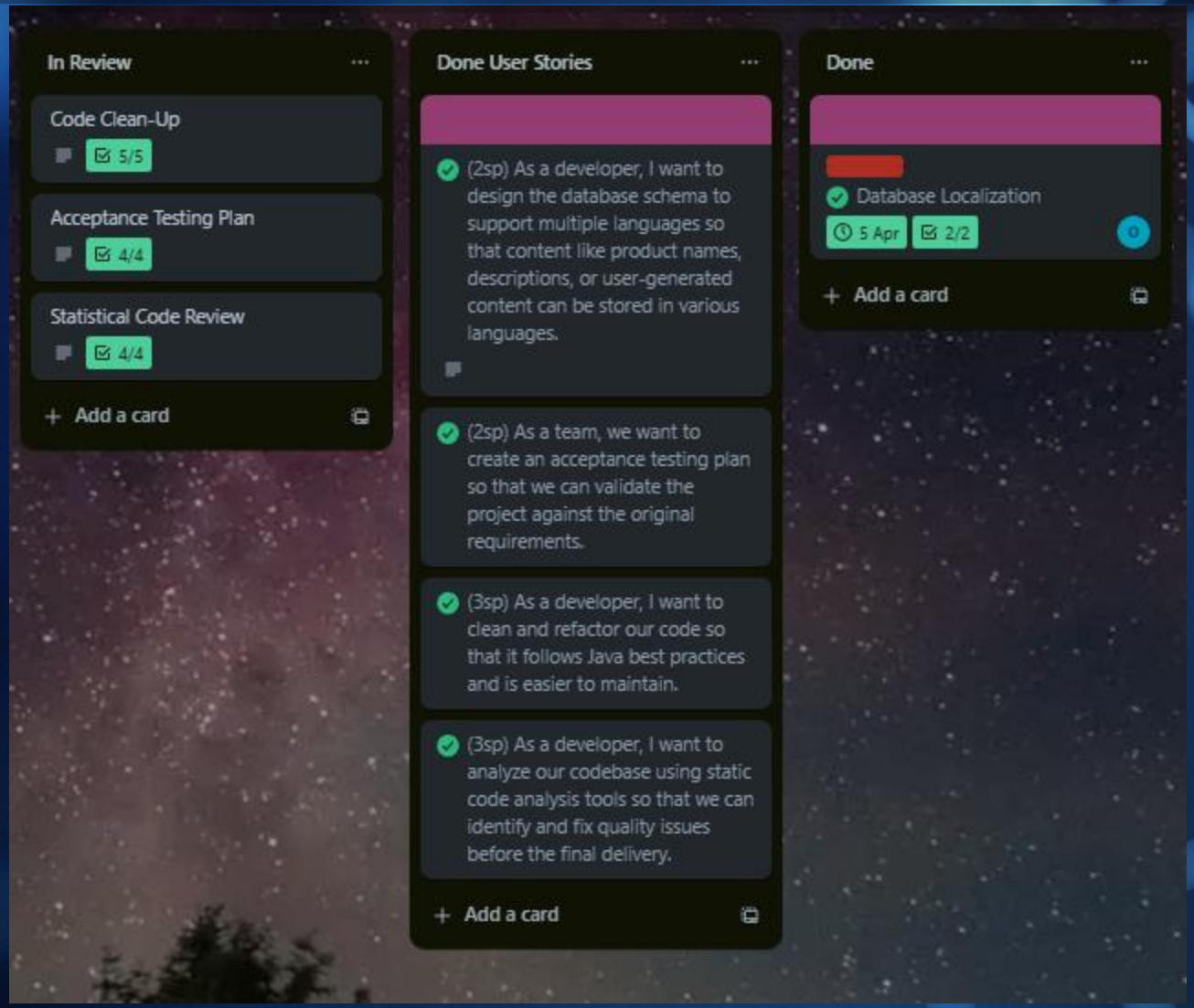
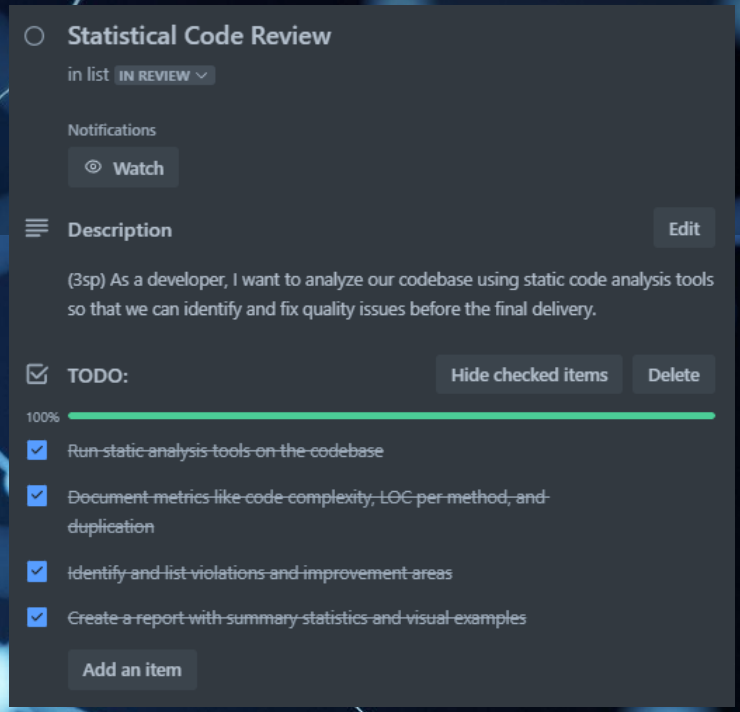
Sprint 6

