Testing

Healthcare Record Implementing Blockchain

3rd Year Project

Traian Svinti 14432128

David Talan 14387991

Supervisor: Geoff Hamilton

Introduction

In the process of creating the proposed system, many bugs and problems where encountered due to insufficient knowledge of the tools that were used and also due to the fact that many of these tools are not well documented and relatively new.

To make sure the system was as fully functional and bug free as possible we carried out multiple types of testing such as; Integration testing, Unit testing and user interface testing.

Integration Testing

Integration testing was carried out to make sure that the different components, when put together worked correctly. The main integration being done being the connection between the Django component and the non-SQL database, MongoDB.

The original setup was Django was done using the build in database SQLlite. Knowing that Django supports multiple databases, there was no perceived problem of incorporating a non-SQL database. When integrating MongoDB, it was found that there is "no streamlined way of using MongoDB". To set up the build in database the code below was needed in the settings.py file in Django.

```
87  DATABASES = {
88      'default': {
89          'ENGINE': 'django.db.backends.sqlite3',
90          'NAME': 'mydatabase',
91      }
92  }
```

After Django was fully set up, online resources where found that said simply changing the name would change the database, of course given that the database was installed. This proved to not work.

Struggling to integrate our backend with our database, the group found a tool called Djongo, made specifically for both Django and MongoDB. Once this tool was installed simply changing the name in the settings.py file to point towards Djongo resolved the problem while fully incorporating MongoDB as our main database.

```
80  DATABASES = {
81      'default': {
82          'ENGINE': 'djongo',
83          'NAME': 'my_database',
84      }
85  }
```

To ensure that Djongo was working as intended with Django and MongoDB we decided to try to run each system independently as when Django was started it would automatically run MongoDB and to be sure this was working we uninstalled MongoDB and the system would not working clarifying that Djongo was attempting to run MongoDB but failed as it was not installed.

User Interface Testing

The majority of the project relies on the User Interface of the web application. It must display the user's medical record in a presentable manner so that everything is coherent and responsive as the group wanted.

Initial UI prototype

The group created an early iteration of the prototype that had the main functionalities that the group intended to have. Images were added, a log in page, the 'New Patient' and 'Existing Patient' pages were also implemented.

Log In Page Tests

When testing the log in page, the group encountered a problem where the Enter button does not let you log in. The only way that a user can log in was when they press the log in button itself. To fix the bug itself, the group implemented a javascript logIn() function to handle this.

```
function logIn(){
   var usr = document.getElementById("userIn").value;
   var pass = document.getElementById("passwordDoctor").value;
   if (usr == "doctor" && pass == "a"){
      return true;
   }
   else{
      alert("Login Invalid - Try again.");
   }
   return false;
}
```

Before the Javascript function was added, the group discovered that the <form> tag must be modified to include an 'onsubmit' event. This 'onsubmit' also required an input type 'submit' in the closing form tag. The onsubmit = 'return logIn()' triggers the function and executes the code in the image above.

The function gets the value of the two forms, username and password, and compares them to certain values. In the initial iteration, the group only set the username as "doctor" and the password as "a" to let the user log in.

Here are some **decision coverage testing** test cases used to check if the logIn() function works. It will simply check the 'if' statement in the function to make sure it evaluates to both the true and false values.

Test Case (Username)	Input	Expected Output	Output
1	doctor	Redirect to next page (valid log in)	Redirected
2	David	Invalid log in	Invalid log in
3	12345	Invalid log in	Invalid log in

Test Case (Username)	Input	Expected Output	Output
1	a	Redirect to next page (valid log in)	Redirected
2	abcd	Invalid log in	Invalid log in
3	12345	Invalid log in	Invalid log in

Another problem related to the log in process was the Log In button at the bottom of the page. Whenever the log in button was pressed, it would log in and redirect the user, even if the log in details were incorrect. This was simply fixed by changing the input type in the button tag to “onsubmit” so that it uses the logIn() Javascript function.

Other Tests

For the ‘New Patient’ page, the group created forms to be filled in. The group wanted the forms to be filled in to create a detailed profile for a user. A bug that was discovered during testing was that the user could press the ‘next’ button at the bottom of the page, even though the forms weren’t filled in fully. This was easily resolved within the tags, only needing to add ‘required’ within the input type tag.

This was tested using Equivalence testing. The test cases were grouped into two categories, having less than 10 inputs for the 10 forms required, and having all 10 inputs for the forms required. The grouping was done like this because the group expected the button will only work if all forms are filled in, creating one group. Any other number of forms filled in, if it was less than the 10 forms require, the group expected to throw an invalid submission. This created the second group.

Test Case	Input	Expected output	Output
1	Less than 10 forms filled in	Invalid submission	Invalid submission
2	All 10 forms filled in	Valid submission, redirect to next page	Valid submission, redirect to next page

Final Prototype UI Testing

When testing the Final prototype of our UI the group wanted to make sure that every small aspect was tested whether working/implemented fully or not.

Forgot password on login screen

When the user accesses the login page they are show input fields for their 'username' and 'password'. Underneath these fields is a button labelled 'Forgot password?'. We tested this button with the results below:

Test Case	Input	Expected output	Output
1	Pressing 'Forgot password?' button	Redirect to password retrieval	null

The team did not have sufficient time to implement this feature as it was not such a priority compared to other features such as working logins and User connected to user profiles.

Restricting username and password length

In the initial brainstorm and to keep the environment of a simple and user-friendly user experience, the group believe that 'username' and 'password' input fields on the login screen should be restricted to 15 characters.

Test Case	Input	Expected output	Output
1	Username input	15 characters maximum	~infinite

2	Password Input	15 characters maximum	~infinite

This was yet another concept which did not get implemented due to insufficient time and different priorities.

Logout button not showing confirmed logout

The logout button was implemented using Django Post functions and not a standard 'href' redirect like in html. Due to this the logout button functioned correctly but did not confirm to the user that logout was successful.

Test Case	Input	Expected output	Output
1	Clicking Logout button	Confirmation of logout	Logout without confirmation

The logout button was confirmed to be working by copying the url which is http://127.0.0.1:8000/personalInfo/details/ and logging out. Once the user was logged out the url was pasted back in but instead of going to the details page, the login page will be displayed.

Adding doctor button not functional

On the details page on the "Previous Doctors" tab, we are presented with 2 doctors taken from the database as current and last doctor. This information can be edited by the doctor on the admin page, specifically on the User's Profile.

Test Case	Input	Expected output	Output
1	Clicking '+' button	New doctor information to be inputted	null

Automatic redirection after creating new patient to patient information

When a doctor logs in with his admin account and creates a new user, the doctor must then press 'home' and then 'User Profile' and select the new patient to add information to.

Test Case	Input	Expected output	Output
1	Adding new user	Redirects to patient profile information	Patient created confirmation

Added ability to have empty forms

Once a doctor creates a new patient, the group encountered a problem where a profile could not be created due to missing information from fields. This was fixed by adding 'blank=true' in the model fields.

Test Case	Input	Expected output	Output
1	Creating new user	User profile is created	User profile created.