**Project Report: Agile Project Management with Jira**

**Project Name:** Password Manager Development

**Student Name:** Avinav Mendu [BT23CSE053]

**1. Project Aim**

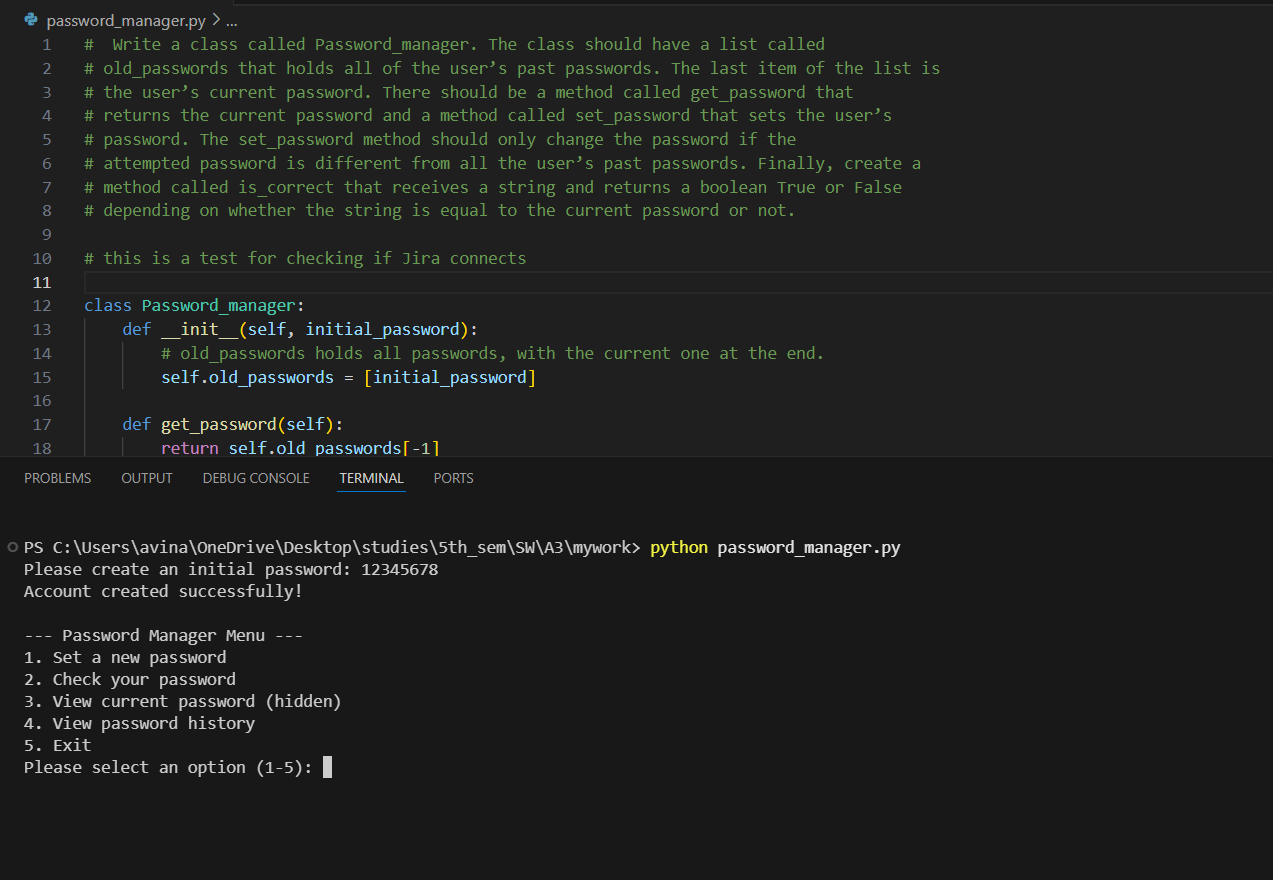
The primary objective of this project was to demonstrate a complete software development lifecycle for a Python application using Agile methodologies. This involved developing a functional Python program and managing the entire process—from planning and task assignment to code integration and testing—using Atlassian Jira as the project management tool. The project also aimed to integrate Jira with GitHub to establish end-to-end traceability from a planned task to the corresponding source code.

**2. Python Program: The Password Manager**

The software component of this project is a command-line based Password\_manager application written in Python. The program allows a user to manage their passwords securely by ensuring that new passwords have not been used previously.

