PMS Requirements Documentation

Team Black

Contents

Preface

The following document is intended for all persons who are dealing with the PMS application. In our case, this means that the members of the Team Black and the customer side, which is represented by our instructor. This document should be interpreted without technical background.

Document History

Date	Version	Person	Changes
22.03.2017	V0.1	Lukas, Remo, Yannick, Henrik, Patrick, Gabriel	Creation of the Document
22.03.2017	V0.2	Yannick	Added Preface, Introduction
22.03.2017	V0.3	Remo, Patrick	Added User Requirements
22.03.2017	V0.4	Lukas	Added System Architecture
22.03.2017	V0.5	Gabriel	Added User Requirements
22.03.2017	V0.6	Patrick	Added System Evolution Testing
22.03.2017	V0.7	Henrik	Added System Requirements
22.03.2017	V1.0	Remo	Finalize the document

Introduction

The PMS-Application which is developed by Team Black is focused to simplify the everyday life of people with borderline disease and it allows doctors to monitor their patients more easily.
Key Features of the Application are management of patients addresses, manage physicians' appointments, manage medicine intake, monitoring fluctuations and the usability. For emergencies, a function is to be installed which gives the patient a quicker access to the help he needs.
Important is the implementation of the existing legal provisions regarding data protection.
Due to the simple usability/availability, common and widely used software technologies are becoming familiar in development. To meet the requirements, the PMS is to be provided in the form of a smartphone app.

Glossary

Acronym	Meaning
PMS	Patient Management System
GUI	Graphical User Interface

Use Cases Definition

We have defined the following user requirements for our application:

U1. **Display prescribed / non-prescribed drugs**
U2. Add non-prescribed drugs
U3. Edit non-prescribed drugs
U4. Reminder for drugs
U5. Display appointments
U6. Add appointments
U7. Reminder for appointments
U8. SOS Button
U9. **Emotional-state-tracking**
U10. PIN Code

