Date: 4.6.19
Version Number: 1.0

Project Name: 2019B.giron.aptik Smartspace

Final Report

Authors:

Giron Aptik	307863258
Shai Bremer	204313753
Elad Madar	203748751
Gal Vigoda	201166717
Lael Avraham	311212088

1. Table of Content

2. Project Requirement Document

2.1. Introduction

An application to control electrical appliances in smart space and display them on a map.
With this help you can perform remote operations on different products. (i.e Turn on and off an element, Change the element Functions).

2.2. Purpose of System

Our purpose is to Help people to control from distance on electrical appliances in smart space.

2.3. Scope of system

2.3.1. Our scope includes

- Management of User Accounts:
  - User Registration
  - User Edit
  - User Log-in

- Management of elements
  - Adding an element
  - Update an element
  - View All elements
  - View specific element

- Elements – Performing actions on elements in the house.
  - -Lamp. (on/off)
  - - Boiler. (on/off)
  - -TV (Setting volume,on/off)
  - Check in/out
  - Eco

- Management of internal score system:
  ○ Edit scores of players when performing check in / out.

2.3.2. Our scope excludes

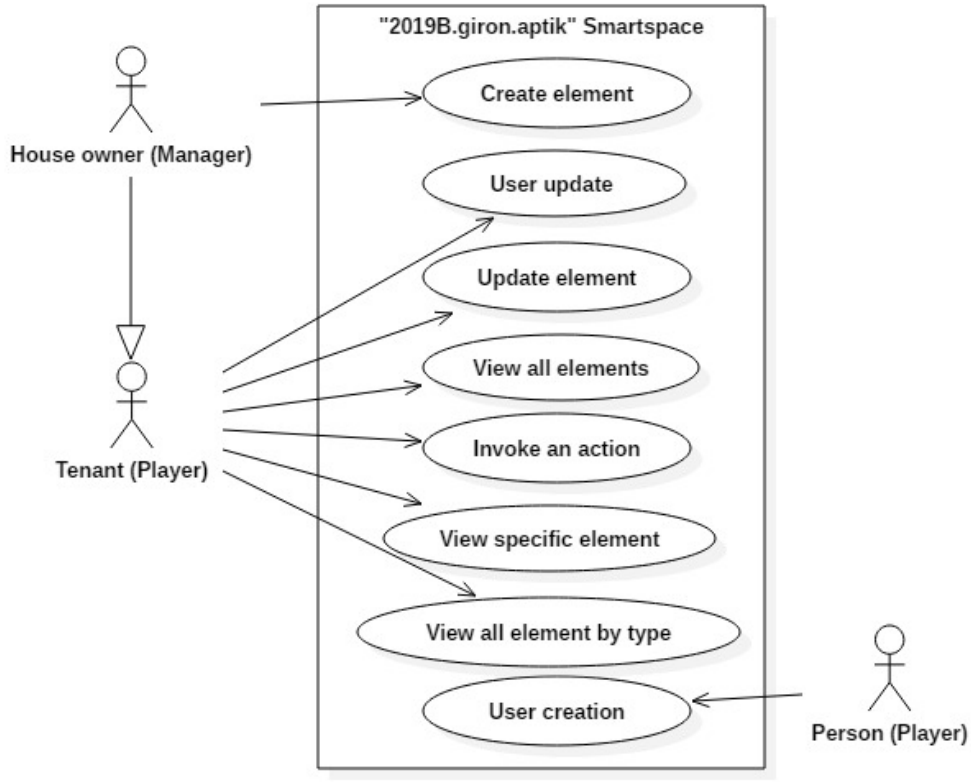
- Security against cyber attacks.
- External messaging through systems like email, whatsapp or phone.
- External verification through systems like email.

2.4. **Actors and Goals**

Actor name	Description	Goals	Role
Tenant (Player)	Make actions on the elements	Remote control of electrical appliances and finding them on a map	Primary
House Owner (Manager)	The House owner can position objects on the map	Add and remove new elements	Primary
Person	A potential future user		Support

2.5. **Functional Requirements**

2.5.1. Use Case Diagram



2.5.2 Use Case Details

2.5.2.1 Use Case 1: User Creation

Goal: Creating new user account

Actors: Person

Basic Flow:

Person	System
1.Person sending request to sign up	
	2.The system request to fill New User form
3.The person submits the form	
	4.The system verifies person's form 5.The system creates new User

Alternative Flow:

Person	System
1.Person sending request to sign up	
	2.The system request to fill New User form
3.Person Submint the Form with invalid role	
	4.The system throw a message for illegal Role

2.5.2.2 Use Case 2: User update

Goal: Update user info

Actors: Tenant, House owner

Basic Flow:

Tenant, House Owner	System
1.Identify user (Login)	
	2.Confirm identity
3.Asking for edit his own info	
	4.System ask to update user form
5.User edit the information	
	6.Validate the information 7.Update user profile on DB

Alternative flow:

Tenant, House Owner	System
1.Identify user (Login)	
	2.Confirm identity
3.Asking for edit email	
	4.The system throws an error message

2.5.2.3 Use Case 3: Create Element
Goal: Add element to the DB
Actors: House owner
Basic Flow:

House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Ask to add element	
	4.Request element information
5.Submitting element information with null key	
	5.Verifies the form 6.Create element and add it to DB

Alternative Flow:

House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Ask to add element	
	4.Request element information
5.Submitting element information with key value	
	6.Throw an error

2.5.2.4 Use Case 4: Update Element
Goal: Change element details
Actors: House owner, Tenant
Basic Flow:

House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Ask to update element	
	4.Request element information
5.Submitting element information	
	5.Verifies that element exist 6.Verifies the update form 7.Updates the relevant element in DB.

Alternative Flow:

House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Ask to update element	
	4.Request element information
5.Submitting element information with invalid creator information	
	5.Verifies that element exist 6.Verifies the update form 7.Throw related error

Alternative Flow:

House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Ask to update element	
	4.Request element information
5.Submitting element information with new key value	
	5.Verifies that element exist 6.Verifies the update form 7.Throw related error

2.5.2.5 Use Case 5: View All Elements
Goal: View of all existing elements at home
Actors: House owner, Tenant
Basic Flow:

Tenant, House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Ask to see all elements	
	4.Gather all Elements

Alternative Flow:

Tenant, House owner	System
1.Identify user (Login)	
	2.Confirm Identity 3.User not allowed to log in 4.Throws relevant error

2.5.2.6 Use Case 6: View Specific Element
Goal: View of all existing elements at home
Actors: House owner, Tenant
Basic Flow:

Tenant, House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Ask to see element	
	4.Request Element Information
5.Submitting Element information	
	5.Verifies Element information 6.Displaying the Element

Alternative Flow:

Tenant, House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Ask to see element	
	4.Request Element Information
5.Submitting Element information	
	5.Element doesn't exist 6.Throws relevant error

2.5.2.7 Use Case 7: View All Element by Type
Goal: View of all existing elements at home
Actors: House owner, Tenant
Basic Flow:

Tenant, House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Ask to see elements	
	4.Request Elements Type
5.Submitting Element Type	
	5.Verifies Elements information 6.Gather all related elements

Alternative Flow:

Tenant, House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Ask to see elements	
	4.Request Elements Type
5.Enter invalid type	
	5.Element doesn't exist 6.Throws relevant error

2.5.2.8 Use Case 8: Invoke an action
Goal: Perform an action on element
Actors: House owner, Tenant
Basic Flow:

Tenant, House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Create ECHO action	
	4.Verifying that the action with null key 5.Veryfing that the action type existing 6. Performing the ECHO

Alternative Flow: Check In/Out (Open/ Close Door)

Tenant, House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Create Check-in/out action with relevant element id.	
	4.Verifying that the action with null key 5.Veryfing that the action type existing 6. Verifying that the relevant element exists with relevant type 7.Change the status property respectively

Alternative Flow: Turn On/Off Elements

Tenant, House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Create Turn On/Off action with relevant element id.	
	4.Verifying that the action with null key 5.Veryfing that the action type existing 6. Verifying that the relevant element exists 7.Change the state property respectively

