FlowMate

First Sprint Review

Inspect, adapt, deliver: our Sprint 1 at a glance.

Presented by

Team 5		
Della Corte Alessio	IE232000	a.dellacorte23@studenti.unisa.it
Pecoraro Sara	IE23200027	s.pecoraro19@studenti.unisa.it
Sabatino Ester	IE232000	e.sabatino14@studenti.unisa.it
Siddiqa Ayesha	IE23200046	a.siddiqa1@studenti.unisa.it

Sprint Recap

What We Delivered

We implemented the rule creation and period check process. We also implemented the time trigger, message and audio reproduction. In addition, the basic GUI was created.

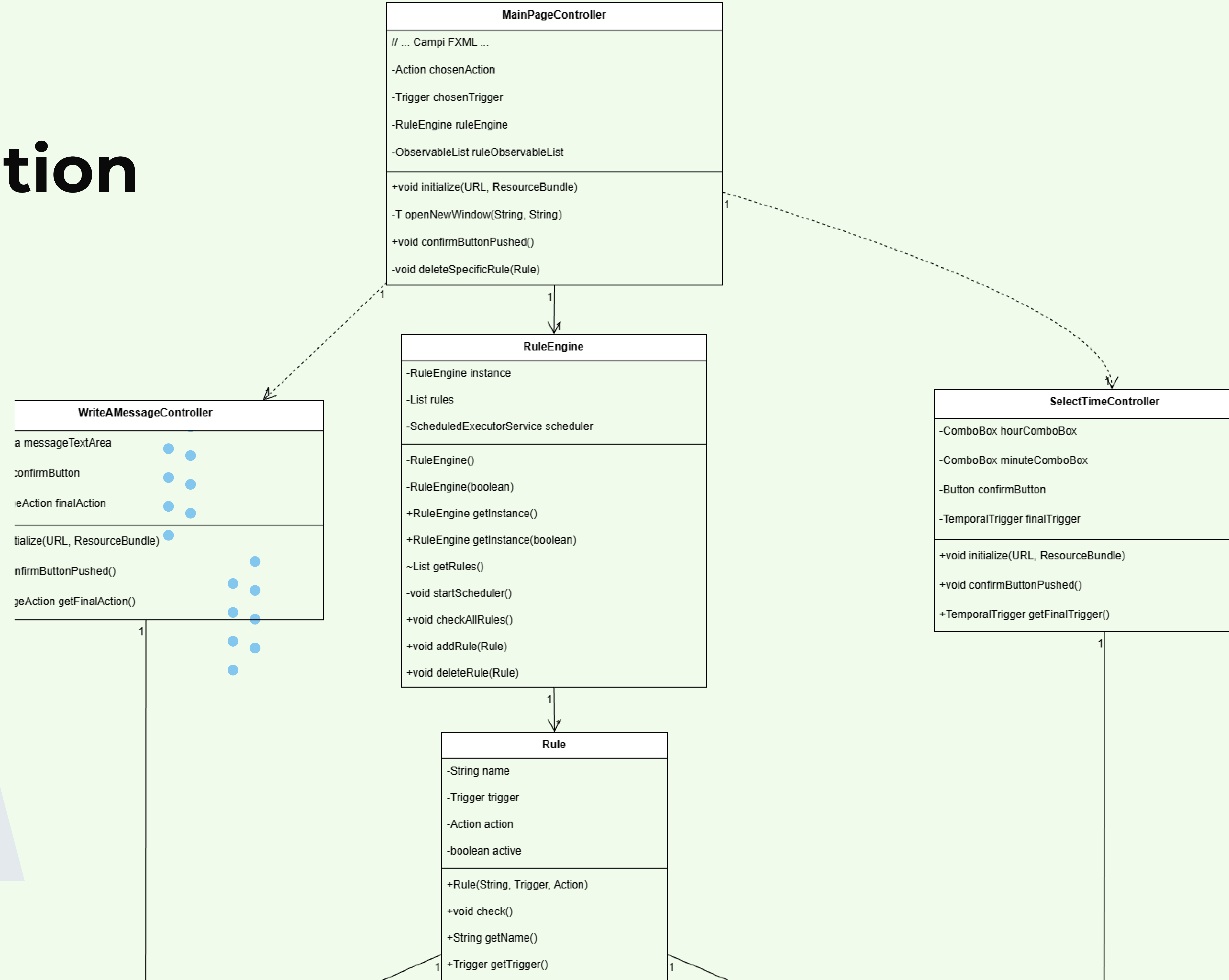
Original Commitments

Our original goal was to complete all the task related to the first 8 User Stories.

Velocity Achieved

We managed to achieve a velocity of 11 story points.

Implementation Achieved



MainPageController

// ... Campi FXML ...

