**Assignment – 2**

|  |  |
| --- | --- |
| Student Name/ID Number: | Syukur Sidiq Nur Alam |
| Unit Number and Title: | Develop Project Proposal |
| Academic Year: |  |
| Unit Assessor: |  |
| Project Title: | Develop Enterprise Applications |
| Issue Date: |  |
| Submission Date: |  |
| Internal Verifier Name: |  |
| Date: |  |

|  |
| --- |
| **Learner declaration** |
|  |
| I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.  Student signature: Date: 27/06/2023 |

# **System Requirements: In Group**

### **SDLC (software development life cycle):**



SDLC, or Software Development Life Cycle, is a process that guides the development and modification of software products. It involves a series of planned activities to ensure efficient and high-quality software development. SDLC provides a framework for designing, developing, testing, and deploying software applications. It helps teams manage the entire development process, from start to finish, while meeting budget and requirements. By following SDLC, software development teams can create reliable and effective software solutions.

**How does SDLC work?**

SDLC works by providing a structured and systematic approach to software development. It outlines a series of phases or stages that software projects typically go through. Here's an overview of how SDLC works:

1. **Planning and Requirement Analysis:** This stage involves gathering requirements from customers, analyzing market surveys, and conducting feasibility studies. The project approach and quality assurance requirements are planned, and risks are identified.
2. **Defining Requirements:** The product requirements are clearly defined and documented in a Software Requirement Specification (SRS) document. Approval from the customer or market analysts is obtained.
3. **Designing the Product Architecture:** Based on the SRS, the product architecture is designed. Multiple design approaches may be proposed and evaluated. The best approach is selected, and the architectural modules and data flow are defined in a Design Document Specification (DDS).
4. **Building or Developing the Product:** The actual development work begins in this stage. The programming code is generated based on the DDS, following coding guidelines and using appropriate programming languages. This stage involves writing the code and implementing the defined design.
5. **Testing the Product:** Product testing is a continuous activity throughout the SDLC stages. In this specific stage, defects are reported, tracked, fixed, and retested until the product meets the defined quality standards in the SRS.
6. **Deployment in the Market and Maintenance:** Once the product is tested and ready, it is deployed in the market. This may happen in stages or targeted market segments. User acceptance testing (UAT) may be conducted to gather feedback. After deployment, maintenance is performed to support the existing customer base.

Based on feedback received, the product may be released with or without suggested enhancements in the target market segment. After the product is released, maintenance is performed to support existing customers and ensure its smooth functioning.

### **SDLC models: consider at least 4 models research:**

#### **Waterfall model**

* **What is Waterfall Model?**

The waterfall model is a straightforward and sequential approach to software development. It divides the development process into distinct phases, where each phase must be completed before moving on to the next. The model follows a linear flow without overlapping phases.

* **Characteristic**

1. **Sequential and Linear Process Flow :** The waterfall model follows a step-by-step approach where each phase is completed before moving on to the next. There is a clear order and no overlap between phases.
2. **Phases Do Not Overlap:** In the waterfall model, each phase is distinct and finishes before the next one begins. There is a sequential progression from one phase to another.
3. **Emphasis on Thorough Documentation:** Documentation is crucial in the waterfall model. Each phase requires comprehensive documentation, ensuring clarity and facilitating the transition to the next phase.
4. **Clear and Stable Requirements Upfront:** The waterfall model assumes that project requirements are well-defined and stable from the start. It focuses on gathering detailed requirements and freezing them early to minimize changes.

* **When we use waterfall model ?**

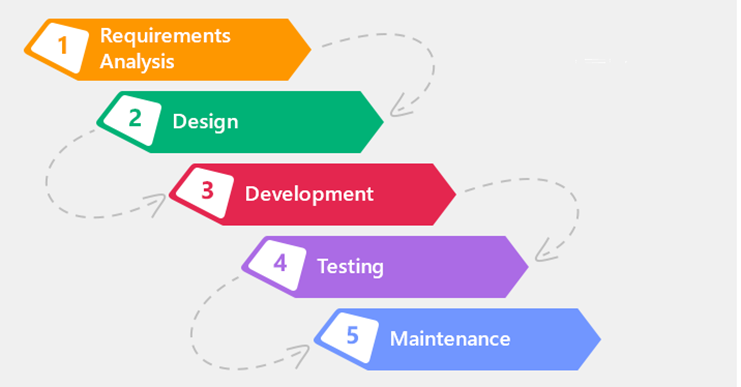
The waterfall model is most suitable for projects that have the following characteristics:

