**Project in Integrative Software Engineering.**

**Final Sprint.**

**Date of submission:** 30/5/21.

**Course name:** Integrative Software Engineering.

**Course code:** 10143.

**Lecturer:** Mr. Eisenstein Eyal.

**Presenters:** Iftach Avraham, Arad Pelled, Shahar Hikri, Meron Leshem, Ronen Gurevich, Avihai Adler and Noam Muallem.

Project Requirements Document.

## **1. Introduction:**

The system is a course management system for students and staff. The main purpose of this system is to allow students to register for courses and to give the staff members the ability to manage the courses and their details.

## **1.1 Purpose of the system:**

As discussed before, the purpose of the system is to give staff the ability to manage courses: add, edit or remove a course. And the students have the ability to register and resign from courses. It will also let them watch all the courses they are registered to.

Another purpose of the system is to give the administrative staff the ability to change the participants of the courses.

In addition, the system gives the lecturers the ability to update the grades of the students.

## 

## **1.2 Scope of the system:**

The scope of the system is to be responsible for the registration of the students, managing the courses (add, remove and update) and searching a specific course and its details.

The system is not taking part in the flowing operation of the course, for example: submission of tasks, managing the course studying material and so on.

Every operation that includes payment, scholarship, disciplinary complaint and such as, is not a part of the system.

## **2. Actors and goals:**

Student:

Type of actor: Primary.

Description: Every person that enrolled as a student of the institute.

Goals: Registration, resignation and browse of courses.

Administrative staff:

Type of actor: Primary.

Description: Every person that is working for the relevant institution and

qualified to change and modify courses.

Goals: Add, remove and update courses, sign and remove students

from courses.

Lecturer:

Type of actor: Primary.

Description: Every person that enrolled as a lecturer of a course that takes

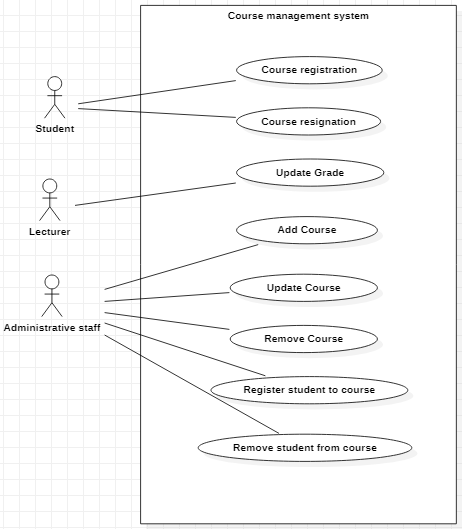
place at the institution.

Goals: Update students grades in courses that the specific lecturer is giving.

## **3. Functional Requirements:**

1. Users will be required to log in to the system before using it.
2. The system will allow the search of a course for every actor.
3. A student can see the courses he is registered to.
4. A student can register to a course only if he’s not registered to.
5. A student can resign from a course he already registered for.
6. An Administrative staff member can add a new course.
7. An Administrative staff member can update an existing course.
8. An Administrative staff member can remove an existing Course.
9. An Administrative staff member can add a student to a Course.
10. An administrative staff member can remove a student from a course.
11. A Lecturer can update the grade of each student attending his courses.

## **3.1 Use case diagram:**



## 

## **3.2 Use cases:**

**Use Case: Course registration**

**Goal: Registering to a course**

**Participating actors: Student**

**Main flow:**

1. Student logs in to the system
2. System verifies the student has an existing user
3. Student searches for desired course by chosen criteria
4. System checks if the course exists
5. System shows course’s details
6. Student registers to the course
7. System checks if there is anything preventing the student from registering to the course
8. System notifies the student of successful registration

**Alternate flow:**

4a. The course does not exist:

4a1. System notifies that the course does not exist.

**Alternate flow:**

7a. The student is already registered to the course:

7a1. System notifies that the student is already registered.

