Task and Project Management Developer Guide

VERSION 1.0

Priya Bhavana (2860256)
Lacholla Anuradha (2871807)
Srinivas(2845259)

Contents

Introduction and Background

Introduction

This application is a Task and project management tool designed to streamline tasks, projects, user interactions, and document management. The application will have various functionalities, allowing users to register, modify profiles, create, and track projects, manage tasks, and handle file attachments.

The purpose of this document is to provide a comprehensive understanding of the requirements and specifications for developing a web-based Task and Management Application. The scope of this application encompasses:

- Streamlining project management processes.
- Improving overall team productivity and communication.

1.1 Objectives

This guide outlines the objectives and requirements for developing a web-based Task and Management Application. The goal is to enhance project management and team collaboration, thereby boosting team productivity and communication.

Primary Objectives

Streamline Project Management: Facilitate efficient handling of various aspects of project management.

Enhance Team Collaboration: Implement features that improve communication and cooperation among team members.

Software Development Objectives

User Registration Module

Objective: Implement a user registration system.

Requirements:

Capture user details: First Name, Last Name, Email, UserID, and Password.

Ensure secure account creation and data handling.

Project Management Module

Objective: Develop a module for creating and managing projects.

Requirements:

Enable creation of projects with essential details: Name, Description, Start Date, and End Date.

Provide functionality for project archiving.

Task Management within Projects

Objective: Facilitate efficient task management.

Requirements:

Allow creation and tracking of tasks within projects.

Include features to set and monitor progress and deadlines.

Collaboration Tools

Objective: Develop tools for enhancing team communication.

Requirements:

Implement features like task comments and file attachments.

Ensure real-time collaboration is possible within the application.

1.2 Risks

1 Project Risks

Resource Availability

Description: Limited resources may lead to project delays or hinder feature implementation.

Mitigation: Prioritize tasks and allocate resources efficiently. Consider outsourcing specific components if necessary.

Technology Dependencies

Description: Dependency on third-party libraries or APIs could pose integration challenges or compatibility issues.

Mitigation: Carefully evaluate and test third-party components. Have contingency plans for alternative solutions.

User Adoption

Description: Ensuring user adoption of the new system may be challenging.

Mitigation: Conduct user surveys, involve end-users in the design process, and provide user training and onboarding.

Risk Name	Probability	Impact	RM3 Pointer
Resource Availability	High	High	12
Technology Dependencies	Moderate	High	9
User Adoption	Moderate	Moderate	6
Incomplete Task Creation	Moderate	High	6
Real-time Collaboration Failure	Moderate	High	6
Missed Deadlines	Low	High	4

