**Assignment – 2**

|  |  |
| --- | --- |
| Student Name/ID Number: | Muhammad Kemal/BDSE-0922-084 |
| Unit Number and Title: | Develop Project Design |
| Academic Year: | 2022 |
| Unit Assessor: | Arvinder Kaur |
| Project Title: | Develop Enterprise Applications |
| Issue Date: |  |
| Submission Date: |  |
| Internal Verifier Name: |  |
| Date: | 22 June 2023 |

|  |
| --- |
| **Learner declaration** |
|  |
| I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.  Student signature:  Date: 22 June 2023 |

**MEALS ON WHEELS**

**Assignment 2 - Project Design**

****

**MUHAMMAD KEMAL (BDSE-0922-084)**

**SOFTWARE ENGINEERING**

**LITHAN EDUCLAAS**

**(2023)**

# **System Requirements: In Group**

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

The phases and processes involved in developing software applications are described in the Software Development Life Cycle (SDLC) framework. It provides a systematic strategy for developing software, ensuring that activities are well planned, carried out, and maintained.

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

The SDLC (Software Development Life Cycle) is a systematic approach for creating software applications. It divides the development process into various phases, each of which has its own set of duties and results.

The normal operation of the SDLC is described in the following detail:

Gathering requirements: The SDLC's initial stage involves gathering the requirements of stakeholders. It's important to comprehend the needs, objectives, and expected functionality of the product. To concisely and clearly express the requirements, stakeholders work closely with project managers, business analysts, and developers.

System Design: The system design process starts after the requirements have been acquired. During this phase, the development team creates the user interface, database structure, software architecture, and other elements. The objective is to produce a blueprint that describes how the program will be created and how various components will work together.

Implementation: The actual coding and software development happen during this stage. Based on the design criteria, developers use the right programming languages and tools to write code. They adhere to coding standards, best practices, and version control to guarantee code quality, readability, and maintainability.

Testing: The software is tested after the implementation phase to find any flaws or problems and correct them. To verify the software's functionality, performance, and usability, many testing techniques, including unit testing, integration testing, system testing, and user acceptability testing, are used.

Deployment: The software is prepared for deployment after completing the testing phase. The program is installed, set up, and made available to end users in the production environment.

Maintenance: The software then moves into the maintenance phase. Performance, stability, and security of the software are all kept under observation throughout this stage. Users' complaints are addressed, and any necessary changes or improvements are provided. Routine maintenance procedures ensure the longevity and proper performance of the program.

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

#### **Waterfall model**

* **What is Waterfall Model?**

The first SDLC methodology used for software development was the waterfall approach. This approach is applied methodically, beginning with the stage of system requirements and continuing through the stages of analysis, design, coding, testing/verification, and maintenance. Each step must be accomplished one at a time.

**Characteristic**

* Sequential and Linear: The development process follows structured and sequential steps.
* Good Documentation: Emphasis on extensive documentation for each phase.
* Limited Customer Engagement: Customer interaction is limited to early and late stages.
* Limited Iteration: Changes can be difficult after the phase is over.
* Predictability and Planning: Enables comprehensive planning based on a clear sequence.
* Suitable for Stable Requirements: Ideal for projects with clear and stable requirements.
* Not Suitable for Complex Projects: Less flexible for projects with complex or changing requirements.
* Verification and Validation: Focus on verification and validation activities in each phase.
* Review and Approval Documented: The review and approval process is important.