1. **Clear, well-defined requirements:** The project requirements are clearly understood and documented upfront, without significant changes expected during the development process.
2. **Stable technology and minimal uncertainty:** The technology and tools to be used in the project are well-established and stable, with minimal risk of unexpected issues or uncertainties arising during development.
3. **Fixed scope and strict deadlines:** The project has a well-defined scope that is unlikely to change significantly, and there are strict deadlines that need to be adhered to.
4. **Limited or no requirement for user involvement during development:** The project does not require extensive user involvement or frequent feedback and iterations during the development process. User input is not a critical factor in shaping the product.

* **Pros and Cons about waterfall model**

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| 1. Simple and easy to understand | 1. Limited flexibility for changes and adaptations |
| 1. Clear project requirements and scope | 1. Limited customer feedback until late in the cycle |
| 1. Thorough documentation for each phase | 1. High risk of customer dissatisfaction |
| 1. Well-suited for small and straightforward projects | 1. Difficult to estimate accurate time and cost |
| 1. Enables a structured and systematic approach | 1. Challenges in handling complex and evolving projects |

* **Waterfall Model Structure**



#### **Spiral model**

* **What is spiral model**

The spiral model is a software development process model that combines iterative development with the systematic, controlled aspects of the waterfall model. It emphasizes risk analysis and allows for incremental releases or refinement of the product through each iteration around a spiral.

* **Characteristic**

1. **Iterative and Incremental:** The development process occurs in repeated iterations, with each iteration refining the product based on feedback and lessons learned.
2. **Risk Analysis:** The model places a high emphasis on identifying, estimating, and mitigating risks throughout the development process.
3. **Phased Approach:** The model is divided into distinct phases, such as identification, design, construct/build, and evaluation, which are executed in iterations.
4. **Customer Involvement:** Continuous communication and feedback from the customer play a crucial role in shaping the product at each iteration.

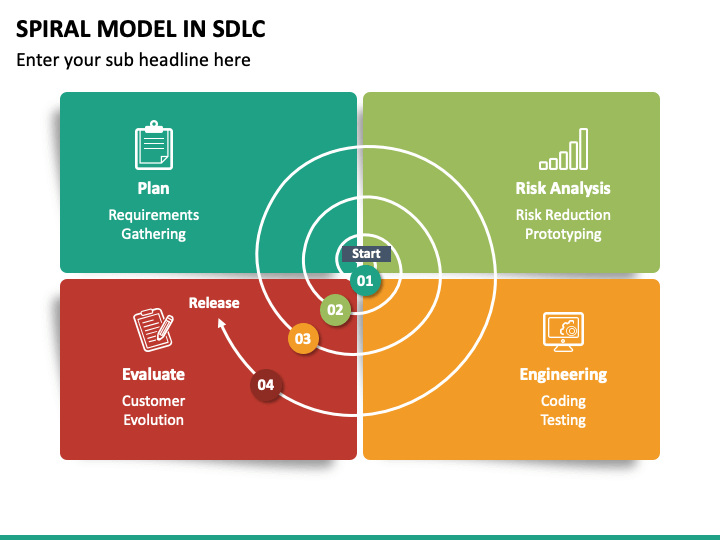
* **When we use spiral model ?**

1. Projects with changing or uncertain requirements.
2. Medium to high-risk projects that require a focus on risk analysis and management.
3. Long-term projects where economic priorities and requirements may evolve over time.
4. Projects with complex requirements that need evaluation and clarification.
5. Projects introducing a new product line that requires phased releases to gather customer feedback.
6. Projects anticipating significant changes during the development cycle.

