Requirements for Bid Tracking System

# Overview

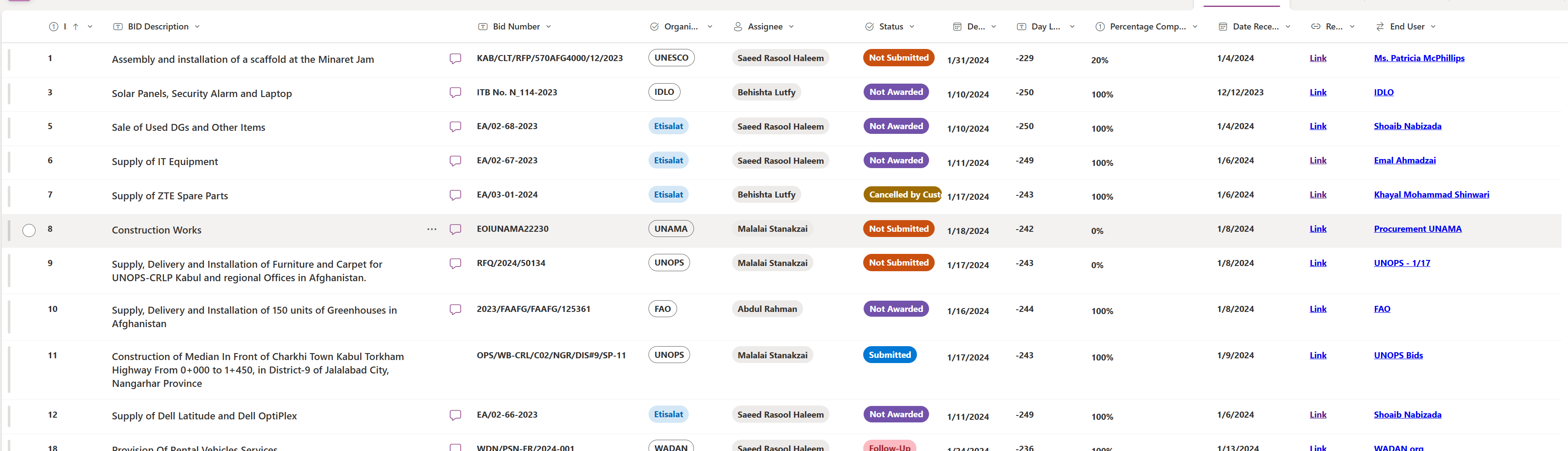
This document outlines the requirements for a system designed to track our bidding activities. The goal is to create a robust and efficient platform that allows us to monitor bids, assign tasks to team members, track customer interactions, and manage all related documents.

# Core Requirements

## User Management

* User Roles: The system should support multiple user roles such as Admin, Manager, and Team Member.
* Permissions: Each user role should have specific permissions. For example, only Admins can delete new bids, while Team Members can create, update and edit bids.
* User Profiles: Each user should have a profile that includes contact information, role, and assigned projects and tasks under each project. The team members are only able to see the bids which are only assigned to him/her but admin and manger should be able to see all the bids on a pipe line view based

We currently use this system. When clicking on a project or bid, a page should open displaying all relevant details, such as documents, team comments, tasks, and key information about the bid.



Please review the end user shown in the screenshot. This individual belongs to the organization where we submitted our bids. We need the system to display all previous bids submitted under this person's name and their completion statuses, so we can understand how many bids we have been awarded within the same organization.

## Bid Management

* **Create Bids:** The system must enable users to generate new bid records, including fields like *Title, Bid Number, Organization “Customer”, Assignee, Status, Deadline, Days Left, Percentage Complete, Date Received, Reference, and End User*.
* **Edit Bids:** Users should be able to edit bid details and update their status as needed. The system must support the following statuses, with the flexibility to modify them in the future: *"Not Submitted," "Not Awarded," "Cancelled by Customer," "Submitted," "Follow-Up," and "Awarded to Us."*
* Bid Dashboard: A dashboard to view all active and closed bids, with filters for sorting by date, client, and status.

## Task Assignment

* **Task Creation:** Users should be able to create tasks under each bid with details like task name, description, assignee, and due date. The team must be notified if the task status changes, someone comments on the task, the due date passes, or a subtask is created.
* **Task Assignment:** Tasks should be assignable to specific team members. Notifications should be sent to assignees upon task assignment.
* **Task Tracking:** The system should facilitate monitoring task progress and status updates (e.g., Not Completed, Completed). Each task must include only these fields: *Task Name, Assignee, Due Date, and Completed*.

Below are the tasks assigned to the staff. Everyone can comment on or add sub-tasks for other team members if necessary. Once a task is completed, it goes to the manager for approval to verify if it has been actually completed. Please check the screenshot below for your reference.

A screenshot of a phone

Description automatically generated

## Customer Tracking

* **Customer Profiles:** The system must maintain comprehensive profiles for each customer, which include their contact information and interaction history. It should also record the number of bids submitted to each customer and handle multiple contacts within the same organization. For instance, when submitting bids to Etisalat, different team members might be involved in each project; therefore, all contacts under Etisalat need to be documented. Their contact details, including phone number and email address, must be stored.
* Bid Submission Tracking: The system must keep track of the number of bids submitted to each customer and their statuses (Accepted, Rejected, Pending). If a bid is rejected, a loss comment must be mandatory for future reference.
* Customer Activity Reports: Generate reports summarizing bids submitted to each customer and their status.

## Document Management

* Upload Documents: Users should be able to upload documents related to each bid. Supported formats should include PDF, DOCX, XLSX, etc.
* Access Control: Permissions to view, edit, or delete documents should be based on user roles.
* There will be two types of documents: one that we receive from customer bid documents and another tab for the proposals we submit to the customer. All these must be stored on the bid.

