Project Overview

Project Title:

Task Reminder Application

Student Name:

Dakarai Mahisa

Project Type:

Automated Task Reminder & Tracking Application

Technology Stack:

Java 17, Spring Boot, Spring Security, Thymeleaf, MySQL, JPA/Hibernate, Maven, Postman, Git, GitHub

Overview

The Task Reminder Application is a Spring Boot‑based web application that enables users to create, manage, and track tasks effectively. It supports task lifecycle management including creation, updates, prioritization, status tracking, and due-date‑based reminders.

The project was developed using **Agile methodology**, with features implemented incrementally across multiple sprints, each focusing on specific functional and technical goals.

Key Features

- Task CRUD operations with status and priority
- Pagination, sorting, and filtering
- Overdue, upcoming, and due-today task categorization
- RESTful APIs for task management
- Secure user registration, login, and session control
- OTP-based email verification and encrypted passwords
- Reminder engine for near-due tasks
- User profile management and CSV export

Objective and Outcome

The objective of this project is to design a secure and maintainable task management system while gaining practical experience in enterprise Java development and Agile practices. The project successfully delivers a production-style application with structured APIs, secure authentication, and professional documentation suitable for internship evaluation.

Table of Contents

DOCUMENT 1 — 28/11/2025

Documentation

Task: Project requirements – Install IntelliJ IDE, Java 17, MySQL Database

Student Name(s): Dakarai Mahisa

Date Completed: 28-Nov-25

1. Objective:

- The objective of this task is to gather all required tools and software needed for starting the Task Reminder Application.
- To ensure the full development environment is properly installed and configured.

2. Steps Taken:

- Identified all project requirements for the Task Reminder Application.
- Downloaded and installed IntelliJ IDEA.
- Installed Java 17 for backend development.
- Installed MySQL database and confirmed it is running properly.

3. Challenges Encountered (if any): -N/A

4. Verification:

- Verified that IntelliJ IDEA launches successfully.
- Confirmed Java 17 installation through terminal version check.

DOCUMENT 2 — 01/12/2025

Documentation

Task: Set up Spring Boot Project and Build Skeleton

Student Name(s): Dakarai Mahisa

Date Completed: 01-Dec-25

1. Objective:

- The objective of this task is to set up the initial Spring Boot project for the Task Reminder Application.
- To configure Maven and add required dependencies for web and database operations.

2. Steps Taken:-

- Created a new Spring Boot project named *com.taskreminder.app* using start.spring.io.
- Selected Maven project structure.
- Added required dependencies including Spring Web, Thymeleaf, and MySQL Driver.
- Built the initial project scaffolding inside IntelliJ IDEA.
-

3. Challenges Encountered (if any): -N/A

4. Verification:

- Verified project builds successfully without errors in IntelliJ.
- Confirmed all dependencies load correctly in the pom.xml.
- Ensured directory structure follows standard Spring Boot conventions.

DOCUMENT 3 — 02/12/2025

Documentation

Task: Create HTML Templates and Core Packages

Student Name(s): Dakarai Mahisa

Date Completed: 02-Dec-25

1. Objective:

- The objective of this task is to define the essential components of the application including UI templates, controllers, entities, and services.
- To lay the foundation for CRUD operations.

2. Steps Taken:

- Added tasks.html file inside Resources -> templates.
- Created controller package containing:
  - ✓ HomeController
  - ✓ TaskController
- Created entity package containing Task class.
- Created service package containing TaskService class.

3. Challenges Encountered (if any): N/A

4. Verification: -

- Verified correct mapping between controllers and templates.
- Validated that the application starts and routes load without issues.
- Ensured all packages and class files are recognized by Spring Boot.

DOCUMENT 4 — 03/12/2025

Documentation

Task: Add Task Buttons, GitHub Integration, and UI Enhancements

Student Name(s): Dakarai Mahisa

Date Completed: 03-Dec-25

1. Objective:

- The objective of this task is to enhance UI functionality and facilitate CRUD operations via HTML pages.
- To push the project to GitHub and update UI styling for a better user experience.