* **Pros and Cons about spiral model**

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| 1. Accommodates changing requirements | 1. **Management complexity:** The iterative nature of the model increases the complexity of project management. |
| 1. Extensive use of prototypes | 1. **Project end may not be known early:** It can be challenging to estimate the project's completion due to the iterative nature of the model. |
| 1. Improved requirement capture | 1. **Not suitable for small or low-risk projects:** The spiral model may be too costly and time-consuming for smaller or low-risk projects. |
| 1. Early user involvement and feedback | 1. **Process complexity:** The model requires skilled project management and coordination to ensure smooth execution. |
| 1. Better risk management | 1. **Spiral may go on indefinitely**: Without proper management, the iterative nature of the model may lead to indefinite iterations. |

* **Spiral model structure**



#### **V Model**

* **What is V-Model?**

The V-Model is a software development life cycle (SDLC) model that follows a sequential and structured approach. It is called the V-Model because the execution of processes happens in a sequential manner in a V-shaped. It is also known as the Verification and Validation model, as it emphasizes testing at each corresponding development stage.

* **Characteristic**

1. **Sequential and structured:** The V-Model follows a sequential order, with each phase having a corresponding testing phase.
2. **Verification and validation:** It emphasizes the verification of requirements during development and the validation of the final product.
3. **Testing-centric:** Testing activities are integral to each phase, ensuring quality and adherence to requirements.
4. **Rigidity:** The V-Model is less flexible to changes, and modifications to requirements can be costly.
5. **Clear deliverables:** Each phase has specific deliverables and a review process, making it easier to manage and monitor progress.

* **When we use V model?**

1. **Well-defined requirements:** The V-Model is suitable when requirements are clear, documented, and fixed.
2. **Stable product definition:** When the product's definition is stable and unlikely to change significantly during development.
3. **Well-understood technology:** The project team should have a good understanding of the technology used.
4. **Short project duration:** The V-Model works well for smaller projects with a shorter duration.
5. **Minimal requirement changes:** It is not suitable for projects where requirements are prone to frequent changes.

* **Pros and Cons**

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| 1. **Easy to understand and apply:** The simplicity of the model makes it easy to comprehend and implement. | 1. **High risk and uncertainty:** The rigidity of the model increases the risk of changes and uncertainty. |
| 1. **Phases completed in a disciplined manner:** Each phase is completed before proceeding to the next, ensuring thoroughness. | 1. **Not suitable for complex projects:** It may not be suitable for complex and object-oriented projects that require more flexibility. |
| 1. **Suitable for well-defined requirements:** Works well when requirements are clear and documented. | 1. **Challenging for long-term projects:** The V-Model may not be ideal for projects with long durations and ongoing development. |
| 1. **Clear deliverables and review process:** Specific deliverables and reviews make it easier to manage and monitor progress. | 1. **Difficult to accommodate requirement changes:** Changes in requirements are costly and challenging to implement once in the testing stage. |

* **V model Structure**



#### **Agile model**

* **What is agile model?**

The Agile model is an iterative and incremental software development approach that focuses on adaptability and customer satisfaction by delivering working software products in small incremental builds. It involves cross-functional teams working simultaneously on various areas such as planning, requirements analysis, design, coding, unit testing, and acceptance testing. Each iteration, typically lasting from one to three weeks, concludes with a working product demonstration to the customer and stakeholders.

* **Characteristic**

1. Iterative and incremental development: The project is divided into small iterations, with each iteration delivering a working software build that incrementally adds features.
2. Adaptability: Agile recognizes that each project is unique and adapts the development approach accordingly to best suit the project's requirements.
3. Customer collaboration: Continuous customer interaction is essential to understand and incorporate evolving product requirements.
4. Rapid delivery: Agile emphasizes rapid delivery of working software, allowing for early feedback and course correction.
5. Self-organization and teamwork: Agile promotes self-organization and motivation within cross-functional teams, fostering collaboration and synergy.
6. Emphasis on working software: Rather than relying solely on documentation, Agile prioritizes demoing working software as the primary means of communication with customers.