**Core Features:**

* Set an initial password.
* Change the current password, validating that the new password is unique.
* Verify if a given password attempt is correct.
* View a history of all previously used passwords.



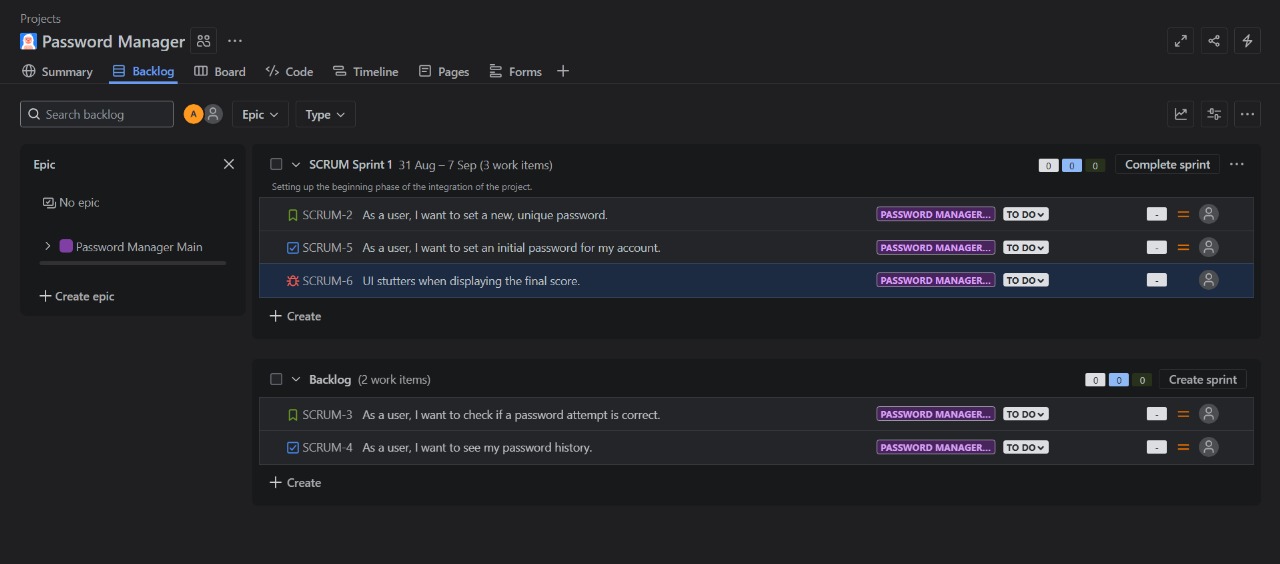
**3. Jira Project Management Workflow**

To manage the development of the Password Manager, an Agile Scrum methodology was implemented within a Jira Software project. The workflow was broken down into several distinct phases.

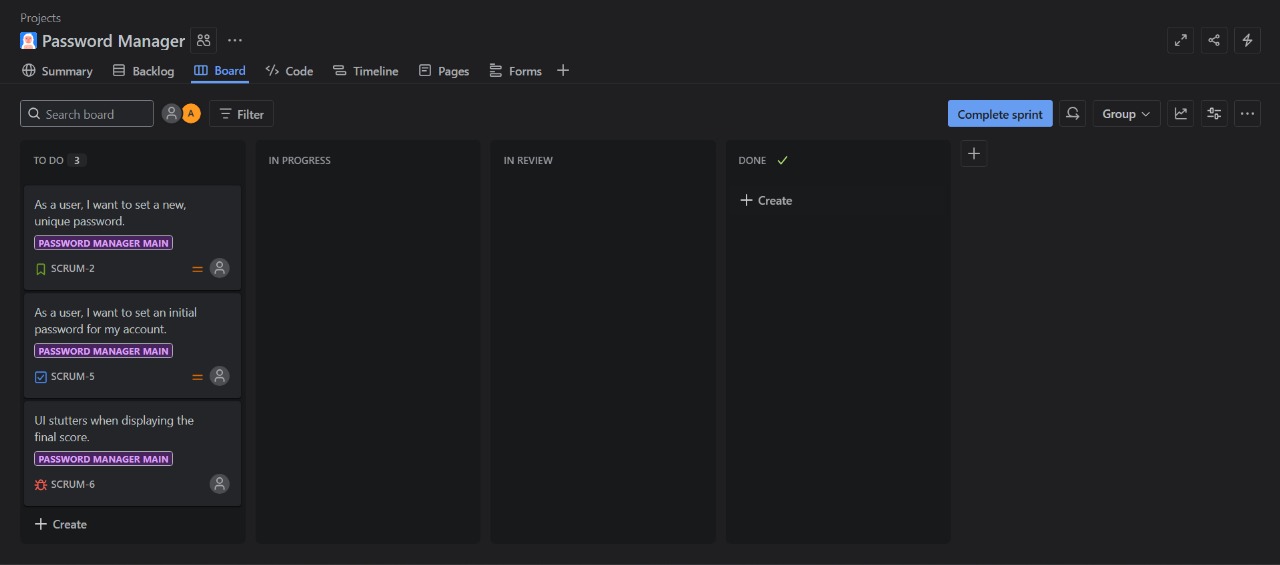
**Phase 1: Project Setup & Backlog Population**

