PES UNIVERSITY

Department of Computer Science and Engineering

UE21CS341A: Software Engineering

Format 3

Project Plan Document

Table of Contents :

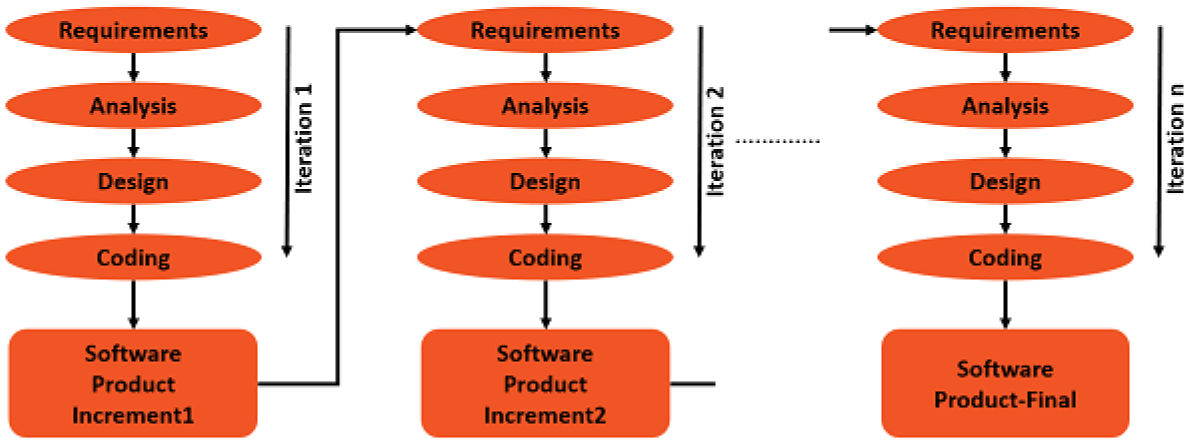
Serial Number (Sl No)	**Topics**
1	*Lifecycle used*
2	*Tools used*
3	*Project Deliverables*
4	*WBS*
5	*Rough Estimate in Man Days*
6	*UML Diagrams*

Lifecycle Used :

For our project AquaDB Management System (a warehouse management system project) that includes various components like inventory management, employee management, and store-related information, the choice of a software development lifecycle (SDLC) model is what we have estimated to be critical to the project's success.

Under SDLC we have opted for the Incremental Development approach.

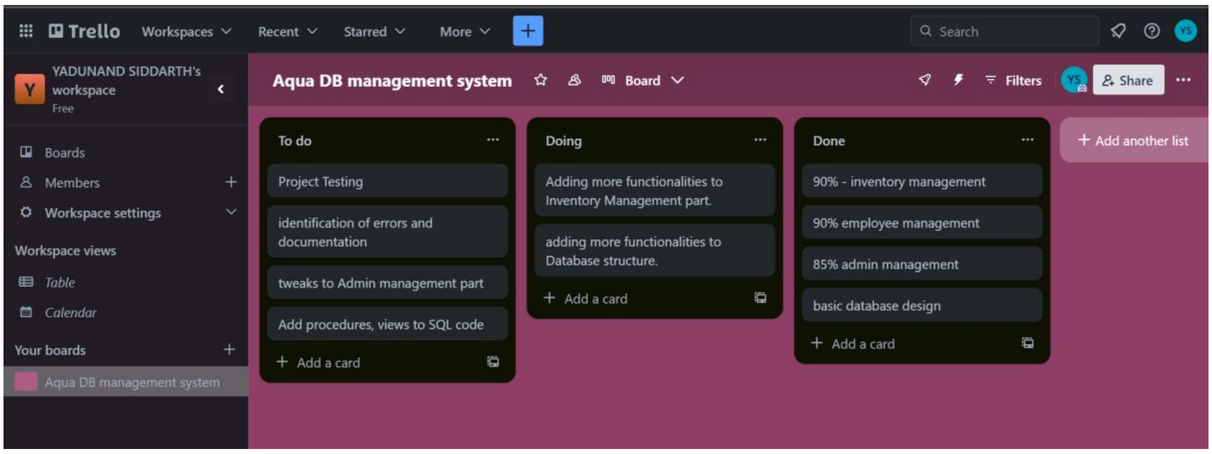
Incremental development involves dividing the project into smaller, manageable parts or increments. Each increment represents a portion of the system's functionality. It allows us to build and deliver discrete, functional components of the system incrementally. In the context of our project, we can use incremental development to focus on the inventory management module as the first increment, followed by the employee management module in the second increment. Incremental development is suitable for delivering specific, critical functionality early in the project and then build upon it.