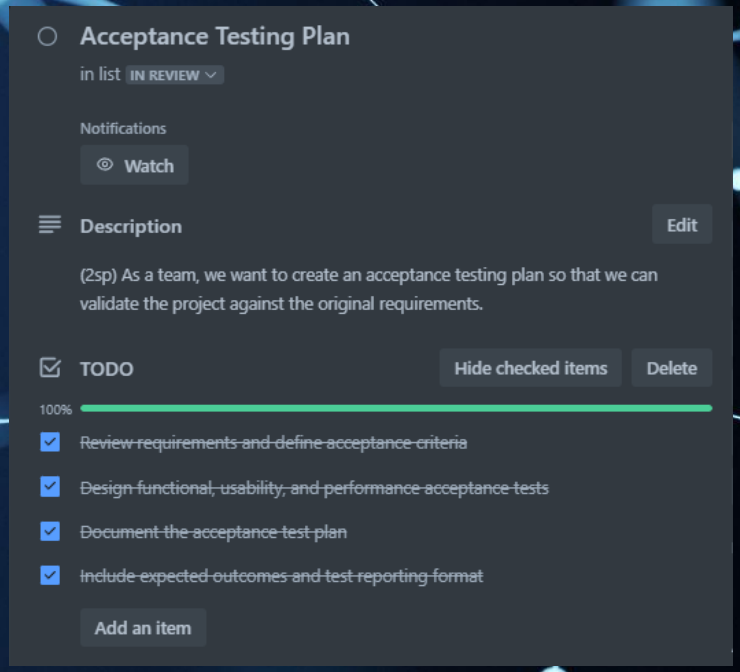
Ryhmä: Eetu O & Oleg I

Scrum master: Eetu O

Trello


In Review
Code Clean-Up
5/5
Acceptance Testing Plan
4/4
Statistical Code Review
4/4
Add a card
Done User Stories
(2sp) As a developer, I want to design the database schema to support multiple languages so that content like product names, descriptions, or user-generated content can be stored in various languages.
(2sp) As a team, we want to create an acceptance testing plan so that we can validate the project against the original requirements.
(3sp) As a developer, I want to clean and refactor our code so that it follows Java best practices and is easier to maintain.
(3sp) As a developer, I want to analyze our codebase using static code analysis tools so that we can identify and fix quality issues before the final delivery.
Add a card
Done
Database Localization
5 Apr
2/2
Add a card


Statistical Code Review
in list IN REVIEW
Notifications
Watch
Description
Edit
(3sp) As a developer, I want to analyze our codebase using static code analysis tools so that we can identify and fix quality issues before the final delivery.
TODO:
Hide checked items
Delete
100%
Run static analysis tools on the codebase
Document metrics like code complexity, LOC per method, and duplication
Identify and list violations and improvement areas
Create a report with summary statistics and visual examples
Add an item