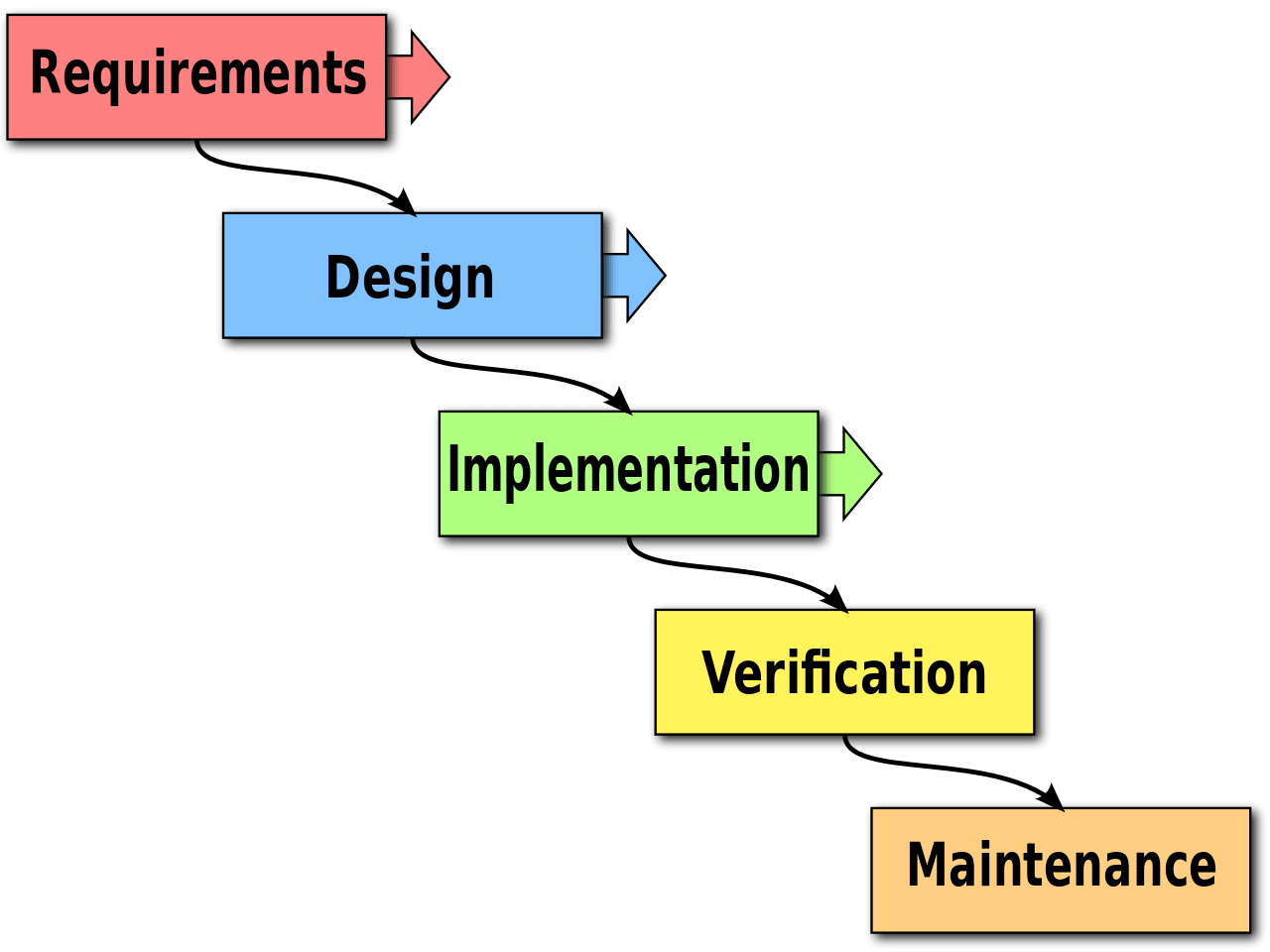
**Where do we use it?**

* Requirements are very well documented, clear and fixed.
* Product definition is stable.
* Technology is understood and is not dynamic.
* There are no ambiguous requirements.
* Ample resources with the required expertise are available to support the product.
* The project is short.

**Pros and Cons**

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| 1. Simple and easy to understand and use 2. Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process. 3. Phases are processed and completed one at a time. 4. Works well for smaller projects where requirements are very well understood. | 1. No working software is produced until late during the life cycle. 2. High amounts of risk and uncertainty. 3. Not a good model for complex and object-oriented projects. 4. Poor model for long and ongoing projects. |

* **Waterfall Model Structure**



#### **Spiral model**

* **What is the spiral model**

The Spiral Model is a method that can be used in software development. The spiral model is a combination of the prototyping model and the waterfall model. A prototyping model that focuses on presenting or presenting to the user with input and output formats will then evaluate the software. The waterfall model focuses on systematic or sequential software development processes. Spiral pressure model in the risk analysis for each stage.