**Use Case: Course resignation**

**Goal: Resigning from a course**

**Participating actors: Student**

**Main flow:**

1. Student logs in to the system
2. System verifies the student has an existing user
3. Student enters all courses page
4. System shows all courses
5. Student chooses the course he wants to quit
6. System shows the chosen course’s details
7. Student presses the ‘quit’ button
8. System try to remove the student from the course
9. System notifies student of successful resignation

**Alternate flow:**

8a. The student is not registered to the course:

8a1. System notifies that the student is not registered.

**Alternate flow:**

3a. The student enters his courses page:

3a1. System shows all of his courses.

3a2. back to main flow at 5.

**Use Case: Add course**

**Goal: adding new course to the system**

**Participating actors: Administrative Staff**

**Main flow:**

1. Admin logs in to the system
2. System verifies Admin has an existing user
3. Admin clicks on ‘Add Course’ button
4. System shows an empty course details form
5. Admin fills course’s details and submits
6. System checks that details are valid
7. System add the course to the courses list
8. System notifies of successful addition

**Alternate flow:**

6a. Details are invalid:  
6a1. System notifies that the entered details are invalid  
6a2. System notifies the admin of invalid details  
6a3. Admin is asked to re-enter the invalid details  
6a4. return to main flow at step 5

**Use case: Update course**

**Goal: Updating any detail of a course**

**Participating actors: Administrative staff**

**Main flow:**

1. Admin logs in to the system
2. System verifies Admin has an existing user
3. Admin asks the system to show existing courses
4. Admin chooses the course he wants to update
5. System shows form with the chosen course’s details
6. Admin changes the desired details
7. Admin clicks ‘save’
8. System verifies that details are valid
9. System updates the course in any relevant instance

**Alternate flow:**

3a. Admin Searches the desired course:

3a1. back to main flow at 5.

**Use case: Remove course**

**Goal: Remove a course from the system**

**Participating actors: Administrative staff**

**Main flow:**

1. Admin logs in to the system
2. System verifies Admin has an existing user
3. Admin asks the system to show all courses
4. Admin chooses the course he wants to remove
5. System shows form with the chosen course’s details
6. Admin clicks ‘remove’
7. System removes the Course from the course list
8. System notifies of successful removal

**Alternate flow:**

3a. Admin Searches the desired course:

3a1. back to main flow at 5.

**Use Case: Register a student to course**

**Goal: Registering a student to a course**

**Participating actors: Administrative staff**

**Main flow:**

1. Admin logs in to the system
2. System verifies Admin has an existing user
3. Admin searches for desired course by chosen criteria
4. System checks if the course exists
5. System shows course’s details
6. Admin clicks ‘register a student’ button
7. System shows register form
8. Admin fills the student’s email
9. System checks if there is anything preventing the student from registering to the course
10. System notifies the admin of successful registration

**Alternate flow:**

4a. The course does not exist:

4a1. System notifies that the course does not exist.

**Alternate flow:**

7a. The student is already registered to the course:

7a1. System notifies that the student is already registered.

**Use Case: Remove a student from course**

**Goal: Remove a student from course**

**Participating actors: Administrative staff**

**Main flow:**

1. Admin logs in to the system
2. System verifies the Admin has an existing user
3. Admin searches for chosen course
4. System shows Course’s details
5. Admin opens list of students registered to the course
6. System shows list of students
7. Admin chooses a student he wants to remove
8. System removes the selected student from the course

**Alternate flow:**

3a. Admin enters all courses page:

3a1. System shows all courses.

3a2. Admin chooses the desired course

3a3. back to main flow at 4.

