

Implementation Document: Displaying Ticket Time Duration in Redmine

Implementation Document: Displaying Ticket Time Duration in Redmine.....	1
1. Objective.....	27
2. Environment Setup.....	27
3. Prerequisites.....	28
4. Implementation Steps.....	29
4.1. Redmine Container Setup.....	29
4.2. Python Script for Time Calculation.....	29
4.3. Time Display in Redmine.....	35
5. Testing.....	36
6. Troubleshooting.....	36
7. Conclusion.....	36

TEST CASES- 1

Scenario 1: Login Page Accessibility

Summary: Verify that the Redmine page is accessible and loads correctly.	
<p>Given: The user has a web browser with internet access.</p> <p>And: The user has the correct URL of the Redmine.</p> <p>When: The user navigates to the Redmine login page URL.</p> <p>Then: The Redmine login page will load successfully.</p> <p>And: The username and password fields will be visible.</p> <p>Result: Passed</p>	<p>Screenshot</p>

TEST CASES- 2

Scenario 2: Unsuccessful Login to Redmine with Invalid Credentials

Summary: Verify that a user with invalid Username or password cannot log into the Redmine website and receives an appropriate error message[Invalid user or password].

Given: The Redmine login page is accessible.

When: The user enters an invalid username or password into the respective fields.

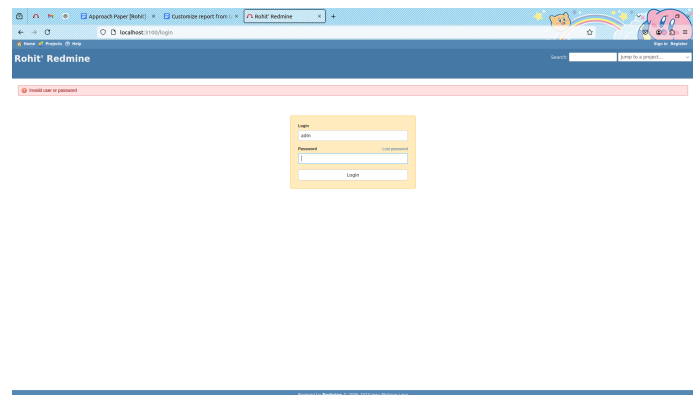
And: The user clicks the "Login" button.

Then: The user will see an error message indicating "Invalid username or password".

And: The user will remain on the login page.

Result: Passed

Screenshot



TEST CASES- 3

Scenario 3: Login Attempt with Empty Credentials

Summary: Verify that a user cannot log into the Redmine website when the username and password fields are left empty.

Given: The Redmine login page is accessible.

When: The user leaves the username and password fields empty.

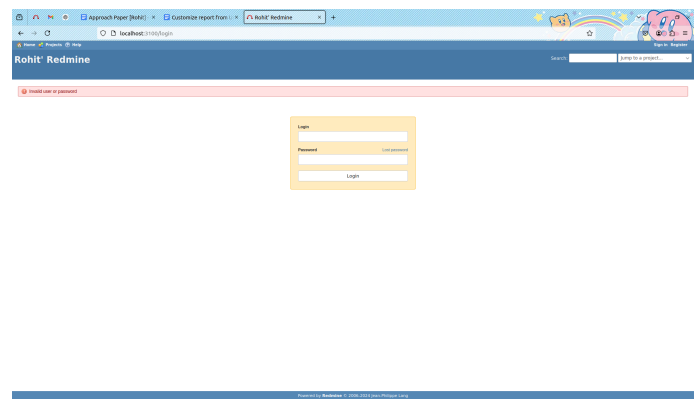
And: The user clicks the "Login" button.

Then: The user will see an error message indicating "Invalid user or password".

And: The user will remain on the login page.

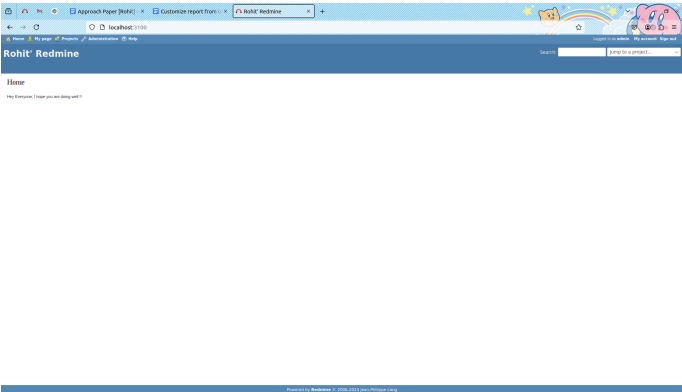
Result: Passed

Screenshot



TEST CASES- 4

Scenario 4: Successful Login to Redmine

Summary: Verify that a user with a valid Username and password can successfully log into the Redmine website.	
<p>Given: The Redmine login page is accessible.</p> <p>And: The user has a valid username and password.</p> <p>When: The user enters the valid username and password into the respective fields.</p> <p>And: The user clicks the "Login" button.</p> <p>Then: The user is redirected to the Redmine dashboard page.</p> <p>And: The user's login status will be displayed as logged in.</p> <p>Result: Passed</p>	<p>Screenshot</p> 

TEST CASES- 5

Scenario 5: Successful Navigation to Projects Page

Summary: Verify that a logged-in user can successfully navigate to the Projects page from the dashboard.

