University of the Witwatersrand

Requirements Analysis Document

Team Name:

Level Seven Crew

Product Name:

HealFolio

Team Members:

Jan Badenhorst
Adam Lerumo
Tumbone Asukile
Daniel da Silva

Senior Lecturer:

Dr. Terence van Zyl

August 14, 2016

Contents

1 Introduction

1.1 *Purpose of the System*

The software *(HealFolio)* is aimed at improving service delivery for all types of medical practitioners; specifically the filing system currently used by medical practitioners. This software will provide doctors with easier access to patient records and can provide more security. It will also help them make more accurate diagnoses, leading to better prescriptions and more, when the software is further developed.

1.2 *Scope of the System*

The software will be use by doctors, assistants and even patients themselves. By using a web based application the each of the following users of the software can easily get to and use the information by accessing the user friendly interface. This will make it possible for both the service provider and receiver to give accurate diagnosis and critical feedback

1.3 *Objectives and Success Criteria of the Project*

Objectives:

- To make the precious minutes count in an emergency and avoid any long term complications.
- Take away the unnecessary revisiting or come backs to doctors to a minimal.
- Get accurate diagnosis at the first visit for recurring problems.
- Taking the admin work away for patients, especially for the elderly.

Success:

- Making the software easy to use and understand.
- Open to adaptations and future upgrades.
- Robust and workable on most platforms. *(Mobile friendly, Linux)*
- Having the system to be alert in failure to connect to server, which makes it virtually impossible to lose information.

1.4 *References*

Software Engineering

1.5 *Overview*

Utilizing the resources available and time to create the optimal running application that will incorporate every feature desired by the clients. Keeping in mind future growth in user numbers by keeping efficiency to the maximum.

2 Current System

2.1 *Private Practices*

Most practices keep electronic records of basic patient information, including but not limited to:

- ID number
- Name
- Visitations
- Medical aid number
- Medical aid scheme
- Prescriptions

These systems are largely primitive in nature and have only the bare minimum information required to operate. They don't have a lot of functionality in terms of interactive notifications. For example, it is a common problem that some patients get treatment on another patient's medical aid.

It is also important to note that these systems are practice-specific. A patient's file would have to be formally requested by another practice if the patient goes elsewhere. This is critical because some medications interact poorly with eachother and can be *fatal*.

A large concern for a centralized system is *security*. The idea of having everyone's medical history saved to a server and accessed by any doctor is dangerous ground, security would be paramount to ensure legal compliance, for example-legally-a person has sole discretion on whether to disclose their HIV status or not.

Another concern is *ease of use*. A standard consultation with a GP is ten to fifteen minutes long. The current system has been streamlined to a great proportion due to the time taken to enter the information. The age-old add-age *time is money* applies; especially in a profession as highly incentivised as this.

2.2 *Government Practices*

Government practices, at least in the Ekhuruleni Municipality, share a centralized database of patient information. It is currently unknown what information is kept however this is largely irrelevant due to the fact that the system is not commonly used to it's full potential. This is mainly due to the high volume of patients at government hospitals as a result of external factors like poverty in the country.

This re-iterates the importance of *ease of use* in that doctors can not be spending large proportions of their time capturing data.

The concerns in the *Private Practices* sector above generalize to government practices. In summary, the only differences are:

- Government practices already have a central system
- This system is not well used and likely not easy to operate

3 Proposed system

3.1 *Overview*

The proposed system aims to address issues with the current two systems (Private *and* Government practices) while also connecting the two and making patient history ultra portable. Secondary to these tasks is enabling patients to view their own history as well as practice information. This *may* be integrated into the main application later.

3.2 *Functional Requirements*

The functional requirements are generalized as follows:

- All information on patients must be kept in a database accessible to the application at all times
- Doctor able to view patient information including but not limited to:
  - ID number
  - Name
  - Allergies
  - Date of birth
  - Age (calculated)
  - Gender
  - Race
  - Medical aid provider
  - Medical aid scheme
  - Medical aid number
  - Main member ID number
  - Main member name
  - Main member medical aid number
- Doctor able to view patient medical records which consists of the following:
  - Visitations
  - Diagnoses
  - Prescriptions
- Doctor able to view reports such as:
  - Chronic conditions
  - Critical information (eg Allergies, gender, diagnoses etc)
  - Family medical history?
- Doctor able to add information for an existing patient such as a new visitation, diagnoses, prescriptions.
- Doctor able to add new patients.

