FlowMate

2ND SPRINT REVIEW

Persistence, Files, and States: Enhancing FlowMate's Core.

Team 5

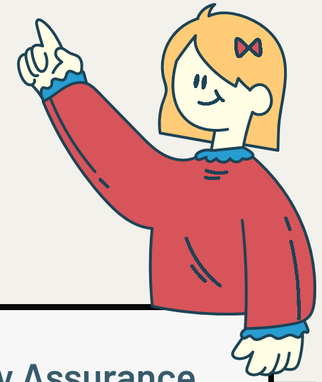
Della Corte Alessio
a.dellacorte23@studenti.unisa.it

Pecoraro Sara
s.pecoraro19@studenti.unisa.it

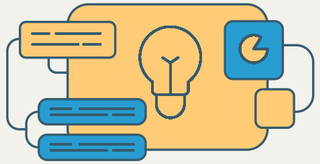
Sabatino Ester
e.sabatino14@studenti.unisa.it

Siddiqa Ayesha
a.siddiqa1@studenti.unisa.it

Sprint Recap



What We Delivered



We implemented rule persistence, active/inactive states, and extensive file operations (Move/Copy/Delete/Write), fully integrated into the GUI.

Original Commitments

Our goal was to complete User Stories 08 to 13, focusing on rule lifecycle management, data saving, and filesystem interactions.

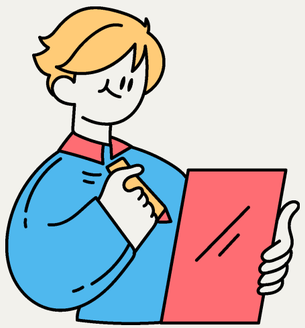
Velocity Achieved

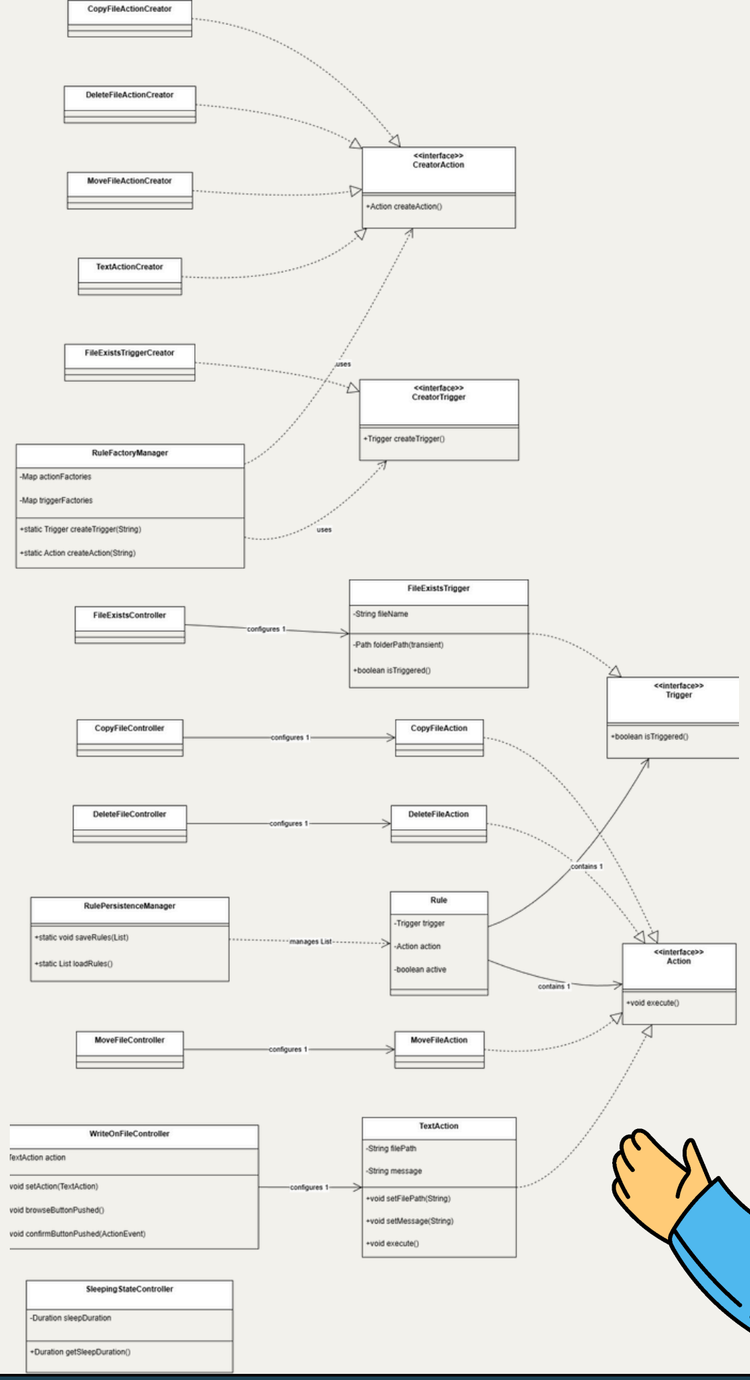
We achieved 34 Story Points, covering all planned User Stories.