-Action chosenAction

-Trigger chosenTrigger

-RuleEngine ruleEngine

-ObservableList ruleObservableList

+void initialize(URL, ResourceBundle)

-T openNewWindow(String, String)

+void confirmButtonPushed()

-void deleteSpecificRule(Rule)

WriteAMessageController

a messageTextArea

onfirmButton

eAction finalAction

tialize(URL, ResourceBundle)

nfirmButtonPushed()

geAction getFinalAction()

RuleEngine

-RuleEngine instance

-List rules

-ScheduledExecutorService scheduler

-RuleEngine()

-RuleEngine(boolean)

+RuleEngine getInstance()

+RuleEngine getInstance(boolean)

~List getRules()

-void startScheduler()

+void checkAllRules()

+void addRule(Rule)

+void deleteRule(Rule)

SelectTimeController

-ComboBox hourComboBox

-ComboBox minuteComboBox

-Button confirmButton

-TemporalTrigger finalTrigger

+void initialize(URL, ResourceBundle)

+void confirmButtonPushed()

+TemporalTrigger getFinalTrigger()

Rule

-String name

-Trigger trigger

-Action action

-boolean active

+Rule(String, Trigger, Action)

+void check()

+String getName()

+Trigger getTrigger()

What Went Well

Successful Collaborations

We're highlighting where teamwork really clicked. Good partnerships should be repeated.

Process Improvements

Despite initial problems, the team managed to achieve a great amount of the User Stories we predicted.

Technical Wins

We managed to find an IDE that fitted our needs.

Stakeholder Engagement

We managed to walk through our problems cooperating and orginizing our work.

What Didn't Work

✓

Task Division and Definition

We first divided tasks without a precise logic, so the code implementation had many dependencies between team members and didn’t cover all the necessary aspects

✓

Version collision

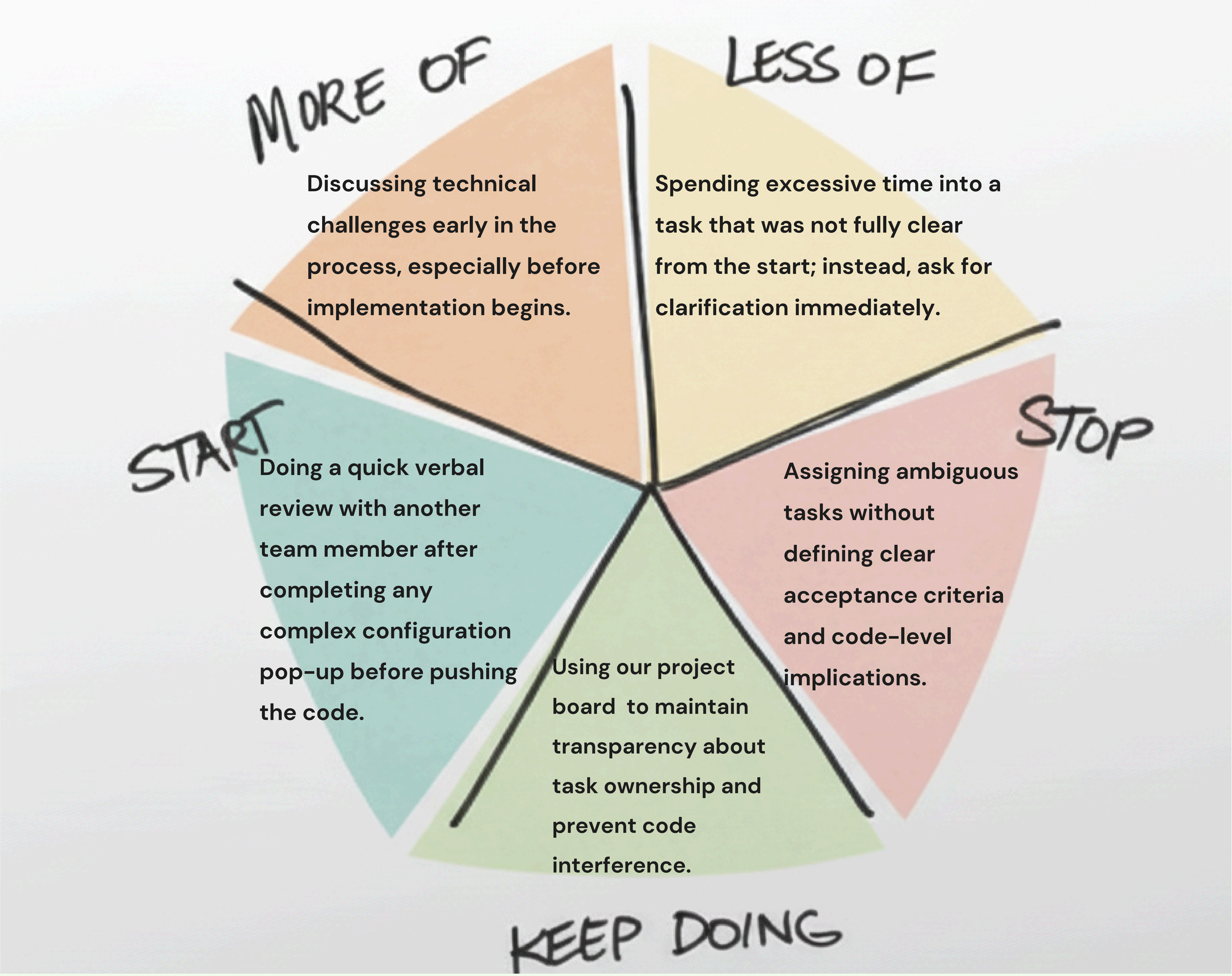
We encountered problems making different versions of NetBeans, JDK, JavaFX and JUnit all together

