Habit Tracker

Desktop application for effective time management

Appendix

Aleksandra Wiktoria Bruska, 202400723
Cristina Semikina, 202400720
Jan Jakub Walczak, 202400722
David Fuchs, 202400725

Supervisor: Jørn Martin Hajek

Aarhus University
13 December 2024

~12000 characters

Process description

Collaboration agreement

This Collaboration Agreement outlines our shared principles and guidelines to foster effective, respectful, and collaborative teamwork, enabling us to achieve our project goals efficiently while maintaining transparency and strong communication.

Key Agreements and Values:

1. **Meetings and Collaboration**
    - Punctuality at all meetings is mandatory.
    - Meeting cancellations must be communicated with at least 24 hours' notice.
    - All members will focus solely on the shared project during meetings and collaboration sessions.
2. **Honesty and Transparency**

    Open and honest communication is fundamental for trust and effective collaboration.

3. **Balancing Effort with Boundaries**

    Each team member strives for their best performance while respecting personal boundaries and work-life balance.

4. **Timely Communication**

    Prompt responses to team communications, especially on WhatsApp, are expected to ensure smooth workflow and coordination.

5. **Efficient Meetings**
    - The Scrum Master will schedule all meetings before a sprint, draft the agenda for the next meeting, and send reminders.
    - Members are expected to be present, actively participate, and avoid distractions such as phone use during meetings.
6. **Ownership of Tasks and Organizational Duties**

    Roles such as note-taking, moderation, and communication are shared among team members to promote fair workload distribution.

7. **Open-Mindedness**

    We encourage and respect diverse perspectives, fostering creativity and openness to new ideas.

8. **Focus on Project Goals**

   Our primary emphasis is on achieving the Minimum Viable Product (MVP) through iterative progress, with regular updates and checkpoints.

9. **Tool Usage**

   - Notion is used for task management and project organization.
   - WhatsApp is used for general communication and quick updates.

10. **Version Control with Git**

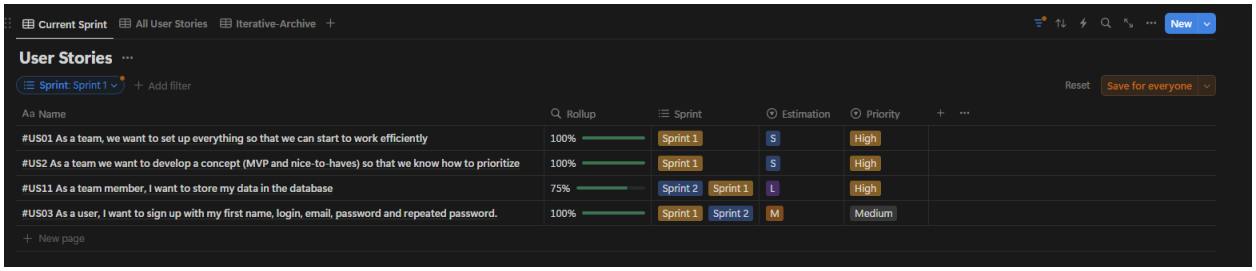
    - No direct pushes to the main branch; members must create branches and submit merge requests that require testing and approval by another member. Self-approval of merge requests is prohibited.
    - Keep branches up-to-date and avoid prolonged usage to minimize merge conflicts.
    - Use Git for code only; other documentation and tasks are managed on Notion unless otherwise discussed.