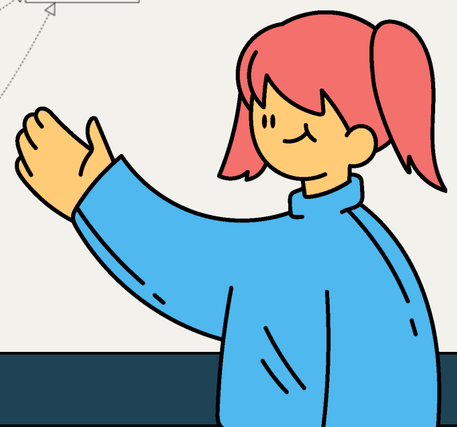
Quality Assurance

We enforced reliability by implementing comprehensive Unit Tests for every new feature, ensuring robust behavior for persistence, sleep logic, and file operations.

Implementation Achieved







What Went Well

Successful Collaborations

Despite working on separate GUI components, the team coordinated closely to ensure a consistent look and feel. We established a shared design standard early on, resulting in a seamless and polished user experience.

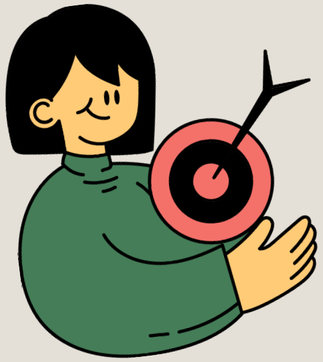
Parallel Workflow Strategy

We identified critical dependencies early, allowing us to develop the backend logic and the frontend simultaneously. This parallel approach eliminated bottlenecks and doubled our implementation speed.

Enhanced User Experience

We made the interface more responsive and informative. Instead of just a static list, the GUI now visually indicates the state of every rule, and we standardized the design of all new popup windows to ensure a consistent, professional look.

WHAT DIDN’T WORK



UNDERESTIMATED REFRACTORING

We assumed adding persistence would be simple, but refactoring consumed significantly more time than planned, compressing the schedule for other tasks.

AMBIGUOUS TASK DEFINITIONS

Some User Stories lacked precise boundaries. This ambiguity made it difficult to determine when a specific task was truly "Done," leading to confusion during the development and testing phases.

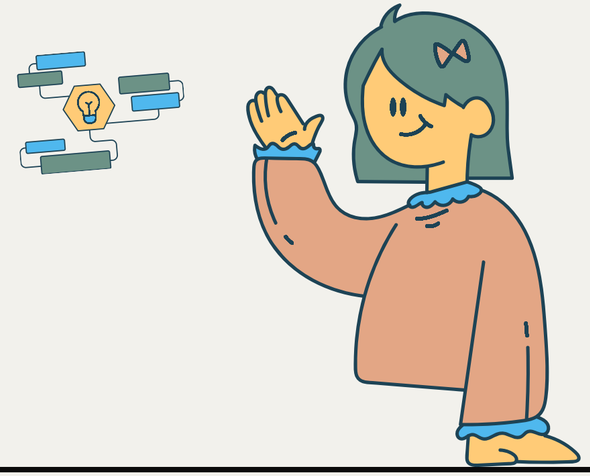
NON-ATOMIC USER STORIES

In some User Stories, lack of atomicity led us to overlook requirements, resulting in the partial implementation of the specific feature.

Sprint Retrospective

START	MORE OF	LESS OF	KEEP DOING	STOP
				
In the next sprint, we plan to further standardize architectural decisions, applying design patterns earlier and validating them with small prototypes.	In future sprints, we aim to invest more time in early design reviews, testing rule execution, and documenting key technical decisions.	We plan to reduce last-minute refactoring and duplicated logic by clarifying responsibilities earlier in the development process.	We will continue using design patterns and maintaining a clear separation between UI and business logic, as this approach has proven effective so far.	We decided to avoid direct instantiation of concrete classes and late architectural changes, as they increase coupling and refactoring costs.
				