Given: The user is logged into the Redmine website.

And: The user is on the Redmine dashboard.

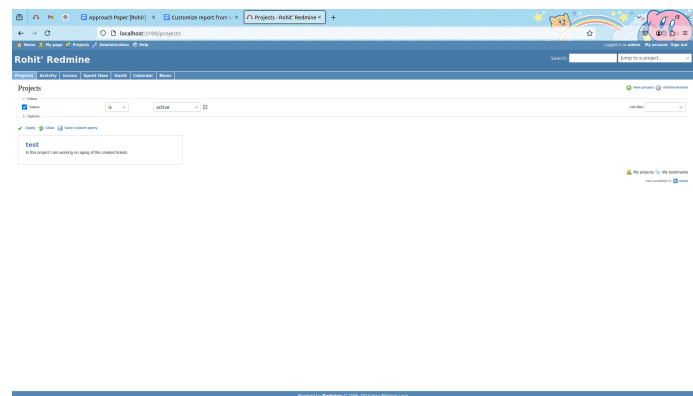
When: The user clicks on the "Projects" button on the dashboard.

Then: The user will be redirected to the Projects page.

And: The Projects page will display a list of projects.

Result: Passed

Screenshot



TEST CASES- 6

Scenario 6: Projects Button Functionality for Team Lead

Summary: Verify that the Projects button functions correctly for Team Lead.

Given: The Team Lead is logged into the Redmine website.

And: The user is on the Redmine dashboard.

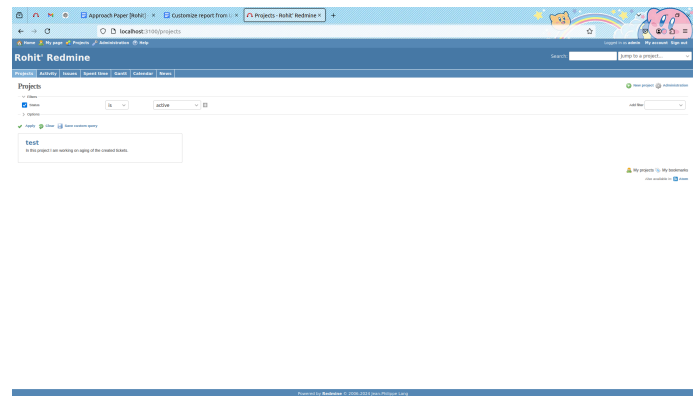
When: The user clicks on the "Projects" button on the dashboard.

Then: The user will be redirected to the Projects page.

And: The Projects page will display a list of projects relevant to the user's role and permissions.

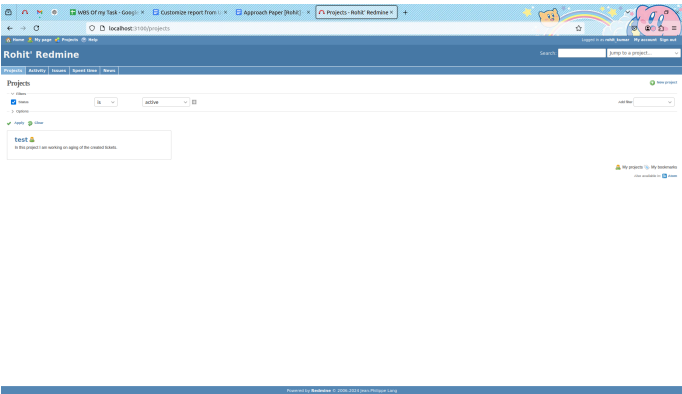
Result: Passed

Screenshot

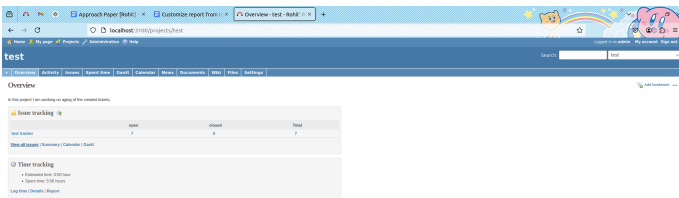


TEST CASES- 7

Scenario 7: Projects Button Functionality for Employee

Summary: Verify that the Projects button functions correctly for Employee	
<p>Given: The Employee is logged into the Redmine website.</p> <p>And: The user is on the Redmine dashboard.</p> <p>When: The user clicks on the "Projects" button on the dashboard.</p> <p>Then: The user will be redirected to the Projects page.</p> <p>And: The Projects page will display a list of projects relevant to the user's role and permissions.</p> <p>Result: Passed</p>	<p>Screenshot</p>  A screenshot of a web browser displaying the Redmine 'Projects' page. The browser's address bar shows 'localhost:3000/projects'. The page header includes the Redmine logo and navigation links like 'New project', 'My projects', and 'My issues'. The main content area is titled 'Projects' and contains a table with columns for 'Project', 'Status', and 'Action'. The table lists several projects, including 'Project A', 'Project B', and 'Project C', each with a corresponding status and action links. The page footer shows the Redmine version and copyright information.

