**PRACTICAL-2**

**Aim:** Prototype model from the different software engineering models for my PULMS module

**PROTOTYPE MODEL:**

Selecting the Prototype Model for software development, such as for a system like PULMS (Public University Learning Management System), offers several advantages

**Below are the key reasons to choose the Prototype Model:**

**1. Evolving Requirements:**

**Reason**: The Prototype Model is ideal when the system requirements are unclear or evolving. In many projects, especially large-scale systems like PULMS, users may not have a precise idea of what they want initially. Prototyping allows requirements to evolve over time as users interact with early versions of the system and provide feedback.

**Benefit**: You can quickly adapt to changing needs, helping to prevent miscommunication or misunderstanding of user requirements.

**2. Early User Feedback:**

**Reason**: A key advantage of the Prototype Model is its ability to gather early feedback from users. Users get to interact with a working version of the system early in the process, which helps them clarify their needs and expectations.

**Benefit:** The system can be refined based on real-world user feedback, reducing the risk of developing features that are not useful or desired by the users.

**3. Improved Communication with Stakeholders:**

**Reason:** Since prototypes are tangible and interactive, stakeholders (students, instructors, administrators) can provide more specific and actionable feedback. It fosters better communication between developers and users.

**Benefit:** By showing stakeholders a working prototype, you ensure that everyone involved has a clearer understanding of the system's capabilities, and any misunderstandings about the functionality can be addressed early.

**Reasons not to select other models:**

**1. Waterfall Model:**

The Waterfall Model is not flexible and doesn’t allow for iterative changes based on user input, making it less suitable for PULMS, where ongoing user engagement and requirement evolution are essential.

**2. Spiral Model:**

While the Incremental Model works well for certain types of systems, its reliance on pre-determined increments may be less adaptable for PULMS, where user feedback can lead to changes in feature requirements throughout development.

**3. Agile Model:**

While Agile can be beneficial in environments where flexibility and user feedback are paramount, the constant iterations and demands for frequent releases could be challenging for larger systems like PULMS that may need a more structured approach to ensure all features integrate smoothly.

**PUMLS**

Software engineering model

**1. Requirements Gathering:**

**Objective:** In this initial phase, you gather high-level requirements that are essential for building the prototype. The goal is to identify core functionalities and features needed in PULMS but without focusing on detailed specifications at this point.

**Process:**

Engage with key stakeholders (e.g., students, faculty, administrators) to understand the essential features of the system.

Identify major functionalities like course management, student registration, assignment submission, grading system, discussion boards, and user profiles.

Focus on the basic requirements that will form the foundation of the prototype. Detailed features, performance expectations, and non-functional requirements (like security) may be deferred for later stages.

**Example Requirements for PULMS:**

Users can log in and view a dashboard of courses.

Students can submit assignments and view grades.

Instructors can upload course content, create assignments, and grade submissions.

**2. Quick Design:**

**Objective:** After gathering the basic requirements, a quick design of the prototype is created. This design will focus on just the most important functionalities that need to be demonstrated in the prototype.

**Process:**

Sketch out basic user interfaces (UI) and define user flows.

Use simple design tools (e.g., wireframes or mockups) to visualize how the PULMS system will look and work.

Ensure the design represents core functionality (e.g., user login, dashboard view, assignment submission interface) without building a complete, fully-detailed system.

At this point, the design doesn't focus on technical complexities or backend features but aims to demonstrate key workflows.

Example Quick Design for PULMS:

**UI Elements:** Basic wireframes for student and instructor dashboards.

**User Flow:** Steps for a student to log in, view courses, submit assignments, and check grades.

**Functional Flow:** How instructors upload content and grade assignments.

**3. Build Prototype:**

**Objective:** Build a working prototype based on the quick design. This prototype will have basic functionality but won't be fully developed or scalable.

**Process:**

Develop the prototype focusing on the core functionality identified in the requirements phase.

Use rapid development tools to create a basic version of the system that allows users to interact with it (e.g., using simple front-end and back-end tools).

The prototype will include basic operations such as user login, course enrollment, assignment submission, and basic grading, but may not have complex backend systems, data security, or scalability.

The prototype is built quickly to demonstrate the intended features to users.

**Example Prototype Features for PULMS:**

A student login screen that allows students to access their courses.

A simple "Upload Assignment" feature for instructors.

Basic notification functionality when assignments are graded.

**4. Evaluate Prototype:**

**Objective:** After the prototype is developed, evaluation is conducted with real users (students, instructors, administrators) to gather feedback on how the prototype meets their needs and expectations.

**Process:**

Present the prototype to stakeholders (students, faculty, and administrators) to interact with and provide feedback.

Gather insights on the usability, functionality, and design. Ask questions about whether the features are intuitive, whether the system meets their needs, and where improvements can be made.

Evaluate user interactions with the system to identify issues in navigation, design, or workflow.

Document feedback on both the positive aspects of the system and areas that require improvement.

**Example Evaluation for PULMS:**

Students may comment on whether the assignment submission process is clear and easy to follow.

Instructors might provide feedback about how easy it is to upload course materials and grade assignments.

Administrators may highlight concerns related to security and data privacy.

**5. Refine Prototype:**

**Objective:** Based on the feedback from the evaluation phase, the prototype is refined. This may involve adding new features, improving usability, or fixing problems identified in the evaluation stage.

**Process:**

Modify the prototype to incorporate the feedback from users. This can include enhancing UI/UX, adding new features (e.g., notifications for assignment deadlines), or removing unnecessary elements.

Refine the design and functionality to address the issues discovered during evaluation. This may involve iterative cycles of development and feedback until the system reaches an acceptable state.

The system may undergo design iterations (UI redesign, improved navigation) or feature enhancements (e.g., adding automated grading).

**Example Refinements for PULMS:**

Improve the UI for assignment submission based on user feedback (e.g., adding drag-and-drop for file uploads).

Add a notification system to alert students when new content is uploaded or when assignments are graded.

Implement a basic grading system that automatically calculates grades based on student submissions.

**6. Testing:**

**Objective:** The refined prototype undergoes testing to ensure that the features work as expected and that the system meets both functional and non-functional requirements.

**Process:**

Conduct functional testing to ensure that each feature of the PULMS prototype works correctly. This includes user authentication, assignment submission, and grading workflows.

Perform usability testing with real users to evaluate the system's ease of use and identify any user experience (UX) problems.

Test for bugs and ensure that there are no major issues in the prototype’s functionality.

Depending on the phase of the project, security testing (e.g., ensuring that user data is protected) and performance testing (e.g., checking system response times) might also be conducted.

**Example Testing for PULMS:**

Functional Testing: Ensure that students can submit assignments correctly and that instructors can grade them.

Usability Testing: Evaluate if users can easily navigate the system and complete tasks without confusion.

Security Testing: Test whether the student and instructor data are protected, particularly in assignment submission and grading.

**7. Final Refinement and Iteration:**

After testing, more refinements are made to address any issues found during testing, and the prototype continues to evolve until it meets the desired level of functionality and user satisfaction.

The process may continue through several cycles of prototyping, testing, and refinement, ultimately leading to a fully functional and stable PULMS system ready for deployment.