2.0 Architecture

Software Architecture

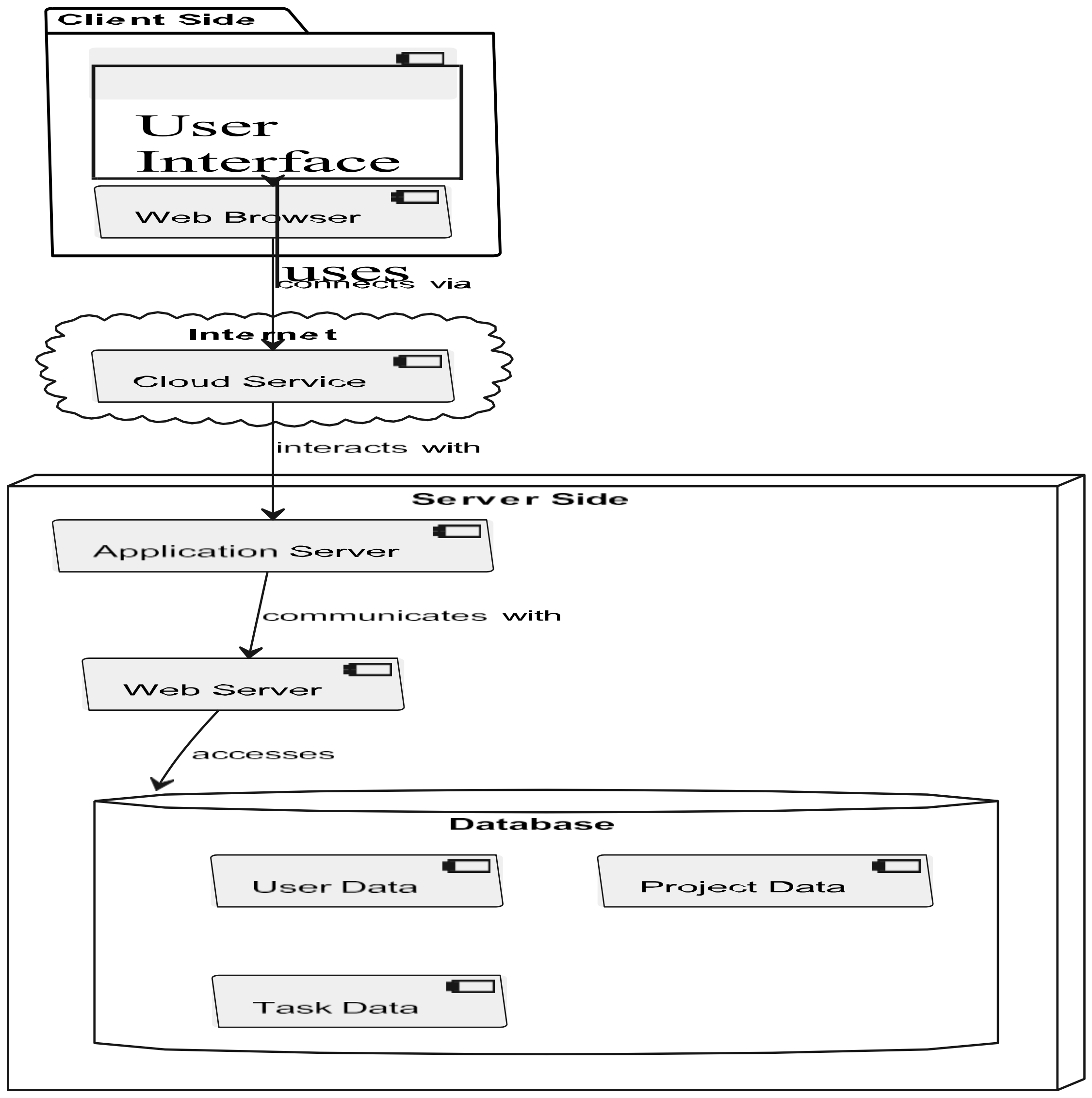
Communication: Users can create comments.

File Management: Allows users to upload attachments and subsequently update them.

Task Management: Users can create new tasks and track their progress or status.

Project Management: Facilitates the creation of new projects, monitors their progress, and offers archiving options.

User Management: New users can register, existing profiles can be edited, and accounts can be deleted if needed.



2.1 Database Design

USER Table

Purpose: Stores information about the users of the application.

Attributes:

username: Unique identifier for a user (Primary Key).

password: Password for user account security.

email: User's email address.

role: The role of the user within the system (e.g., admin, project manager, team member).

PROJECT Table

Purpose: Contains details of the projects managed within the application.

Attributes:

projectName: Unique name of the project (Primary Key).

projectDescription: Detailed description of what the project is about.

startDate: The date when the project is scheduled to start.

endDate: The date when the project is scheduled to end.

TASK Table

Purpose: Holds information on various tasks that are part of a project.

Attributes:

taskName: Unique name of the task (Primary Key).

taskDescription: Detailed description of the task.

dueDate: The deadline by which the task needs to be completed.

status: Current status of the task (e.g., pending, in progress, completed).

COMMENT Table

Purpose: For users to leave comments on tasks.

Attributes:

commentText: The actual text content of the comment.

commentDate: The date when the comment was made.

ATTACHMENT Table

Purpose: To store files attached to tasks.

Attributes:

fileName: The name of the file (Primary Key).

fileType: The type or extension of the file (e.g., .docx, .pdf).

filePath: The path where the file is stored.