Alternative Flow: Change Element Volume

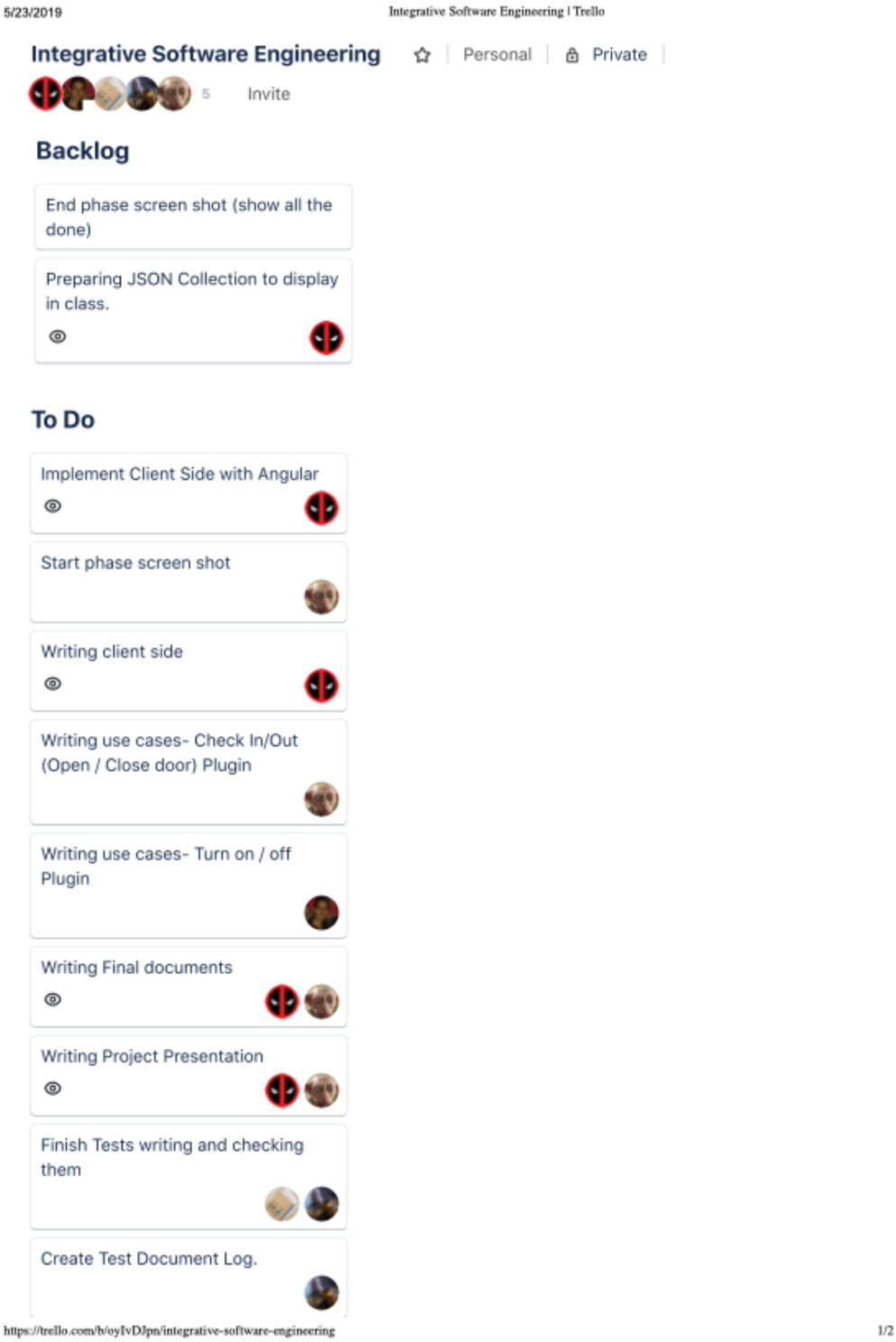
Tenant, House owner	System
1.Identify user (Login)	
	2.Confirm Identity
3. Create Volume Change action with relevant element id and requested volume.	
	4.Verifying that the action with null key 5.Veryfing that the action type existing 6. Verifying that the relevant element exists with relevant property 7.Change the volume property as required by user.

2.6. **Non-Functional Requirements**

Requirement Number	Requirement Description	Requirement Type
1	Users doesn't need to install anything, only a browser and a network access is required	U - Usability
2	In case of server crash, the DB saved in mongoDB cloud.	R - Reliability
3	Server will show alert messages on incorrect input.	S - Supportability
4	Server will show a log on which services were requested.	S - Supportability

3. Appendix: Project progress report

3.1 Kanban board of Final project (23/5/19)

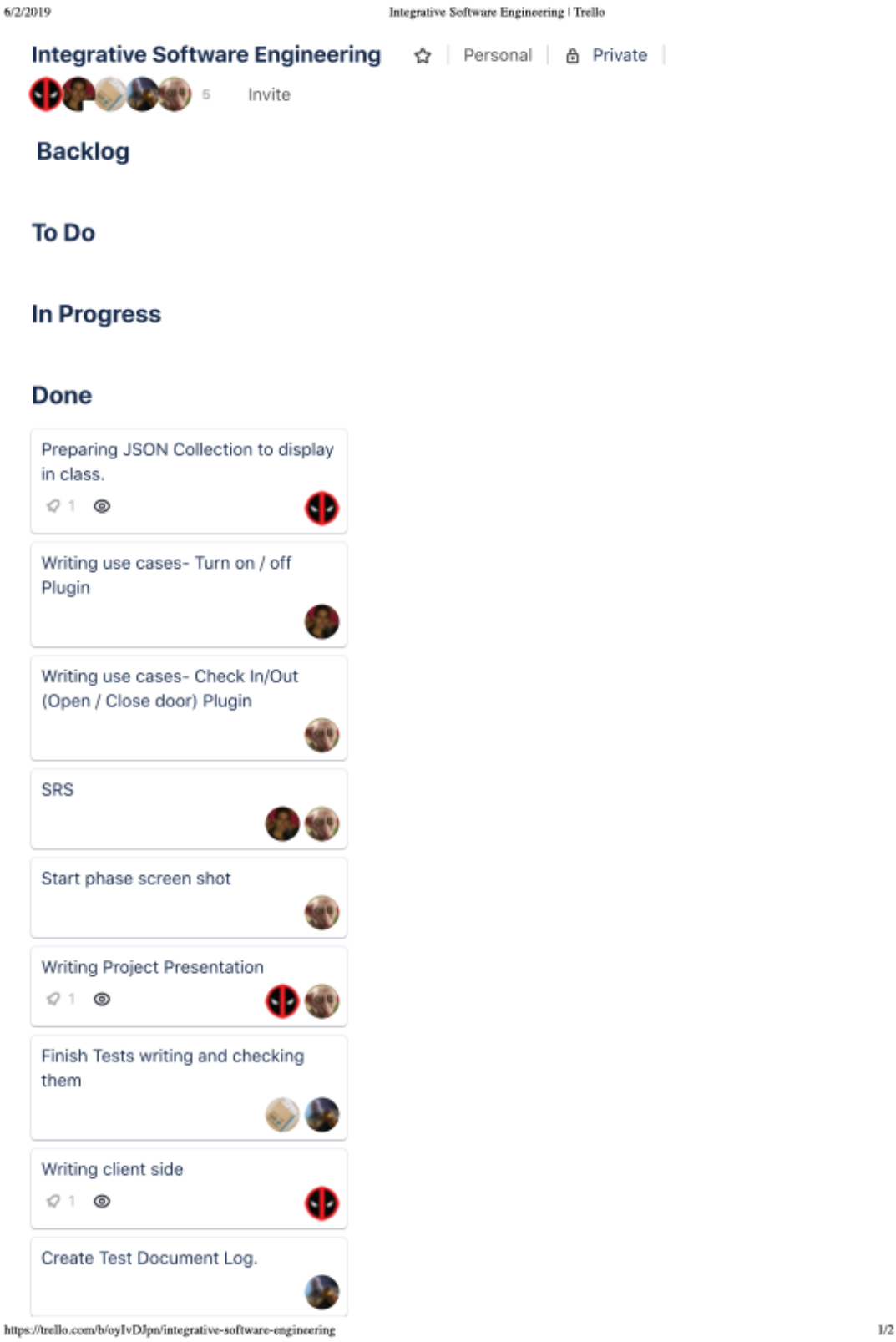


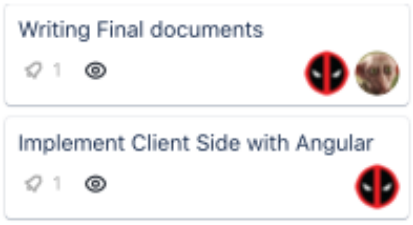
In Progress

SRS

Done

3.2 Kanban board of Final project (4/6/19)




Writing Final documents
1
Implement Client Side with Angular
1

3.3 List of Students

3.3.1 Giron Aptik:

1. ID: 307863258.
2. Roles: Team Leader, DevOps, Developer.
3. Avatar: 

3.3.2 Shai Bremer:

1. ID: 204313753.
2. Roles: Product manager, Developer.
3. Avatar: 

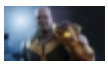
3.3.3 Elad Madar:

1. ID: 203748751,  .
2. Roles: QA engineer, Developer.
3. Avatar: 

3.3.4 Gal Vigoda:

1. ID: 201166717.
2. Roles: Technical Writer, UIX Engineer, Developer.
3. Avatar: 

3.3.5 Lael Avraham:

1. ID: 311212088.
2. Roles: DBA, Developer.
3. Avatar: 

3.4. General Summary of works

3.4.1. What Worked for us:

In this Milestone we tried to divide the tasks according to time and to meet them. This made the team members become a bit more with attention to work. In addition, we divided into pairs and sat on the work physically so that the work was more efficient

3.4.2. Improvements:

Team members should care a lot more, not wait for the team leader to give them work and chase them, but to understand the current situation and take on the part they think is relevant to them and want to do it.

3.4.3. Problems:

We had delays with the ending number of the SRS document which took too long to be unforeseen. In addition, client-side development was complex because upgrades were added that were not initially planned.