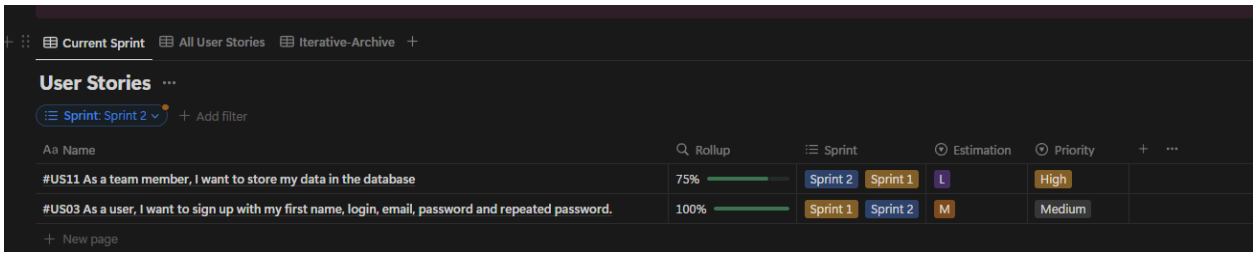
Acknowledgment

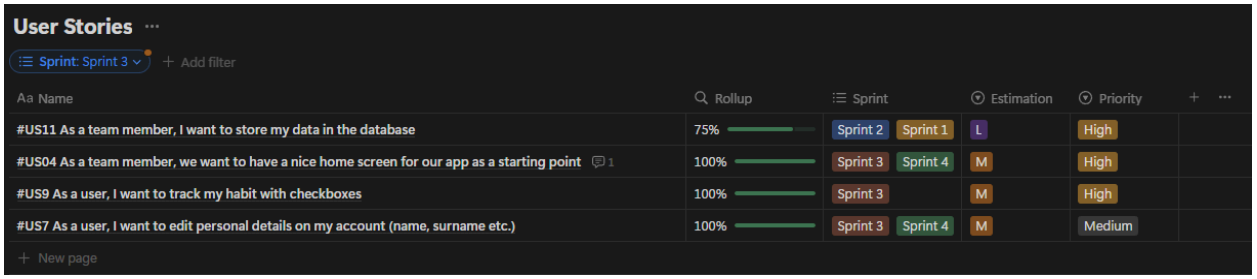
By signing, all team members agree to adhere to these guidelines, fostering a collaborative, respectful, and goal-driven work environment.

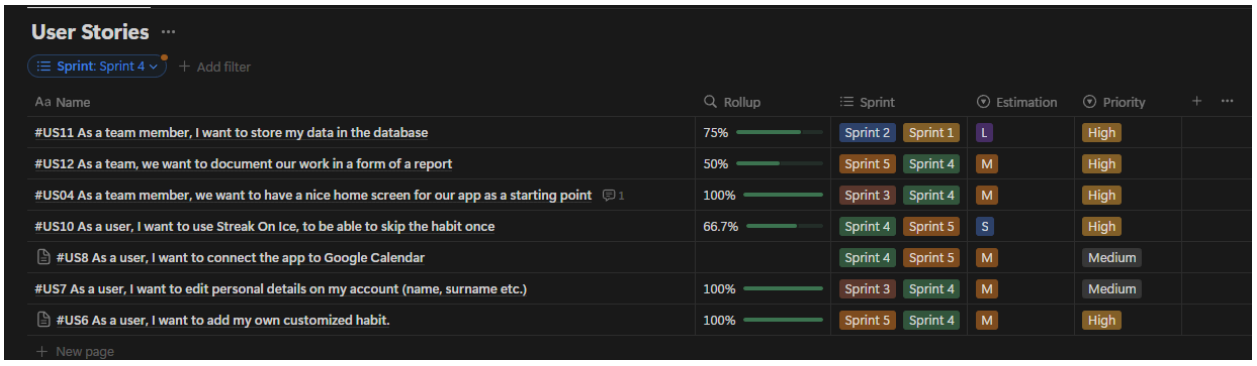
David Fuchs, Jan Jakub Walczak, Aleksandra Wiktoria Bruska, Cristina Semikina

