**Use cases**

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Login | |
| **Actors** | User | |
| **Parameters** | ID, Password | |
| **Preconditions** |  | |
| **Actions** | The user starts the system  The system presents a “Login Screen” to the user  The user inserts his ID and Password  The user presses the “Login” button  The system verifies the user’s input | |
| **Result** | The system grants permission to log in or raises the relevant error | |
| **Data** | | **Result** |
| The user is registered and inserts his ID and the correct password | | Pass |
| The user is not registered | | Fail |
| The user is already logged in | | Fail |
| The user inserts his ID and an incorrect password | | Fail |
| The user inserts incorrect ID and the correct password | | Fail |
| The user inserts ID that is shorter or longer than a valid ID number | | Fail |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name** | | Register a new user | | | |
| **Actors** | | Project Manager / Contractor | | | |
| **Parameters** | | User ID | | | |
| **Preconditions** | | The project manager / contractor is logged in | | | |
| **Actions** | | The project manager / contractor is typing the user’s ID | | | |
| **Result** | | A new user is created | | | |
| **Data** | | | | **Result** | |
| The project manager / contractor types a valid ID | | | | Pass | |
| The project manager / contractor types a wrong ID | | | | Fail | |
| **Use Case Name** | Assign a role to a user | | | |
| **Actors** | Project Manager / Contractor | | | |
| **Parameters** | User ID | | | |
| **Preconditions** | The project manager / contractor is logged in | | | |
| **Actions** | The project manager / contractor is typing the user’s ID and assigning a new role to it | | | |
| **Result** | The user is assigned to a new role | | | |
| **Data** | | | **Result** | |
| The project manager / contractor types a valid ID and choosing a role | | | Pass | |
| The project manager / contractor types a wrong ID | | | Fail | |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Unassign a role to a user | |
| **Actors** | Project Manager / Contractor | |
| **Parameters** | User ID | |
| **Preconditions** | The project manager / contractor is logged in  The user has a role | |
| **Actions** | The project manager / contractor is typing the user’s ID and unassigning its role from it | |
| **Result** | The user is unassigned from its role | |
| **Data** | | **Result** |
| The project manager / contractor types a valid ID and choosing a role | | Pass |
| The project manager / contractor types a wrong ID | | Fail |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Create a new project | |
| **Actors** | Contractor | |
| **Parameters** | Project name | |
| **Preconditions** | The contractor is logged in | |
| **Actions** | The contractor clicks the “Create a new project” button and enters its name | |
| **Result** | A new project is created | |
| **Data** | | **Result** |
| The contractor clicks the “Create a new project” button and enters a valid name | | Pass |
| The contractor clicks the “Create a new project” button and enters a non-valid name | | Fail |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Create a project template | |
| **Actors** | Contractor | |
| **Parameters** | Project, template name | |
| **Preconditions** | The contractor is logged in, there is at least 1 project | |
| **Actions** | The contractor chooses a project  The contractor selects “Save as a template” | |
| **Result** | A new project template is created | |
| **Data** | | **Result** |
| The contractor clicks the “Save as template” button and enters a valid name | | Pass |
| The contractor clicks the “Save as template” button and enters a non-valid name | | Fail |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Create a new project from a pre-defined template | |
| **Actors** | Contractor | |
| **Parameters** | Project name, old template | |
| **Preconditions** | The contractor is logged in, the new project’s name is valid | |
| **Actions** | The contractor clicks the “Create a new project from template”  The contractor selects requested template and enters a new name | |
| **Result** | A new project is created | |
| **Data** | | **Result** |
| The contractor selects “Create a new project from template” but none exists | | Fail |
| The contractor clicks the “Create a new project from template” button and chooses a template | | Pass |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Edit a project | |
| **Actors** | Contractor | |
| **Parameters** |  | |
| **Preconditions** | The contractor is logged in | |
| **Actions** | The contractor chooses one of its projects  The contractor clicks the “Edit the project” button  The contractor edits the wanted details | |
| **Result** | The project is newly edited | |
| **Data** | | **Result** |
| The contractor clicks the “Edit the project” button and edits the wanted details with valid datum | | Pass |
| The contractor clicks the “Edit the project” button and edits the wanted details with non-valid datum | | Fail |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add a new stage to a project | |
| **Actors** | Contractor | |
| **Parameters** | The contractor is logged in | |
| **Preconditions** |  | |
| **Actions** | The contractor clicks the “Add a new stage” button  The contractor writes the wanted details and adds the stage | |
| **Result** | A new stage is added to the project | |
| **Data** | | **Result** |
| The contractor clicks the “Add a new stage” button and writes the wanted details | | Pass |
| The contractor clicks the “Add a new stage” button and does not write the wanted details | | Fail |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Edit a stage in a project | |
| **Actors** | Contractor | |
| **Parameters** | The contractor is logged in | |
| **Preconditions** |  | |
| **Actions** | The contractor chooses a project  The contractor chooses stage  The contractor selects edit stage and edit with non-empty datum | |
| **Result** | The stage is newly edited | |
| **Data** | | **Result** |
| The contractor edits a stage with the wanted details | | Pass |
| The contractor edits a stage with empty data | | Fail |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Removes a stage from a project | |
| **Actors** | Contractor | |
| **Parameters** | Stage | |
| **Preconditions** | The contractor is logged in | |
| **Actions** | The contractor chooses a stage  The contractor clicks the “Remove the stage” button | |
| **Result** | The selected stage is removed from the project | |
| **Data** | | **Result** |
| The contractor clicks the “Remove the stage” button | | Pass |
| The stage doesn’t exist | | Fail |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add a new mission to a stage | |
| **Actors** | Contractor | |
| **Parameters** | Mission details, stage | |
| **Preconditions** | The contractor is logged in | |
| **Actions** | The contractor enters to a stage  The contractor clicks the “Add a new mission” button  The contractor writes the wanted details and adds the mission | |
| **Result** | A new mission is added to the selected stage | |
| **Data** | | **Result** |
| The contractor clicks the “Add a new mission” button and writes the wanted details | | Pass |
| The contractor clicks the “Add a new mission” button and does not write the wanted details | | Fail |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Edit a mission in a stage | |
| **Actors** | Contractor | |
| **Parameters** | The contractor is logged in | |
| **Preconditions** |  | |
| **Actions** | The contractor chooses a project  The contractor chooses stage  The contractor chooses mission  The contractor selects edit mission and edit with non-empty datum | |
| **Result** | The mission is newly edited | |
| **Data** | | **Result** |
| The contractor edits a mission with the wanted details | | Pass |
| The contractor edits a mission with empty data | | Fail |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Removing a mission from a stage | |
| **Actors** | Contractor | |
| **Parameters** | The contractor is logged in | |
| **Preconditions** |  | |
| **Actions** | The contractor enters to a stage  The contractor chooses a mission  The contractor clicks the “Remove the mission” button | |
| **Result** | The selected mission is removed from the stage | |
| **Data** | | **Result** |
| The contractor clicks the “Remove the mission” button | | Pass |
| The mission doesn’t exist | | Fail |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Updating Mission Status | |
| **Actors** | Contractor, Project Manager, Work Manager | |
| **Parameters** | Project | |
| **Preconditions** | The actor is logged in | |
| **Actions** | The actor chooses a project  The actor chooses stage  The actor chooses a mission  The actor selects "Update mission Status"  The actor selects new status | |
| **Result** | The mission status will change accordingly | |
| **Data** | | **Result** |
| The actor has access to the project | | Pass |
| The actor doesn’t have access to the project | | Fail |
| Work manager sets status from “invalid” to “Done” | | Fail |
| Project manager/ Contractor sets status from “invalid” to “Done” | | Pass |