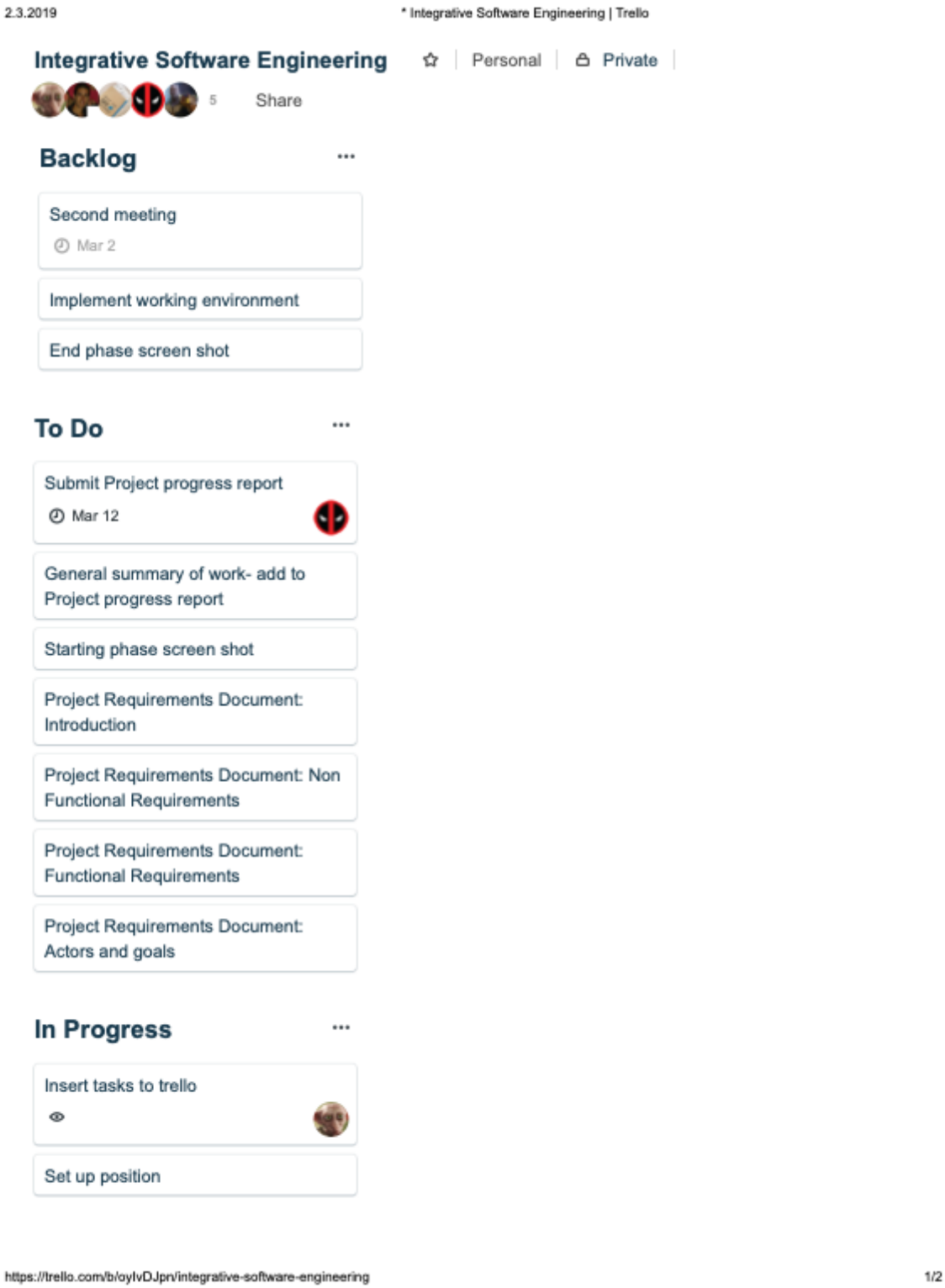
3.4.4. Why did not we finish what we started:

We have completed everything we planned for the final submission.

4. Appendix: Conclusion Report

4.1.1 Sprint 0: Project progress report

4.1.2 Kanban board of project from spring 0 initiation (2/3/19)



Done

Meeting

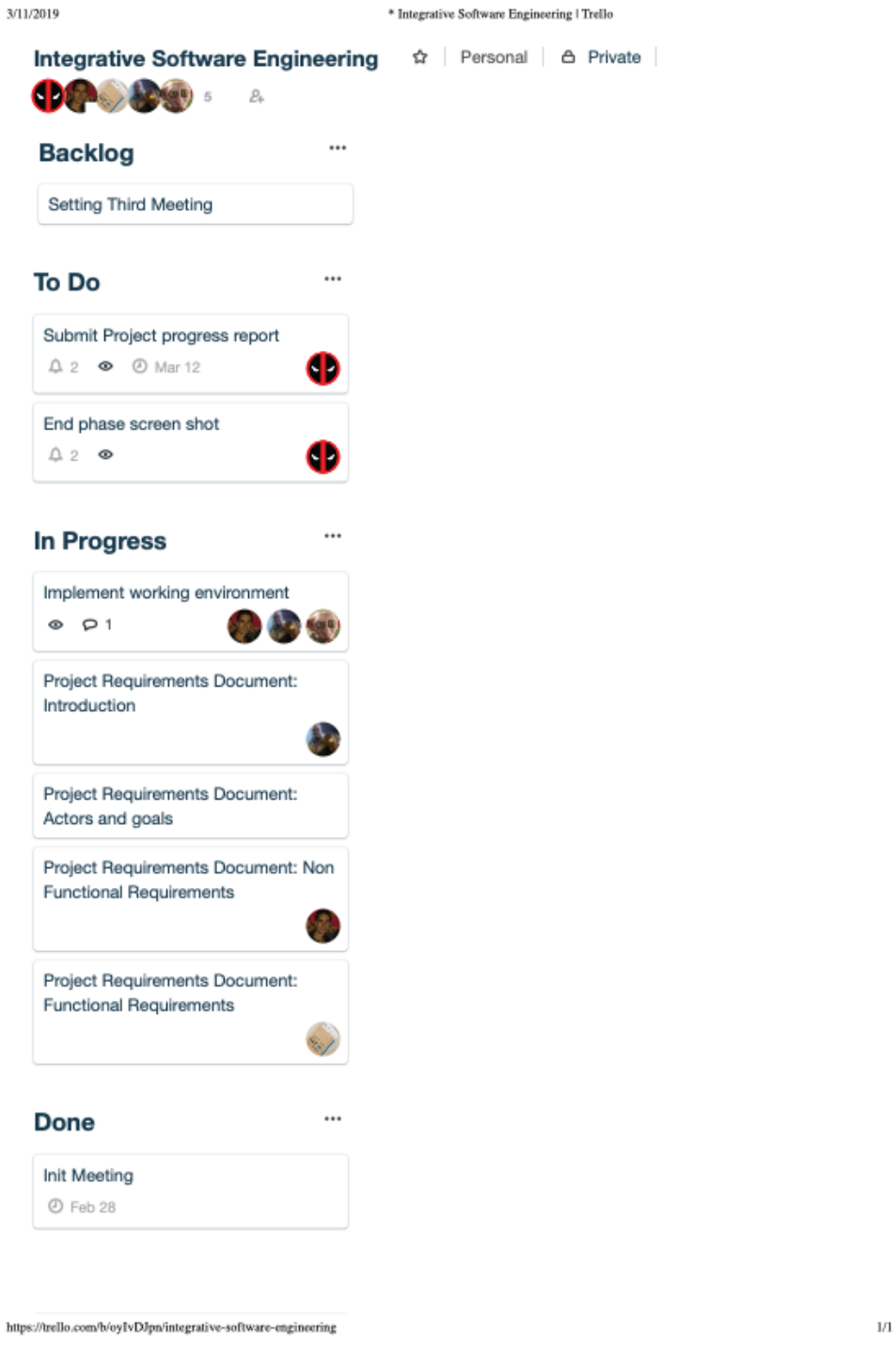
Feb 28

Open Project Log

Upload team's names onto the forum

Choose subject

4.1.3 Kanban board from end of sprint 0 (11/3/19)



4.1.4 General summary of work

- **What went well for the team and should be continued on next assignments:**

  The team members wanted to take a role in a field they did not know about to learn things in rooms. -Intervention and knowledge sharing of one team member to another in a field where he is more powerful

- **What should be improved in team work:**
  -Try to set future meetings faster then the first 2 meetings.

  -Keep in sync with all group members

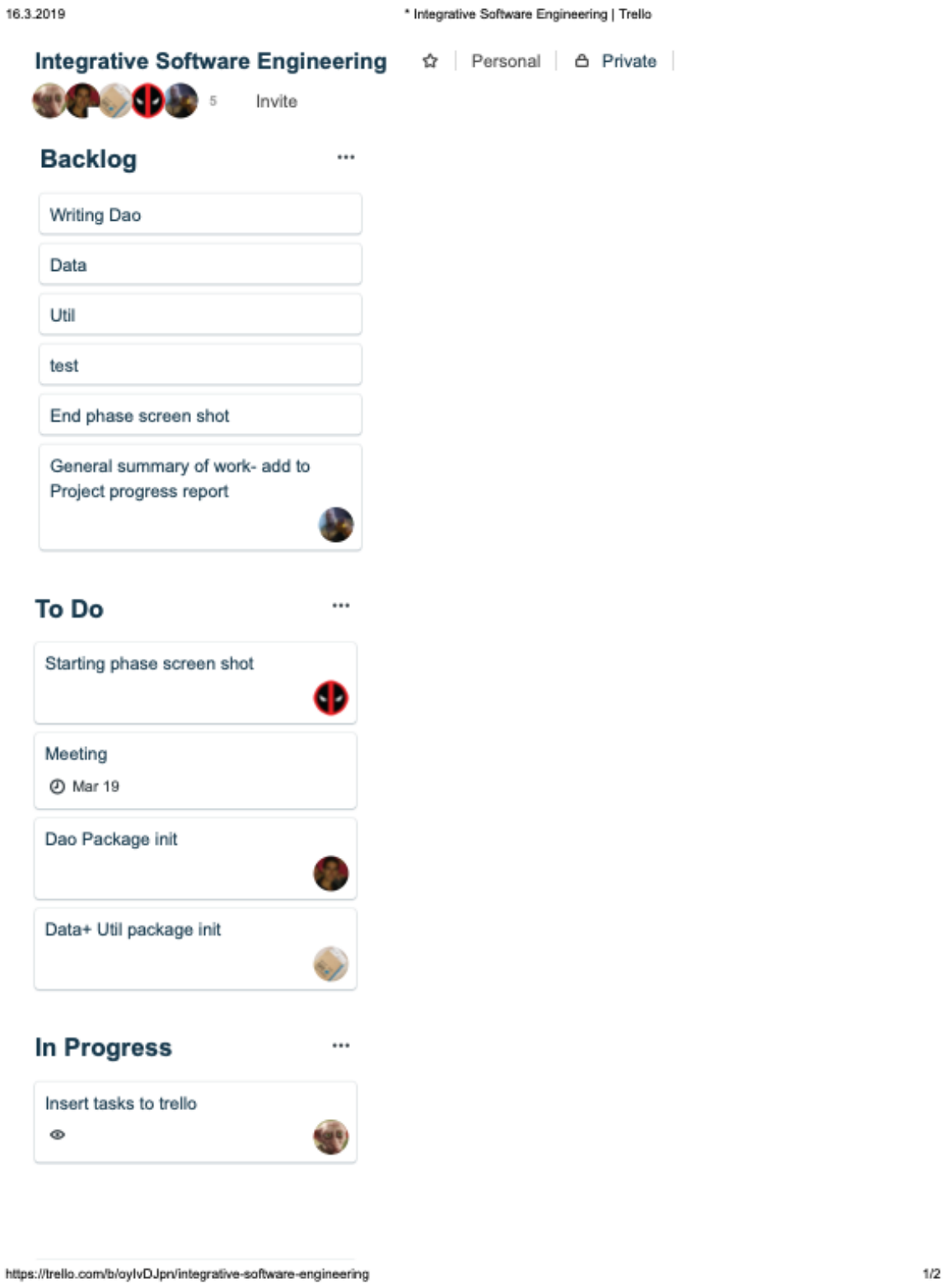
- **What problems did the team encountered through the work:**
  -Implementing the new environment, we had a problem with JDK 11 and JDK 8.