**Use Case: Update Grade**

**Goal: Update student’s grades in a course**

**Participating actors: Lecturer**

**Main flow:**

1. Lecturer logs in to the system
2. System verifies the lecturer has an existing user
3. Lecturer asks the system to see all courses
4. System shows all the active courses
5. Lecturer chooses the course he wants to change grade in
6. System shows the list of students that learning the course
7. Lecturer submits grades and presses the ‘save’ button
8. System checks that the new grades are valid
9. System saves the new grades

**Alternate flow:**

8a. Lecturer updates a grade that is not between 1-100\includes letters or signs  
8a1. System notifies the Lecturer of invalid grade  
8a2. System asks the Lecturer to enter valid grade  
8a3. Return to main flow at step 7

**Alternate flow:**

3a. Lecturer asks the system to search a specific course:  
3a1. Lecturer searches the course he wants to change grade in  
3a2. Return to main flow at step 6

## **4. Non - Functional Requirements:**

| **Requirement type** | **Requirement Description** | **Requirement Number** |
| --- | --- | --- |
| Usability - U | The system should be easy to learn | 1 |
| Reliability - R |  |  |
| Performance - P | The system should allow multiple students registration at the same time | 2 |
| Supportability - S | System should run on any PC or mobile devices | 3 |

**Technical Document- Appendix.**

**Programming Language:** - Java  
  
**Environments:** - Eclipse  
 - Postman  
 **External Libraries:** - Spring boot  
 - Tomcat  
 - Jackson

- Hibernate

- JPA  
  
**Test:** - JUnit

**Database:** - H2

- MS-SQL  
  
**Team Management:** - Bitbucket  
 - Trello

- Google Drive

- Github

**Client Side:**

- React

- Heroku

**Kanban Boards:**

22.2.21 - sprint 1