2. Steps Taken:

- Added add-task.html file and integrated Add Task button.
- Added Delete Task functionality.
- Added edit-task.html page and implemented Edit Task button.
- Pushed project updates to GitHub repository.
- Improved the CSS styling of all existing HTML files to enhance UI.

3. Challenges Encountered (if any): N/A

4. Verification:

- Verified add, edit, and delete features load properly in the browser.
- Confirmed UI enhancements display correctly across all pages.
- Ensured GitHub repository contains all latest commits without conflicts.

DOCUMENT 5 — 04/12/2025

Documentation

Task: Backend Validation of Task Reminder App, Frontend Validation, and Debugging

Student Name(s): Dakarai Mahisa

Date Completed: 04-Dec-25

1. Objective:

- The objective of this task was to implement backend and frontend validation for the Task Reminder application to ensure that only valid data is processed.
  This included preventing empty or invalid form submissions, enabling user-friendly error messages on the UI, and conducting debugging activities to ensure proper app behavior during validation failures.

2. Steps Taken:

- Added the **Spring Boot Validation dependency** in pom.xml to enable backend validation support.

- Applied **@NotBlank** and other relevant validation annotations to fields in the **Task entity class**.
- Configured controllers to return validation errors back to the frontend so users remain on the same page until valid data is entered.
- Implemented **frontend validation** to ensure forms do not submit when required fields are empty.
- Added error-handling logic so that any backend validation failure is clearly displayed on the UI for the user.
- Performed step-by-step debugging to ensure both backend and frontend validations were functioning as expected.
- Ensured that confirmation prompts (e.g., Delete Task) remain functional and aligned with validation behavior.

3. Challenges Encountered (if any):

- Maven initially failed to load the validation dependency.
  The issue was resolved by restarting IntelliJ IDEA, after which the dependency loaded correctly.

4. Verification:

- Verified that the **Add Task** form does not submit until all required fields are entered.

- Confirmed that invalid input triggers clear validation messages and keeps the user on the same page.
- Ensured that backend validation errors are displayed properly on the UI.
- Tested the **Delete Task** feature, confirming that the browser displays the confirmation message “Are you sure you want to delete this task?” before removing the task.
- Confirmed that frontend validation and backend validation work together seamlessly to prevent invalid data entry.

DOCUMENT 6 — 05/12/2025

Documentation

Task: Setting up MySQL Database, Removing In-Memory Data, Connecting Task Reminder App with MySQL, Configuring Environment Variables

Student Name(s): Dakarai Mahisa

Date Completed: 05-Dec-25

1. Objective:

- The objective of this task is to replace the previously used in-memory data with a fully configured MySQL database.

- To establish a stable backend connection between the Task Reminder Application and MySQL.
- To configure environment variables to securely store database credentials and ensure reliable application startup.

2. Steps Taken:

- Installed and configured MySQL database for backend integration.
- Removed all in-memory hardcoded data from the Task Reminder App.
- Updated application configuration to connect the Spring Boot project with MySQL.
- Created required database schema and tables.
- Set up environment variables to securely manage database username, password, and connection details.
- Tested the database connection through the application to ensure proper CRUD functionality.

3. Challenges Encountered (if any):

- Faced **"Access was denied"** error while connecting to the MySQL database.
- Executed several commands in Command Prompt to diagnose authentication issues.
- Identified that incorrect credentials were being used and rectified them by verifying the correct MySQL username and password.
- A **white error page** appeared on the frontend due to backend connection failure.
- Resolved the issue by verifying the API configuration and updating the database connection properties.

4. Verification:

- Successfully connected the Task Reminder App to the MySQL database without further authentication errors.
- Verified data persistence by inserting and retrieving tasks from the database.
- Ensured that the frontend loads correctly without the white error screen.
- Confirmed stable API responses through browser testing and console logs.

DOCUMENT 7 — 08/12/2025

Documentation

Task: Creating Feature/Status-Change Branch, Implementing “Mark as Done” Button, and Raising Pull Request

Student Name(s): Dakarai Mahisa

Date Completed: 08-Dec-25

1. Objective:

- The objective of this task is to introduce a new feature that enables marking tasks as completed within the Task Reminder Application.
- To create a dedicated Git branch for isolating and managing the status-change feature development.
- To implement the frontend “Mark as Done” button and ensure its integration with the backend status update API.
- To follow proper version control practices by committing changes, pushing the feature branch to GitHub, and creating a pull request for review and merging.

