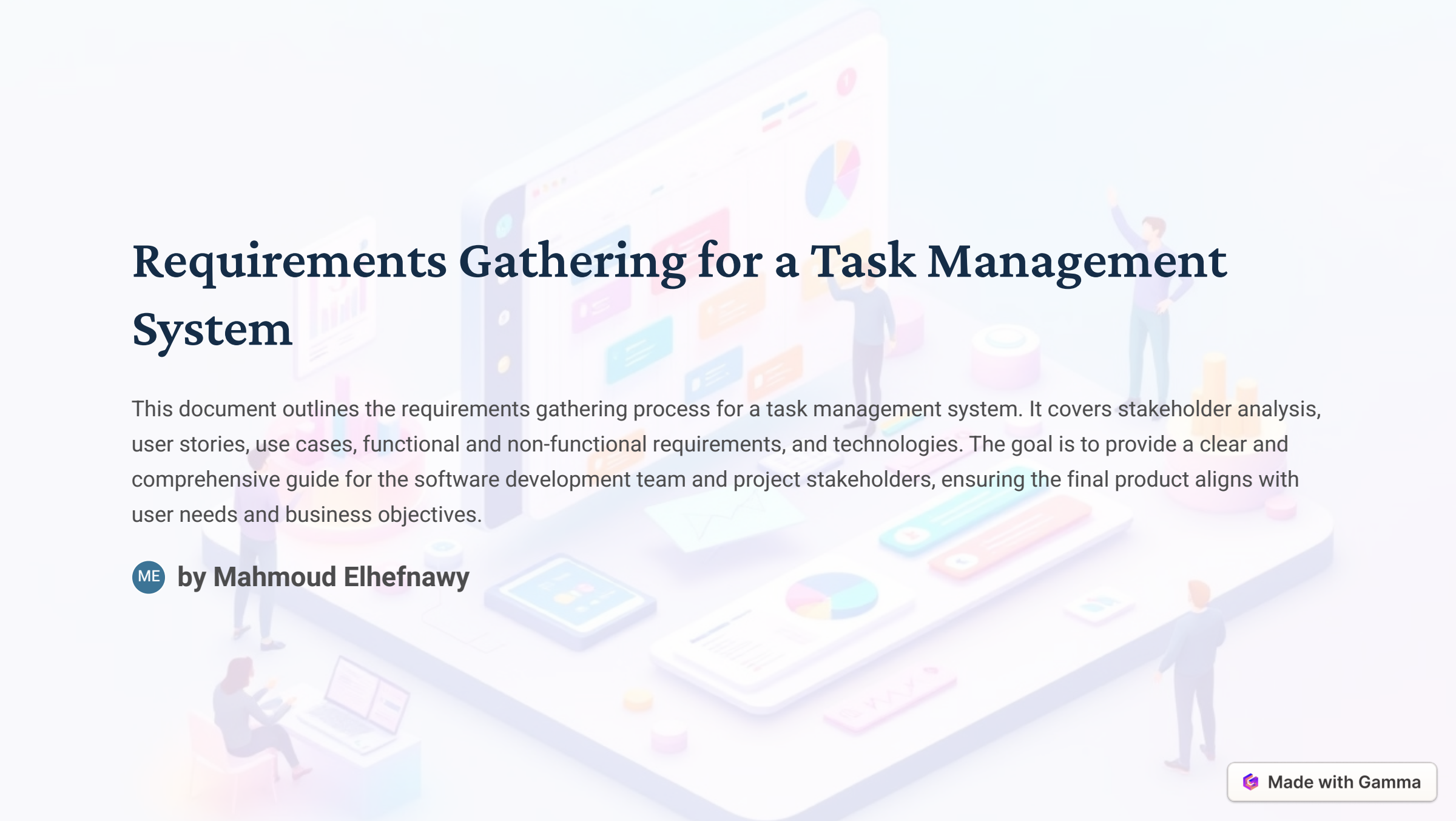
Requirements Gathering for a Task Management System



This document outlines the requirements gathering process for a task management system. It covers stakeholder analysis, user stories, use cases, functional and non-functional requirements, and technologies. The goal is to provide a clear and comprehensive guide for the software development team and project stakeholders, ensuring the final product aligns with user needs and business objectives.

ME **by Mahmoud Elhefnawy**

Stakeholder Analysis

Stakeholders are individuals or groups who have an interest in the system. Identifying their needs ensures the application aligns with their expectations and provides value to all parties involved.

Key Stakeholders

- End Users: Individuals who will use the application to create, update, and manage tasks.
- Project Managers: Those who oversee task progress and need visibility into task statuses.
- Developers: The team building the application, who need clear requirements to implement features.
- Business Owners: Those who want the application to improve productivity and task management.

