

# **Teamder**

## **Integrative Software Engineering Project**

### **Team members:**

Diana Ukrainsky

Keren Rachev

Anat Moroshek

Rivka Doskoch

Vadim Lazarevich

Eden Harel

### **Due:**

25.05.2022

## Table Of Contents:

<b>Project Requirements Document - Teamder</b>	<b>3</b>
Introduction	3
Actors and goals	3
Functional Requirements	4
Use Case Diagram	6
Use Case Details	7
Non Functional Requirements	15
<b>Technologies</b>	<b>16</b>
<b>Project Summary</b>	<b>17</b>
Kanban Snapshots:	17
Summary	26
<b>Final Sprint report</b>	<b>27</b>

## 1. Project Requirements Document - Teamder

### 1.1. Introduction

1.1.1. Teamder is a social platform that allows its users to connect with each other based on shared interests. The app strives to bring people together so they can meet, share projects and communicate in a way they were not able to before. The main goal of the app is to help people to connect for the first time - find the topic that they find interesting and the people they want to connect with, and then give them a platform where they can communicate to coordinate further meetings, ideas and plans.

#### 1.1.2. Purpose of System

1.1.2.1. Help the user to find a group of people with similar interests.

1.1.2.2. Connect people together and allow basic communication within the app.

#### 1.1.3. Scope of System

### 1.2. Actors and goals

#### 1.2.1. Group Member

1.2.1.1. Primary/Support: Primary

1.2.1.2. Description:

1.2.1.2.1. Main user for the app. Will be able to join/create groups based on personal interests.

1.2.1.3. Goals:

1.2.1.3.1. Swipe through groups

1.2.1.3.2. Join groups

1.2.1.3.3. Leave groups

1.2.1.3.4. Create groups

1.2.1.3.5. Participate in group chat

### 1.2.2. Group Manager

1.2.2.1. Primary/Support: Primary

1.2.2.2. Description:

1.2.2.2.1. Group member with special permissions within a specific group.

1.2.2.3. Goals:

1.2.2.3.1. Edit group permissions

1.2.2.3.2. Add group members

1.2.2.3.3. Remove group members

1.2.2.3.4. Delete group

1.2.2.3.5. Group member goals

## 1.3. Functional Requirements

### 1.3.1. Functional Requirements by Users

#### 1.3.1.1. Group Member:

1.3.1.1.1. Group members should be able to create a new user.

1.3.1.1.2. Group members should be able to join an existing group.

1.3.1.1.3. Group members should be able to create a new group.

1.3.1.1.4. Group member should be able to update his bio.

1.3.1.1.5. Group members should be able to update his preferences.

1.3.1.1.6. Group members should be able to send messages in group chat.

1.3.1.1.7. Group members should be able to read group descriptions.

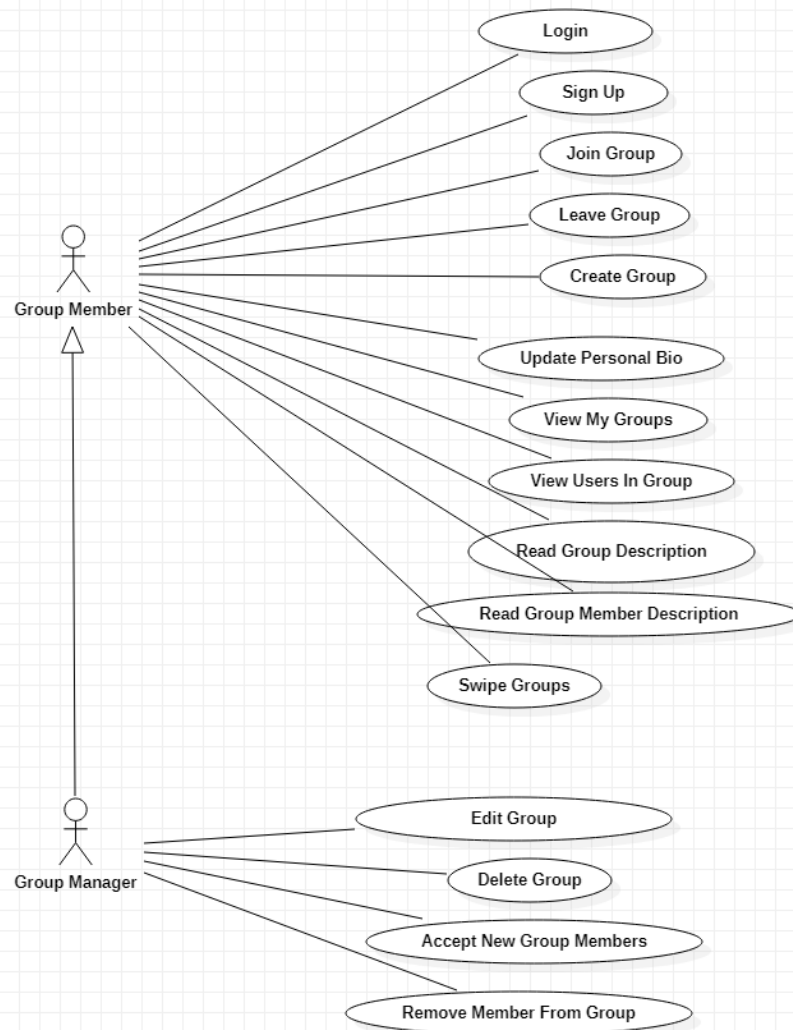
1.3.1.1.8. Group members should be able to search group tags.