Requirements

Identifier	Priority	Requirement
REQ1	Done	Doctor be able to view own added patient information.
REQ2	Done	Doctor able to view patient information.
REQ3	Discuss	Doctor able to view various reports.
REQ4	Done	Doctor able to add new patients.
REQ5	Done	Doctor able to log in to system.
REQ6	Discuss	Doctor able to update patient information.
REQ7	Done	Administrator adding doctor.
REQ8	Done	Doctor able to sign-up.
REQ9	Done	Patient able to sign-up.
REQ10	Done	Doctor able to view information from referred patient.
REQ11	4	Add verification for doctor to view referred patient information.
REQ12	5	Patient able to update patient information.
REQ13	3	Patient able to remove doctors that has privileges to view the patients information.
REQ14	Done	Doctor able to add practice information.
REQ15	1	Add automatic verification for doctor of successful sign-up.
REQ16	2	Add input verification for sign-up.
REQ17	2	Add input verification for log-in.
REQ18	Done	Patient able to log in to system.
REQ19	5	Doctor able to update practice details.
REQ20	Done	Doctor able to add diagnosis.
REQ21	Done	Doctor able to add prescription.
REQ22	Done	Application loading screen on log-in.
REQ23	Done	Add follow-up function for previous diagnosis.

3.3 *Nonfunctional Requirements*

Nonfunctional requirements identified by the shortfalls of the current system and general knowledge of the field are as follows:

3.3.1 *Usability and Interface*

Usability and Interface are two key elements in the success of the project. They are important in ensuring a doctor's time is used effectively and not wasted on capturing data. They are also key in ensuring the system is actually used, especially in high volume scenarios such as government hospitals.

3.3.2 *Reliability*

The system needs to be reliable to ensure that there are no gaps in the patient's history which could result in medication clashes and other undesirable circumstances. A likely solution would be to cache data entered locally if the server is down and then send it to the server once it is restored. Likewise the data for patients who regularly attend a given practice can be cached at the practice in case the systems go down. This would minimize the impact of downtime on the doctors and patients.

3.3.3 *Performance*

Performance is important in that a doctor cannot wait long periods of time for a patient's file to load. Caching data for patients who attend a particular practice can help with this, however if the patient is new to the practice performance and reliability could be at stake.

3.3.4 *Supportability*

In software engineering and hardware engineering, serviceability (also known as supportability,) is one of the -ilities or aspects (from IBM's RAS(U) (Reliability, Availability, Serviceability, and Usability)). It refers to the ability of technical support personnel to install, configure, and monitor computer products, identify exceptions or faults, debug or isolate faults to root cause analysis, and provide hardware or software maintenance in pursuit of solving a problem and restoring the product into service. Incorporating serviceability facilitating features typically results in more efficient product maintenance and reduces operational costs and maintains business continuity.

3.3.5 *Implementation*

The implementation needs to be easy as a patient might switch from a practice using the system to a practice that isn't yet. In this case the patient may insist on the new practice using the system at least for them. The doctor would need to be able to pull the patient's records without having to download an application or sit through an extended sign up process.

3.3.6 *Legal*

The system would have to adhere to many regulations and laws in place such as the privacy of a patient's HIV status. A decision has to be made as to whether something is declared the doctor's responsibility or the application is made more secure. A good example would be whether or not any doctor can access any patient's information. The responsibility could either be placed on doctors to not use the database for personal reasons or it could be made secure by only allowing doctors access to a patient's information when a patient grants it to them.

3.4 *System Models*

3.4.1 *Scenarios*

- The doctor needs to sign-up for the application to recognize and verify the particular practice number.

3.4.2 *Use Case Model*

Use cases

Actor	Actor's goal	Use case name
Doctor	Doctor able to sign-up and add practice details.	DocSign(UC1)
Doctor	Doctor able to log-in and add a patient.	DocLog(UC2)
Patient	Patient able to sign-up and add personal details.	PatSign(UC3)
Doctor	Doctor able to able to view patients information.	DocView(UC4)
Doctor	Doctor able to able to view referred patients information.	DocRef(UC5)
Doctor	Doctor able to log-in, view patient list and add diagnosis to relevant patient.	DocDiag(UC6)
Doctor	Doctor able to log-in, view patient list and add prescription to relevant diagnosis.	DocScript(UC7)

Detailed Use Case 1

Use Case: UC1 **Name/Identifier:** DocSign(UC1)
Related Requirements: REQ5, REQ8, REQ14 **Initiating Actor:** Doctor **Actors Goal:** Doctor able to sign-up and add practice details. **Participating Actors:** Admin will check to see if the practice is registered for the doctor to sign-up. **Preconditions:** That the application has no practice of the same description. **Post-conditions:** That the application has registered the practice and the doctor is able to add the practice information. **Flow of Events for Main Success Scenario:** → The doctor finds the home page and clicks the sign-up button. ← The client is taken to the sign-up page. → After entering the sign-up page, the practice details are inserted. ← Admin checks if the practice is registered and adds the doctor to the database. → If the practice is registered the doctor will be able to add patients by pressing the add patient button. ← The client is taken to the add patient page where the doctors patients are added. **Flow of Events for Extensions (Alternate Scenarios):** → If the practice is not registered. ← Admin will send an email to the client to of unsuccessful sign-up.