Used Patterns



Factory Method

Used to dynamically create Actions and Triggers according to user choices. Object creation is centralized, keeping the main controller independent from concrete implementations.

Singleton

We apply it by making the RuleEngine constructor private. This allows the system components, to access the shared engine instance via a static method rather than creating separate objects.

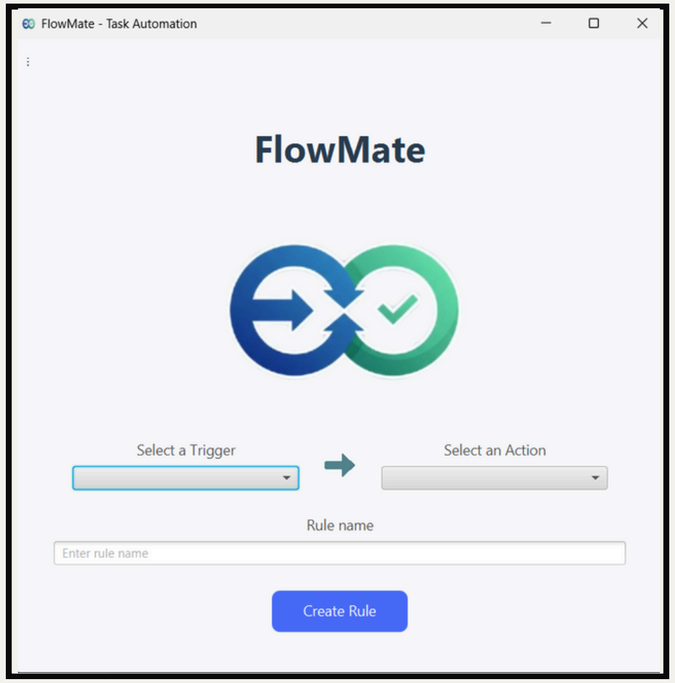
MVC

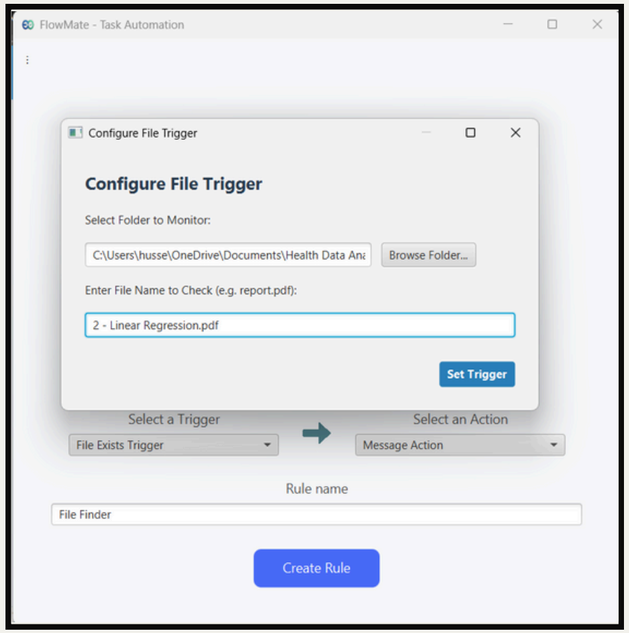
The application follows the MVC architectural pattern, separating UI, logic, and data

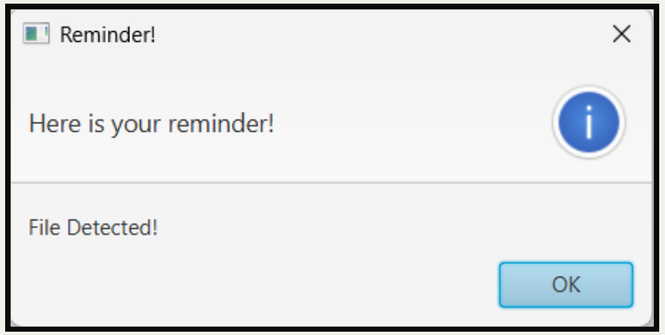
State

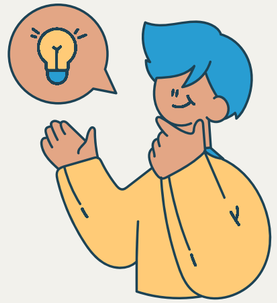
The State pattern is used to manage the lifecycle of a rule. Each rule can be in different states (e.g. Active, Sleeping), each implemented as a separate class that defines its behavior.

