Phase II: User Requirements and Application Specifications

Submission Deadline: 26.03.2025, 23:59

1. Chosen Development Model:

The chosen development model is the **Incremental Model**, which is part of the Agile methodology.

Justification:

The Incremental Model is selected because it allows the system to be developed in small, manageable parts (increments). Each increment delivers a functional module, enabling early feedback and reducing risks. This approach is beneficial as it:

- **Provides flexibility** to accommodate changes in requirements.
- **Allows early testing and user feedback** to improve system quality.
- **Reduces risks** by breaking the project into smaller, more manageable stages.
- **Ensures faster delivery** of functional components rather than waiting for full completion.

2. User Requirements:

a. Stakeholders:

a) **Admin (Finance & Management)**

- **Role:** Manages financial records, oversees project budgets, and ensures smooth workflow.
- **Interest:** Wants an efficient system for tracking finances, assigning tasks, and managing worker performance.

b) **Manager/Team Leader**

- **Role:** Assigns tasks to workers, monitors progress, and ensures deadlines are met.
- **Interest:** A clear dashboard to assign, track, and update tasks for workers in real time.

c) **Workers (Construction Crew)**

- **Role:** Receive tasks from the manager and update progress. May also check and manage inventory.
- **Interest:** A simple and mobile-friendly interface to receive assignments, update status, and communicate with the manager.

d) **End-User (Client)**

- **Role:** Contacts workers in real time, tracks construction progress, and provides feedback.
- **Interest:** A real-time communication system and progress-tracking feature to stay updated on their project's development.

b. User Stories:

a) **User Story: Admin (Finance & Management)**

- **As an Admin**, I want to be able to view and manage the financial records and expenses for the construction project, so that I can ensure that the budget is being followed and any financial discrepancies can be caught early.
- **Benefit:** Ensures that the project remains within budget and provides transparency in financial tracking for the company and stakeholders.

b) **User Story: Manager/Team Leader**

- **As a Manager**, I want to assign tasks to workers and monitor their progress in real-time, so that I can ensure the project is on track and deadlines are met.
- **Benefit:** Allows the manager to stay organized and responsive to delays, ensuring smooth task delegation and project management.

c) **User Story: Worker (Construction Crew)**

- **As a Worker**, I want to receive task assignments from the manager, update my progress, and check the inventory for needed materials, so that I can efficiently complete my work and notify the team of any inventory shortages.
- **Benefit:** Streamlines communication, helps workers stay on task, and ensures that the project moves forward without material delays.

d) **User Story: End-User (Client/Homeowner)**

- **As an End-User**, I want to communicate with workers in real-time and track the progress of my construction project, so that I can provide feedback and stay updated on the status of my house being built.
- **Benefit:** Keeps the client informed, helps them feel involved in the process, and provides a direct line of communication to ensure satisfaction with the project.

e) **User Story: Inventory Manager (Optional)**

- **As an Inventory Manager**, I want to update and track the construction material inventory in real-time, so that I can prevent shortages and ensure that workers have what they need.
- **Benefit:** Helps keep the project running smoothly by ensuring the timely availability of materials, preventing work stoppages due to lack of resources.

3. Functional Requirements:

a. Brief Description:

The system should allow the manager to assign tasks to workers and track their progress. Workers will be able to update the status of their tasks, marking them as either completed or in progress. The admin will have the ability to view and manage all financial data related to the project, such as tracking expenses and monitoring the budget. Workers should also be able to check the inventory of construction materials, and in case of a shortage, the system will notify the manager or admin for necessary action.

In addition, the system should enable real-time communication between the workers and clients, allowing for constant updates on the construction project's progress. The client will have access to a user-friendly interface where they can track the overall status of the project. Workers and managers will receive notifications for new tasks, task updates, or any changes made to existing tasks.

Admins will have full control over tasks, finances, and user access levels. They can manage the entire system, while workers and clients will only have access to specific areas based on their roles. Finally, the admin will be able to generate various reports, such as financial reports and project progress updates, to keep all stakeholders informed.

b. Acceptance Criteria:

- **Task Assignment (Manager to Workers)**

    - The manager can create and assign tasks to workers.
    - Each task must include a description, deadline, and priority level.
    - Workers should be able to mark tasks as “in progress” or “completed.”
    - The system should update the task status in real-time.

- **Task Updates (Workers and Manager)**

    - Workers can update the task status (e.g., “completed” or “in progress”).
    - Manager can track the progress of each task in the system.
    - The system should notify the manager when a worker marks a task as completed.
    - The task details must be updated instantly and available to both workers and managers.