✓

Repository problems

We first had problems with the repository connection and code pushing

Sprint retrospective: Starfish diagram



Discussing technical challenges early in the process, especially before implementation begins.

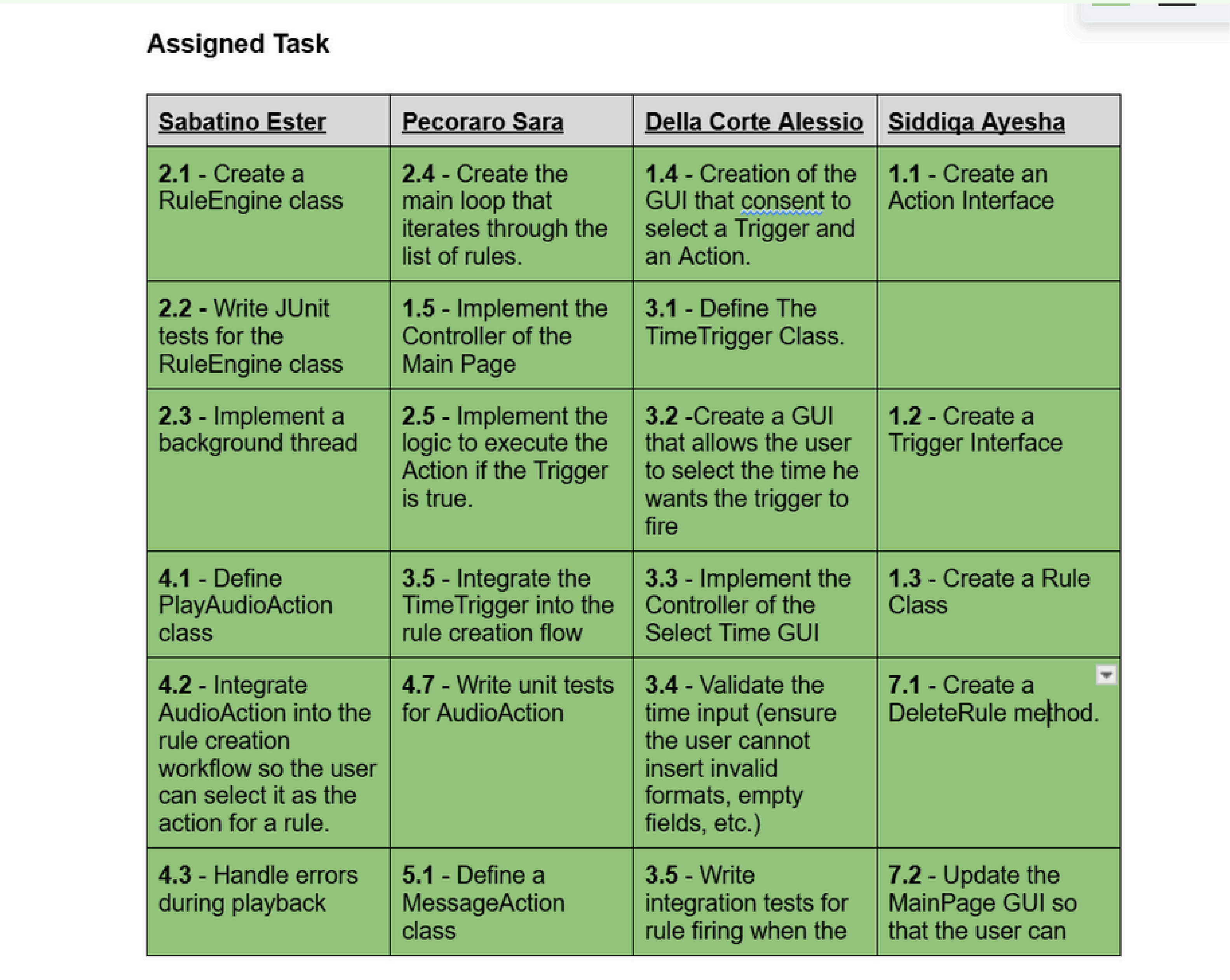
Spending excessive time into a task that was not fully clear from the start; instead, ask for clarification immediately.

