Shri Vile Parle Kelavani Mandal's

DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING

(Autonomous College Affiliated to the University of Mumbai)

NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Information Technology

COURSE CODE: DJ19ITL602 **DATE:**

COURSE NAME: Software Engineering Laboratory **CLASS: T.Y.BTech**

NAME: Anish Sharma

EXPERIMENT NO.2

CO/LO Analyze real world problem using software engineering principles.

AIM / OBJECTIVE: To select a SDLC model and elaborate the set of activities done under that model.

DESCRIPTION OF EXPERIMENT:

The experiment involves selecting an appropriate Software Development Life Cycle (SDLC) methodology for building **Codeforces Visualizers**. The project involves creating a visual tool that can represent Codeforces data, such as problem-solving patterns, leaderboard dynamics, contest statistics, or other performance-related metrics.

We will examine the SDLC model that is best suited for this project and elaborate on the phases involved, including requirements gathering, design, implementation, testing, deployment, and maintenance.

QUESTIONS:

1. Select a suitable software development methodology / SDLC model for your project under development.

For this project, **Agile** methodology would be a suitable choice due to the following reasons:

- **Frequent changes and iterations:** Codeforces contests happen frequently, and you might need to add new features, adjust visualizations based on feedback, and tweak designs.
- **User feedback integration:** The visualizer will benefit from constant user feedback, which Agile encourages with its iterative process.
- **Collaboration:** Agile allows close collaboration among developers, designers, and stakeholders (or users) to improve the visualizer continuously.

Agile offers flexibility and adaptability, which are key for rapid prototyping, incorporating new features, and addressing bugs based on user feedback.

2. Enlist and elaborate possible set of activities to be performed under each phase of SDLC model / methodology selected.

For the **Agile** methodology, we can break it down into the following phases:

1. **Planning Phase (Sprint 0):**
   - **Define goals and scope:** Understand the main goals of the visualizer (e.g., track user performance, visualize contest data, etc.).
   - **Stakeholder meetings:** Engage with Codeforces users or stakeholders to gather their needs and expectations.
   - **Set up Agile framework:** Create a backlog of features and prioritize them. Define sprints and milestones.
2. **Design Phase (Sprint 1 and onwards):**
   - **UI/UX Design:** Design wireframes and mockups for the visualizer tool. Ensure a user-friendly interface that's easy to navigate.
   - **System architecture design:** Plan out how the data will flow through the system, from Codeforces API integration to data visualization.
3. **Implementation Phase (Multiple Sprints):**
   - **Front-end Development:** Build the user interface (UI) of the visualizer, incorporating the design from earlier.
   - **Back-end Development:** Set up the back-end infrastructure to handle data fetching from Codeforces API, processing, and storing user data.
   - **Codeforces API Integration:** Integrate with Codeforces to pull contest data, rankings, problem-solving histories, etc.
   - **Visualization Algorithms:** Implement algorithms to display data in meaningful visual formats (charts, graphs, leaderboards, etc.).
4. **Testing Phase (During and after every Sprint):**
   - **Unit Testing:** Test individual components like data parsing, visualizations, and API interactions.
   - **Integration Testing:** Test how components work together, ensuring seamless data fetching and visualization.
   - **User Testing:** Present early prototypes to users to gather feedback and make improvements.
   - **Bug Fixing:** Any bugs identified in previous iterations will be fixed during this phase.
5. **Deployment Phase:**
   - **Production Environment:** Deploy the visualizer to a live environment, such as a web server or cloud platform.

- **Beta Testing:** Allow a selected group of users to interact with the live version, report bugs, and provide feedback.
- **Monitoring and Optimization:** After deployment, monitor for performance issues and optimize the application for better user experience.

6. **Maintenance Phase:**
    - **Continuous Monitoring:** Regularly monitor the application for bugs and performance issues.
    - **Feature Updates:** As new contests occur on Codeforces, the visualizer will need updates for new features (e.g., new chart types, additional metrics).
    - **User Feedback and Improvements:** Collect continuous feedback from users to improve functionality.

OBSERVATIONS / DISCUSSION OF RESULTS:

This section will interpret the outcome of each phase in the Agile SDLC model.

Sprint Planning and Implementation:

As we proceed with the design and development phases, we notice that user feedback significantly influences which features are prioritized. For example, after an initial user testing phase, it was evident that users preferred simple leaderboard visualizations over complex graphs. As a result, we adjusted the backlog, shifting the focus to providing quick performance summaries rather than detailed graphs.

Integration and Testing:

When integrating with the Codeforces API, challenges arose with rate limits and latency in fetching data, which was unexpected. As a result, we had to implement caching to reduce API calls, improving performance during the visualizer's use.

Deployment and Maintenance:

The deployment phase highlighted the importance of ensuring that the visualizer could handle a surge in users, particularly during or right after a contest. This meant ensuring scalability of the backend infrastructure.

CONCLUSION:

Based on the results of this experiment, we conclude that Agile methodology is highly suited for building the **Codeforces Visualizers** because of its iterative nature and focus on user feedback. It enabled us to adapt the features and UI based on continuous insights from users, improving the tool with each sprint.

Key takeaways:

- **User feedback is crucial** for shaping the tool's direction and prioritizing features.
- **Flexibility** in design and implementation is necessary to adapt to unexpected challenges, such as API issues.
- **Continuous improvement** is a hallmark of the Agile process, allowing for incremental updates and refinements to the visualizer as Codeforces contests evolve.

This approach ensures that the visualizer is not only functional but continuously evolving based on actual user needs.

QUESTIONS

1. Compare Traditional SDLC models with Agile models

REFERENCES:

Website References:

1. Software Engineering: A Practitioner's Approach: Roger Pressman