Acceptance Testing Plan
in list IN REVIEW
Notifications
Watch
Description
Edit
(2sp) As a team, we want to create an acceptance testing plan so that we can validate the project against the original requirements.
TODO
Hide checked items
Delete
100%
Review requirements and define acceptance criteria
Design functional, usability, and performance acceptance tests
Document the acceptance test plan
Include expected outcomes and test reporting format
Add an item

Work Excel

Average käytetty aika:

n. 14h

6	3	inclass week3
6	2	google api + product db table localized
6	1.5	product view + add/edit buttons
6	2	lecture online
6	2	download code review tool
6	1	preparing code review report 1
6	2	refactoring part 1

6	2	Sprint planning
6	2	Fixing dashboard bugs
6	1	adding missing translations
6	1	Fixing form translation bug
6	2	lecture
6	2	refactoring + javadoc
6	1	Splting functions into utilities
6	2	Sonarqube check

Statistical Code Review Before Cleanup

Note: This report was generated before code refactoring and test improvements. All findings and metrics reflect the current state of the codebase prior to clean-up actions and acceptance test implementation. The upcoming refactoring will address the identified issues such as high complexity or code duplication.

Static Analysis Overview (SonarQube)

Metric	Value	Notes
Lines of Code	5,462	Medium-sized codebase
Statements	2,854	Logic-heavy project
Functions	464	Average ~12 lines per function
Classes / Files	64 / 65	Good class-to-file structure
Comments (%)	11.1%	Satisfactory documentation level, will be higher as we will add documentation
Cyclomatic Complexity	710	Quite high — requires refactoring in specific files
Cognitive Complexity	276	Acceptable, the lower the better
Duplicated Lines	1,022 (12.9%)	High duplication — must be reduced
Code Coverage	19.8%	Low but OK — GUI and view modules are not covered as they are JavaFX based
Security Issues	0	Clean and safe

