**Capstone Project (CPRO306)**

**Assessment 3: Final SRS Report**

**Project Title:** **Austin Hospital Management System**

| **Team Members** | | |
| --- | --- | --- |
| **No.** | **Name** | **SID** |
| **1** | **Bianca Carvalho** | **K221552** |
| **2** | **Lunshwa Shakya** | **K231451** |
| **3** | **Santiago Ortiz** | **K200370** |
| **4** | **Sulav Homagai** | **K220028** |
| **5** | **Willow Postlethwaite** | **K191297** |

[**1. Project Description 3**](#_eo1qyi4bdw0a)

[1.1. Business Case 3](#_k3dd8glhjrz2)

[1.2. Purpose and Objectives 3](#_ehmkrpb0ms7u)

[1.3. Detailed Scope 3](#_rquossqximak)

[1.4. Proposed System Design Specifications 4](#_1a73m6n1x88v)

[1.5. Methodology 4](#_e239vs5m4tbt)

[**2. System Analysis and Requirements 5**](#_goxj70c47jp6)

[2.1. Software and Hardware Requirements 5](#_kh0kahcejcev)

[2.2. Functional and Non-Functional Requirements 5](#_ijvitjo0ohs7)

[2.3. User Requirements (use cases, use case diagram) 6](#_uklm7g25iir7)

[**3. Database Design 22**](#_ccdqbgafldy6)

[3.1. Entity Relationship Diagram (ERD) 22](#_hv3twzv2uv63)

[3.2. Data Flow Diagram (DFD) 23](#_di5xrlkzk4z0)

[3.3. Security Implementation 30](#_5ya5em5jcc26)

[**4. User Interface Design 31**](#_ye73j8urmaxl)

[4.1. User Interface Storyboard 31](#_lfq45xwygpms)

[4.2. Input Data Forms 36](#_9v2a7mjvbdf5)

[4.3. Output Report Forms 37](#_lciz6q2d3850)

[**5. Test Plan & System Implementation Plan 39**](#_lvcprlsjm2hf)

[5.1 System Implementation Plan 39](#_d4jfongcacsa)

[5.2 Test Plan 39](#_rm28w0dguz64)

[**6. Feedback 40**](#_lik7jbujamwl)

[**7. References 40**](#_5kcii6fv7b9i)

# 

# **1. Project Description**

## **1.1. Business Case**

The system that Austin Hospital has currently is ineffective and lacks efficiency for the clinic, staff and patients alike. The current process for patients is to call the clinic or attend in person to book an appointment which staff and patients have expressed to the Hospital Board is a time-consuming and inconvenient method. The current system additionally does not allow for updating, notification and scheduling processes which can cause confusion, missed appointments, miscommunication and frustrations for all parties. The Austin Hospital Executive Board wants to implement a new Management System that will be developed to help the hospital to automate the internal daily activities. The final product is a solution to help both patients and the Hospital clinic suites with AES Encryption implemented for security.

## **1.2. Purpose and Objectives**

The project focus is for the Austin Hospital IT & Development Team to build an application that enables AES Encryption to protect data and prevent fraud on the system. The system would also have the intent to:

* A user-friendly system to improve Doctor-Patients communication and allow for real-time updates.
* Doctors can easily access Patient records and manage appointments.
* Then the clinic can easily create new users and update records for the patient's health records.
* Implement the AES Encryption for security and protections required.
* Provide notification and alert systems for patients to receive messages.

## **1.3. Detailed Scope**

* **User Registration**: AHMS requires patients to be able to register and login using a secure and unique UserID and password.
* **Doctor Filter:** AHMS will need the functionality for patients to filter through doctors to choose, the filters the system requires is doctors gender, speciality, profiles etc.
* **Appointment Management:** Doctors require the ability to accept, reject or modify an appointment request, and patients require a confirmation message or notification with real-time updates on modifications, cancellations etc.
* **Patient History:** Doctors will be able to search their patient records and history at the clinic using the patient UserID.
* **Encryption & Security:** The system requires AES Security Encryption to secure data using a public key cryptosystem.
* **Notification System:** Doctors and patients need the function to receive alerts and notifications on their respective devices to ensure they have automated reminders and communication.

## **1.4. Proposed System Design Specifications**

* **UI (User Interface):**  The application requires four panel interfaces for the different user requirements and functions;
* Doctor
* Patient
* Hospital/ Clinic/ Suite
* Admin
* **Register & Logins:** The application requires a secure registration and logins, with AES Encrypting the data stores.
* **Doctor Profiles:** Detailed profile information unique to each doctor on staff at the hospital, with appointment management functions.
* **Patient Profiles:** Detailed profile information unique to each patient and their health history at the hospital, with appointment management functions.
* **Admin Profiles:** Detailed profile information unique to each admin on staff at the hospital, with admin management functions i.e. can amend users, delete users etc.
* **Notification Systems:** The system requires notifications for reminders, updates and cancellation alerts in real-time.
* **Data Encryption & Security:** The system requires AES Security Encryption to secure all shared data.

## **1.5. Methodology**

* Requirements Analysis
* The Development and Project Team will gather detailed requirements from the stakeholders using the system
* Define the functional and non-functional requirements
* System Design
* The UI/ UX interfaces are designed
* Design and design system architecture, server components, database design etc.
* Implementation
* Develop a mobile application with the required features and functionalities
* Ensure AES-Encryption is implemented for secure data stores
* Testing
* The QA Team will conduct Unit Testing, Integration Testing, Security Testing and System Testing to ensure the system components meets its function expectations
* Deployment
* Ensure the app is available and deployed on the application stores i.e. GooglePlay, AppStore
* Provide user documentation and training for the required stakeholders
* Maintenance
* Ensure maintenance and monitoring is conducted on the system regularly
* Identify bugs or errors
* Provide support desk and procedures to updates.

