Experiment No. 1

Date:

Roll No:

Aim: Application of at least two Traditional Process Models.

Theory:

Software Development Life Cycle

Definition:

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

Purpose of SDLC:

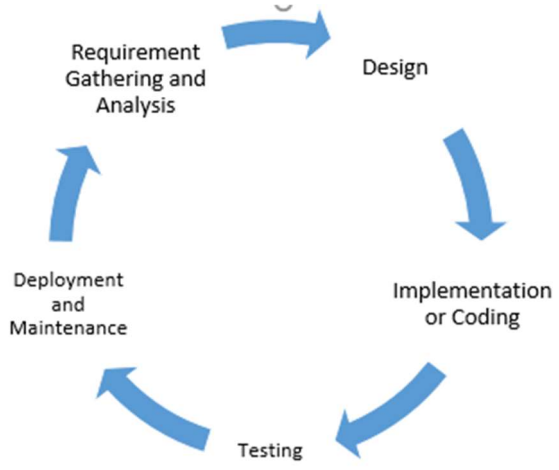
The software development lifecycle (SDLC) is the cost-effective and time-efficient process that development teams use to design and build high-quality software. The purpose of SDLC is to minimize project risks through forward planning so that software meets customer expectations during production and beyond. This methodology outlines a series of steps that divide the software development process into tasks you can assign, complete, and measure.

Purposes of SDLC:

- To increase visibility of the development process for all stakeholders involved
- For efficient estimation, planning, and scheduling
- Inorder to improve risk management and cost estimation
- For systematic software delivery and better customer satisfaction

Stages of SDLC:

1) Requirement Gathering and Analysis
2) Design
3) Implementation or Coding
4) Testing
5) Deployment
6) Maintenance



Stage no.	Stage name	Deliverables
01	Requirement Gathering and Analysis	Business Requirement Documentation (BRD), Software Requirement Specifications (SRS), Technical Requirement Specifications documents are created which serves as the input for next phase- Design.
02	Design	System Architecture, HLD(High level Design), LLD(low level Design), Detailed Design Specifications (DDS) are created which serves as the input for next phase – Coding.
03	Implementation or Coding	Working software is developed which serves as the input for the next phase –Testing.
04	Testing	Test Summary Report, Test results, QA plan, Revised bugs list, User Acceptance test are submitted which serves as the input for the next phase – Deployment.
05	Deployment	Deployed software, Customer's review, Live Production environment, and Data are submitted which serves as the input for the next phase – Maintenance
06	Maintenance	Updated version of the product, Code maintenance, Live system

Software Quality Parameters:

Software quality parameters, also known as software quality characteristics or software quality attributes, are the criteria used to assess and evaluate the quality of a software system. These parameters help determine how well the software meets the desired standards and requirements. Here are some commonly recognized software quality parameters:

1. Functionality: This parameter evaluates whether the software meets its intended purpose and functions correctly. It includes assessing features, usability, accuracy, and compliance with functional requirements.

2. Reliability: Reliability measures the ability of the software to perform its functions consistently and predictably under specified conditions. It includes aspects such as fault tolerance, error handling, availability, and recovery from failures.

3. Performance: Performance refers to how well the software system performs in terms of speed, efficiency, scalability, and resource usage. It includes response time, throughput, processing speed, and the ability to handle varying workloads.

4. Usability: Usability evaluates how user-friendly the software is and how well it accommodates user needs. It includes factors such as ease of learning, ease of use, intuitiveness, consistency, and user satisfaction.

5. Maintainability: Maintainability measures how easily the software can be modified, enhanced, and repaired. It includes factors such as modularity, code readability, simplicity, extensibility, and the presence of documentation.

6. Portability: Portability assesses the ease with which the software can be transferred or adapted to different environments, platforms, or operating systems. It includes factors such as compatibility, adaptability, and compliance with standards.

7. Security: Security measures the software's ability to protect data, prevent unauthorized access, and resist attacks or vulnerabilities. It includes aspects such as authentication, authorization, data encryption, secure communication, and vulnerability management.

8. Testability: Testability evaluates how well the software can be tested to identify defects or issues. It includes factors such as observability, controllability, repeatability, and the presence of appropriate testing techniques and tools.

9. Interoperability: Interoperability assesses the software's ability to interact and work seamlessly with other systems or components. It includes factors such as compatibility with different protocols, data formats, and interfaces.

10. Compliance: Compliance evaluates whether the software adheres to specific standards, regulations, or industry best practices. It includes factors such as legal requirements, industry-specific standards, and internal organizational policies.