1.3.1.2. **Group Manager:**

- 1.3.1.2.1. Group managers should be able to do everything that a group member can.
- 1.3.1.2.2. Group managers should be able to change their group's description.
- 1.3.1.2.3. Group managers should be able to delete their group.
- 1.3.1.2.4. Group managers should be able to accept new members to their group.
- 1.3.1.2.5. Group managers should be able to remove members from their group.
- 1.3.1.2.6. Group managers should be able to change join permissions for their group.

## 1.4. Use Case Diagram

### 1.4.1.



## 1.5. Use Case Details

### 1.5.1.

<b>Use Case Name</b>	Sign Up (Front)
<b>Goals</b>	Create new user
<b>Participating Actors</b>	Group Member, Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"><li>1. Actor enters the app.</li><li>2. Actor chooses 'Sign Up'</li><li>3. Actor enters sign up details.</li><li>4. A new user is made for Actor.</li></ol>
<b>Alternate Workflow</b>	

### 1.5.2.

<b>Use Case Name</b>	Sign In (Front)
<b>Goals</b>	Login to existing user
<b>Participating Actors</b>	Group Member, Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"><li>1. Actor enters the app.</li><li>2. Actor chooses 'Sign In'</li><li>3. Actor enters the sign in details.</li><li>4. User is logged in to the app.</li></ol>
<b>Alternate Workflow</b>	

1.5.3.

<b>Use Case Name</b>	Join Group
<b>Goals</b>	Allow user to join a group
<b>Participating Actors</b>	Group Member, Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects personal interests.</li> <li>3. Actor swipes right for the groups he wants to join.</li> <li>4. Actor is being added to said group.</li> </ol>
<b>Alternate Workflow</b>	<ol style="list-style-type: none"> <li>1. Actor login to the app</li> <li>2. Actor clicks the 'Surprise me' button.</li> <li>3. Actor swipes right for the groups he wants to join.</li> <li>4. Actor is being added to said groups.</li> </ol>

1.5.4.

<b>Use Case Name</b>	Create Group
<b>Goals</b>	Allow user to create a new group
<b>Participating Actors</b>	Group Member, Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects side menu</li> <li>3. Actor selects the 'Start Project' button.</li> <li>4. Actor enters group details.</li> <li>5. Actor selects the 'Create Group' button.</li> </ol>
<b>Alternate Workflow</b>	



1.5.5.

<b>Use Case Name</b>	Update Personal Bio
<b>Goals</b>	Allow user to change his personal bio
<b>Participating Actors</b>	Group Member, Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects the side menu.</li> <li>3. Actor selects 'My Profile'</li> <li>4. Actor selects the 'Edit' button.</li> <li>5. Actor enters new bio.</li> <li>6. Actor selects the 'Update Bio' button.</li> </ol>
<b>Alternate Workflow</b>	

1.5.6.

<b>Use Case Name</b>	Read Group Description
<b>Goals</b>	Allow user to read a group's description
<b>Participating Actors</b>	Group Member, Group Manager
<b>Basic Workflow</b> (Actor views description of groups that he participates in)	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects the 'My Groups' button.</li> <li>3. Actor selects the relevant group.</li> </ol>
<b>Alternate Workflow #1</b> (Actor views description of groups that he participates in)	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects the side menu.</li> <li>3. Actor selects 'My Groups'</li> <li>4. Actor selects the relevant group.</li> </ol>
<b>Alternate Workflow #2</b> (Actor views description of groups that he participates in)	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects 'Chats'</li> <li>3. Actor enters a group chat.</li> <li>4. Actor selects 'Group Description'.</li> </ol>
<b>Alternate Workflow #3</b> (Actor views description of groups that he does not participates in)	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects personal interests.</li> <li>3. Actor clicks on group photo.</li> </ol>

## 1.5.7.

<b>Use Case Name</b>	Read Group Member Description
<b>Goals</b>	Allow user to read another group member's personal description
<b>Participating Actors</b>	Group Member, Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects 'My Groups'</li> <li>3. Actor enters a relevant group's description.</li> <li>4. Actor selects the 'Members' button.</li> </ol>
<b>Alternate Workflow #1 (Actor views member description of members in groups that he participates in)</b>	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects the side menu.</li> <li>3. Actor selects 'My Groups'</li> <li>4. Actor selects the relevant group.</li> <li>5. Actor selects the 'Members' button.</li> </ol>
<b>Alternate Workflow #2 (Actor views member description of members in groups that he participates in)</b>	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects 'Chats'</li> <li>3. Actor enters a group chat.</li> <li>4. Actor selects 'Group Description'.</li> <li>5. Actor selects the 'Members' button.</li> </ol>
<b>Alternate Workflow #3 (Actor views member description of members in groups that he does not participates in)</b>	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects personal interests.</li> <li>3. Actor clicks on group photo.</li> <li>4. Actor selects the 'Members' button.</li> </ol>

1.5.8.