# **2. System Analysis and Requirements**

## **2.1. Software and Hardware Requirements**

The hardware requirements for AHMS are:

* Processor: A multicore processor like intel core i5 for windows user and M1 chip for macOS user.
* RAM: A minimum of 4 GB RAM is recommended.
* External storage: An external hard drive can be used as a backup storage.
* Other peripherals: Keyboard, mouse, printer, and scanner.

The software requirements for AHMS are:

* Operating system: Windows 10 or later for windows users and macOS Big Sur 11.0 or later is recommended.
* XAMPP web server: It is a free open source web server platform.
* A compatible web browser: Web browsers like chrome, safari can be used.

After the deployment in the live server AHMS can be accessed by using the web browser.

## **2.2. Functional and Non-Functional Requirements**

The functional requirements for each user in AHMS are:

* Admin can create the user such as doctor, clinic staff, and cashiers.
* Users can login into the system.
* Users can update their profile details.
* Doctor can view the complaint, post the solution to the complaint.
* Doctors can register clinic details.
* Patients can post the AES key and the doctor can view them.
* Patients can search doctors based on their gender and speciality.
* Cashier can charge the patient and record the payment.

Some of the other functional requirements of the system are:

* Authentication: AHMS uses password-based authentication for the users.
* Reporting: Reports can be generated individually or in bulk.
* Admin tools: Admin tools such as creating new users, and generating reports.
* User interface: A user friendly interface that is easy to navigate.
* Backup and recovery: A copy of the recent databases is uploaded in Amazon s3 ensuring the smooth recovery.

The non-functional requirements of AHMS are:

* Responsiveness: AHMS is designed to be responsive in different device dimensions.
* Privacy: Protection of patient data from unauthorised access.
* Security: Protection from unauthorised access using passwords.
* Scalability: Capacity to be expandable to handle growth in users, transactions, and data volume for any future occurrences.
* Reliability: AHMS delivers consistent performance.
* Quality: AHMS provides a bug-free and an efficient system
* Documentation: Documentation of the project as user manuals, technical guides, and specifications.
* Response Time: Time taken to respond to user actions or requests.
* Maintainability: AHMS can be easily updated, and modified, for future enhancement.
* Performance: The speed, efficiency, and resource usage of AHMS is reflected on response time and the quality.
* Reusability: Other pages can be added simply cloning the design elements.

## **2.3. User Requirements (use cases, use case diagram)**

The use cases of the AHMS reflect the users or stakeholders that are involved in the hospital and their relative actions. The users such as admin, doctor, patient, clinic, and cashier are called actors.

**Admin**

* Create id & password

| CREATE ID & PASSWORD Use Case | |
| --- | --- |
| Name: | Create id & password |
| Actor/Role: | Admin |
| Description: | Describes the process to create users as doctor, clinic staff, and cashiers. |
| Successful completion: | 1. Admin logs into the system. 2. Goes to Users. 3. Create a New User button. 4. Add new User details. 5. Click create. |
| Alternative: |  |
| Precondition: | Admin should have the details of the user |
| Postcondition: | Users as doctor, clinic staff, or cashier are created. |
| Assumptions: | User does not have an account. |