Doing a quick verbal review with another team member after completing any complex configuration pop-up before pushing the code.

Assigning ambiguous tasks without defining clear acceptance criteria and code-level implications.

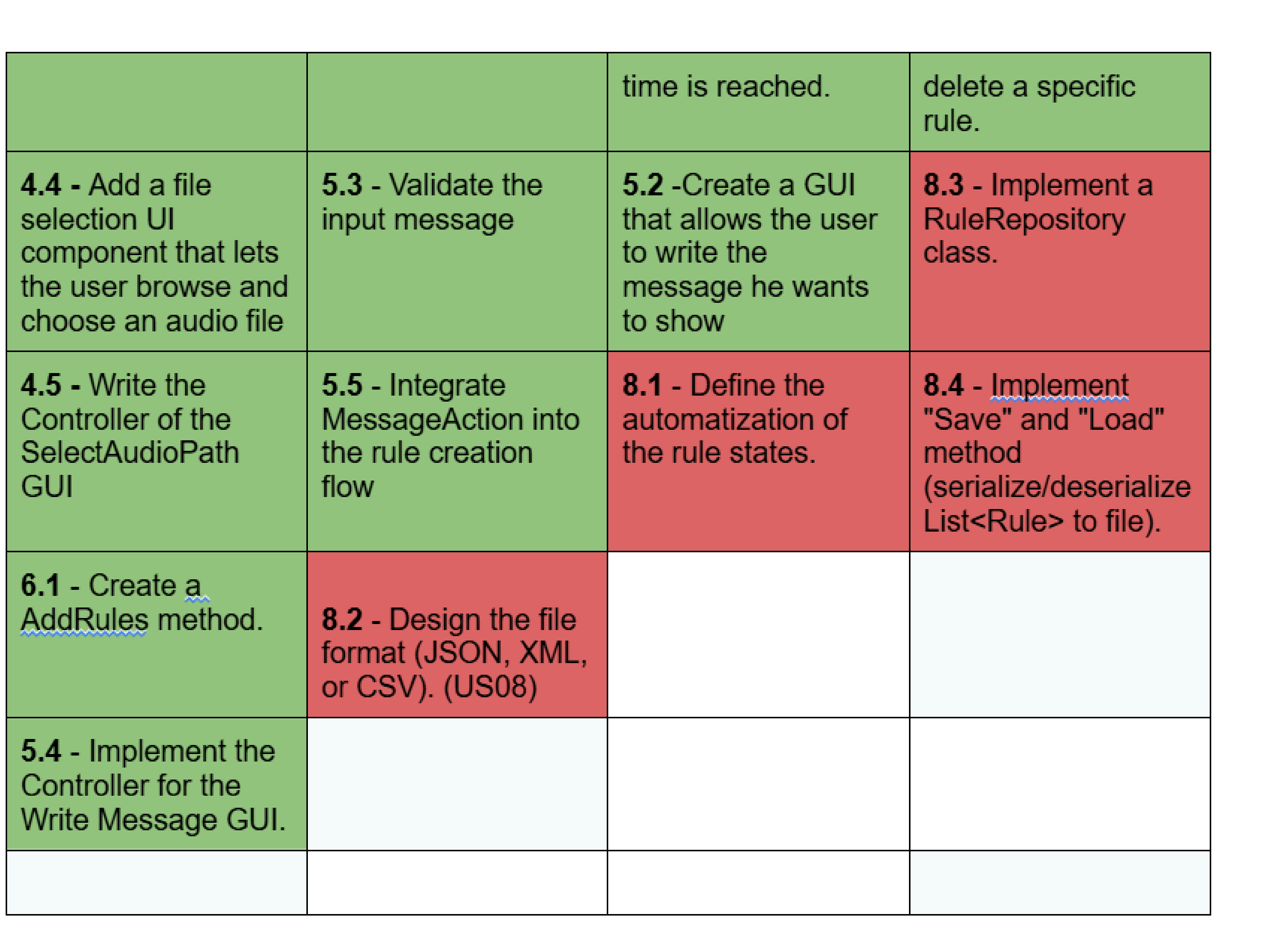
Using our project board to maintain transparency about task ownership and prevent code interference.

