“Find My Patient”

Software documentation



Submitted By :

SHAZIL KHAN

GERARD KLOMPHAAR

Table of Contents

[1 Subject description](#h.gjdgxs)

[2 Subject analysis](#h.30j0zll)

[2.1 User requirements](#h.wokgx73vpw9d)

[2.2 Application Feasibility](#h.1fob9te)

[2.3 Data description](#h.62lmuuh7j696)

[2.4 Expected results](#h.2et92p0)

[2.5Algorithms study](#h.tyjcwt)

[2.6 Scope of the application (limits, evolutions)](#h.8yhk0sq9rgev)

[3 Conception](#h.1t3h5sf)

[3.1 Chosen algorithm](#h.4d34og8)

[3.2 Architecture](#h.2s8eyo1)

[Overview](#h.17dp8vu)

[Architectural MVC pattern](#h.lnxbz9)

[DAO&template pattern](#h.35nkun2)

[Search strategy pattern](#h.1ksv4uv)

[3.3 Global application flow](#h.44sinio)

[4 Console operations description](#h.xq3u7ek1x3ga)

[4.1 <One section by operation>](#h.3j2qqm3)

[5. Configuration instructions](#h.1y810tw)

[6. Commented Screenshots](#h.4i7ojhp)

[7. Bibliography](#h.2xcytpi)

# 1. Subject description :

This project is part of the Java course lectures delivered at EPITA. To test and show a sufficient amount of knowledge a software application will be designed and developed.

As basis of the application the content created during the lectures will be used. This is a part of DAO practices covered in java lectures. The functionality will be extended where necessary to create the application.

The application describes complete detail for a patient search throughout the database using DAO. Most of the time in hospital based systems the whole assembly is linked to complete system. Whereas, if you need to place a patient in a specific place, there is a look up for Ward no, Room no, or a special patient’s care unit for his/her placement. We have opted a small module for searching a patient through his own personal details that includes:

* Patient’s ID
* Patient’s SSN
* Firstname
* Lastname
* Date of Birth
* Cell Number
* Email
* Display complete Name
* and specially room he is placed in.

In context to DAO, this approach is a specific target on a patient placement that defines an important module in a hospital’s management.

In the subject analysis the requirements, features and results will be set.The Conception chapter displays the design of the application.

# 2. Subject analysis :

Subject analysis is extracted from the idea of Hospital patient’s search, while a direct look up is required in the management system all the details of the patient as mentioned above in description are taken into account and analysed on the basis of DAO criteria that includes the following :

* CREATE
* READ
* UPDATE
* DELETE

The requirements for the project are subtracted from the link below :

<http://www.thomas-broussard.fr/work/java/courses/project/fundamental.xhtml>

The subject analysis part focusses broadly on the 6 major part of project that are described in details below:

## 2.1 User requirements :

The required features of the application are:

1. Do operations on identity information
   1. Access
   2. Create
   3. Modify
2. Persist Identity data in a database
   1. Database
3. Work process
   1. Be robust
   2. Capable of good performance
4. Authentication
   1. E.g. Login + password.

Additional features:

1. Propose a simple but efficient user interface

## 2.2 Application Feasibility :

To determine the feasibility of the application we look at the requirements.

|  |  |  |
| --- | --- | --- |
| Requirement | Analysis | Risk |
| 1.1, 1.2, 1.3 | These operations are handled during the class and are well known, can be applied in multiple ways. | Low risk |
| 2.1 | Create a database connection and doing operations are handled in class, still there minimal knowledge about connections, drivers and SQL statements in the team. | Medium risk |
| 3.1 | To enforce a robust application unit testing critical parts should be performed.  Code review or pair programming will be done on request by the implementer. | Low risk |
| 3.2 | ‘Good performance’ requirement is a broad and vague specification, especially since the target audience is also unknown.We will re-define good performance for the chosen targets following;   * Error handling and reporting to the end user. * Responsible GUI, (no GUI freezes > 1s) | Medium risk |
| 4.1 | Authentication can be done in multiple ways since there is no specification we will chose the method which is most suitable to the use case. | Low risk |