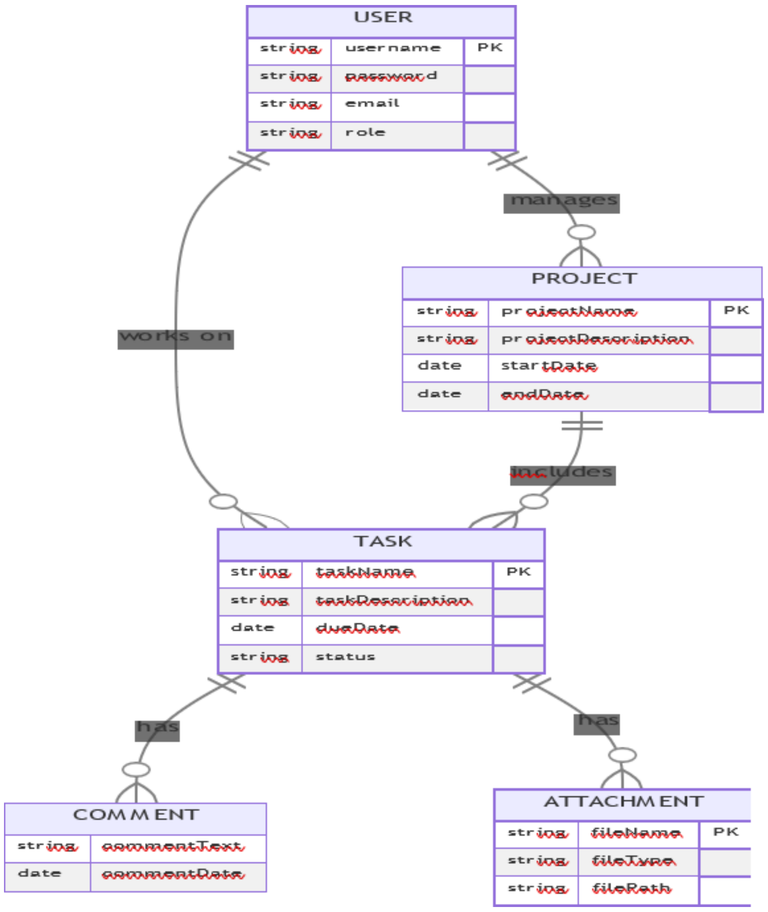
Relationships

USER to PROJECT: Indicates a many-to-many relationship, meaning a user can be associated with multiple projects, and a project can have multiple users.

PROJECT to TASK: A one-to-many relationship, where a project can have multiple tasks but a task is associated with only one project.

TASK to COMMENT: A one-to-many relationship, indicating that a task can have multiple comments.

TASK to ATTACHMENT: Also a one-to-many relationship, showing that a task can have several attachments, but each attachment is linked to a single task.



Following is a description of the tables in presented in the ERD as well as their relations:

Table Name	**Related To**	**Description**
USER	PROJECT, TASK	This table stores user account information, including usernames, passwords, emails, and roles.
PROJECT	USER, TASK	Contains details of projects, such as project names, descriptions, start and end dates.
TASK	PROJECT, USER, COMMENT, ATTACHMENT	Holds details on tasks including names, descriptions, due dates, and statuses, linked to projects and users.
COMMENT	TASK	For storing comments made by users on tasks, showing a one-to-many relationship with the TASK table.
ATTACHMENT	TASK	To store file attachments for tasks, with a one-to-many relationship indicating multiple attachments per task.

Following is a data dictionary:

Project

Name	**Data Type**	**Constraints**	**Description**
ProjectID	Integer	Unique, Mandatory	A unique identifier for each project.
ProjectName	String (max 256 chars)	Mandatory	The name or title of the project.
Description	String (max 1000 chars)	Optional	A brief description of the project.
StartDate	Date	Mandatory	The date when the project begins.
EndDate	Date	Optional	The anticipated completion date for the project.
Status	String	Mandatory, Choices: Ongoing, Completed, On Hold	The current status of the project.