****

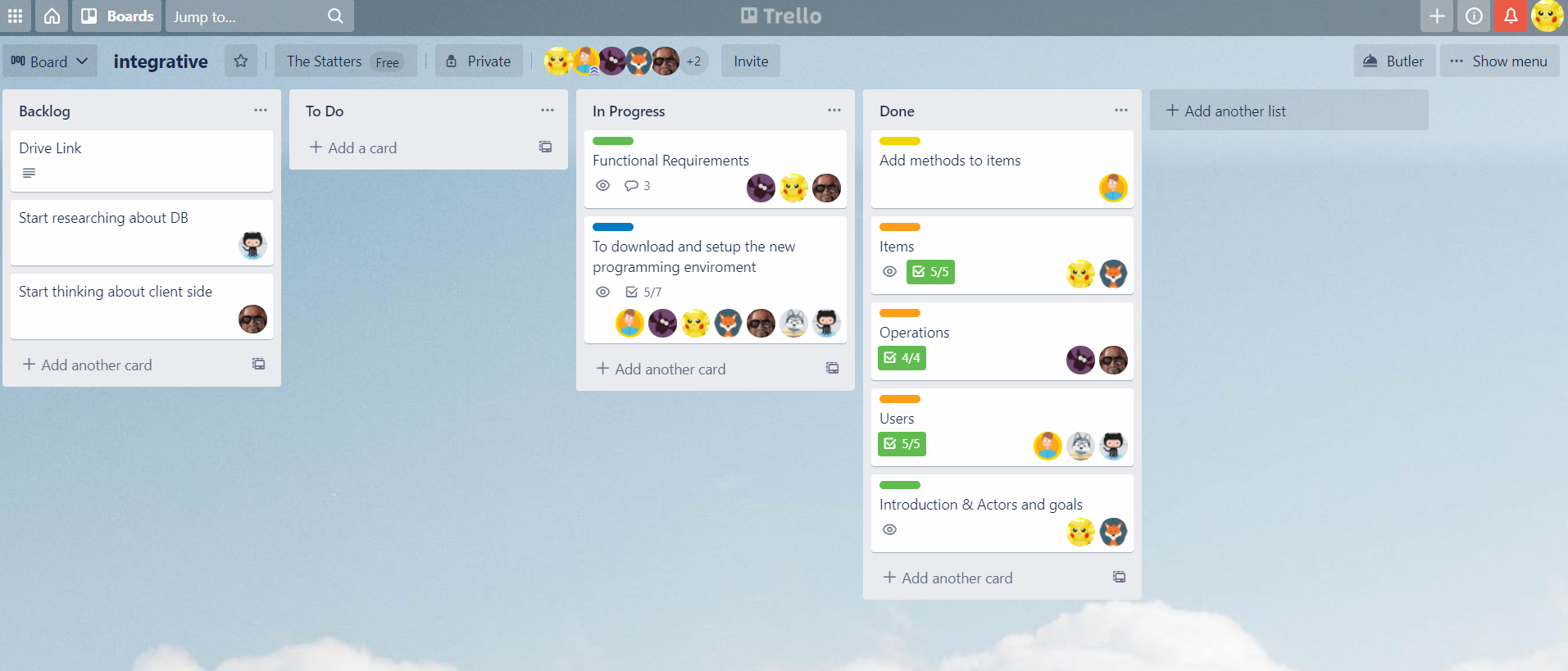
6.3.21 - sprint 1



8.3.21 - sprint 2



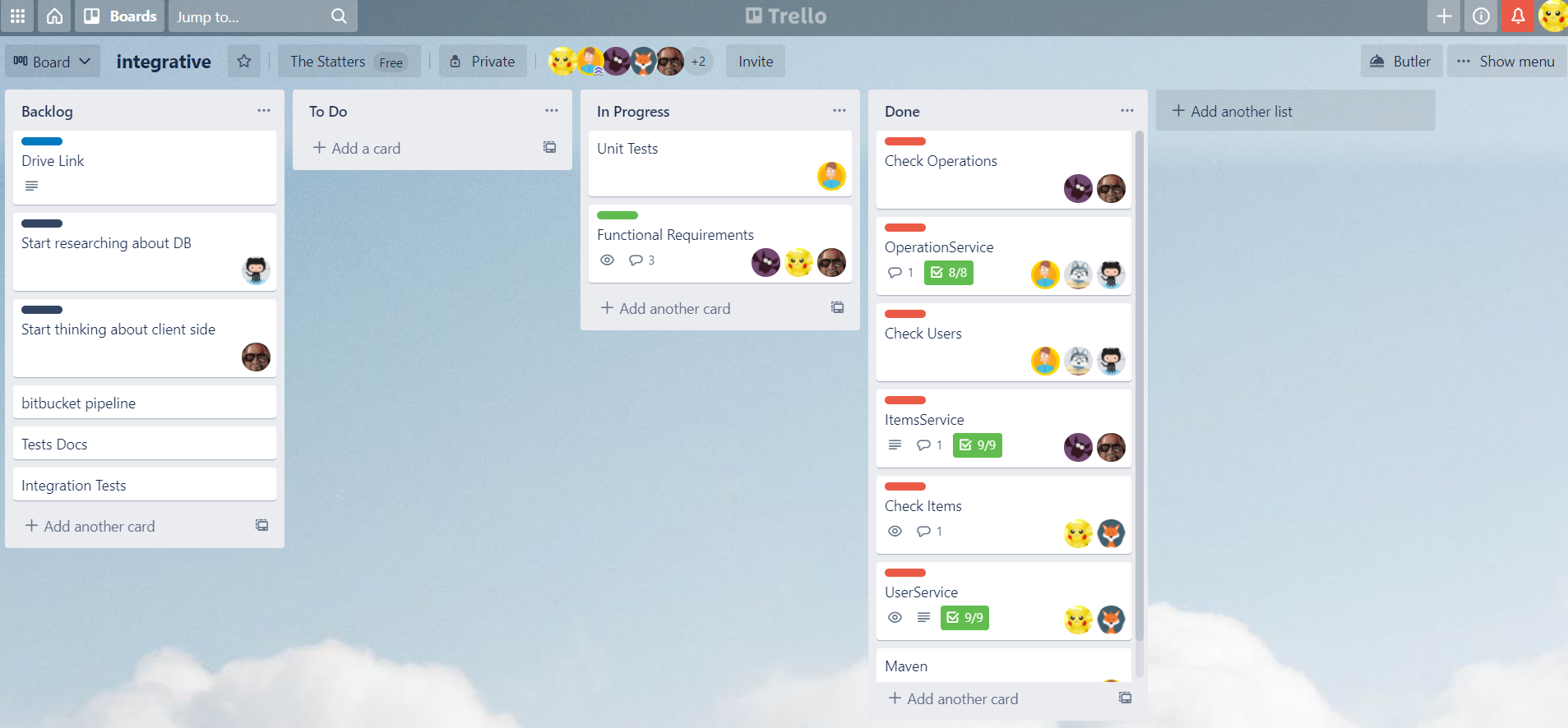
19.3.21 - sprint 2



sprint 3



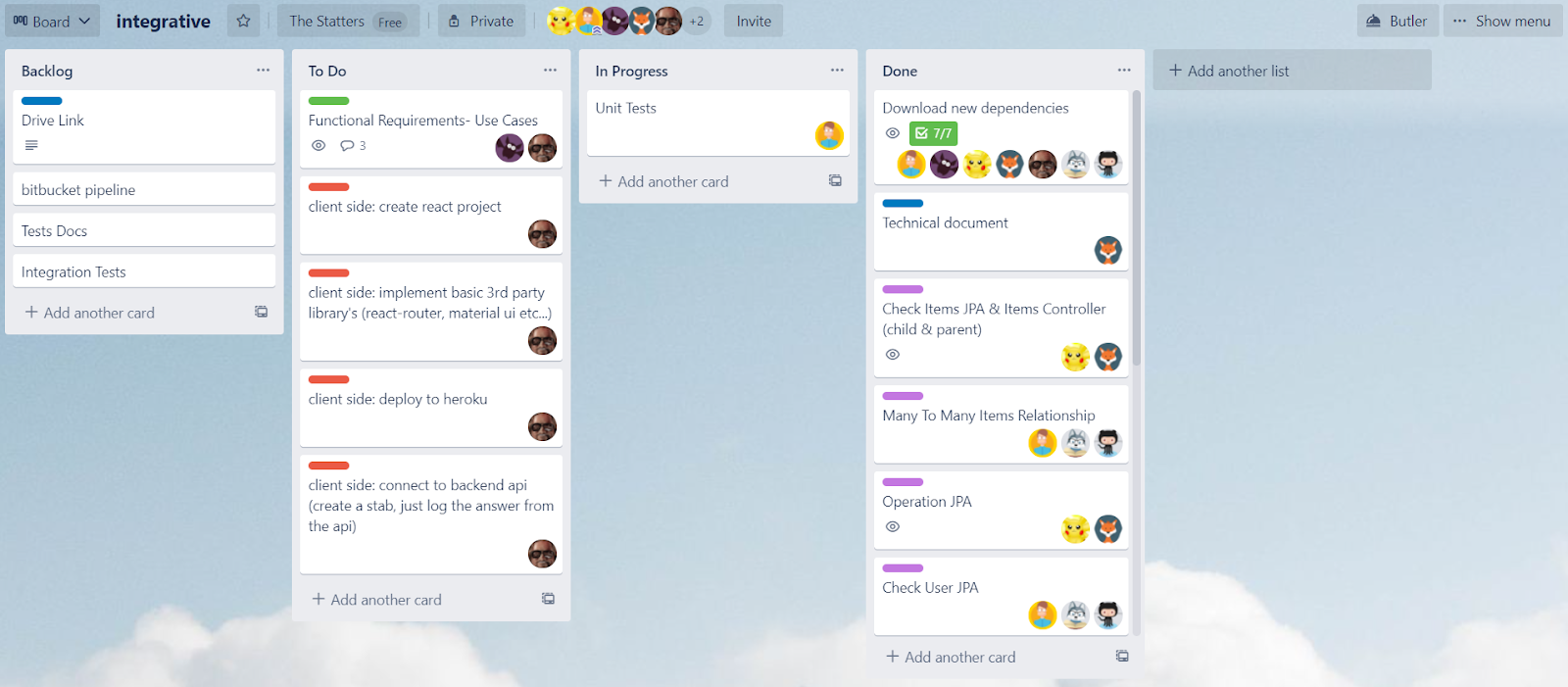
10.4.21 - sprint 3

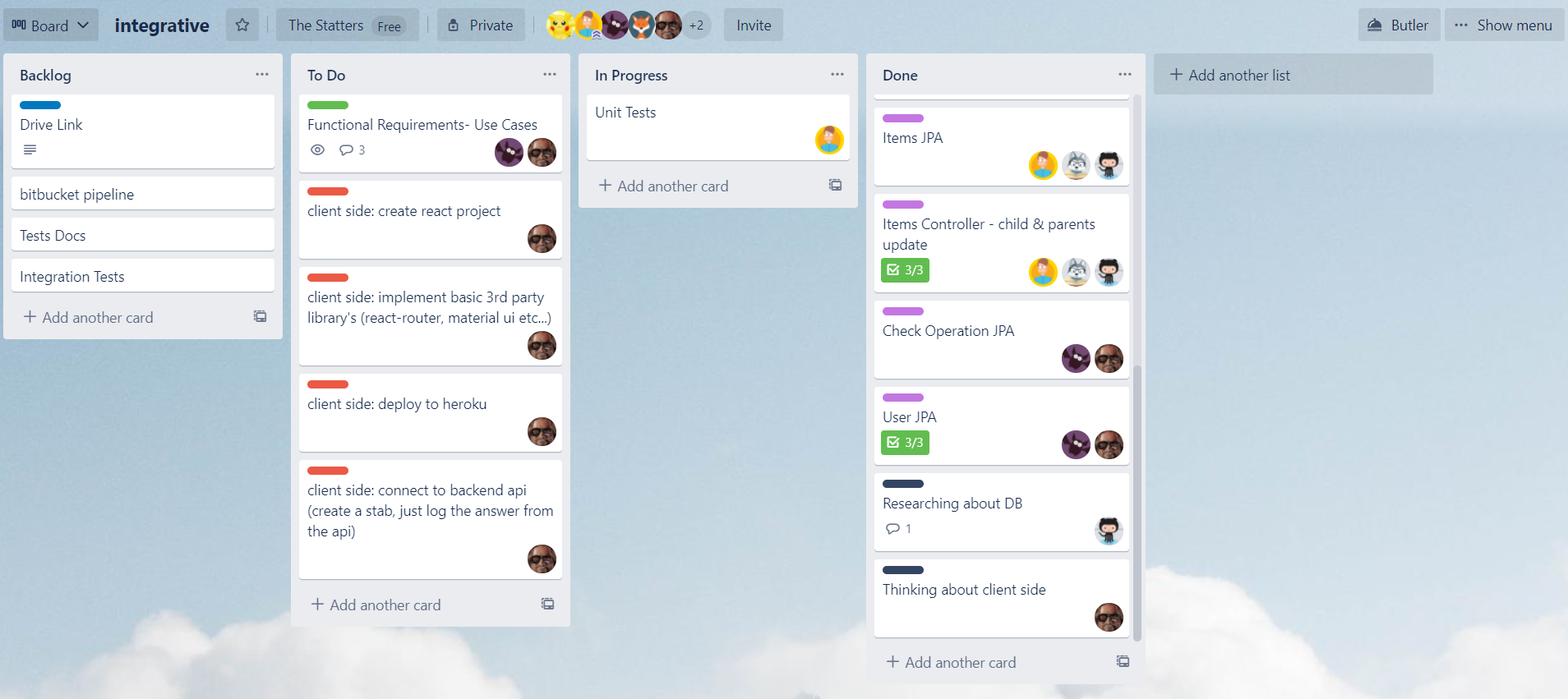


sprint 4

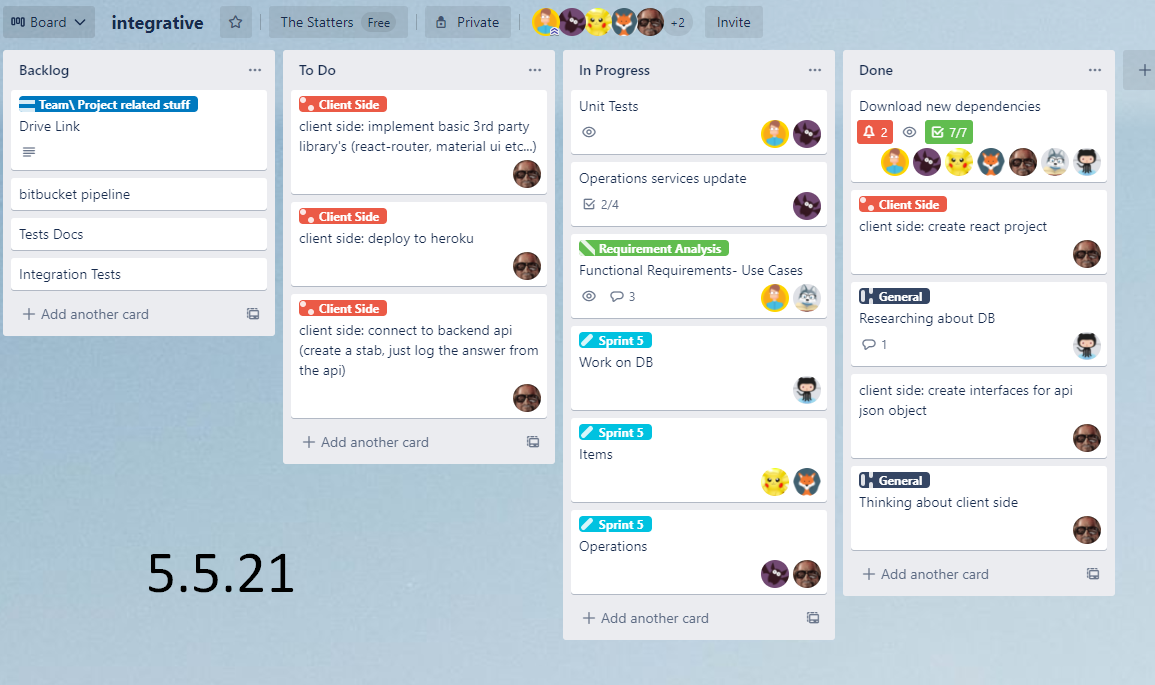


23.4.21 - sprint 4

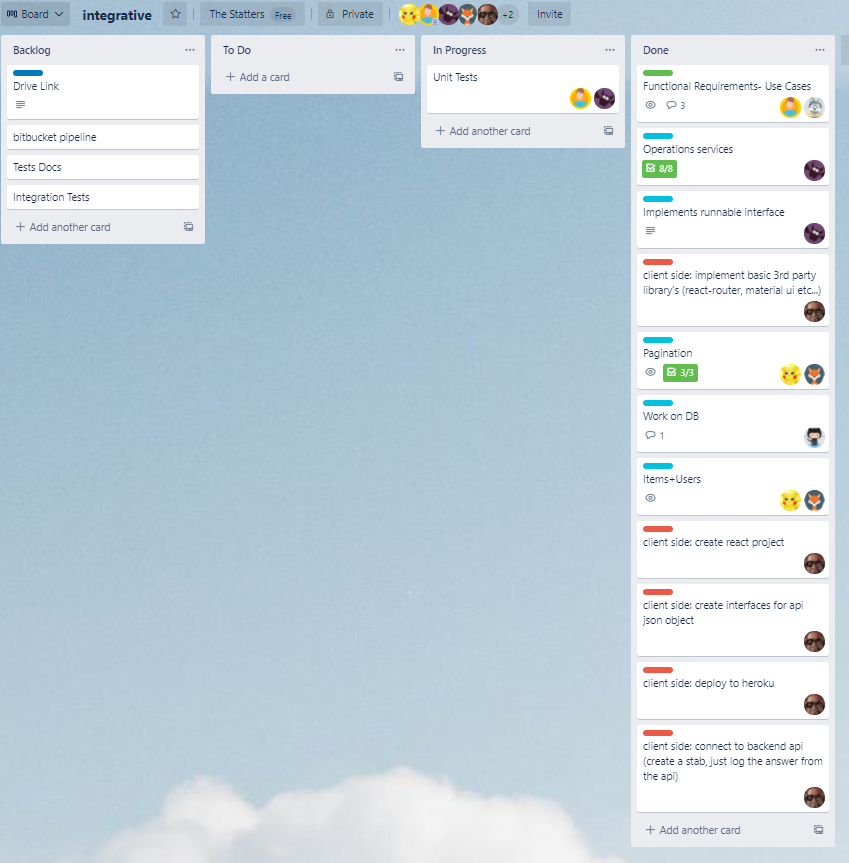




sprint 5



29.5.21- sprint 5

****

**Final Summary:**

* **What went well for the team and should be continued on the next projects of work?**

The mini-teams were our best choice in this project. That worked perfectly. In addition, the swapping of project parts between the mini-teams each sprint was good. It leads to all team members knowing and understanding the whole project and not only a specific part.

* **How could we be improved in team work?**

