Requirements Document Team Red

# Preface

# Introduction

* Need for the System

Our system allows the management of mental patients. The System is aimed at doctors caring for patients suffering from a mental disorder, in particular Borderline Syndrome. The problem, that doctors have, with borderline patients is, that coordination with these patients is difficult, as the may be homeless and frequently miss appointments. That makes it hard for the doctor to track and even contact the patient. Some people suffering from Borderline also often frequent the emergency room, as they are looking for attention.

## Systems function

The system aims to aid the doctor in treating patients with borderline. The functions which it will provide are:

* Documentation of therapy notes.
* Prescription and Medication management
* Therapy scheduling

## How it works with other systems

No specific interactions between other systems are planned at this point. But a system requirement is that, the system can be extended with interfaces to other Systems where needed.

## How it fits in to the buyers’ business strategy

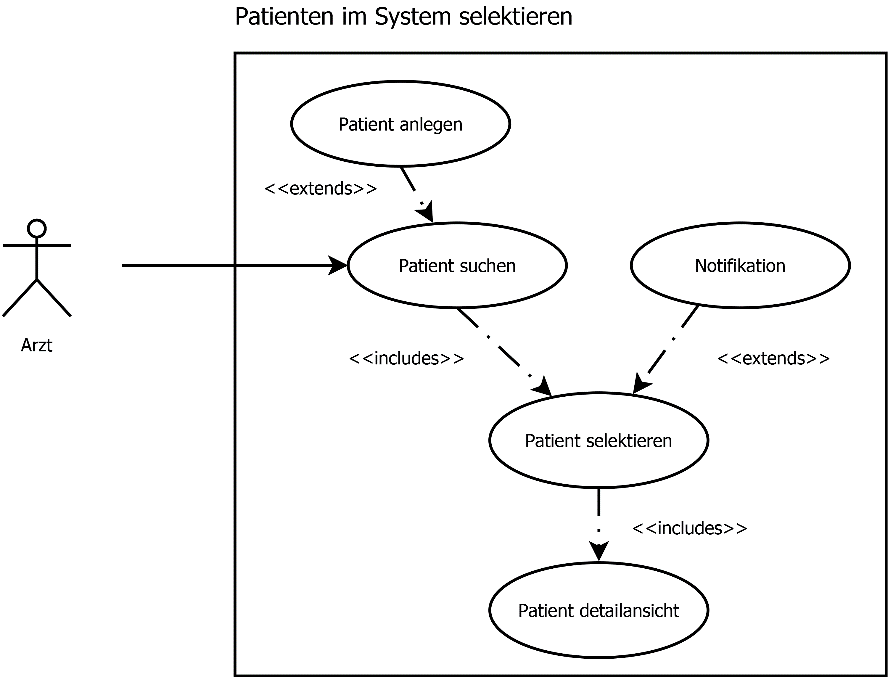
Task and workflows of the customer are streamlined and all tasks are brought into one app. An example of that is that medication and appointment management can all been done from within the app.