Task

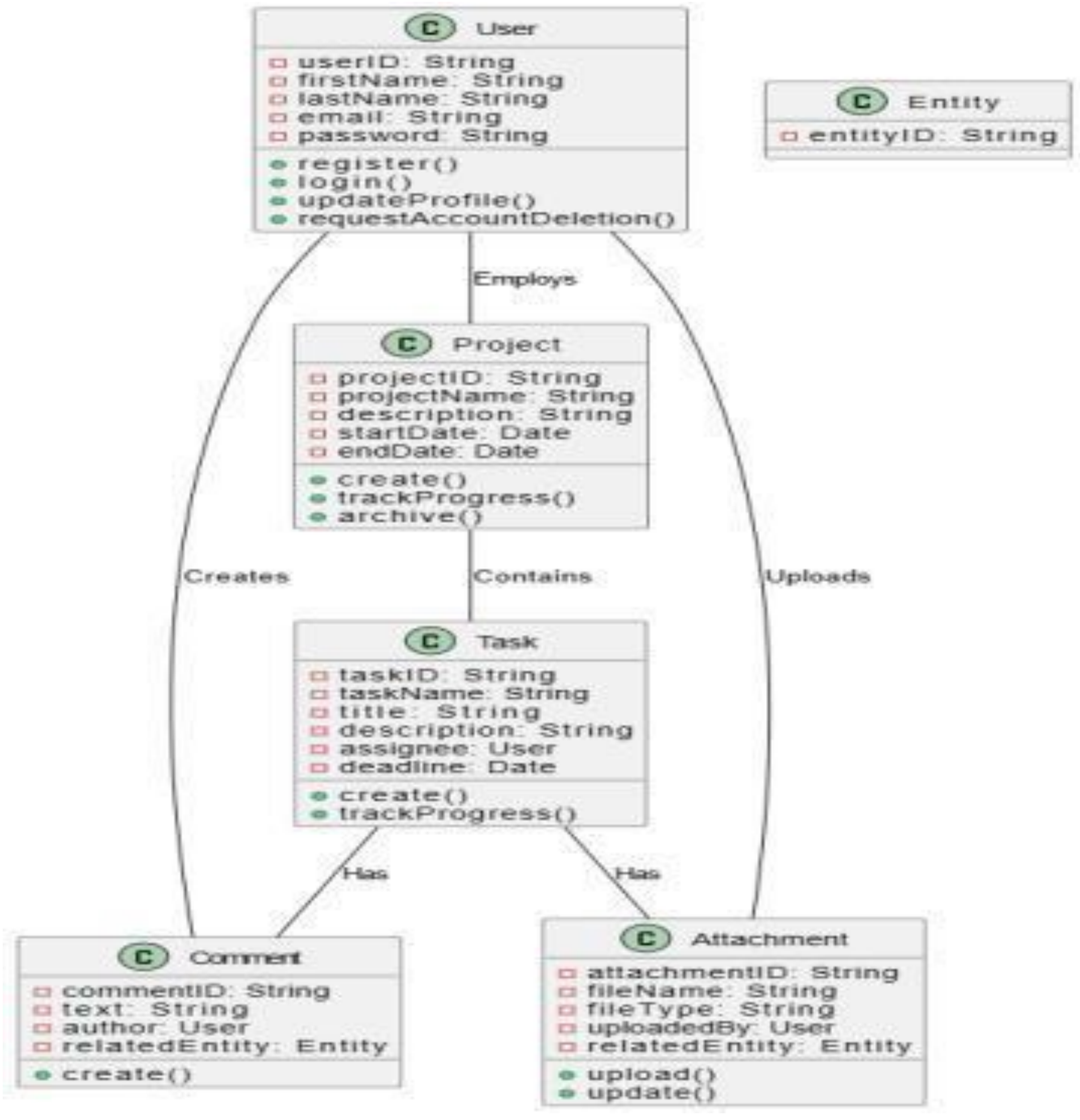
Name	Data Type	Constraints	Description
TaskID	Integer	Unique, Mandatory	A unique identifier for each task.
TaskName	String (max 256 chars)	Mandatory	The name or title of the task.
Description	String (max 1000 chars)	Optional	A brief description of what the task involves.
AssignedTo	Foreign Key (UserID)	Mandatory	The user to whom the task is assigned.
DueDate	Date	Optional	The expected completion date for the task.
Priority	String	Mandatory, Choices: High, Medium, Low	The importance or urgency of the task.
Status	String	Mandatory, Choices: To Do, In Progress, Completed	The current status of the task.
AssociatedProjectID	Foreign Key (ProjectID)	Mandatory	The project to which the task belongs.

User:

Name	Data Type	Constraints	Description
UserID	Integer	Unique, Mandatory	A unique identifier for each user.
FullName	String (max 256 chars)	Mandatory	The full name of the user.
Email	String	Unique, Mandatory	The email address of the user.
Role	String	Mandatory, Choices: Manager, Team Member, Team Lead	The role of the user within the office.

2.0 Class Level Design

The static structure is represented through class diagrams. For example, the User class includes attributes like username, password, and methods like login() and register(). The Project class includes attributes like projectId, projectName, and methods like createProject() and updateProject().



2.0.1 Classes

This folder contains classes related to data bearing classes as well as some miscellaneous classes.

2.0.1.1 User Class

Attributes:

userID (String)

firstName

(String)

lastName

(String) email

(String)

password

(String)

Operations:

register()

login()

updateProfile(

)

requestAccountDeletion

() Description:

This class represents a user in the system, with personal details and methods to handle account management.