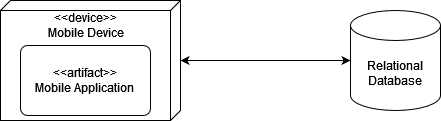
|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Viewing Project Information | |
| **Actors** | Contractor, Project Manager, Work Manager | |
| **Parameters** | Project | |
| **Preconditions** | The actor is logged in | |
| **Actions** | The actor selects a project | |
| **Result** | The actor receives all the project information | |
| **Data** | | **Result** |
| The actor has access to the project | | Pass |
| The actor does not have access to the project | | Fail |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Uploading proof to a mission | |
| **Actors** | Work Manager | |
| **Parameters** | proof file, project, mission | |
| **Preconditions** | The work manager is logged in and has access to the project | |
| **Actions** | The work manager selects a project  The work manager selects a stage  The work manager selects a mission  The work manager selects “upload proof”  The work manager selects and uploads the proof file from his device | |
| **Result** | The proof file is uploaded and saved for the specific mission | |
| **Data** | | **Result** |
| The work manager uploads a file with a decent size | | Pass |
| The work manager uploads a file with a very large size | | Fail |
| The work manager uploads an empty file | | Fail |

**System Architecture**

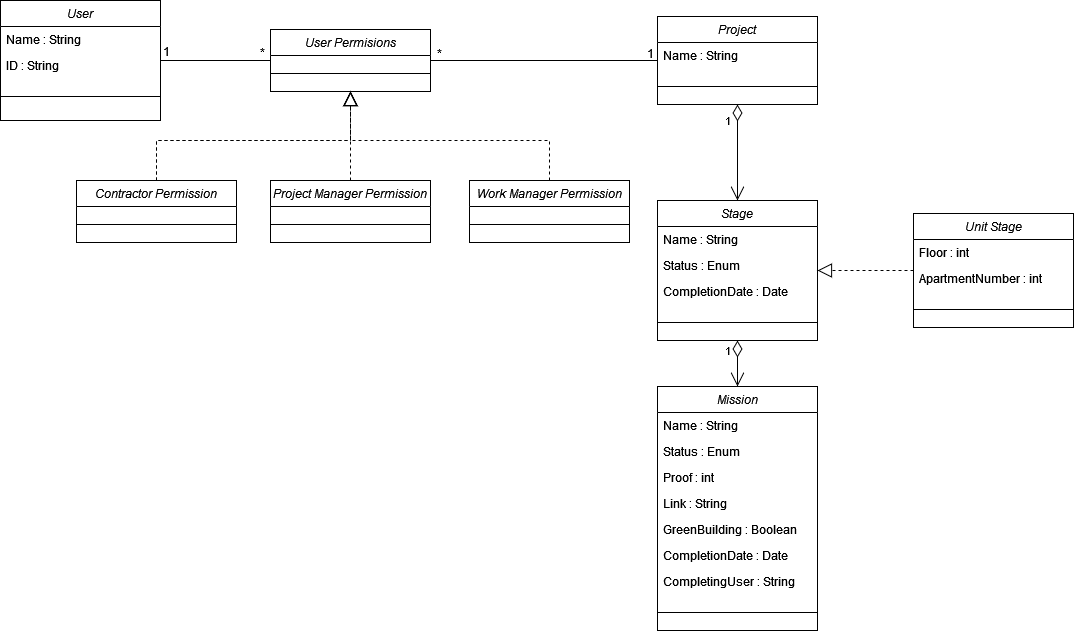
Mobile application: Written in Typescript and using React Native, the mobile application will run on the user’s phones and every query will require internet access. Data will not be saved locally.

