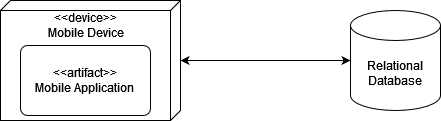
**System Architecture**

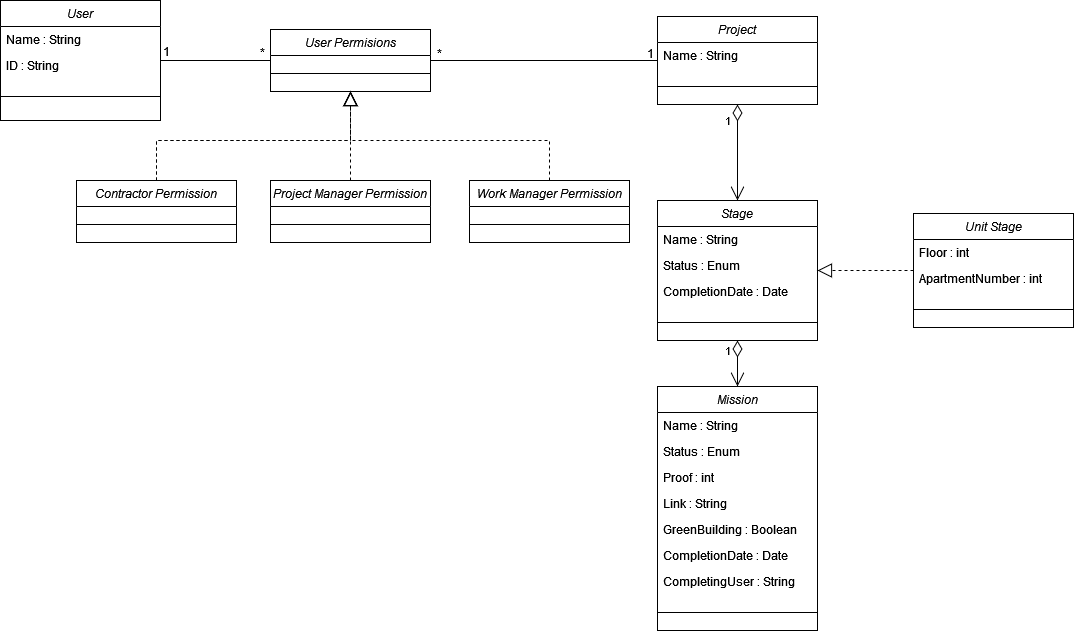
Mobile application: Written in Typescript and using React Native, the mobile application will run on the user’s phones and every query will require internet access. Data will not be saved locally.

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

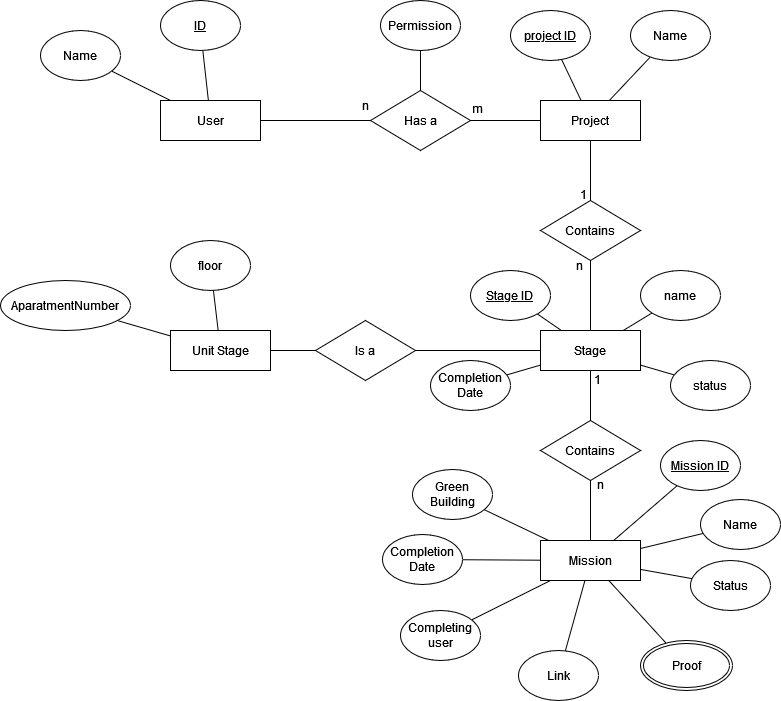


**Data Objects**

Class diagram representing our data domain:



Entity relationship model:



**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:**

These tests are already written under every Use Case in the “ARD” document.  
To prevent duplication, the tests will not be written again, but you can find them in the ARD document under “Use Cases”

**Non-Functional requirements testing:**

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.

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.

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.

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.