- **Why did we not complete all planned work:**
  We changed the first subject that we thought about after the second lesson, so we didn't finished
  the Requirements Document

- **What is expected for the next stages of your project?**
  Finish the Requirements Document before we start the next stage. -Set advance times and pair to them

4.2.1 Sprint 1: Project progress report

4.2.2 Kanban board of project from spring 1 initiation (16/3/19)



Project Requirements Document: Introduction

Project Requirements Document: Actors and goals

Project Requirements Document: Non Functional Requirements

Project Requirements Document: Functional Requirements

Done

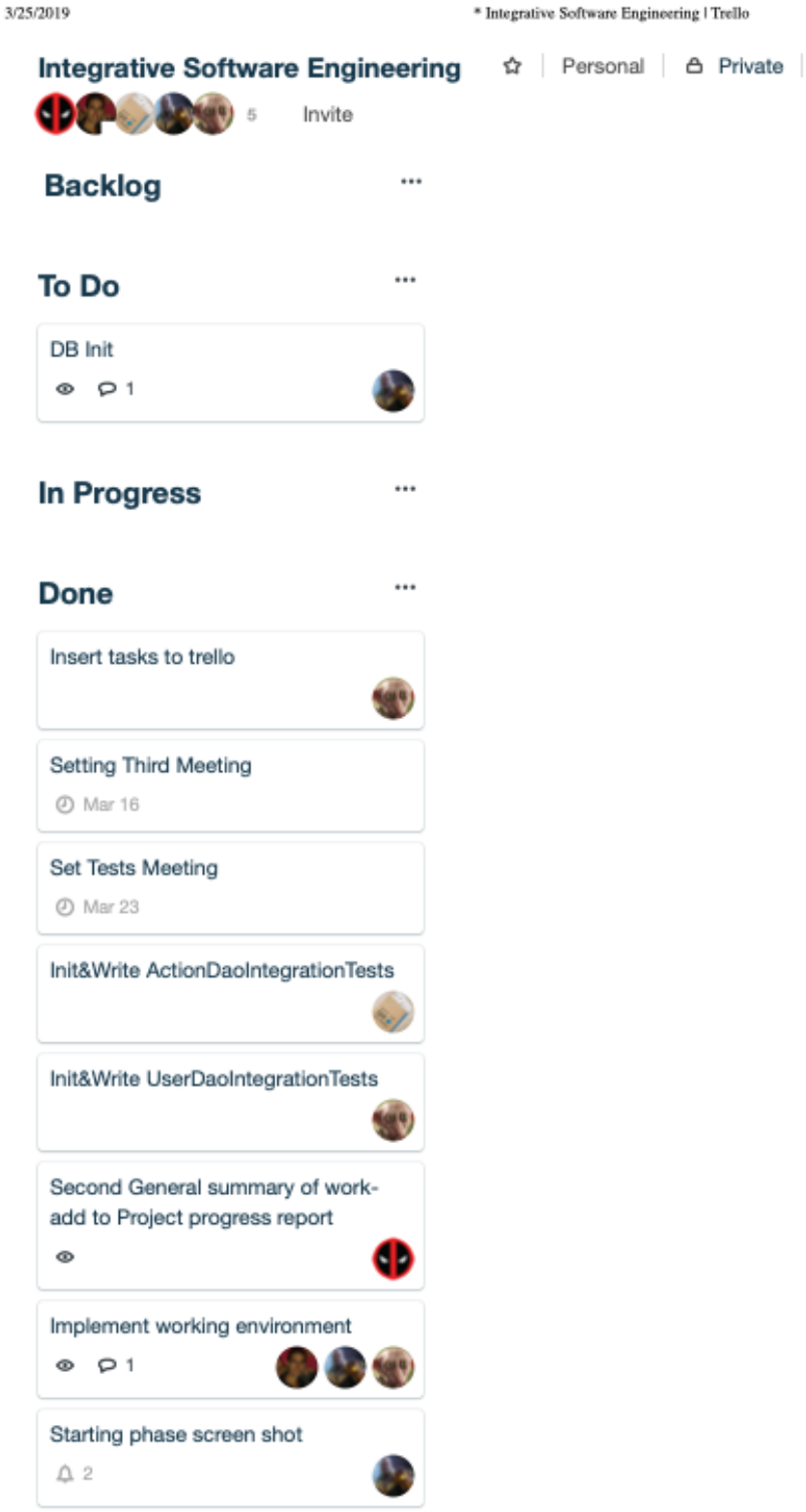
Setting Third Meeting

Mar 16

Implement working environment

1

4.2.3 Kanban board from end of sprint 1 (25/03/19)



Init&Write ElementDaoIntegrationTests

Writing Dao Implementations

Writing Util - EntityFactoy Implementation

test

Project Requirements Document: Functional Requirements

2

Dao Package init

Data+ Util package init

Project Requirements Document: Non Functional Requirements

2

Project Requirements Document: Actors and goals

Tests Requirements for the next Meeting

Project Requirements Document: Introduction

Meeting

Mar 19

Data Final inits

Registration to MongoDB

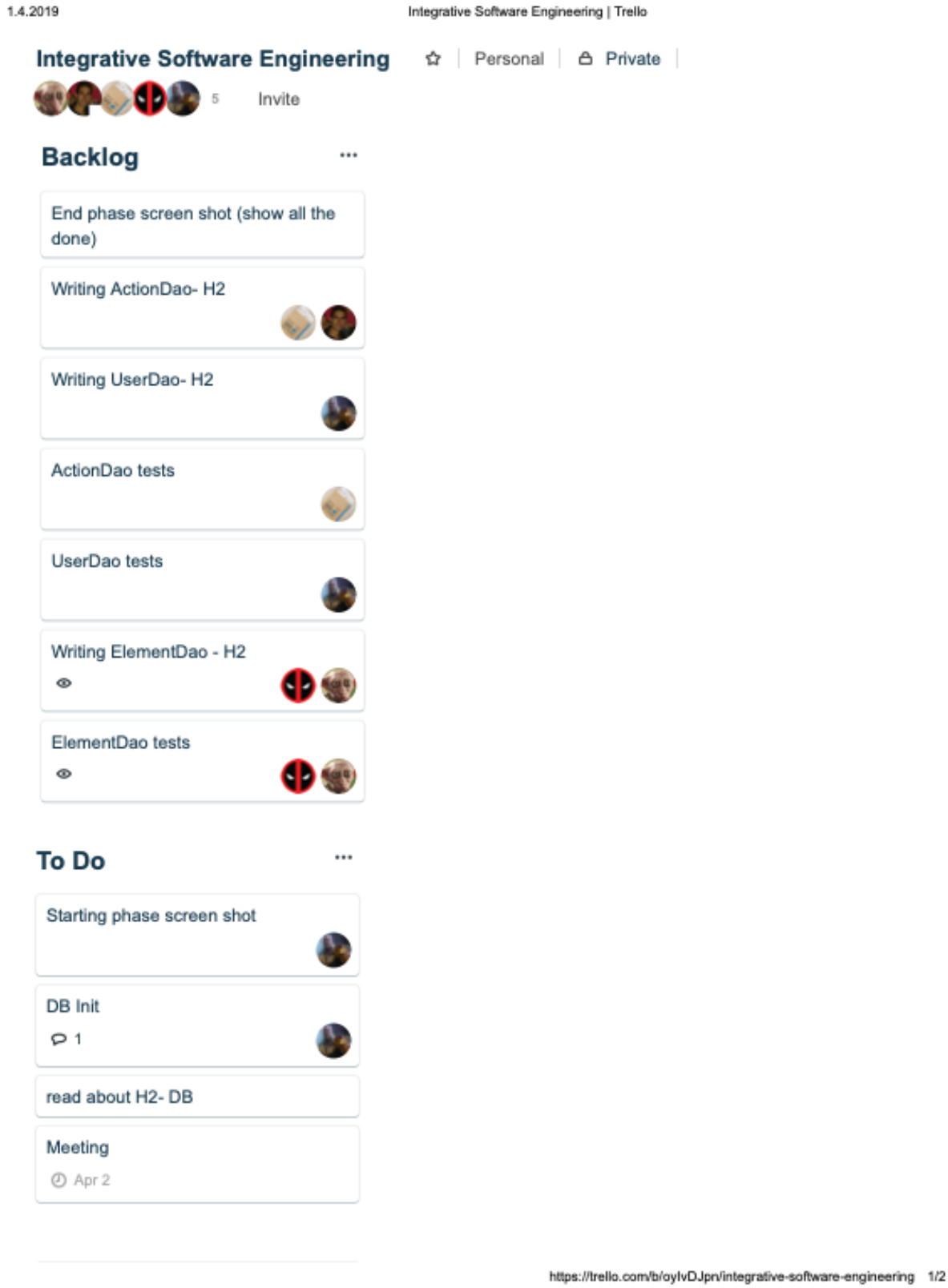
End phase screen shot

4.2.4 General summary of work

- **What Worked for us:** The team members got used to the Trello and the GIT, everyone know that The meetings were much more effective than the previous one.
- Everyone quickly understood what work was left and quickly divided the work. There was cooperation and assistance among the team members.
- **Improvements:** Meeting deadlines and trying to finish the tasks as quickly as possible to handle more things or help others finish their tasks.
- **Problems:** There were some things that were not clear after the last lesson, but they were helped by other groups, and that is how we completed the gap very quickly and we started working.
- **Why did not we finish what we started:** In this submission we were able to meet everything we set ourselves.
- **What's next:** Deepen the existing tests and implement a more dense schedule in the following tasks.