* **When we use agile model?**

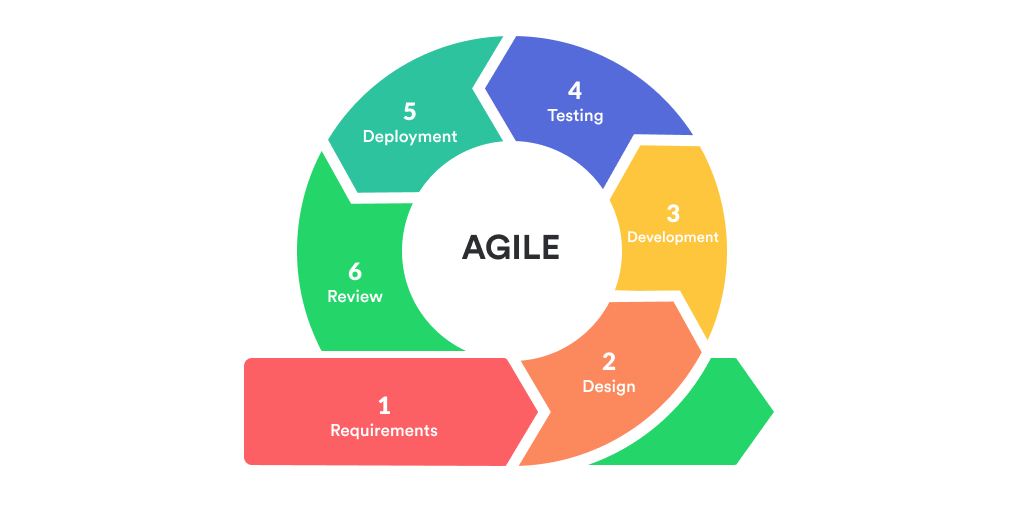
The Spiral model is a risk-driven software development approach that combines elements of both waterfall and iterative development models. It is typically used when a project has significant risk factors and a high degree of uncertainty. Here are some key points about the Spiral Model:

1. Risk management: The Spiral Model incorporates risk analysis and mitigation throughout the software development process.
2. Iterative approach: The development process progresses in iterations, with each iteration refining and adding new functionality.
3. Prototyping: The Spiral Model encourages the use of prototyping to gather feedback and validate design decisions.
4. Phases: The model consists of multiple phases, including planning, risk analysis, engineering, and evaluation. Each iteration focuses on these phases.
5. Feedback and evaluation: The project continually gathers feedback and evaluates the software to make informed decisions for subsequent iterations.

* **Pros and Cons about agile model**

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| 1. **Risk management:** The model emphasizes risk analysis and mitigation, reducing the chances of major failures. | 1. **Complexity:** The Spiral Model can be more complex and time-consuming to implement compared to other models. |
| 1. **Flexibility:** The Spiral Model accommodates changing requirements and can adapt to evolving project needs. | 1. **Resource-intensive:** The emphasis on risk analysis and prototyping may require additional resources and expertise. |
| 1. **Feedback-driven:** Frequent evaluation and feedback loops ensure that the product aligns with stakeholder expectations. | 1. **Lack of transparency:** The model's iterative nature may make it challenging to estimate project timelines and costs accurately. |
| 1. **Iterative refinement:** The model allows for incremental development and improvement over successive iterations. | 1. **Documentation overhead:** The Spiral Model typically requires comprehensive documentation, which can be time-consuming and resource-intensive. |

* **Agile model structure**



### **Selected SDLC Model : Waterfall Model**

Our teams have decided to choose the waterfall model for this project due to several factors :

1. The project description provided a comprehensive understanding of the project's background, scope, and requirements.
2. The features and deliverables were well-defined
3. The waterfall model aligns with this need for a clear understanding of project requirements from the outset.

By choosing the waterfall model, we can ensure that the project remains focused on meeting the predefined requirements and scope. The well-defined nature of the waterfall model allows for a structured approach, minimizing the need for frequent changes and iterations. This approach provides clarity and stability throughout the development process, ensuring that the project stays on track to meet its goals within the specified timeframe.