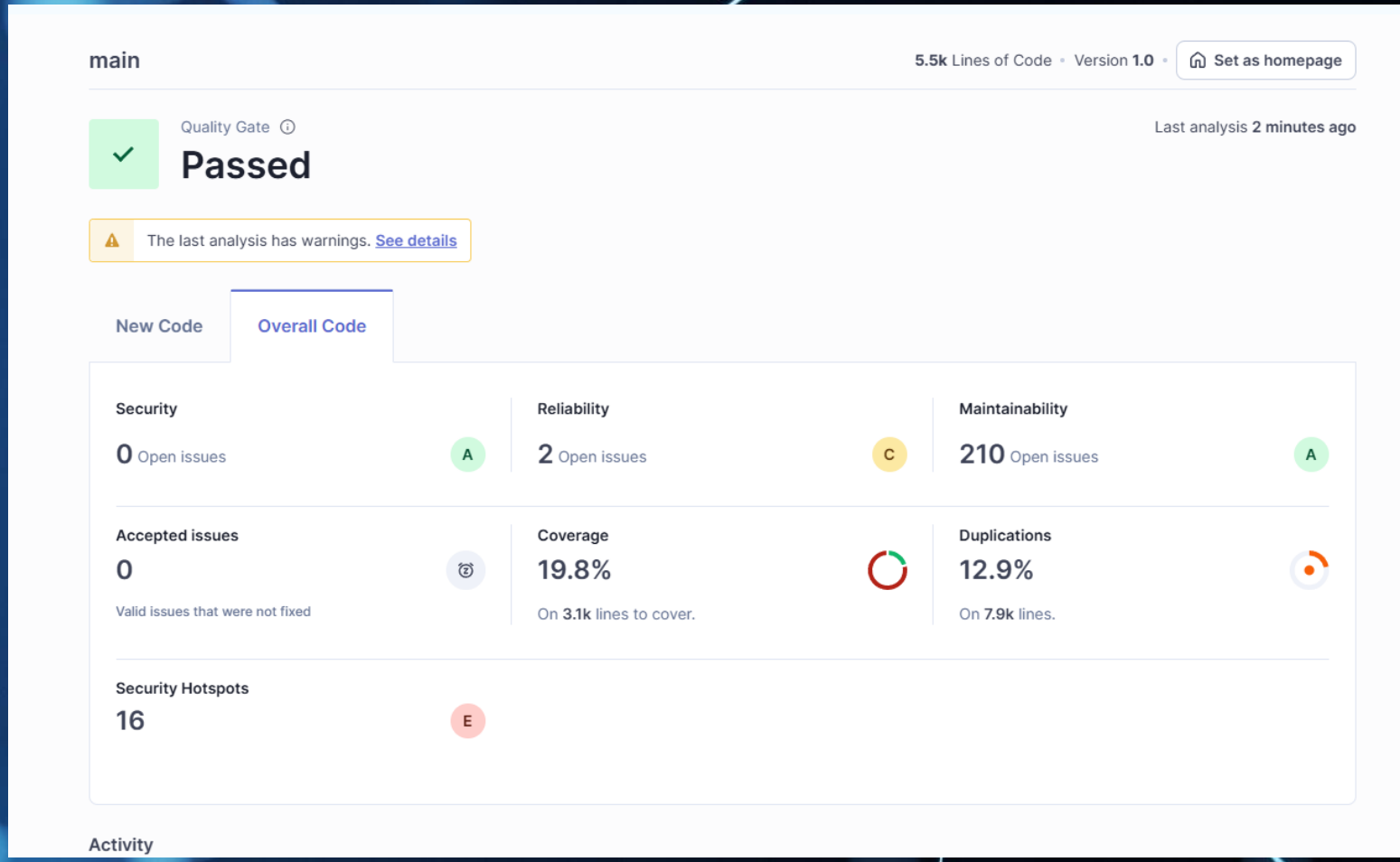
Complexity Hotspots

Cyclomatic Complexity (Top 5 Files)

File	Score
DashboardController.java	78
Product.java	33
EditWindowController.java	27
EntityController.java	23
StockController.java	23

Cognitive Complexity (Top 5 Files)

File	Score
DashboardController.java	54
ProductDAO.java	25
EditWindowController.java	23
SignupController.java	22
EntityController.java	18



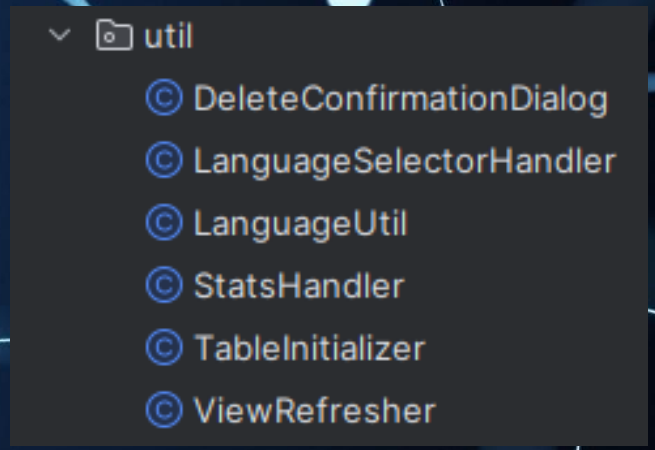
Code Cleanup

"Note: Javadoc comments were added to previously undocumented classes to improve code readability and understanding."

```
15      /**
16       * Persists a new customer to the database.
17       * @param customer the Customer object to be saved
18       */
19      public void addCustomer(Customer customer) {  13 usages  oiivantsov
20          EntityManager em = MariaDbJpaConnection.getInstance();
21          em.getTransaction().begin();
22          em.persist(customer);
23          em.getTransaction().commit();
24          em.close();
25      }
26
27      /**
28       * Retrieves all customers from the database.
29       * @return a list of all Customer objects
30       */
31      public List<Customer> getCustomers() {  oiivantsov
```

"Note: CheckStyle and Spotbugs were implemented, and reports can be found from the project's documents folder"

Note: Large classes, such as DashboardController, were refactored into smaller, reusable utility components to reduce code duplication and improve maintainability.



Acceptance Test Plan

Acceptance Criteria

- The user must be able to add and edit entities (e.g., products, orders).
- The application must persist data across sessions - we use DB.
- The UI must update live upon data changes - for example, language changing.
- Error messages must be clear and actionable.
- The system must not crash on invalid input or empty fields.

Acceptance Tests (To Be Conducted)

Functional Tests

Test Case	Expected Result
Add new product	Product appears in table
Edit existing entity	Changes persist and reflect in UI
Delete entity	Entity is removed from DB and UI

Usability Tests

- User is able to navigate the system without documentation.
- All buttons are visible and labelled clearly.
- Responsive layout tested on desktop and laptop.

Performance Tests

- Application loads in under 2-3 seconds.
- No noticeable lag when switching views or saving changes.