* **Characteristic**

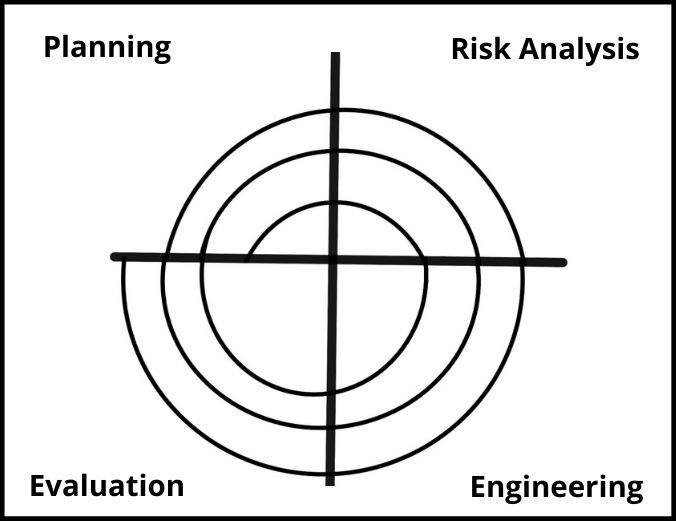
-Identification: This phase initially involves gathering the business requirements and then identifying system, subsystem, and unit requirements in subsequent spirals. It emphasizes continuous communication between the customer and the system analyst to understand the system requirements. The product is deployed in the identified market at the end of the spiral.

-Design: The Design phase begins with the conceptual design in the baseline spiral and progresses to architectural design, logical design of modules, physical product design, and the final design in subsequent spirals. This phase focuses on developing the design of the software product.

-Construct or Build: The Construct phase involves the production of the actual software product in each spiral. A Proof of Concept (POC) is developed in the baseline spiral to get customer feedback. In subsequent spirals, working models of the software called builds are produced with increasing clarity on requirements and design details. These builds are sent to the customer for feedback.

-Evaluation and Risk Analysis: This phase includes risk analysis, which involves identifying, estimating, and monitoring technical feasibility and management risks such as schedule slippage and cost overrun. After testing the build, the customer evaluates the software and provides feedback at the end of the first iteration.

* **Spiral model structure**



* **When using a spiral model**
* When the creation of a prototype is appropriate.
* When costs and risk evaluation are important.
* For medium to high-risk projects.
* Long-term project commitment is unwise because of potential changes to economic priorities.
* Users are unsure of their needs.
* Requirements are complex.
* New product line.
* Significant changes are expected (research and exploration)

**Pros and Cons**

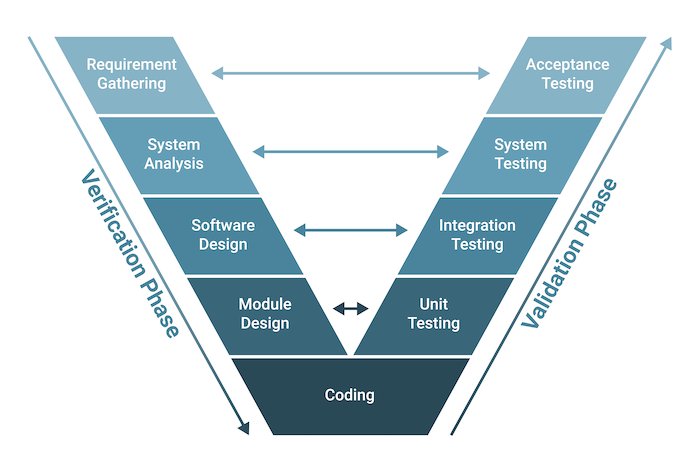
|  |  |
| --- | --- |
| **Pros** | **Cons** |
| 1. Changing requirements can be accommodated. 2. Allows extensive use of prototypes. 3. Requirements can be captured more accurately. 4. Users see the system early. 5. Development can be divided into smaller parts, and the risky parts can be developed earlier, which helps in better risk management. | 1. It is harder to manage. 2. The project's end may not be recognized right away. 3. It is inappropriate for little-risk or low-risk initiatives and might be costly for modest enterprises. 4. Process is difficult 5. Spiral might continue forever. 6. Numerous intermediary steps necessitate much paperwork. |

#### **V Model**

* **What is V-Model?**

A testing phase is linked to each relevant development step in the V-Model, which is an extension of the waterfall model. This implies that there is a testing phase directly related to every phase of the development cycle. This is a very rigorous strategy, the next phase doesn't begin until the previous phase is finished.