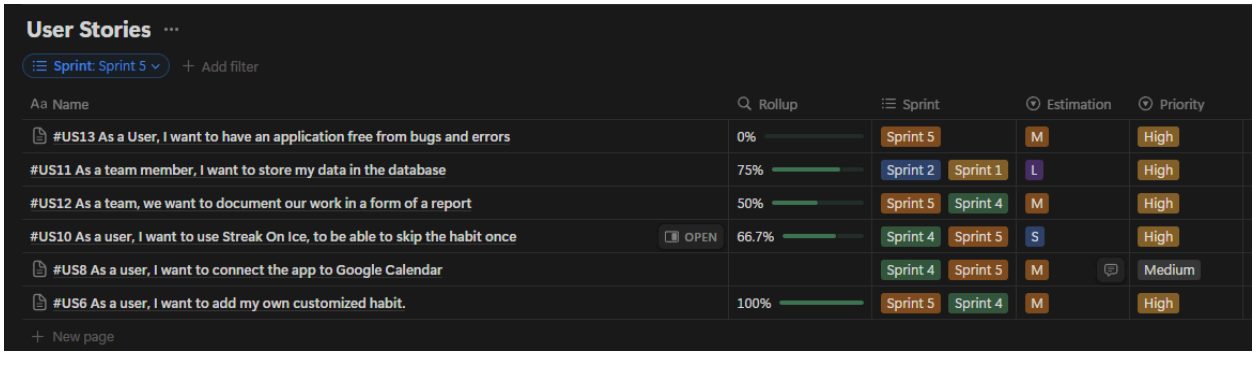
Development process

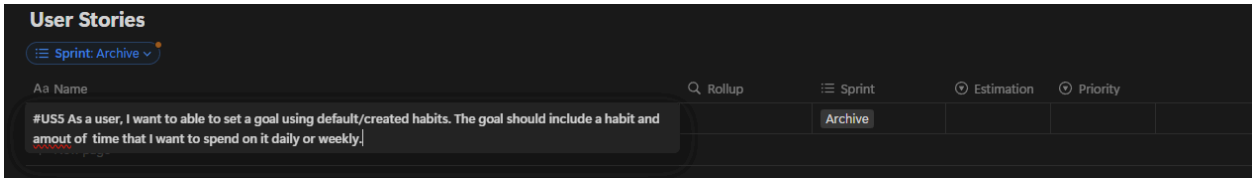


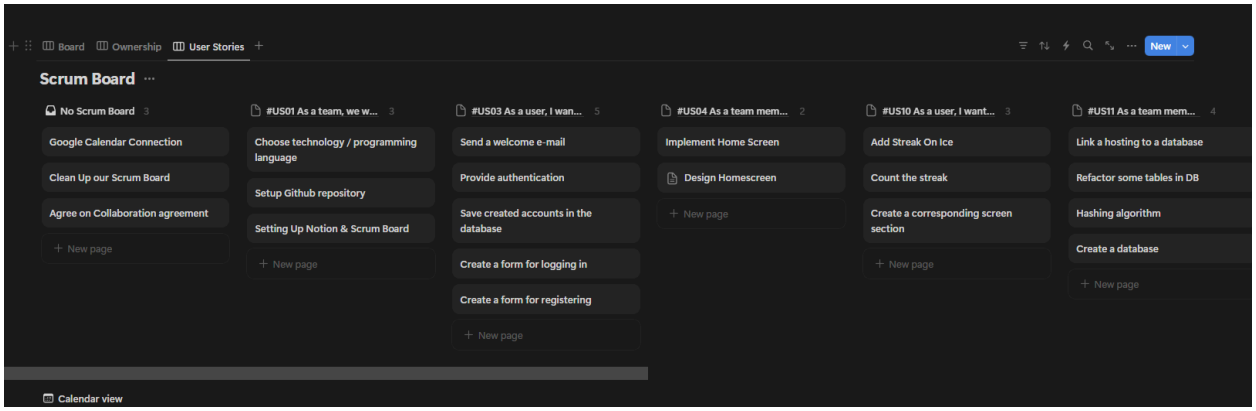


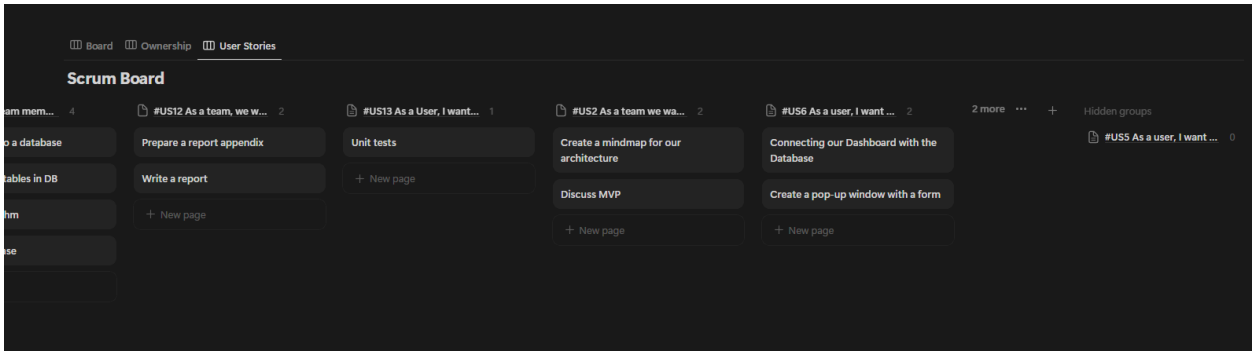




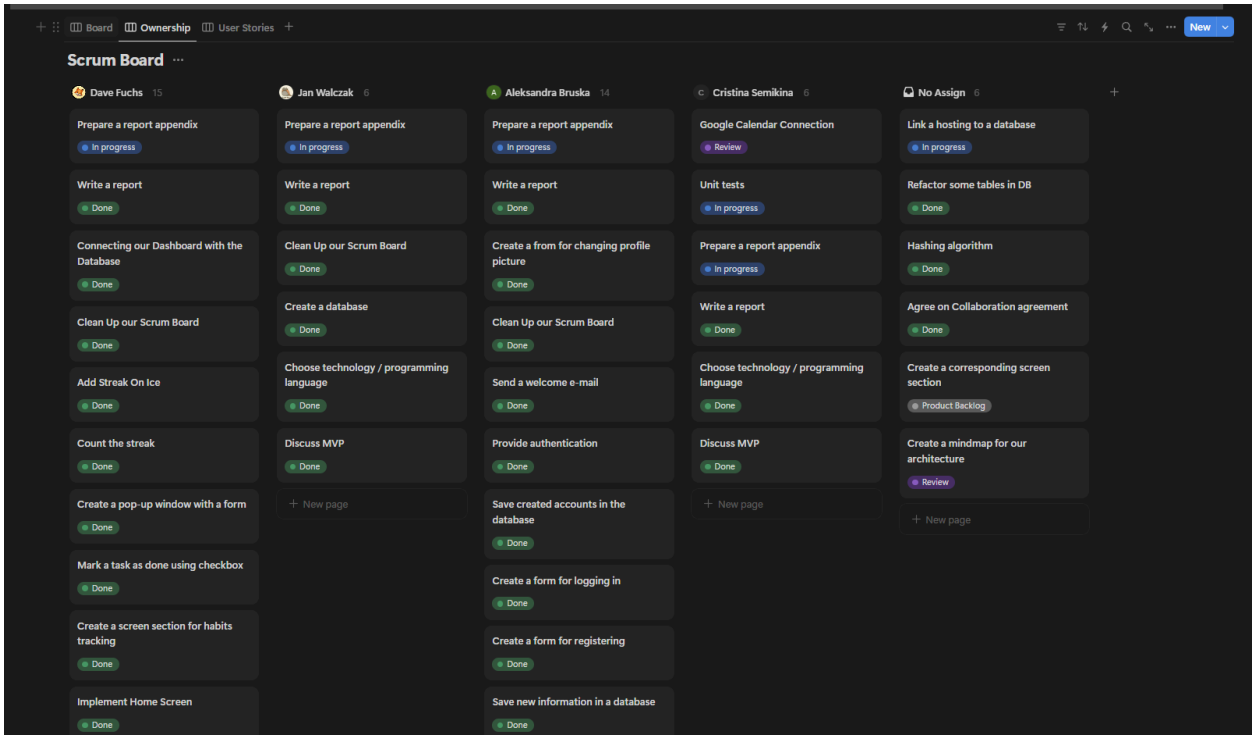


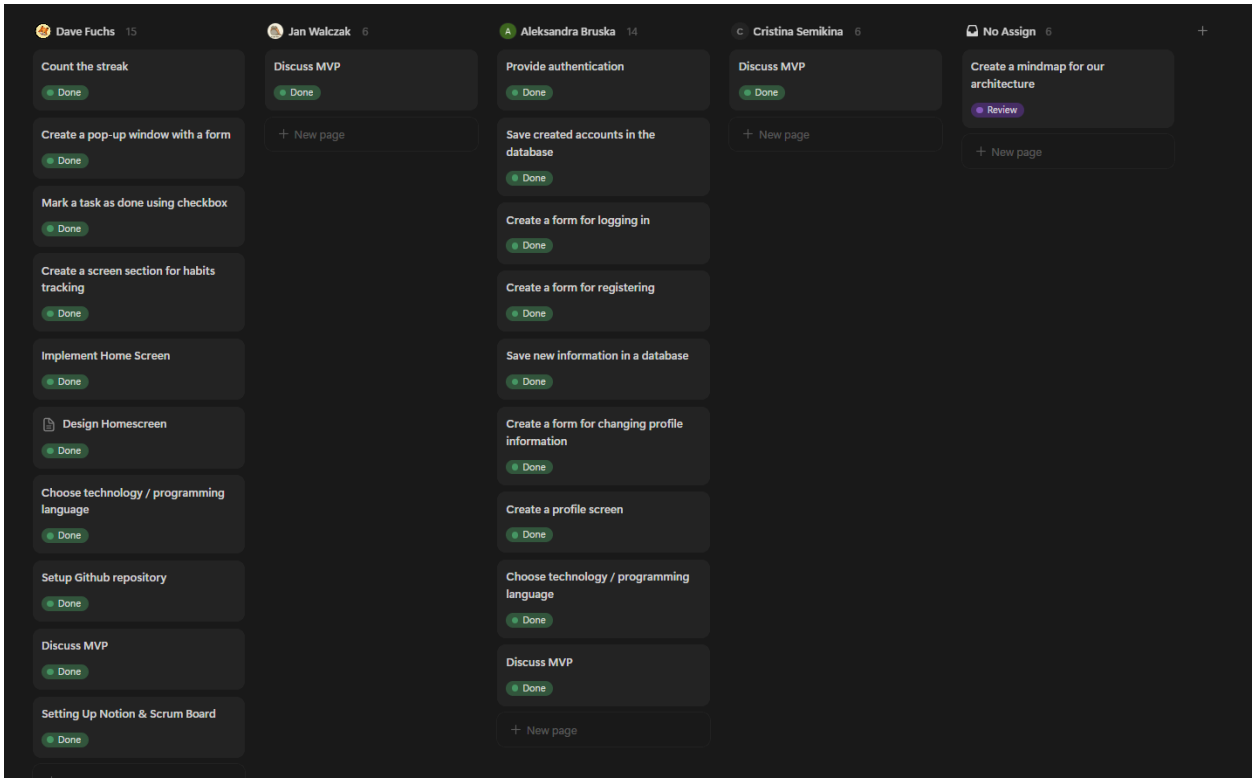

User Stories
Sprint: Archive
Aa Name
Rollup
Sprint
Estimation
Priority
#US5 As a user, I want to able to set a goal using default/created habits. The goal should include a habit and amout of time that I want to spend on it daily or weekly.
Archive