Detailed Use Case 2

Use Case: UC2 **Name/Identifier:** DocLog(UC2)
Related Requirements: REQ1, REQ4, REQ5 **Initiating Actor:** Doctor **Actors Goal:** Doctor able to log-in and add a patient. **Preconditions:** That the applicant has to be signed-up to be able to log-in and add patient. **Post-conditions:** Patient details can be added to the practice records. **Flow of Events for Main Success Scenario:** → The doctor finds the home page and clicks the log-in button. ← The client is taken to the log-in page. → After entering logging-in, the practice details can be viewed, as well as the add patient button. ← Clicking the add patient button takes the client to corresponding page and the patient details can be added. **Flow of Events for Extensions (Alternate Scenarios):** → If the incorrect log-in details are entered, the client won't be able to proceed.

Detailed Use Case 3

Use Case: UC3 **Name/Identifier:** PatSign(UC3)
Related Requirements: REQ9 **Initiating Actor:** Patient **Actors Goal:** Patient able to sign-up and add personal details. **Preconditions:** That the application has no patient of the same description. **Post-conditions:** That the application has registered the patient and save the client information. **Flow of Events for Main Success Scenario:** → The patient finds the home page and clicks the sign-up button. ← The client is taken to the sign-up page. → Client adds personal details and presses the save button. ← The client is taken to the patient's page. **Flow of Events for Extensions (Alternate Scenarios):** → If the incorrect sign-up details are entered, the client won't be able to proceed.

Detailed Use Case 4

Use Case: UC4 **Name/Identifier:** DocView(UC4)
Related Requirements: REQ2, REQ5 **Initiating Actor:** Doctor **Actors Goal:** Doctor able to able to view patients information. **Preconditions:** That the applicant has to be logged-in and patient information available. **Post-conditions:** Patient saved information can be viewed. **Flow of Events for Main Success Scenario:** → The doctor logs-in. ← The client is taken to the doctor's page. → Client clicks on the patients page. ← View of all patients can be seen, client proceeds to click on relevant patient. → Relevant patient information is displayed. can be added. **Flow of Events for Extensions (Alternate Scenarios):** → If the relevant patient is not on the database or authenticated then no information is displayed.

Detailed Use Case 5

Use Case: UC5 **Name/Identifier:** DocRef(UC5)
Related Requirements: REQ5, REQ10 **Initiating Actor:** Doctor **Actors Goal:** Doctor able to able to view referred patients information. **Preconditions:** That the applicant and patient has to be registered as well as patient authentication given. **Post-conditions:** Patient saved information can be viewed. **Flow of Events for Main Success Scenario:** → The doctor logs-in. ← The client is taken to the doctor's page. → Client clicks on the add patients page. ← Referred patient is added and verification is sent to patient. → Patient receives verification and accepts. ← Client can now see referred patient information. **Flow of Events for Extensions (Alternate Scenarios):** → If referred patient does not verify, doctor won't be able to see information.

Detailed Use Case 6

Use Case: UC6 **Name/Identifier:** DocDiag(UC6)
Related Requirements: REQ2, REQ5, REQ20 **Initiating Actor:** Doctor **Actors Goal:** Doctor able to log-in, view patient list and add diagnosis to relevant patient. **Preconditions:** That the applicant and patient has to be registered. **Post-conditions:** Patient will be saved by application. **Flow of Events for Main Success Scenario:** → From the patient page the client clicks on relevant patient. ← The client is taken to the patient information page. → Client clicks on the add diagnosis button. ← The diagnosis page is displayed, filled in and saved. **Flow of Events for Extensions (Alternate Scenarios):** → No diagnosis can be given if the patient does not exist.

Detailed Use Case 7

Use Case: UC7 **Name/Identifier:** DocScript(UC7)
Related Requirements: REQ5, REQ20, REQ21 **Initiating Actor:** Doctor **Actors Goal:** Doctor able to log-in, view patient list and add prescription to relevant diagnosis. **Preconditions:** The patient has to have a diagnosis. **Post-conditions:** Prescription will be saved by application. **Flow of Events for Main Success Scenario:** → From the patient page the client clicks on relevant patient. ← The client is taken to the patient information page. → Client clicks on the add diagnosis button. ← The diagnosis page is displayed, filled in and clicks on the prescription button. → The prescription box is opened. ← Prescription is filled in and saved. **Flow of Events for Extensions (Alternate Scenarios):** → No prescription can be given if the patient does not have a diagnosis.