* **Characteristic**
  1. Project and Requirements Planning - Allocate resources: This stage involves planning the project and allocating the resources required for software development. This includes identifying project needs, scheduling, and allocating teams and other resources.
  2. Product Requirements and Specification Analysis - complete specification of the software system: Product requirements analysis is carried out to produce a complete specification of the software system to be developed. This includes determining functional and non-functional requirements and preparing detailed specification documents.
  3. Architecture or High-Level Design - defines how software functions fulfill the design: This stage involves architectural design or high-level design that determines how software functions will fulfill the predefined design. This design provides an overview of the structure and components of the system.
  4. Detailed Design - develop algorithms for each architectural component: At this stage, detailed designs are created for each predefined architectural component. More detailed algorithms and logic are developed for each software component.
  5. Coding - transform algorithms into software: This stage involves implementing software code based on predefined designs and algorithms. The software code is developed according to the relevant programming language.
  6. Unit Testing - check that each module acts as expected: At this stage, the software modules are tested separately to verify that each module functions as expected. This test is carried out at the smallest component level in the software.
  7. Integration and Testing - check that modules interconnect correctly: This stage involves integration testing in which modules that have been tested separately are tested again to verify that they can connect and interact correctly according to the established design.
  8. System and Acceptance Testing - check the entire software system in its environment: At this stage, the software system is thoroughly tested to verify that it functions properly in the intended environment. This test aims to ensure that the system meets the requirements that have been set from the start and can be accepted by users.
  9. Production, Operation, and Maintenance - Provide for enhancement and corrections: This stage involves delivering the software product to the production environment and ensuring smooth operation. This stage also includes software system maintenance, including provision of necessary upgrades and fixes throughout the product life cycle.
* **V model Structure**



* **When do we use the V model?**
* Excellent choice for systems requiring high reliability – hospital patient control applications.
* All requirements are known up-front.
* When it can be modified to handle changing requirements beyond the analysis phase.
* Solution and technology are known.

**Pros and Cons**

|  |  |
| --- | --- |
| Pros | Cons |
| 1. Emphasize planning for verification and validation. 2. Each deliverable must be testable. 3. Project management can track progress by milestones. 4. Easy to use. | 1. Easily handle concurrent events 2. Handle iterations or phases 3. Easily handle dynamic changes in requirements 4. Contain risk analysis activities requirements |

#### **Agile model**

* **What is the Agile model?**

The Agile model is an incremental software development approach emphasizing flexibility, collaboration, and iterative cycles. It is designed to deliver working software in small increments, with each increment building upon the functionality of the previous one. The Agile model aims to respond quickly to changing requirements and customer feedback, allowing for greater adaptability throughout development.

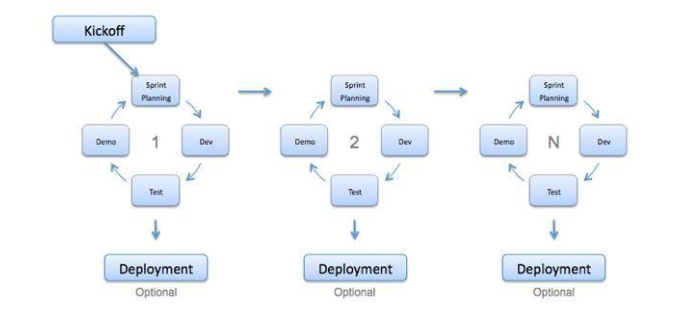
* **Characteristic**

Key characteristics of the Agile model include:

1. Incremental Development: The software is developed and delivered in small increments or iterations, typically called sprints. Each iteration focuses on specific functionality and is designed to be a complete, usable product.
2. Rapid Cycles: The development process comprises short, time-boxed cycles. These cycles, often from one to four weeks, enable frequent releases and continuous improvement based on customer feedback and changing requirements.
3. Building on Previous Functionality: Each iteration adds new features or enhancements to the existing functionality. The development team aims to deliver a working product at the end of each iteration, ensuring that the software evolves gradually.
4. Thorough Testing: Testing is integral to the Agile model. Each iteration undergoes comprehensive testing to maintain software quality. Testing activities include unit testing, integration testing, and user acceptance testing.
5. Time-Critical Applications: The Agile model is particularly suitable for projects with time-critical requirements or where the business landscape is subject to frequent changes. It allows for quick adaptations, making responding to emerging market needs or technological advancements easier.
6. Extreme Programming (XP): Extreme Programming is one of the methodologies commonly associated with the Agile model. XP emphasizes continuous integration, test-driven development, pair programming, and frequent communication to foster collaboration and deliver high-quality software.

The Agile model promotes an iterative and collaborative approach to software development, ensuring that the final product meets the customer's needs while accommodating changes and improvements throughout the project lifecycle.

