Universidad de las Fuerzas Armadas "ESPE"

CLASS NAME: OBJECT ORIENTED PROGRAMMING
TEACHER: EDISON LASCANO
NRC: 14575 *HOMEWORK: #6*
TOPIC: Object List, Diagram

Coding Ninjas

MEMBERS: - Lopez David
-Revilla Antonio
-Paspuel Oliver
- Ñato Carlos

DailyDev

Index

I INTRODUCTION

1. Introduction

This Software Requirements Specification for the development of a Java-based application has been prepared with Asana in mind. Its structure is made based on the standard IEEE *Recommended Practice for Software Requirements Specification ANSI/IEEE 830- 1998*.

The purpose of the Software Requirements Specification (SRS) document is to provide detailed and comprehensive guidance for the development implementation and understanding of Java-based work management application. This document clearly establishes the objectives, functions, and functional and nonfunctional objectives, functions and requirements of the software.

This document is intended for a diverse audience.

1.1. Purpose

The main purpose of our project is to develop a comprehensive task management system that allows users and teams to efficiently organize, monitor, and optimize their tasks and projects. Inspired by renowned platforms like Asana, our system will provide a transparent and detailed view of tasks, enabling users to make informed decisions, set priorities, and enhance their productivity. Our project aims to offer an adaptive solution in the realm of task management, prioritizing efficiency, transparency, and collaboration, with the goal of increasing productivity and improving organization across individual, team, and organizational environments.

1.2. System Scope

The project consists of developing a Java-based application that serves as a work management tool similar to Asana. The application includes features for managing work tasks, creating notifications, and integrating daily messages with communication platforms like Slack, the system to be developed is called Daily Dev.

The application will have the ability to track and categorize various types of tasks, provide timely reminders for upcoming tasks, and generate automated daily messages to keep user informed about their daily schedule, Additionally, to enhance collaboration and communication, the system must seamlessly integrate with Slack The system will also include a dashboard with various relevant services, such as a task calendar, a goal view to review your performance, reports with statistics and graphs, etc. In the future, there will also be an organizational mode where the organization can create projects with many users connected to the same project with many users connected to the same project with different roles, such as task manager, developer, organization leader, etc.

The benefits of the system are to enable users to have a comprehensive view of their task, make informed decisions, prioritize tasks effectively, and optimize their workflow to increase productivity.

The application aims to provide a comprehensive solution for teams to manage tasks efficiently, display notifications for important events, and also facilitate daily messaging to improve team communication.

1.3. Definitions, acronyms and abbreviations

(SRS) Software Requirements Specification

Asana.- is a team work management platform that allows users to organize tasks, projects, and collaborate efficiently, facilitating project coordination and tracking in real time.

Slack.- is a business communication platform that simplifies collaboration and information sharing among teams, offering instant messaging channels, file sharing, and integration tools to enhance team productivity.

1.4 References

Amaya Mosquera, P. A. (2019). Implementation of tools for operations (check in–check out) platform Asana.

Mota, A. (2022, diciembre 16). Cómo Realizar una Buena Especificación de Requisitos de Software (SRS). Innevo.com. https://blog.innevo.com/especificacion-requisitos-software

Especificación de requisitos de software (SRS): consejos y plantillas. (2019, octubre 8). Visure Solutions. https://visuresolutions.com/es/plantilla-de-consejos-de-srs-de-especificaci%C3%B3n-de-requisitos-de-software/

1.5. Document Overview

The document will include the following contents:

Overall Description .- This section details all the factors influencing the product and its requirements, It provides a description of the context of the requirements.

Specific Requirements .- This section contains requirements at a level of detail sufficient to enable designers to create a system that meets the requirements. It also allows the testing team to plan and conduct tests that demonstrate whether the system meets these requirements or not.

Appendices .- These can contain all kinds of information relevant to the SRS.

