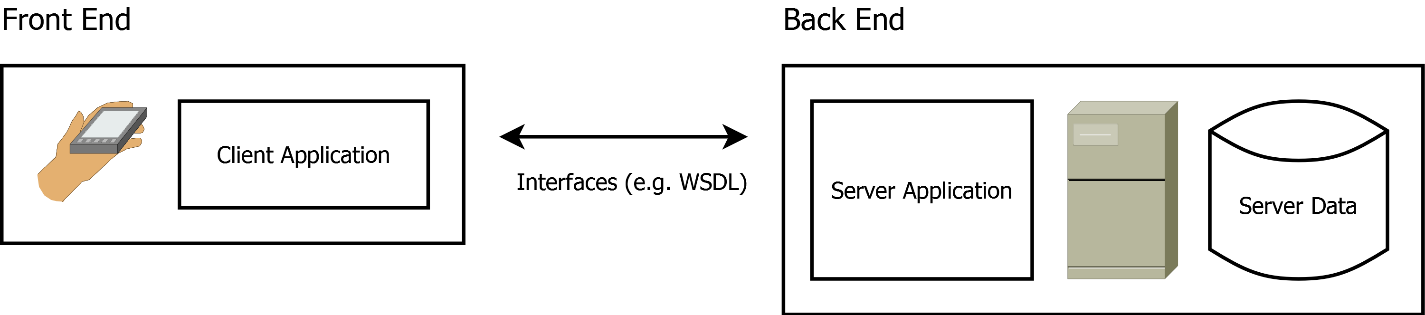
# Requirements Document MHC-PMS (Team Red)

## System Model



### Client Application

The client-side application runs on the user’s handheld (mobile phone or table). It contains the definition of the graphical user interface and cached data from the server.

### Interfaces

The communication between the client-side application and the backend is provided by dedicated web services (e.g. SOAP, Rest). The communication is secured by a (private-/public) key based authentication system.

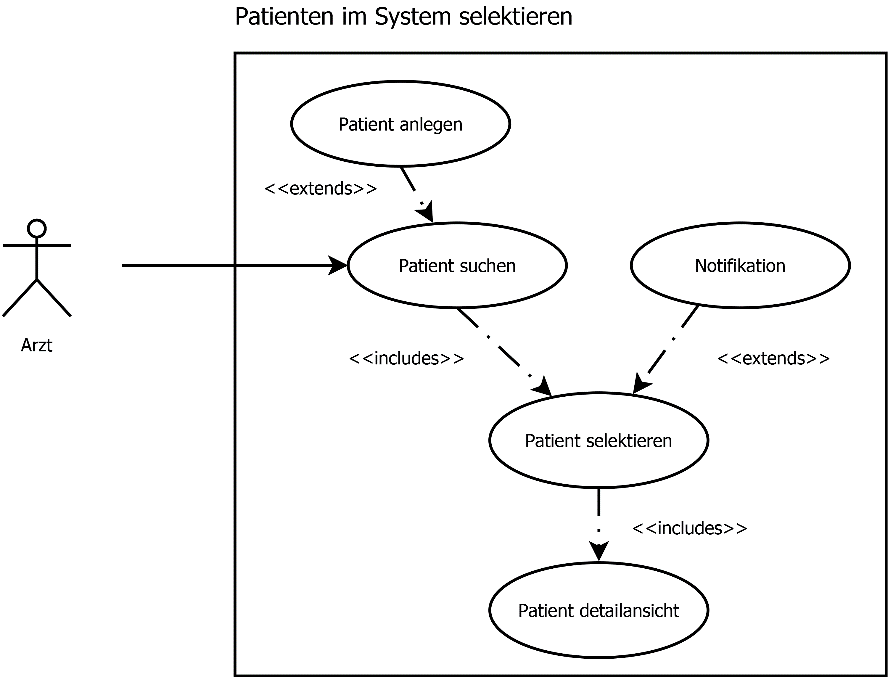
### Server Application

The Server-side application provides all data access methods to access the application’s data pool. The methods can be accessed by interfaces (mentioned in the previous section).