* **Agile model structure**



* **When do we use the agile model?**

1. When new changes are needed to be implemented.
2. New changes can be implemented at a minimal cost.
3. Very limited planning is required to get started with the project.
4. Assumes that the end users’ needs are ever-changing in a dynamic
5. business and IT world.
6. Changes can be discussed, and features can be added.
7. This effectively gives customers the finished system they want or need.
8. Get more freedom of time and options.

**Pros and Cons**

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| 1. Implement the simplest solution to today's problem: Agile encourages finding simple and practical solutions to immediate problems rather than over-engineering or overthinking. 2. Customer satisfaction by rapid, continuous delivery of valuable software: Agile focuses on delivering working software in short cycles, enabling frequent customer feedback and ensuring their satisfaction. 3. People and interactions are emphasized rather than process and tools: Agile prioritizes effective communication, collaboration, and teamwork among individuals involved in the project, recognizing their importance over rigid processes or tools. 4. Customers, developers, and testers constantly interact with each other: Agile promotes ongoing interaction and collaboration between the development team, customers, and testers to ensure a shared understanding and alignment of goals. 5. Continuous attention to technical excellence and good design: Agile emphasizes the importance of maintaining technical excellence and incorporating appropriate design principles throughout development. 6. Regular adaptation to changing circumstances: Agile is designed to accommodate changes and adapt to evolving requirements, enabling flexibility and responsiveness. 7. Test-driven development: Agile promotes the practice of writing tests before writing the code, ensuring that the software meets the specified requirements and remains stable during iterative development. 8. Shared code ownership: Agile encourages collective code ownership, fostering collaboration and knowledge sharing among team members. | 1. Difficulty assessing effort for significant deliverables: Estimating effort and timelines can be challenging when dealing with significant deliverables at the beginning of the project. 2. Lack of emphasis on necessary designing and documentation: Agile's focus on iterative development and rapid delivery can sometimes lead to inadequate documentation or a lack of comprehensive design, which may be required for specific projects or organizations. 3. Lack of direction without clear customer representation: Agile heavily relies on customer collaboration and input. If the customer representative is not transparent or available, it can lead to a lack of direction and potential misunderstandings. 4. Decision-making requiring experienced resources: Agile assumes senior programmers or experienced team members can make critical decisions during development. This may limit the involvement or growth opportunities for newer or less experienced programmers. |

### **Selected SDLC Model**

* **Waterfall Module**

**Front-end Tech**

|  |  |
| --- | --- |
| **HTML** | **Pros:**   * Easy to follow the sequential development process. * Clear documentation can be created for each phase.   **Cons:**   * Limited flexibility for accommodating changes during development. |
| **React JS** | **Pros:**   * Well-defined structure and clear separation of components. * Extensive documentation and community support facilitate the sequential development process.   **Cons:**   * Changes or additions to requirements after the initial phase can be challenging. |
| **VUE js** | **Pros:**   * Clearly defined phases align well with the Waterfall model. * Comprehensive documentation aids in sequential development.   **Cons:**   * Limited flexibility in accommodating changing requirements or iterative development. |
| **Angular Js** | **Pros:**   * Phased development aligns with the Waterfall model. * The well-documented framework supports a systematic approach.   **Cons:**   * Challenging to incorporate changes or feedback once a phase is completed. |

**Back-end Tech.**

|  |  |
| --- | --- |
| Laravel | **Pros:**   * The well-structured framework aligns with the Waterfall model. * Extensive documentation facilitates sequential development.   **Cons:**   * Limited flexibility to accommodate evolving requirements or changes in later stages. |
| Spring Boot | **Pros:**   * Clearly defined phases support the Waterfall model. * Vital documentation and community support aid in sequential development.   **Cons:**   * Challenges arise when accommodating changes or feedback after the initial phase. |
| **Java** | **Pros:**   * Strongly typed language aligns well with the Waterfall model's structured approach. * Extensive documentation and resources support a sequential development process.   **Cons:**   * The rigidity of the Waterfall model may make it challenging to adapt to changing requirements. |