Updated Sprint Backlog

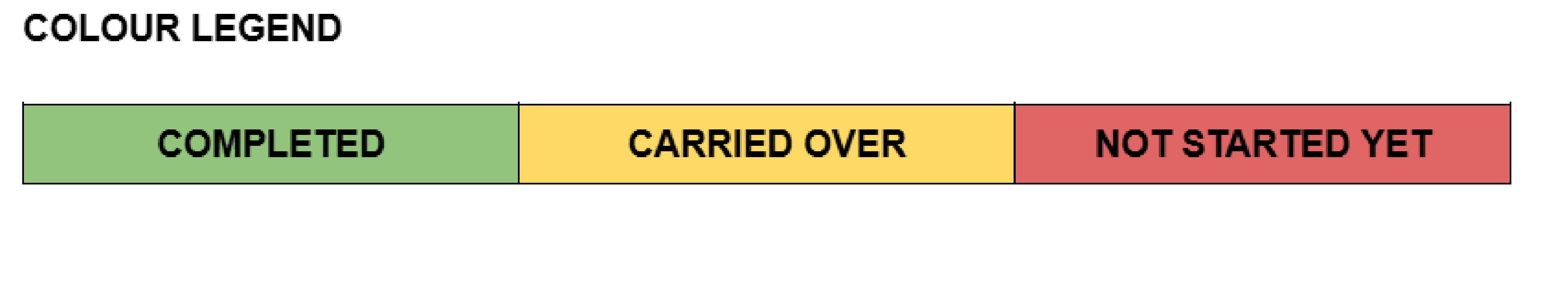


Assigned Task

Sabatino Ester	Pecoraro Sara	Della Corte Alessio	Siddiqa Ayesha
2.1 - Create a RuleEngine class	**2.4** - Create the main loop that iterates through the list of rules.	**1.4** - Creation of the GUI that consent to select a Trigger and an Action.	**1.1** - Create an Action Interface
2.2 - Write JUnit tests for the RuleEngine class	**1.5** - Implement the Controller of the Main Page	**3.1** - Define The TimeTrigger Class.	
2.3 - Implement a background thread	**2.5** - Implement the logic to execute the Action if the Trigger is true.	**3.2** -Create a GUI that allows the user to select the time he wants the trigger to fire	**1.2** - Create a Trigger Interface
4.1 - Define PlayAudioAction class	**3.5** - Integrate the TimeTrigger into the rule creation flow	**3.3** - Implement the Controller of the Select Time GUI	**1.3** - Create a Rule Class
4.2 - Integrate AudioAction into the rule creation workflow so the user can select it as the action for a rule.	**4.7** - Write unit tests for AudioAction	**3.4** - Validate the time input (ensure the user cannot insert invalid formats, empty fields, etc.)	**7.1** - Create a DeleteRule method.
4.3 - Handle errors during playback	**5.1** - Define a MessageAction class	**3.5** - Write integration tests for rule firing when the	**7.2** - Update the MainPage GUI so that the user can



		time is reached.	delete a specific rule.
4.4 - Add a file selection UI component that lets the user browse and choose an audio file	**5.3** - Validate the input message	**5.2** -Create a GUI that allows the user to write the message he wants to show	**8.3** - Implement a RuleRepository class.
4.5 - Write the Controller of the SelectAudioPath GUI	**5.5** - Integrate MessageAction into the rule creation flow	**8.1** - Define the automatization of the rule states.	**8.4** - Implement "Save" and "Load" method (serialize/deserialize List<Rule> to file).
6.1 - Create a AddRules method.	**8.2** - Design the file format (JSON, XML, or CSV). (US08)		
5.4 - Implement the Controller for the Write Message GUI.			



COLOUR LEGEND

🟩 COMPLETED	🟨 CARRIED OVER	🟥 NOT STARTED YET

Creational Singleton Pattern

Pattern definition

The Singleton is a creational design pattern that restricts a class to a single unique instance, ensuring a global point of access to it throughout the application lifecycle.

