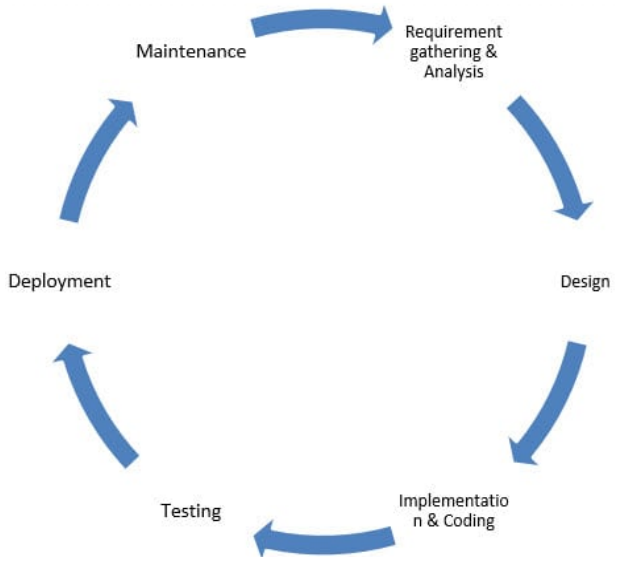
Assignment 1: SDLC Overview – Create a one-page information that outline the SDLC phases (Requirements, Design, Implementation, Testing, Deployment), highlighting the importance of each phase and how they Interconnect.

Sol:

SDLC:

The Software Development Life Cycle (SDLC) is a process used by software development organizations to plan, design, develop, test, deploy, and maintain software applications. It encompasses several phases, each crucial for ensuring the success and quality of the final product.



Phase of SDLC:

1. Requirements Phase:

- **Importance:** This phase involves gathering and analyzing requirements from stakeholders to understand the scope and objectives of the project.
- **Interconnect:** Clear and comprehensive requirements serve as the foundation for all subsequent phases, guiding design, implementation, testing, and deployment decisions.

2. Design Phase:

- **Importance:** In this phase, developers create a blueprint of the software solution based on the gathered requirements.

- **Interconnect:** The design phase translates requirements into technical specifications, laying out the architecture, data structures, algorithms, and user interfaces to be implemented.

3. Implementation Phase:

- **Importance:** Developers write, code, and integrate the design components to build the actual software system.
- **Interconnect:** Implementation transforms the design into executable code, ensuring that the software product aligns with the specified requirements and design.

4. Testing Phase:

- **Importance:** This phase involves testing to identify and rectify defects, errors, and bugs in the software.
- **Interconnect:** Testing validates the functionality, performance, security, and usability of the software, ensuring that it meets the specified requirements and design standards.

5. Deployment Phase:

- **Importance:** In this phase, the software is deployed to the production environment for end-users to use.
- **Interconnect:** Deployment ensures that the software is installed, configured, and operational in the target environment, ready for release to customers or stakeholders.

6. Maintenance Phase:

- **Importance:** Provide ongoing support, updates, and enhancements to the software to address issues and improve performance.
- **Interconnect:** Maintenance sustains the software's usability and reliability over its lifecycle, responding to changing requirements and evolving user needs.

Assignment 2:

Software Development Life Cycle (SDLC): - Food Delivery Platform

1.Introduction:

This document outlines the software development life cycle for 'Food Delivery Platform'. In this document, there are all requirements and phase for developing 'Food Delivery Platform.

2.Project Scope:

The scope of 'Food Delivery Platform' passes through various components and functionalities aimed to provide facility of the delivery of food from restaurants to customers.

SDLC Phase:

Planning:

There are some features to planning this project that are collected by asking and interviewing.

1.User Interface:

- **Customer Interface**: Allows customers to browse restaurant, view menus, place order, track delivery and provide feedback.
- **Restaurant Interface:** Enables restaurant owners to manage their menu, receive order etc.
- **Delivery Driver Interface:** It provides information about orders, navigations, delivery status etc.

2.Order Management:

- Customer should be able to easily browse restaurants, select item from menus, customize order, and place them.
- Customer should be able to check status.
- Restaurants should receive orders promptly and accurately.

3.Menu Management:

- Restaurant should be able to showcase their menus with descriptions, prices and images.

4.Delivery Logistics:

- Maintain a pool of delivery drivers, manages their schedules, and track their performance.
- Monitor the status and location of delivery drivers in real-time to provide accurate delivery.