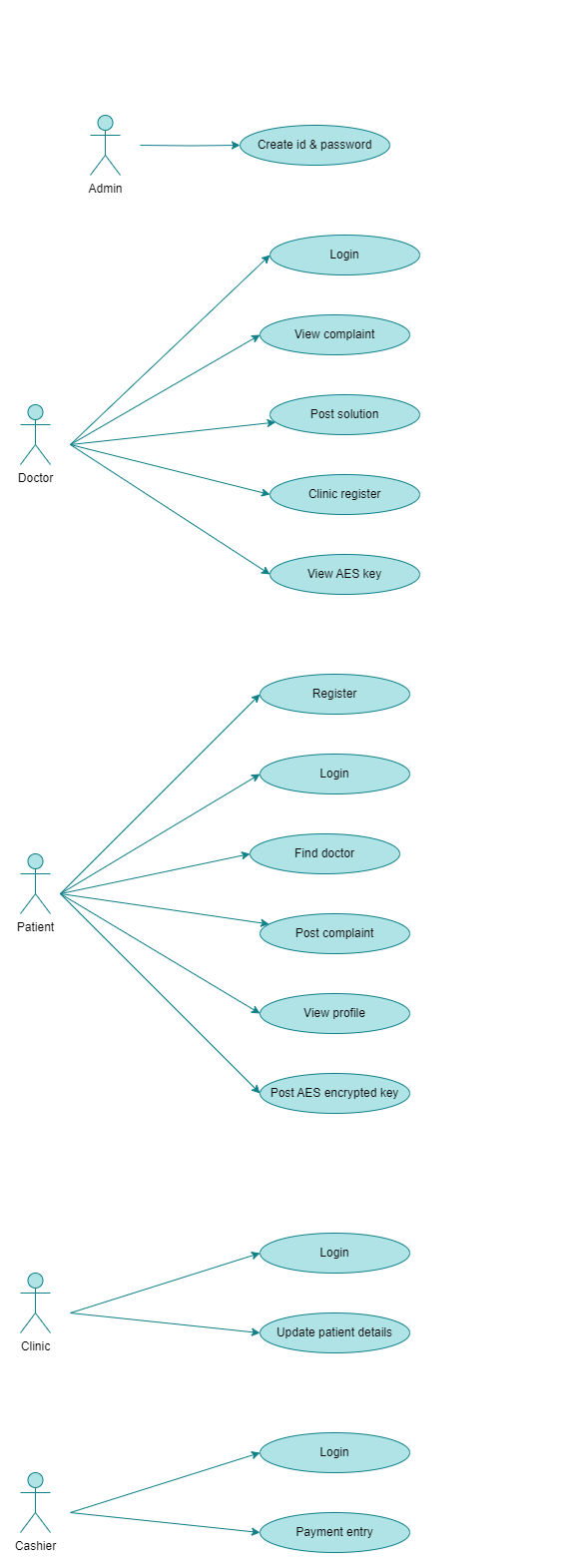


Fig. Admin creating id & password use case

* Login: Access the system

| LOGIN Use Case | |
| --- | --- |
| Name: | Login |
| Actor/Role: | Admin/ Doctor/ Patient / Clinic Staff/ Cashier |
| Description: | Describes the process to login as admin, doctor, clinic staff, and cashiers. |
| Successful completion: | 1. Admin logs into the system and successfully accesses it. 2. Doctor logs into the system and successfully accesses it. 3. Patient logs into the system and successfully accesses it. 4. Clinic staff logs into the system and successfully accesses it. 5. Cashier logs into the system and successfully accesses it. |
| Alternative: |  |
| Precondition: | Users should have username, and password |
| Postcondition: | User logs in into the dashboard. |
| Assumptions: | Users have an account. |

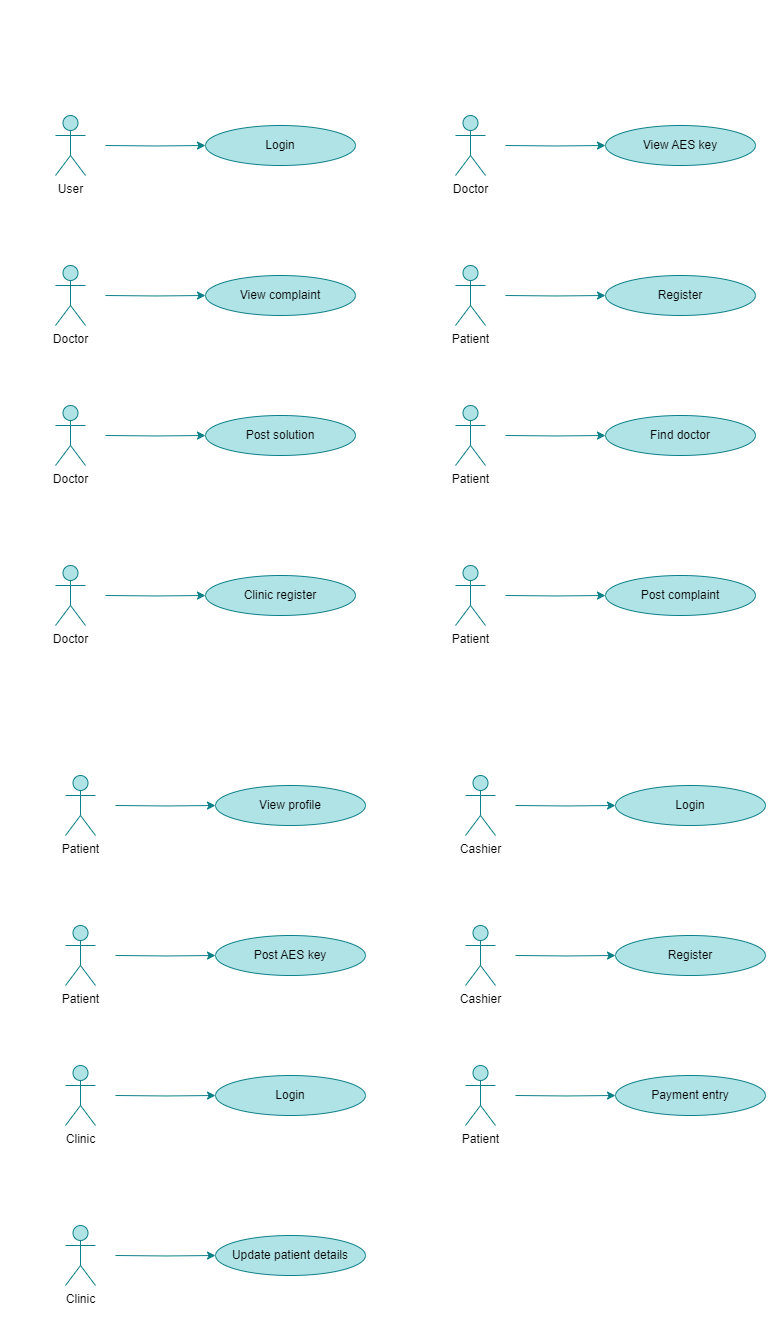


Fig. User login use case

**Doctor**

* View Complaint: View patient complaints.

| VIEW COMPLAINT Use Case | |
| --- | --- |
| Name: | View complaint |
| Actor/Role: | Doctor |
| Description: | Describes the process to view a complaint. |
| Successful completion: | 1. Doctor logs into the system and successfully accesses the "View Complaint" section. 2. Doctor views the list of patient complaints. 3. Doctor selects a specific complaint to view its details. |
| Alternative: |  |
| Precondition: | Doctor should be logged in. |
| Postcondition: | Doctor should be able to view the complaint. |
| Assumptions: | Doctors have complaints. |