**Table 1 - Requirement analysis**

## 2.3 Data description :

From the requirements we can extract the necessary data models. There is one obvious model which is the Patient model, this model contains all the details of a single patient. Also a User model will be created which will contain the details and login information about the users of the application.

|  |  |
| --- | --- |
| Patient | User |
| Id (unique) | Id (unique) |
| Front name | Username |
| Last name | Password |
| Social security number | Application rights |
| Telephone number |  |
| Email |  |
| Room |  |

**Table 2 – Data models**

## 2.4 Expected results :

The results that are expected from this project are based on the patient’s details. Once a patient is registered along with all his details. A thorough search mechanism is invoked based on read approach. All of the personal details are compared through a matcher in DAO which compares all the identities with the input entered by authorised users. The results are extracted from the database and displayed on the GUI. On the basis of requirements the update and delete operations are taken into account to get the final expected results.

## 2.5 Algorithms study :

To provide features according to the user requirements certain algorithms are needed.

**Saving a the POJO’s *User* and *Patient :***

Saving the POJO data models *User* and *Patient* can be done with different solutions, the most applicable solutions are:

* Using a normal file text-based
* XML file
* JSON file
* Serialisation
* Database

File based methods are possible but require more effort and are not as flexible as an database. Serialisation of the objects will create too much memory constraints and is not scalable. The application will use a Apache Derby database with a JDBC connection.

**Patient and User data model DAO :**

Saving the data models is done using a DAO abstraction, this provides a loosely coupled design so that the concrete DAO implementation easily can be changed.

In the Application two models need to be saved using a DAO. The basis of both DAO’s will be the same, so to prevent code duplication, the template pattern is used to provide the basis functionalityand the concrete classes will provide the specialisation necessary for the model.

**Searching Patient(s) based on specific fields**

The application will provide basic search functionality. To provide this, different fields of the Patient could be matched against certain criteria. Using the strategy pattern, an interface is provided so that the search functionality can be easily extended when needed.

**Providing a view**

The view provided to the end user can be done in different ways, the two most applicable are:

* Web-based view
* Standalone Application

For the application we will use a standalone application, this is best fit for the target. The architecture between the models, business logic and the view will be created using the MVC architecture as basis.

## 2.6 Scope of the application (limits, evolutions) :

***Limits:***

The application is created for a small amount of parallel users.

***Evolutions:***

Following features could may be interesting for future extensions:

* Use a connection pool for the JDBC connections.
* Create interfaces for Controller and Management class and use Dependency injection.

# 3 Conception :

## 3.1 Chosen algorithm :

In the table below an overview of the chosen algorithms is displayed

|  |  |
| --- | --- |
| Subject | Solution |
| Saving a POJO | DAO + JDBC + Derby |
| Patient and User model DAO | Template pattern |
| Search Patients | Strategy pattern |
| Business logic and view seperation | MVC pattern |
| Telephone number |  |
| Email |  |
| Room |  |

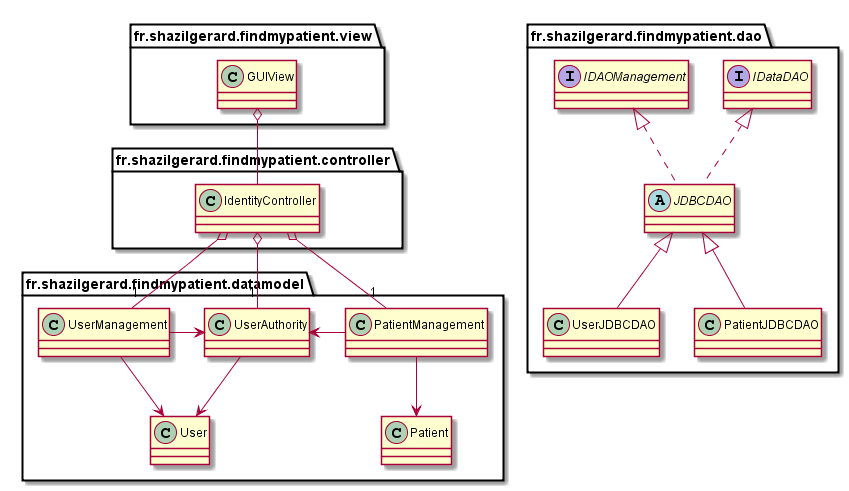
**Table 3 – Data models**

## 

## 3.2 Architecture :

### *Overview*

The general architecture of the application is depicted in Figure 1 below.