5.Payment Processing:

- Offer customers various payment method such as credit/debit cards, cash on delivery and online banking.

Analysis:

There are some requirements to analyze it at the time of gathering the information.

1.Functional Requirements Analysis:

User Roles: Define different types of users (customers, restaurants, delivery drivers, administrators) and their respective functionalities.

User Registration and Authentication: Specify how users will register, login, and manage their accounts securely.

Menu Management: Detail how restaurants will upload, update, and manage their menus.

Ordering Process: Describe how customers will browse menus, place orders, make payments, and track delivery status.

Delivery Management: Specify how delivery drivers will receive orders, navigate to the restaurant, pick up orders, and deliver them to customers.

Payment Integration: Define supported payment methods and how transactions will be processed securely.

Ratings and Reviews: Include features for customers to rate restaurants and delivery drivers and leave reviews.

2.Non-Functional Requirements Analysis:

Performance: Specify expected response times, system availability, and scalability requirements.

Security: Detail measures for securing user data, transactions, and preventing unauthorized access.

Usability: Describe the user interface design principles, accessibility standards, and support for multiple languages.

Compatibility: Specify supported devices (web, mobile) and operating systems (iOS, Android).

Reliability: Define backup and recovery procedures, fault tolerance, and recovery plans.

Design:

In the designing phase of a food delivery platform, several key components should be considered to ensure the platform meets user needs. Here are some elements to focus on:

User Interface (UI) Design:

- Design intuitive and user-friendly interfaces for customers, restaurants, and delivery partners.
- Prioritize ease of navigation, clear calls-to-action, and visually appealing layouts.
- Ensure responsiveness across various devices such as smartphones, tablets, and desktops.

User Experience (UX) Design:

- Conduct user research to understand the needs, preferences, and pain points of different user groups.
- Develop user personas and user journey maps to visualize the end-to-end experience.
- Design seamless workflows for ordering food, tracking deliveries, and providing feedback.

Restaurant Management Interface:

- Create a dashboard for restaurant owners to manage menu items, update prices, and track orders.
- Include features for inventory management, order notifications, and performance analytics.

Delivery Partner Interface:

- Develop a dedicated app or portal for delivery partners to receive order assignments, navigate to customer locations, and update order statuses.
- Integrate real-time tracking and communication features to streamline the delivery process.

Payment Gateway Integration:

- Integrate secure payment gateways to facilitate online transactions.
- Support multiple payment methods such as credit/debit cards, digital wallets, and cash on delivery.

Search and Filtering Functionality:

- Implement robust search functionality for users to find restaurants, cuisines, and specific dishes.

Order Management System:

- Design a centralized system to manage incoming orders, assign delivery partners, and track order status in real-time.

- Include features for order prioritization, order history, and order fulfillment analytics.

Security and Data Privacy:

- Implement robust security measures to protect user data, payment information, and sensitive business data.

Implementation:

- Develop the server-side infrastructure, including databases, APIs, and application logic.
- Choose appropriate technologies and frameworks based on scalability, performance, and security requirements.
- Implement user authentication, authorization, and session management functionalities.
- Build responsive and interactive user interfaces based on the UI/UX designs created during the design phase.
- Utilize frontend technologies such as HTML, CSS, JavaScript, and frontend frameworks like React, Angular, or Vue.js.
- Ensure compatibility with various browsers and devices for a consistent user experience.
- Develop the order management system to handle incoming orders, process payments, and assign delivery partners.
- Implement features for order tracking, status updates, and notifications to keep users informed throughout the order lifecycle.
- Document code, APIs, and system architecture to facilitate maintenance and future development efforts.
- Utilize version control systems such as Git to track changes, collaborate with team members, and manage code revisions effectively.

Testing:

1.Functional Testing:

- Conduct functional testing to ensure that all features and functionalities of the platform work according to the specified requirements.
- Test various scenarios, including ordering food, updating menus, processing payments, and tracking deliveries, to verify correct behavior.

2.Usability Testing:

- Evaluate the user interface and user experience to ensure that it is intuitive, user-friendly, and aligns with user expectations.
- Gather feedback from representative users through surveys, interviews, and usability testing sessions to identify areas for improvement.

3.Compatibility Testing:

- Test the platform on different devices, browsers, and operating systems to ensure compatibility and consistent performance across various platforms.
- Verify that the platform functions correctly on mobile devices, tablets, and desktop computers with different screen sizes and resolutions.