Stakeholder Needs

- End Users: Easy-to-use interface, ability to categorize tasks, set priorities, and filter tasks.
- Project Managers: Ability to track task progress and generate reports.
- Developers: Clear functional and non-functional requirements to guide development.
- Business Owners: A reliable, scalable, and secure application.

Understanding the distinct needs of each stakeholder group is crucial for designing a system that is both effective and user-friendly. This analysis forms the foundation for subsequent stages of the requirements gathering process.

User Stories & Use Cases

User stories and use cases describe how users will interact with the system. They help define the system's behavior from the user's perspective, ensuring that the application meets their practical needs.

User Stories

1. As a user, I want to create a task with a title, description, due date, priority level, and category so that I can organize my work.
2. As a user, I want to update task details (e.g., status, priority, due date) so that I can keep my tasks up-to-date.
3. As a user, I want to delete tasks so that I can remove completed or unnecessary tasks.
4. As a user, I want to filter tasks by status (e.g., completed, in-progress) so that I can focus on specific tasks.
5. As a user, I want to search for tasks by title or category so that I can quickly find what I need.
6. As a user, I want to assign tasks to team members so that responsibilities are clear.

Use Cases

- Create Task: User fills out a form with task details (title, description, due date, priority, category) and submits it.
- Update Task: User selects a task, edits its details, and saves changes.
- Delete Task: User selects a task and confirms deletion.
- Filter Tasks: User selects a filter (e.g., "in-progress") and views only tasks matching that status.
- Search Tasks: User enters a search term and views tasks with matching titles or categories.
- Assign Task: User selects a task and assigns it to a team member.

Functional Requirements: Task Management Features

Functional requirements define the specific features and functionalities the system must have. These requirements are crucial for guiding the development team in building a system that meets the core needs of its users.

- Create, update, and delete tasks.
- Assign tasks to team members.
- Categorize tasks (e.g., work, personal, urgent).
- Set priority levels (e.g., low, medium, high).
- Add due dates to tasks.
- Filter tasks by status (e.g., completed, in-progress).
- Search tasks by title or category.
- Display a list of tasks with details (title, description, due date, priority, category, status).

Each of these features plays a critical role in enabling users to effectively manage their tasks and collaborate with team members. Proper implementation of these requirements is essential for the success of the task management system.

Functional Requirements: User Interface & Backend

User Interface Features

- Intuitive and responsive design.
- Forms for creating and updating tasks.
- Buttons for actions like delete, filter, and search.
- Task list with sorting and filtering options.

Backend Features

- RESTful API for CRUD operations (Create, Read, Update, Delete).
- Database to store task details (e.g., MongoDB).
- Authentication and authorization for secure access.

The user interface should be designed to be user-friendly, allowing users to easily interact with the system. The backend features should provide a robust and secure foundation for storing and managing task data, as well as providing secure access.

Non-Functional Requirements

Non-functional requirements define the system's quality attributes, such as performance, security, usability, reliability, and scalability. These requirements are essential for ensuring the system is not only functional but also meets the expectations of users and stakeholders in terms of its overall performance and quality.

Performance

The application should load tasks within 2 seconds. The API should handle up to 1,000 concurrent users.

Security

User data should be encrypted in transit (HTTPS). Implement authentication (e.g., JWT) to ensure only authorized users can access the system. Protect against common vulnerabilities like SQL injection and XSS.

Usability

The interface should be intuitive and easy to navigate. The application should be accessible on both desktop and mobile devices.

Technology Stack

The selection of appropriate technologies is crucial for building a task management system that meets the defined requirements. Here's how the technologies align with the requirements:

Frontend

- **React**: For building a dynamic and responsive user interface.
- **Redux**: For state management (e.g., managing task lists and filters).
- **HTML/CSS/JavaScript**: For structuring and styling the application.

Backend

- **Node.js/Express**: For building a RESTful API to handle CRUD operations.
- **MongoDB**: For storing task data in a NoSQL database.

API Integration

Frontend (React) communicates with the backend (Node.js/Express) via API calls.

Next Steps

Following the requirements gathering process, the next steps involve translating these requirements into a tangible plan for development.

1

Wireframes & Prototypes

Create mockups of the user interface to visualize the application.

2

Database Design

Define the schema for tasks (e.g., fields like title, description, due date, priority, category, status).

3

API Design

Define endpoints for CRUD operations (e.g.,

```
GET /tasks
```

,

```
POST /tasks
```

,

```
PUT /tasks/:id
```

,

```
DELETE /tasks/:id
```

).

4

Development

Implement the frontend and backend based on the requirements.

5

Testing

Test the application for functionality, performance, and security.

These steps will ensure that the development process is aligned with the defined requirements, leading to a successful implementation of the task management system.