Board
Ownership
User Stories
New
Scrum Board
No Scrum Board 3
Google Calendar Connection
Clean Up our Scrum Board
Agree on Collaboration agreement
New page
#US01 As a team, we w... 3
Choose technology / programming language
Setup Github repository
Setting Up Notion & Scrum Board
New page
#US03 As a user, I wan... 5
Send a welcome e-mail
Provide authentication
Save created accounts in the database
Create a form for logging in
Create a form for registering
New page
#US04 As a team mem... 2
Implement Home Screen
Design Homescreen
New page
#US10 As a user, I want... 3
Add Streak On Ice
Count the streak
Create a corresponding screen section
New page
#US11 As a team mem... 4
Link a hosting to a database
Refactor some tables in DB
Hashing algorithm
Create a database
New page
Calendar view

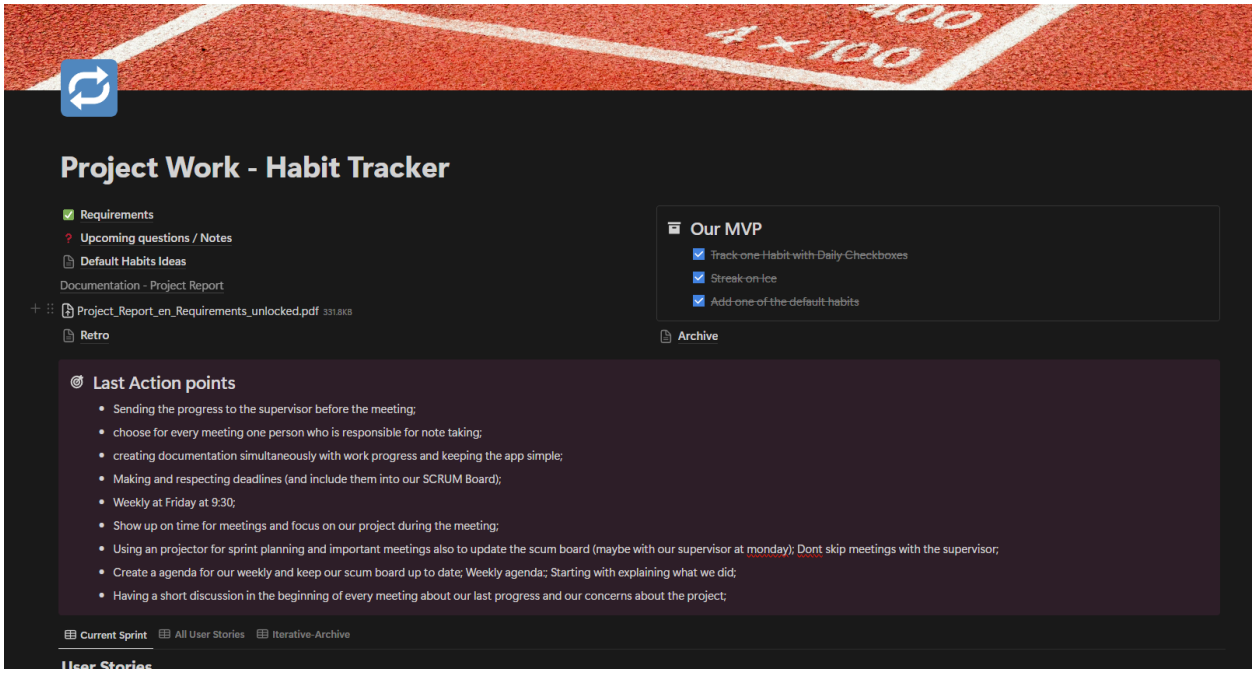

Board
Ownership
User Stories
Scrum Board
#US12 As a team, we w... 2
Prepare a report appendix
Write a report
New page
#US13 As a User, I want... 1
Unit tests
New page
#US2 As a team we wa... 2
Create a mindmap for our architecture
Discuss MVP
New page
#US6 As a user, I want ... 2
Connecting our Dashboard with the Database
Create a pop-up window with a form
New page
2 more
Hidden groups
#US5 As a user, I want ... 0