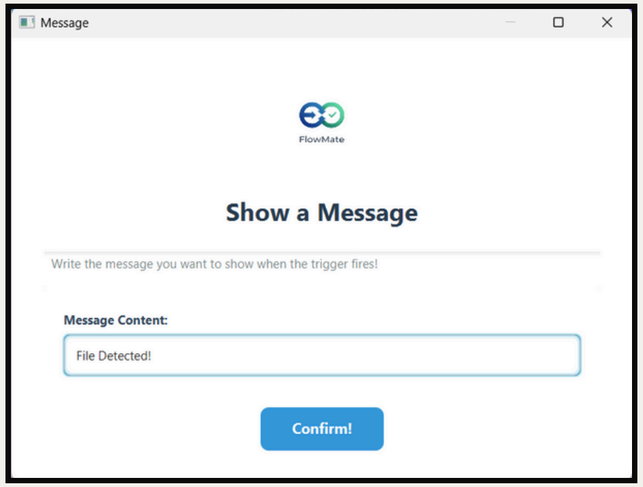
GUI Showcase

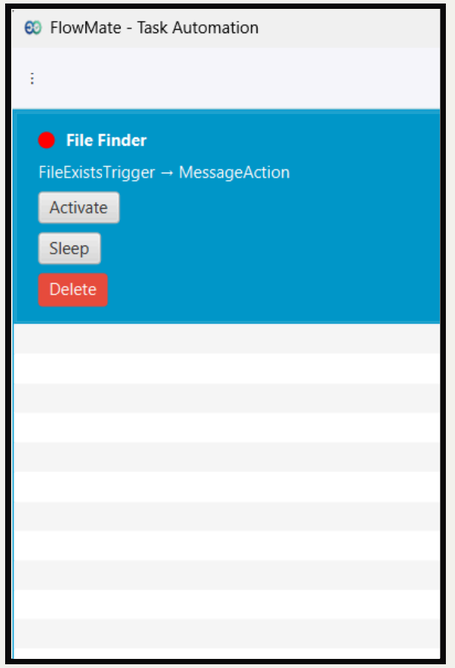


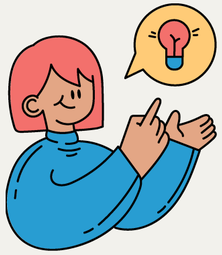


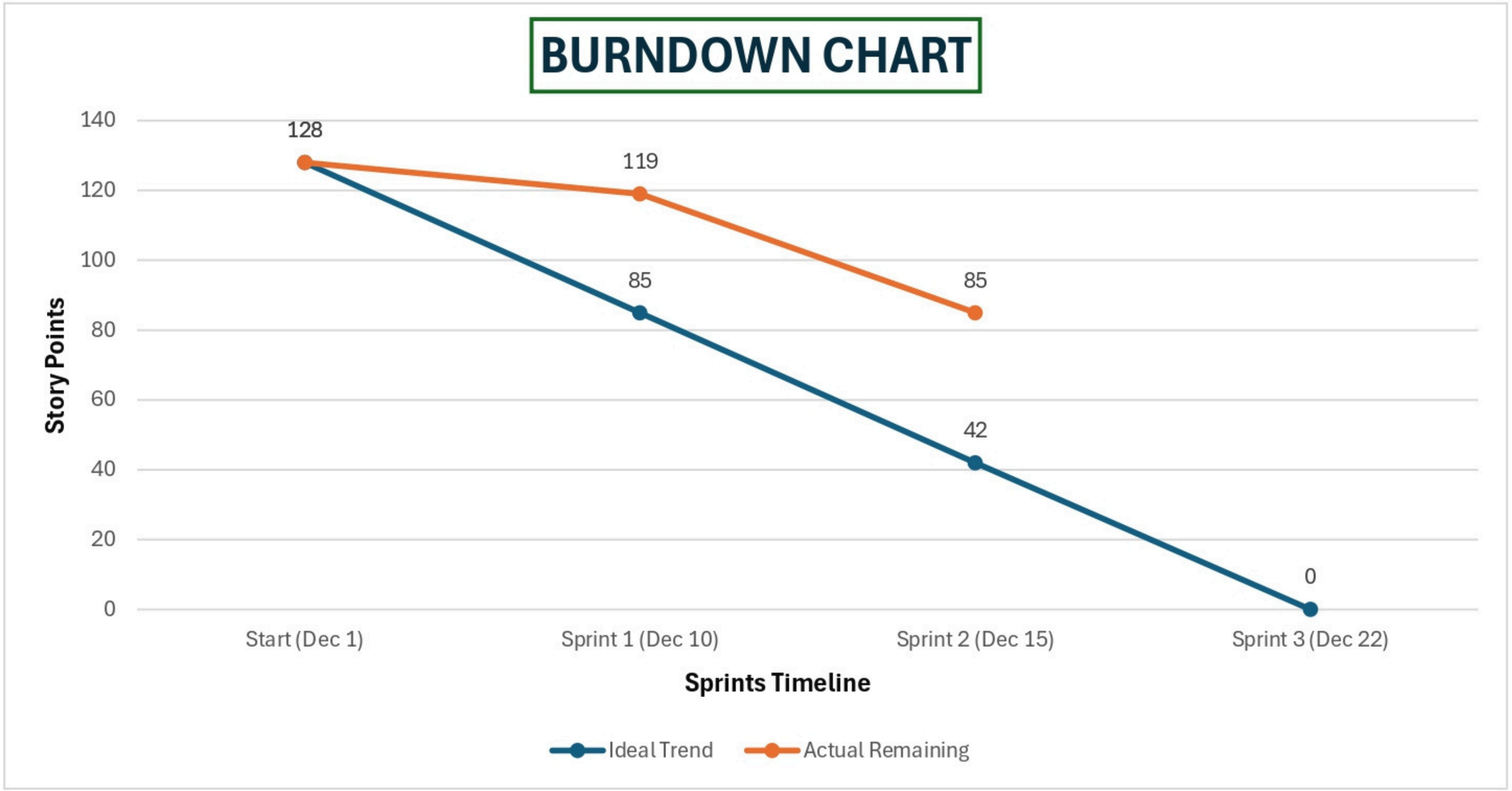






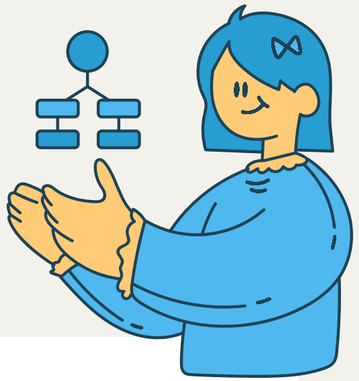





BURNDOWN CHART
140
120
100
80
60
40
20
0
Story Points
128
119
85
85
42
0
Start (Dec 1)
Sprint 1 (Dec 10)
Sprint 2 (Dec 15)
Sprint 3 (Dec 22)
Sprints Timeline
Ideal Trend
Actual Remaining

Updated Sprint Backlog

Sabatino Ester	Pecoraro Sara	Della Corte Alessio	Siddiqa Ayesha
8.2 - Update the GUI so that the user can handle the active/inactive state of a specific rule.	**8.1** - Implement the activation/de-activation logic and integrate it in the application.	**9.1** - Implement a "SaveOnFile" method.	**9.2** - Implement a "LoadFromFile" method that is automatically executed when the application is run.
10.1 - Implement the sleep logic and integrate it in the application.	**8.3** - Write Unit Tests for the rule state behaviour.	**10.2** - Update the GUI so that the user can handle the sleeping state of a rule.	**9.3** - Write Unit Tests for the “SaveOnFile” and “LoadFromFile” methods.