Scenario 8: Successful Navigation to a Specific Project

Summary: Verify that a user can select and enter a specific project from the Projects page.																	
<p>Given: The user is logged into the Redmine website.</p> <p>And: The user is on the Projects page</p> <p>When: The user clicks on the desired project's name.</p> <p>Then: The user will be redirected to the selected project's overview page.</p> <p>And: The project overview page will display relevant project details such as description, members, and recent activity.</p> <p>Result: Passed</p>	<p>Screenshot</p>  <p>The screenshot shows a web browser window with the URL 'localhost:3000/projects/test'. The page title is 'test'. The navigation bar includes links for Home, Projects, Issues, Time tracking, and Settings. The main content area displays the 'test' project overview. It includes a table of issues with columns for status, priority, and count. Below the table, there is a section for 'Time tracking' with a table showing time spent on the project.</p> <table><tr><th>Status</th><th>Priority</th><th>Count</th></tr><tr><td>Open</td><td>High</td><td>1</td></tr><tr><td>Assigned</td><td>Medium</td><td>2</td></tr><tr><td>Completed</td><td>Low</td><td>1</td></tr></table> <p>Time tracking</p> <table><tr><th>Project</th><th>Time spent</th></tr><tr><td>test</td><td>10:00</td></tr></table>	Status	Priority	Count	Open	High	1	Assigned	Medium	2	Completed	Low	1	Project	Time spent	test	10:00
Status	Priority	Count															
Open	High	1															
Assigned	Medium	2															
Completed	Low	1															
Project	Time spent																
test	10:00																

Scenario 9: Project Selection Functionality for Team Lead

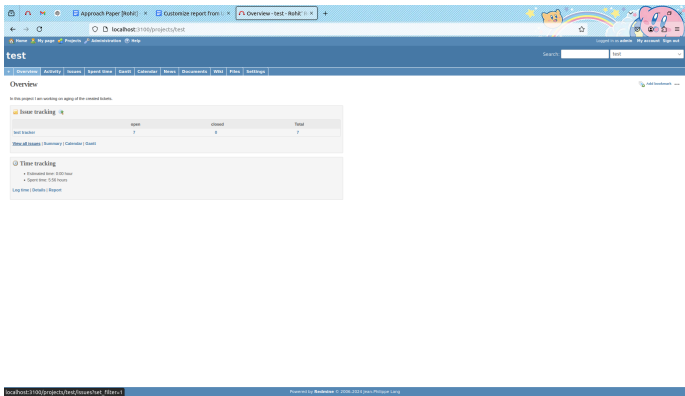
Summary: Verify that Team Lead can select and enter a specific project.

Given: The user is logged into the Redmine website with a specific role.
And: The user is on the Projects page.

When: The user clicks on the desired project's name.
Then: The user will be redirected to the selected project's overview page.
And: The project overview page will display relevant project details based on the user's role and permissions.

Result: Passed

Screenshot



Scenario 10: Project Selection Functionality for employee

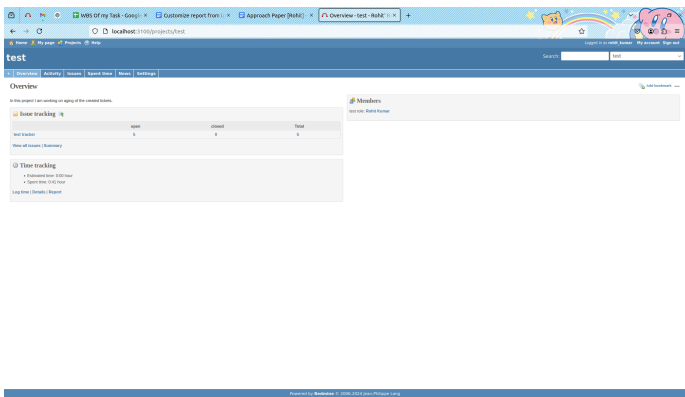
Summary: Verify that the employee can select and enter a specific project.

Given: The user is logged into the Redmine website with a specific role.
And: The user is on the Projects page.

When: The user clicks on the desired project's name.
Then: The user will be redirected to the selected project's overview page.
And: The project overview page will display relevant project details based on the user's role and permissions.

Result: Passed

Screenshot



Scenario 11: Issue table Visibility on Project Entry

Summary: Verify that the Issue table is automatically visible when a user enters a specific project.

Given: The user is logged into the Redmine website.
And: The user is on the overview page of a specific project.

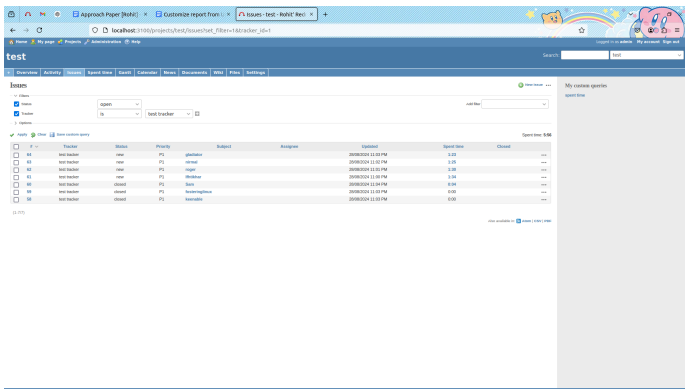
When: The user navigates to the specific project from the Projects page.