Tools Used :

Planning Tool:

As a planning tool we have decided to use Trello. It is a tool that offers visual and flexible way to organize tasks and collaborate within the AquaDB management system project. It uses boards, lists, and cards to represent project workflows and tasks, making it easy to track progress, assign responsibilities, and set priorities.



Aqua DB management system

To do
- Project Testing
- identification of errors and documentation
- tweaks to Admin management part
- Add procedures, views to SQL code

Doing
- Adding more functionalities to Inventory Management part.
- adding more functionalities to Database structure.

Done
- 90% - inventory management
- 90% employee management
- 85% admin management
- basic database design

Design Model:

We are using multi-tier architecture for a AquaDB management system which can help organize and separate various components and functionalities within the system. In this context, the multi-tier architecture can comprise the following layers:

Presentation Layer:

- User Interface (UI): This layer represents the user-facing part of the system, where administrators, employees, and other users interact with the PROJECT. It includes web-based or mobile-friendly UI components.
- Admin Dashboard: Administrators access an intuitive and feature-rich dashboard for managing the system, which includes inventory, employee information, and transactions.
- Employee Portal: Employees access their profiles, view payroll information, and submit advance requests through an employee portal.
- Store Interface: If applicable, a store interface may be included for managing store-related information.

Application Layer:

- Business Logic: This layer contains the core business logic and application services of the PROJECT. It processes requests from the presentation layer, communicates with the data layer, and orchestrates various functions.

- Inventory Management Service: Manages all aspects of inventory, including product information, stock levels, location tracking, and product transactions (buying, renting, pre-ordering).
- Employee Management Service: Handles employee profiles, payroll processing, and advance request management.
- Transaction Processing: Manages and processes various types of transactions, such as product purchases, rentals, and pre-orders.

Data Layer:

- Database Management System (DBMS): The data layer includes the database(s) where structured and unstructured data is stored and managed.
- Inventory Database: Stores product details, stock levels, transaction records, and related data.
- Employee Database: Contains employee profiles, payroll information, and advance request records.
- Store Database (if applicable): Stores store-related data and information.

Using multi-tier architecture offers several advantages:

Modularity: It allows for the separation of concerns, making it easier to develop, maintain, and scale each layer independently.

Scalability: You can scale each layer as needed to accommodate growing data and user loads.

Security: It enables the implementation of security measures at each layer to protect data and restrict access.