**Figure 1 - General application architecture**

### *Architectural MVC pattern:*

The application base design is around the model view controller structure. This should ensure that the business logic is separated from a view. The controller will instantiate all the necessary components to run the business logic.

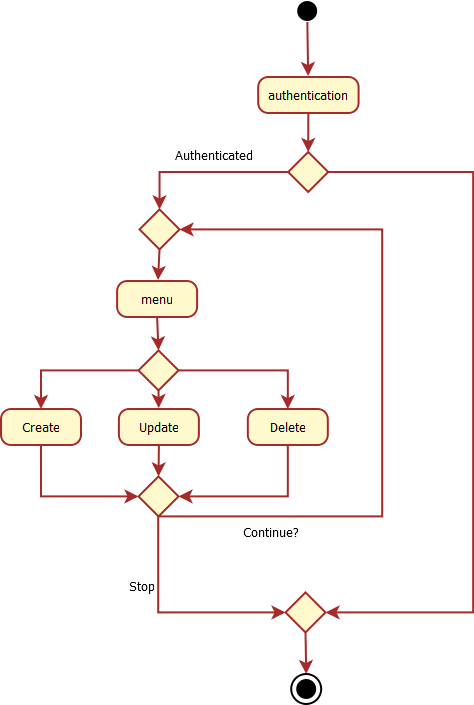
### *DAO&template pattern:*

Since there are two data models which both require the same CRUD operations, it is chosen to create an abstract JDBCDAO which will implement the general CRUD implementation to a database. The specifics for each model will be handled in the specialization classes UserJDBCDAO and PatientJDBCDAO.

### *Search strategy pattern*

Searching for specific fields of both models will be implemented using the strategy pattern. This ensures that new search methods can be easily added.

### 3.3 Global application flow :

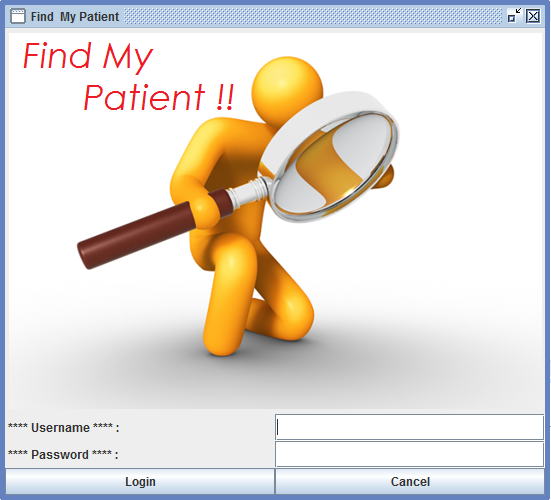


# 4. Console operations description :

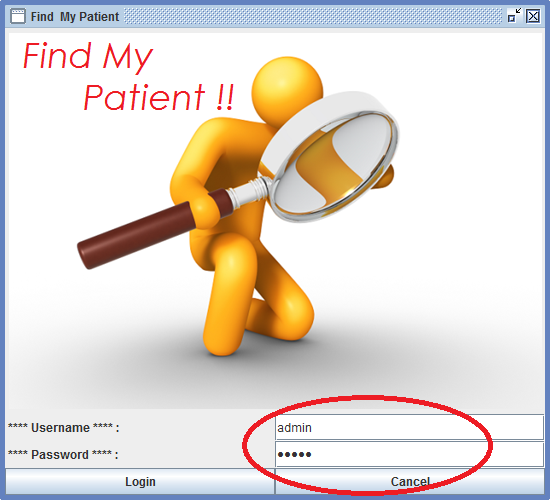
The console operation are described as follows :