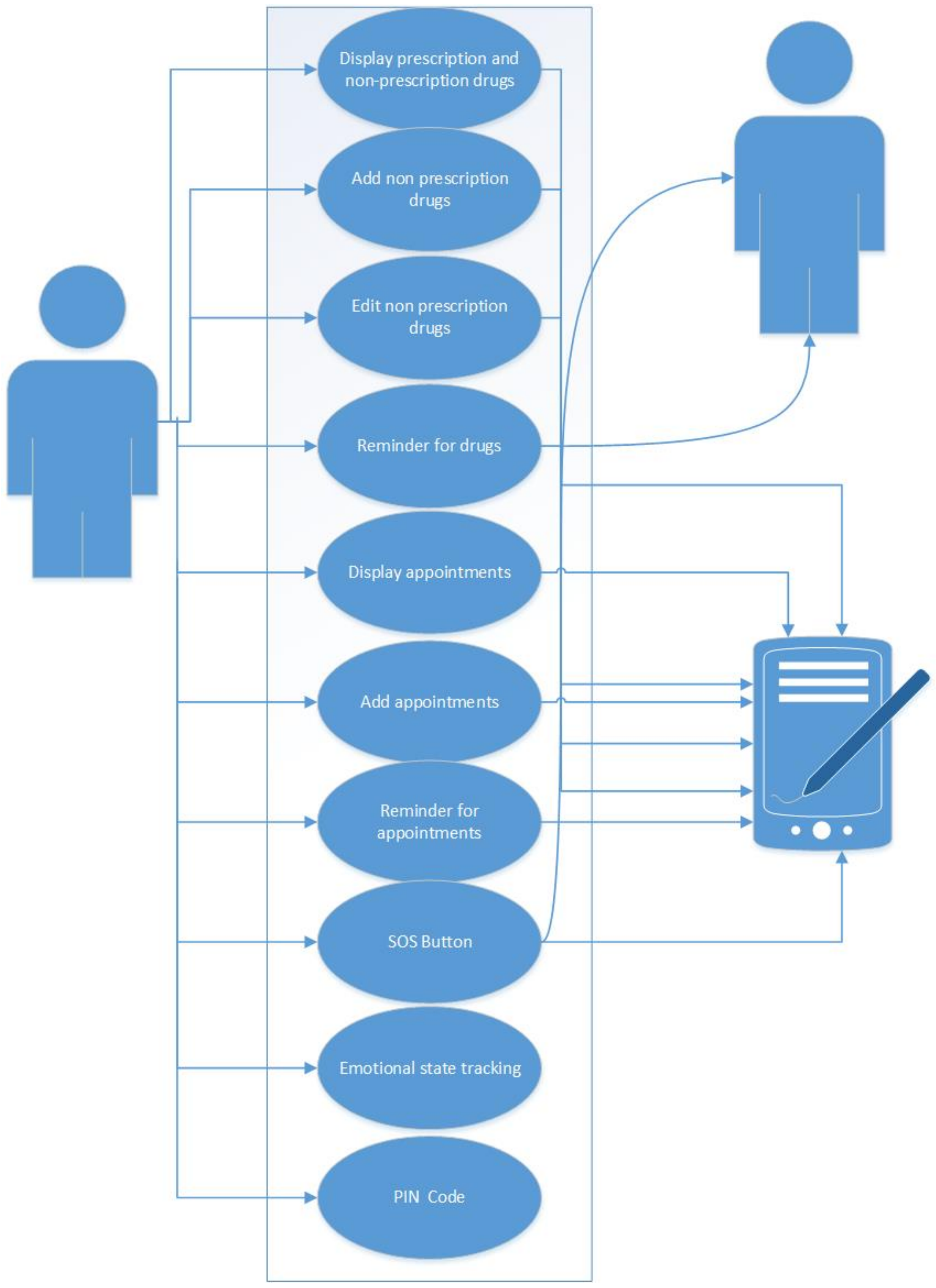

Display prescription and non-prescription drugs
Add non prescription drugs
Edit non prescription drugs
Reminder for drugs
Display appointments
Add appointments
Reminder for appointments
SOS Button
Emotional state tracking
PIN Code

Image 1 Use Case Diagramm

Use Case 1

Nr. and Name:	U1.1
Scenario:	Display prescribed/non-prescribed drugs
Short Description:	Show the amount and frequency of the different drugs
Actors:	Patient
Starting Event and Preconditions:	Application is open
Result and Postconditions:	Patient is informed

Steps:

Nr.	Actor	Description
1.1.1	Patient	Patient opens the drugs section
1.1.2	Patient	Patient enters personal pin-code
1.1.3	Patient	Patient reads which drug he has to take

Exceptions, Variants:

Nr.	Actor	Step
1.1.1	System	System can't load the saved drugs -> Restart the application
1.1.2	Patient	Patient doesn't remember his personal pin-code -> Doctor resets his pin

..

Use Case 2

Nr. and Name:	U1.2
Scenario:	Track emotions of patient
Short Description:	Track the emotion of the patient 3-5 times per day
Actors:	Patient, Doctor
Starting Event and Preconditions:	Application is closed
Result and Postconditions:	Patient and Doctor is informed

Steps:

Nr.	Actor	Description
1.2.1	Doctor	Doctor enables Emotion tracking on Server for patient
1.2.2	Patient	“Emotion-tracking”-popup appears on user device
1.2.3	Patient	Patient selects emotion and confirms
1.2.4	Doctor	Doctor evaluates data

Exceptions, Variants:

Nr.	Actor	Step
1.2.1	System	User does not select an emotion -> Popup remains in foreground