Figure 1: Use case diagram (Account management)

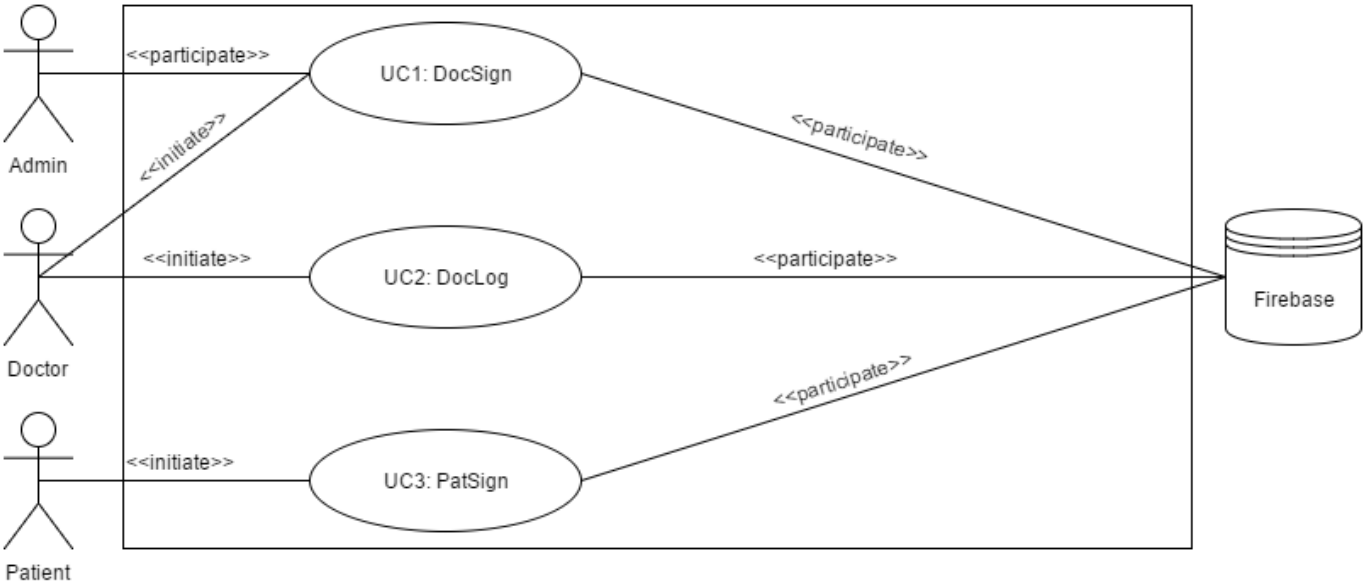


Figure 2: Use case diagram (Referral)