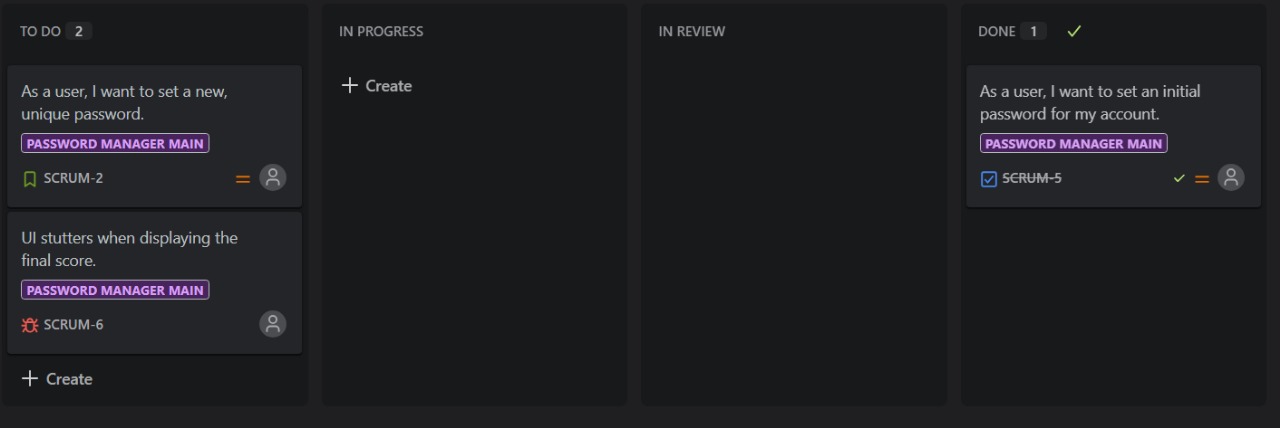
A Scrum project titled "Password Manager" was created in Jira. The initial phase involved translating the application's requirements into a structured product backlog. This was organized using Jira's issue hierarchy:

* **Epic:** A single Epic, Password Manager Main, was created to group all related work.
* **User Stories & Tasks:** Functional requirements were written as User Stories (e.g., "As a user, I want to set a new, unique password") and Tasks.
* **Bugs:** Potential issues were logged as Bugs (e.g., "UI stutters when displaying the final score").

**Phase 2: Sprint Execution**

The highest priority items from the backlog were selected for development in a two-week sprint, "SCRUM Sprint 1." The sprint board was used to visualize the flow of work from TO DO -> IN PROGRESS -> DONE.