4.Performance Testing:

- Conduct performance testing to assess the platform's responsiveness, scalability, and reliability under different load conditions.
- Use tools such as load testing and stress testing to simulate concurrent user activity and measure the platform's performance metrics, including response times and throughput.

5.Security Testing:

- Perform security testing to identify and address potential vulnerabilities and weaknesses in the platform's architecture and code.
- Conduct penetration testing, vulnerability scanning, and security code reviews to identify and mitigate security risks, such as data breaches and unauthorized access.

6.Localization and Internationalization Testing:

- Verify that the platform supports multiple languages, currencies, and cultural conventions to accommodate users from different regions and countries.
- Test localization features, such as language translations and date/time formats, to ensure accuracy and cultural appropriateness.

7.Accessibility Testing:

- Assess the platform's accessibility features to ensure that it is usable by people with disabilities, including those with visual, auditory, motor, or cognitive impairments.

8.User Acceptance Testing (UAT):

- Involve stakeholders and end-users in user acceptance testing to validate that the platform meets their expectations and business requirements.
- Obtain feedback from users on their overall satisfaction with the platform and address any identified issues or concerns before deployment.

Maintenance:

Bug Fixes and Issue Resolution:

- Continuously monitor the platform for any bugs, errors, or issues reported by users or identified through automated monitoring systems.
- Prioritize and address reported issues promptly, releasing bug fixes and patches as needed to maintain platform stability and functionality.

Security Updates Management:

- Stay safe against emerging security by keeping software dependencies and libraries up to date.
- Regularly apply security patches and updates to mitigate known security risks and protect against potential cyber threats, such as data breaches or malware attacks.

User Feedback and Feature Enhancements:

- Gather feedback from users, restaurants, and delivery partners to understand their needs, preferences, and pain points.
- Use user feedback to prioritize feature enhancements and improvements that enhance the user experience, streamline workflows, and add value to the platform.

Regulatory Compliance and Legal Updates:

- Stay informed about changes to relevant regulations and legal requirements, such as data privacy laws, food safety regulations, and payment processing standards.
- Ensure that the platform remains compliant with applicable regulations and standards through regular audits and updates to policies and procedures.

Assignment 3: Agile Principles Application - Write a two-paragraph reflection on how the Agile values of individuals and interactions, working solutions, and customer collaboration apply to the development of the community event app.

Sol:

In developing our community event app, we found that Agile values played a crucial role in guiding our development process. First and foremost, emphasizing individuals and interactions meant that everyone on the team felt valued and heard. We actively encouraged open communication and collaboration, which led to a stronger sense of teamwork and allowed us to tap into each team member's unique skills and perspectives.

This collaborative environment not only fostered creativity but also ensured that everyone was aligned with our project goals.

Additionally, prioritizing working solutions and customer collaboration was instrumental in shaping the app's development. By delivering functional components of the app incrementally, we were able to gather feedback from our target audience early and often. This iterative approach enabled us to make necessary adjustments and improvements based on real user experiences, ultimately resulting in a community event app that truly met the needs and expectations of our users. Overall, embracing Agile principles helped us stay flexible, responsive, and focused on delivering value to our community throughout the development process.

Assignment 4: Scrum Framework Overview - Prepare a one-page cheat sheet on the Scrum framework that includes roles, responsibilities, artifacts, and ceremonies. Provide a brief example of a Sprint task list for the earlier mentioned app project.

Sol:

Scrum Framework:

Scrum is a management framework that teams use to self-organize and work towards a common goal. It describes a set of meetings, tools, and roles for efficient project delivery.

Roles:

- **Product Owner:** Represents stakeholders and ensures the team builds the right product.
- **Scrum Master:** A Scrum master is a facilitator for an Agile development team.
- **Development Team:** Cross-functional, self-organizing group delivering increments of product.

Responsibilities:

- **Product Owner:** Prioritizes the Product Backlog, defines acceptance criteria, and communicates about project vision.

- **Scrum Master:** The Scrum Master is responsible for creating and onboarding project teams, integrating them into the organization and providing a clear vision of the product.
- **Development Team**: Estimates tasks, delivers potentially shippable increments, and self-manages.

Artifacts:

- **Product Backlog:** Prioritized list of features, changes, enhancements, and bug fixes.
- **Sprint Backlog:** Subset of Product Backlog items selected for the Sprint.
- **Increment:** Potentially shippable product functionality completed during a Sprint.