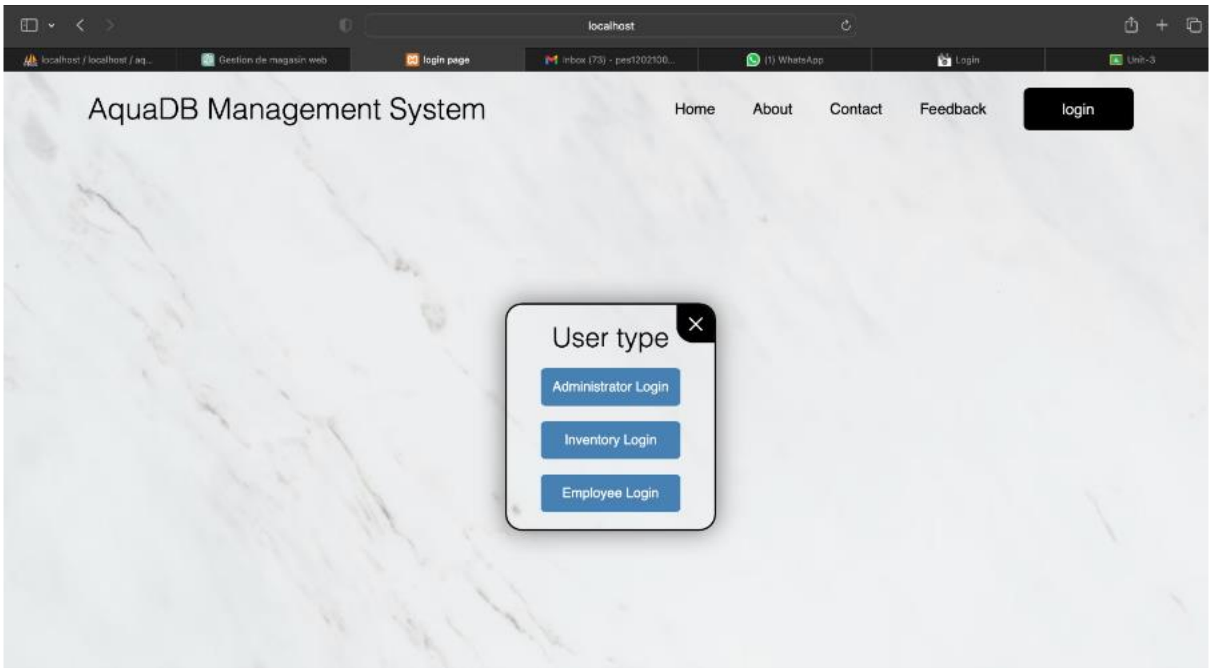
Flexibility: Different technologies and tools can be used in each layer to best fit the requirements.

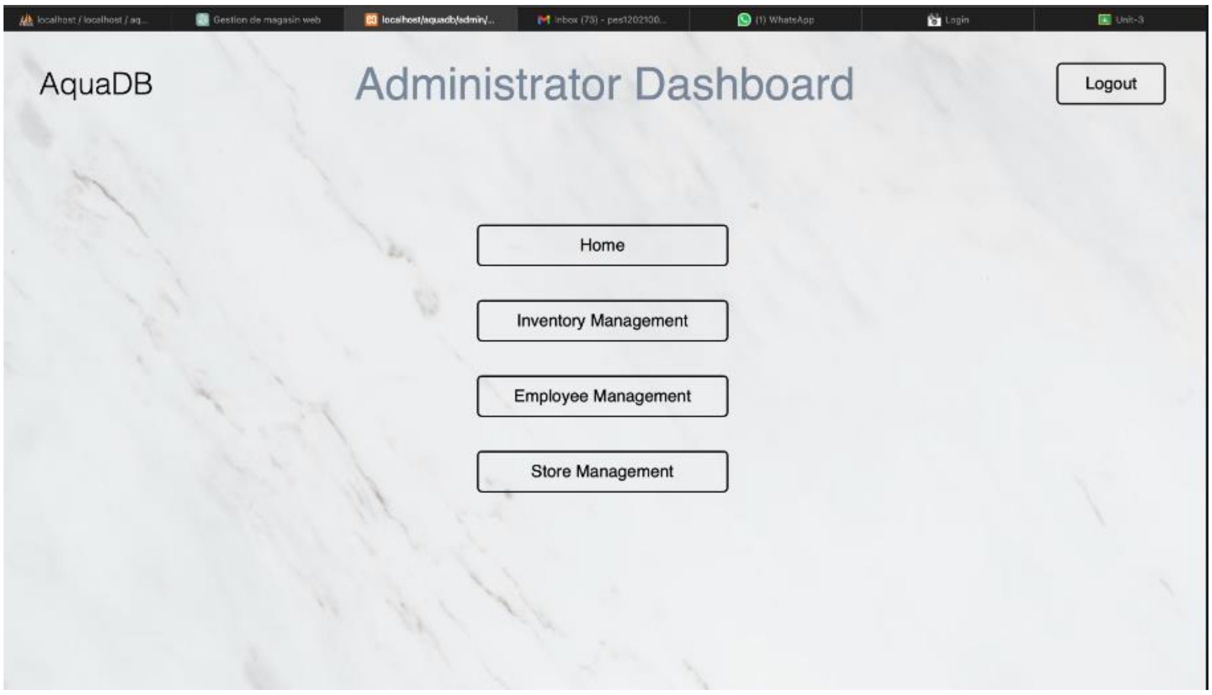
By using multi-tier architecture, AquaDB can efficiently manage inventory, employee data, and transactions while providing an accessible and user-friendly interface for administrators and employees. This architecture helps maintain system integrity, performance, and security while allowing for future enhancements and expansions.

