RedWare AG

Requirements Document MHC-PMS

Software Engineering FS2017 – Task 4

Team Red
3-24-2017

TABLE OF CONTENTS

PREFACE

This document is intended for the management, the sales department and for people who will be providing the technical infrastructure for the customer. The reader of this document should have some IT specific knowledge. This document is not intended for end users.

VERSION INFORMATION

Version	Created	Comment	Edited by
1.0	24.03.2017	Creation of initial version	Team Red

INTRODUCTION

Our system allows the management of patients with mental disorders. The System is aimed at doctors caring for patients suffering from a mental disorder, 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.

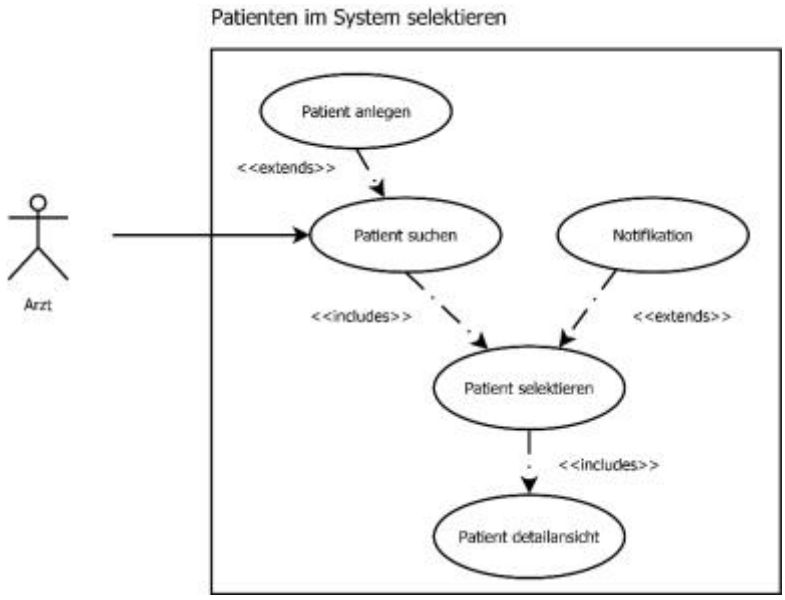
HOW IT FITS IN TO THE BUYERS' BUSINESS STRATEGY

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.

USER REQUIREMENTS DEFINITION

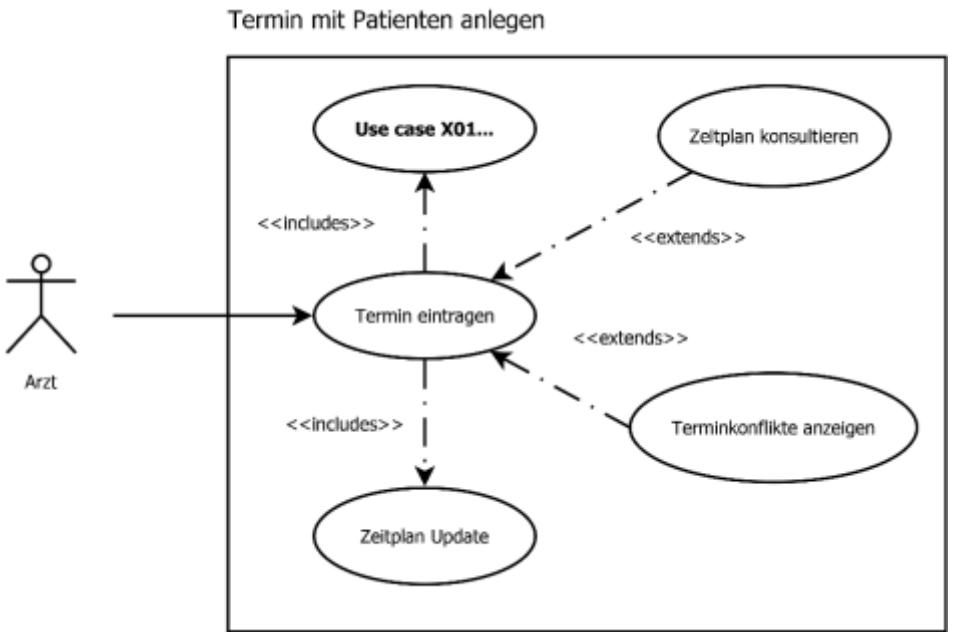
SERVICES PROVIDED FOR THE USER:

USE CASE X01: SEARCH (AND REGISTER) PATIENT:



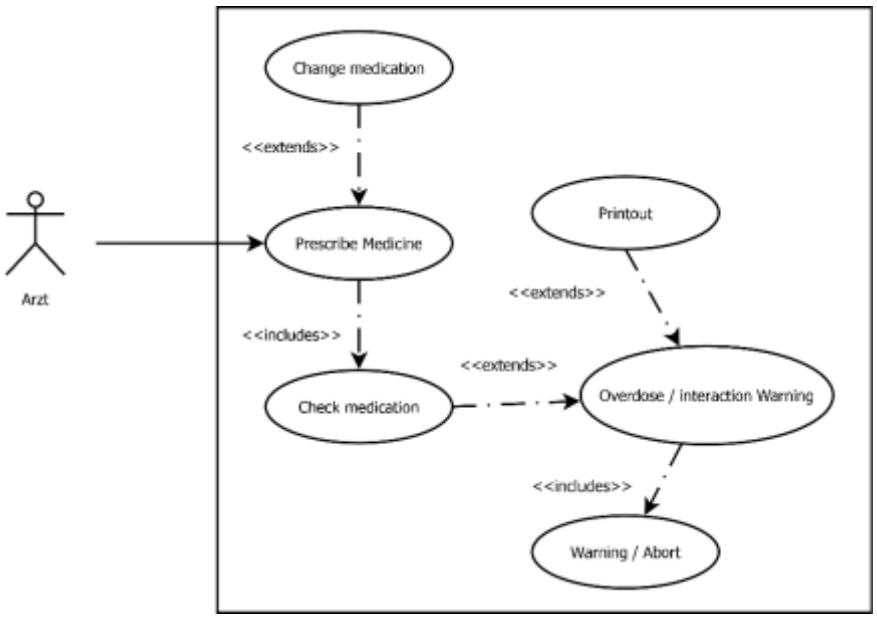
- The User can search for registered Patients on a Form
- The User can overview all necessary details of a patient as well as previously created appointment logs on a details page.
- The User can register a new patient in the system giving all necessary details.
- The user can send an automatic notification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