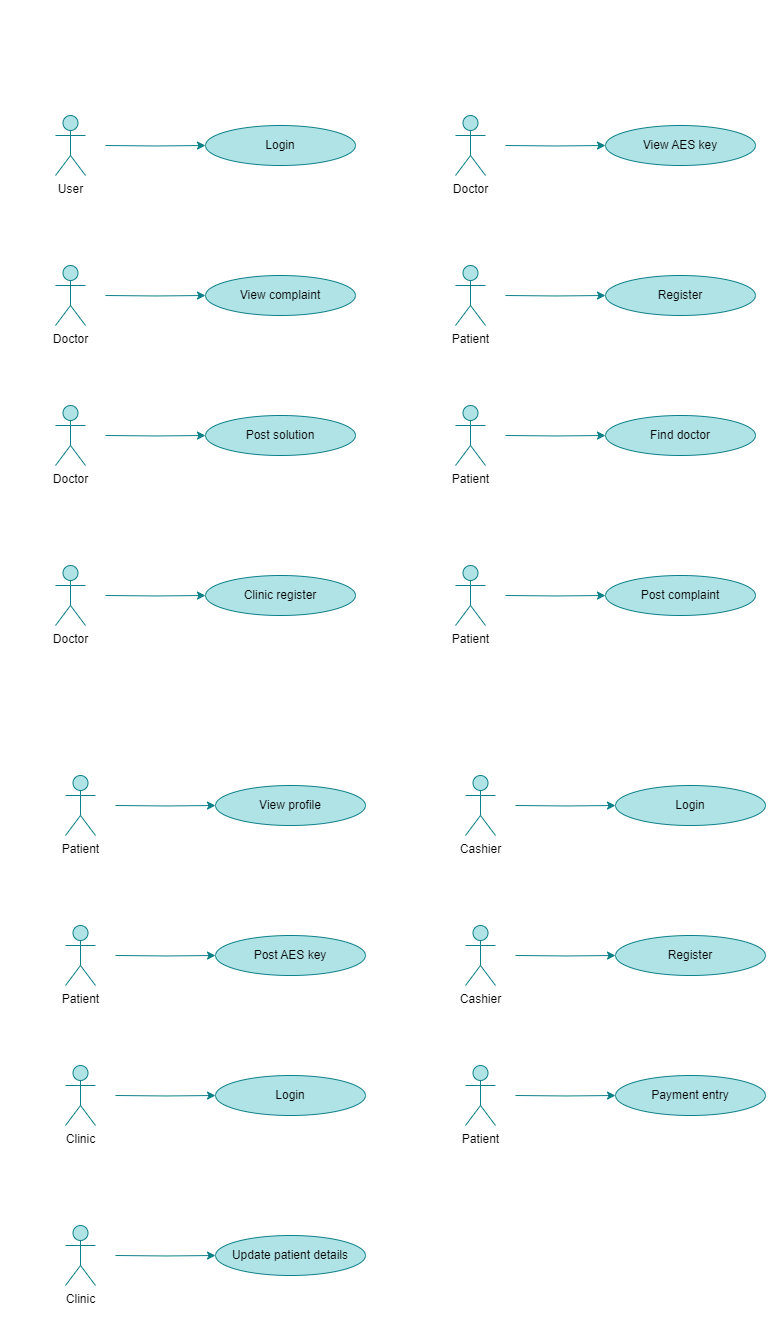


Fig. Doctor view complaint use case

* Post Solution: Provide solutions for patient complaints.

| POST SOLUTION Use Case | |
| --- | --- |
| Name: | Post solution |
| Actor/Role: | Doctor |
| Description: | Describes the process on how to post solution after |
| Successful completion: | 1. Doctor logs into the system and successfully views the specific patient complaint. 2. Doctor provides a solution to the complaint by posting a response. 3. The system confirms that the solution has been posted successfully. |
| Alternative: |  |
| Precondition: | Doctor should be able to view the complaint. |
| Postcondition: | Doctor should be able to post the solution for the complaint. |
| Assumptions: | Doctor has complaints from the patient. |