Ceremonies:

- **Sprint Planning:** Collaborative session to plan the work for the Sprint.
- **Daily Standup:** Daily meeting to sync up on progress, discuss any obstacles, and plan the day's work.
- **Sprint Review:** Demo of completed work to stakeholders, gathering feedback, and updating Product Backlog.
- **Sprint Retrospective:** Reflection meeting to inspect and adapt processes, identifying what went well and what can be improved.

Example of Sprint Task List for Community Event App:

Task	Assigned to	Status
Task1. Design wireframe for event details page	UX/UI Designer	In process
Task2. Implement user registration feature	Development Team	To do
Task3. Create database schema for events	Development Team	Done

Task4. Develop event creation form	Development Team	In process
Task5. Write unit test for event creation feature	Development Team	To do

Assignment 5: Agile Project Planning - Create a one-page project plan for a new software feature using Agile planning techniques. Include backlog items with estimated story points and a prioritized list of user stories**.**

Sol:

Project Plan: New Software Feature

Feature Name: Enhanced User Admin Login Feature

Project Overview: Enable administrators to securely log in to the software system to manage user accounts and access administrative functionalities.

Backlog Items:

1.Create Login Page (3 Story Points)

- Design a login page UI.
- Implement basic authentication functionality.

2.Implement Authentication Logic (5 Story Points)

- Develop server-side logic for user authentication.
- Implement password encryption for secure storage.

3.Admin Dashboard Layout (2 Story Points)

- Design the layout for the admin dashboard.
- Include necessary sections for user management.

4.User Authentication API (8 Story Points)

- Develop API endpoints for user authentication.
- Include error handling and validation.

5.Admin User CRUD Operations (13 Story Points)

- Implement functionalities for adding, editing, and deleting admin users.
- Integrate with the database for persistence.

6.Session Management (5 Story Points)

- Implement session management to handle user sessions securely.
- Include mechanisms for session expiration and logout.

Prioritized User Stories:

1. As an admin user, I want to access the login page, so that I can log in to the system securely.
2. As a system, I need to authenticate admin users, so that only authorized users can access administrative functionalities.
3. As an admin user, I want a well-designed admin dashboard, so that I can efficiently manage user accounts and access other administrative features.
4. As a developer, I need to create API endpoints for user authentication, so that the front-end can communicate securely with the server.
5. As an admin user, I need the ability to add, edit, and delete other admin users, so that I can manage administrative access effectively.
6. As a system, I must manage user sessions securely to prevent unauthorized access and ensure data integrity.

Project Timeline:

Sprint 1 (2 weeks)

- Sprint Goal: Create login page and implement authentication logic.
- Items: User Story 1, User Story 2

- Estimated Duration: 2 weeks.

Sprint 2 (2 weeks)

- Sprint Goal: Develop admin dashboard layout and user authentication API.
- Backlog Items: User Story 3, User Story 4
- Estimated Duration: 2 weeks.

Sprint 3 (1 week)

- Sprint Goal: Implement functionalities for adding, editing, and deleting admin users and session management to handle user sessions securely.
- Backlog Item: User Story 5, User Story 6

Estimated Duration: 2 weeks.

Assignment 6: Daily Standup Simulation - Write a script for Daily Standup meeting for a development team working on the software feature from Assignment 5. Address a common challenge and incorporate a solution into the communication flow.

Sol:

Daily Standup Meeting Script

Date: 06/05/24 Time: 9:00 a.m.

Duration: 15 minutes

Attendees:

- [Team Member 1]
- [Team Member 2]
- [Team Member 3]
- [Scrum Master]

Agenda:

1. Brief Updates:

   Each team member shares:

   - What they worked on yesterday.
   - What they plan to work on today.
   - Any blockers or challenges they're facing.

2. Discussion:

   - Address any blockers or challenges raised by team members.
   - Collaborate on solutions and aid overcome obstacles.

3. Action Items:

   - Document any action items or follow-up tasks to resolve blockers.
   - Assign responsibilities and set deadlines for resolving issues.

Meeting Script:

- **Scrum Master:** Good morning, everyone! Let's start our daily standup meeting. Today, we'll focus on our progress towards implementing the enhanced user admin login feature.
- **Team Member 1:** Yesterday, I worked on implementing the login page. Today, I plan to start working on authentication logic. No blockers.
- **Team Member 2:** I worked on creating admin dashboard yesterday. Today, I'll continue with creating API. No blockers.
- **Team Member 3:** I have completed CRUD Operation features yesterday. Today, I'll developing session management features. I don't have any blockers.