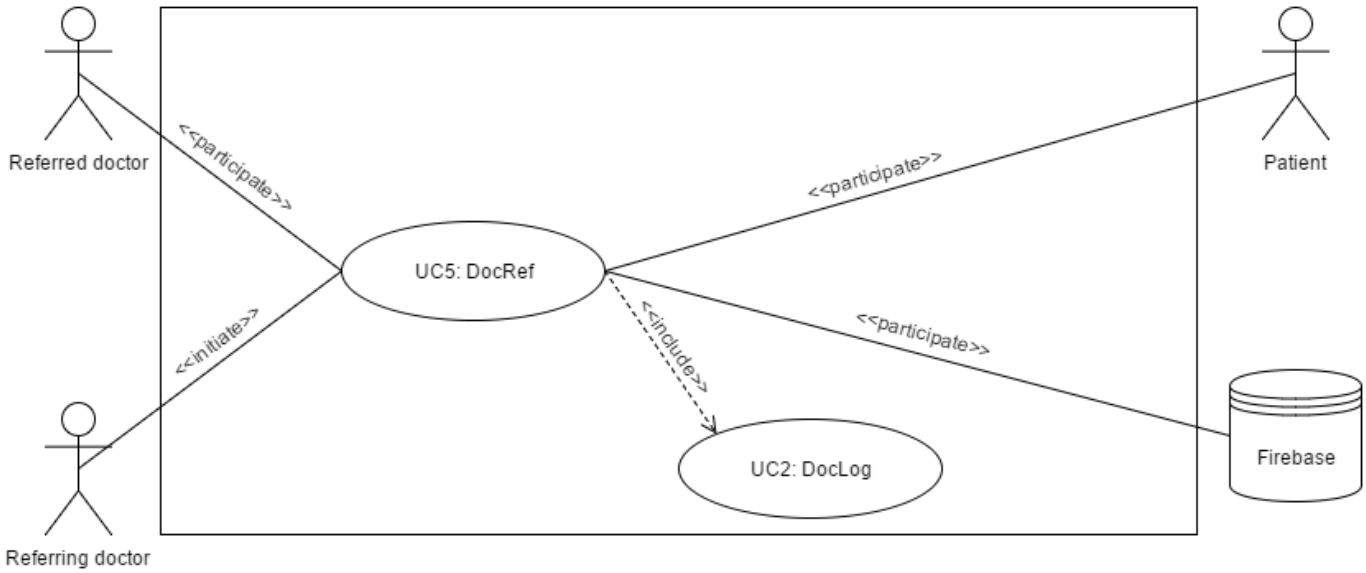


Figure 3: Use case diagram (Doctor add information)

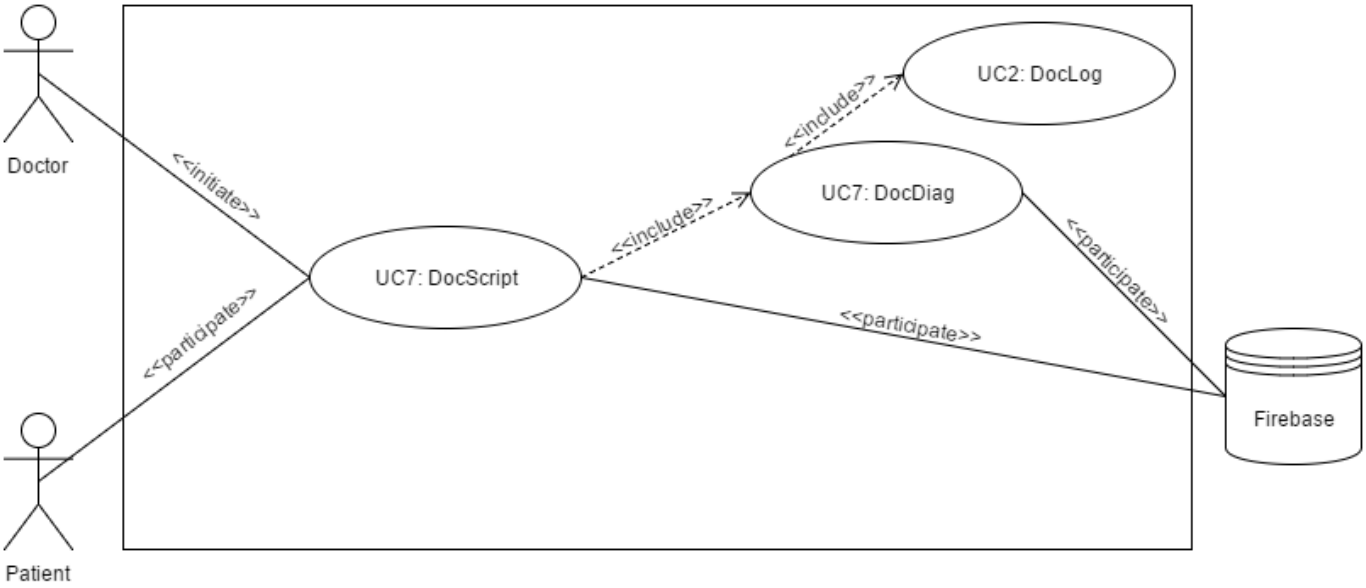


Figure 4: Use case diagram (View patient information)