## 4.1 <One section by operation>

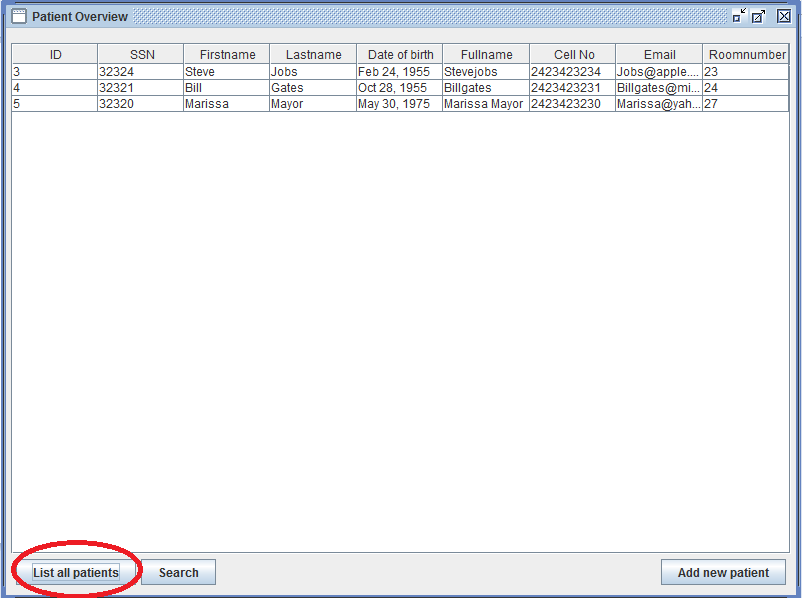
**STEP 1** : When you run the program the following window will appear.



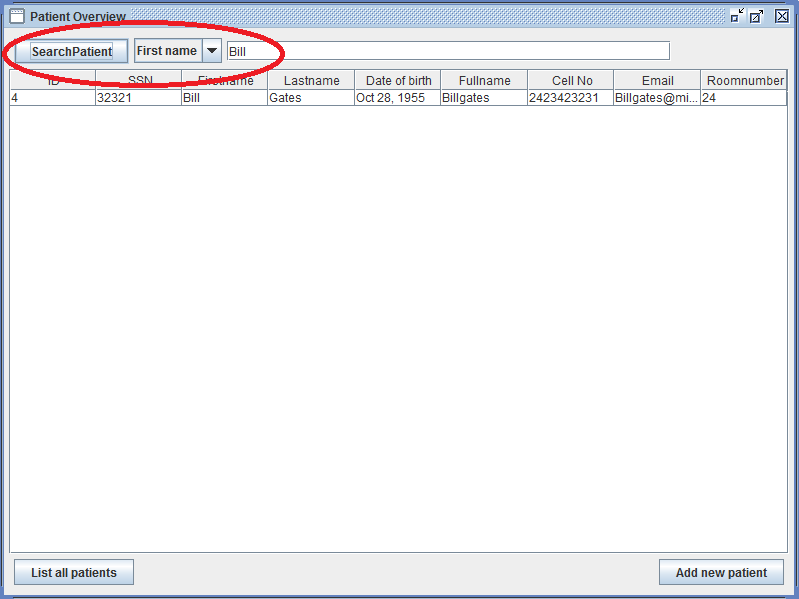
**STEP 2** : When the “User Authorization” windows appear. Please provide your login details against the fields Username and Password. As show below in the red circle.



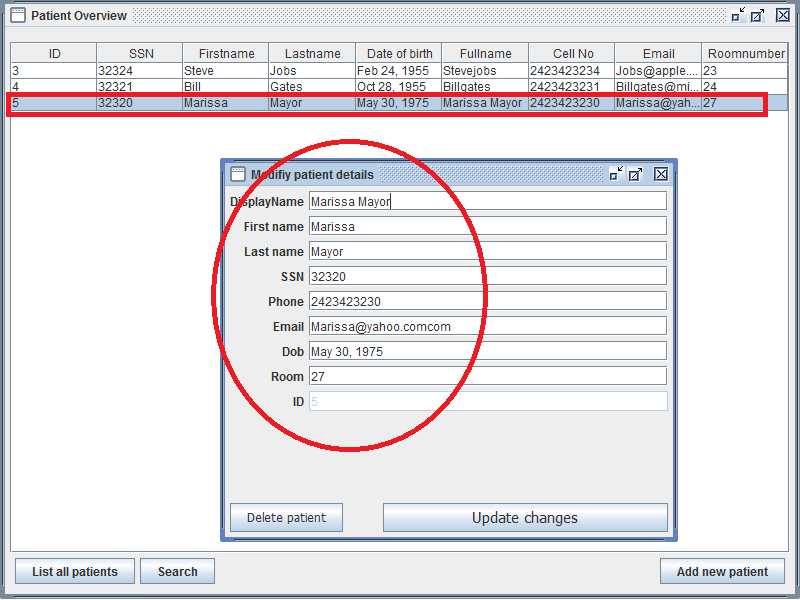
**STEP 3** : When the patient overview window pops up press the “List all patients” button in order for the list to get populated. There you can see all the detail of all the patients registered.



