and user interface layout.
- Component Identification: Identify major components such as authentication services, product catalog, shopping cart, and checkout process.

4.2. Detailed Design

- UI/UX Design: Create wireframes and mockups for the home page, product pages, cart, and checkout process.
- Database Design: Design the database schema, including tables for users, products, advertisements, orders, etc.
- API Design: Define the APIs needed for user authentication, product retrieval, sorting, cart management, and order processing.

4.3. Deliverables

- High-Level Design Document.
- UI/UX Wireframes and Mockups.
- Database Schema.

5. Development

5.1. Setup

- **Development Environment:** Configure the development environment with necessary tools and frameworks.
- **Version Control:** Set up a version control system (e.g., Git).

5.2. Implementation

- **Authentication:** Develop and integrate the user authentication module.
- **Home Page**: Implement the home page with product listings and advertisements.
- **Sorting Functionality:** Add functionality for sorting products by name and brand.
- **Cart Management:** Develop the shopping cart functionality to add, remove, and update products.
- **Checkout Process:** Implement the checkout process with cash on delivery payment option.

5.3. Deliverables

- Source Code.
- Configuration Files.
- Development Documentation.

6. Testing

6.1. Testing Types

- **Unit Testing:** Test individual components to ensure they work correctly.
- **Integration Testing:** Verify that different components work together as expected.
- **System Testing:** Test the complete system to ensure it meets the specified requirements.
- **User Acceptance Testing (UAT):** Conduct testing with actual users to validate that the application meets their needs.

6.2. Testing Activities

- Develop test cases based on requirements.
- Execute tests and document results.
- Track and resolve bugs and issues.

6.3. Deliverables

- Test Cases and Test Plans.
- Test Execution Reports.
- Bug Reports and Resolutions.

7. Deployment

7.1. Deployment Plan

- **Prepare Production Environment:** Set up the production environment, including servers and databases.
- **Deploy Application:** Deploy the application code to the production environment.
- **Post-Deployment Testing:** Conduct smoke testing to ensure the application is functioning correctly in the production environment.

Project Name: eCommerce Application Development
Project Manager: [Pavani]

1. Requirement Analysis

Task	Description	Responsibility	Deliverable	Start Date	End Date	Status
Stakeholder Interviews	Conduct interviews with stakeholders to gather initial requirements.	Business Analyst	Interview Notes, Requirement List	01/02/2024	05/02/2024	In Progress
Market Research	Analyze competitor solutions and market trends.	Business Analyst	Market Analysis Report	05/02/2024	10/02/2024	In Progress
Requirements Documentation	Analyze competitor solutions	Business Analyst	Requirements Specification Document	10/02/2024	20/02/2024	Not Started

2. Define Requirements

Task	Description	Responsibility	Deliverable	Start Date	End Date	Status
Requirement Prioritization	Rank requirements based on importance and feasibility.	Business Analyst	Prioritized Requirements List	21/02/2024	23/02/2024	In Progress
Requirement Validation	Review requirements with stakeholders to ensure clarity	Business Analyst	Validated Requirements Document	24/02/2024	28/02/2024	In Progress

	and completeness.					
Feasibility Study	Assess technical and economic feasibility of requirements.	Technical Lead	Feasibility Report	1/03/2024	5/02/2024	In Progress

3. Design

Task	Description	Responsibility	Deliverable	Start Date	End Date	Status
High-Level Design	Develop the overall system architecture, including components and interactions.	Solution Architect	System Architecture Diagram	5/03/2024	10/03/2024	InProgress
UI/UX Design	Create wireframes and mockups for user interfaces.	UI/UX Designer	UI/UX Wireframes and Mockups	11/03/2024	14/03/2024	Not Started
Database Design	Design the database schema including tables and relationships.	Database Designer	Database Schema Design	15/03/2024	17/03/2024	In Progress
API Design	Define the APIs required for system interactions.	Solution Architect	API Specifications Document	17/03/2024	20/03/2024	Not Started

4. Development

Task	Description	Responsibility	Deliverable	Start Date	End Date	Status
Environment Setup	Configure the development environment and tools.	DevOps Engineer	Development Environment Setup	20/03/2024	23/03/2024	Not Started
Code Development	Write and implement the code based on design specifications	Developers	Source Code	24/03/2024	25/03/2024	Not Started
Unit Testing	Test individual components to ensure	Testers	Unit Test Reports	26/03/2024	27/03/2024	Not Started

	they function correctly.					
Integration	Combine components and ensure they work together seamlessly.	Developers	Integrated System	27/03/2024	30/03/2024	Not Started

5. Testing

Task	Description	Responsibility	Deliverable	Start Date	End Date	Status
System Testing	Test the complete system to ensure it meets requirements and functions as expected.	Testers	System Test Reports	1/04/2024	3/04/2024	Not Started
User Acceptance Testing (UAT)	Conduct testing with end-users to verify the system meets their needs and expectations.	Testers, Users	UAT Feedback and Reports	4/04/2024	5/04/2024	Not Started
Bug Fixing and Issue Resolution	Identify and fix bugs and issues discovered during testing.	Developers, Testers	Bug Fixes and Updates	5/04/2024	7/04/2024	Not Started