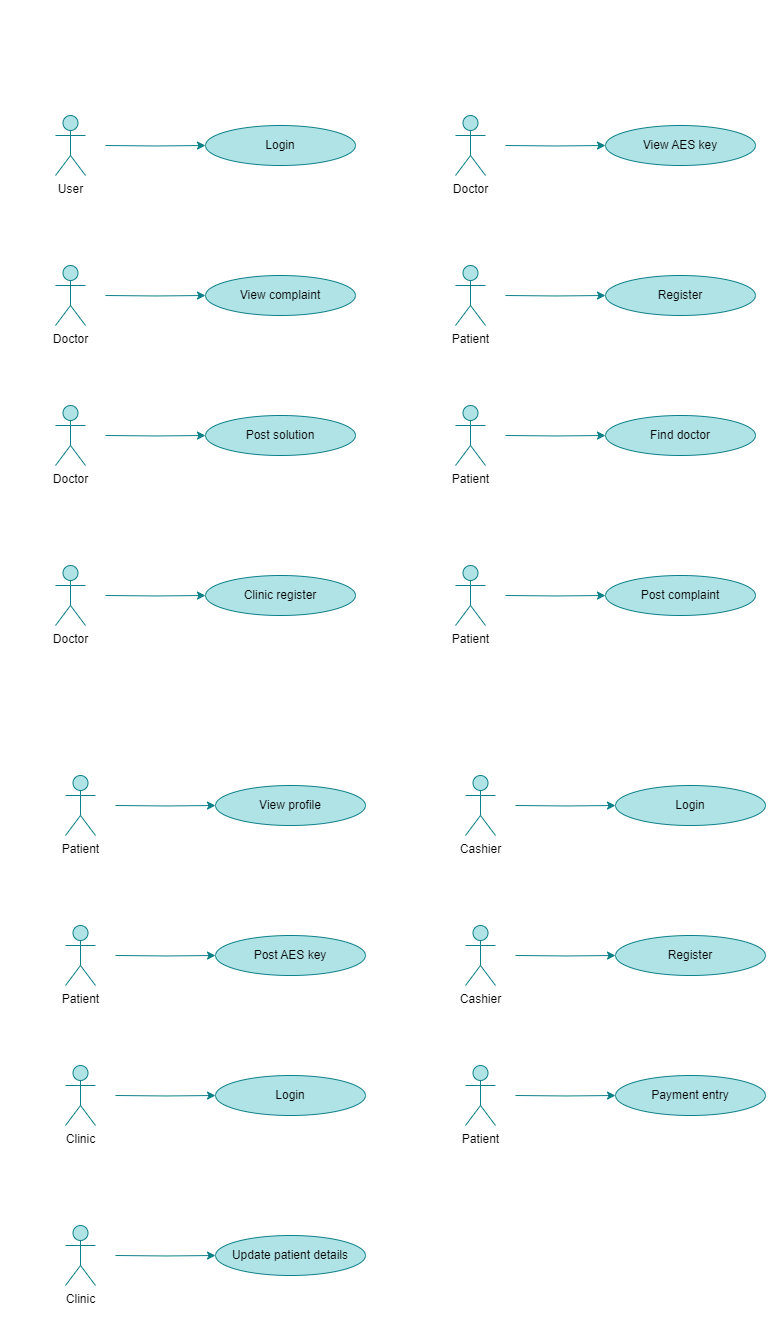


Fig. Doctor post solution use case

* Clinic Register: Register clinic details.

| CLINIC REGISTER Use Case | |
| --- | --- |
| Name: | Clinic register |
| Actor/Role: | Doctor |
| Description: | Describes the process to register clinic details. |
| Successful completion: | 1. Doctor logs into the system and accesses the "Clinic Register" section. 2. Doctor enters the clinic details (name, address, contact information) into the form. 3. Doctor submits the form, and the clinic details are successfully registered in the system. |
| Alternative: |  |
| Precondition: | Doctor should be working in the clinic. |
| Postcondition: | Doctor is able to add or update the clinic details. |
| Assumptions: | Some changes were made in the clinic. |

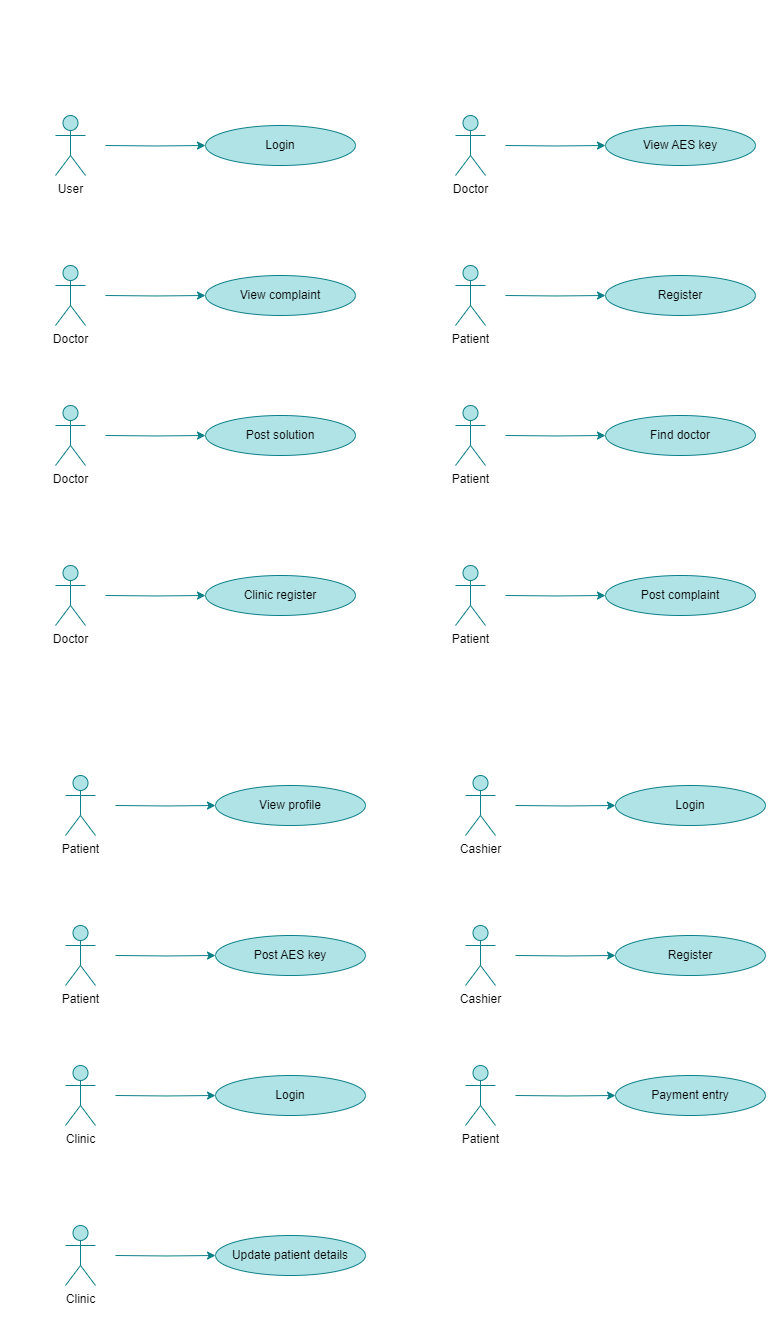


Fig. Doctor clinic register use case

* View AES Key: Access encrypted keys shared by patients.

| VIEW AES KEY Use Case | |
| --- | --- |
| Name: | View AES Key |
| Actor/Role: | Doctor |
| Description: | Describes the process to view AES keys generated by a patient. |
| Successful completion: | 1. Doctor logs into the system and successfully accesses the "View AES Key" section. 2. Doctor enters the required password for decryption. 3. The system decrypts the AES key and displays it to the doctor. |
| Alternative: |  |
| Precondition: | Patient should generate the AES key. |
| Postcondition: | Doctor is able to view the AES key of the patient. |
| Assumptions: | Patient generated the AES key. |

