**Activity Title: "Develop Your Own Software Engineering Plan"**

**Objective:**

Students will simulate the process of planning and designing a software project by following the key phases of the Software Development Life Cycle (SDLC). This will enhance their understanding of software engineering principles and methodologies.

**Materials:**

- Pen and paper or digital tools for documenting their process

- Whiteboard (optional for group work)

- SDLC phase definitions (can be shared with the students)

**Group Size:**

4-6 students per group, or individual work for smaller classes.

**Time Required:**

45 minutes (30 minutes for activity, 15 minutes for presentation and discussion)

**Activity Breakdown**

**Part 1: Project Selection (5 minutes)**

Each group or individual will choose one type of software project to work on. For instance:

- A mobile fitness app with tracking features

- A social networking platform for a specific niche

- A task management tool for personal productivity

- A simple 2D game for kids

**Deliverable:**

Write a brief description of the project, including its primary goal and main features.

**Part 2: Plan the Software Development Life Cycle (SDLC) (25 minutes)**

Each team will move through the SDLC phases to create a high-level plan for their software project.

**Phase 1: Requirements Gathering (5 minutes)**

Teams identify the main requirements of their software from the end user's perspective.

- Example: "The fitness app should allow users to log workouts, track progress, and set fitness goals."

**Deliverable:** List 5-7 key functional and non-functional requirements for the software.

**Example requirements:**

1. Allow users to create personal profiles.

2. Provide daily workout recommendations.

3. Sync workout data with wearable devices (e.g., smartwatches).

4. Allow users to track progress with visual charts.

5. Ensure data privacy and protection.

**Phase 2: System Design (5 minutes)**

Teams create a rough system design, outlining how the software will be structured and which components it will need.

- Task 1: Design the software architecture (e.g., frontend, backend, database).

- Task 2: Sketch a rough wireframe for the user interface (UI) showing key screens.

**Deliverable:**

1. Simple block diagram of the system architecture (frontend, backend, database).

2. Rough UI wireframe (main screens).

**Example Design:**

- Frontend: Mobile app with a home screen, profile page, workout page, and tracking dashboard.

- Backend: Cloud server for storing user data.

- Database: Stores user profiles, workout history, and progress data.

**Phase 3: Development Plan (5 minutes)**

Teams will plan how to develop the software, including breaking down tasks, choosing programming languages, and setting up teams.

**Deliverable:**

- List of development tasks and roles (e.g., frontend development, backend development, database setup).

- Decide on the tech stack (e.g., Frontend: React Native, Backend: Node.js, Database: MongoDB).

- Estimate a timeline for development (e.g., 2 months for MVP).

**Example Development Plan:**

1. Frontend Development: Build the mobile app using React Native.

2. Backend Development: Set up API endpoints to handle workout logging and user authentication.

3. Database Setup: Implement MongoDB to store workout data and user profiles.

4. Testing: Plan to conduct user testing and fix bugs by the end of month 2.

**Phase 4: Testing Plan (5 minutes)**

Students develop a plan for testing the software, including methods for catching bugs, ensuring functionality, and gathering user feedback.

**Deliverable: A simple testing strategy for:**

1. Unit Testing (e.g., testing individual app features such as logging a workout).

2. Integration Testing (e.g., ensuring the app communicates correctly with the database).

3. User Acceptance Testing (e.g., having users test the app for usability and functionality).

**Example Testing Plan:**

1. Unit Tests: Test individual features like profile creation and workout logging.

2. Integration Tests: Test the interaction between the app and the cloud backend.

3. User Testing: Create a small beta test group to provide feedback on the user experience.

**Phase 5: Maintenance Plan (5 minutes)**

Teams will think about the long-term maintenance of the software, including how to handle future updates and bug fixes.

**Deliverable:**

- Plan for software updates and bug fixes post-launch.

- Plan for monitoring system performance and user feedback.

**Example Maintenance Plan:**

1. Schedule monthly updates to add new features (e.g., new workout types).

2. Monitor system for bugs and errors using a crash reporting tool.

3. Allow users to submit feedback through the app to prioritize future improvements.

**Part 3: Present and Reflect (15 minutes)**

Each group will present their SDLC plan to the class. They should explain:

1. The chosen software project.

2. How they approached each SDLC phase.

3. Key challenges or decisions they made.

**Discussion Questions:**

- Which phase did you find most challenging, and why?

- How would you improve your planning process?

- What did you learn about software engineering from this exercise?