4.3.1 Sprint 2: Project progress report

4.3.2 Kanban board of project from spring 2 initiation (1/4/19)



General summary of work- add to Project progress report

In Progress

Insert tasks to trello

Finish SOW

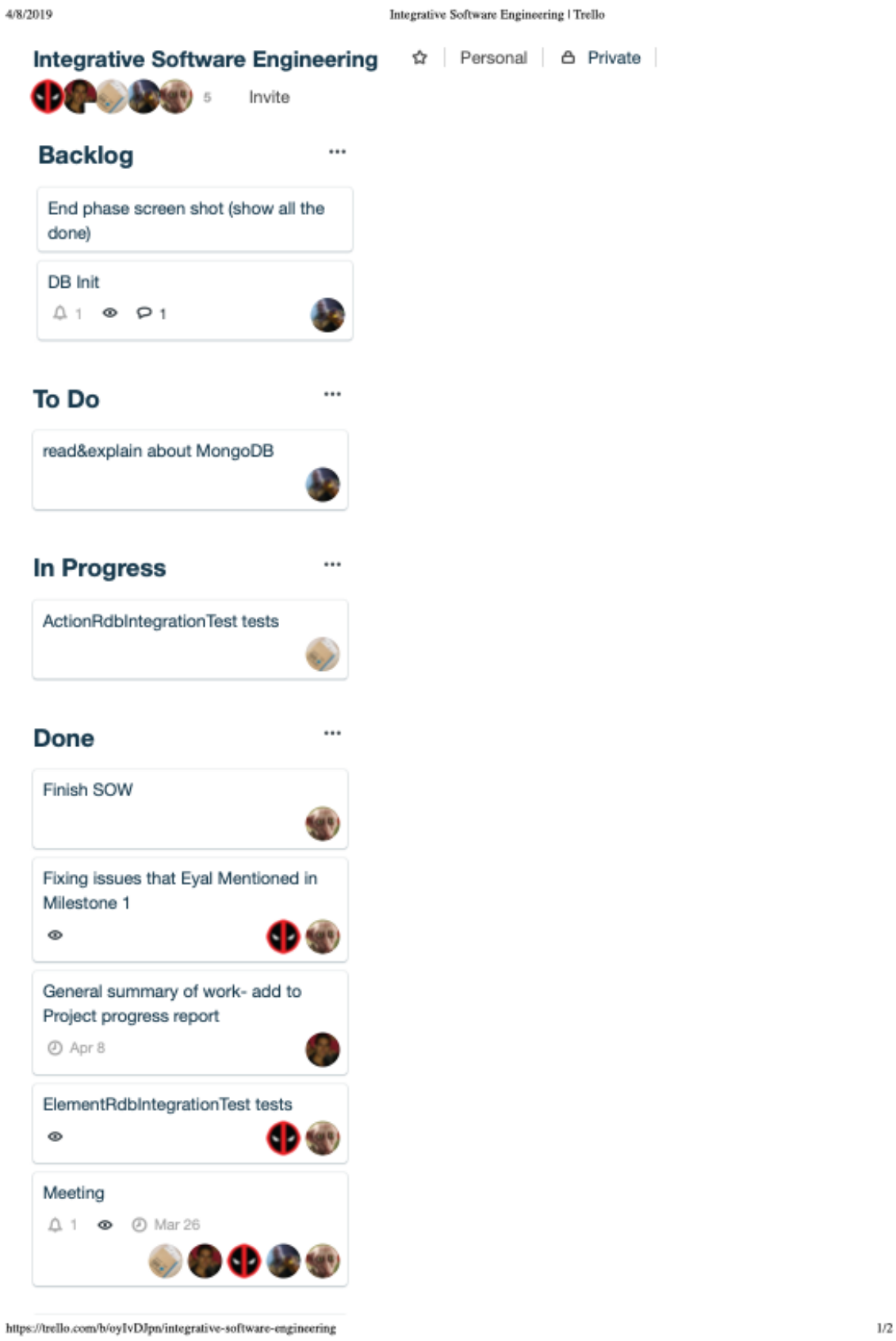
List of appendices technology

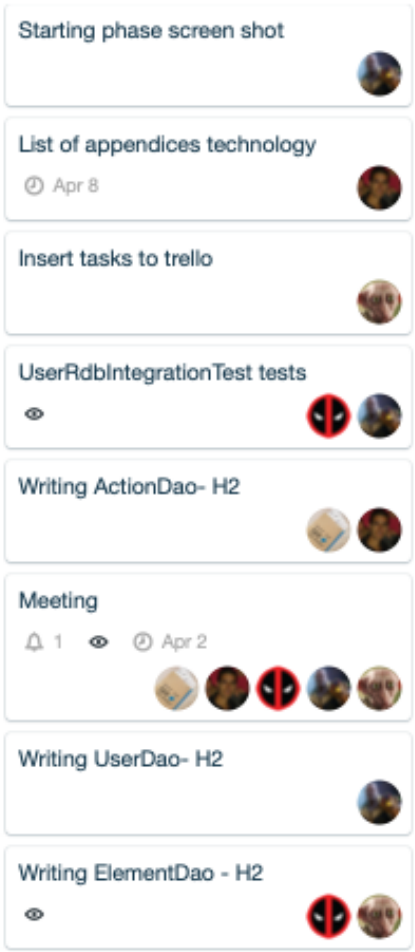
Done

Meeting

Mar 26

4.3.3 Kanban board from end of sprint 2 (8/4/19)



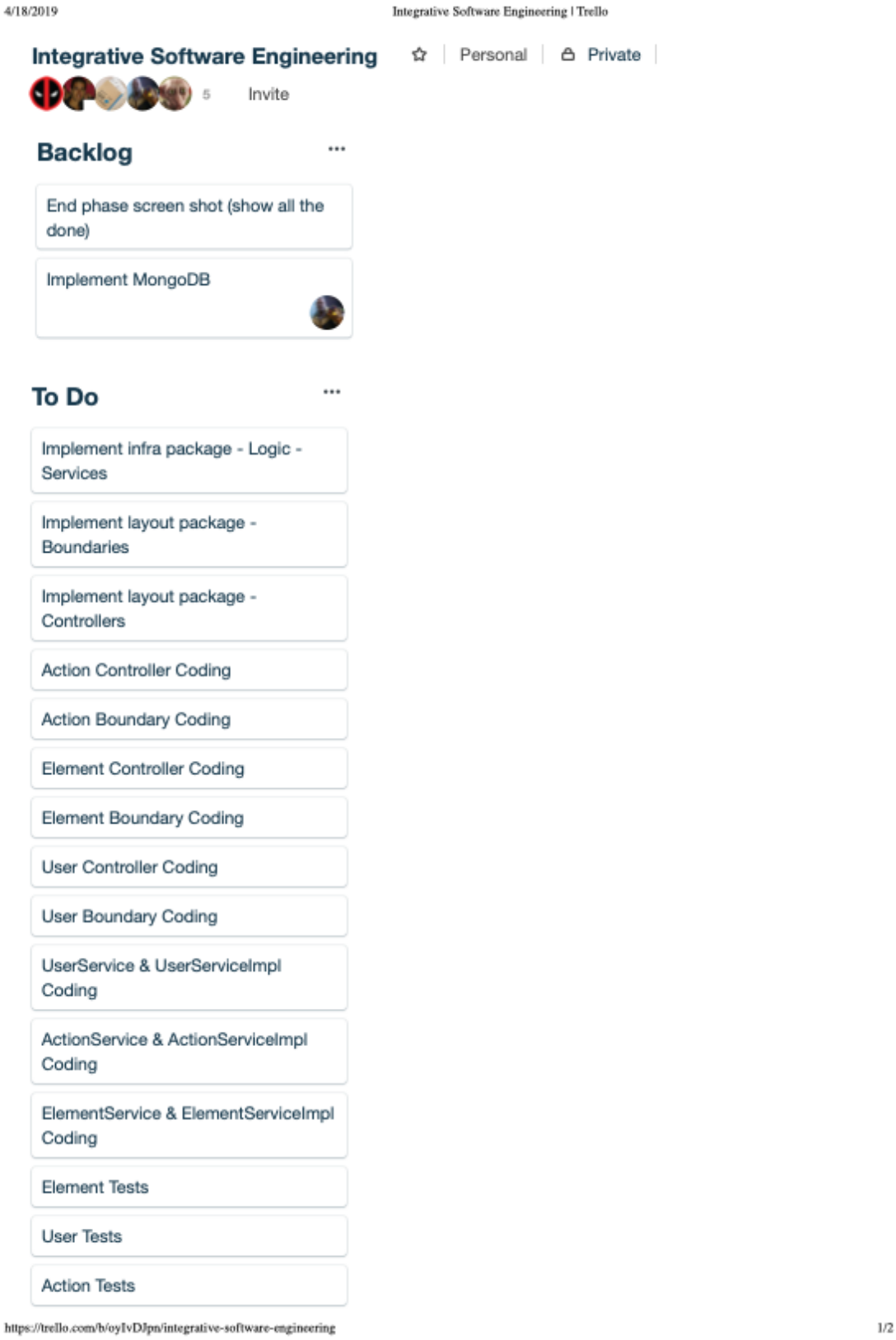

Starting phase screen shot
List of appendices technology
Apr 8
Insert tasks to trello
UserRdbIntegrationTest tests
Writing ActionDao- H2
Meeting
1
Apr 2
Writing UserDao- H2
Writing ElementDao - H2

4.3.4 General summary of work

- **What Worked for us:** The team members continue used the Trello and to upload the classes to the bitbucket -git. The Skype-meeting was helped us to divided the work between us and work correctly together. We maintain a workflow, each one enters the Trello and checks its tasks and updates them accordingly, know to update the team in the Project log and WhatsApp for new push to git or new update.
- **Improvements:** Meeting deadlines and trying to finish the tasks as quickly as possible to handle more things or help others finish their tasks.
- **Problems:** We had a major fault with Spring boot after adding the DAO with DB, it took us too much time to manage and fix but we managed it eventually.