Distribution of work





Project administration



Conflict management

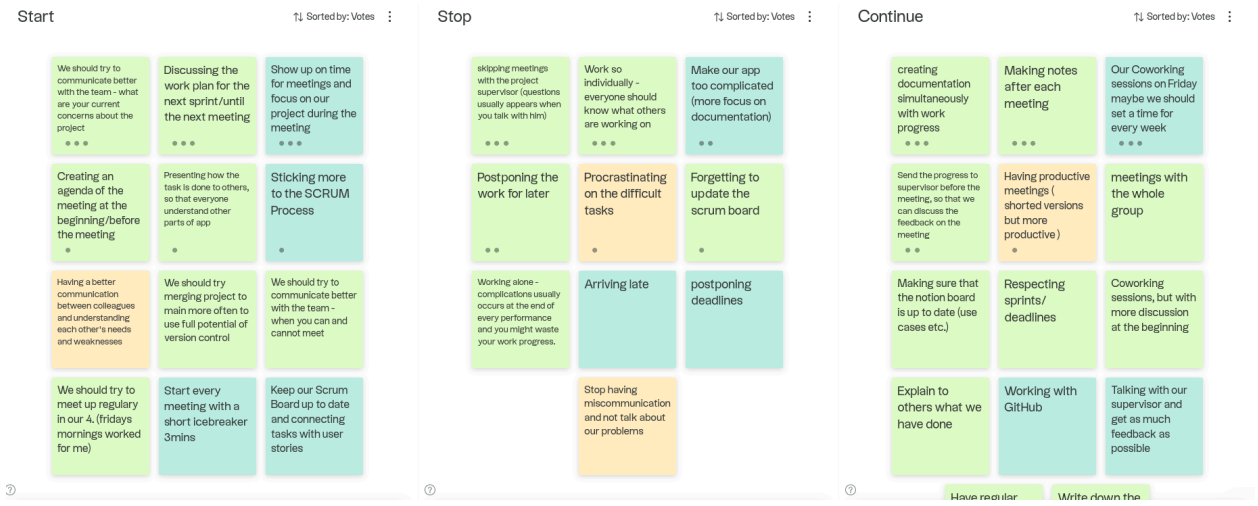
Written by Jan Walczak

For expectations conflict we have used “RetroTool”. We have divided our problems into three sections.

“Start” – which was supposed to suggest what things we should start doing, to improve our performance.

“Stop” – which was supposed to tell other members what we are fed up with or to outline things that are not good for our performance.

“Continue” – which was supposed to point things that worked well for us.



Every person had 5 minutes for each of those sections, to pinpoint their thoughts and doubts. After filling all sections, we had another 5 minutes for each one, to vote for three specific thoughts in each section, which we thought was most important for us. In a total of 30 minutes, we had proposed possible solutions and agreed on bringing them into play.

Design of the database

Initial tables content description:

Written by Jan Walczak

Below, you can see the initial idea how data should be organised inside the database.

User
Table of all users. Possibility of extension and deletion is not excluded.
Distinguishable from usernames.

Name of a row	Is a primary key?	Type
UserName	yes	Text, max. 64 characters. Letters, special characters and numbers.
Name	no	Text, max. 64 characters. Only letters
Surname	no	Text, max. 64 characters. Only letters.
Age	no	Natural number
Email	no	Text, max 64 characters. Letter, special characters and numbers.
Password	no	Text, max 64 characters. Letters, special characters and numbers

Journal
Table of all journals. Possibility of extension and deletion is not excluded.
Distinguishable from IDs. Used to store multiple habits for a specific user.

Name of a row	Is a primary key?	Type	Description
Journal_ID	yes	Natural number	The unique id for a journal. Auto-incremented value
Current_Streak	no	Natural number	The current-streak counter for a given journal. Describes how many days in a row user accomplished their goal.
Start_Date	no	Date	Date that a journal was created on
End_Date	no	Date	Date that a journal was closed on
Activity_Type	no	Text, max 64 characters. Only letters	Describes a type of activity that journal is about.

Habit
Table of all habits. Possibility of extension and deletion is not excluded.
Distinguishable from IDs. Used for creating journals and describing what is its purpose.

Name of a row	Is a primary key?	Type	Description
Habit_ID	yes	Natural number	The unique id for a habit. Auto-incremented value
Name	no	Text, max. 64 characters. Only letters.	The name of the habit
Description	no	Text, max 1024 characters. Letters, special characters and numbers	The text that describes what a habit is about.

Habit category
Table of all habits categories. Possibility of extension and deletion is not excluded.
Distinguishable from names. Used to group habits.

Name of a row	Is a primary key?	Type	Description
Habit_Cat_Name	yes	Text, max 64 characters. Only letters	The unique name for a habit category.
Description	no	Text, max 1024 characters. Letters, special characters and numbers.	The text that describes what a category is about.

Goal		
Table of all goals. Possibility of extension and deletion is not excluded. Distinguishable from IDs. Used to set a specific goal to journal.		
Name of a row	**Is a primary key?**	**Type**
Goal_ID	yes	Natural number
Description	no	Text, max 1024 characters. Letters, special characters and numbers.
Value	no	Text, max 64 characters. Only letters

Entry
Table of all entries. Possibility of extension and deletion is not excluded.
Distinguishable from IDs. Used for creating entries to an existing user's journal.

Name of a row	Is a primary key?	Type
Entry_ID	yes	Natural number
Date	no	Date
Value	no	Text, max 64 characters. Letters and numbers.

Final tables content description:

Written by Jan Walczak

Below, you can see the final idea how data should be organised inside the database.