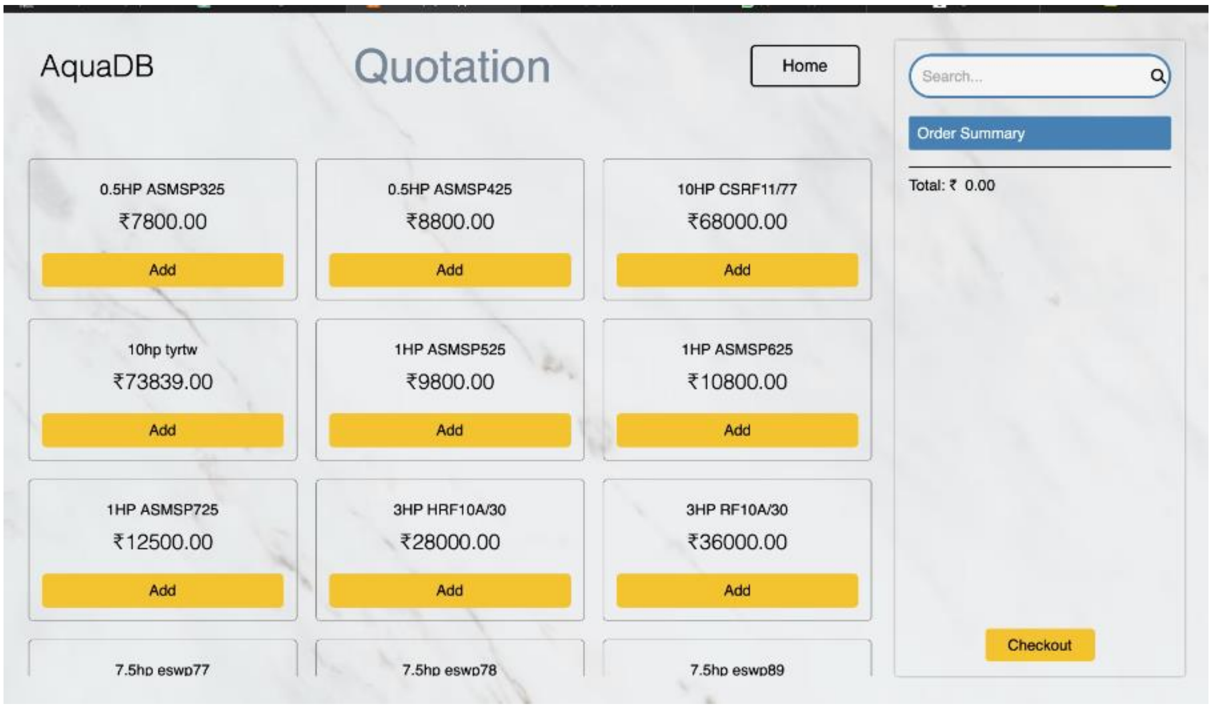
<u>Design Tool:</u>

Used Wireframe platform to design a basic structure and implemented html related pages taking that as a reference. Looked up some other HTML projects on Google and GitHub for design inspiration.