- **Why did not we finish what we started:** We did not write the tests on the scale we wanted but we met all the tasks.
- **What's next:**
  - Continue to deepen the existing tests
  - Implement a denser schedule in the following tasks.
  - Preparing to next milestone with an expansion of the tests and improvement of the project according to the comments.

4.4.1 Sprint 3: Project progress report

4.4.2 Kanban board of project from spring 3 initiation (18/04/19)



In Progress

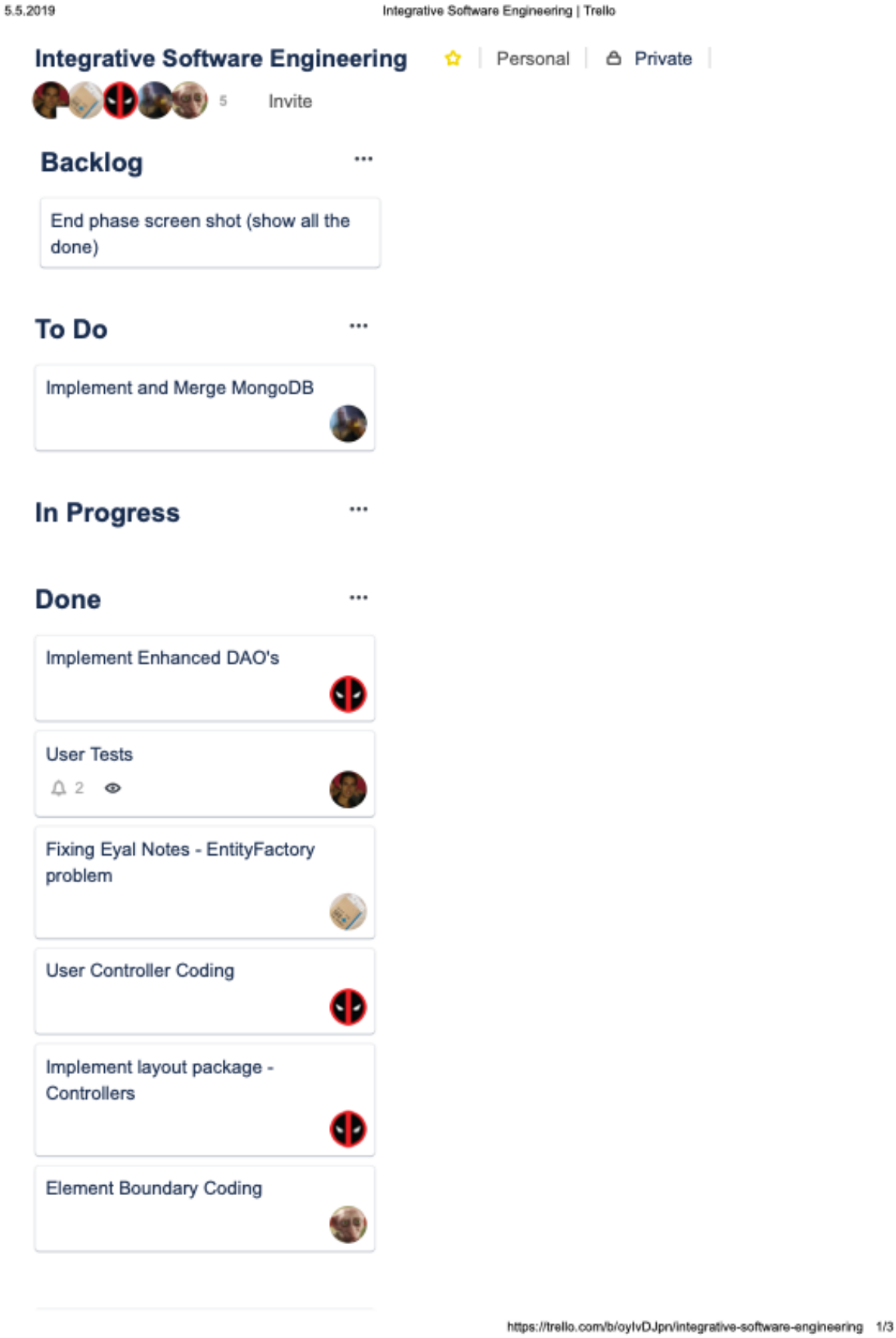
Fixing Eyal Notes - EntityFactory problem

Implement Enhanced DAO's

Fixing Eyal Notes - Key Problems

Done

4.4.3 Kanban board from end of sprint 3 (5/5/19)



ActionService & ActionServiceImpl Coding

2

Implement infra package - Logic - Services

UserService & UserServiceImpl Coding

Action Boundary Coding

Action Controller Coding

Element Tests

Fixing Eyal Notes - Key Problems

Action Tests

ElementService & ElementServiceImpl Coding

Implement layout package - Boundaries

User Boundary Coding

Element Controller Coding

4.4.4 General summary of work

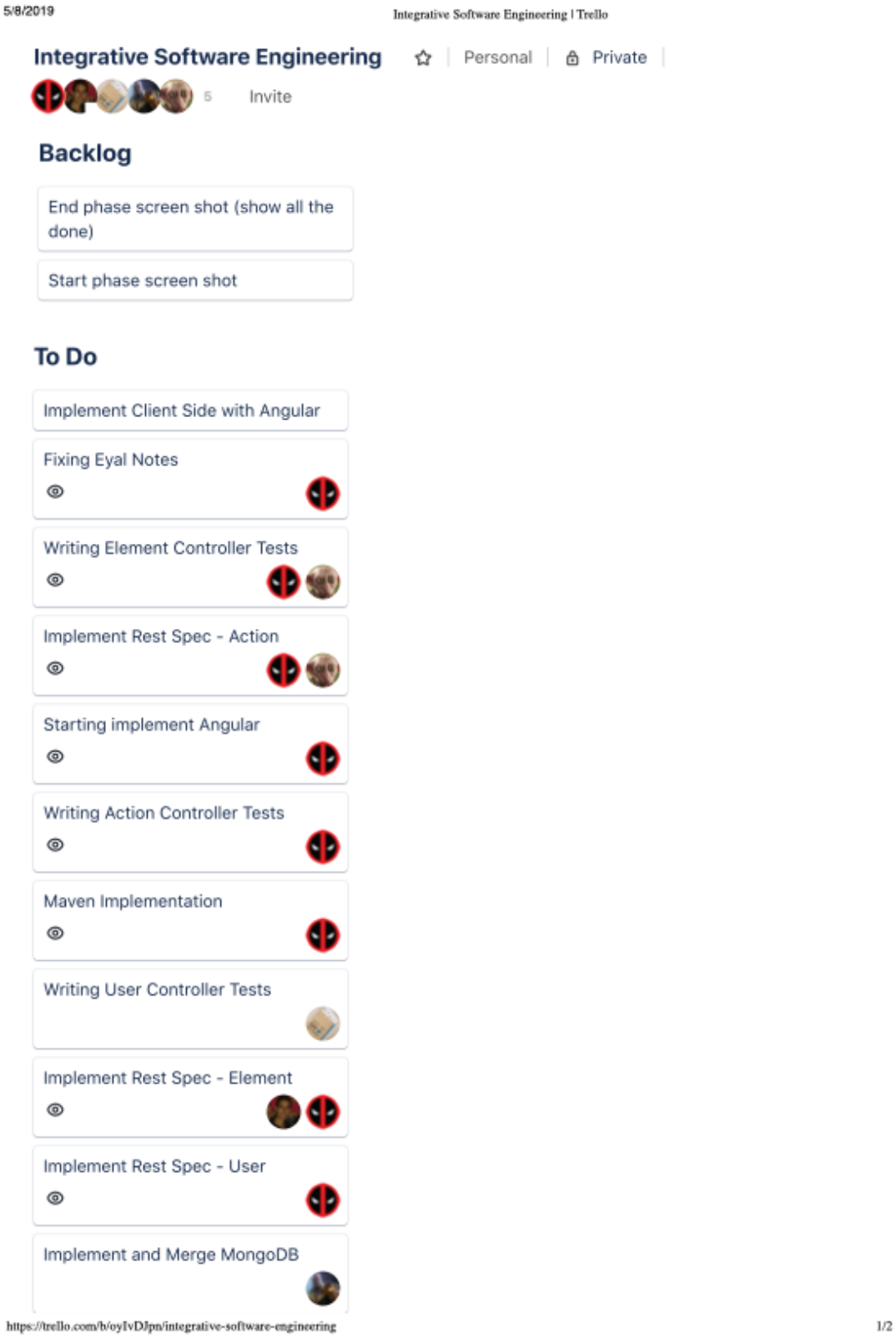
- **What Worked for us:** The team members continue used the Trello and to upload the classes to the bitbucket - git. Everyone Get used to the development tool we work: bitbucket and Trello.
- **Improvements:**

  *Meeting deadlines and trying to finish the tasks as quickly as possible to handle more things or help others finish their tasks.