2.0.1.2 Project Class

Attributes:

projectID (String)

projectName

(String) description

(String) startDate

(Date) endDate

(Date) Operations:

create()

trackProgress()

archive()

Description:

Represents a project with a unique ID, name, descriptive details, and a timeline. It has methods to create the project, track its progress, and archive it.

Relationship:

"Employs" indicates a one-to-many relationship with the User class, meaning a user can be employed on multiple projects.

2.0.1.3 Task Class

Attributes:

taskID (String)

taskName

(String) title

(String)

description

(String) assignee

(User) deadline

(Date)

Operations:

create()

trackProgress()

Description:

Represents tasks within a project. Each task has a unique ID, name, title, description, an assignee who is a User, and a deadline.

Relationship:

"Creates" and "Contains" suggest that a Project can create and contain multiple Tasks.

2.0.1.4 Comment Class

Attributes:

commentID

(String) text

(String)

author (User)

relatedEntity

(Entity)

Operations:

create()

Description:

Represents comments made by users. Each comment has a unique ID, the content of the comment, the author who is a User, and a related entity, which could be any object in the system.

Relationship:

"Has" indicates a Task can have multiple Comments.

2.0.1.5 Attachment Class

Attributes:

attachmentID

(String) fileName

(String) fileType

(String) fileContent

(String)

uploadedBy (User)

relatedEntity

(Entity) Operations:

upload()

update()

Description:

Represents files or documents attached to an entity in the system. Each attachment has a unique ID, file details, the user who uploaded it, and the entity it relates to.

Relationship:

"Uploads" and "Has" suggest that a User can upload multiple Attachments and a Task can have multiple Attachments.

2.1 Software Interfaces

The "Task and Project Management" application is designed with a robust architecture that leverages both modern and established technologies to ensure a responsive, intuitive user experience, and a reliable backend processing environment.

The software interfaces encompass various frameworks and technologies that serve different facets of the application, from user interaction to data management and version control.

Frontend Interface

The frontend interface of the application is built using HTML, CSS, and JavaScript. This combination provides a solid foundation for creating a dynamic and responsive user interface (UI).

HTML is utilized to structure the content of the application, while CSS is employed to handle the presentation, including layout, colors, and fonts. JavaScript adds interactivity to the frontend, enabling real-time updates without the need to reload the page, and allows for the creation of rich user interface components.

This stack ensures that the application is accessible across various devices and browsers, offering a consistent experience for all users.

Backend Interface

The backend is powered by Python using the Flask framework, which provides a lightweight and flexible platform for web application development.

Flask's minimalistic and modular design allows for the easy creation of scalable web services. It also facilitates the integration with the SQLite database, which is used for data persistence.

SQLite offers a lightweight, file-based database solution that eliminates the need for a separate server, making it an excellent choice for applications with moderate traffic and data storage requirements. This setup ensures that server-side processing is efficient and maintainable.

Version Control and Collaboration

Version control is managed through GitHub, a web-based platform that offers distributed version control and source code management functionality.

GitHub is instrumental in tracking changes in the application's codebase, enabling multiple developers to work on different features simultaneously without conflict. It provides tools for code review, project management, and integrates seamlessly with the deployment pipelines.

The use of GitHub facilitates collaborative development, ensuring that updates, bug fixes, and new features are integrated systematically and with full traceability.

Conclusion

As we conclude the development phase of the Task and Management Systems Project, it's important to reflect on the achievements and lessons learned throughout this journey. The project has successfully met its primary goals, creating a comprehensive platform that facilitates efficient project management, task assignment, and progress monitoring.

The system's adaptability across web and mobile applications provides a versatile solution catering to diverse project management needs.

3.0 Possible Future Improvements

Refining Real-Time Updates:

Enhancing the system's ability to provide immediate updates and notifications. This will improve the responsiveness and efficiency of task and project tracking.

Scalability Improvements:

Focusing on scaling the system to handle a larger number of users and projects without compromising performance. This includes optimizing database management and server capabilities.

Optimization of User Experience:

Continuous improvement of the user interface and interaction design to ensure a seamless and intuitive experience for all users. This may involve updating UI elements, improving navigation, and ensuring accessibility standards.

Third-Party Integrations:

Exploring opportunities to integrate with other tools and platforms. This could extend the system's functionality, such as adding advanced analytics, reporting features, or connecting with other productivity tools.