User
Table of all users. Possibility of extension and deletion is not excluded.
Distinguishable from usernames.

Name of a row	Is a primary key?	Type
UserName	yes	Text, max. 64 characters. Letters, special characters and numbers.
Name	no	Text, max. 64 characters. Only letters
Surname	no	Text, max. 64 characters. Only letters.
Age	no	Natural number
Email	no	Text, max 64 characters. Letter, special characters and numbers.
Password	no	Text, max 64 characters. Letters, special characters and numbers. Represented by hash, to secure data.
ImageID	No	Text, max 64 characters. Letters, special characters and numbers.

Habit
Table of all habits, assigned to users. Possibility of extension and deletion is not excluded.
Distinguishable from IDs. Used for storing data about users' habits, the purpose is similar to the "Journal" table from previous design.

Name of a row	Is a primary key?	Type	Description
Habit_ID	yes	Natural number	The unique id for a habit. Auto-incremented value
HabitName	no	Text, max. 64 characters. Only letters.	The name of the habit
CurrentStreak	no	Natural number	Number of days that user performed his habit in a row.
LongestStreak	no	Natural number	Maximal number of days ever stored in the table

Entry
Table of all entries, assigned to users' habits. Possibility of extension and deletion is not excluded.
Distinguishable from IDs. Used for storing individual entries for each habit, when the user logs it on the dashboard.

Name of a row	Is a primary key?	Type	Description
EntryID	yes	Natural number	The unique id for a habit. Auto-incremented value
Date	no	Date	The date, when entry was posted by a user
Value	no	True / False value	Field for checking if a user clicked the checkbox on the dashboard for this day. Used for counting days in "Habit" table

Initial relations description

Written by Jan Walczak

Below you can see how each table is related to others. This is still our initial design, which was changed later on.

Name	**Table no. 1**	**Table no. 2**	**Type of a relation**	**Description**
User has habits	User	Habit	0… n : 0…n	Relation assign users to habits. One user can have multiple habits, and one habit could have multiple users. There could be a user without a habit, and habit could be without a user.
Habit has a certain category	Habit	Habit_Category	0…n : 1	Habit always has a category. Habit_Category could have multiple habits. There could be a Habit_Category without a Habit assigned
User has a journal	User	Journal	1 : 0…n	User could have multiple journals. Journal is always assigned for a certain user.
Journal has a habit	Journal	Habit	0…n : 1	Journal always has a habit. Habit could be assigned to multiple journals
Journal has an entry	Journal	Entry	1 : 0…n	Entry always has a journal. Journal could have multiple entries.
Journal has a goal	Journal	Goal	1 : 1	Journal always has a goal and goal always has a journal.

Final relations description

Written by Jan Walczak

Below you can see how each table is related to others. This is our final design

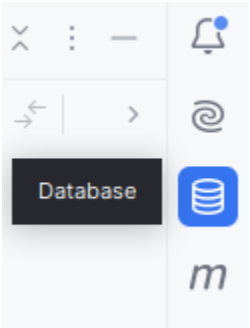
Name	Table no. 1	Table no. 2	Type of a relation	Description
User has habits	User	Habit	1 : 0…n	Relationship assigns habits to users. One user can have multiple habits, but each habit must be related to only one user. There could be a user with no habits.
Habit has entries	Habit	Entry	1 : 0…n	Relationship assigns entries to habits. One habit can have multiple entries, but each entry must be related to only one habit. There could be a habit with no entries.

Instructions of how to run our prototype

Written by Jan Walczak

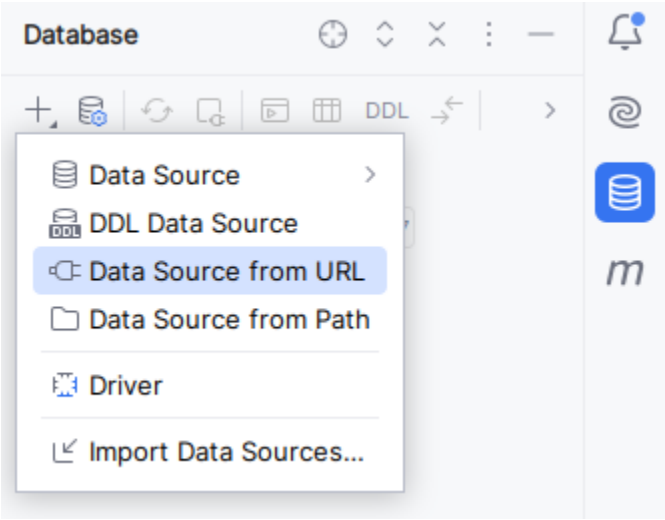
Our prototype was created on Apache **Maven 3.9.6** (Red Hat 3.9.6-6) with **Java 21.0.5**, we have used IntelIJ ultimate edition to easily work with the database's extension built in this IDE.