- **Financial Management (Admin)**

    - Admin can view and manage all financial records related to the project.
    - The system must automatically update the budget as expenses are entered.
    - The admin can generate reports of project costs and expenses.

- **Inventory Management (Workers)**

    - Workers can check and update the inventory of materials.
    - The admin should be able to view all inventory data at any time.
    - Inventory data should be updated in real-time.

- **Real-Time Communication (Client and Workers)**

    - Clients can communicate directly with workers via the system’s messaging platform.
    - Workers should have a clear view of client communication.
    - The system should allow both workers and clients to view project status updates in real-time.

- **Project Progress Tracking (Manager and Client)**

    - The client can view the overall status of the project at any time.
    - Managers can view individual task progress and overall project milestones.
    - The system should update the project’s progress in real-time as tasks are completed.
    - The system should generate a visual representation of the project’s progress (e.g., a timeline or Gantt chart).

4. Non-Functional Requirements:

a. Brief Description:

The system must be efficient in its performance, ensuring that tasks such as assigning duties, tracking task completion, managing finances, and communicating with the workers are carried out swiftly. Users should never experience delays; all actions should occur within a few seconds to ensure smooth operation. In addition to performance, the system's usability is key. It should be intuitive and easy to navigate for all types of users, from administrators and managers to workers and clients. The interface should be simple, with clear labels for buttons and sections, and support documentation should be readily available to assist users unfamiliar with the system.

b. Acceptance Criteria:

a) **Performance:**

- The system should respond to any user input, such as clicking a button or navigating between pages, within **2 seconds** or less.
- Task assignments, task completion updates, and financial updates should reflect changes in real-time, without requiring users to refresh the page.

b) **Usability:**

- The system interface should have an average of **90%** user satisfaction from usability testing (e.g., in terms of how easily users can find and complete key tasks).
- Navigation should be straightforward, with **clear labels** on all buttons and sections, ensuring that users can understand the purpose of each action without confusion.
- Support documentation should be accessible with one click from the main interface, and it should be easy to follow, ensuring that users can get help when they need it.
- At least **80%** of the users should be able to complete basic tasks without assistance after a brief tutorial.

5. Application Specifications:

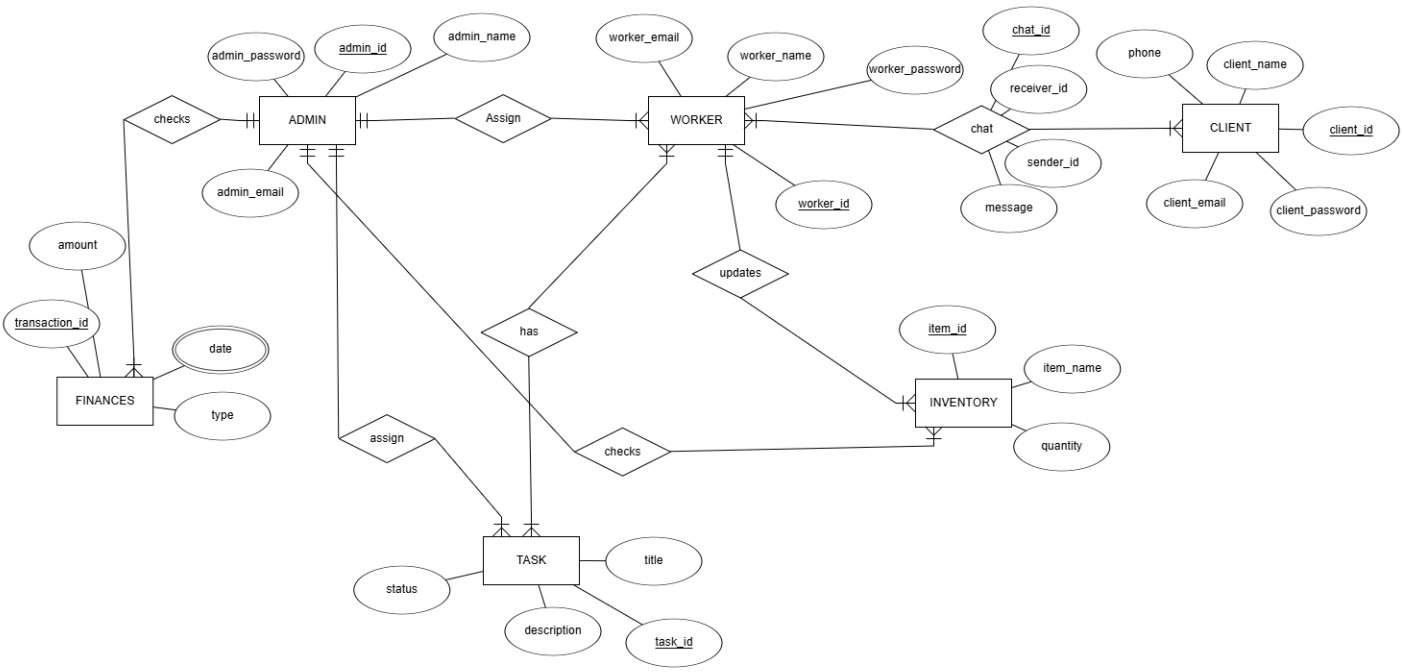
a. Architecture:

The system architecture follows a **Client-Server** model with **MVC (Model-View-Controller)** design for efficient management.

1. **Client Layer (Front-End)**: The user interacts with the system through a web interface. The front-end is built using HTML, CSS, and JavaScript, displaying data for admins, workers, managers, and clients.
2. **Server Layer (Back-End)**: The server handles the logic and business processes, built with PHP or Node.js. It implements the **MVC pattern**, where:
    - **Model**: Handles data and business logic (e.g., tasks, finances).
    - **View**: Displays data to users.
    - **Controller**: Handles user input and updates the model or view.
3. **Database Layer**: The system stores data in a relational database (PHPMYADMIN), including user data, tasks, finances, and inventory. Tables link data, such as users to tasks, and tasks to projects.
4. **(OPTIONAL) Communication & Security**: Communication between the client and server happens via RESTful APIs, secured with **SSL/TLS** encryption. Authentication is handled using **JWT** or session tokens.

b. Database Model:

.



c. Technologies Used:

Frontend Technologies:

- **HTML, CSS, JavaScript**: These are standard technologies for building the web interface. HTML structures the content, CSS styles it, and JavaScript makes the interface interactive. This combination is chosen for its broad support and ease of integration with other tools.
- **Bootstrap**: A front-end framework that helps quickly design responsive and mobile-friendly web pages. It ensures the interface looks good on all devices and saves development time.

2. **Backend Technologies**:

- **PHP**: PHP is a popular server-side scripting language chosen for its compatibility with web servers and ease of connecting to MySQL databases. It's well-suited for handling the backend logic of task management and communication with the database.
- **Node.js (Optional)**: If more scalability or real-time features are required, Node.js can be used due to its non-blocking, event-driven nature, making it suitable for handling multiple simultaneous connections.
- **MyCookiePHP**: This PHP framework will handle user authentication and session management. It will manage user login sessions, remembering users across sessions using cookies, and ensuring that users stay logged in securely.

3. **Database**:

- **phpMyAdmin**: A relational database management system that provides an easy-to-use interface for managing the database. It is used to store information about users, tasks, finances, and inventory. Its reliability, scalability, and simplicity make it ideal for handling complex queries and relationships between tables.

4. API

QuickBooks API: This API will be used to handle all financial operations, such as managing invoices, tracking expenses, processing payments, and generating financial reports. By integrating the QuickBooks API, the system can automate financial tasks, improve accuracy, and ensure real-time financial tracking, enhancing the overall project management experience.

d. User Interface Design:

For the **User Interface Design**, I will use **Figma** to create a prototype of the system. The prototype will include a main **webpage** where users can select their role before logging in as either an **Admin, Worker, or Client**. Each role will have a distinct interface:

- **Admin Interface**: Displays worker task assignments, finances, and inventory management. The admin can monitor project progress, manage expenses, and approve material requests.
- **Worker Interface**: Shows a task **to-do list** assigned by the admin, which workers can update in real-time as they complete tasks. Workers can also **request materials** when needed and **chat with clients** about project details.
- **Client Interface**: Allows clients to **view real-time updates** on work progress, access the chat with workers, and monitor project status, including completed and pending tasks. Additional features may include invoice tracking or feedback submission.

e. Security Measures:

The system will use **authentication** to ensure only authorized users can access their accounts, with passwords stored securely using hashing. **Encryption** will protect sensitive data like finances and messages. **Role-based access control** will make sure admins, workers, and clients only see what they need. **Cookies and sessions** will help keep users logged in safely, and security checks will prevent attacks like SQL injection. Regular **backups** will keep data safe in case of problems.