6. Deployment

Task	Description	Responsibility	Deliverable	Start Date	End Date	Status
Production Environment Setup	Prepare and configure the production environment.	DevOps Engineer	Production Environment Setup	8/04/2024	10/04/2024	Not Started
Deployment Execution	Deploy the application to the production environment.	DevOps Engineer	Deployed Application	11/04/20241	5/04/2024	Not Started
Post-Deployment Testing	Conduct initial testing in the production environment to ensure correct deployment.	Testers	Post-Deployment Test Reports	16/04/2024	19/04/2024	Not Started

Assignment:

Write a BDD test case to verify user is able to withdraw money from bank account

Feature: Withdraw money from the Bank Account

Scenario: Successfully withdraw money from Bank Account

Given the User has a Bank Account with a balance of Rs.5000

And the user is authenticated

When the User wants to withdraw an amount of Rs.1000 from the bank account

Then the bank account balance should be Rs.4000

And the transaction history should include a withdrawal of RS.1000

Assignment-5:

Produce a comparative infographic of TTD, BDD, and FDD methodologies. Illustrate their unique approaches, benefits, and suitability for different software development contexts. Use visuals to enhance understanding.

Test-Driven Development (TDD):

Test Driven Development (TDD) is a development technique which focuses more on the implementation of a feature of a software application/product.

- **Approach**: Write tests before code; iterative cycle of test → code → refactor.
- **Icon**: Code bracket with a checkmark.

1. Process of TDD:

1. Add test case
2. Run the test cases and watch test fails
3. Update the code
4. Run the test cases again
5. Refactor the code (Optional)
6. Repeat the steps for another test case

2. Unique Approaches

- **Cycle**:

    - **Write a Test**: Define a new test based on requirements.
    - **Run the Test**: Confirm it fails (expected result).
    - **Write Code**: Implement code to make the test pass.
    - **Run Tests**: Ensure all tests pass.
    - **Refactor**: Improve code while keeping tests green.
- **Visual**: Circular flow diagram with labeled steps

3. Benefits

- **Benefits**:
    - **Early Bug Detection**: Bugs are caught early in the development process.
    - **Improved Code Quality**: Encourages writing only necessary code.
    - **Refactoring**: Easier to improve and clean up code.
- **Visual**: Checkmarks and bug icons.

4. Suitability for Different Contexts

- **Suitable For**:
    - **Small to Medium Projects**: Where code quality and early bug detection are critical.
    - **Evolving Codebases**: Projects requiring frequent refactoring.
- **Less Suitable For**:
    - **Complex Systems**: Without clear requirements or user stories.

Behavior Driven Development (BDD):

Behavior Driven Development (BDD) is a development technique that focuses more on a software application's behavior.

Approach: Write tests in a natural language format to describe behavior; collaboration between stakeholders and developers.

Icon: Speech bubble with a checklist.

1. Process of BDD:

1. Write the behavior of the application
2. Write the automated scripts
3. Then Implement the functional code
4. Check if the behavior is successful and if it not successful then fix it
5. Organize the code (Optional)
6. Repeat the steps for another behavior.

2. Unique Approaches

- **Cycle**:
    - **Define Behavior**: Write user stories or scenarios in natural language.
    - **Automate Scenarios**: Create automated tests that match the scenarios.
    - **Develop**: Implement code to satisfy scenarios.
    - **Review**: Ensure scenarios are tested and validated.
- **Visual**: Flowchart with “Given-When-Then” structure for scenarios.

3. Benefits

- **Benefits**:
    - Enhanced Communication: Clear understanding between developers and stakeholders.
    - User-Centric: Focuses on user requirements and behavior.
    - Executable Specifications: Scenarios serve as documentation and tests
- **Visual**: Speech bubble with a gear and people icons.

4. Suitability for Different Contexts

Suitable For:

- **Complex Projects**: Requiring collaboration and clear communication.
- **User-Centric Applications**: Where user behavior and requirements are critical.

Less Suitable For:

- **Highly Technical Systems**: With less focus on user behavior.

Feature-Driven Development (FDD):

FDD stands for Feature-Driven Development. It is an agile iterative and incremental model that focuses on progressing the features of the developing software.

Approach: Model-driven and feature-based; focuses on delivering features based on client requirements.

Icon: Feature flag or feature list.

1. FDD Lifecycle

- Build overall model
- Build feature list
- Plan by feature
- Design by feature
- Build by feature

2. Unique Approaches

Cycle:

- **Develop Overall Model**: Create a model of the system.
- **Build Feature List**: Define and prioritize features.
- **Plan by Feature**: Plan and design each feature.
- **Design by Feature**: Design each feature in detail.
- **Build by Feature**: Implement and test each feature.

Visual: Sequential steps diagram with feature emphasis.

3. Benefits

Benefits:

- **Feature-Oriented**: Delivers working features frequently.

- **Scalable**: Suitable for large projects with many features.
- **Clear Milestones**: Features provide clear goals and progress tracking.

Visual: Checklist with feature flags and milestone icons.

5. Suitability for Different Contexts

Suitable For:

- **Large Projects**: With multiple features and stakeholders.
- **Feature-Rich Applications:** Where incremental delivery of features is important.

Less Suitable For:

- **Projects with Undefined Features:** Where features are not well-defined at the start.