1. Install IntelIJ, Maven and Java, according to the above versions.
2. Sync “pom.xml” file to be sure that all dependencies are installed properly.
3. Install MySQL on your computer.
4. Create a user for MySQL
    a. Username: “root”
    b. Password: “admin”
5. Create a database inside MySQL command line:
    a. Name: “project”
    `create database project;`
    b. Enter database
    `use project;`
    c. Insert following SQL files located in …/!sql_to_create_database
        i. create.sql
        ii. insert.sql
        Note: if you opened mysql terminal inside this folder you can just use `source filename.sql;` command
6. Go to “database” tab inside IntelIJ



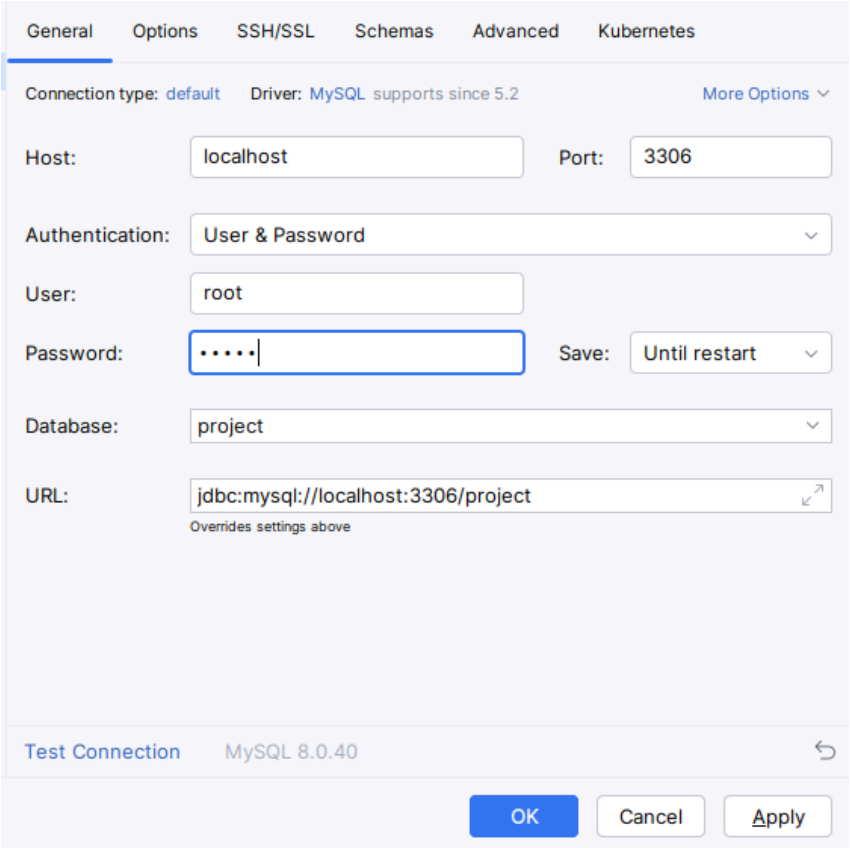
Note: If you cannot see it, go to View > Tool Windows > Database

7. Click “**+**” sign > Data source from URL



a. Paste url `jdbc:mysql://localhost:3306/project`
b. Choose driver: “MySQL”

8. Enter following fields:



Where user and password are the same as in **4.** point

9. Run “Main” file in “com.example.habittracker”
   Note!!!
   Email system works when the “secrets.properties” file, which contains our login and password, is attached to the application. As this is vulnerable data, we are not delivering this file, so method which calls email system:

   CreateAccountController.java >
   ```
   //EmailController.sendWelcomeEmail(email.getText(),
   name.getText());
   ```

   remains commented

Instructions on How to Run the Tests

Written by Cristina Semikina

Prerequisites

Before running the tests, ensure the following:

- You have a working JavaFX development environment.
- The necessary dependencies, such as JUnit, Mockito, and any database drivers, are included in the *pom.xml* file.
- The FXML files used in the application are correctly placed in the *resources* folder of the project.

Configure JVM Arguments

To resolve issues with JavaFX module access during testing, it's needed to add a specific JVM argument. This ensures that the tests can interact correctly with the JavaFX platform.

Steps to Add JVM Arguments:

1. Open the IDE (e.g., IntelliJ IDEA or Eclipse).
2. Navigate to the Run/Debug Configurations.In the **VM options** field, add the following argument:

```
--add-exports javafx.graphics/com.sun.javafx.application=ALL-UNNAMED
```

3. This allows the test framework to access internal JavaFX classes required for running JavaFX applications in a test environment.

Running the Tests

1. Open the test class you want to run, such as *DashboardControllerTest* or *LoginControllerTest*.
2. Right-click anywhere in the test class and select **Run 'TestClassName'**.
3. Ensure the *VM options* are configured correctly if you encounter errors related to JavaFX.