**Front-end Tech**

|  |  |  |
| --- | --- | --- |
| Technologies | Pros | Cons |
| HTML | Simple and easy to learn | Limited interactivity and dynamic functionality |
| Widely supported and compatible | Requires manual handling of state and UI updates |
| Good for static content | Steeper learning curve for complex layouts |
| React JS | Efficient and fast rendering | Steeper learning curve for beginners |
| Component-based architecture for reusability | Requires additional tools and configuration for complex applications |
| Large ecosystem and community support | Not suitable for small and simple projects |
| Vue JS | Easy integration with existing projects | Smaller ecosystem compared to React and Angular |
| Simple syntax and gentle learning curve | Limited enterprise-level support |
| Flexible and scalable | May have performance issues with larger applications |
| Angular JS | Full-featured framework with extensive tooling | Steeper learning curve compared to other front-end frameworks |
|  | Strong support for large-scale applications | Requires adherence to Angular conventions and patterns |
|  | Two-way data binding for automatic UI updates | Limited performance optimizations for smaller projects |

**Back-end Tech.**

|  |  |  |
| --- | --- | --- |
| Technologies | Pros | Cons |
| Laravel | Elegant syntax and developer-friendly | Requires PHP as the programming language |
| Robust ecosystem and community support | Limited scalability for extremely high-traffic applications |
| Built-in ORM for database interaction | Slower performance compared to some other frameworks |
| Spring Boot | Highly scalable and enterprise-ready | Requires knowledge of Java programming language |
| Excellent support for microservices architecture | Steeper learning curve for beginners |
| Integrated security and dependency management | Initial setup and configuration can be complex |
| Java | Platform-independent and widely used | Verbose syntax compared to some other languages |
| Strong ecosystem and extensive libraries | Requires knowledge of object-oriented programming concepts |
| Excellent performance and scalability | Longer development time for certain tasks |
| Express JS | Minimalistic and flexible framework for building APIs | Requires additional modules for advanced features and functionalities |
|  | Middleware support for handling HTTP requests | Lacks some built-in features compared to more opinionated frameworks |
|  | Efficient and lightweight | Limited support for database interaction out of the box |

**DATABASE:**

|  |  |  |
| --- | --- | --- |
| Technologies | Pros | Cons |
| MySQL | High performance and scalability | Lacks some advanced features of other databases |
| Wide community support and extensive documentation | Limited support for NoSQL capabilities |
| Good for relational data and complex queries | May have limitations in handling large datasets |
| PostgreeSQL | Feature-rich and highly extensible | Requires more resources compared to some other databases |
| Excellent support for complex queries and advanced SQL | Steeper learning curve for beginners |
| High data integrity and reliability | May have limited support for certain programming languages |
| SQLite | Lightweight and easy to set up | Not suitable for large-scale deployments |