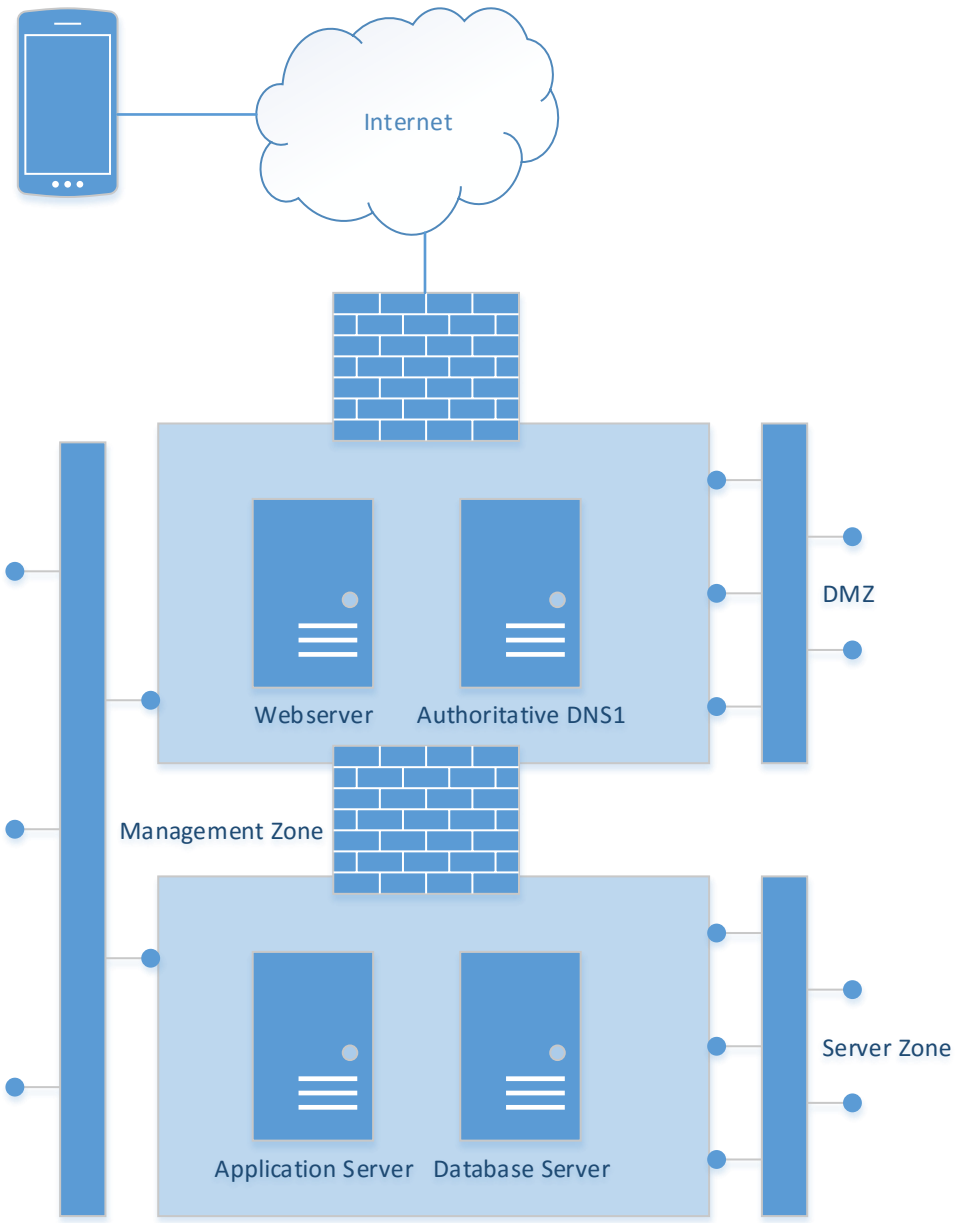
USE CASE M01: PRESCRIPTION



- 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

The system will be implemented in a multilayer architecture. The individual services will be split among individual server/container, on their most atomic level. The servers will be placed in two different security zones. The frontend Web and DNS server will be run in the DMZ zone. The database and application server will be run in a separate security zone. Services in to these cannot establish a connection to the outside. Only explicitly allowed connections are allowed to be established. For maintenance, an Out-of-band Management zone will be provided.



SYSTEM REQUIREMENTS SPECIFICATION

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 must 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 unauthorized changes.

BACKUP: in case of any kind of damage 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 house calls).

LOGGING: access and changes must be logged.

FUNCTIONAL

USER MANAGEMENT: to fulfill the non-functional requirement data protection a user's and user rights must be managed.

PATIENT MANAGEMENT: patients and their medication must be managed.

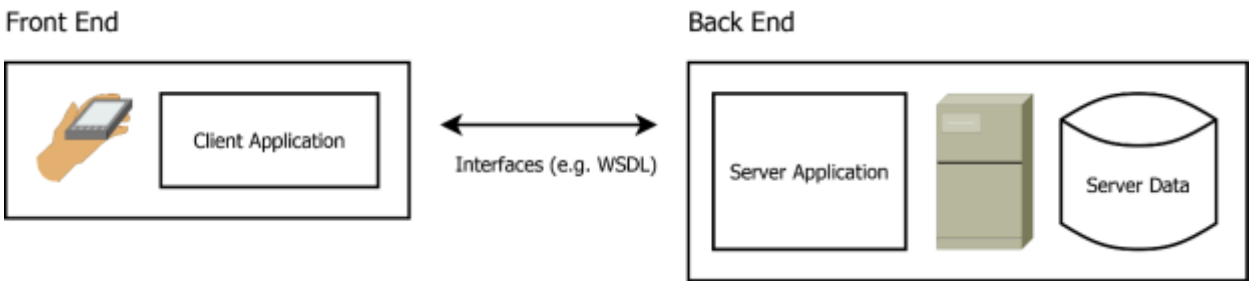
CALENDAR: appointments must be manageable.