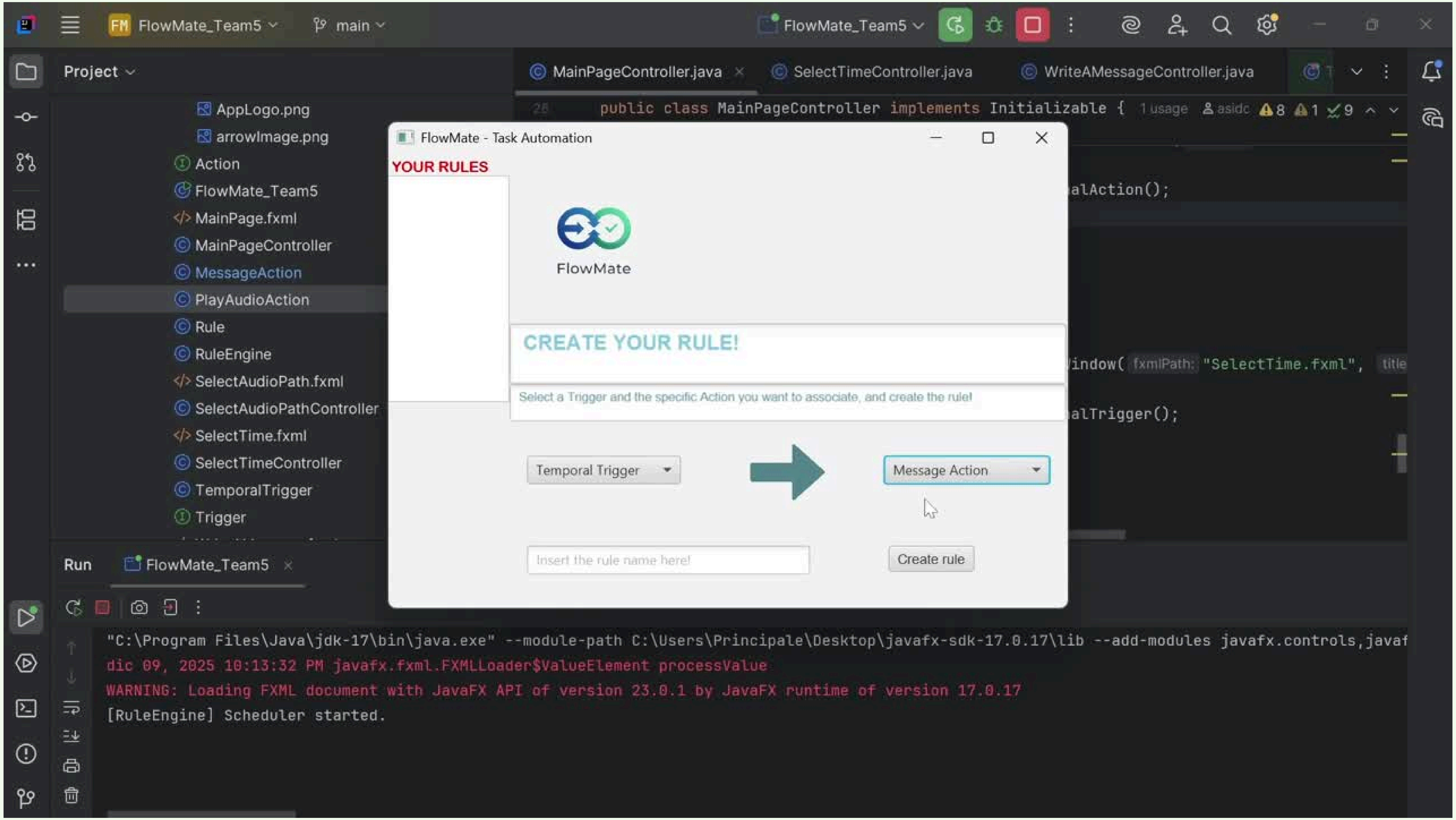
Our application

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.

Good design principles

Enforces strict resource optimization by managing a single background thread. Guarantees data consistency, ensuring that the rules defined and the rules checked by the executor always reside in the same shared state.

Product Demo



```
public class MainPageController implements Initializable {
alAction();
indow( fxmlPath: "SelectTime.fxml",  title
alTrigger();
```

YOUR RULES

CREATE YOUR RULE!

Select a Trigger and the specific Action you want to associate, and create the rule!