# Glossary

# User requirements definition (Ali)

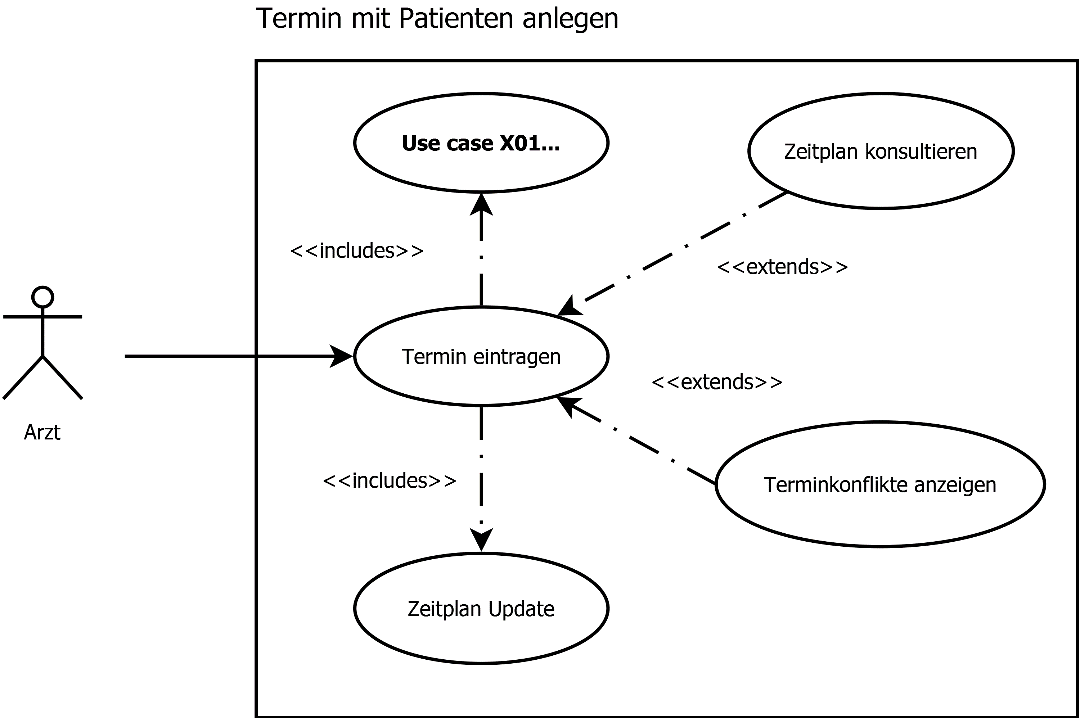
### **Services provided for the User:**

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



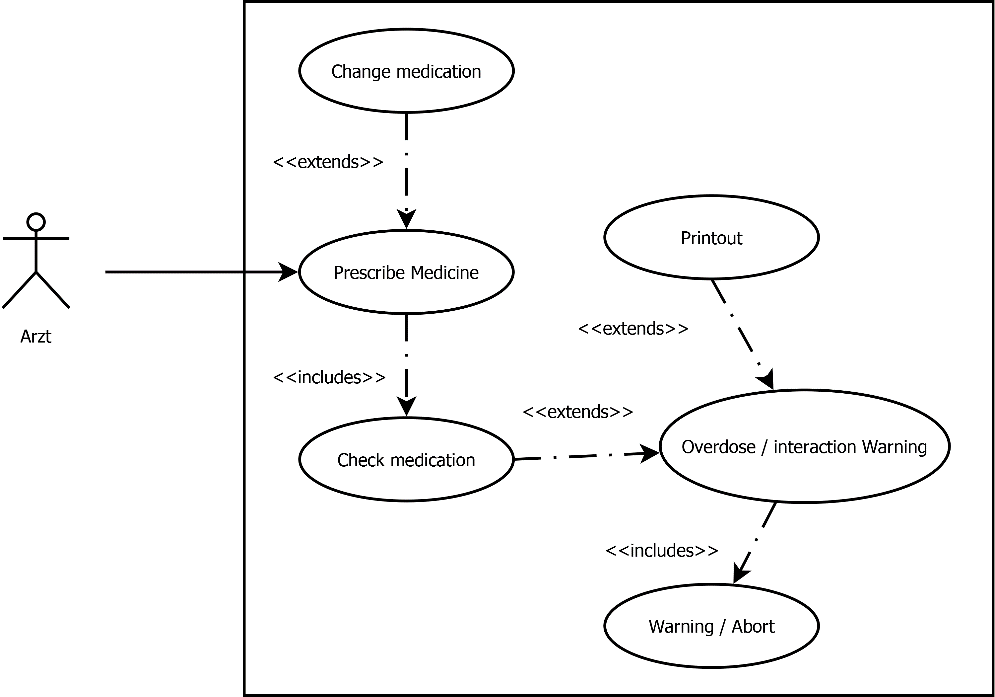
* The User can search for registered Patients on a Form
* The User can overview all necessary details of a patient as well as previously created appointment logs on a details page.
* The User can register a new patient in the system giving all necessary details.
* The user can send an automatic notification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

# System architecture (Flo)

Das System wird in einer mehrstufigen Architektur implementiert. Die einzelnen Services werden dabei möglichst atomar auf einzelne Server/Container aufgeteilt. Die Services werden in zwei unterschiedlichen Sicherheitszonen stehen. Der Frontend Web- und DNS Server werden in der DMZ Betrieben. Die Datenbank und Applikationsserver stehen in einer separaten Serverzone. Services in diesen Zonen können keine Verbindungen gegen aussen aufbauen. Nur explizit erlaubte Verbindungen dürfen aufgebaut werden. Für Wartungsarbeiten steht eine Out-of-band Managementzone zur Verfügung.



Systeminstanzen

Das Gesamtsystem wir dabei parallel in drei Instanzen betrieben. Dies stellt die Sevice Qualität des Kunden sicher und unterstützt die Entwicklung neuer Funktionen und das ausführen der automatisierten System Tests.

# System requirements specification (Adrian)

System requirements

non-functional

Usability: Most users don't have much of a technical background but should still be able to use all functionality of our software. The software reuses common patterns for user interactions from other software our customers use often.

Offline availability: Doctors must always be able to give medication and to do so they have to be able to check what medication a patient needs. Therefore they must be able to access patient data even when they can't connect to the server.

