**RevTaskManagement**

**(Web Application)**

Center of Excellence

12-September-2023

Version 1.0

Copyright ©: 2023, Revature.

# Table Of Contents

[Application Overview 2](#_Toc660958343)

[Core Functional Requirements 2](#_Toc1752385842)

[Standard Functional Scope 2](#_Toc1778011964)

[Definition of Done 3](#_Toc322412955)

[Competency wise scoping 3](#_Toc1771156511)

[Non-Functional Expectations 5](#_Toc1638327516)

# Application Overview

The primary goal of the RevTaskManagement project is to develop a robust and user-friendly task management application that empowers users to manage their tasks and projects effectively. The application aims to streamline task management, increase productivity, and provide a central hub for organizing and tracking work-related and personal tasks. Also, it can be used to assign tasks to others by defining different milestones to track their work progress.

# Core Functional Requirements

The Task Management Application will include the following key features and functionalities:

As a user, I should be able to

1. Create accounts and login securely.
2. View my Profile.
3. Update my Profile.
4. Create my task that can have titles, descriptions, due dates, priority levels, and labels/tags.
5. Edit my task.
6. View my task.
7. Delete my task.
8. Categorize my tasks.
9. Assign Task to others if required.
10. Define Milestones for the task.
11. View Milestones
12. Update milestones of my own defined task.
13. Delete milestones of my own defined task.
14. Update the milestones only if the task is assigned by others.
15. To organize tasks into lists or projects.
16. To set task priorities as High, Medium, and Low
17. Sort tasks by due date, priority, or custom order.
18. Filter task based on labels or categories.
19. Track the task status with respect to the defined milestone
20. Close the task once it is completed.
21. Receive email or notification for upcoming tasks and deadlines. - *Nice to have*

# Standard Functional Scope

Registered users should be able to log in, change the password and request for a forgotten password (will be sent to their registered email).

# Definition of Done

* Working application demonstration.
* Sharing the associates’ code repo for technical review with:
  + ERD diagram
  + Application architecture diagram

# Competency wise scoping

|  |  |  |
| --- | --- | --- |
| **Competency** | **Application Type** | **Expectations** |
| Java SQL REST | Console Based Application | **User Inputs:**   1. Ability to accept the user inputs from console 2. Providing recommended format in which the user to key in the inputs 3. Validate the user given inputs for format and convert to appropriate type for application usage.   **System outputs:**   1. Use formatted outputs for better readability and understanding. E.g., currency and date values should be formatted well. 2. Display the reports in the appropriate format such as tables etc.   **User Navigation:**   1. Provide a number-based menu items for the user to navigate for different use cases 2. Handling user selections and providing appropriate screen / feature to the user   **Validation and Error Handling:**   1. Validate the user inputs for its types and format. 2. Display functional related user messages (either for input/error/output) - no system error codes. 3. Handle the exceptions and errors gracefully.   **Logging:**   1. Ensure the application is using proper logging framework and methods. 2. Ensure the application’s log level is configured using configuration files so that it can be changed without changing the code. 3. Also ensure that the application logging is configured to output to the mentioned log file.   **Testing**:   1. Ensure sufficient test cases are written using appropriate testing frameworks. 2. Ensure the code coverage closed to 70%   **Coding Standard:**   1. Use the industry coding standards and conventions. 2. Modular based code development for better reusability. 3. Ensure proper usage of resource objects such as database connectivity objects to avoid resource leakages. 4. Ensure proper usage of design patterns and application layering (such as Business Service, DAO Layer etc.) wherever applicable. |
| Web Fundamentals (HTML, CSS and JS) | Web Navigational Prototype | **User Experience:**   1. Have an intuitive design for the user to work with the application without any training or guidance 2. Have clean & consistent UI, color theme and easy to use navigations 3. Use bootstrap framework for responsive pages 4. Have proper tab indexing for users to navigate between the fields without usage of mouse.   **User Inputs & outputs:**   1. Have appropriate HTML fields for the user inputs 2. Wherever possible use the client-side validations for the user input 3. Display the appropriate user info/error message with appropriate colors and icons   **Performance:**   1. Use compressed images / assets to increase the page performance 2. Use the application validated using the Chrome’s Lighthouse tool and improved based on the report   **Dataset:**   1. For any prepopulated data such as to render table rows use JSON file as the DataSource and use standard open-source library to read and render in the web pages.   **General standards:**   1. Ensure the w3 standards are implemented for better accessibility. E.g., Using alt attribute for image tag. 2. Ensure the SEO recommended meta tags are added. |
| Web Development with Angular | Enhanced Web Navigational Prototype using Angular | **Same expectation as like for the Web fundamental competency with the below additions**  **Framework Specific**   1. Ensure the appropriate APIs are used for any of the API calls 2. Ensure the routing is centrally configured 3. Best practices & design patterns are to be followed 4. Implement the end-to-end testing framework and get to know the headless execution of end-to-end framework.   **Deployment artifacts:**   1. The deployment artifacts should be minified and obfuscated if required.   **Security**:   1. Ensure Route Guarding/Authenticated Routing is implemented. 2. Ensure that the secrets are stored as environment variables using secure credential storage. |
| Spring Boot | Identifying and Developing All APIs for any front-end application to consume | **REST Standards:**   1. Ensure the REST standards are followed for API naming, HTTP Operation and Response (output definition) 2. Secure the protected APIs 3. Define a common URL pattern for public and secure APIs 4. Proper documentation of APIs with Input and Output Samples to be documented   **Modularity:**   1. If the application is enhanced from an existing console-based project, try to reuse the existing modules. 2. Refactor the existing code based on the current tech stacks   **Logging:**   1. Ensure the application is using proper logging framework and methods. 2. Ensure the application’s log level is configured using configuration files so that it can be changed without changing the code. 3. Also ensure that the application logging is configured to output to the mentioned log file. 4. Ensure the centralized logging implementation   **Testing**:   1. Ensure sufficient test cases are written using appropriate testing frameworks. 2. Ensure the code coverage closed to be 70%   **Security**:   1. Ensure that the secrets are stored as environment variables using configuration files or secure credential storage. |
| Devops Azure GHA | Implementing DevOps Practices and CI/CD Pipelines for Cloud-Based Applications | 1. Ensure to implement DevOps best practices, emphasizing collaboration, automation, and continuous improvement throughout the software development and deployment lifecycle. 2. Leverage cloud services promptly to meet specific application hosting and deployment needs. 3. Ensure consistent and repeatable resource deployment by applying Infrastructure as Code (IaC) principles to define and provision cloud resources programmatically. 4. Implement cloud-specific security measures to protect applications and data. 5. Create and manage CI/CD pipelines using GitHub Actions to automate building, testing, and deploying applications. 6. Implement pipelines in either scripted or declarative form to streamline the software delivery process. |

# Non-Functional Expectations

* Application development is supposed to follow the Scrum process.
* The codebase should be hosted in a remote repository like GitHub.
* Application password should be encrypted using appropriate hashing algorithms.
* Applications should use the recommended authentication token such as JWT or another equivalent.