## User Requirements

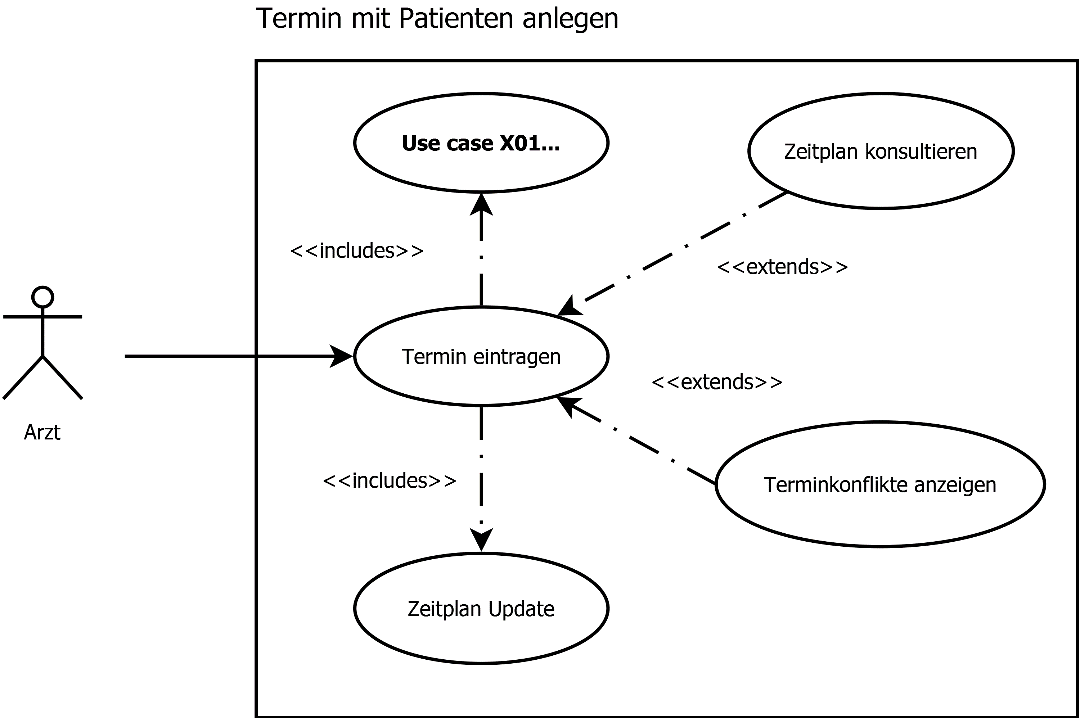
### **Services provided for the User:**

#### Use case X01: Search (and Register) Patient:



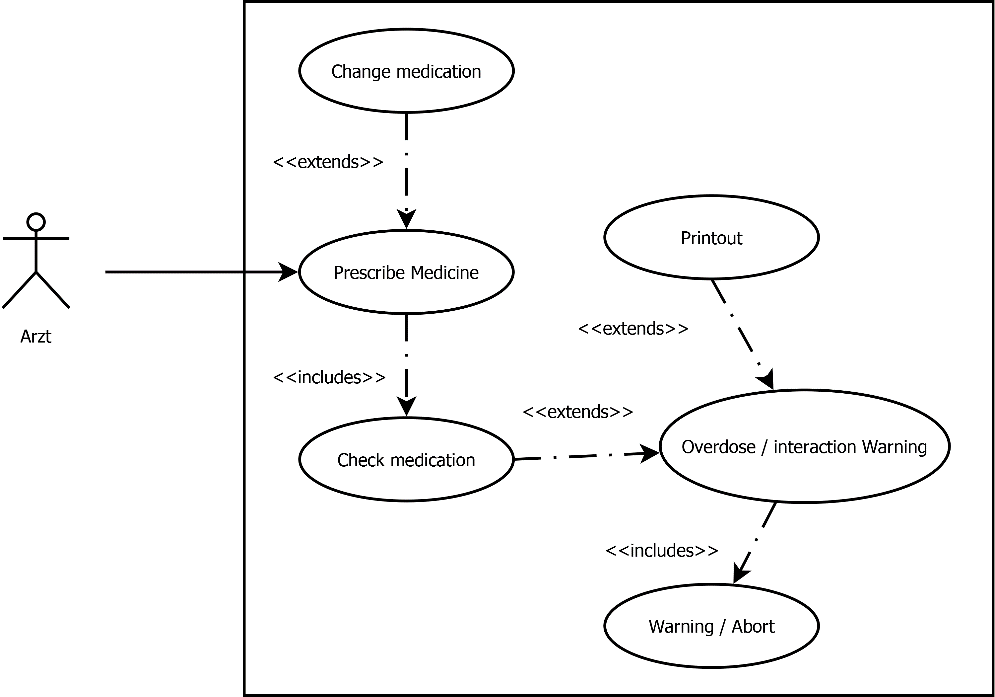
* The User can search for registered Patients on a Form
* The User can overview all necessary details of a patient as well as previously created appointment logs on a details page.
* The User can register a new patient in the system giving all necessary details.
* The user can send an automatic notification to the patient (email or SMS).

#### Use case C06: Create Appointment



* The user can search for a specific appointment by known details.
* The user can overview all his appointments on a timetable, by a daily, weekly and monthly view.
* The timetable acts as a scheduling mechanism and can be shared between users.
* The user can open a details page for each entry in the timetable where additional information (e.g. Comments) are listed.
* The user can create a new appointment and enter it to the timetable. A mechanism shows if there are conflicting appointments. Any new appointment is registered in the timetable.

#### Use case M01: Prescription



* The user can lookup current medication of a patient.
* The user can change the current medication of a patient. He can add medication and define the dosage with a form.
* A validation mechanism prevents the user from prescribing toxic amounts of a certain medication and checks for dangerous conflicting medication. If the medication is dangerous, a warning is shown and the user cannot proceed.
* The user can print any entry or a summary of the patient’s current medication, the printout may as well be used for signed medical records.

### **Nonfunctional System requirements:**

* **The Application runs on portable devices such as smart phones or tablets.**
* **The user interface must be quick and responsive.**
* **A dedicated server must be used provide a centralized data pool. The server and the data access mechanisms must be scalable.**

## **System Evolution**

* The software must be able to run on the newest common portable devices on the market.