**DATABASE:**

|  |  |
| --- | --- |
| MySQL | **Pros:**   * Well-established and widely used database aligns with the Waterfall model. * A structured approach enables comprehensive planning and documentation.   **Cons:**   * Limited flexibility for accommodating changing database requirements during development. |
| PostgreSQL | **Pros:**   * Clearly defined phases facilitate sequential development. * Comprehensive documentation aids in the Waterfall model's structured approach.   **Cons:**   * Incorporating changes to database requirements after the initial phase can be challenging. |
| SQLite | **Pros:**   * A lightweight and easily integrated database aligns with the Waterfall model's structured approach. * Clear documentation supports sequential development.   **Cons:**   * Limited scalability and flexibility for accommodating evolving database needs. |

**IDE:**

|  |  |
| --- | --- |
| VS code | **Pros:**   * The sequential workflow of the Waterfall model can be quickly followed. * Lightweight IDE facilitates documentation and planning.   **Cons:**   * Occasional crashes or lags may disrupt the sequential development process. |
| INTELLIJ IDE | * **Pros:** * IDE that is comprehensive and enables a systematic approach. * Extensive features and tools aid in sequential development.   **Cons:**   * Higher resource requirements may impact the performance of the Waterfall model. |
| STS | **Pros:**   * Tailored for Spring framework development, aligning with the Waterfall model. * Seamless integration with the Spring ecosystem supports a structured approach.   **Cons:**   * Limited support for other technologies may restrict flexibility in the sequential development process. |

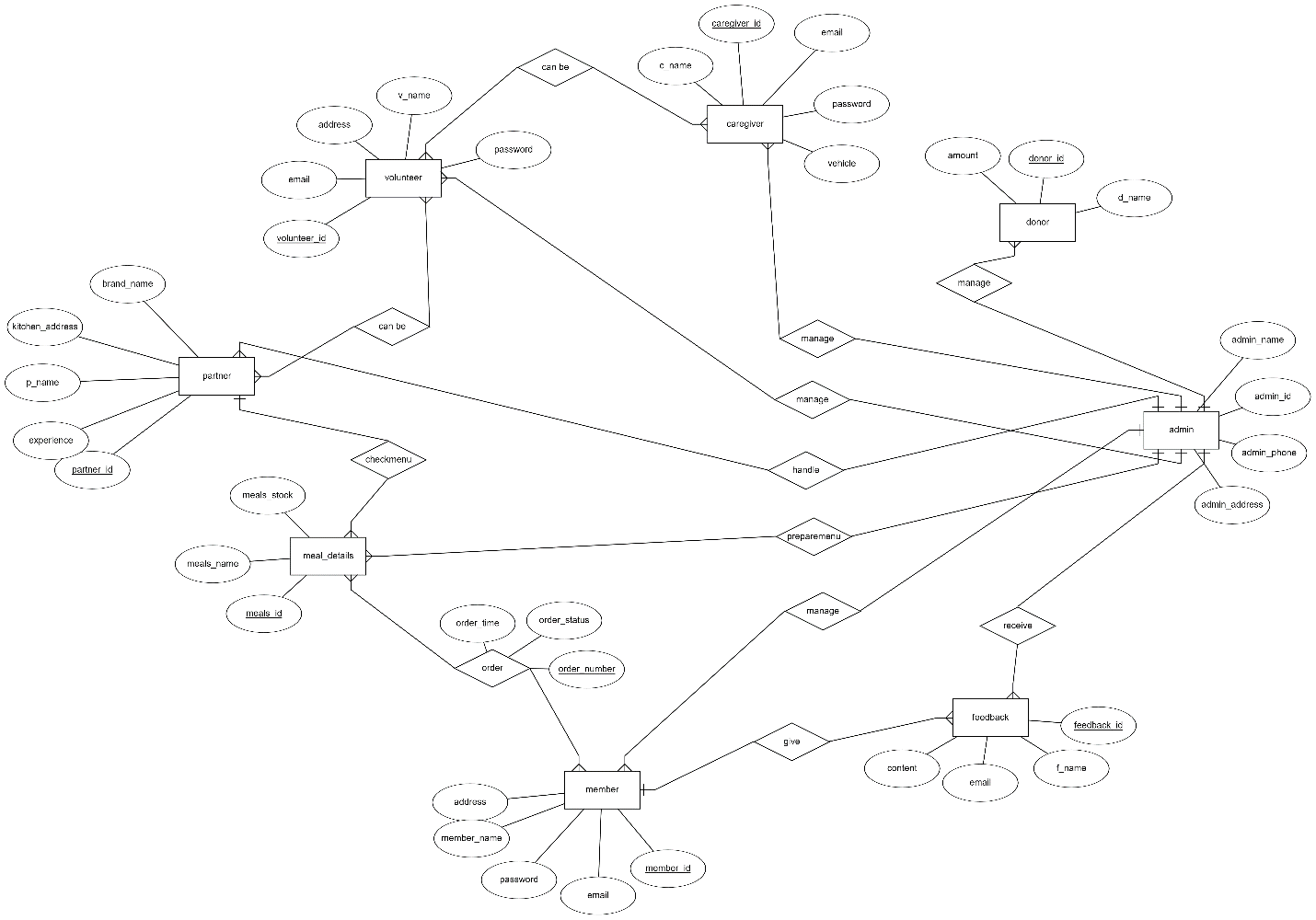
Overall, implementing the Waterfall model with the given technologies can provide a structured and systematic approach to development. The strengths lie in the alignment of the technologies with the sequential nature of the model, comprehensive documentation, and support available. However, the rigidity of the Waterfall model may limit the ability to accommodate changes or evolving requirements, which can be considered a significant drawback in a dynamic and fast-paced development environment.