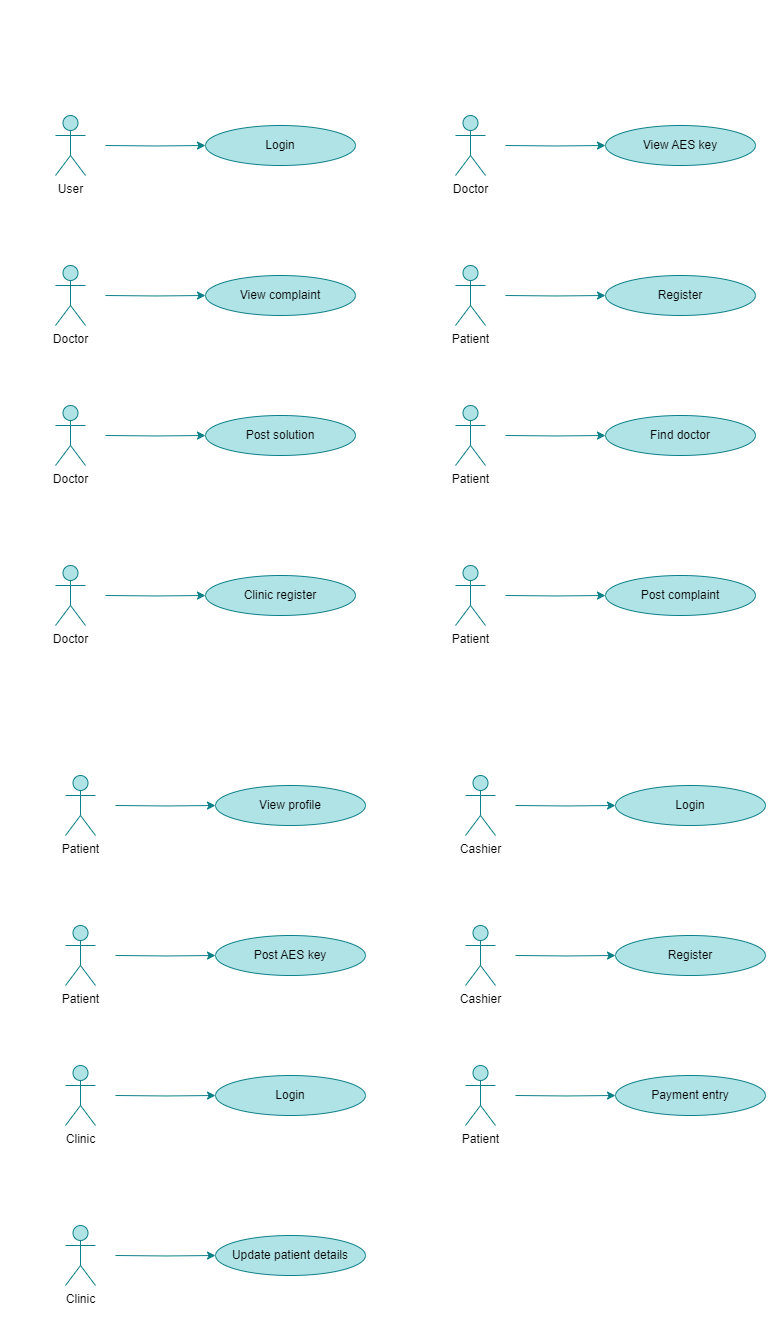


Fig. Doctor view AES key use case

**Patient**

* Register: Sign up for the system.

| REGISTER Use Case | |
| --- | --- |
| Name: | Register |
| Actor/Role: | Patient |
| Description: | Describes the process to create a patient account. |
| Successful completion: | 1. Patient accesses the registration page. 2. Patient enters all the required information (such as name, contact details, and any other necessary information) into the registration form. 3. Patient submits the registration form, and the system successfully creates an account. |
| Alternative: |  |
| Precondition: | Patient should have the details in hand. |
| Postcondition: | Patient created an account with AHMS. |
| Assumptions: | Patient requires medical attention. |

