Software Development Life Cycle (SDLC) Description for "Swiftly Planner" Project

For the "Swiftly Planner" project, we implemented the phases of the Software Development Life Cycle (SDLC) in an intensive and flexible manner, drawing inspiration from the principles of a "Mini Scrum" methodology. This approach was crucial to ensure the delivery of a functional and well-documented prototype.

1. Planning Phase

- **Definition:** This is the initial stage where project requirements are understood, objectives are defined, and roles are assigned.
- **Our Application:**
    - **Rapid Meeting:** We held an immediate team meeting to comprehensively understand the scope of the "Swiftly Planner" project and its core requirements (Core Features and required Artifacts).
    - **Product Backlog Definition:** We created a list of all necessary features and documentation (e.g., creating tasks, assigning them, Class Diagram, Swift Code, etc.), which served as our Product Backlog.
    - **Role Assignment:** Roles were clearly distributed within the team (Team Leader, Code Developers, Designer/Documenter), enabling parallel and efficient work.
    - **Environment Setup:** A GitHub Repository was immediately set up to ensure effective collaboration and code version management.

2. Design Phase

- **Definition:** In this phase, the system's structure is determined, components are designed, and their interactions are defined.
- **Our Application:**
    - **Data Model Design:** We defined the fundamental data structures (Task, User, Sprint Structs) in Swift, which formed the basis of our design.
    - **Class Diagram:** A Class Diagram was designed to illustrate these models and their relationships, providing a clear view of the application's logical structure.

    - **Sequence Diagram:** We designed a Sequence Diagram for a key operation, "creating a new task," to demonstrate the flow of interactions between different objects in the system.

3. Implementation Phase

- **Definition:** This is the stage where the actual code is written based on the defined design.
- **Our Application:**
    - **Core Code Development:** The development team wrote Swift code to create and manage tasks (creation, assignment, marking as "Done").
    - **Swift Concepts Application:** Emphasis was placed on utilizing required fundamental Swift concepts such as Structs/Classes, Optionals (if let, guard let, nil coalescing, optional chaining), Arrays, Dictionaries, and Higher-Order Functions (filter, map).
    - **Error Handling:** try-catch blocks were integrated for simple examples of data validation (mock validation), demonstrating how potential errors would be handled.
    - **Parallel Work:** Different parts of the code were developed concurrently by team members, with continuous integration of changes via GitHub.

4. Testing Phase

- **Definition:** This phase involves verifying that the system functions correctly and meets the requirements.
- **Our Application:**
    - **Rapid Internal Review (Peer Review):** Team members conducted quick peer reviews of each other's code to identify logical or typographical errors.
    - **Playground Testing:** Xcode Playgrounds were used to immediately test core code functionalities and functions, ensuring they operated as expected.
    - **Requirement Verification:** A final quick review of all "Core Project Components" was performed to ensure that every requirement was met in the code or documentation.

5. Deployment/Handover Phase

- **Definition:** This is the final stage where the product is delivered to the client or relevant party.
- **Our Application:**
    - **Final GitHub Integration:** All team contributions were ensured to be merged into the main branch on GitHub, with the repository confirmed to be clean and organized.
    - **Presentation Preparation:** We prepared a concise presentation (5-7 minutes) to highlight our architectural design, core code, and application of Swift concepts.
    - **Demo:** A live demonstration (optional Playground demo) will be provided to illustrate the basic code functionality.
    - **Documentation Handover:** All required Artifacts (diagrams, SDLC description, code on GitHub) will be submitted as part of the final deliverable.

Conclusion: Despite the tight timeframe, we successfully applied an intensive and flexible software development life cycle. This approach allowed us to collaborate effectively, design a logical application structure, write essential code that meets the requirements, and provide documentation that demonstrates our understanding of the process.