2. Overall Description

"Daily Dev" will be developed as a stand-alone application in Java that will communicate with users through an intuitive and easy-to-use graphical interface (GUI). This interface will provide users with a friendly visual experience, allowing them to interact effectively with the various functions and features that the application has to offer.

2.1 Product Perspective

Ths application stands out for its flexibility when integrating with external communication platforms, allowing daily messaging functions.

At this initial stage of development, our focus is on creating an application that is not only functional but also effective in meeting the essential needs of the users. Integration with external platforms adds a strategic component, allowing fluid and efficient daily communication. This interplay between task management and team collaboration is seen as a key element in improving overall productivity and effectiveness in the workplace.

2.2 Products Functions

The core functions of "Daily Dev" span from detailed task management to improving team communication and collaboration. these functions include:

Task Management:

- Allows users to efficiently create, assign, and monitor tasks.
- Tracks the progress of tasks and allows for the setting of deadlines.

Notifications and Reminders:

- Empowers users to generate personalized notifications and reminders.
- Keeps users informed about crucial events and approaching deadlines.

Integration with Slack:

- Facilitates seamless collaboration by integrating with the Slack communication platform.
- Enables smooth and effective daily communication within teams.

Dashboard:

- Provides a centralized dashboard with comprehensive services.
- Include a task calendar for scheduling, a goal view to assess performance, and report generation for relevant statics.

- Enhances the user's ability to gain insights into their overall performance and task-related statistics.

Future Organizational Mode:

- Planning for a future update includes an organizational mode.
- Enable organizations to create projects with multiple users connected to the same project, each with distinct roles such as task manager, developer, and organization leader.
- Enhances collaboration on a broader scale within the organizational structure.

These functions collectively contribute to the development of a robust and versatile application, addressing various aspects of task management, communication, and organizational collaboration.

2.3 User Characteristics

“Daily Dev”is designed to cater to the needs of various roles within a team or organization. User characteristics include:

Individual Users:

- Aimed at those who wish to manage their own tasks to enhance personal efficiency.
- Provides an intuitive interface and personalized tools for a user-centric experience focused on individual effectiveness.

Team Collaboration:

- Offers essential collaborative tools for teams needing to manage joint tasks.
- Facilitates effective communication and enhances coordination among team members.

Organization Leaders:

- Provides a holistic view of team performance, enabling organizational leaders to assess overall productivity.
- Offers data-driven tools for strategic decision-making and long-term planning.

These user characteristics are designed with the intent to ensure that “Daily Dev” is inclusive and capable of meeting the specific needs of a variety of users within a team and organizational environment.

2.4 Constraints

The development and operative of “ Daily Dev” will be influenced by some significant constraints:

Java Platform:

- The application will be developed in Java, limiting its implementation to environments compatible with this technology.

Integration with Slack:

- Integration with Slack is a key constraint, as daily communication relies on the functionality of this external platform.

User Roles:

- The functionality of user roles within the system is currently limited to basic tasks, with future expansions planned in subsequent developments.

These constraints influence the scope and specific capabilities of “ Daily Dev” and will be taken into account during the development and implementation of the system.

2.5 Assumptions and Dependencies

- The generation of daily messages depends on the user´s approval at the end of the workday, which implies the user´s availability and willingness to confirm the daily generation.
- It is assumed that users will actively participate in the process, committing to recording their work and providing approval for the daily generation.

2.6 Future Requirements

- Implementing analysis and statistics functions that allow users to assess their productivity over time. This could include metrics such as time dedicated to specific tasks, efficiency in deadline management, etc.

- Including support for multiple languages, which would expand accessibility and the global adoption of the application.

3. Specific Requirements

3.1 Functional requirements

3.1.1. Task management

- User task creation

  User can create task with titles, descriptions, and deadlines

- Task Status

  Tasks can be marked as completed or in progress

3.1.2. Notification creation

- Customizable Notifications

  Users can create notifications with customizable messages

- Association with tasks

  Notifications can be associated with specific task or events

3.1.3. Daily Messages

- Slack Integration

  Integration with Slack API for sending daily messages

- Configuration Options

  Configuration options for selecting message content and recipients

3.2. Non functional Requirements

3.2.1. Performance

- Responsive UI

  The application should provide a responsive user interface

- Timely Delivery

  Notifications and messages should be delivered in a timely manner.