<b>Use Case Name</b>	Swipe Groups
<b>Goals</b>	Allow user to join existing groups by swiping
<b>Participating Actors</b>	Group Member, Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects personal interests.</li> <li>3. Actor selects the 'Swipe!' button.</li> <li>4. Actor swipes right to join group, left to search for the next group.</li> </ol>
<b>Alternate Workflow</b>	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects the 'Surprise Me' button.</li> <li>3. Actor swipes right to join group, left to search for the next group.</li> </ol>

1.5.9.

<b>Use Case Name</b>	Leave Group
<b>Goals</b>	Allow user to leave a group
<b>Participating Actors</b>	Group Member, Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"> <li>1. Actor logs in to the app.</li> <li>2. Actor selects 'My Groups'</li> <li>3. Actor selects the relevant group.</li> <li>4. Actor selects 'Leave Group'</li> <li>5. Actor approves 'Leave Group' in pop-up</li> </ol>
<b>Alternate Workflow</b>	

1.5.10.

<b>Use Case Name</b>	Edit Group
<b>Goals</b>	Allow user to edit group settings
<b>Participating Actors</b>	Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"><li>1. Actor logs in to the app.</li><li>2. Actor selects 'My Groups'.</li><li>3. Actor selects a group.</li><li>4. Actor selects 'Edit'.</li></ol>
<b>Alternate Workflow</b>	

1.5.11.

<b>Use Case Name</b>	Delete Group
<b>Goals</b>	Allow user to delete a group from the app
<b>Participating Actors</b>	Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"><li>1. Actor logs in to the app.</li><li>2. Actor selects 'My Groups'.</li><li>3. Actor selects a group.</li><li>4. Actor selects 'Edit'.</li><li>5. Actor selects 'Delete'</li></ol>
<b>Alternate Workflow</b>	

1.5.12.

<b>Use Case Name</b>	Remove member from group
<b>Goals</b>	Allow user to remove members from existing group
<b>Participating Actors</b>	Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"><li>1. Actor logs in to the app.</li><li>2. Actor selects 'My Groups'.</li><li>3. Actor selects a group.</li><li>4. Actor selects 'Members'.</li><li>5. Actor selects a member.</li><li>6. Actor selects 'Remove'.</li></ol>
<b>Alternate Workflow</b>	

1.5.13.

<b>Use Case Name</b>	Accept new group members
<b>Goals</b>	Allow user to accept new members to an existing group
<b>Participating Actors</b>	Group Manager
<b>Basic Workflow</b>	<ol style="list-style-type: none"><li>1. Actor logs in to the app.</li><li>2. Actor selects 'My Groups'.</li><li>3. Actor selects a group.</li><li>4. Actor selects 'Pending'</li></ol>
<b>Alternate Workflow</b>	

## 1.6. Non Functional Requirements

<u>Requirement Type</u>	<u>Requirement Description</u>
<u>Usability</u>	<u>1. The system would use simplified icons for easy navigation</u> <u>2. The system should be efficient to use with minimal clicks as possible.</u> <u>3. The system will use clean and easy to understand language.</u>
<u>Reliability</u>	
<u>Performance</u>	
<u>Supportability</u>	<u>1. The system would be easy to install and run.</u> <u>2. Only core team members would be able to change the system.</u>