System Architecture

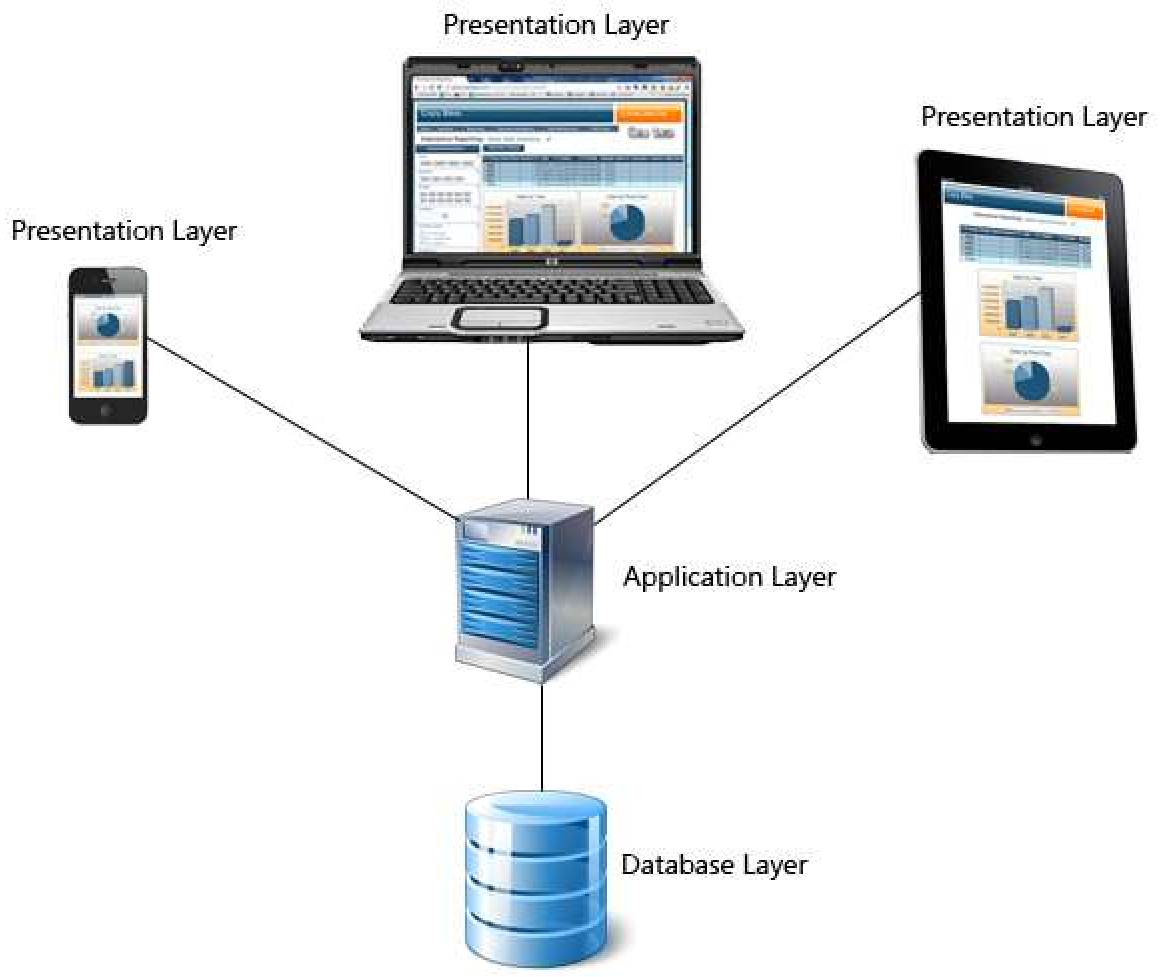


Image 2 System Architecture

Database Layer

The persistent data is stored in a relational database. The DBMS is not given.
In this layer, you take care of the disaster recovery.

Application layer

The application layer serves as abstraction between the presentation and the database layer.
It implements the logic of the application.

Presentation Layer (Mobile/Client)

The presentation layer will display the data given from the application layer. The user is able to do some various actions on it, which will be passed to the application layer. It contains also some simple logic like input validation.