A few designed pages of our project:







0.5HP ASMSP325 ₹7800.00

0.5HP ASMSP425 ₹8800.00

10HP CSRF11/77 ₹68000.00

10hp tyrtw ₹73839.00

1HP ASMSP525 ₹9800.00

1HP ASMSP625 ₹10800.00

1HP ASMSP725 ₹12500.00

3HP HRF10A/30 ₹28000.00

3HP RF10A/30 ₹36000.00

7.5hp eswp77

7.5hp eswp78

7.5hp eswp89

Version Control :-

Using Git as a version control tool for AquaDB Management System is a strategic choice that offers numerous benefits. Git enables efficient management of the project by allowing to track changes, collaborate seamlessly, and maintain code stability.

In the context of this project, Git helps us to:

- Track Inventory and Employee Management Changes: Git tracks every change made to the project's source code, ensuring transparency in inventory management, employee profiles, and other critical components.
- Facilitate Collaboration: Team members can work concurrently on different aspects of our PROJECT, managing buying, renting, pre-ordering of products, employee profiles, payroll, and advance requests. Git enables collaboration by merging changes and resolving conflicts effectively.
- Maintain Data Integrity: With Git, you can ensure the integrity of our data, safeguarding inventory information, employee records, and store-related data.
- Version Control and Rollbacks: In the event of issues or unexpected changes, Git provides the capability to roll back to previous versions, preventing data loss and maintaining system stability.
- Efficient Bug Tracking: Git facilitates efficient bug tracking and resolution, helping administrators address issues in real-time and improve the overall performance of the project.
- Release Management: Git simplifies the management of releases, allowing you to tag specific versions and monitor changes made to our PROJECT.

- Documentation and Collaboration: Combined with documentation and issue tracking tools, Git ensures that our project remains well-documented and that tasks and bugs are properly tracked.

Development Tool :

Using Visual Studio Code (VS Code) and MySQL Workbench as development tools for our AquaDB Management system project is a powerful combination.

Visual Studio Code (VS Code):

- VS Code is an open-source code editor that offers a wide range of extensions and integrations.
- It provides a lightweight yet powerful environment for writing, debugging, and managing code.
- With VS Code, our team can efficiently work on the project, including writing code, collaborating, and ensuring code quality.

MySQL Workbench:

- MySQL Workbench is a comprehensive database design and management tool for MySQL databases, which is commonly used in web development.
- It allows us to design, model, and manage our database schema, tables, and data.
- MySQL Workbench helps us handle the storage and retrieval of critical data related to our project, such as inventory, employee profiles, and transaction records.

Bug Tracking Tool :

Using GitHub Issues for our warehouse management system (PROJECT) project is a practical choice:

- Seamless Integration: If the project's source code is hosted on GitHub, GitHub Issues provides a seamless way to manage bugs and issues alongside the codebase.
- Collaboration: GitHub Issues facilitates team collaboration, allowing team members to create, assign, and prioritize tasks, including bug fixes.
- Customizable Workflows: Can tailor our bug tracking workflow to match our project's needs, from issue creation to resolution.
- Visibility: The tool offers transparency by tracking the status of issues, which is helpful for stakeholders to monitor progress.
- Integration: GitHub Issues integrates well with other development tools and services, enhancing overall project management.

Testing tool:

Using Selenium and DBunit for ourAquaDB management system (PROJECT) project is a practical choice:

- Selenium: Selenium is an excellent choice for end-to-end testing. It can automate the testing of our web application, including interactions with the HTML, CSS, PHP, and SQL components.
- DBUnit: DBUnit is an extension of PHPUnit and is useful for testing database interactions in PHP applications. It allows you to set up a test database, populate it with data, and perform assertions against the database.

Deliverables :

Reuseable Components:

- HTML/CSS Templates: Reuse existing HTML/CSS templates as a starting point for building web pages. Justification: This saves time and effort in designing the user interface from scratch. Templates can be customized to match the project's branding and layout requirements.
- PHP Frameworks and Libraries: Reuse PHP frameworks and libraries for handling common functionalities like user authentication, database connections, and form validation.
- SQL Database Schema: Reuse a well-designed SQL database schema for inventory management, user profiles, and transaction records. Justification: A well-structured database schema saves time and ensures data integrity.