Relational database: An SQL database located on a remote server, responsible for saving all the data object’s attributes.

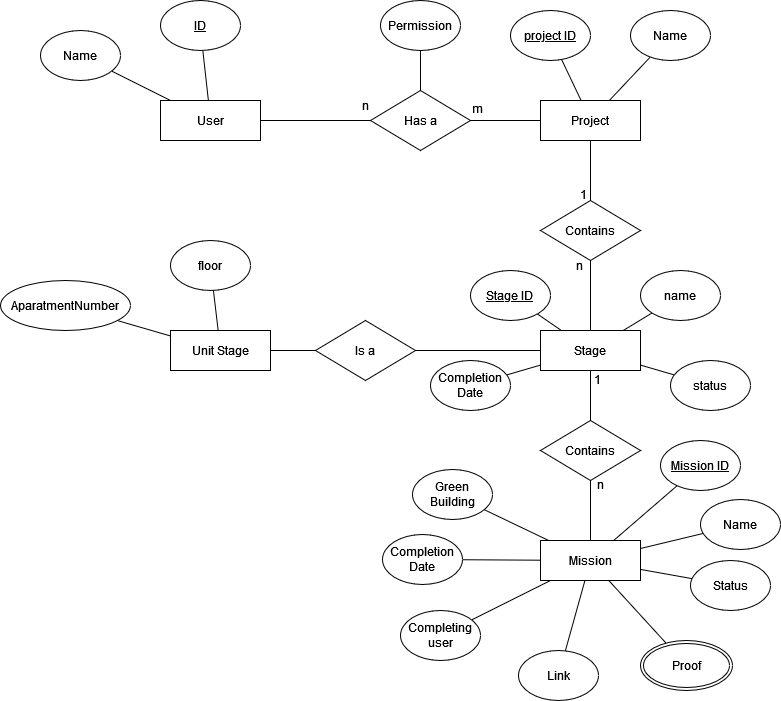


**Data Objects**

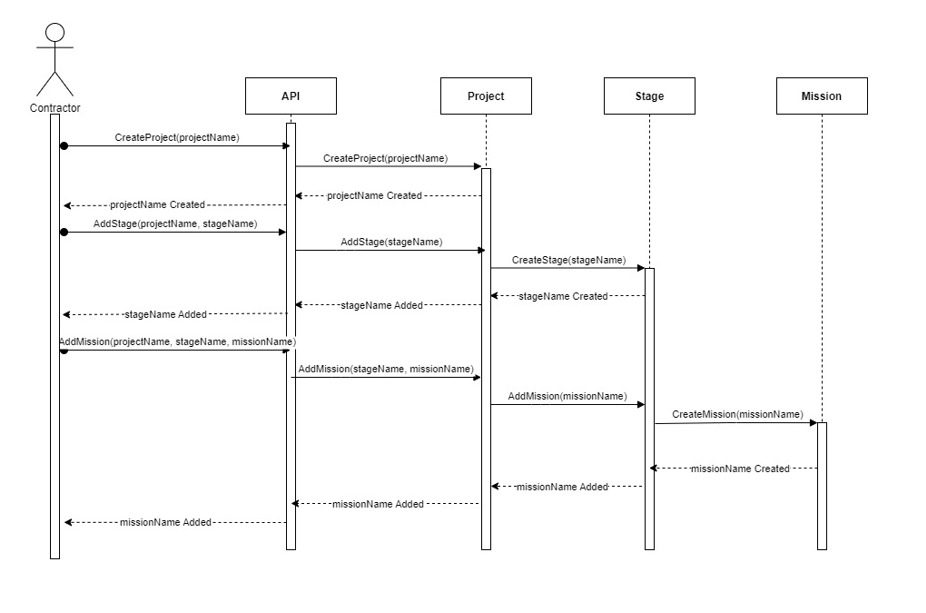
Class diagram representing our data domain:



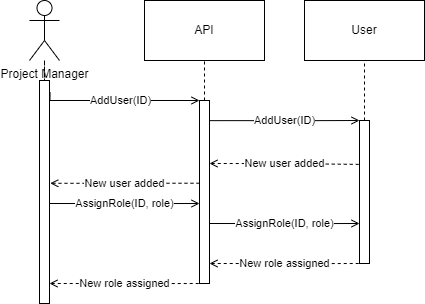
Entity relationship model:

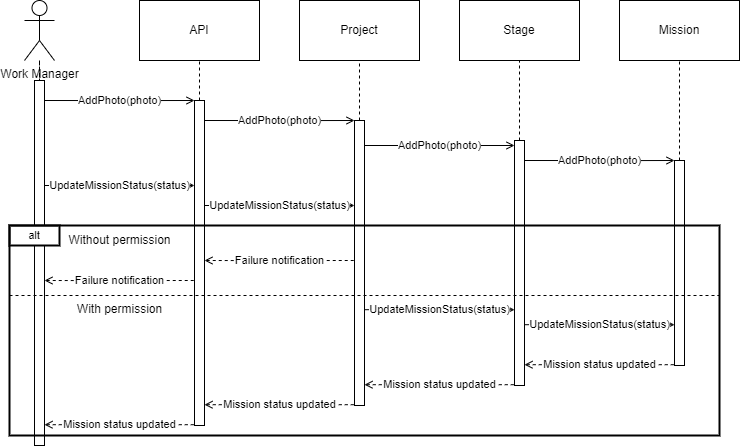


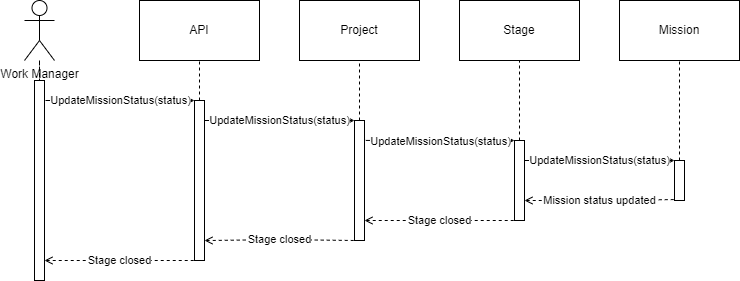
**Sequence Diagrams**

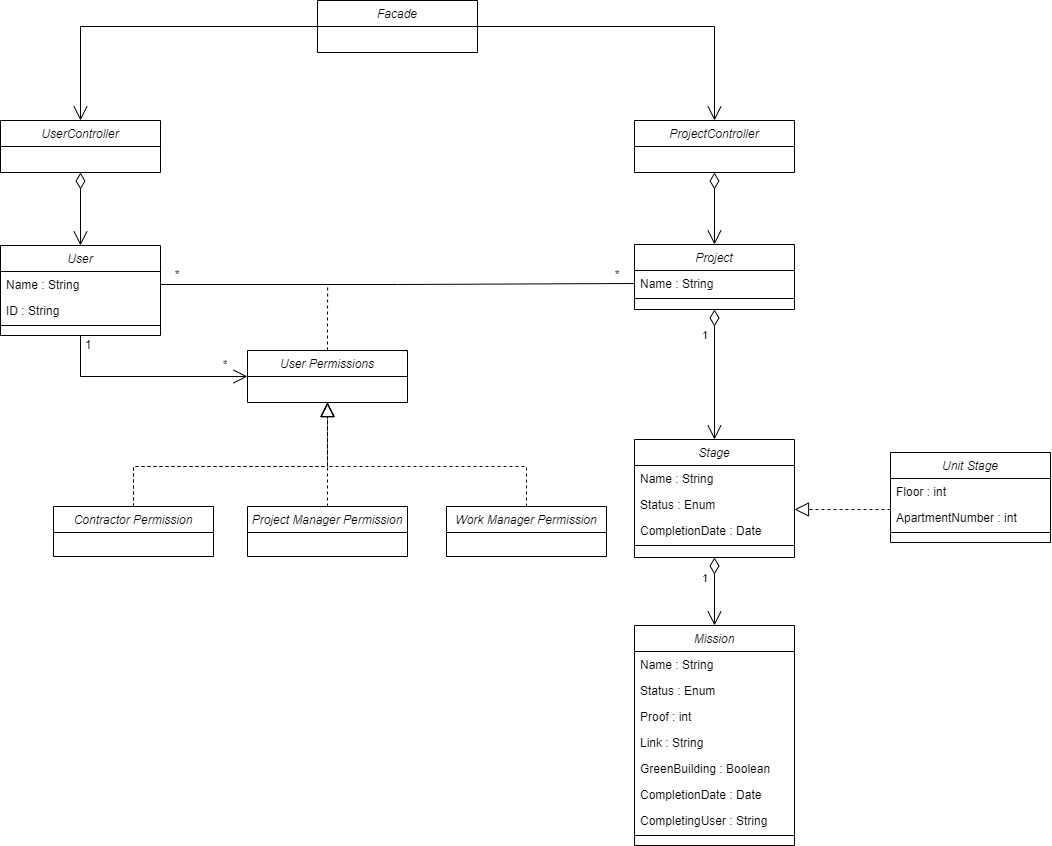
Creating a project:

Registering a user and assigning a role:



Uploading proof and updating mission status:

Closing last mission in stage:

**Class Diagrams**

**Class Description**

**ProjectController**

Attributes:

* List of the projects.

Methods:

* Add a new project to the list with non-empty data.
* Add a project to the list from a template.
* Save a project as a template.
* Edit a project in the list with non-empty data.
* Remove a project from the non-empty list (an admin).

**Project**

Attributes:

* Its name.
* List of its stages.

Methods:

* Add a stage to the list with non-empty data.
* Edit a stage in the list with non-empty data.
* Remove a stage from the non-empty list.

**Stage**

Attributes:

* Its name and status.
* When all its missions are done it automatically adds the stage completion date.
* List of its missions.

Methods:

* Add a mission to the list with non-empty data.
* Edit a mission in the list with non-empty data.
* Remove a mission from the non-empty list.

**Mission**

Attributes:

* Its name and status.
* Mark if it meets the green building standard.
* A proof that the task was done – a int.
* Links to the documentation (תיעוד) and standard (תקן).
* When done it automatically adds the completion date and the name of the user who completed it.