## 2. Technologies

### 2.1. IDEs:

2.1.1. Eclipse

### 2.2. Java Frameworks:

2.2.1. Spring

2.2.2. Jackson: JSON processor

2.2.3. Hibernate

2.2.4. Tomcat

### 2.3. Client:

2.3.1. Android Studio

2.3.2. Retrofit

### 2.4. Testing Utilities:

2.4.1. Postman

2.4.2. Junit

2.4.3. Java Maven

### 2.5. Client Mockup:

2.5.1. Mockplus.com

### 2.6. Databases:

2.6.1. H2

2.6.2. MongoDB

### 2.7. Source Code Repository:

2.7.1. Bitbucket

### 2.8. Task Management:

2.8.1. Trello Kanban board

2.8.2. Whatsapp group

### 2.9. Meeting Hosting:

2.9.1. Zoom

### 2.10. OS:

2.10.1. Windows 10 64 bit

2.10.2. Windows 11 64 bit

### 2.11. Documentation:

2.11.1. Google Docs

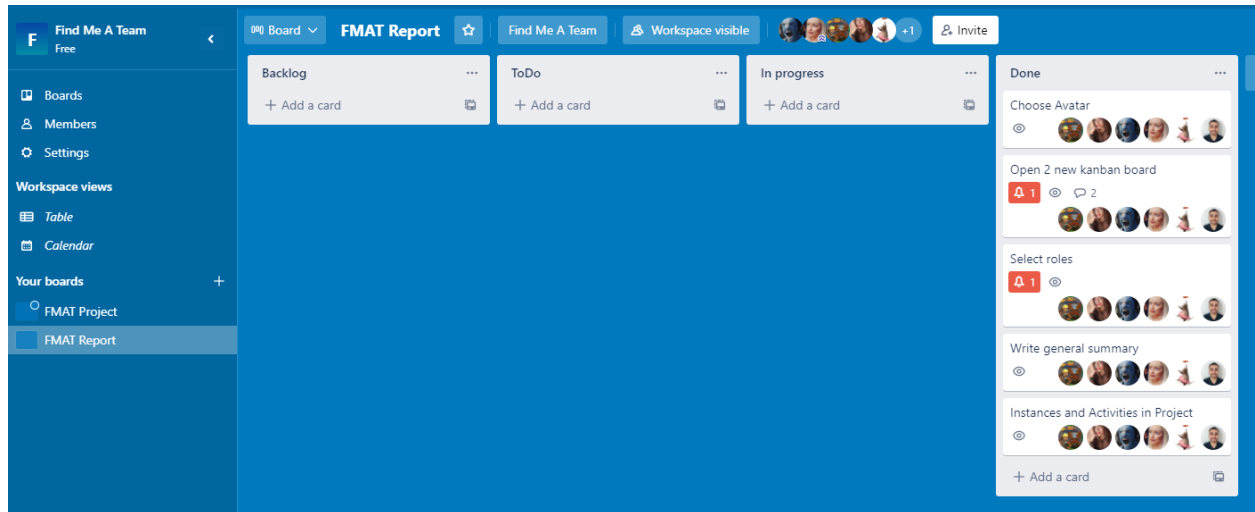
2.11.2. Microsoft Office Word



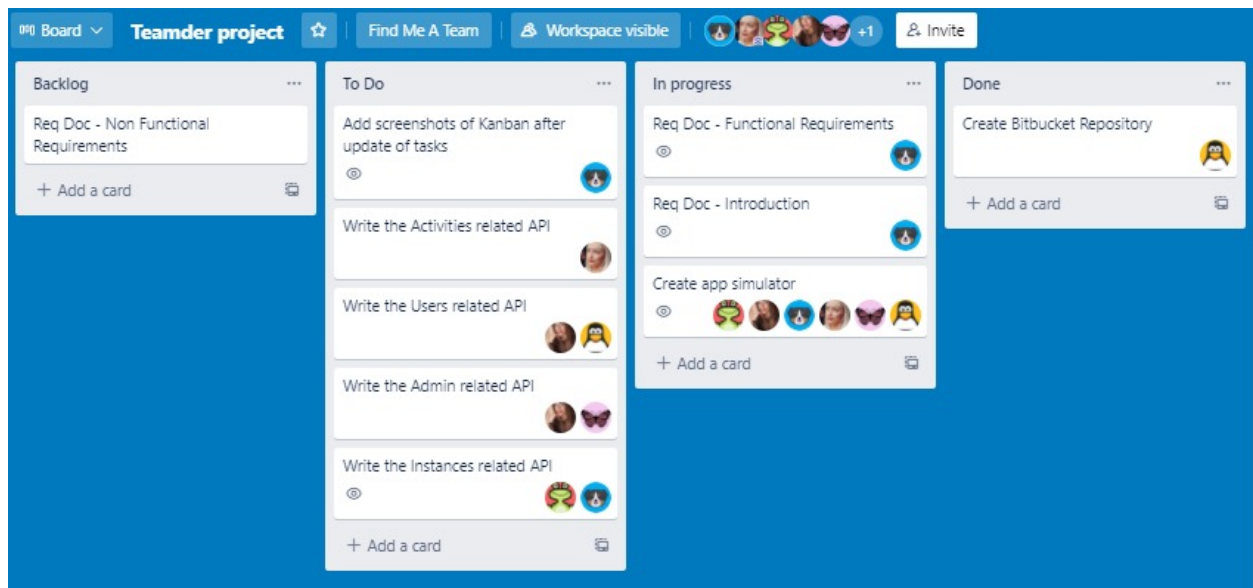
### 3. Project Summary

#### 3.1. Kanban Snapshots:

##### 3.1.1. Date: 25/02/2022



##### 3.1.2. Date: 11/03/2022



### 3.1.3. Date: 20/03/2022

The screenshot shows a Jira Kanban board for the 'Teamder project'. The board is organized into five columns: Backlog, To Do, In progress, Done, and Done. The 'Backlog' column contains one card: 'Req Doc - Non Functional Requirements'. The 'To Do' column is empty. The 'In progress' column contains three cards: 'Req Doc - Functional Requirements', 'Req Doc - Introduction', and 'Create app simulator'. The 'Done' column contains 15 cards, including 'Add screenshots of Kanban after update of tasks', 'Write the Instances related API', 'Write the Admin related API', 'Write the Users related API', 'POST function for ActivityBoundary - should create new ActivityId', 'PUT function for InstanceBoundary should get InstanceBoundary', 'Add option for automatic ID generator', 'Rename DemoController to Controller', 'Rename DELETEUser function to DELETEAllUsers', 'Req Doc - Actors and goals', 'Delete Message class', 'GET all instances function should not get any path variables in the URL', 'Write the Activities related API', 'POST function for ActivityBoundary - return InstanceBoundary object that has the same Id as the InstanceBoundary that the activity used', 'PUT function for UserBoundary should update fields of an UserBoundary object', 'Add comments', 'PUT function for InstanceBoundary should update fields', 'POST function for ActivityBoundary - change the parameter name', 'Separate functions', 'For Timestamps use Date functions to get the real current timestamp', 'Do manual testing to the API interface', 'Change PUT and DELETE functions to return void and not String', and 'Create Bitbucket Repository'. Each card has a title, a description, and a list of assignees. The board also features a top navigation bar with 'Board', 'Teamder project', 'Find Me A Team', 'Workspace visible', and an 'Invite' button.