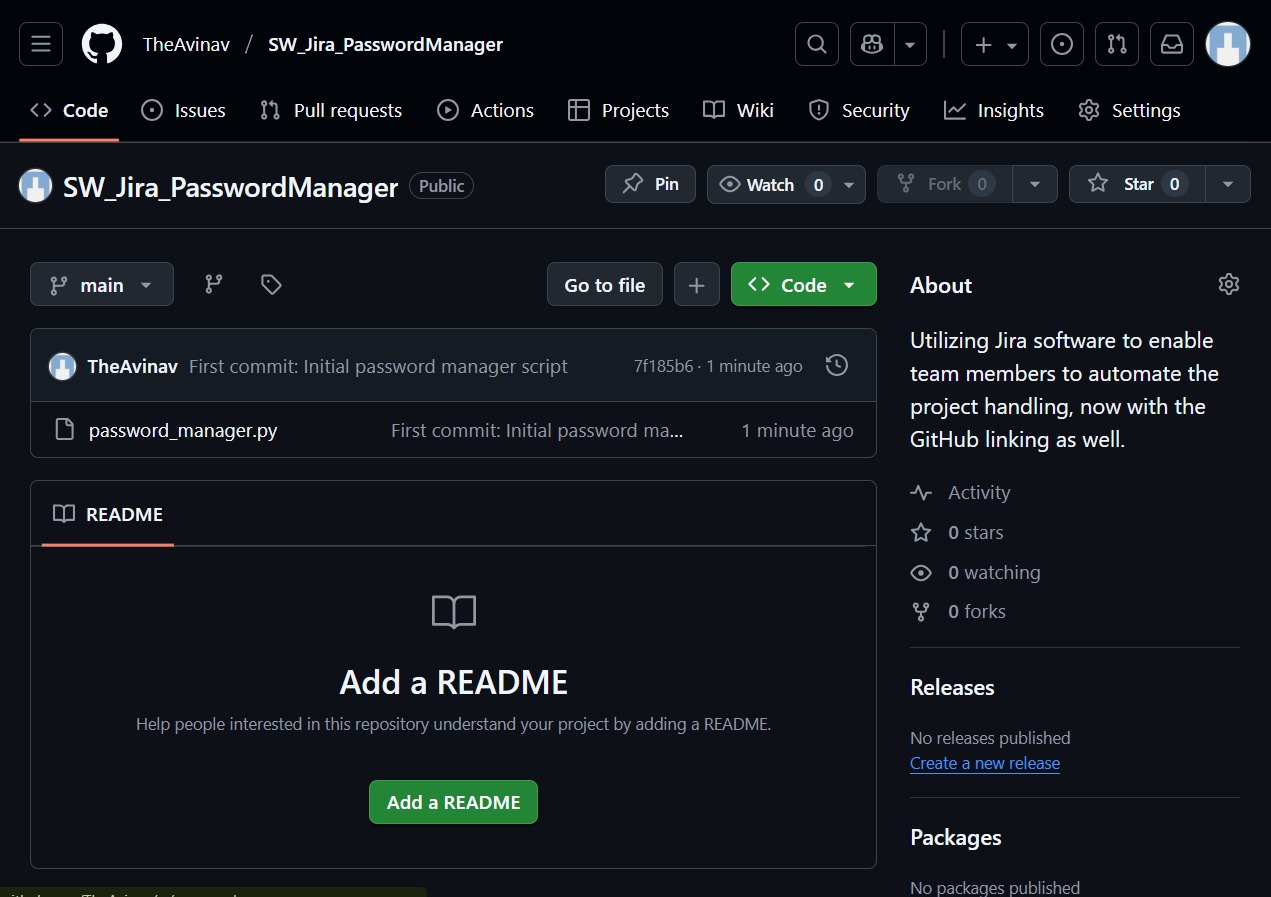


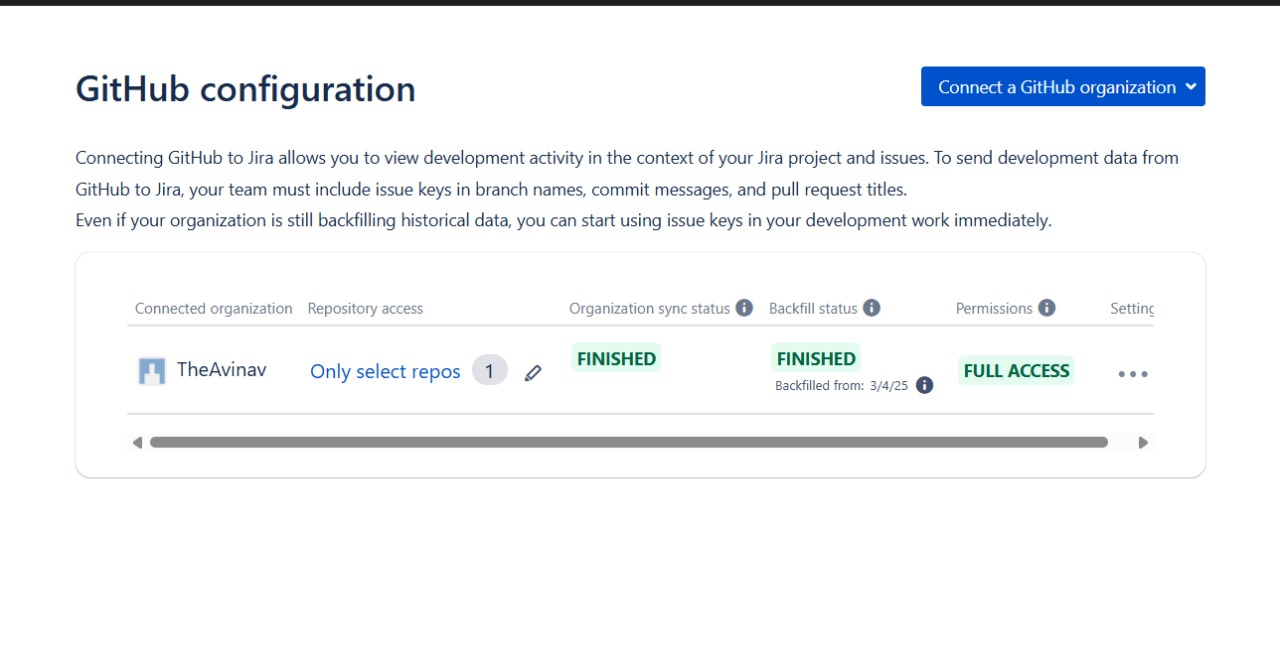


**Phase 3: Advanced Feature Implementation**

To further refine the project management process, several advanced Jira features were utilized:

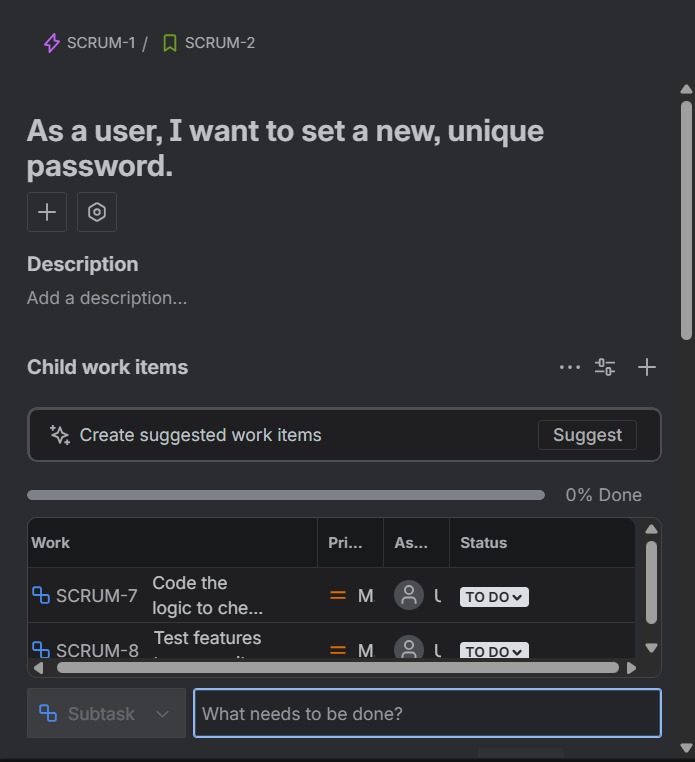
* **Sub-tasks:** The user story for setting a new password was broken down into smaller, actionable sub-tasks: Code the logic and Test features. This demonstrates the decomposition of complex work.
* **Automation:** An automation rule was configured to automatically assign any newly created Bug to the project lead. This reduces manual administration.
* **GitHub Integration:** Jira was connected to a GitHub repository hosting the Python code. Commits made to the code were tagged with the relevant Jira issue key. This created a direct link within the Jira ticket to the specific code changes, providing full traceability.