2. Steps Taken:

- Created a new Git branch named **feature/status-change** to isolate development changes from the main branch.
- Developed and integrated a “Mark as Done” button into the frontend interface.
- Connected the button to the backend API endpoint responsible for updating the task status.
- Implemented logic to update the task entity’s status field (e.g., from PENDING to DONE) in the database.
- Tested the feature locally to ensure correct UI updates and backend status persistence.
- Committed all related changes and pushed the **feature/status-change** branch to the GitHub repository.
- Created a pull request from **feature/status-change** to the **main** branch for review.

3. Challenges Encountered (if any): NA

4. Verification:

- Verified that the “Mark as Done” button correctly triggers the status update API.
- Confirmed that the task status changes to “DONE” in the database and persists across sessions.

DOCUMENT 8 — 09/12/2025

Documentation

Task: Creating REST API Endpoints and Testing Them Through Postman Application

Student Name(s): Dakarai Mahisa

Date Completed: 09-Dec-25

1. Objective:

- The objective of this task is to design and build fully functional REST API endpoints required for the Task Reminder Application.
- To ensure that all API routes support CRUD operations for task management, including creation, retrieval, updating, and deletion.
- To validate each endpoint's request and response behavior using the Postman application to confirm API correctness, reliability, and error handling.
- To establish a consistent backend interface that the frontend can consume without failures or incompatibilities.

2. Steps Taken:

Defined necessary REST endpoints in the Spring Boot controller, including:

- POST /tasks – Create a new task
- GET /tasks – Fetch all tasks
- GET /tasks/{id} – Retrieve a specific task
- PUT /tasks/{id} – Update task details
- PATCH /tasks/{id}/status – Update task status
- DELETE /tasks/{id} – Remove a task

3. Challenges Encountered (if any): NA

4. Verification:

- Successfully executed all CRUD operations using Postman without any failures.
- Verified that data persisted correctly in the MySQL database after POST, PUT, PATCH, and DELETE calls.
- Ensured that error handling logic returned appropriate status codes and messages for invalid requests.

DOCUMENT 9 — 11/12/2025

Documentation

Task: Implementing Pagination, Sorting, and Filtering of Tasks

Student Name(s): Dakarai Mahisa

Date Completed: 11-Dec-25

1. Objective:

- The objective of this task is to enhance the Task Reminder Application by implementing pagination, sorting, and filtering functionalities.
- To enable efficient handling and display of large task datasets by dividing results into manageable pages.
- To provide users with flexible sorting options based on different task attributes such as due date, priority, and status.
- To implement filtering capabilities that allow users to refine task results based on specific criteria.

2. Steps Taken:

- Implemented pagination logic using Spring Boot to limit and navigate task records per page.
- Added sorting functionality to allow tasks to be ordered in ascending or descending order based on selected fields.
- Integrated filtering options for task attributes such as status, priority, and title.
- Updated controller methods to accept pagination, sorting, and filtering parameters through request queries.
- Modified the service-layer logic to process filtered task lists before applying sorting and pagination.
- Enhanced the frontend interface to support pagination controls, sorting dropdowns, and filter inputs.

3. Challenges Encountered (if any):

- Encountered a white error page when applying descending sorting to the task list.

- Identified that the issue was caused by using an immutable list created via toList(), which does not support in-place sorting.
- The application threw runtime errors when attempting to modify the immutable collection during sorting.

4. Solution Implemented:

- Resolved the issue by replacing the immutable list with a mutable list implementation.

- Converted the result set into a modifiable list (e.g., ArrayList) before applying the sorting algorithm.
- Successfully implemented the sorting logic using the mutable list, ensuring compatibility with both ascending and descending order.

5. Verification:

- Verified that pagination works correctly across all task pages without data inconsistency.
- Confirmed that sorting functions properly in both ascending and descending order for all supported fields.
- Tested filtering options individually and in combination to ensure accurate task retrieval.
- Ensured that the white error page no longer appears when applying descending sorting.

