Gynaecological Patient Information Management System:

Software Test Documentation

Team Pentec:

Ruth Ojo 12042804
Liz Joseph 10075268
Trevor Austin 11310856
Maria Qumayo 29461775
Lindelo Mapumulo 12002862

Final Version

September 25, 2015

Contents

1 INTRODUCTION

1.1 Objectives

The following doucment contains the detailed outline of the PIMS testing process and results. Bellow are the main modules that were tested.

- User Login
- PIMS Artificial Intelligence
- PIMS Statistics
- PIMS Notifications
- PIMS Space

Testing was done on the system mainly for the following 5 reasons:

- To ensure that the system meets both functional and non functional requirements arroding to the given spesifications
- System stress, that is to make sure that the system does not fail with multiple users or any other factors because it is expensive to resolve and fix at a later stage.
- To handle and resolve System failures and bugs appropriatly and in good time.
- To point out the defects and errors that were made during and after the development phases.
- To ensure that the final product is of top, professiona, software engineering standards.

1.2 Testing Strategy

1.3 Scope

1.4 Reference Material

1.5 Definitions and Acronyms

2 TEST ITEMS

2.1 Program Modules

2.2 Job Control Procedures

2.3 User Procedures

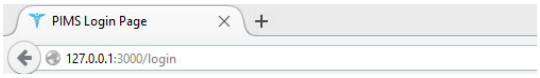
2.4 Operator Procedures

3 FEATURES TO BE TESTED (Functional Requirements)

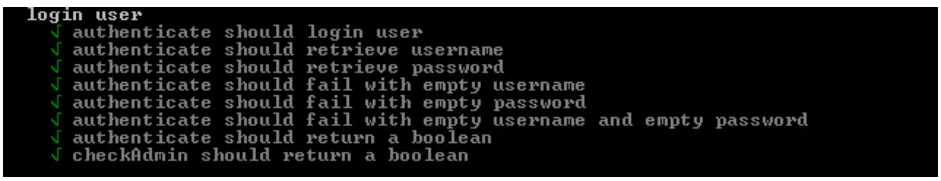
3.1 PIMS Login

It is a priority that a user logs into the system for security reasons to be able to interact with it. When testing the login use case we tested for the user?s authentication and rights. We tested weather or not he has admin or normal user rights.

The local login page



Successfu.l login unit tests



Conditions

The following pre and post conditions are defined for adding a new user.

Pre conditions

- User must not be logged in
- User must have valid credentials, a valid username and password.
- User credentials must be found in the Mongo database.

Post conditions

- User successfully logged in according to his user rights

Code snippets for user login use case, with test data ?a? . Testing that a valid username is entered

```
    it("authenticate should retrieve username", function(done){
      login.authenticate("a", "g", function(err){
          User.findOne({username: "a", password: "g"}, function(found){
              found.username.should.equal("a");
          });

      });
      done();
    });
```

Code snippets for user login use case, with test data ?g? . Testing that a valid password is entered

```
it("authenticate should retrieve password", function(done){
    login.authenticate("a", "g", function(err){
        User.findOne({username: "a", password: "g"}, function(found){
            found.password.should.equal("g");
        });

    });
    done();
});
```

Code snippets for user login use case. Testing that the user is logged in after valid authentication

```
describe("login user", function(){
  it("authenticate should login user", function(done){
      login.authenticate("a", "g", function(err){
          User.findOne({username: "a", password: "g"}, function(found){
              found.username.should.equal("a");
              found.password.should.equal("g");
          });

      });
      done();
  });
```

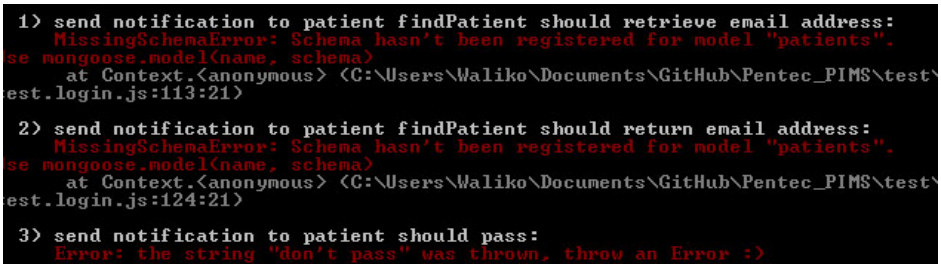
Remark

Login ensures security and access control. Testing regarding the logging in all pass, thus the PIMS system is secure.

3.2 PIMS Notifications

Send notification is a feature that is not in the requirement documentation but was asked to be added in by our client. It allows him to send notifications to remind patience of their next follow up. The send notification use case unit tests failed because the schema could not be found, see figure bellow.

Test failure causes



The following pre and post conditions for the send notification must hold true.

Conditions

The following pre and post conditions are defined for adding a new user.

Pre conditions

- User must be logged in as administrator.
- Patient must exist in the system database.
- Patient must have an active email account.

Post conditions

- A notification informing patient about their next appointment is sent to the users email account.

Code snippets for send notification use case that looks for a patient email address in the database to send a notification to.

```
it("findPatient should return email address", function(done){
    notification.findPatient("sue", function(found){
        found.should.equal("nodemailingtest@gmail.com");
    });
    done();
});
```

Remark

Since Send notification fails unit testing, the module needs to be revised and tested again.

3.3 PIMS Edit Profile

3.3.1 Pims Login

Pims edit user profile should be able to allow the admin user to update his profile and edit his information accordingly.