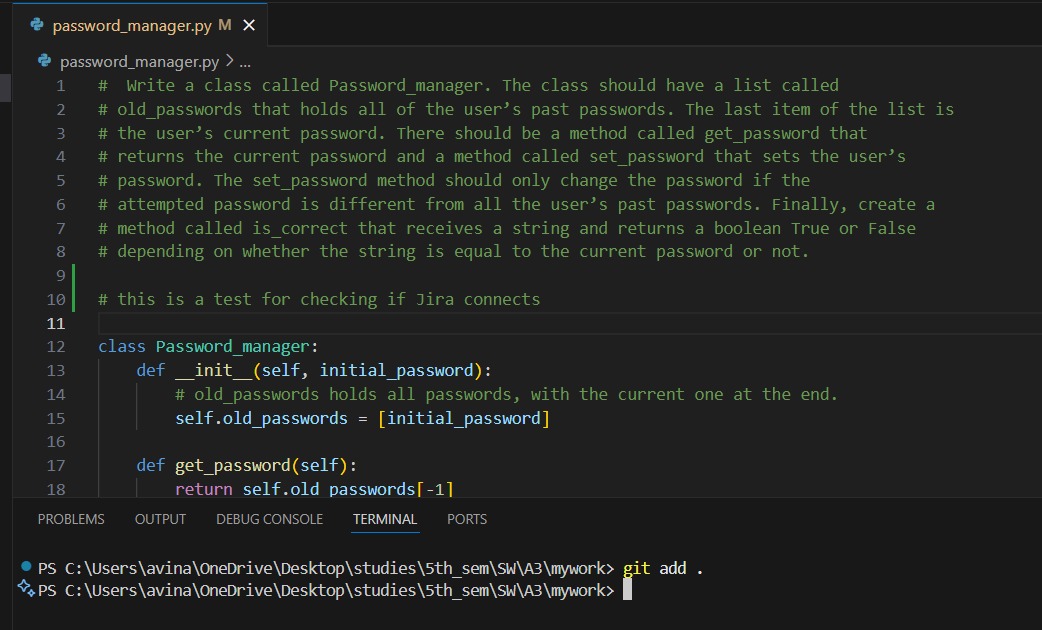


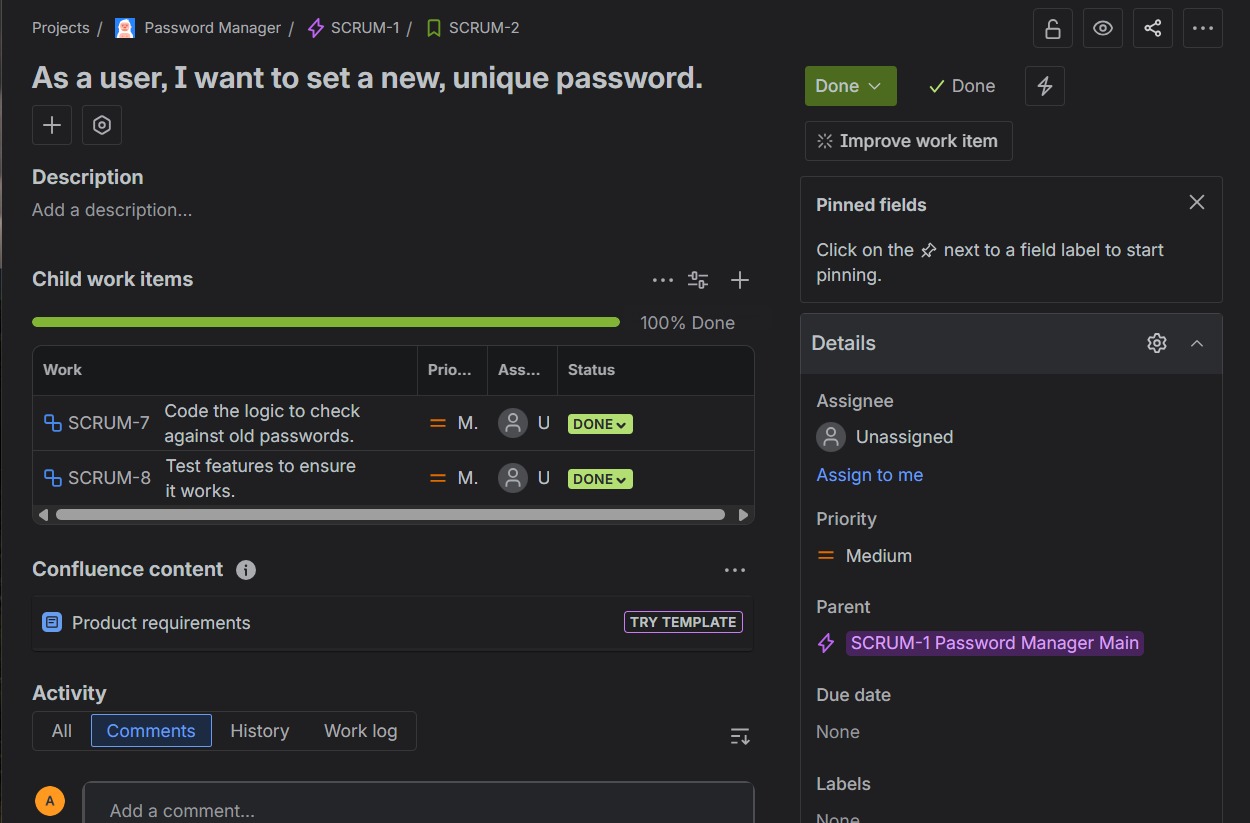
**4. Test Management and Traceability**

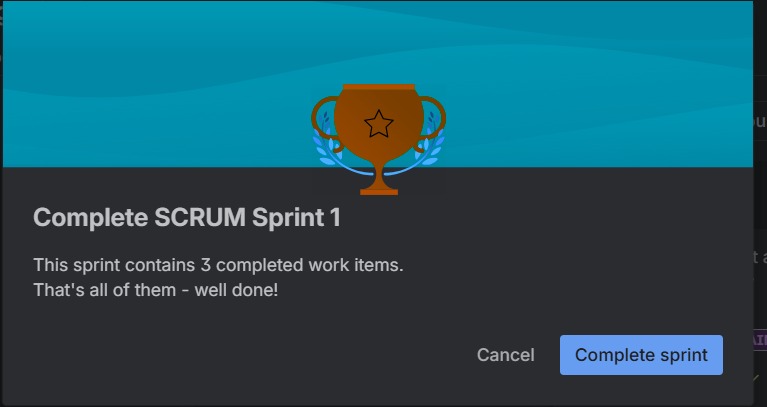
To fulfill the requirement for software testing within Jira without a paid plug-in, a custom issue type was created.

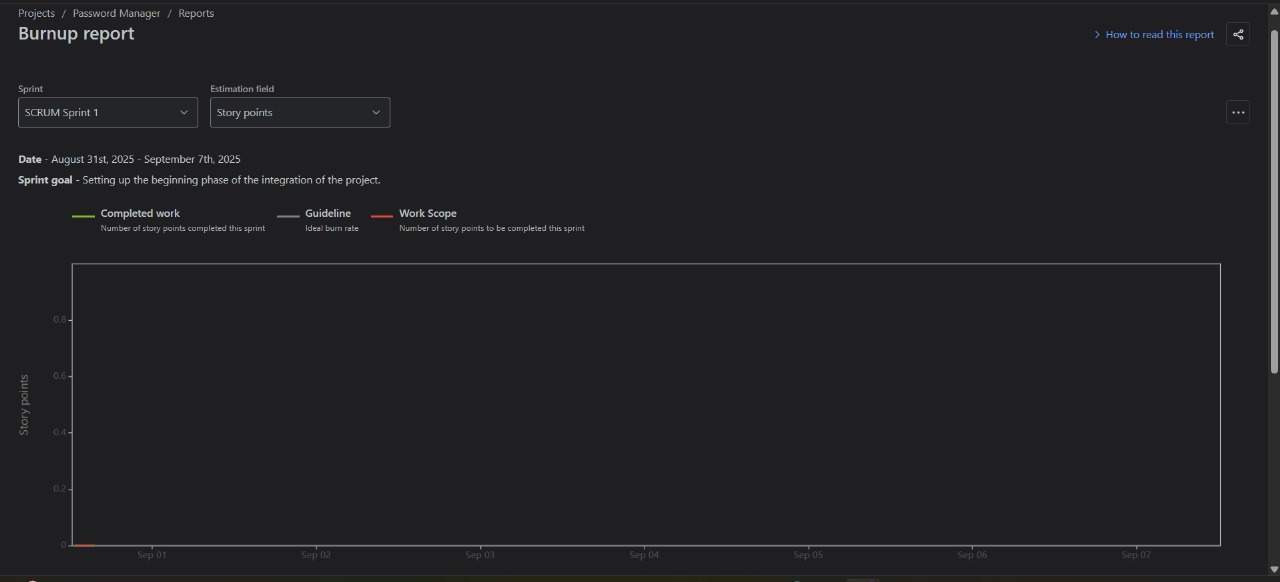
1. **Custom Issue Type:** A new issue type called Test Case was created.
2. **Test Case Creation:** A Test Case issue was created with a descriptive summary (Verify password cannot be reused) and detailed steps in the description field.
3. **Issue Linking:** Crucially, this Test Case was linked to the original User Story using Jira's "Link Issue" feature. This establishes **traceability**, proving that a specific requirement was tested. The Test Case was then moved to DONE to signify that the test passed.