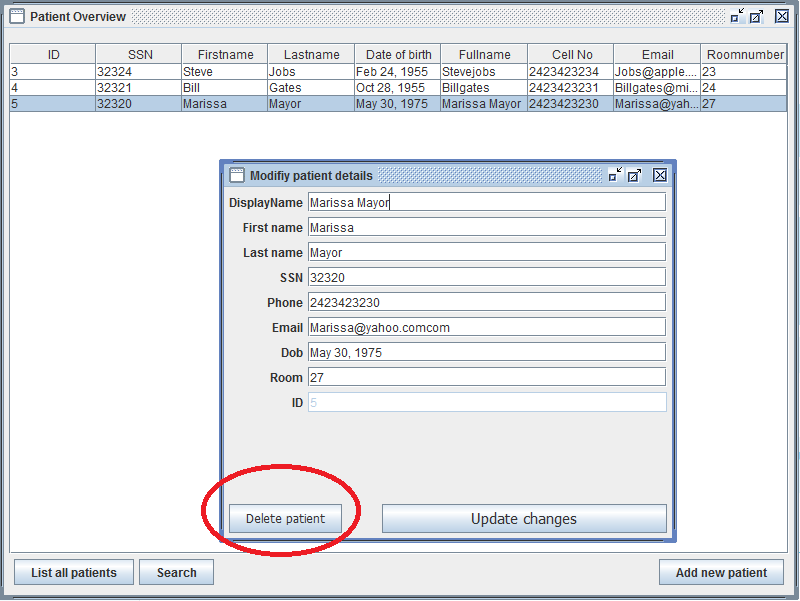
**STEP 4** : This step gives you an option to search by patient details by using the drop down menu from the top left corner.



**STEP 5** : Click a specific field of the person in patient’s overview window or select search by operations from the patient’s overview window and select the details of the person to edit and update.



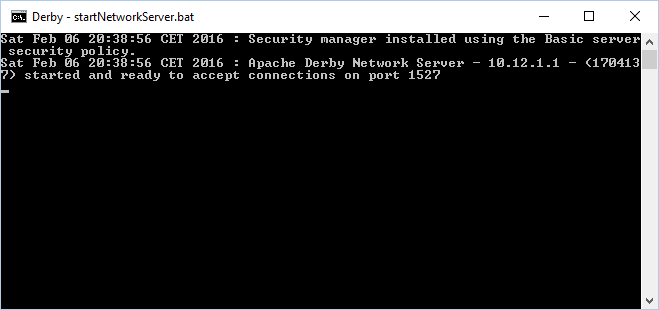
**STEP 6** : There is a delete button in the Modify patient window, while clicking that button all the details of the desired specific user will be erased from the database completely.



# 

# 6.Configuration instructions :

To run the application Derby should be running on the local machine. Make sure its running on the default port 1527 like below.



The zip file contains three main folders:

* DerbyDB
* Documentation
* Sourcecode
* Executable

***DerbyDB*** contains the a test database called ‘FindPatientDB’ this should be placed in the derby bin folder for a quick start.

***Documentation*** contains files and documentation of the project, this document is leading.

***Sourcecode*** contains the application source code

***Executable*** contains an exported and finished version of the’Find my patient’ application.

The default login for the application is:

Username: **admin**

Password: **admin**

# 

# 7. Bibliography

Architectural reference:

<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

<https://en.wikipedia.org/wiki/Dependency_injection>

Java configuration example using properties:

<http://www.mkyong.com/java/java-properties-file-examples/>