DOCUMENT 10 — 15/12/2025

Documentation

Task: Enhancing Task Management with Enums, UI Updates, and Styling

Student Name(s): Dakarai Mahisa

Date Completed: 15-Dec-25

1. Objective:

- The objective of this task is to improve code maintainability and user experience in the Task Reminder Application.
- To introduce enums for defining constant values for task status and priority.
- To enhance the task listing UI by adding a *Completed At* column.
- To improve the frontend appearance by integrating additional CSS styling.

2. Steps Taken:

- Created enum classes for task status and task priority to replace hardcoded values.
- Updated the Task entity to use enums for status and priority fields.
- Added a *Completed At* column in tasks.html to display task completion timestamps.
- Modified controller and service logic to handle enum-based values correctly.
- Enhanced the frontend with additional CSS for improved layout, readability, and visual consistency.

3. Challenges Encountered (if any): NA

5. Verification:

- Verified that task status and priority are correctly handled using enums.
- Confirmed that the *Completed At* column displays accurate data for completed tasks.
- Validated that the UI renders correctly with the new CSS changes applied.

DOCUMENT 11 — 20/12/2025

Documentation

Task: Adding JUnit Tests for Service Layer Methods

Student Name(s): Dakarai Mahisa

Date Completed: 20-Dec-25

1. Objective:

- The objective of this task is to ensure correctness and reliability of the service layer in the Task Reminder Application.
- To validate business logic implemented in service methods using unit tests.
- To improve code quality and reduce the risk of regressions during future changes.

2. Steps Taken:

- Added JUnit and Mockito dependencies to the project configuration.
- Created test classes corresponding to service layer components.
- Mocked repository dependencies to isolate service logic.
- Written unit tests for core service methods such as create, update, fetch, and delete tasks.
- Verified different scenarios including valid inputs and edge cases.

3. Challenges Encountered (if any):

- Properly configuring mocks for repository interactions.
- Handling optional and null-return scenarios in test cases.

4. Solution Implemented:

- Used Mockito annotations to mock repository dependencies.
- Defined clear test data and expectations for each service method.
- Ensured service logic is tested independently of the database layer.

5. Verification:

- Confirmed that all service layer tests execute successfully.
- Validated expected behavior for each tested service method.
- Ensured test coverage for critical business logic paths.
- Verified no impact on application runtime behavior.

DOCUMENT 12 — 23/12/2025

Documentation

Task: Implementing Overdue, Upcoming, and Due-Today Task APIs

Student Name(s): Dakarai Mahisa

Date Completed: 23-Dec-25

1. Objective:

- The objective of this task is to enhance task tracking by categorizing tasks based on due dates.
- To provide dedicated APIs for overdue, upcoming, and due-today tasks.
- To improve task visibility and prioritization for users.

2. Steps Taken:

- Defined service-layer methods to identify tasks based on due date conditions.
- Implemented REST APIs for overdue, upcoming, and due-today tasks.
- Updated repository queries or filtering logic to support date-based retrieval.
- Integrated the new APIs with existing task response structures.

3. Challenges Encountered (if any):

- Handling date comparisons accurately across different task states.
- Ensuring completed tasks are excluded from overdue and upcoming results.

4. Solution Implemented:

- Used date-based logic to compare task due dates with the current date.
- Filtered tasks by status before applying due date conditions.
- Ensured consistent API responses for all three task categories.

5. Verification:

- Verified that overdue tasks are correctly identified.
- Confirmed that due-today tasks are returned only for the current date.
- Validated that upcoming tasks include only future-dated tasks.
- Ensured APIs return accurate results without affecting existing endpoints.

DOCUMENT 13 — 03/01/2026

Documentation

Task: Implementing User Registration and OTP-Based Account Verification

Student Name(s): Dakarai Mahisa

Date Completed: 03-Jan-26

1. Objective:

- To enable new users to register using email and password.
- To ensure secure account activation through OTP-based email verification.
- To prevent unauthorized access by validating user identity during registration.