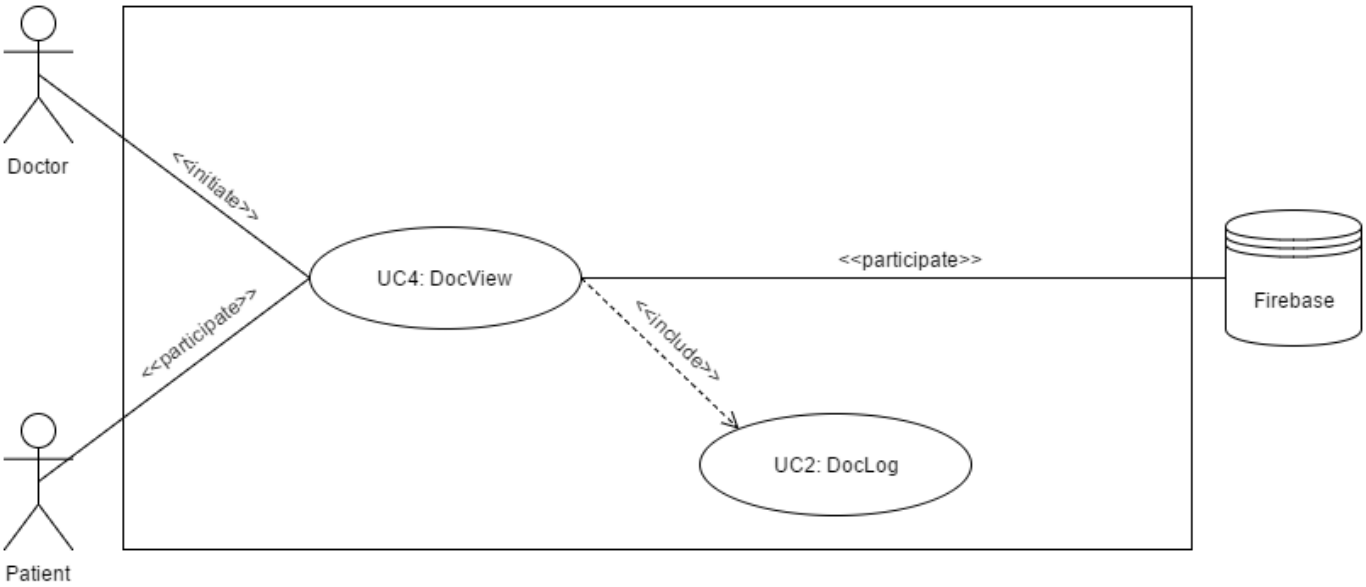
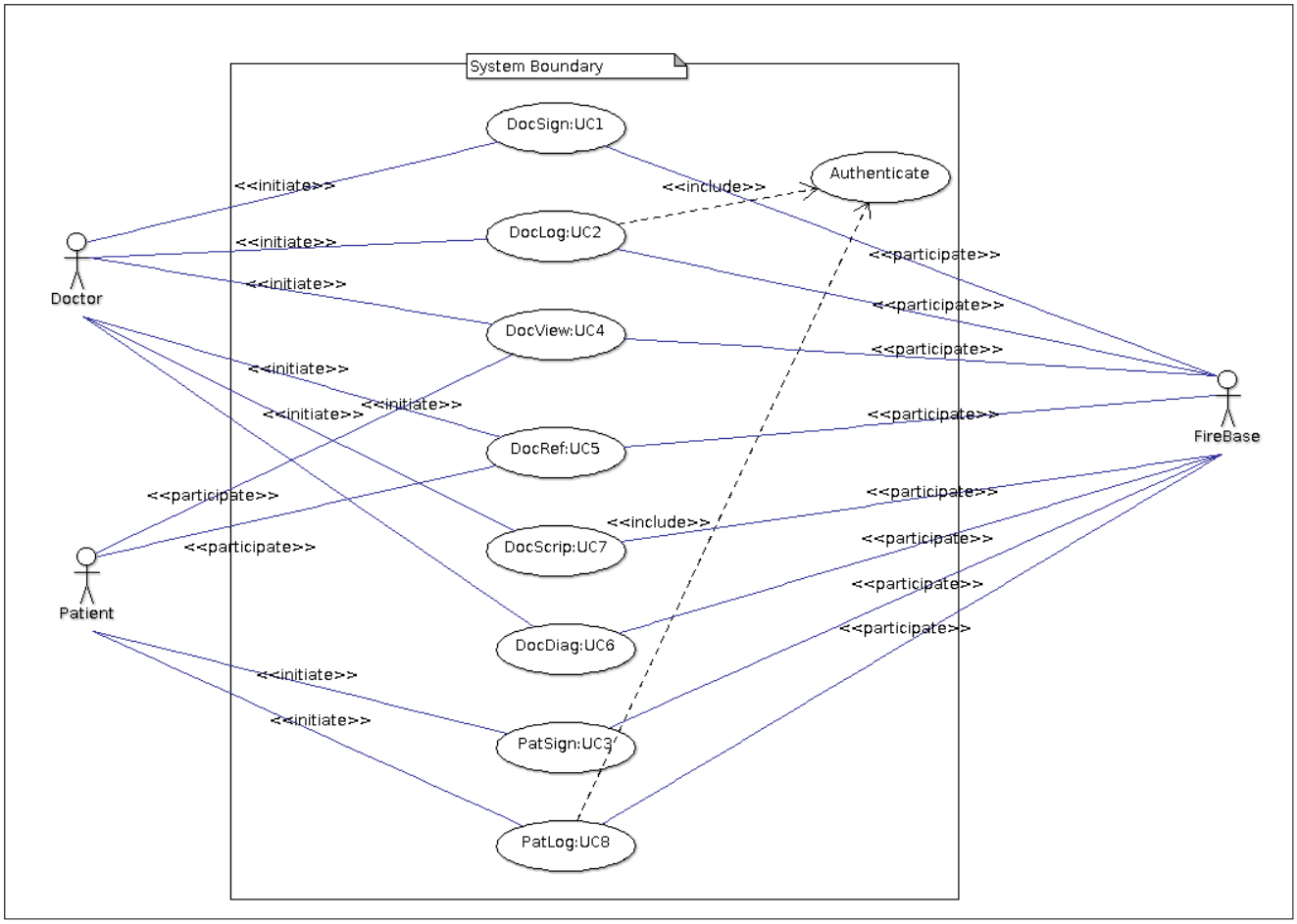