Board: Teamder project

Find Me A Team | Workspace visible | +1 | Invite

**Backlog**

- Req Doc - Non Functional Requirements
- + Add a card

**To Do**

- + Add a card

**In progress**

- Req Doc - Functional Requirements
- Req Doc - Introduction
- Create app simulator
- + Add a card

**Done**

- Add screenshots of Kanban after update of tasks
- Write the Instances related API
- Write the Admin related API
- Write the Users related API
- POST function for ActivityBoundary - should create new ActivityId
- PUT function for InstanceBoundary should get InstanceBoundary
- Add option for automatic ID generator
- Rename DemoController to Controller
- Rename DELETEUser function to DELETEAllUsers
- Req Doc - Actors and goals
- Delete Message class
- GET all instances function should not get any path variables in the URL
- + Add a card

**Done**

- Write the Activities related API
- POST function for ActivityBoundary - return InstanceBoundary object that has the same Id as the InstanceBoundary that the activity used
- PUT function for UserBoundary should update fields of an UserBoundary object
- Add comments
- PUT function for InstanceBoundary should update fields
- POST function for ActivityBoundary - change the parameter name
- Separate functions
- For Timestamps use Date functions to get the real current timestamp
- Do manual testing to the API interface
- Change PUT and DELETE functions to return void and not String
- Create Bitbucket Repository
- + Add a card

### 3.1.4. Date: 25/03/2022

The image shows a Jira board with the following columns and tasks:

- Backlog**
  - Req Doc - Non Functional Requirements
  - Change all Set functions to return void
  - Change ActivityId and InstanceId classes to something more generic because they both have the same attributes
  - Search if all of the boundary objects have attributes with name just like the jsons in the rest API document
  - Look if all the classes start with big letters
  - Write Integration tests using Junit
  - Write business logic tests
  - Code review in POSTInstance and PUTInstance
  - Code review in User Controller and Admin Controller after implementing the flow
  - Code review in ExportAllActivities and DeleteAllUsers
  - Code review in GETInstance and GETAllInstances
- Backlog**
  - Implement test in AdminTests function testDeleteAllUsersActuallyDeletesAllUsers()
  - Code review in DeleteAllActivities and ExportAllUsers
  - Implement test in AdminTests function testDeleteAllActivitiesActuallyDeletesAllActivities()
  - Implement test in AdminTests function testDeleteAllInstancesActuallyDeletesAllInstances()
  - AdminTests: test Export All Users Actually Exports AllUsers
  - Admin Tests: test Export All Activities Actually Exports All Activities
  - Code review in DeleteAllInstances
- To Do**
  - Technical - Update Java libraries
  - Update app domain
  - Change Functions names in controllers to what they do and not what REST command is used
  - Change domain strings to use the spring configuration file
  - Create Logic package under job
  - Class Implementations - Data
  - Class Implementations - ActivitiesService
  - Class Implementations - InstancesService
  - Class Implementations - UsersService
  - InstancesServiceMockup - getSpecificInstance
  - InstancesServiceMockup - updateInstance
  - InstancesServiceMockup - getAllInstances
- To Do**
  - Implement Flow - Activity Controller
  - InstancesServiceMockup - deleteAllInstances
  - ActivitiesServiceMockup - invokeActivity
  - Move all packages to be under job package
  - UsersServiceMockup - getAllUsers
  - ActivitiesServiceMockup - deleteAllActivities
  - Implement Flow - Admin Controller
  - ActivitiesServiceMockup - getAllActivities
  - UsersServiceMockup - deleteAllUsers
  - UsersServiceMockup - updateUser
  - UsersServiceMockup - login
  - Change from Vector to List
  - UsersServiceMockup - createUser
  - Implement Flow - User Controller
  - Implement Flow - Instance Controller
  - InstancesServiceMockup - createInstance
- In progress**
  - Req Doc - Functional Requirements
  - Req Doc - Introduction
  - Write Sprint 3 report
  - Create app simulator
- Done**
  - + Add a card

### 3.1.5. Date: 05/04/2022

The screenshot displays a Kanban board for a project named "Teamdr project". The board is organized into columns representing different stages of work: Backlog, To Do, In progress, Done, and Done. Each column contains task cards with titles, descriptions, and progress indicators. The "In progress" column is the widest and contains the most cards, including a detailed card for "Create app simulator" which shows a code snippet for an "ActivitiesService" interface and its implementation. The "Done" columns contain cards for completed tasks, such as "Look if all the classes start with big letters" and "InstancesServiceMockup - createInstance". The board also features a top navigation bar with options like "Board", "Find Me A Team", "Workspace visible", and a "Share" button.