Methods:

* Update its status.
* Upload a proof with valid data.

**UserController**

Attributes:

* A list of Users.

Methods:

* Add a new user to the system.

**User**

Attributes:

* Its name and ID number.
* Its permissions for projects.

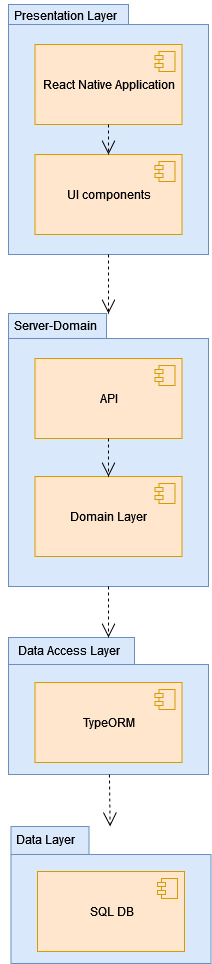
Methods:

* Login with a valid ID number.
* IsLoggedIn with a valid ID number.
* Add a new user to the system.
* Assign and unassign a role to a user.
* Add, edit, and remove a new project with non-empty data.
* Add, edit, and remove a new stage with non-empty data.
* Add, edit, and remove a new mission with non-empty data.
* Update a mission status.

**UserPermissions**

There are three types of permissions: contractor, project manager and work manager, and each permission correlates to a different project. Each instance is responsible for maintaining user's permission to some projects.

**Packages**



**User Interface Draft**

Our goal is to create an intuitive, easy to use, comfortable user interface.  
Note that all screens have a visible “Back” button and relevant screens have titles in the top, indicating the “path” the user has taken in the app. This will provide better “orientation” for the user while navigating the app.

**Graphical user interface, application

Description automatically generatedStartup Screen:**

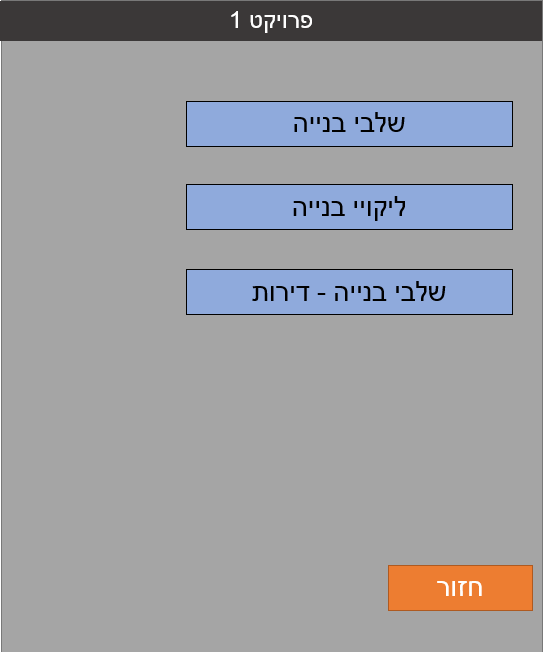
The user inputs Username and Password by pressing on the blank box which opens the keyboard on their mobile device. The user types the required data with the keyboard.  
The user then presses the Login button. If the login failed the error message will be displayed in red text. If the user logged in successfully the application displays the next screen.

**Chart

Description automatically generated with medium confidenceAll Projects Screen:**

The system displays only the relevant projects for the logged user, by showing buttons with the projects names. By clicking a project button, the system will display the project properties screen. “Managing projects and users” button will be visible to “Contractor” user only. This button will display the “Managing Projects and Users screen”.

**Project Properties Screen:**



The system displays the project properties which the user can choose from. Each property will send the user to a different screen. Clicking the “Building Stages” button will display “Build Stages Screen”, and “Building Faults” will display “Building Faults Screen”. Click the “Building Stages – Units” button will display the “Building Stages – Choose Floor Screen”.

**Building Stages Screen:**

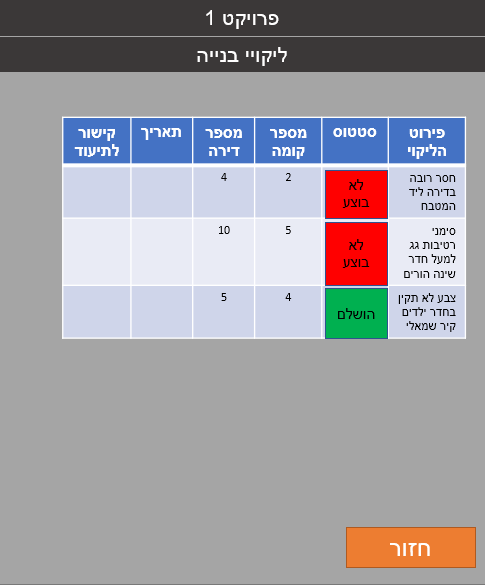
The system displays the list of stages in the chosen project. The project name can be seen at the top of the screen. Also, a Back button is displayed which returns us the “All Projects Screen”.  
The stages names are buttons, by clicking a stage button the system will display the next screen.

**Current Stage Screen:**

**Timeline

Description automatically generated**

The system displays the list of missions for the chosen stage in the chosen project. The system displays at the top of the screen the project name and the stage name under it. This will help the user to keep his “orientation” while navigating the system.  
Missions status can be changed by clicking of the status button for a specific mission, which opens a drop-down menu to select the new status from.