Build Components :

- Custom PHP Code: Build custom PHP code to create the logic and functionality specific to the WMS, such as user management, inventory tracking, and order processing. Justification: Custom code is required to address the unique business logic and requirements of the project.
- SQL Queries: Develop SQL queries to retrieve, update, and manage data within the database. Custom SQL queries are necessary to meet the project's specific data retrieval and manipulation needs.
- User Interface Components: Build user interface components using HTML, CSS, and JavaScript for creating web pages, forms, and interactive features. Custom user interface components are essential to match the project's unique design and user experience requirements.
- Testing Scripts: Create testing scripts using tools like Selenium or PHPUnit for testing the application's functionality, including HTML, CSS, PHP, and SQL components.

Custom testing scripts are needed to validate the project's specific functionality and integrations.

- Documentation: Develop project-specific documentation, technical documentation, and code comments. Custom documentation ensures that project can be understandable, maintainable and ensure efficient use.
- Bug Tracking and Issue Management: Implement a bug tracking and issue management system tailored to the project's needs. Custom issue management is necessary to track and resolve project-specific bugs and improvements.
- Integration Code: Develop custom integration code to connect with external systems or devices, such as barcode scanners or payment gateways. Custom integration code is required to meet the project's specific hardware and software integration needs.

Work Breakdown Structure (WBS) :

Project Planning and Initiation

- Define Project Scope and Objectives
- Stakeholder Identification and Engagement
- Project Kick off

Requirements Analysis and Design

- User Requirements Gathering
- Functional Requirements Specification
- Architecture Design
- Database Design
- User Interface Design

Development

- Front-End Development (HTML, CSS, JavaScript)
- Back-End Development (PHP, SQL)
- API Integration

Inventory Management

- Inventory Transactions
- Product Catalog Management
- Order Processing

Employee Management

- User Profiles
- Payroll Management
- Advance Request and Management

User Interface

- Dashboard
- Reporting

Security

- Access Control
- Data Encryption
- Security Audits and Compliance

Additional Features

- Barcode Scanning
- Mobile Access
- Integration with External Systems

Quality Assurance and Testing

- Test Planning
- Unit Testing
- Integration Testing
- User Acceptance Testing
- Performance Testing