**Backlog**

- Write business logic tests
- Add class IdGenerator consisting the attribute of AtomicLong to use in project(see description)

**To Do**

- Req Doc - Non Functional Requirements

**In progress**

- Req Doc - Functional Requirements
- Write Sprint 3 report
- Req Doc - Introduction
- Create app simulator
- Technical - Update java libraries
- Class Implementations - ActivitiesService
- Class Implementations - Data
- Class Implementations - UserService

**Done**

- Look if all the classes start with big letters
- Change Functions names in controllers to what they do and not what REST command is used
- Implement Flow - User Controller
- UserServiceMockup - createUser
- Change from Vector to List
- UserServiceMockup - login
- UserServiceMockup - deleteAllUsers
- Implement Flow - Admin Controller
- UserServiceMockup - getAllUsers
- Move all packages to be under job package
- ActivitiesServiceMockup - invokeActivity
- InstancesServiceMockup -

**Done**

- InstancesServiceMockup - createInstance
- Implement Flow - Instance Controller
- Code review in ExportAllActivities and DeleteAllUsers
- Search if all of the boundary objects have attributes with name just like the jsons in the rest API document
- Change ActivityId and InstanceId classes to something more generic because they both have the same attributes
- Change all Set functions to return void
- Code review in DeleteAllActivities and ExportAllUsers
- Code review in DeleteAllInstances
- Admin Tests: test Export All Activities Actually Exports All Activities
- AdminTests: test Export All Users Actually Exports AllUsers
- Implement test in AdminTests function testDeleteAllUsersActuallyDeletesAllUsers()

### 3.1.6. Date: 08/04/2022

Backlog

Write business logic tests

Add tests of exceptions

Add exceptions classes with relevant exception thrown

Discuss about Instance attributes (See details)

Choose DB

Messages Page

Group Participants

Android Activities - messages, group participants, group chat page (See screenshot attached)

Main Page

Android Activities - Main page and swipe page and group description (See screenshot attached)

Personal Preferences

Android Activities - Main page and swipe page and group description (See screenshot attached)

+ Add a card

To Do

Sign Up Page 1

Sign Up Page 2

Android Activities - Sign In & Sign up pages (See screenshot attached)

Delete AtomicLong that is unused

Download new dependencies

Replace Atomic Long implementation to UUID

Create class InstanceDao that implements the CrudRepository interface(see description)

Create and implement InstancesService.java

Search User by user entity id

Req Doc - Non Functional Requirements

+ Add a card

To Do

in update user there is no need to convert the boundary to entity the because null values can be sent and the converter doesn't check for them.

Check that there is validation in the project before submission

Write sprint 4 report

Create and implement UserService/pa class

Create class UserDao that implements the CrudRepository interface

activityServiceMockup change the check if domain is valid to check if equals to the domain variable in the class

Create class ActivityDao that implements the CrudRepository interface(see description)

Create and implement ActivitiesService.java

Check if DOMAIN is read from the properties file (Not hardcoded)

+ Add a card

In progress

+ Add a card

Done

+ Add a card

21

### 3.1.7. Date: 26/04/2022

**Backlog**

- Write business logic tests
- Add tests of exceptions
- Add exceptions classes with relevant exception thrown
- Discuss about Instance attributes (See details)
- Choose DB
- Main Page
- Android Activities - Main page and swipe page and group description (See screenshot attached)
- Android Activities - My Group page, Side-bar and bottom nav (See screenshot attached)

**To Do**

- Android Activities - Sign In & Sign up pages (See screenshot attached)
- Android Activities - Personal preferences and my profile page (See screenshot attached)
- Create Group Page 1
- Create Group Page 2

**In progress**

**Done**

- Delete AtomicLong that is unused
- Download new dependencies
- Replace Atomic Long implementation to UUID
- Create class InstanceDao that implements the CrudRepository interface(see description)
- Create and implement InstanceServicePa
- Create class UserDao that implements the CrudRepository interface
- Create class ActivityDao that implements the CrudRepository interface(see description)
- Create and implement ActivitiesServicePa
- Search User by user entity id
- Req Doc - Non Functional Requirements
- Save User role as a string in database

### 3.1.8. Date: 29/04/2022

**Teamder project** | Find Me A Team | Workspace visible | Invite

**Backlog**

- Write business logic tests
- Add tests of exceptions
- Add exceptions classes with relevant exception thrown
- Discuss about Instance attributes (See details)

**Android Activities - Main page and swipe page and group description (See screenshot attached)**

**Android Activities - My Group page, Side-bar and bottom nav (See screenshot attached)**

**Backlog**

- Messages Page
- Group Participants
- Android Activities - messages, group participants, group chat page (See screenshot attached)
- Personal Preferences
- Android Activities - Personal preferences and my profile page (See screenshot attached)
- Create Group Page 1
- Create Group Page 2
- Android Activities - Create group pages (See screenshot attached)

**To Do**

- instances 1
- instances 3
- update admin api 1
- update admin api 2
- update admin api 3

**In progress**

- Rewrite non-functional requirements
- instances 2
- Sign Up Page 1
- Sign Up Page 2
- Android Activities - Sign In & Sign up pages (See screenshot attached)

**Done**

- Choose DB
- Initial integration with MongoDB
- Create MongoDB account for team, Cluster and permissions.
- Create EnhancedInstanceService Class
- Create Custom Exception which returns 404 status code.
- Create Custom Exception which returns NOT ALLOWD status code for permissions checks.

### 3.1.9. Date: 10/05/2022

The image displays a Jira board with a Kanban workflow. The board is organized into four main columns: Backlog, To Do, In progress, and Done. Each column contains several cards representing tasks or user stories. The cards are color-coded and include a title, a brief description, and a 'Add a card' button at the bottom. The tasks are related to an Android application project, including tasks like 'Implement Delete/Member function', 'Create Enum ActivityType', and 'Create MongoDB account'. The board also shows a 'Backlog' column with a list of tasks and a 'To Do' column with a list of tasks. The 'In progress' column is currently empty. The 'Done' column contains several completed tasks, including 'update admin api 1', 'update admin api 2', and 'update admin api 3'. The board is set against a blue background.