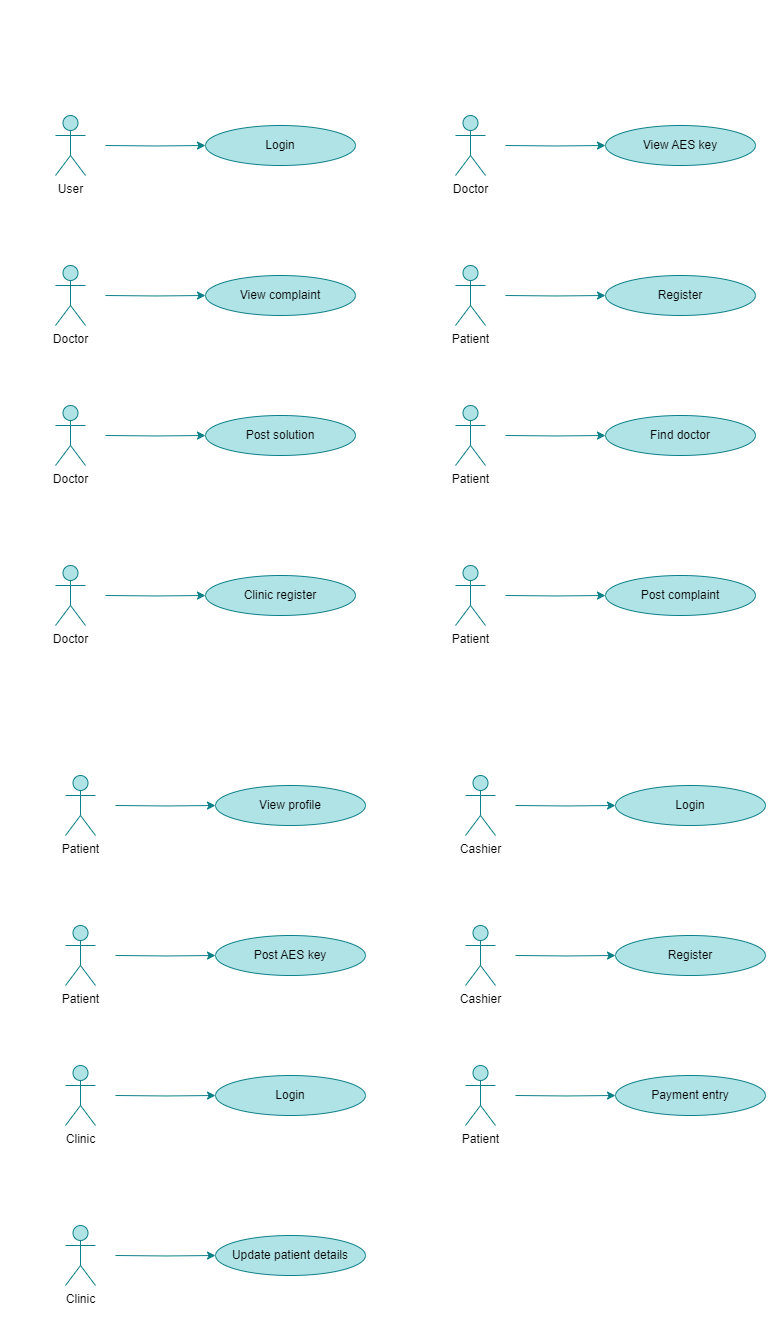


Fig. Patient register use case

* Login: Access the system.

| LOGIN Use Case | |
| --- | --- |
| Name: | Login |
| Actor/Role: | Patient |
| Description: | Describes the process to login into the patient portal. |
| Successful completion: | 1. Patient accesses the login page. 2. Patient enters their username and password. 3. Patient successfully logs into the patient portal. |
| Alternative: |  |
| Precondition: | Patient should have an existing account. |
| Postcondition: | Patient is able to login using the correct username and password. |
| Assumptions: | Patient wants to book a treatment. |

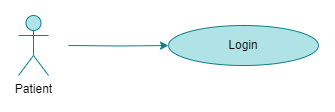


Fig. Patient login use case

* Find Doctor: Search for doctors based on filters (e.g., gender, specialty).

| FIND DOCTOR Use Case | |
| --- | --- |
| Name: | Find Doctor |
| Actor/Role: | Doctor |
| Description: | Describes the process to find the doctor for the specific medical condition. |
| Successful completion: | 1. Patient accesses the "Find Doctor" feature. 2. Patient selects the appropriate filters (e.g., gender, specialty). 3. Patient successfully finds a doctor that matches the selected criteria. |
| Alternative: |  |
| Precondition: | Patient should know the medical condition. |
| Postcondition: | Patients should be able to find the doctor using the filter as specialisation. |
| Assumptions: | Patient knows what they want to get treated for. |

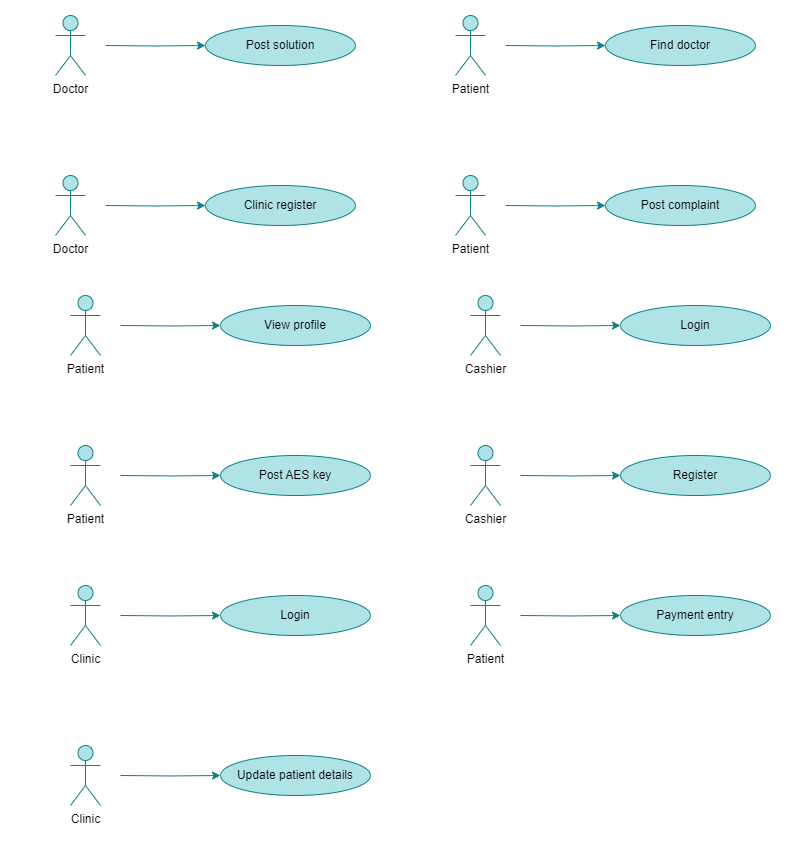


Fig. Patient find doctor use case

* Post Complaint: Submit complaints or health issues.

| POST COMPLAINT Use Case | |
| --- | --- |
| Name: | Post complaint |
| Actor/Role: | Patient |
| Description: | Describes the process on how to post a complaint. |
| Successful completion: | 1. Patient accesses the "Post Complaint" section. 2. Patient enters the details of their complaint. 3. Patient submits the complaint, and the system successfully records it. |
| Alternative: |  |
| Precondition: | Patient should have a complaint. |
| Postcondition: | Patient should be able to post the complaint. |
| Assumptions: | Patient is not happy with the treatment. |