Figure 5: Use case diagram Overview

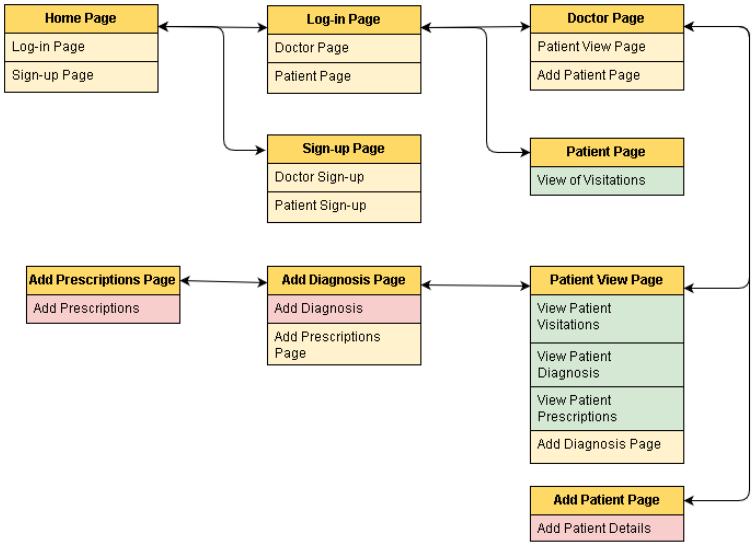


3.4.3 *Object model*

Object modeling is an object-oriented analysis and design technique. The goal of object modeling is to create a model comprising:

- A set of related object type definitions representing concepts from the 'real world' (problem domain) that are needed in the system.
- Descriptions of how objects of those types interact with each other to form the functions of the system.

Object Model

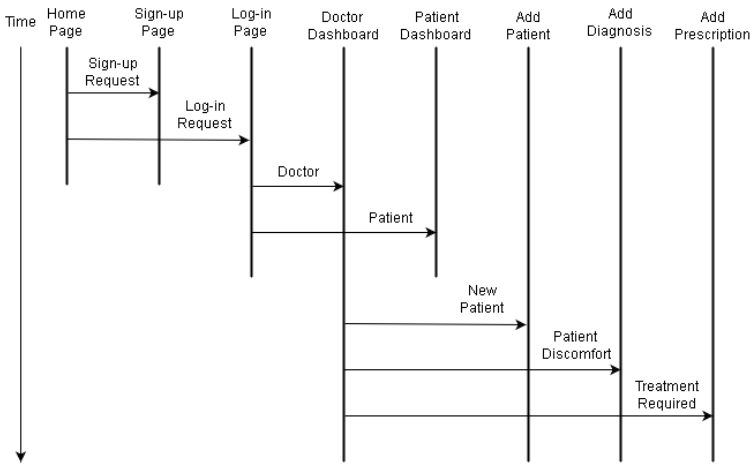


3.4.4 *Dynamic model*

A dynamic model represents the behavior of an object over time. It is used where the object's behavior is best described as a set of states that occur in a defined sequence. The components of the dynamic model are:

- **States**
- **State transitions**, are modeled as being instantaneous.
- **Events**, events trigger state transitions.
- **Actions**, occur on state transitions.
- **Activities**, is performed while an object is in a specific state.

Dynamic Model



3.4.5 *User interfaces*

Navigational paths and screen mock-ups:

3.5 *Test Driven Development*

3.5.1 *Acceptance Tests for Requirements*

Acceptance Test Case (**ATC1**) for REQ1:

- Ensure app is connected to Firebase and doctor is logged in, select "Patients" (pass: list of patients displayed).
- Ensure app is connected to Firebase and doctor is logged in, select a patient from list (pass: Basic patient information is displayed).
- Ensure app is connected to Firebase and doctor is logged in, viewing basic patient info, click "View Patient Information" (pass: Patient information is displayed).

ATC4 for REQ4:

- Add ID of patient in database and select "Request" (pass: Request is displayed to patient when s/he logs in).
- Patient selects "accept" (pass: Patient appears in aforementioned doctor's list of patients).