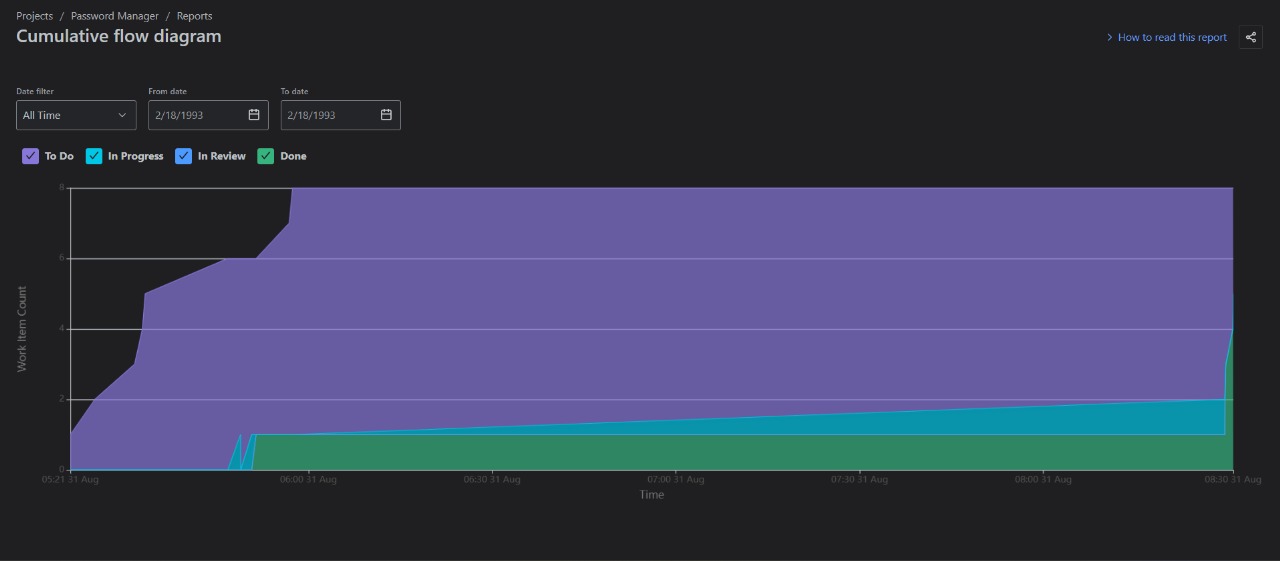
**5. Timeline of Events**











**6. Jira Features Utilized**

Below is a summary of the key Jira features used in this project, demonstrating a comprehensive application of the Scrum framework.

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Backlog** | The product backlog was used as the single source of truth for all project requirements, including stories, tasks, and bugs. [Screenshot of Backlog] |
| **Sprints** | A two-week sprint was planned, started, and completed to deliver a functional increment of the software. [Screenshot of completed Sprint Report] |
| **Scrum Board** | The board provided a visual representation of the work in progress during the sprint, with columns for TO DO, IN PROGRESS, and DONE. [Screenshot of Board] |
| **Sub-tasks** | Larger stories were broken down into smaller, manageable sub-tasks to better track progress and distribute work. [Screenshot of a story with its sub-tasks] |
| **Automation** | An automation rule was created to auto-assign new bugs, demonstrating how to streamline administrative tasks. [Screenshot of the automation rule configuration] |
| **Reports (CFD & Burndown)** | After completing the sprint, the Cumulative Flow Diagram (CFD) and Burndown Chart were analyzed to track workflow and progress against the plan. [Screenshot of your Burndown Chart or CFD] |

**7. Conclusion**

This project successfully demonstrated the end-to-end management of a software project using Jira and the Agile Scrum framework. By structuring work into a backlog, executing a focused sprint, and integrating with development tools like GitHub, a high degree of organization and traceability was achieved. The use of custom issue types for test cases further proved the flexibility of Jira in adapting to a team's specific needs without relying on external plugins. The process resulted in a clear, manageable, and well-documented workflow from initial requirement to final, tested code.