Then: The Issue table will be visible on the project overview page, in the issues section.

And: The Issue table will display relevant reports and data for the project, which includes the time duration column in the table.

Result: Passed

Screenshot



Scenario 12: Issue table Data Accuracy on Project Entry

Summary: Verify that the data displayed in the visible Issue table is accurate and up-to-date when entering a specific project.

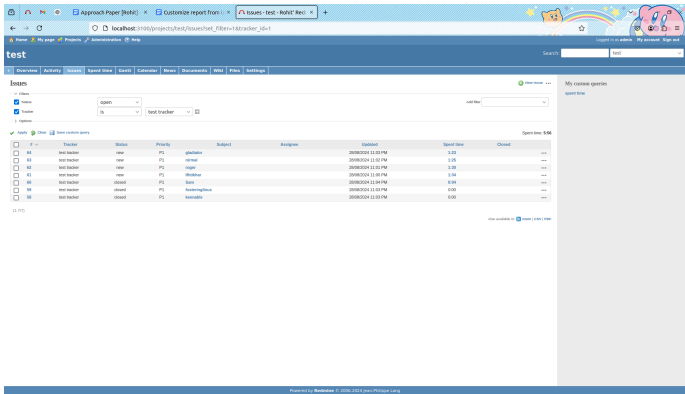
Given: The user is logged into the Redmine website.
And: The user is on the overview page of a specific project.

When: The user navigates to the specific project from the Projects page.

Then: The data displayed in the Issue table will be accurate and reflect the current status and time duration column is up-to date

Result: Passed

Screenshot



Scenario13: Issue table Functionality for Team Lead on Project Entry

Summary: Verify that Team Lead can view the visible Issue table within a specific project.

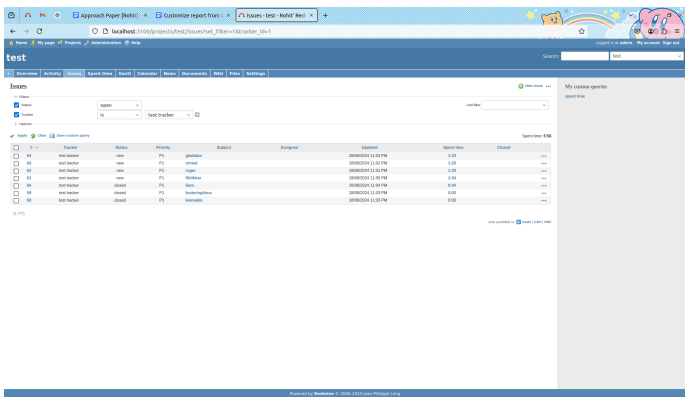
Given: The user is logged into the Redmine website with a specific role.
And: The user is on the overview page of a specific project.

When: The user navigates to the specific project from the Projects page.

Then: The Issue table will be visible on the project overview page.
And: The Issue table will display relevant data based on the user's role and permissions.

Result: Passed

Screenshot



Scenario 14: Issue table Functionality for Employee on Project Entry

Summary: Verify that the Employee can view the visible Issue table within a specific project.

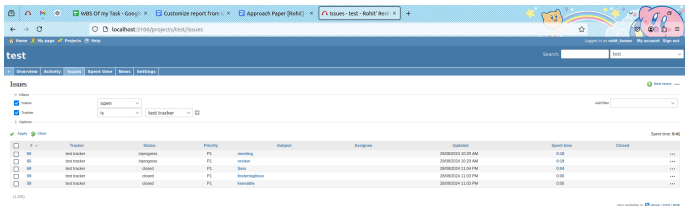
Given: The user is logged into the Redmine website with a specific role.
And: The user is on the overview page of a specific project.

When: The user navigates to the specific project from the Projects page.

Then: The Issue table will be visible on the project overview page.
And: The Issue table will display relevant data based on the user's role and permissions.

Result: Passed

Screenshot



Scenario 15: Report Download Button functionality

Summary: Verify that a download button for the report in CSV format is functional for the Issue table page.

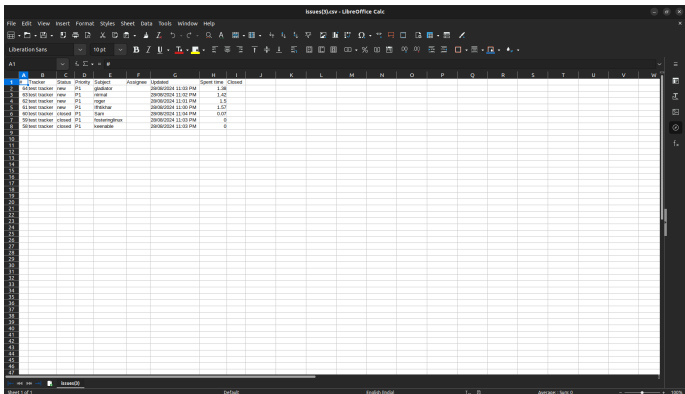
Given: The user is logged into the Redmine website.
And: The user is on the overview page of a specific project with the Issue table visible.