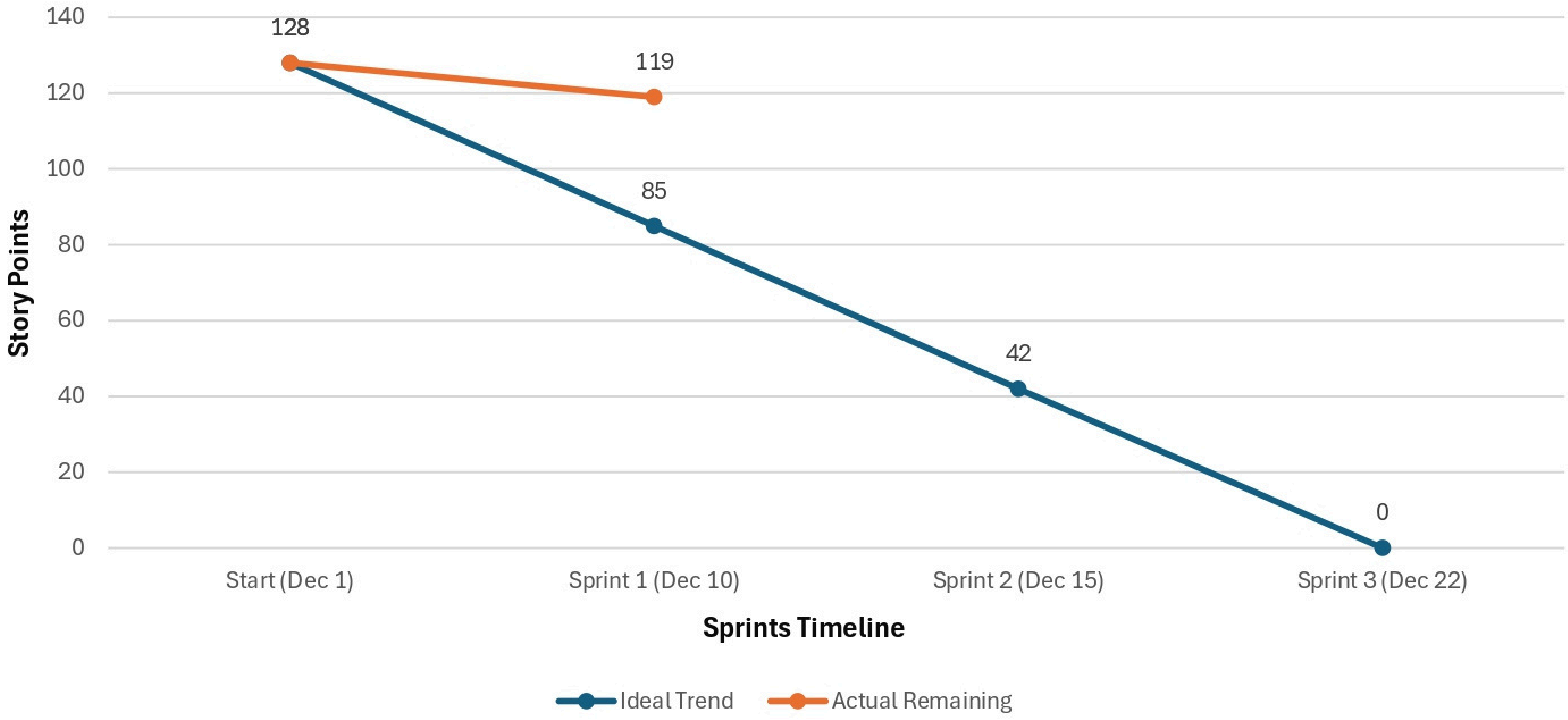
Temporal Trigger → Message Action

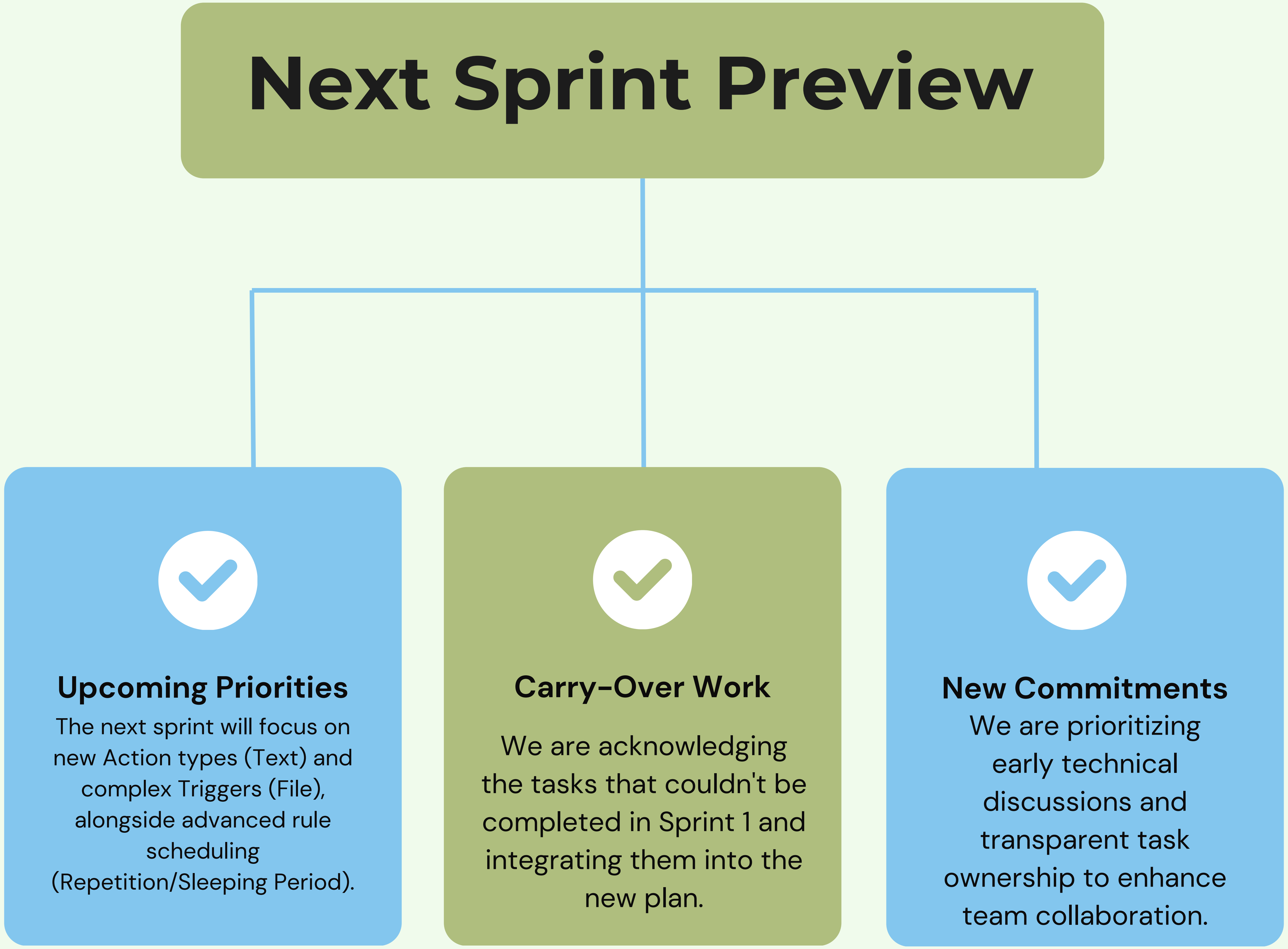
Insert the rule name here!