Documentation

- User Manuals
- Technical Documentation

Project Management

- Project Progress Tracking
- Risk Management
- Change Control

Deployment

- Installation
- Configuration
- Data Migration

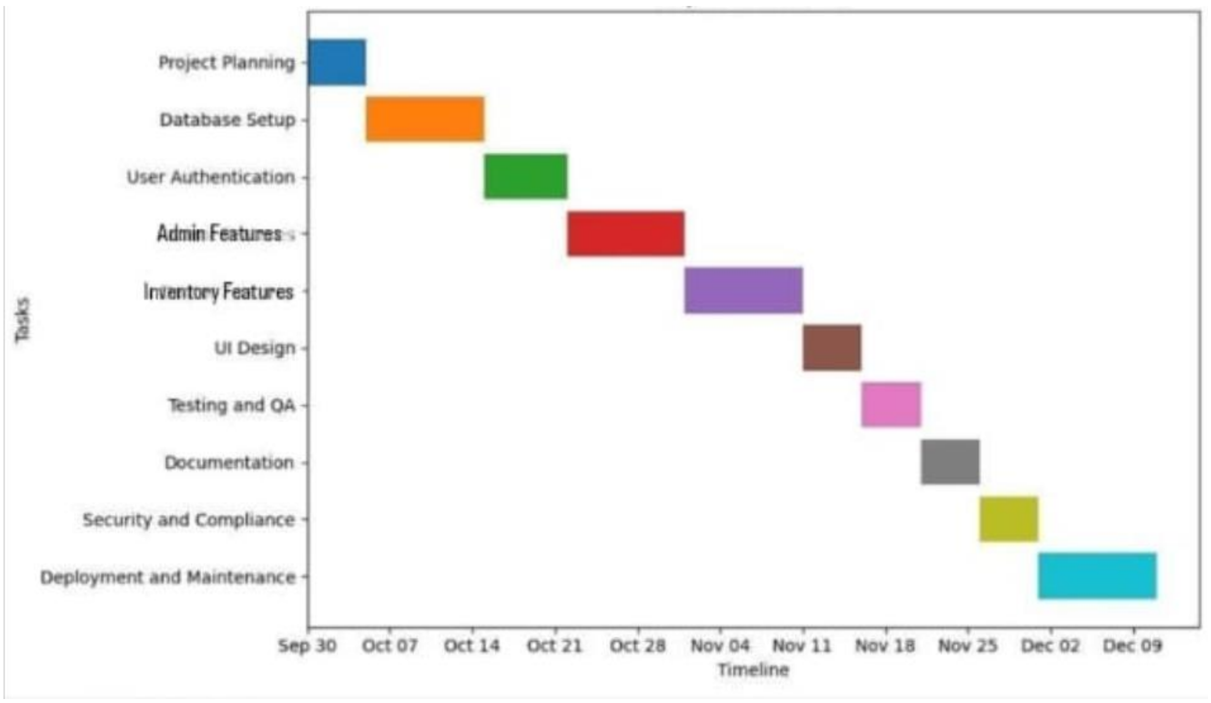
Closing and Handover

- Project Evaluation
- Handover to Operations
- Project Sign-off

Ongoing Support and Maintenance

- Bug Tracking and Resolution
- Updates and Enhancements
- Performance Monitoring
- User Support

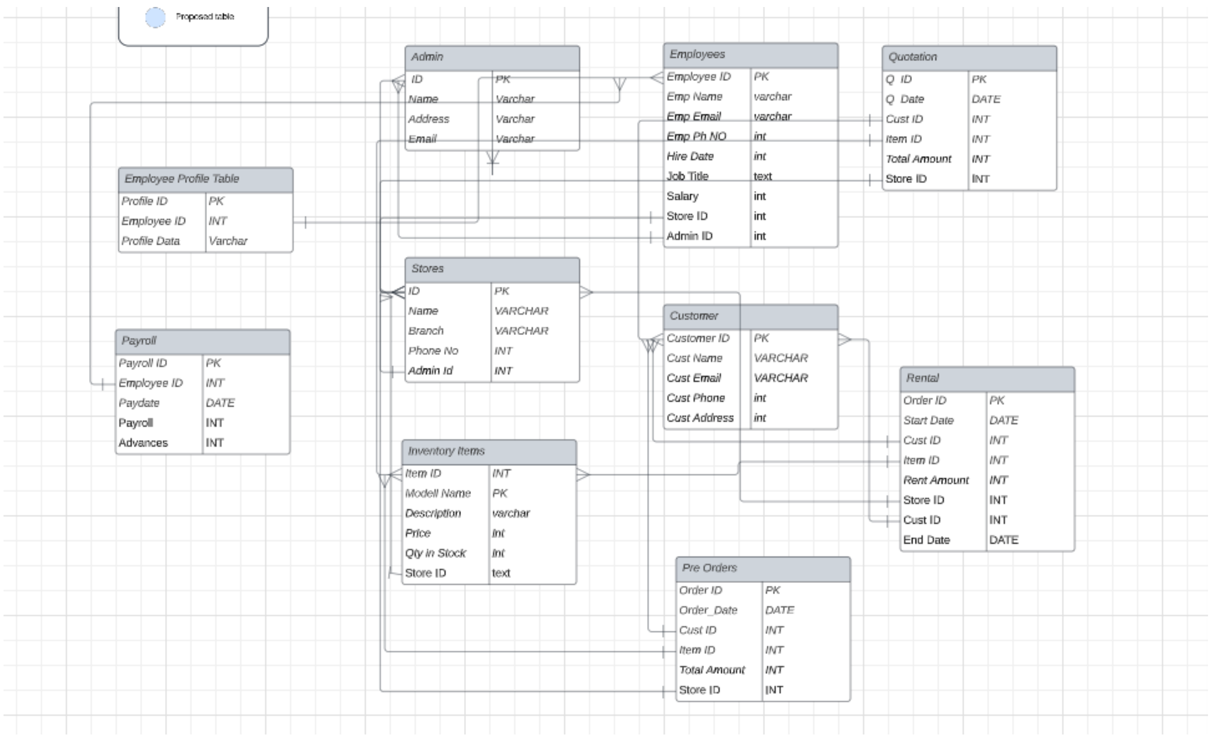
Rough Estimate in Man Days :