When: The user clicks the CSV button

Then: The report table is downloaded, as it was asked to export.
And: The table includes the time duration column in the report.

Result: Passed

Screenshot



ID	Status	Priority	Category	Assignee	Subject	Time Spent
1	closed	P1	general		20000000 11:00 PM	1:30
2	closed	P1	bug		20000000 11:00 PM	1:45
3	closed	P1	bug		20000000 11:00 PM	1:15
4	closed	P1	feature		20000000 11:00 PM	1:15
5	closed	P1	task		20000000 11:00 PM	0:00
6	closed	P1	improvement		20000000 11:00 PM	0
7	closed	P1	upgrade		20000000 11:00 PM	0

TEST CASES- 16

Scenario 16: Team Lead Fetches Report for Assigned Project

Summary: Verify that a Team Lead can fetch the report for a project they are involved in.

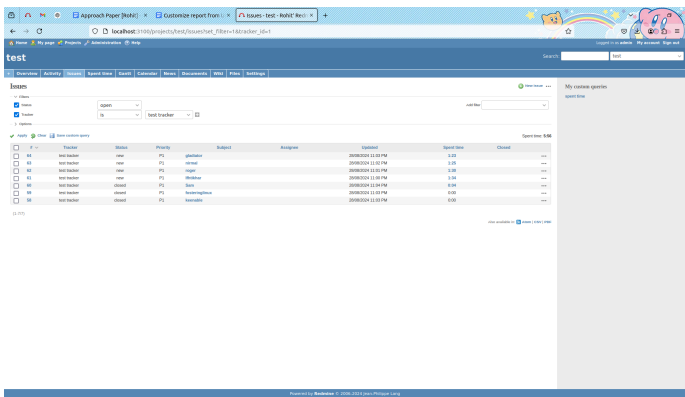
Given: The Team Lead is logged into the Redmine website.
And: The Team Lead has been assigned to a specific project.
And: The report collection frequency is up-to-date..

When: The Team Lead navigates to the overview page of the assigned project.
And: The Team Lead clicks the download button for the daily report.

Then: The system will provide the daily report in CSV format for download.
And: The report will contain accurate and up-to-date data for the project, including the time duration column.

Result: .Passed

Screenshot



Result: .Passed

[illegible]

TEST CASES- 18

Scenario 18: Verify Update of "Time Duration" Column

Summary: Verify that the "Time Duration" column in the table is updated on a regular basis.

Given: The user is logged into the Redmine website.

And: The user is on the overview page of a specific project.

When: The user views the Issue table.

And: A day passes, and the user views the Issue table again.

Then: The "Time Duration" column should reflect the updated duration for the current day.

And: The data in the "Time Duration" column should be accurate and reflect the total time elapsed from the start date.

Result: .Passed

Screenshot

The screenshot shows the AWS IAM console interface. In the left sidebar, the 'test' group is selected. The main content area displays a table of permissions for the 'test' group. The table has columns for Name, Action, Resource, Policy, and Status. The 'test' permission is highlighted in blue. The table shows that the 'test' permission is associated with the 'test' policy and is in a 'Granted' status. The 'test2' permission is also listed, associated with the 'test2' policy and in a 'Granted' status.

Name	Action	Resource	Policy	Status
test	test	test	test	Granted
test2	test2	test2	test2	Granted

Approach:- Integrating Python Script for Enhanced Reporting

Task Description

Customise report from Redmine: Integrate a python script to extend its reporting capabilities, focusing on detailed time tracking. Users and Team Leads can utilise the Script within the Redmine environment for generating customizable reports, including the time duration column.

Context of the Task

In the context of project management within organisations, Redmine serves as a robust tool for issue tracking and project management. Enhancing its reporting capabilities with script that allows for detailed time tracking, improving overall project oversight and management efficiency.

Benefits:

For Users: Provides seamless integration within Redmine, allowing users to generate personalised project reports and track time spent on tasks more effectively.

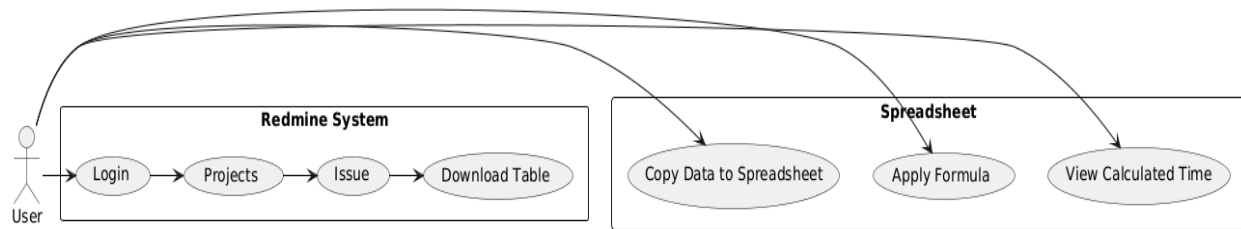
For Team Leads: Offers comprehensive reporting capabilities within Redmine, facilitating better oversight of team activities and project progress.