* The protected data server must be periodically migrated to the newest technologies to provide data security.
* New minor versions are released frequently to treat newly detected bugs in order of their priority (e.g. voting system). Minor versions can also be used to implement new features on a modular basis.
* A new Major version is released every year. Technology changes (library updates, migrations) can only be performed on new major versions. Features with impact on the system architecture can only be implemented for an new major version.

## **Testing**

### Test Cycles

To ensure that the software’s functionality works initially expected, a continuous testing system is provided:

* Unit Testing: On each software build, the automatic test runs are started (Junit, MS Test) to test the software’s basic functionality. If any of the test run fails, the build is not successful and will not be deployed on the testing environment.
* System Integration Tests: On each release cycle, the software is deployed on a dedicated (integration) environment where the interfaces and the backend-frontend communication is tested with mock data.
* User Acceptance Tests: On each major release, all features and functionalities are tested by a dedicated testing team.

### Test Environments

The development und testing life cycle is applied on the following set of environments:

* Development Environment: The nightly builds and all developer builds are deployed on this environment, it’s used to run automated unit tests and developer tests.
* Test environment: The weekly builds are deployed on this environment, it is used to create automated and manual system integration tests (backend-frontend communication, data processing).
* Integration Environment: A new version is deployed on this environment before any new Release. It’s used to perform the user acceptance tests (End to end testing) by a dedicated testing team.
* Production Environment: The Released version is deployed on this environment. The functionality can be overviewed and controlled by logging and an automated bug report system if the user allows it.