**Backlog**

- Fix JUnit test suite
- Add tests of exceptions
- Android Activities - Main page and swipe page and group description (See screenshot attached)
- Android Activities - My Group page. Side-bar and bottom nav (See screenshot attached)

**To Do**

- Implement Delete/Member function
- Implement EditGroup function
- Implement Join group function
- Implement DeleteGroup function
- Discuss about Instance attributes ( See details )
- Android client

**In progress**

- Implement Accept/Join/Member

**Done**

- update admin api 1
- update admin api 2
- update admin api 3
- Create UserServiceEnhanced class
- Create Custom Exception which returns NOT ALLOWED status code for permissions checks.
- Create Enum ActivityType: JOIN\_GROUP, EXIT\_GROUP, ACCEPT\_NEW\_MEMBER, DELETE\_GROUP, DELETE\_MEMBER
- Create Enum InstanceType: USER, GROUP
- Rewrite non-functional requirements
- Write Sprint 5 report
- Instances 1
- Instances 2
- Instances 3



### 3.1.10. Date: 24/05/2022

The screenshot displays a Jira Kanban board for the 'Teamder project'. The board is organized into columns representing the workflow stages: Backlog, Backlog, To Do, In progress, Done, and Done. The first 'Backlog' column contains two cards: 'Add tests of exceptions' and 'Fix Junit test suite'. The 'To Do' column is empty. The 'In progress' column is empty. The first 'Done' column contains several cards, including 'Swipe groups', 'Read group description', 'Android client', 'View Users in groups & View my groups', 'Create Group Page 1', 'Create Group Page 2', 'Greet 1 Almost finished', 'Android Activities - Create group pages (See screenshot attached)', 'Sign Up Page 1', 'Sign Up Page 2', 'Android Activities - Sign in & Sign up pages (See screenshot attached)', 'Make final presentation', 'Make final Kanban report', and 'Join Group & Leave Group'. The second 'Done' column contains cards for 'update admin api 2', 'Create UserServiceEnhanced class', 'Create Custom Exception which returns NOT ALLOWD status code for permissions checks.', 'Accept new group members & Remove member from group', 'Write final Sprint report', 'Create Custom Exception which returns 404 status code.', 'Main Page', 'Android Activities - Main page and swipe page and group description (See screenshot attached)', 'Create EnhancedInstanceService Class', 'Create MongoDB account for team, Cluster and permissions.', 'Implement DeleteGroup function', and 'Implement Join group function'. Each card includes a title, a description, a status, and a list of assignees. The board also features a 'Board' dropdown, a 'Teamder project' title, a 'Workspace visible' indicator, and a 'Share' button.

### **3.2. Summary**

#### **3.2.1. General:**

3.2.1.1. The Teamder project was a group effort from start to finish. The team was well coordinated and enthusiastic to meet the course's requirements and develop software on a scale we had never done before.

#### **3.2.2. What went well and should be preserved for future projects:**

3.2.2.1. Role assignment allowed each team member to use his/hers strengths for the benefit of the team.

#### **3.2.3. What can be improved for future projects:**

3.2.3.1. Setting realistic expectations for the amount and complexity of app functionality we want to implement for the first stages of development.