3.2.2. Security

- User Authentication and Authorization

  User authentication and authorization mechanism to ensure data privacy.

- Secure Communication

  Secure communication with external platforms.

3.3. Performance Requirements

- Response Time

  The application should have fast response times to ensure a smooth user experience.

- Resource Consumption

  Maintain efficient resource consumption, such as memory usage and processing capacity, to ensure that the application is lightweight and does not overload users' devices.

- Real-Time Updates

  Ensure the efficient delivery of real-time updates, especially for collaborative functions such as daily meetings and notifications.

3.4 Design Constraints

- Integration with External Tools

  The design should consider the limitations or specific requirements of external tools it integrates with, such as GitHub, Asana, and Slack.

- User Interface

  The user interface should be designed intuitively and be user-friendly, especially considering the possibility of users with various levels of technical skill.

3.5 System Attributes

- Security

  Data security must be a priority, with robust measures to protect user information and ensure compliance with privacy regulations.

- Efficiency

  The application should be efficient in terms of response time and resource consumption to provide a fast and smooth user experience.

- Analytics and Reports

  It must have analytical capabilities to collect data on the usage of the application and generate reports that allow users to assess their productivity and performance.

4. Appendices.

Interview Video Link: https://www.youtube.com/watch?v=9yyJ8qHsiTI&ab_channel=olivier
Interview Screenshot:

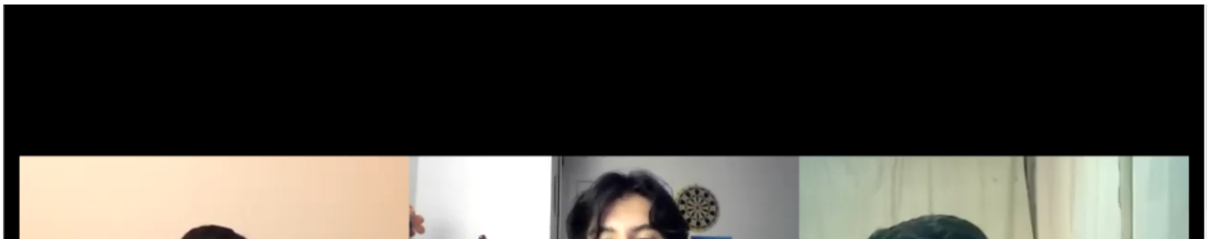
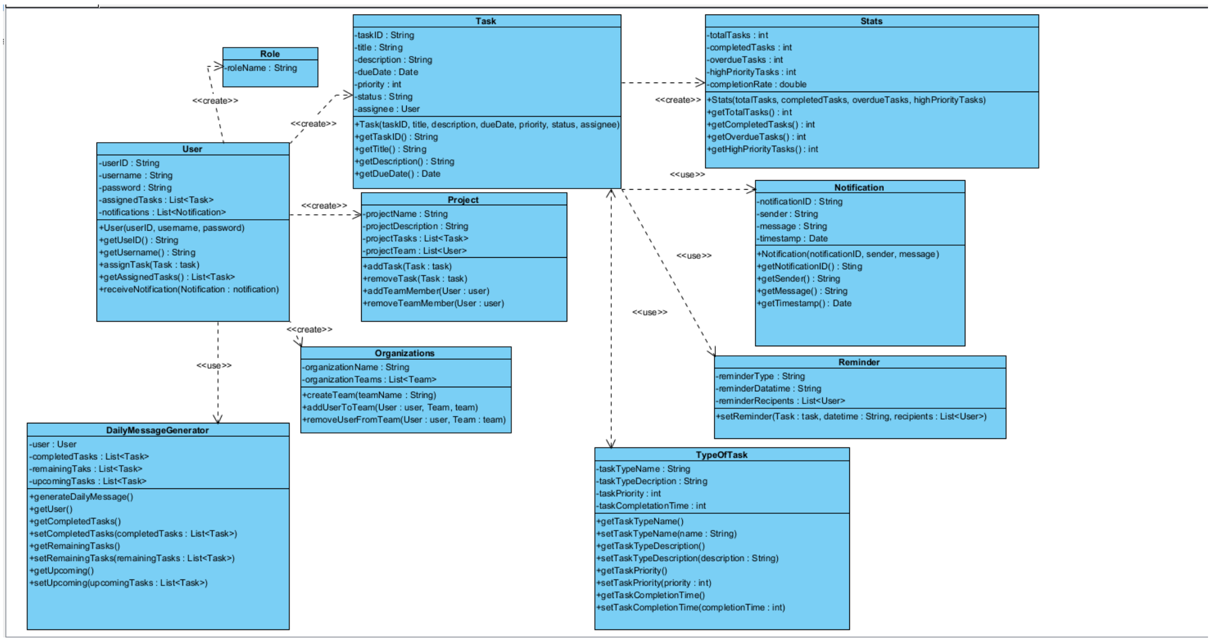