THERAPY NOTES: The doctor must 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 entrees).

PRINTING: patient data, calendar entrees and therapy notes must be printable

SYSTEM MODELS



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

- 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 a new major version.
- Data backup and recovery must be possible, due to the domain requirements of the medical field, which state that records must be stored for at least 10 years.
- Loss less migrations from older version to newer versions must also be possible. Or there must be data migration plans to migrate data from older to newer versions of the system.

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 and 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.

GLOSSARY

SOAP Simple Object Access Protocol is a protocol to exchange data between different systems over a network

REST Representational State Transfer is a principle often applied on APIs which means that the same address always refers to the same dataset

API an Application Programming Interface is used to offer some of a programs functionality to other programs/services

FRONTEND what the end user sees and what he interacts with

BACKEND what's going on behind the scenes (frontend), not visible for the user

WSDL Web Services Description Language describes a SOAP interface

DMZ (demilitarized zone) is a subnetwork used for the servers that link requests from the outside to the local servers without exposing them to the public

APPENDICES

USE CASE SCENARIO

Nr. and Name:	C06 Create Appointment
Scenario:	Doctor creates an Appointment for a session with a patient.
Short Description:	A new appointment is needed
Actors:	Dr. Health (Doctor), Application
Starting Event and Preconditions:	Patient has contacted doctor and desires to create an appointment
Result and Postconditions:	A new appointment is created and added to the doctor's timetable

Steps:

Nr.	Actor	Description
1	Dr. Health	Dr. Health consults his timetable for an available time slot
1.1	Dr. Health	Dr. Health enters time and duration for new appointment
2	Dr. Health	Dr. Health looks up patient in the system
2.1	Dr. Health	If the patient was not found, the doctor registers the patient
2.2	Dr. Health	If more details about the patient are required, Dr. Health opens patient's details page
2.3	Application	Application notifies Dr. Health if there are any appointment conflicts

Exceptions, Variants:

Nr.	Actor	Step
1	Application	Timetable is not available (cannot reach server)
1.1	Dr. Health	Wrong input format
2	Dr. Health	Patients list not available (cannot reach server)
2.1	Dr. Health	Interface error on register patient
2.1	Application	Patient already exists (all keys match)
2.3	Application	Timetable is not available (cannot reach server)

Nr. and Name:	M 01 Prescription
Scenario:	Medication Prescription
Short Description:	Doctor prescribes medication for the patient. In the prescription process the dosages and drug interactions are validated
Actors:	Dr. TopHat, MHCPMS App, DB, Printer
Starting Event and Preconditions:	Patient needs medication to treat illness. May already be taking some medication. Medication may need to be changed.
Result and Postconditions:	Prescription is validated and submitted to DB, print prescription.

Steps:

Nr.	Actor	Description
1	Dr.	Doctor wants to change the patients' medication.
2	Dr. / App	Doctor chooses medication in App and sets dosage.
2.1	Dr. / App	Running medication is altered.
3	App	Dosage and drug interaction is validated.
4	App	App commits altered / new patients' medication to DB
4.1	App	App provides option to print prescription.
4.2	Dr. / App	Doctor prints prescription, app sends prescription to printer.

Exceptions, Variants:

Nr.	Actor	Step
2.0	App	<<Error>> Medicament not found
2.1.1	App	<<Error>> Patient has no current prescriptions.
3.0	App	<<Warning>> Dosage to high
3.1	App	<<Warning>> Conflict, drug interaction between Med.a and Med.b