- Improving the division of teamwork, this time had difficulties due to constraints (military service / illness)**.**
- **Problems:** We had difficulty in implementing JSON as required by Rest API spec but we used an example from the lesson and explanations from the internet. The time we took for each task took to much time than we expected and caused a delay, but we took a safety factor and se we managed on time.
- . **Why did not we finish what we started:** We had finished what we plan.

4.5.1 Sprint 4: Project progress report

4.5.2 Kanban board of project from spring 4 initiation (8/5/19)

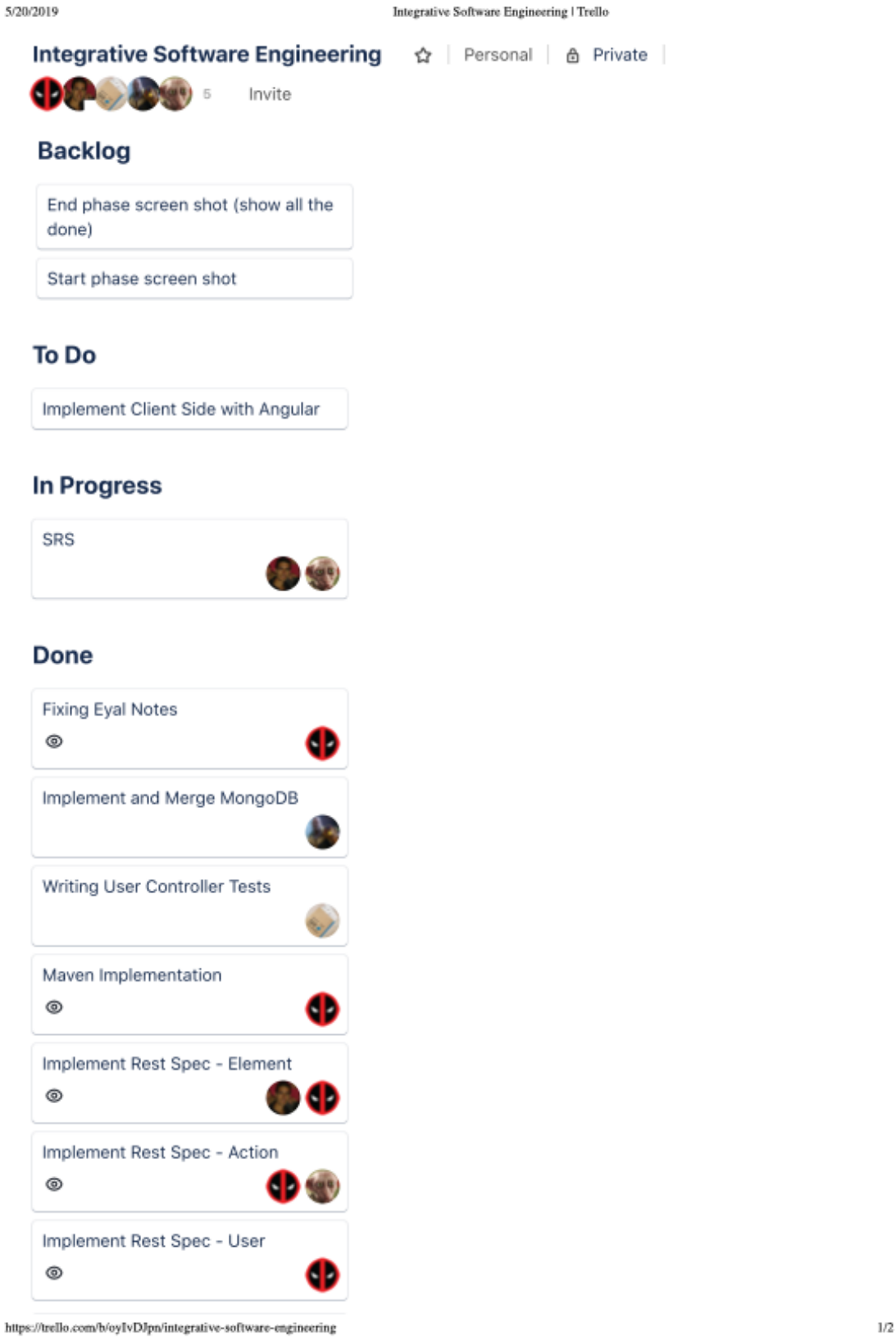


In Progress

SRS

Done

4.5.3 Kanban board from end of sprint 5 (20/5/19)




Writing Element Controller Tests
Starting implement Angular
Writing Action Controller Tests

4.5.4 General summary of work

- Note: We have implemented mongoDB and Maven in the project.

To start Maven we need the press right-click on the project name, Configure -> Convert to Maven (make sure that the pom.xml is in the project folder)

To enter mongoDB web application: https://cloud.mongodb.com Username: integrationcourse@gmail.com, Password: Afeka2019!

- **What Worked for us:** The team members continue used the Trello and to upload the classes to the bitbucket -git. 2 team members sat together on their missions to help each other.
- **Improvements:** Meeting deadlines and trying to finish the tasks as quickly as possible to handle more things or help others finish their tasks.
- **Problems:** The division of labor within the team is incorrect, we had to spend a lot of time planning to correct the comments from the previous Milestone. We encountered problems in realizing the mongoDB that we were eventually able to solve by adding key attribute to our Entities (That was the only solution we found).
- **Why did not we finish what we started:** We did not write the tests on the scale we wanted but we met all the tasks.
- **What's next:**
    - Implement Client Side
    - Finishing SRS and Presentation.
    - Implement a denser schedule in the following tasks.
    - Preparing to next milestone with an expansion of the tests and improvement of the project according to the comments.

5. Appendix: Technology List

5.1. Google Docs
5.2. Google Drive
5.3. Skype
5.4. TeamViewer
5.5. Eclipse
5.6. Trello
5.7. Java
5.8. Bitbucket
5.9. Spring
5.10. Spring Web
5.11. Spring Boot
5.12. Spring Data JPA
5.13. Spring Test, Assert
5.14. HTTP (POST/PUT/GET/DELETE Requests)
5.15. JSON
5.16. JPA
5.17. Postman
5.18. Junit
5.19. Jackson
5.20. Hibernate
5.21. H2 Database
5.22. mongoDB
5.23. tomcat
5.24. Node.js
5.25. Npm (Node.js Package Manager)
5.26. Angular 8
5.27. Bootstrap

6. Appendix: Installation Guide

Required Installations:

- Java JDK 8
- Eclipse for Java Developers
- Google Chrome (Recommended)

Note:

- There is no need to install any Database.
- There is no need to download or install any external jar libraries As the project is a Maven project.

Download and install Java JDK:

Download Java JDK from oracle website and run the installation:

https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

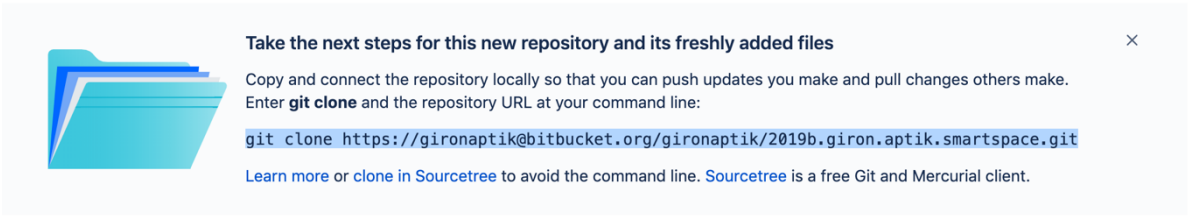
Download and install Eclipse IDE for Java Developrs:

Download Eclipse IDE for Java Developers from Eclipse website and run the installation

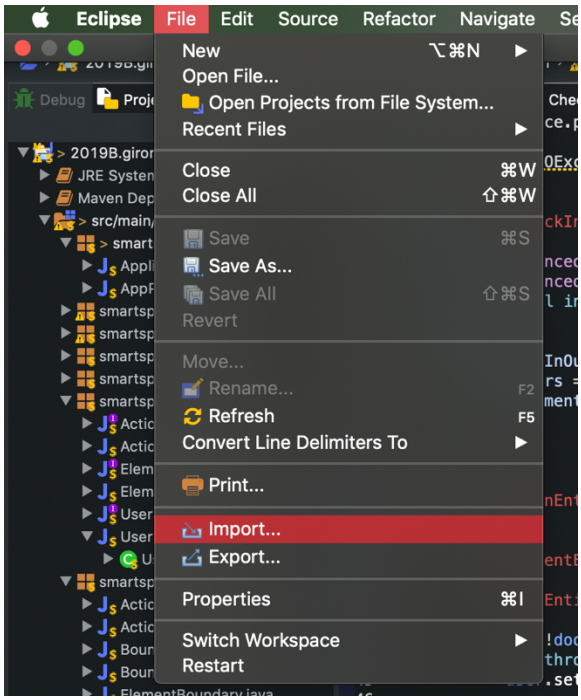
https://www.eclipse.org/downloads/packages/

Creating a project with the source code and run it (Server Side).