Current System

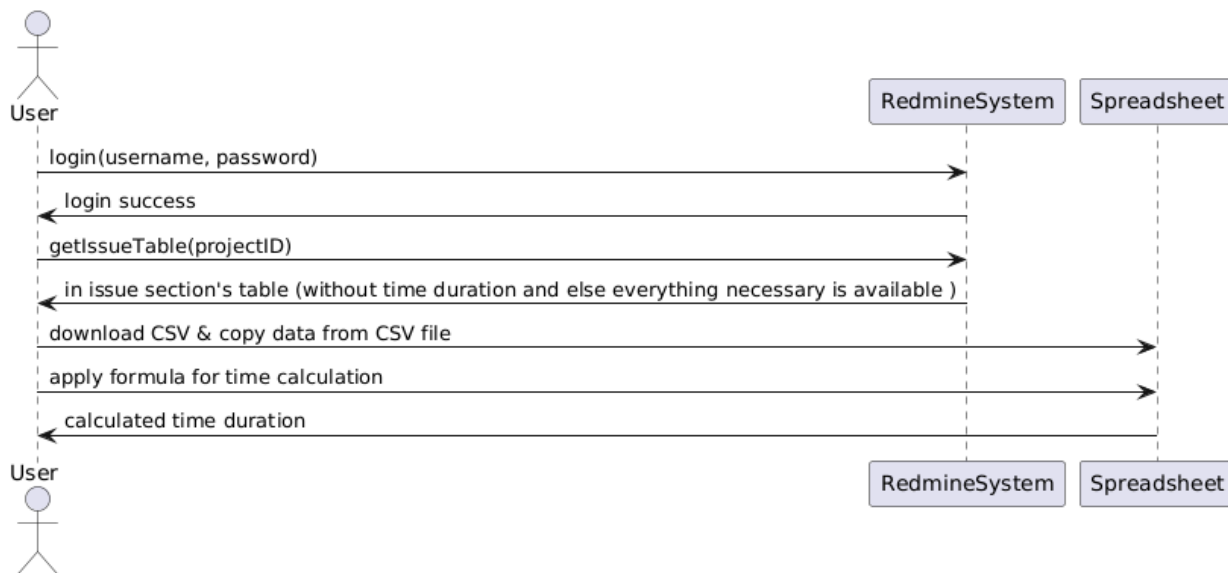
Redmine provides basic reporting functionalities but lacks detailed time tracking features. Users often export data to external tools for in-depth analysis and reporting.

Current Scenario

Use Case Diagram:



Sequence Diagram:



Proposed Solution

Custom Time Tracking Script:

Develop a Python script to calculate the time duration of all issues in Redmine, eliminating the need for additional plugin integration.

Calculation Logic:

- **Closed Issues:** The script calculates time as **closed time - created time**. Once an issue is closed, the script permanently stops time calculation for that issue.
- **Non-Closed Issues:** Time is calculated as **current time - created time**. In subsequent runs, the script saves the last logged time and calculates the time from there.

Automated Reporting:

The script automatically updates the time tracking data, allowing for accurate and up-to-date reporting without manual intervention.

Role-Based Access:

Continue to leverage Redmine's existing role-based access control to restrict or grant permissions for viewing and generating reports.

Export Options:

Enhance the script to include functionality for exporting time tracking data in CSV and PDF formats directly from Redmine to meet reporting needs.

Scope of Work

- **Script Development:** Create and fine-tune the Python script for accurate time calculation.
- **Integration with Redmine:** Ensure the script seamlessly integrates with Redmine's existing database and workflows.
- **Custom Time Reports:** Allow users to generate custom time reports based on the calculated durations.
- **Export Functionality:** Develop export features for the generated reports in CSV and PDF formats.
- **Testing and Quality Assurance:** Perform thorough testing to ensure the script's accuracy, performance, and security.
- **Documentation:** Provide clear documentation for installing, configuring, and using the script within Redmine.
- **Deployment:** Deploy the script to the organization's Redmine instance, ensuring it works correctly for all users.