**Building Faults Screen:**

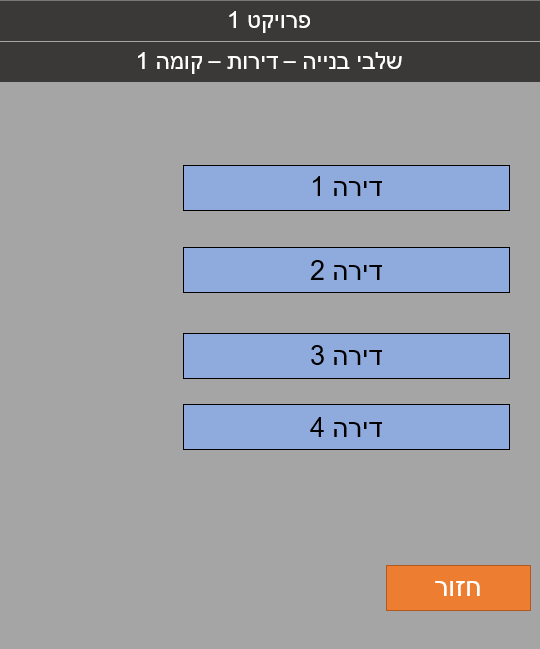
The system displays the building faults with all the relevant data as seen in the picture.

**Building Stages – Choose Floor Screen:**

****

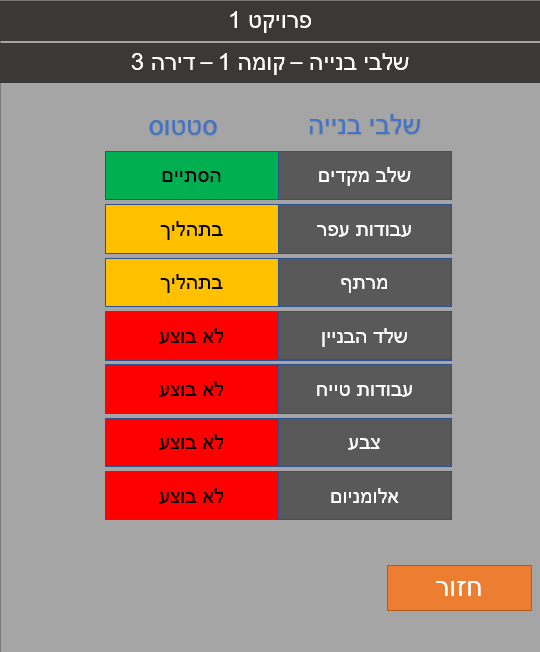
This screen allows the user to choose the floor. After choosing a floor the system will display the “Building Stages – Choose Unit Screen”.

**Building Stages – Choose Unit Screen:**



This screen allows the user to choose the apartment in the floor he already chose.  
Choosing an apartment will lead the user to the “Building Stages – Per Unit Screen”.

**Building Stages – Per Unit Screen:**

****

This screen displays the building stages for the chosen unit (apartment). Every stage is a button which will lead the user to the relevant stage screen.

**Create A New Project Screen:  
Graphical user interface, text, application

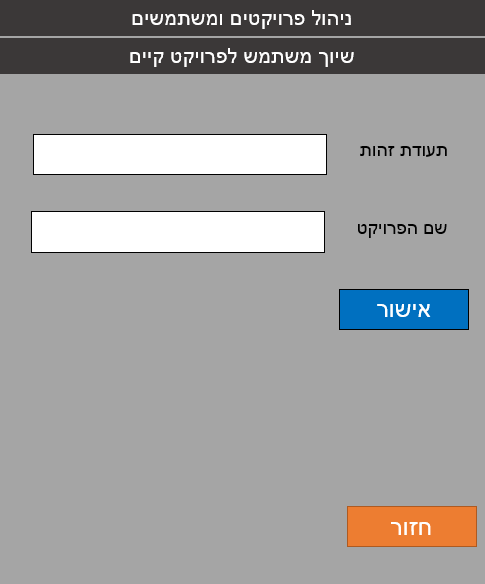
Description automatically generated**

The input is the project name. The user can choose to “confirm” the project creation or go back to the “Managing Projects and Users Screen”.

**Graphical user interface, application

Description automatically generatedRegister A New User Screen:**

The input is the ID and password for the new user, and project name for which the user will be assigned to (optional). The user can choose to “confirm” the registration or go back to the “Managing Projects and Users Screen”.

**Add User to Project Screen:  
**

The input is the user ID and the project name. The user can choose to “confirm” the process or go back to the “Managing Projects and Users Screen”.

**Application Navigation Tree**

**Startup Screen**

**All Projects Screen**

**Managing Projects and Users Screen**

**Register A New User Screen**

**Add User to Project Screen**

**Create A New Project Screen**

**Projects Properties Screen**

**Choose Floor Screen**

**Building Stages Screen**

**Building Faults Screen**

**Choose Unit Screen**

**Building Stages – Per Unit Screen**

**Current Stage Screen**

**Testing**

Our project will be developed using TDD method. That means tests for new features and requirements will be written before the code that implements them.  
Our tests will be composed of unit tests, integration tests and acceptance tests.  
We will be testing data reliability, system functionality, but we will not do any performance testing since the number of users and actions in the system is very narrow and high performance is not required.