Full team meetings were hard to manage. Hard to set a time to all members, many times at least one or two members were missing because we had no time for all members to fit in. Also, the meeting itself sometimes not went so well, we have to be more focused and be practical.

* **What we most enjoy?**

The project felt like real job work and that was fun. This was the first time in our degree to get a taste of how it is like real work.

* **What would we do differently if we start this project now?**

We would start to implement our functional requirements earlier, because we didn't manage the time it should take well and that’s why we have not finished all of our original requirements (we finished our updated requirements).

* **How did zoom influence our work?**

With zoom the teamwork was more efficient. Much more easier to set a meeting on zoom rather than face to face. Also, it’s save a lot of time, we could meet spontanly even for a short time meeting just to fixed some minor things.

**List of students:**

* **Iftach Avraham - Team Leader, Team - 316559772 .**
* **Ronen Gurevich - DevOps, Team - 207898511.**
* **Shahar Hikri - DBA, Team - 313237703 .**
* **Arad Pelled - Scrum Master, Product Manager, Team - 31872373.**
* **Meron Leshem - Technical Writer, Team - 205927841 .**
* **Avihai Adler - QA Engineer, Team - 203111166 .**
* **Noam Mualem - UIX Engineer, Team - 203537188 .**

**Sprint Summary:**

* **What went well for the team and should be continued in the next phases of work?**

As the sprint started, we did a group meeting to clarify what remained to finish the project. Then each group member got a part to be responsible for, such as coding, QA, client, and functions document.

The work went well, we worked simultaneously, each member on his tasks. The challenge was to be synchronized and that works really well, we discuss on WhatsApp every day to see everything goes well.

* **What should be improved in team work?**

Group meeting with full members. We are 7 members in our team, so those meetings sometimes are hard to manage, there are too many opinions and each one wants to say his side. This leads sometimes to mess.

* **What problems did the team encounter through this phase of work?**

When we get closer to the final date, we understand that not all of our functional requirements we defined in sprint 1 will be done. Then we discussed what we could implement with hard work or if we couldn't we need to get rid of it, because at final result, we need all of our requirements to work.

* **Why did we not complete all planned work?**

Some of the functions we defined in sprint 1 were too complicated to finish in this deadline, so we had to decide not to do them.