These software quality parameters help stakeholders assess the overall quality and fitness for purpose of a software system. They provide a framework for evaluating and improving software quality throughout the development and maintenance lifecycle.

Problem Statement:

The existing systems for weather forecasting are based on models outside of India and just fetch data from respective government agencies and other independent weather stations. They then use their calculations to make predictions for the weather that are not as accurate. The current applications provide information that applies to the regions selected by the user. We aim to personalise the result based on the user's location and recent travel activity.

Requirements:

- Accurate and Reliable Data Sources: The foundation of any weather forecast app is access to accurate and up-to-date data. AI-aided apps should be connected to reliable weather data sources, such as meteorological agencies, weather satellites, weather stations, and other relevant data providers.
- Machine Learning Models: The app should utilize advanced machine learning algorithms and models to analyze historical weather data and improve the accuracy of forecasts. Commonly used models include deep learning, neural networks, and ensemble methods.
- Real-time Data Updates: Weather conditions can change rapidly, so the app should provide real-time updates to ensure users get the latest forecasts and information.
- User-Friendly Interface: The app should have an intuitive and user-friendly interface to make it accessible to a wide range of users. The weather information should be presented clearly and in an easy-to-understand manner.
- Geolocation Services: The app should be able to determine the user's location and provide location-specific weather forecasts. This can be done through GPS or by allowing users to manually input their location.
- Multiple Forecast Parameters: Beyond just temperature and precipitation, the app should offer a variety of weather parameters such as humidity, wind speed, UV index, air quality, and more.
- Weather Alerts: The app should be capable of issuing weather alerts and warnings for severe weather conditions like storms, hurricanes, heavy rain, extreme temperatures, etc.
- Customization Options: Users should be able to customize their preferences, such as units of measurement (Celsius vs. Fahrenheit) and the level of detail they want in their forecasts.
- Offline Functionality: While real-time updates are important, there should also be some degree of offline functionality for users in areas with limited internet connectivity.

- Battery Efficiency: Weather apps can be power-hungry, so optimizing for battery efficiency is crucial to avoid draining a user's device quickly.
- Accessibility Features: Considerations should be made for users with accessibility needs, ensuring the app is usable by individuals with visual or auditory impairments.
- Privacy and Security: As with any app dealing with user data, privacy and security should be a top priority. User data, including location information, should be handled responsibly and with consent.
- Forecast Verification: Implementing mechanisms to assess the accuracy of the app's predictions over time can help improve the reliability of forecasts and build trust with users.
- Support and Updates: Regular maintenance, bug fixes, and updates are essential to keep the app running smoothly and to incorporate improvements based on user feedback.
- GIS geographical information system - it captures, integrates, stores, edits, analyzes, shares, and displays geographic information. It is digitally creating and manipulating spatial areas.
- Forecasting Accuracy.

Scope:

The project aims to develop an AI-driven weather forecast app with a user-friendly interface that utilizes advanced machine learning algorithms. It will integrate real-time and historical weather data from reliable sources through APIs, offering location-specific forecasts for users. The app will provide multiple weather parameters, weather alerts, and customizable options. Geolocation services, offline functionality, and battery optimization will enhance the user experience. Accessibility features, privacy compliance, and continuous maintenance with forecast verification will be prioritized. The end product will be a powerful tool offering accurate and timely weather predictions, empowering users to make informed decisions and stay safe in changing weather conditions.

Aim:

- To deliver reliable and up-to-date weather forecasts and information to users.
- To create a well designed weather app that aims to provide an intuitive and user-friendly interface, making it easy for users to access and interpret weather information.
- To deliver timely weather forecasts and notifications to users, keeping them informed about weather changes that may affect their plans or daily activities.

- To offer personalized experiences by allowing users to customize their preferred weather metrics, units of measurement, themes, or notification preferences.
- To provide access to historical weather data, enabling users to explore past weather patterns, analyze trends, or make comparisons.
- To reduce human error and replace it with AI models with an algorithmic pattern analysis system.
- To generate Topographical and geographical Advance maps regarding sites or areas.
- Accurate Weather Analysis in Short Range area rather than a common analysis for wide area.

Conclusion:

Weather forecasts are made by collecting as much data as possible about the current state of the atmosphere (particularly the temperature, humidity and wind) to determine how the atmosphere evolves in the future.) From this experiment we are able to understand the software engineering approach to the problem statement(Weather forecasting App) with thorough research of software requirements. Also studied about the existing systems, its limitations and planned a proposed system with some extended features.