**Selecting Technologies etc. :**

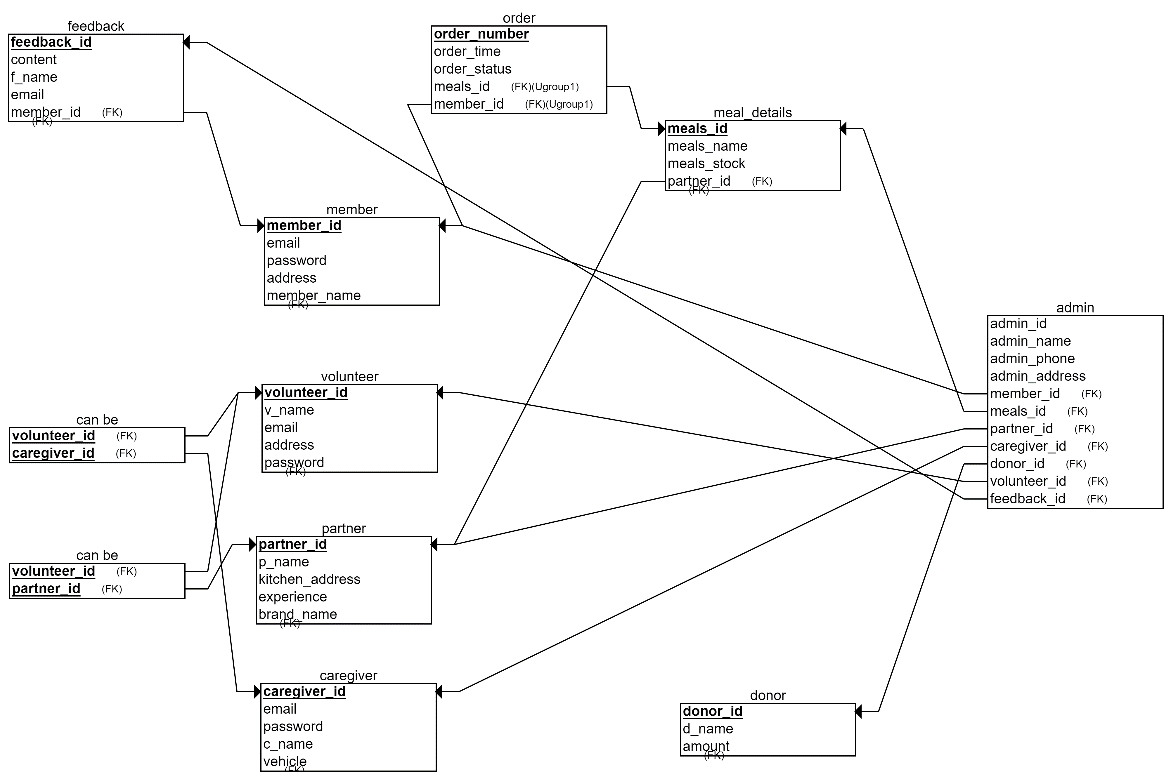
1. **Front-End Tech: HTML, CSS, JS, React JS, Bootstrap**
2. **Back-end Tech.: Spring Boot**
3. **Database: SQL (MySQL)**
4. **IDEs: VSCode**
5. **Tools: Figma, draw.io, Google document, ERDplus, Axure, Postman.**

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

**EERD Diagram**

****

**Physical Diagrram**

****