Data persistence: data must always be intact.

archive/history: older versions of the data must be available to eg. undo unauthorised changes

backup: in case of any kind of dammage of the server/data center there needs to be an external backup

locking: when a user is editing data the case must be locked to prevent conflicts (several users editing the same document)

data protection: patient data is sensitive and needs strict protection

access control: users permissions need to be checked before they can access data

remote access: the system must be accessible remotely (eg. when doctors are doing housecalls)

Logging: access and changes must be logged

functional

user management: to fulfill the non-functional requirement data protection a users and userrights must be managed

patient management: patients and their medication must be managed

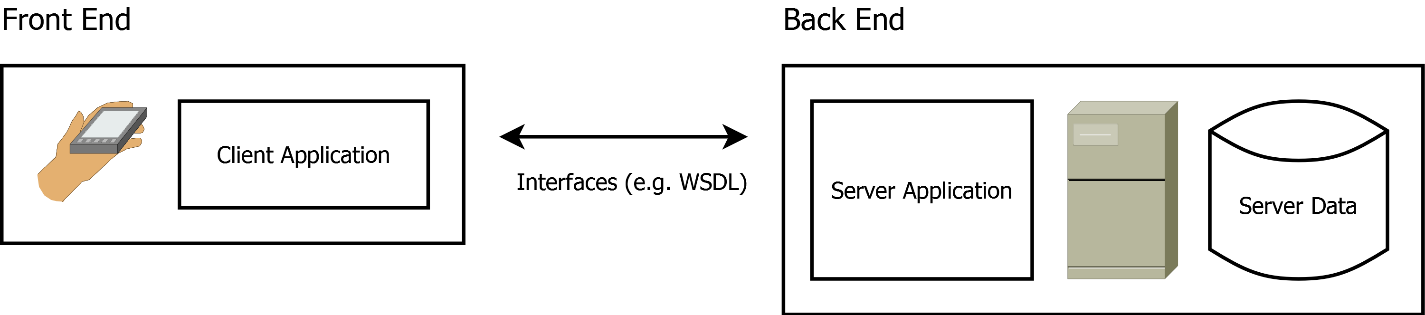
calendar: appointments have to be magageable

therapy notes: The doctor has to be able to store and read notes from therapy sessions

search: users should have the possibility to search data of any kind (patients, therapy notes, calendar entrys)

printing: patient data, calendar entrys and therapy notes must be printable

# System models (Ali)



### 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).

# System evolution (Flo Ali)

* 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.

In der ersten Implementation werden die requirements der Usergruppe “Ärzte” und Patienten mit Diagnosen BPD umgesetzt. Um zukünftige requierments möglichst einfach ins System zu integrieren, sollte die Applikation ein Modularer Aufbau mit klar definierte Schnittstellen beinhalten. Sind Änderungen an den Schnittstellen nötig, soll dies nur bei einem Major Release möglich sein und nur sollte nur als letzte Massnahme eingesetzt werden.

Der generelle LifeCycle der Applikation richtet sich nach den Wartungslaufzeiten der verwendeten Technologien .z.B. Frameworks oder Java.

Nach der ersten Implementation sollte die Systemarchitektur und Datenschemen so weitsichtig entwickelt sein, damit man weitere Nutzergruppen und Diagnosen implementieren kann.

Infolge des agilen Entwicklungsprozesses, wird der Fokus der Erst- und Weiterentwicklung auf den User Requierements sein. Somit soll sich die Applikation den ändernden Anforderungen evolutionär anpassen.

Durch die Domain Requirements gegebenen Anforderungen an die Archivierung der Daten, muss garantiert werden, das generiete Daten auch in 10 Jahren zugänglich gemacht werden können. Gibt es durch die Weiterentwicklung Inkompatibilitäten des Datenbankschemas, muss ein verlustfreier Migrationsprozess der Daten garantiert werden.

# 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.

Die Applikation wird während der Entwicklung sowie im Unterhalt kontinuierlich geprüft. Die Tests werden dabei in die folgenden vier Kategorien unterteilt.

Komponenten Tests

Für jede Klasse muss eine korrespondierende Testklasse existieren, welche sicherstellt, dass die Funktionalität der Komponente korrekt ausgeführt wurde.

Mindestens nach jedem Sprint soll ein neuer Build erstellt werden. Die Tests werden bei jedem erstellen automatisch ausgeführt.

Integrationstests

Test Szenarios für die Integrationstests werden anhand des Testkonzepts abgehandelt. Diese Tests stellen den Betrieb in der Produktiven Umgebung sicher. Das Testkonzept muss dabei neue, sowie bestehende Funktionen abdecken.

Abnahme Tests

Tests Szenarios werden anhand des Testkonzept abgehandelt und stellen den Vertraglich vereinbarten Funktion Umfang, Stabilität und Performance des Systems sicher. Die Resultate dienen als Basis für die Abnahme durch den Kunden.

Monitoring

Die Verfügbarkeit und Performance der gesamt Applikation werden end-to-end durch „Roboter“ automatisierte überwacht. Alle Log Event und Performance Werte werden an ein zentrales System gesendet.

# Appendices

# Index