ATC5 for REQ5:

- Doctor enters valid user-name and password combination and selects "log-in" (pass: Doctor is taken to landing page).
- Doctor enters invalid user-name and password combination (pass: Access is denied).

ATC8 for REQ8:

- Click "sign-up" then "doctor" (pass: Taken to page requesting information).
- Enter all required details and click "sign-up" (pass: Information successfully added to Firebase for review).

ATC9 for REQ9:

- Click "sign-up" then "patient" (pass: Taken to page requesting information).
- Enter all required details and click "sign-up" (pass: Information successfully added to Firebase).

3.5.2 *Acceptance Tests for Uses Cases*

Test Case for UC1

Test-case Identifier: TC1 **Use Case Tested**: UC1 Doctor able to sign-up and add practice details. **Pass/Fail Criteria**: Doctor able to register and add practice details, otherwise test fails. **Input Data**: Practice number.	
Test Procedure:	**Expected Result**:
Set up: Client goes to home page. Step 1: Client clicks on sign-up button. Step 2: Client fills in practice details.	 Sign-up page opens. Application admin checks if practice number is valid, client details saved.

Test Case for UC2

Test-case Identifier: TC2 **Use Case Tested**: UC2 Doctor able to log-in and add a patient. **Pass/Fail Criteria**: Correct details have to be entered to proceed to adding a patient, otherwise test fails. **Input Data**: Log-in details and patient information.	
Test Procedure:	**Expected Result**:
Set up: Client goes to home page. Step 1: Client clicks on log-in button. Step 2: Client fills in log-in details. Step 3: Client clicks on add new patient button. Step 4: Client adds new patient details.	 Log-in page opens. Doctor page opens. Add new patient page opens. Application saves data.

Test Case for UC3

Test-case Identifier: TC3 **Use Case Tested**: UC3 Patient able to sign-up and add personal details. **Pass/Fail Criteria**: Patient able to register details, otherwise test fails. **Input Data**: Log-in details.	
Test Procedure:	**Expected Result**:
Set up: Client goes to home page. Step 1: Client clicks on sign-up button. Step 2: Client fills in details.	 Sign-up page opens. Application checks if data entered is correct, client details saved.

Test Case for UC4

Test-case Identifier: TC4 **Use Case Tested**: UC4 Doctor able to able to view patients information. **Pass/Fail Criteria**: After entering patient data, the details will appear in clients patient page, otherwise test fails. **Input Data**: Log-in details.	
Test Procedure:	**Expected Result**:
Set up: Client goes to doctor's page. Step 1: Client clicks on patients button. Step 2: Client clicks on relevant patient.	 Patients page opens. Patient information is displayed.

Test Case for UC5

Test-case Identifier: TC5 **Use Case Tested**: UC5 Doctor able to able to view referred patients information. **Pass/Fail Criteria**: Referred patient data will be displayed after patient validation. **Input Data**: Referred patient details.	
Test Procedure:	**Expected Result**:
Set up: Client goes to doctor's page. Step 1: Client clicks on add new patients button. Step 2: Enters referred patient's details. Step 3: Referred patient validates request.	 Add new patients page opens. Patient receives validation request. Doctor is able to see referred patient's information.

Test Case for UC6

Test-case Identifier: TC6 **Use Case Tested**: UC6 Doctor able to log-in, view patient list and add diagnosis to relevant patient. **Pass/Fail Criteria**: Doctor able to add diagnosis to patient, otherwise test fails. **Input Data**: Diagnosis for relevant patient.	
Test Procedure:	**Expected Result**:
Set up: Client goes to doctor's page.	
Step 1: Client clicks patients button.	Patients page opens.
Step 2: Client clicks on relevant patient.	Patient information is displayed.
Step 3: Client clicks on add diagnosis.	Diagnosis page is opened.
Step 4: Client adds diagnosis of patient.	Application saves diagnosis of patient and is displayed on patient page.

Test Case for UC7

Test-case Identifier: TC7 **Use Case Tested**: UC7 Doctor able to log-in, view patient list and add prescription to relevant diagnosis. **Pass/Fail Criteria**: Doctor able to add prescription to relevant diagnosis, otherwise test fails. **Input Data**: Prescription for relevant diagnosis.	
Test Procedure:	**Expected Result**:
Set up: Client goes to doctor's page.	
Step 1: Client clicks patients button.	Patients page opens.
Step 2: Client clicks on relevant patient.	Patient information is displayed.
Step 3: Client clicks on add diagnosis.	Diagnosis page is opened.
Step 4: Client adds diagnosis of patient or if diagnosis already present, click the prescription button.	The prescription block opens. The prescription block opens.
Step 5: Client adds prescription for the diagnosis.	Application saves prescription of patient and is displayed on patient page.