Diagram Screenshot



Task
-taskID : String
-title : String
-description : String
-dueDate : Date
-priority : int
-status : String
-assignee : User
+Task(taskID, title, description, dueDate, priority, status, assignee)
+getTaskID() : String
+getTitle() : String
+getDescription() : String
+getDueDate() : Date

Stats
-totalTasks : int
-completedTasks : int
-overdueTasks : int
-highPriorityTasks : int
-completionRate : double
+Stats(totalTasks, completedTasks, overdueTasks, highPriorityTasks)
+getTotalTasks() : int
+getCompletedTasks() : int
+getOverdueTasks() : int
+getHighPriorityTasks() : int

Role
-roleName : String

User
-userID : String
-username : String
-password : String
-assignedTasks : List<Task>
-notifications : List<Notification>
+User(userID, username, password)
+getUseID() : String
+getUsername() : String
+assignTask(Task : task)
+getAssignedTasks() : List<Task>
+receiveNotification(Notification : notification)

Project
-projectName : String
-projectDescription : String
-projectTasks : List<Task>
-projectTeam : List<User>
+addTask(Task : task)
+removeTask(Task : task)
+addTeamMember(User : user)
+removeTeamMember(User : user)

Notification
-notificationID : String
-sender : String
-message : String
-timestamp : Date
+Notification(notificationID, sender, message)
+getNotificationID() : Sting
+getSender() : String
+getMessage() : String
+getTimestamp() : Date

Organizations
-organizationName : String
-organizationTeams : List<Team>
+createTeam(teamName : String)
+addUserToTeam(User : user, Team, team)
+removeUserFromTeam(User : user, Team : team)

Reminder
-reminderType : String
-reminderDatatime : String
-reminderRecipents : List<User>
+setReminder(Task : task, datetime : String, recipients : List<User>)

DailyMessageGenerator
-user : User
-completedTasks : List<Task>
-remainingTaks : List<Task>
-upcomingTasks : List<Task>
+generateDailyMessage()
+getUser()
+getCompletedTasks()
+setCompletedTasks(completedTasks : List<Task>)
+getRemainingTasks()
+setRemainingTasks(remainingTasks : List<Task>)
+getUpcoming()
+setUpcoming(upcomingTasks : List<Task>)

TypeOfTask
-taskTypeName : String
-taskTypeDecription : String
-taskPriority : int
-taskCompletationTime : int
+getTaskTypeName()
+setTaskTypeName(name : String)
+getTaskTypeDescription()
+setTaskTypeDescription(description : String)
+getTaskPriority()
+setTaskPriority(priority : int)
+getTaskCompletionTime()
+setTaskCompletionTime(completionTime : int)