**IDE:**

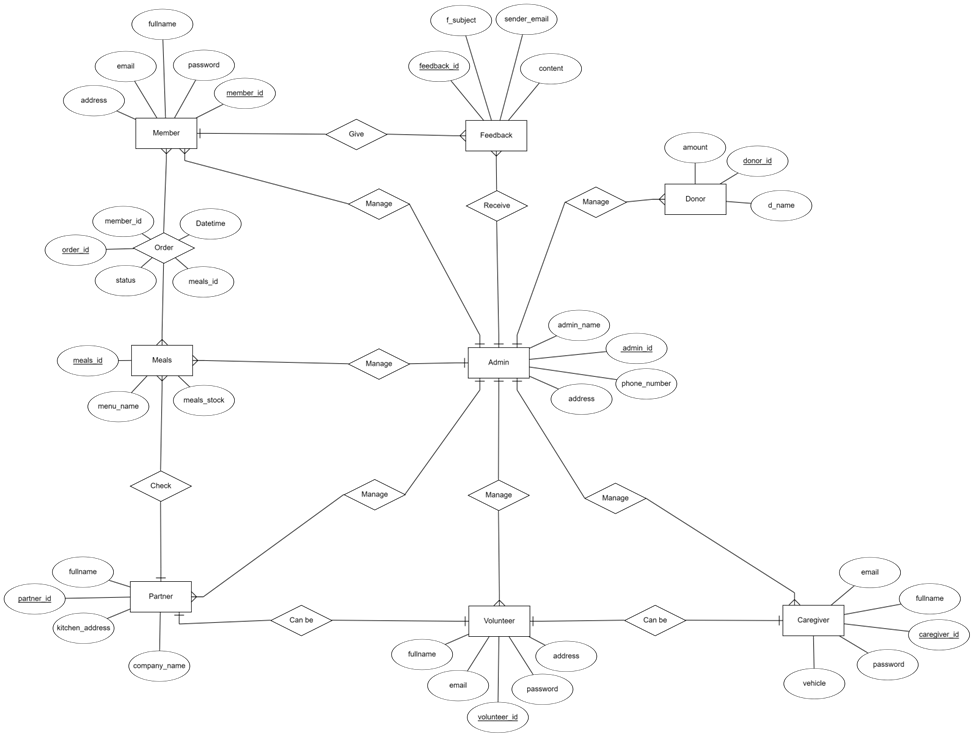
|  |  |  |
| --- | --- | --- |
| Technologies | Pros | Cons |
| VS Code | Lightweight and fast editor with a wide range of extensions | Lack of some advanced features found in full IDEs |
| Excellent support for web development and JavaScript ecosystem | Steeper learning curve for beginners |
| Intuitive user interface and customizable settings | Requires additional configuration for certain languages |
| IntelliJ IDE | Robust and feature-rich IDE for Java and JVM languages | Heavier and slower compared to lightweight code editors |
| Advanced code analysis and debugging tools | Steeper learning curve for beginners |
| Intelligent code completion and refactoring capabilities | Consumes more memory and resources than lightweight editors |
| Seamless integration with build tools and version control | Paid version (Ultimate Edition) includes additional features |
| SQLite | Specialized IDE for Spring Framework development | Limited support for other programming languages |
| Built-in tools for Spring Boot configuration and deployment | Steeper learning curve for beginners |
| Integrated support for testing and debugging Spring projects | Not as feature-rich as general-purpose IDEs |
| Free and open-source | Relatively smaller community compared to other IDEs |

**Selecting Techlogies :**

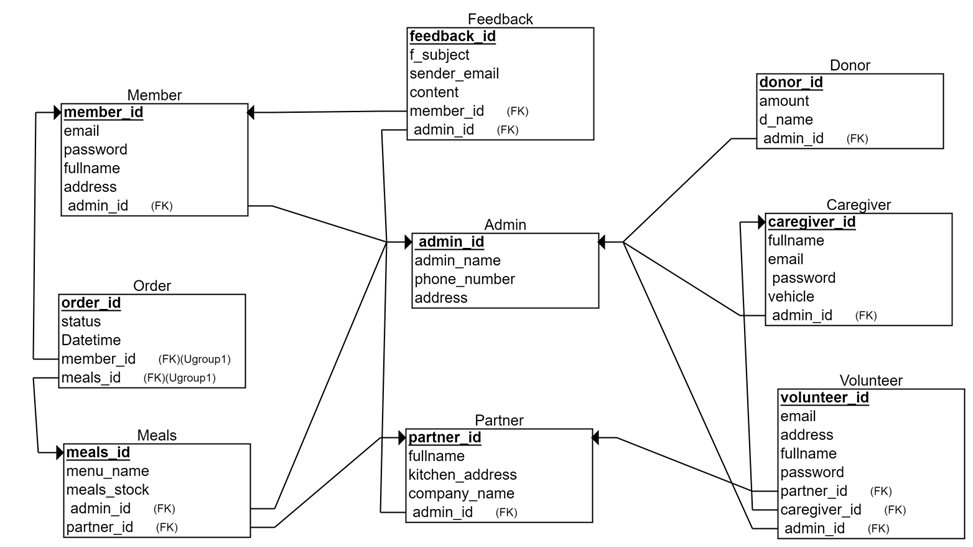
1. **Front-End Tech: HTML, CSS, JS, Boostrap / React Boostrap**
2. **Back-end Tech.: Spring Boot**
3. **Database: MySQL**
4. **IDEs: VSCode for both Front-End and Back-End**
5. **Tools: Figma, draw.io, Microsoft, ERDplus**

### **DB Design: ERD diagram and EERD(Physical diagram)**

**EERD DIAGRAM**

****

**Relational Schema**

****