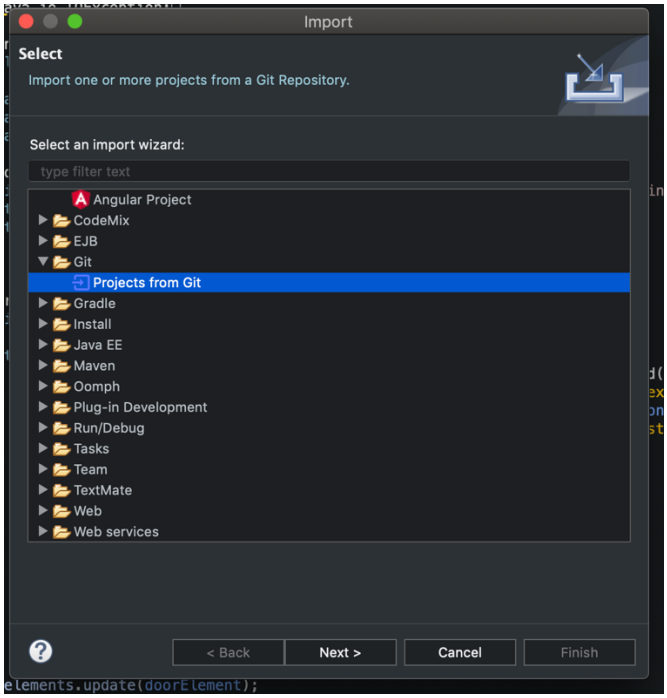
1. Open Eclipse and create new project
2. Copy the Git URL from bitbucket: https://bitbucket.org/gironaptik/2019b.giron.aptik.smartspace/src/master/



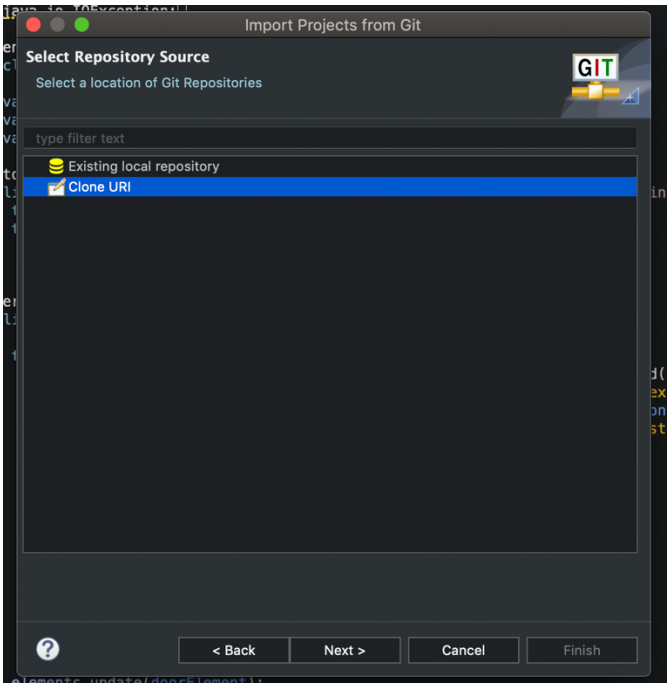
3. In Eclipse click File -> import



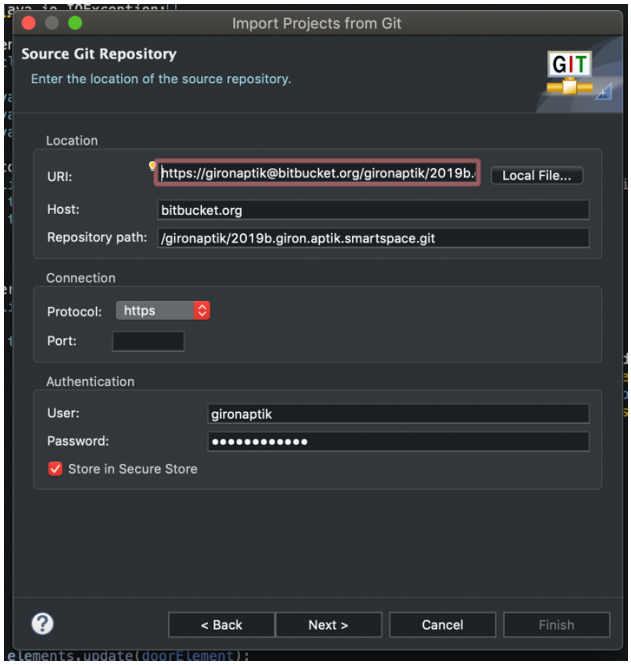
4. Git ->Projects from Git -> Next



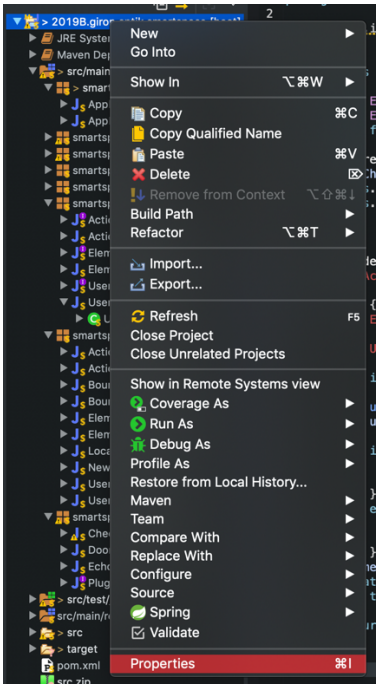
5. Clone URL -> Next



6. Fill the URL you copied in stage 2 and the user and password to the bitucket account.

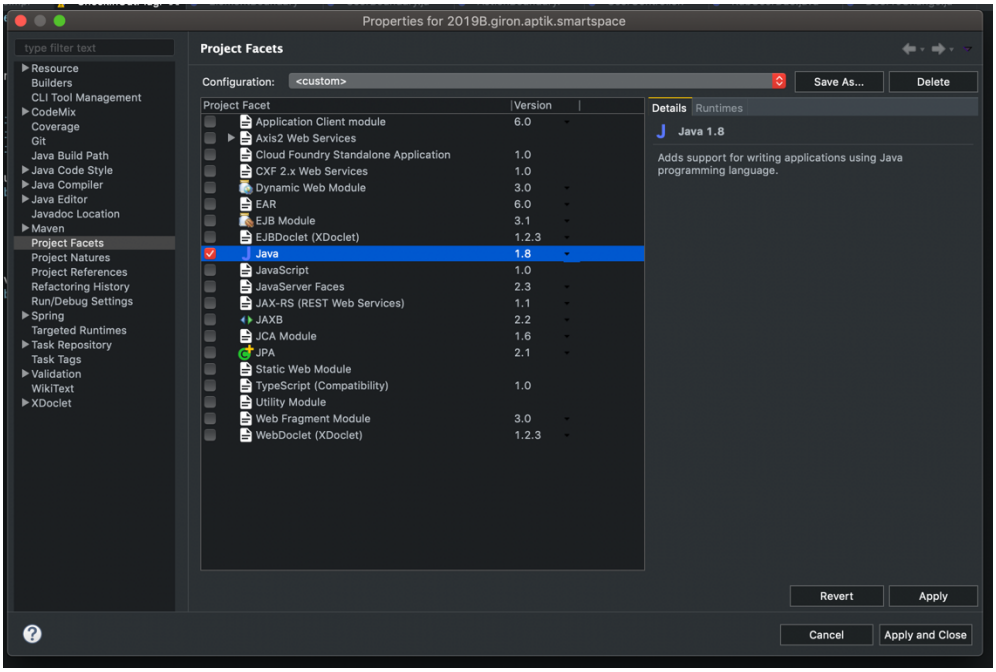


7. Click on Finish
8. Press Right Click on Project Name -> Properties



9. Go to Project Facets and click in convert.

10. Make sure that the Java version is 1.8



11. Click on Apply and Close.
12. Press Right Click on Project Name -> Configure (Similar as stage 8)
13. Press on “Convert To Maven” (It can take a few minutes).
14. Now you have the project in your Eclipse environment and you can run it.
    (Right Click on Applicatio.java -> Run As-> Java Application).

Installing Node.js and Npm (Client Side).

1. Go to Node.js and download LTS version:
   https://nodejs.org/en/

2. Open cmd window and locate the client side folder of our project. (Write cd and drug the folder to the window)

3. Write in the command line :

   npm install -g @angular/cli@latest

4. When its finish writes the next command:

   npm update

   (It’s can take a few minutes)

5. When its finish writes the next command:
   ng serve

6. Now the client is UP and you can see it on http://localhost:4200/

To see the Database from mongoDB cloud:

1. Enter to:
   https://cloud.mongodb.com/

2. Sign in as:
   User Name: integrationcourse@gmail.com
   Password: Afeka2019!