Since usability is a top priority in our project, we will be giving a lot of weight to acceptance tests and customer tests, to receive continuous feedback and improve during development.  
Part of these tests are manual and will be done by the developers and the customer.

**Functional requirements testing:  
  
Unit Testing  
  
Login Test**Input:  
ID and Password  
  
Success:   
ID and Password matches an existing registered user  
  
Fail:   
ID or Password doesn’t match any registered user  
ID or Password are blank

**Update Mission Status Test**Input:  
New status  
  
Success:   
Mission status was changed to the new status  
  
Fail:   
Mission status wasn’t changed

**Update Stage Status Test**Input:  
New status  
  
Success:   
Stagestatus was changed to the new status  
  
Fail:   
Stagestatus wasn’t changed

**Get Stages Test**Input:  
Project name  
  
Success:   
All stages of the projects are returned with the correct status  
  
Fail:   
One or more stages are not returned  
Stages that don’t belong to the project are returned

**Get Missions Test**Input:  
Stage name  
  
Success:   
All missions of the stage are returned with the correct status  
  
Fail:   
One or more missions are not returned  
Missions that don’t belong to the stage are returned  
Missions status is incorrect

**Add Proof to Mission Test**Input:  
Proof link  
  
Success:   
Proof link is added to the mission  
  
Fail:   
Proof link is not added to the mission  
Link is corrupted

**Add Comment to Mission Test**Input:  
Comment text  
  
Success:   
Comment text is added to the mission  
  
Fail:   
Comment text is not added to the mission  
Comment is empty string

**Add Comment to Stage Test**Input:  
Comment text  
  
Success:   
Comment text is added to the stage  
  
Fail:   
Comment text is not added to the stage  
Comment is empty string

**Register User Test**Input:  
User ID and Password  
  
Success:   
User is registered to the system with ID and Password  
  
Fail:   
User is not registered to the system  
ID already exists  
Password doesn’t meet the minimum requirements   
ID or Password are blank

**Create A New Project Test**Input:  
Project Name  
  
Success:   
Project name is valid, and project is added to the system  
  
Fail:   
Project is not added to the system  
Project name is invalid

**Add Mission to Stage Test**Input:  
Mission name (optional: status, links, comment)  
  
Success:   
Mission added to the stage  
  
Fail:   
Mission wasn’t added to the stage  
Mission parameters are invalid

**Update Mission Properties Test**Input:  
Mission name (optional: status, links, comment)  
  
Success:   
Mission properties are updated  
  
Fail:   
Mission properties weren’t updated  
Mission parameters are invalid

**Add Stage to Project Test**Input:  
Stage name  
  
Success:   
Stage added to the project  
  
Fail:   
Stage wasn’t added to the project  
Stage name in invalid

**Remove Mission from Stage Test**Input:  
Mission name  
  
Success:   
Mission was removed from stage  
  
Fail:   
Mission wasn’t removed from stage  
Mission name doesn’t exist

**Remove Stage from Project Test**Input:  
Stage name  
  
Success:   
Stage was removed from project  
  
Fail:   
Stage wasn’t removed from stage  
Stage name doesn’t exist

**Mark Green Building Test**Input:  
Mission name  
  
Success:   
Mission is marked as “Green Building”  
  
Fail:   
Mission isn’t marked as “Green Building”  
Mission name doesn’t exist

**Export Data from Project Test**Input:  
Project name  
  
Success:   
Data returned contains all the correct project data  
  
Fail:   
Data returned missing some data of the project  
Data returned contains irrelevant data  
Data returned is corrupted  
Project name doesn’t exist

**Acceptance Testing**

**Startup Pre-defined Contractor Test**Input:  
pre-defined contractor ID and Password  
  
Successful path:   
User boots the system -> User inserts correct ID and Password of pre-defined the contractor -> User logged in  
  
Failure paths:   
User boots the system -> User inserts incorrect ID -> Error Message

User boots the system -> User inserts incorrect Password -> Error Message

**Work Manager – Update project/stage/mission status Test**Input:  
Project name, Stage name, Mission name, New status (User is a Work Manager or higher)  
  
Successful path:   
User is logged in -> User updates mission status to any status except “Done” -> The system updates the mission’s status accordingly

User is logged in -> User updates mission status to “Done” -> The system saves the system time and date, user ID of the user marked the missions/stage as done and files uploaded as proof-> The system updates the mission’s status accordingly

Failure paths:   
User isn’t logged in

Incorrect Project/Stage/Mission name -> Error message

Invalid status -> Error message

User updates mission in a project he isn’t assigned to -> Error message

**Work Manager – Get Project Test**Input:  
Project name (User is a Work Manager or higher)  
  
Successful path:   
User logged in -> User requests project data -> System returns project data with all the stages and missions under it  
  
Failure paths:   
Incorrect project name -> Error message

User not logged in -> Error message

User requests project he isn’t assigned to -> Error message

**Work Manager – Get Stage Test**Input:  
Project name, Stage name (User is a Work Manager or higher)  
  
Successful path:   
User logged in -> User requests stage data -> System returns stage data with all the missions under it  
  
Failure paths:

Incorrect Project/Stage name -> Error message

User not logged in -> Error message

User requests project he isn’t assigned to -> Error message

**Work Manager – Get Mission Test**Input:  
Project name, Stage name, Mission name (User is a Work Manager or higher)  
  
Successful path:   
User logged in -> User requests mission data -> System returns mission data  
  