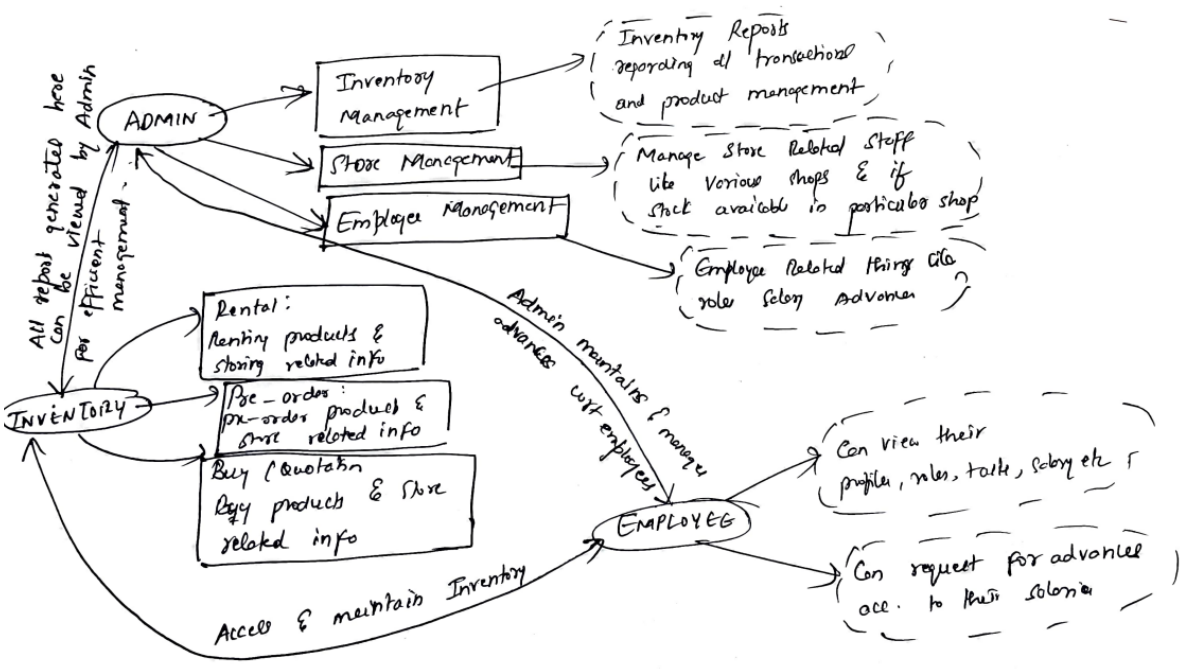


PES UNIVERSITY

BENGALURU

UML Diagrams and Project Structure :




ADMIN
Inventory Management
Inventory Reports reporting all transactions and product management
Store Management
Manage Store Related Staff like Various shops & if stock available in particular shop
Employee Management
Employee Related things like role salary Advances
All report generated here can be viewed by Admin for efficient management.
Rental: Renting products & storing related info
Pre-orders: pre-order product & store related info
Buy / Quotation Buy products & Store related info
INVENTORY
Admin maintains & manage advances wrt employees
EMPLOYEE
Can view their profiles, roles, task, salary etc
Can request for advances acc. to their salaries
Access & maintain Inventory

PES UNIVERSITY

BENGALURU