The Code bellow demonstrates the update of the user name after re-trienving it.

```
describe("update profile", function(){
    it("should retrieve username", function(done){
        User.findOne({username: "Leon"}, function(err, contact) {
            should.not.exist(err);
            contact.username.should.equal("Leon");
        });
        done();
    });

    it("should modify profile details [surname]", function(done){
        User.findOne({username: "Leon"}, function(err, contact) {
            contact.surname = "Snymanss"
            should.not.exist(err);
            contact.surname.should.equal("Snymanss");
```

Edit profile was tested for the following conditions

- Retrieve data
- Modify profile details

The figure bellow depicts the successful testing of the UpdateAuthentication and checkAdmin functions.

```
update profile
  √ should retrieve username
  √ should modify profile details [surname]
  √ should modify profile details [password]
  √ should modify profile details [email]
  √ should modify profile details [user_rights]
```

3.4 PIMS Add User

Adding a user, is a feature only available for the admin user and is a crucial use-case needed for saving a new users to the system.

The add user case passed unit testing successfully as seen in the figure bellow

```
add new user
  √ should save user details in database
```

Conditions

The following pre and post conditions are defined for adding a new user.

Pre conditions

- User must be logged in as admin.
- User to be added must not already exist in the database.
- User must be a medical personnel.

Post conditions

- New user is added and can interact with the system.

The code bellow shows the testing for adding a new user with sample data

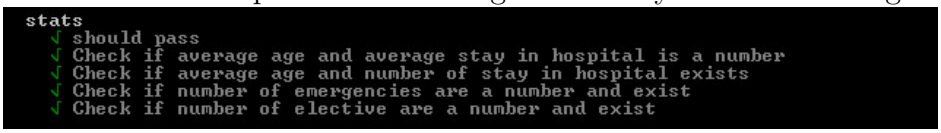
```
describe("login user", function(){
  it("authenticate should login user", function(done){
     login.authenticate("a", "g", function(err){
        User.findOne({username: "a", password: "g"}, function(found){
            found.username.should.equal("a");
            found.password.should.equal("g");
        });

     });
     done();
  });
```

Remark

Add User unit test successfully passes.

3.5 PIMS Statistics

PIMS statistics is another feature for admin user only. For the statistics use case we tested for four different cases. All passed unit test and succeeded.. The add user case passed unit testing successfully as seen in the figure bellow



Conditions

The following conditions had to be met for the statistics tests to pass.

Pre conditions

- User must be logged in as admin.
- Object type has to be a digit.

Post conditions

- Statistical results are returned.

Unit test code to validate the number of elective procedures is indeed a number.

```
it("Check if number of elective are a number and exist", function(done){
    GS.aggregate(
     [{ $match : {"typeOfProcedure.Elective": true , "ProcedureDate": {'$gte': new Date("2014-01-02"),'$lte': new Date("2014-01-2:
{$group : { _id : { month: { $month: "$ProcedureDate" }, day: { $dayOfMonth: "$ProcedureDate" }, year: { $year: "$ProcedureDate" }}

                }] , function(err, myResult)
    {
        should.exist(myResult);
          var num = myResult.length;

          for (var i = 0; i < num; i++) {
                var newElement = {};
                    newElement['date'] = new Date(myResult[i].ourDate).toString('dd-MM-yyyy');
                    newElement['close'] = myResult[i].count;
                    arr.push(newElement);
                    }
         var resBody = { myStatsArry: arr};
          console.log(resBody);
          should.exist(resBody);
          res.json(resBody);
          console.log("POST response sent.");}
    );
          done();
});
```

Unit test codes to validate the average age and stay in hospitals are indeed numbers.

```
it("Check if average age and number of stay in hospital exists", function(done){
          AD.aggregate(
        {$group: {
                    "_id": 1,
                    avgAge   : { $avg: "$Age"  }}
        }, function(err, avg)
        {if (err){throw err;
                res.redirect('stats');}
                else{
                        AD.aggregate(
                            {$group: {
                                "_id": 1,
                                avgStay   : { $avg: "$TotalNumberOfDaysHospital"  }
                                }}, function(err, avgStay)
                            {if (err){
                                throw err;
                                res.redirect('stats');}
                                    else{
                                        var average = JSON.stringify(avg[0].avgAge);
                                        var averageStay = JSON.stringify(avgStay[0].avgStay);
                                        should.exist(average);
                                        should.exist(averageStay);}});}
                            });
                done();
    });
```

Remark

Statistics is the heaviest module the PIMS. More unit testing will be done to ensure accuracy, reliability and currency. For now all tests succeeded and passed.

3.6 PIMS Artificial Inteligence

3.7 PIMS MyAdminSpace

3.8 PIMS MySpace

3.9 PIMSFORMS SaveForLater

3.10 PIMSFORMS SubmitForm

3.11 PIMSFORMS CancelForm

3.12 PIMSFORMS UpdateForm

4 FEATURES NOT TO BE TESTED (Non-Functional Requirements)

4.1 Usability

4.2 Scalability

4.3 Performance

4.4 maintainability

4.5 Reliability

4.6 Secutity

4.7 Monitorability

4.8 Extendability

5 APPROACH

5.1 Component Testing

5.2 Integration Testing

5.3 Conversion Testing

5.4 Job Stream Testing

5.5 Interface Testing

5.6 Recovery Testing

5.7 Performance Testing

5.8 Regression Testing

5.9 Acceptance Testing

6 PASS / FAIL CRITERIA

6.1 Suspension Criteria

6.2 Resumption Criteria

6.3 Approval Criteria

7 Testing Process

7.1 Test Deliverables

7.2 Testing Tasks

7.3 Responsibilities

7.4 Resources

7.5 Schedule

8 Environmental Requirements

8.1 Hardware

8.2 Software

8.3 Security

8.4 Tools

8.5 Publications

8.6 Risks and Assumptions

9 Change Management Procedures

10 Remarks

10.1 Risks and issues

10.2 Product quality

10.3 Possible improvements

11 Conclusion