Failure paths:   
Incorrect Project/Stage/Mission name -> Error message

User not logged in -> Error message

User requests project he isn’t assigned to -> Error message

**Work Manager – Add/Update comment/proof to mission Test**Input:  
Project name, Stage name, Mission name, comment/proof link (User is a Work Manager or higher)  
  
Successful path:   
User is logged in -> User adds/updates comment/proof in mission -> The system updates the mission data accordingly  
  
Failure paths:   
User is not logged in -> Error message

Project/Stage/Mission name is invalid -> Error message

User updated a mission under a project he isn’t assigned to -> Error message

**Project Manager – Register New User Test**Input:  
ID, Password (User is a Project Manager or higher)  
  
Successful path:   
User is logged in -> User registered a new user with ID and Password -> System creates a new user with ID and Password  
  
Failure paths:   
User is not logged in

ID already exists -> Error message

ID or Password parameters are invalid

User registers a new user to a project he isn’t assigned to -> Error message

**Project Manager – Assign/Unassign role Test**Input:  
User ID, New role (User is a Project Manager or higher)  
  
Successful path:   
User is logged in ->User assigns a new role to another user -> System updates the other user’s role/permission  
  
Failure paths:   
User is not logged in -> Error message

User assigns a role to another user that is not assigned to it’s project

User is already assigned with the new role -> Error message

User with user ID doesn’t exist -> Error message

New role is not a valid role in the project -> Error message

**Project Manager – Invalid mission to Done Test**Input:  
Project name, Stage name, Mission name (User is a Project Manager or higher)  
  
Successful path:   
User is logged in -> User changes an invalid mission to Done status -> System changes the mission’s status to Done  
  
Failure paths:   
User isn’t logged in -> Error Message

User isn’t a Project manager -> Error message

Project/Stage/Mission name is invalid -> Error message

**Contractor – Create A New Project Test**Input:  
Project name (User is Contractor)  
  
Successful path:   
User is logged in -> User creates a new project -> A new project is created and added to the system

User is logged in -> User creates a new project from a pre-defined template -> A new project is created and added to the system with the template stages and missions  
  
Failure paths:   
User isn’t logged in -> Error Message

User isn’t a Contractor -> Error message

Project name already exists -> Error message

**Contractor – Update project/stage/mission Test**Input:  
Project name, new updated parameters (User is a Contractor)  
  
Successful path:   
User is logged in -> Updates a parameter in project/stage/mission -> The system updates the project/stage/mission data correctly  
  
Failure paths:   
User is not logged in -> Error message

Project is not a project added by this contractor -> Error message

Invalid update parameters

**Contractor – Add/Remove a stage/mission from project Test**Input:  
Project name, Stage name, Mission name (User is a Contractor)  
  
Successful path:   
User is logged in -> User adds/removes a stage/mission from it’s project -> The system adds/removes the stage/mission from the project  
  
Failure paths:   
User is not logged in -> Error message

Removing a Stage/Mission name that doesn’t exist -> Error message

Project is not a project added by this contractor -> Error message

Adding a Stage/Mission that already exists -> Error message

**Last mission done in stage Test**Input:  
N/A  
  
Successful path:   
A stage with all missions status set to “Done” except for 1 mission -> Last mission’s status is updated to Done -> The system updates the last mission’s status and the whole stage status to “Done”  
  
Failure paths:   
A stage with all missions status set to “Done” except for 1 mission -> Last mission’s status is updated to Done -> The system doesn’t update the last mission’s status or the whole stage status to “Done”

**All missions “To Do” in stage Test**Input:  
N/A  
  
Successful path:   
A stage has all it’s missions with “To Do” status -> One mission’s status is set to “In Progress” or “Done” -> The system updates the mission’s status accordingly and the whole stage status to “In Progress”  
  
Failure paths:   
A stage has all it’s missions with “To Do” status -> One mission’s status is set to “In Progress” or “Done” -> The system doesn’t update the mission’s status accordingly or the whole stage status to “In Progress”

**Get “Green Building” Test**Input:  
Project name  
  
Successful paths:   
User is logged in -> User requests all “Green Building” missions in a project -> The system returns all the missions within this project marked as “Green Building”

User is logged in -> User requests all “Green Building” missions in a project with no “Green Building” missions -> The system returns an empty missions list

Failure paths:   
User is not logged in -> Error message

**Regression Testing**All Unit tests and Acceptance tests should pass before merging a branch into master. Also, manual tests should be performed with a Pass/Fail criteria. This will prevent most bugs on the master branch after implementing new features.

**Non-Functional requirements testing:**

**Req 1** -The system should be able to support 50 users with up to 1 second reaction speed

We will use tools for load and performance testing to simulate 50 users sending actions in our application and measure the reaction time. There are a few tools that provide those capabilities and produce reports such as: Gatling, Apache Jmeter, React Native Performance Monitor.

**Req 2** - The system should be compatible with Android and iOS

The app will be installed on smartphones from both operating systems. A sanity test, with a defined flow and some basic scenarios will be run on every OS. Those tests will be manual tests.

**Req 3** - The system should support text in Hebrew (see “Dictionary” below)

This is a manual test. The tester will go through every screen and button in the application and compare it with the “Dictionary” found in the ARD document.

**Req 4** - The system should support saving at least 500 previous projects

This will be part of the acceptance tests. An automated test that will create 500+ projects and verify their existence in the history.