Why: Rationale for the Solution

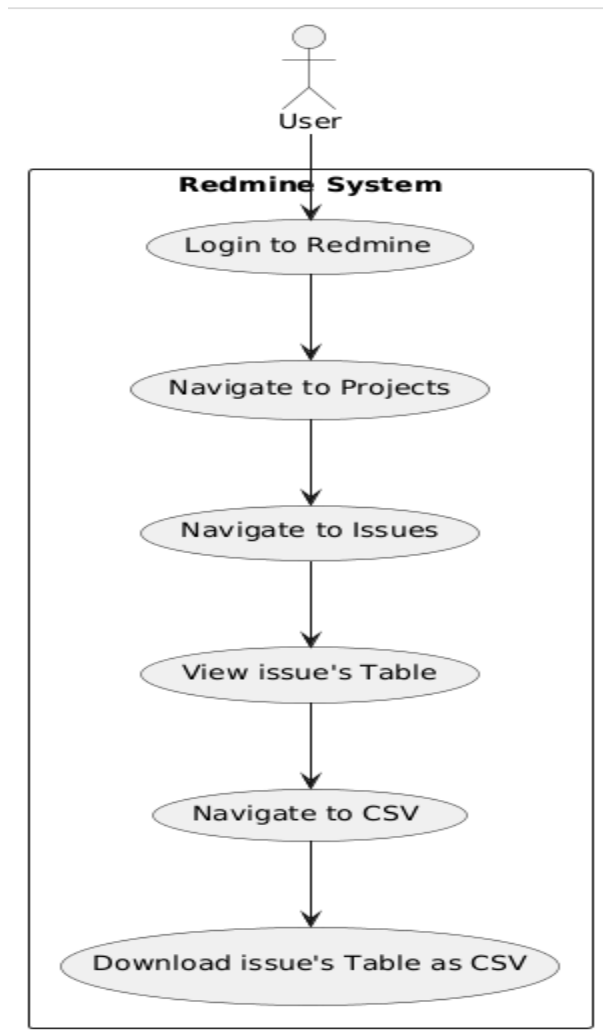
The custom Python script provides a flexible, automated solution for time tracking and reporting without the need for extensive plugin development. It ensures accurate data tracking by automating time calculations and permanently stopping time tracking for closed issues. This approach enhances workflow efficiency and data accuracy, supporting better decision-making and project oversight for users and team leads.

Feature, Technology And Benefit

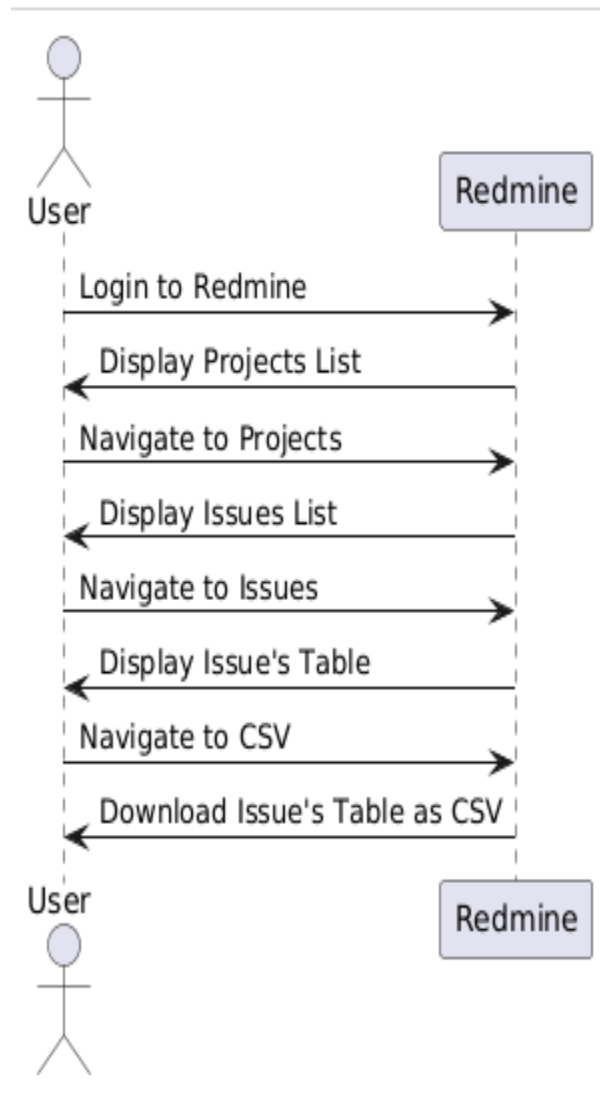
Feature	Technology	Benefit
Custom Time Tracking	Python Script	Automates time calculation for both open and closed issues, enhancing accuracy and efficiency.
Automated Reporting	Python Script	Provides real-time updates and accurate time tracking data without manual intervention.
Export Functionality	Redmine's Existing Framework	Allows users to export time-tracking reports in CSV and PDF formats, facilitating easy sharing and analysis.
Role-Based Access	Redmine's Existing Framework	Maintains secure access to time tracking and reporting features based on user roles, ensuring data integrity.

Proposed Architecture / Application Workflow

Use Case Diagram:



Sequence Diagram:



Pre-requisites

Hardware Requirements

- ☐ 1 Server
- ☐ 8 GB of RAM
- ☐ Minimum of 4 Core CPU

Software Requirements

- ☐ Operating System: Linux distribution like Ubuntu 22.04 LTS, windows 10 & above and android
- ☐ Development Tools: Python3, Visual Studio Code

Networking Requirements

- Ethernet, IP Address

Conclusion

Summary: This project focuses on developing a custom Python script to automate time tracking and reporting within Redmine. By calculating the time duration of issues based on their status, the solution enhances accuracy and efficiency in project management. The approach eliminates the need for additional plugin integration, providing a streamlined, tailored solution that meets the organisation's specific reporting needs. This automated system supports efficient project tracking, better decision-making, and a seamless user experience within Redmine.

1. Objective

To display the total time duration for each ticket in Redmine in two formats:

- **For open tickets:** Time = Current Time - Created Time
- **For closed tickets:** Time = Closed Time - Created Time

The time should be visible on the issues section of Redmine where all the ticket information is displayed. The time will be shown in both **hours** and **DDHHMM** (Days, Hours, and Minutes) format.