#### **3.2.4. What did we enjoy the most:**

3.2.4.1. Positive feedback for hard work and proper implementation of newly learned technologies.

#### **3.2.5. What would we have done differently, given we had today's experience:**

3.2.5.1. Get front-end work started earlier and in parallel to back-end work.

#### **3.2.6. How did remote work affect our project:**

3.2.6.1. Allowed us to work at different times, as team members have different schedules - which was extremely helpful.

#### 4. Final Sprint report







### Integrative Software Engineering

Project Name: Teamder

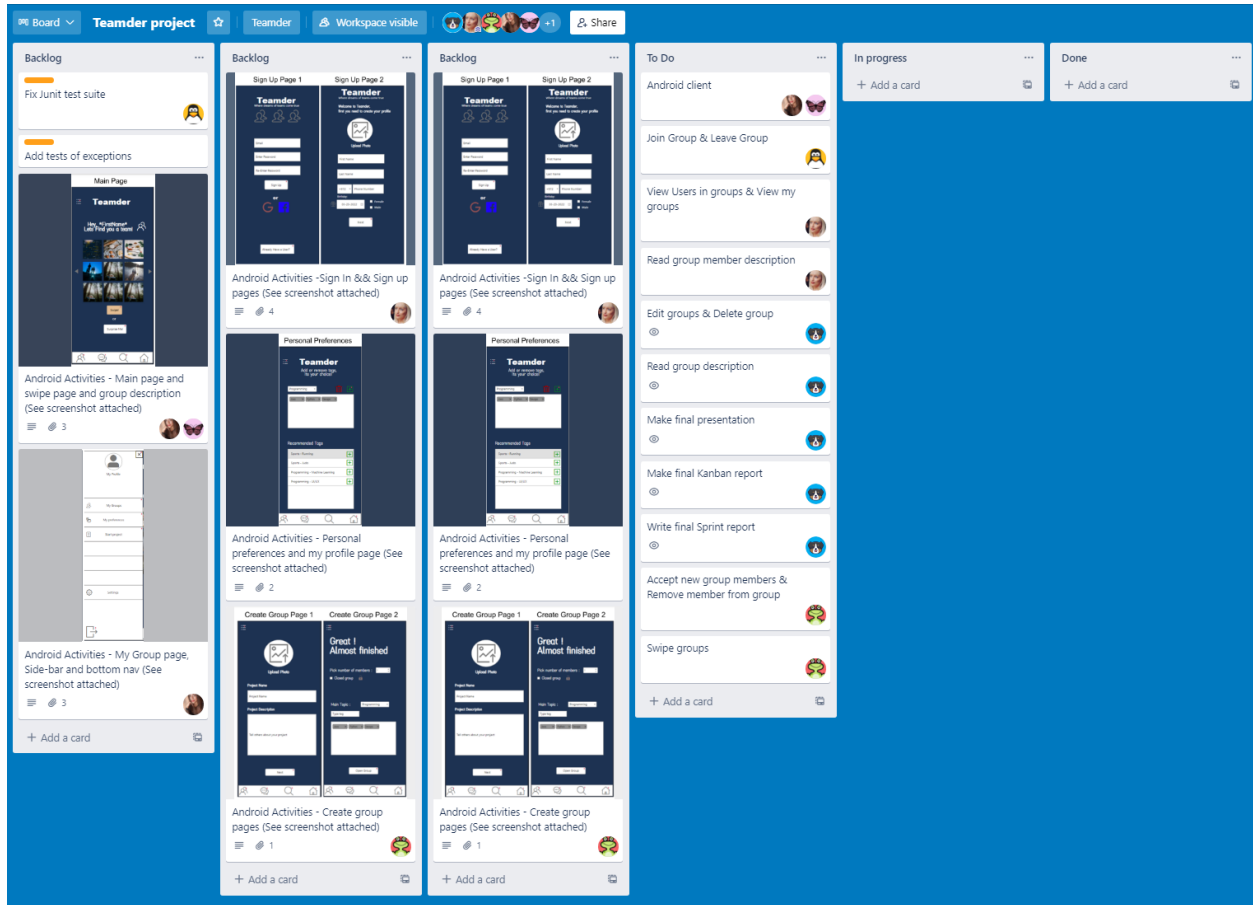
Sprint 6

Due: 25/05/2022

#### 4.1. Students:

Full Name	ID	Avatar	Roles
Diana Ukrainsky	321268112		Team Leader, TEAM
Keren Rachev	318638129		DBA, SCRUM Master, TEAM
Anat Moroshek	312174345		System Architect, TEAM
Rivka Daskoch	324317205		UI/X, TEAM
Vadim Lazarevich	317785053		QA Engineer, DevOps, TEAM
Eden Harel	205518178		Technical Writer, Product Owner, TEAM

## 4.2. Kanban boards:



Taken: 13/05/2022

Board Teamder project Teamder Workspace visible +1 Share

Backlog

Add tests of exceptions

Fix Junit test suite

+ Add a card

Backlog

+ Add a card

To Do

+ Add a card

In progress

+ Add a card

Done

Swipe groups

Read group description

Android client

View Users in groups & View my groups

Create Group Page 1 Create Group Page 2

Android Activities - Create group pages (See screenshot attached)

Android Activities - Sign in & Sign up pages (See screenshot attached)

Make final presentation

Make final Kanban report

Join Group & Leave Group

+ Add a card

Done

update admin api 2

Create UserServiceEnhanced class

Create Custom Exception which returns NOT ALLOWD status code for permissions checks.

Accept new group members & Remove member from group

Write final Sprint report

Create Custom Exception which returns 404 status code.

Main Page

Android Activities - Main page and swipe page and group description (See screenshot attached)

Create EnhancedInstanceService Class

Create MongoDB account for team, Cluster and permissions.

Implement DeleteGroup function

Implement Join group function

+ Add a card

Taken: 24/05/2022

#### **4.3. General summary of work:**

##### **4.3.1. For this sprint we finished the following tasks:**

- 4.3.1.1. Updated Activities: Updated and added activities to support our app.
- 4.3.1.2. Client: Created a running client with partial functionality that includes : Sign in, Sign up, Group creation, Personal bio edit.
- 4.3.1.3. Documentation: Updated and reworked on our documents to better fit our current development state.

##### **4.3.2. What went well**

- 4.3.2.1. The team was well coordinated in task management, therefore there were little to none issues with pushing, pulling, etc.
- 4.3.2.2. Team members are working fast and pushing updates early on.
- 4.3.2.3. All team members made an effort to join the latest meetings in order to have better division of tasks.

##### **4.3.3. What should be improved:**

- 4.3.3.1. Planning ahead - while we were working in sprints, we did not have an endpoint in mind, or a long term schedule and therefore we didn't really know how we wanted our project to look by the end of the last sprint.

##### **4.3.4. What problems did the team encounter:**

- 4.3.4.1. As the last sprint was mainly touch-ups for the tasks we finished already and front-end dev, we did not encounter any major problems.

##### **4.3.5. Why did we not complete all planned work:**

- 4.3.5.1. All of the planned work was finished for this sprint, however we found that we have not left enough time for our client development, therefore it is partially implemented for this sprint.