2. Steps Taken:

- Designed the user entity and registration data model.
- Implemented user registration functionality with email and encrypted password storage.
- Generated one-time passwords (OTP) with an expiry time for account verification.
- Integrated email service to send OTPs to registered email addresses.
- Developed API endpoints for OTP verification and account activation.

3. Challenges Encountered (if any):

- Ensuring secure password encryption and storage.
- Managing OTP expiry and preventing reuse of expired or invalid OTPs.
- Handling edge cases such as duplicate registrations and invalid verification attempts.

4. Solution Implemented:

- Used BCrypt for secure password hashing before persisting user credentials.
- Implemented time-based OTP validation with expiry checks.
- Activated user accounts only after successful OTP verification.
- Added validation and exception handling for registration and verification workflows.

5. Verification:

- Verified user registration flow using Postman and UI-based testing.
- Confirmed OTP email delivery and correctness of generated OTP values.
- Validated successful account activation upon correct OTP submission.

- Tested failure scenarios including invalid OTP, expired OTP, and duplicate email registration.
- Ensured appropriate HTTP status codes and response messages were returned.

DOCUMENT 14 — 09/01/2026

Documentation

Task: Implementing User Login and Authenticated Session Management

Student Name(s): Dakarai Mahisa

Date Completed: 09-Jan-26

1. Objective:

- To enable secure user login using validated credentials.
- To enforce authenticated session management for protected application access.

2. Steps Taken:

- Implemented login functionality with email and password validation.
- Configured Spring Security for session-based authentication.
- Restricted task-related APIs to authenticated users only.
- Implemented logout and session timeout handling.

3. Challenges Encountered (if any):

- Managing secure session handling and access restrictions.
- Preventing unauthorized access to protected endpoints.

4. Solution Implemented:

- Used Spring Security to manage authentication and session lifecycle.
- Applied access control rules to ensure users can access only authorized resources.

5. Verification:

- Verified login success and failure scenarios.
- Confirmed protected endpoints are inaccessible without authentication.
- Validated logout and session expiration behavior.

DOCUMENT 15 — 14/01/2026

Documentation

Task: Implementing Reminder Engine for Pending Tasks and Password Encryption

Student Name(s): Dakarai Mahisa

Date Completed: 14-Jan-26

1. Objective:

- To notify users about pending tasks that are nearing their due dates.
- To enhance application security through encrypted password storage.

2. Steps Taken:

- Designed reminder logic to identify near-due and overdue pending tasks.
- Implemented a reminder engine to trigger notifications for such tasks.
- Integrated BCrypt for secure password encryption.
- Updated authentication flow to validate encrypted passwords.

3. Challenges Encountered (if any):

- Defining accurate criteria for near-due task reminders.
- Ensuring encrypted passwords remained compatible with existing login logic.

4. Solution Implemented:

- Applied date-based checks to identify tasks approaching their due dates.
- Used BCrypt hashing to securely store and validate user passwords.

5. Verification:

- Verified reminder triggers for pending and near-due tasks.
- Confirmed passwords are stored in encrypted form.
- Tested login functionality to ensure successful authentication with encrypted credentials.

DOCUMENT 16— 22/01/2026

Documentation

Task: Implementing User Profile Management and CSV Export Functionality
Student Name(s): Dakarai Mahisa

Student Name(s): Dakarai Mahisa

Date Completed: 22-Jan-26

1. Objective:

- To allow users to create and manage their profile information.
- To enable exporting of task data into CSV format for external use.

2. Steps Taken:

- Designed the user profile data model and relationships.
- Implemented profile creation and update functionality.
- Defined CSV structure for task export.
- Developed functionality to generate and download task data as a CSV file.

3. Challenges Encountered (if any):

- Ensuring profile data consistency across user sessions.
- Formatting task data correctly for CSV compatibility.

4. Solution Implemented:

- Used a structured profile entity to manage user-specific information.
- Implemented standardized CSV generation logic for task exports.

5. Verification:

- Verified profile creation and update workflows.
- Confirmed correct CSV file generation and successful download.
- Validated exported data for accuracy and completeness.