10.3 - Write Unit Tests for the sleep/awake behaviour of a rule.	**11.2** - Implement a GUI that allows the user to specify the path of the file on which he wants to write, and the text he wants to add to that file.	**12.1** - Implement a FileExistsTrigger class.	**12.2** - Implement a GUI that allows the user to specify the path of a folder and the name of a file to check if it exists in that folder.
11.1 - Implement a TextAction class.	**11.3** - Implement a Controller for the WriteOnFileView.	**12.4** - Write Unit Tests for the FileExistsTrigger Class.	**12.3** - Implement a Controller for the SelectAFileExistent View.
11.4 - Write Unit Tests for the TextAction class.	**13.1** - Implement a CopyFileAction class.	**13.9** - Implement a DeleteFileAction class.	**13.5** - Implement a MoveFileAction class.
	13.2 - Implement a GUI that allows the user to select a file and specify in which directory he wants to copy it in	**13.10** - Implement a GUI that allows the user to select the file he wants to delete.	**13.6** - Implement a GUI that allows the user to select a file and specify in which directory he wants to move it in.
	13.3 - Implement a Controller for the CopyFileView.	**13.11** - Implement a Controller for the DeleteFileView.	**13.7** - Implement a Controller for the MoveFileView.
	13.4 - Write Unit Tests for the CopyFileAction class.	**13.12** - Write Unit Tests for the DeleteFileAction class.	**13.8** - Write Unit Tests for the MoveFileAction class.