Create rule

```
"C:\Program Files\Java\jdk-17\bin\java.exe" --module-path C:\Users\Principale\Desktop\javafx-sdk-17.0.17\lib --add-modules javafx.controls,javaf
dic 09, 2025 10:13:32 PM javafx.fxml.FXMLLoader$ValueElement processValue
WARNING: Loading FXML document with JavaFX API of version 23.0.1 by JavaFX runtime of version 17.0.17
[RuleEngine] Scheduler started.
```

SPRINT BURNDOWN CHART





Next Sprint Preview

Upcoming Priorities

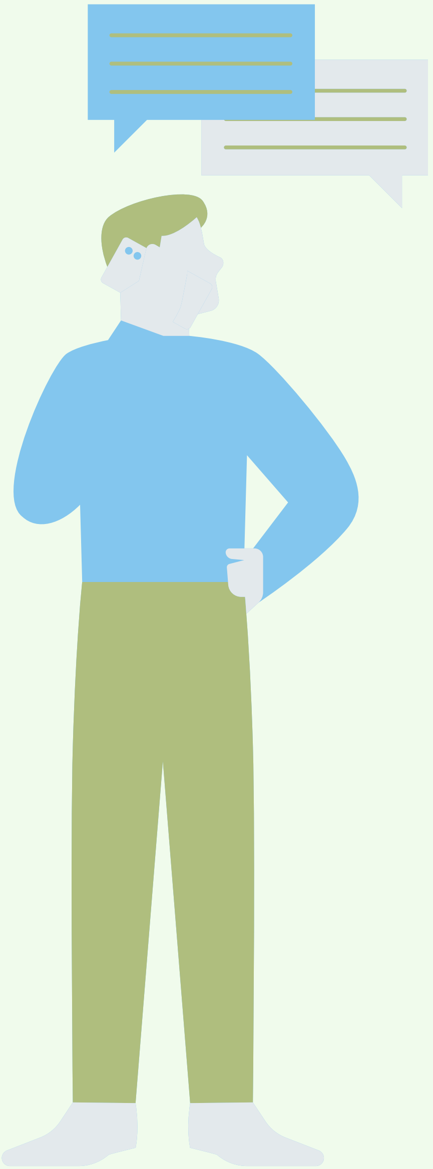
The next sprint will focus on new Action types (Text) and complex Triggers (File), alongside advanced rule scheduling (Repetition/Sleeping Period).

Carry-Over Work

We are acknowledging the tasks that couldn't be completed in Sprint 1 and integrating them into the new plan.

New Commitments

We are prioritizing early technical discussions and transparent task ownership to enhance team collaboration.



2nd Sprint backlog

- US08: Rule State(Active/Inactive)
- US09: Persistence of the rules
- US10: Repetition and Sleeping Period
- US11: Text Action
- US12: File Trigger
- US13: File Actions

Total Story Points: 34 Story Points

THANK YOU FOR THE ATTENTION!