2. Environment Setup

- **Redmine Container:** Podman is used to manage the Redmine container. A container named **redmine-app** was set up locally to run Redmine.

```
cat redmine  
#!/bin/bash  
  
# Create a folder for PostgreSQL data  
echo "Creating folder for PostgreSQL data..."  
mkdir -p /home/rohit/data/redmine/  
postgres  
  
# Change ownership of the folder  
echo "Changing ownership of the folder..."  
podman unshare chown 999:999 /home/rohit/data/redmine/postgres  
  
# Create the Redmine Pod  
echo "Creating the Redmine Pod..."  
podman pod create --name redmine --publish 3100:3000 --publish  
5433:5432  
  
# Run the PostgreSQL container  
echo "Running the PostgreSQL container..."  
podman run -dt \  
  --pod redmine \  
  --name redmine-postgres \  
  -e POSTGRES_DB=keen \  
  -e POSTGRES_USER=postgres \  
  -e POSTGRES_PASSWORD=password \  
  -e POSTGRES_HOST_AUTH_METHOD=trust \  
  -e PGDATA=/var/lib/postgresql/data/pgdata \  
  -v /home/rohit/data/redmine/postgres:/var/lib/postgresql/data \
```

```
docker.io/postgres:latest

# Run the Redmine application container
echo "Running the Redmine application container..."
podman run -dt \
--pod redmine \
--name redmine-app \
-e REDMINE_DB_POSTGRES=127.0.0.1 \
-e REDMINE_DB_PORT=5432 \
-e REDMINE_DB_DATABASE=keen \
-e REDMINE_DB_USERNAME=postgres \
-e REDMINE_DB_PASSWORD=password \
docker.io/redmine:latest
```

- **Python:** A script is developed in Python to calculate the time difference and update it in Redmine.

3. Prerequisites

- **Podman Installed:** Ensure Podman is installed and configured correctly on your system.

sudo apt install podman -y

- **Python Environment:** Python installed with the necessary libraries.

sudo apt install python3 -y

sudo apt install python3-pip -y

- **Redmine API Key:** The API key is required to interact with the Redmine server (Rohit's API key: [aad295ea58d4e0205ad7d7e94ae076e451c3f26f](#)).

- **Redmine URL:** Local Redmine URL (Rohit's Redmine URL: <http://localhost:3100>).

4. Implementation Steps

4.1. Redmine Container Setup

1. Start the Redmine container using Podman:

```
podman start redmine-app
```

2. Ensure the Redmine instance is running at the desired URL (e.g., <http://localhost:3100>).

4.2. Python Script for Time Calculation

1. **Script Logic:**

- Fetch all issues from Redmine via the API.
- For **open tickets**: Calculate time as **current time - created time**.
- For **closed tickets**: Calculate time as **closed time - created time**.
- Format the calculated time in both **hours** and **DDHHMM** format.

2. **Script Execution:**

- Place the script in the desired directory (e.g., [~/Documents/time_spent_test.py](#)).
- Run the Python script:

```
python3 ~/Documents/time_spent_test.py
```

- The script will:
 - Fetch the required issues from Redmine.
 - Calculate the time difference.
 - Update Redmine with the calculated time in the required formats.

4.3. Time Display in Redmine

1. After running the script, the total time duration will be visible in the **Issues** section of Redmine:
 - **For Open Tickets:** Time calculated as `current time - created time`.
 - **For Closed Tickets:** Time calculated as `closed time - created time`.
2. The time will be shown in both:
 - **Hours**
 - **DDHHMM** (e.g., 2 days, 4 hours, and 30 minutes = 020430)

test

OverviewActivityIssuesSpent timeGanttCalendarNewsDocumentsWikiFilesSettings

Issues

Filters

Status

open

Tracker

is

test tracker

Options

Apply

Clear

Save custom query

Spent time: 438:19

#	Tracker	Status	Priority	Subject	Assignee	Updated	Closed	Formatted Spent Time	Created	Spent time
173	test tracker	new	P1	c		02/10/2024 02:02 AM		0D, 02H, 04Min	25/09/2024 11:57 PM	148:06
172	test tracker	new	P1	b		02/10/2024 02:02 AM		0D, 02H, 05Min	25/09/2024 11:56 PM	148:06
171	test tracker	new	P1	a		02/10/2024 02:02 AM		0D, 02H, 05Min	25/09/2024 11:56 PM	148:07

(1-3/3)

My custom queries

spent time

Powered by Redmine © 2006-2024 Jean-Polype Lang

5. Testing

- Confirm the Redmine issues section displays the calculated time for both open and closed tickets.
- Verify that the time format is correct in both hours and DDHHMM.

6. Troubleshooting

- **Redmine Container Not Starting:** Ensure Podman is installed and configured correctly. Use `podman ps` to check if the container is running.
- **Incorrect Time Calculation:** Review the Python script to ensure the correct time difference logic is implemented.
- **No Time Visible:** Ensure the script has correctly updated the fields in Redmine via the API.

7. Conclusion

This implementation allows for the automatic calculation and display of time duration for Redmine tickets, shown in both hours and DDHHMM format, enhancing issue tracking and management.