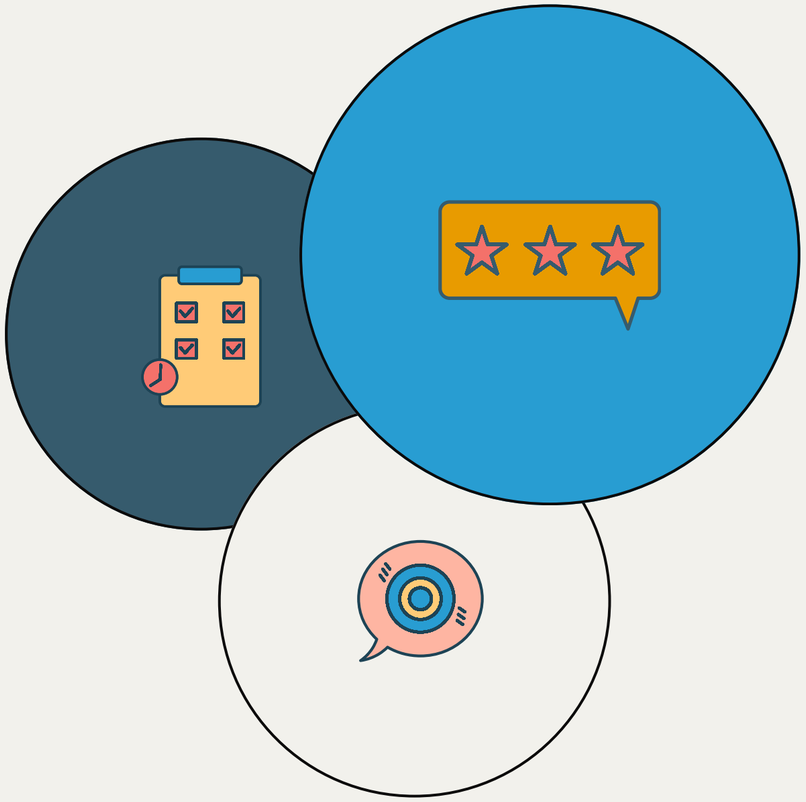


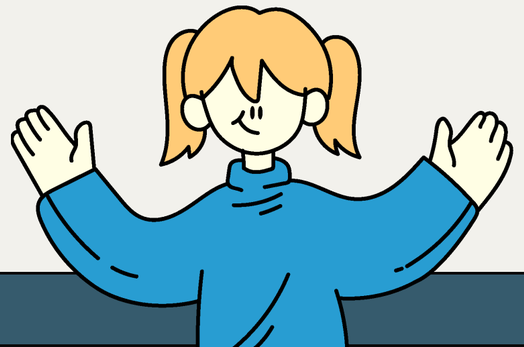
COLOUR LEGEND

COMPLETED	CARRIED OVER	NOT STARTED YET

3rd Sprint Backlog

- US14: External Program
- US15: Day Trigger
- US16: Month Trigger
- US17: Date Trigger
- US20: Manage Counters (Define, View, Change)
- US21: Add Counter to Counter Action
- US22: Counter Comparison





Next Sprint Preview

Advanced Logic

Implement precise time-based scheduling and variable counters to enable complex, state-aware automation.

System Extension

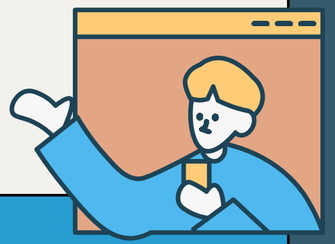
Integrate external program execution and composite actions to expand the application's control capabilities..

User Empowerment

Provide users with granular control over rules and variables, significantly increasing the application's flexibility

Final Delivery

Prioritize rigorous integration testing and UI polish to ensure a stable, production-ready release.



Thank you for your attention!