# Additional Features

## Notifications and Alerts

* Email Notifications: Automated email notifications for task assignments, approaching deadlines, and bid status changes.
* We require a specific notification for bids that have been submitted and are still pending after 90 days. The system should automatically update the status of such bids to "Follow Up" and send reminders to the assigned team members and manager. This reminder will prompt us to contact the customer to verify whether the project has been awarded to another party, canceled, or if any further clarifications are needed.

## Reporting

* Custom Reports: Ability to generate custom reports based on various metrics such as bid success rates, team performance, and document updates.
* Export Options: Reports should be exportable in formats like PDF and Excel.

Companies

* The system should monitor the opportunities for each company separately, rather than collectively. For instance, we need to track the opportunities of companies like HPL and AB Group Inc individually.

Dashboard

* The Dashboard should show all activities related to our bidding opportunities, including the total submitted bids, the number of customers, the team members with the most submissions, project statuses (awarded or not), and the number of bids submitted this week, last week, last month, etc.

**Our current dashboard requires numerous updates.**

A screenshot of a computer

Description automatically generated

## Search and Filter

* Advanced Search: A powerful search function to find bids, tasks, documents, and customer information quickly.
* Filter Options: Filters to sort bids, tasks, documents, and customer profiles by various criteria like date, status, and assignee.

# Security and Compliance

* Data Encryption: All data should be encrypted in transit and at rest to ensure confidentiality.
* Access Logs: Maintain logs of user access and actions for audit purposes.
* Compliance: Ensure the system complies with industry standards and regulations for data privacy and security.

# Technical Requirements

* Scalability: The system should be able to scale to accommodate a growing number of users and bids.
* Integration: Ability to integrate with other tools such as email clients, CRM systems, and project management software.
* Performance: The system should be optimized for fast load times and efficient processing of large datasets.

**System Development Specific Requirements**

1. **Technology Stack:**
   1. Frontend: Use Metronic Vue.js template for frontend, which has TypeScript support and all the codes must be written with TypeScript and Vue.js Composition API <script setup lang=”ts”> coding style.
   2. Backend: Laravel Framework latest version with passport.
2. **Coding Standards and Practices:**
   1. PSR-12 for PHP/Laravel
   2. @typescript-eslint plugin for TypeScript along with Airbnb’s base rules for TypeScript and JavaScript.
   3. Implement best **automated code formatting** tools for Vue.js and Laravel.
   4. Insist on **proper documentation** for both front-end and back-end code.
3. **API Structure and Best Practices:**
   1. **RESTful principles** for API design.
   2. Use **Laravel Passport** for handling authentication/authorization, and implement **role-based access control** (RBAC) properly.
   3. **API versioning** for future-proofing.
   4. Define **standardized API responses** (e.g., JSON format, status codes, error messages).
4. **Security Considerations:**
   1. **input validation** both on the client (Vue.js) and server side (Laravel).
   2. Use **Eloquent ORM’s protection** against SQL injection (e.g., mass assignment rules).
   3. Insist on enforcing strong password rules.
   4. Ensure **rate limiting** and **throttling** to prevent API abuse.
5. **Version Control & Collaboration:**
   1. Use **Git** for version control.
   2. Define the process for **code reviews** and **merge requests.**
   3. Write detailed **commit history** and well-written commit messages.
6. **Database Design:**
   1. Ensure proper database design with **eloquent relationships**, indexing, and use of migrations and seeders.
   2. The **data integrity** should be handled properly and use **transactions** where needed.
   3. Optimize queries, especially for reporting and tracking.
7. **Deployment & DevOps:**
   1. **Dockerization** of the project for easy setup and consistent development environments.
   2. Define a **deployment strategy** for CI/CD.
   3. Provide a **rollback plan** in case of deployment failures.
8. **Project Documentation:**
   1. Document both the **code** (inline comments, markdown files) and **setup processes** (installation instructions, environment configurations).
   2. A well-documented **API reference** for future expansion or some third-party integrations.
9. **Other Recommendations:**
   1. **Implement CI/CD Pipeline:**
      1. CI/CD (Continuous Integration/Continuous Deployment) pipeline to help streamline code integration, testing, and deployment.
   2. **Multi-Tenancy Architecture**:
      1. Implement multi-tenancy to isolate data for each company (customer).
      2. Ensure that users from one company cannot access data from another company.
      3. Use **row-level isolation** strategies for implementing multi-tenancy.
   3. **User Roles & Permissions**:
      1. Define different roles (e.g., Admin, Manager, User, etc…) for each company.
      2. Allow the primary user (company admin) to manage sub-users (create, update, delete, block, etc..).
      3. Use Laravel’s authorization mechanisms like **Policies** and **Gates** to manage access control.
      4. Integrate role-based access control (RBAC) to easily manage permissions for various operations like assigning roles, projects, etc.
   4. **Company & User Management**:
      1. Each company should have its own dashboard to manage projects, roles, and users.
      2. Include features for a company admin to handle sub-user permissions and status (active/inactive).
      3. Super admin user, should be able to manage everything from companies, to users to projects and all the things which is included in the application.
   5. **Project Assignment**:
      1. Add functionality for assigning specific projects to sub-users within the same company.
      2. Allow role-based restrictions on what projects a sub-user can view or manage.
   6. **Audit Logs**:
      1. Implement audit logging for tracking actions like user creation, role changes, and data updates.
      2. Ensure that each company can view their own logs, company admin should be able to view all other sub users’ full log.
   7. **Scalability & Performance**:
      1. Ensure the system is scalable for a growing number of companies and users.
      2. Include options to add more users to each company over time.
   8. **Clean Architecture**:
      1. Implement clean architecture alongside repository pattern (default).