System Requirements Specification

The system manages the following points:

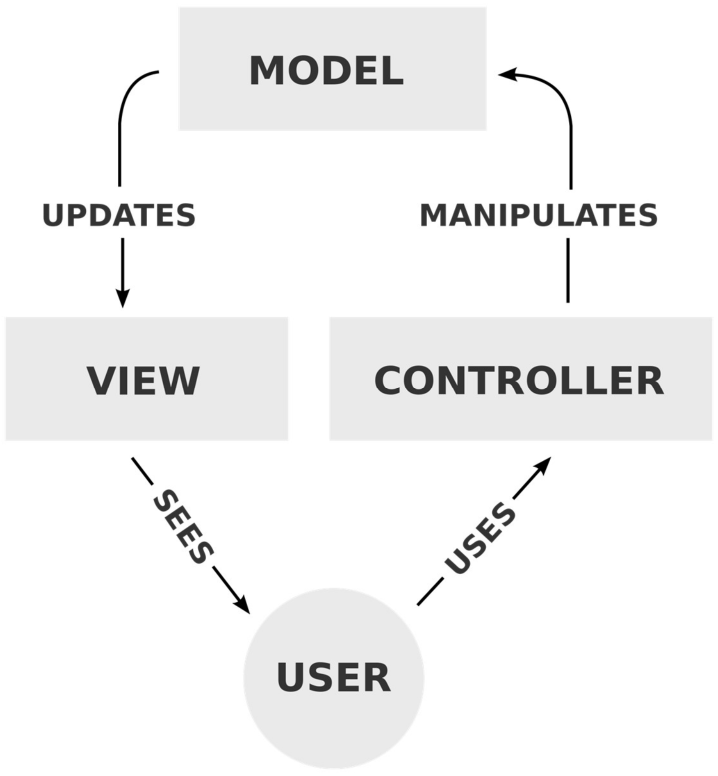
1. Store add and edit prescribed and non-prescribed drugs
2. Schedule and reminder to take drugs
3. Store add and edit appointments
4. Schedule and reminder for appointments
5. Send emergency request to a hospital / doctor
6. Store the emotions (Emotional-state-tracking)
7. Schedule and reminder for emotional-state-tracking
8. Store and check PIN Code
9. Auditing for all actions on the system
10. Simple usability (3 clicks for an action)
11. Using icons and text do display actions
12. Reasonable responsiveness & availability

Non-functional system requirements

- Data-transmission is encrypted
- Programming language is Java (Java 8)
- Relational database
- Disaster-Recovery concept
- Support for Chrome and Firefox
- Support for mobile devices (App, Chrome and Firefox)
- Reasonable scalability

System Model

The application shall be structured in a persistence layer (database), a controller layer (application logic), and a presentation layer (user interface).



Access to the database shall be serialised either by the application or by the persistence layer (eg, RDBMS transactions / ACID). This enables the clear separation between UI, application program code and user data. The database schema shall be normalised, data shall not be duplicated. It should be possible to run the application either standalone or as part of an existing IT infrastructure without interfering with their normal operation.

System Evolution

As this is a school exercise that is most likely going to be mostly useless in six months time (except for the educational value retained in the students, of course) system evolution isn't really an important part to consider.

Testing

No real Patient data will be used for testing until its security can be guaranteed. Because of the special circumstances, no real patients will be used until beta testing and then only supervised. During the development phase, regular Unit-Tests will be performed for the different components. These will be validated before each commit. The validation of the system interfaces will be performed to ensure the regular reviews with the customer are successful. Automated white box Unit-Tests shall be defined and developed before the project begins to ensure the quality can be validated automatically. Integration tests shall be started as early as possible to ensure full system compatibility. In the later phases, the system will be tested as one. Tests will be defined simply by defining inputs and expected outputs following black box procedure. Story testing will be implemented in the end phase to ensure the program works as defined. Here the testing will be done by a third party to ensure neutrality.

Appendices

The user interface will be implemented on the webservers already operated by the customer, extending the infrastructure may be necessary. The database will be implemented in the existing infrastructure, but because of the delicacy of the data, a new server with corresponding back up must be installed. Support of all current portable customer hardware is to be ensured.
The database is adequately powerful to handle all requests in a timely manner. Best practice is implemented to ensure backup and safety is provided.
The webserver is adequately powerful to handle all requests in a timely manner. Adequate back up services to provide 99.9% uptime.
The GUI-Server is adequately powerful to handle all requests in a timely manner
The client hardware can run a reasonably up to date Browser.

Index