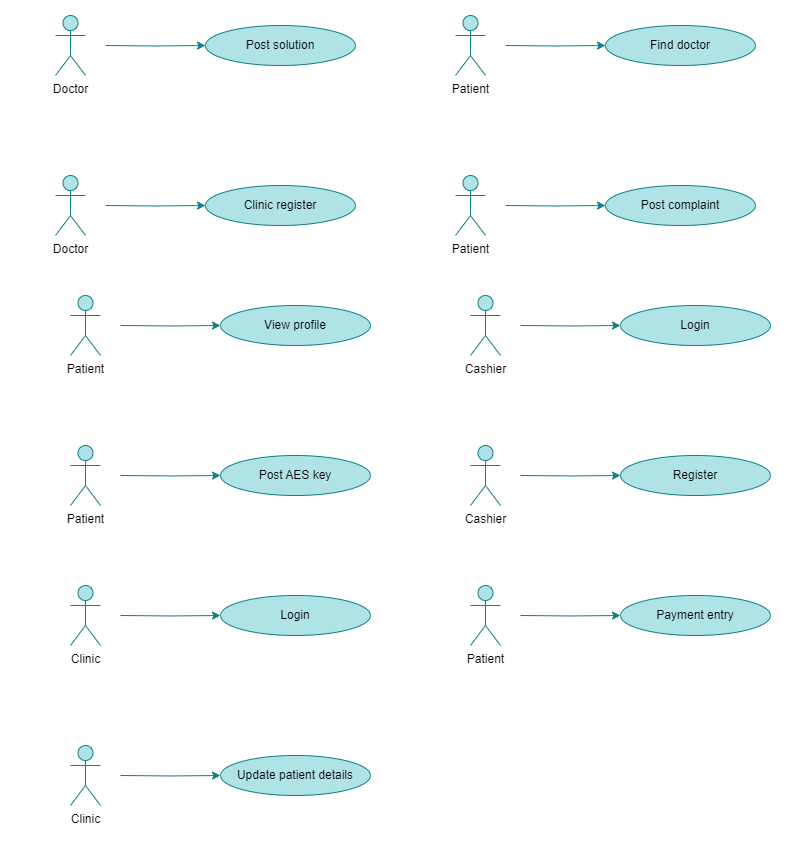


Fig. Patient post complaint use case

* View Profile: Update personal details and view medical history.

| VIEW PROFILE Use Case | |
| --- | --- |
| Name: | View profile |
| Actor/Role: | Patient |
| Description: | Describes the process on how to view patient profile |
| Successful completion: | 1. Patient logs into the system and accesses the "View Profile" section. 2. Patients successfully view their personal details and medical history. |
| Alternative: |  |
| Precondition: | Patient should have an account and be logged in. |
| Postcondition: | Patient are able to view their details |
| Assumptions: | Patient wants to check their details. |

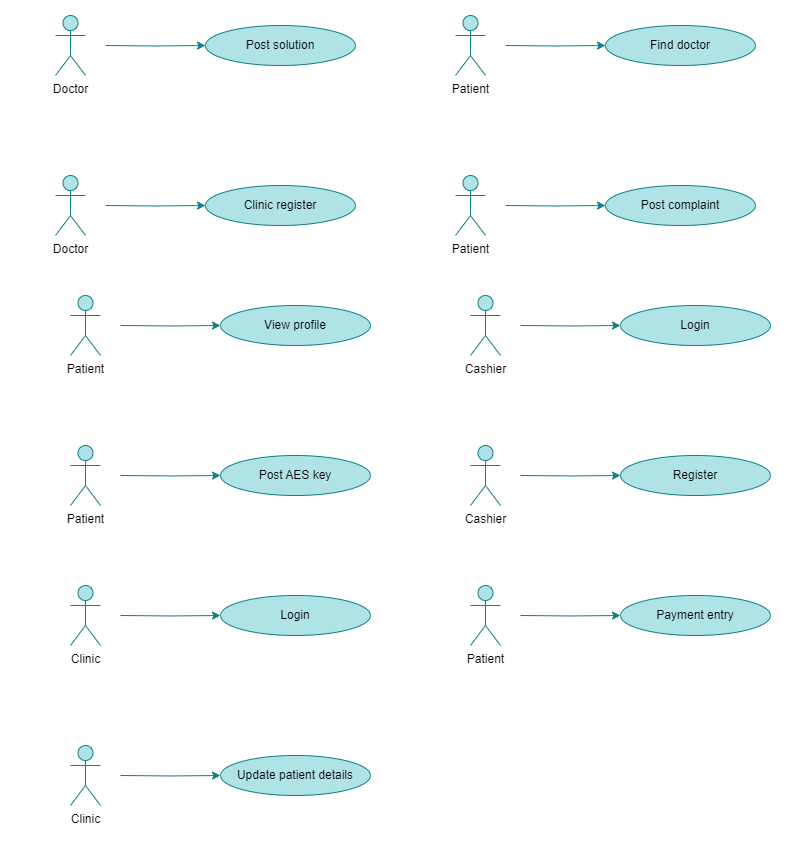


Fig. Patient view profile use case

* Post AES Key: Share encrypted keys with doctors for secure communication.

| POST AES KEY Use Case | |
| --- | --- |
| Name: | POST AES Key |
| Actor/Role: | Patient |
| Description: | Describes the process on how to generate and post AES. |
| Successful completion: | 1. Patient generates an AES key. 2. Patient successfully posts the AES key for secure communication with doctors. |
| Alternative: |  |
| Precondition: | Patients should be logged in and have an issue to post as an AES key. |
| Postcondition: | Patients should be able to post an AES key. |
| Assumptions: |  |

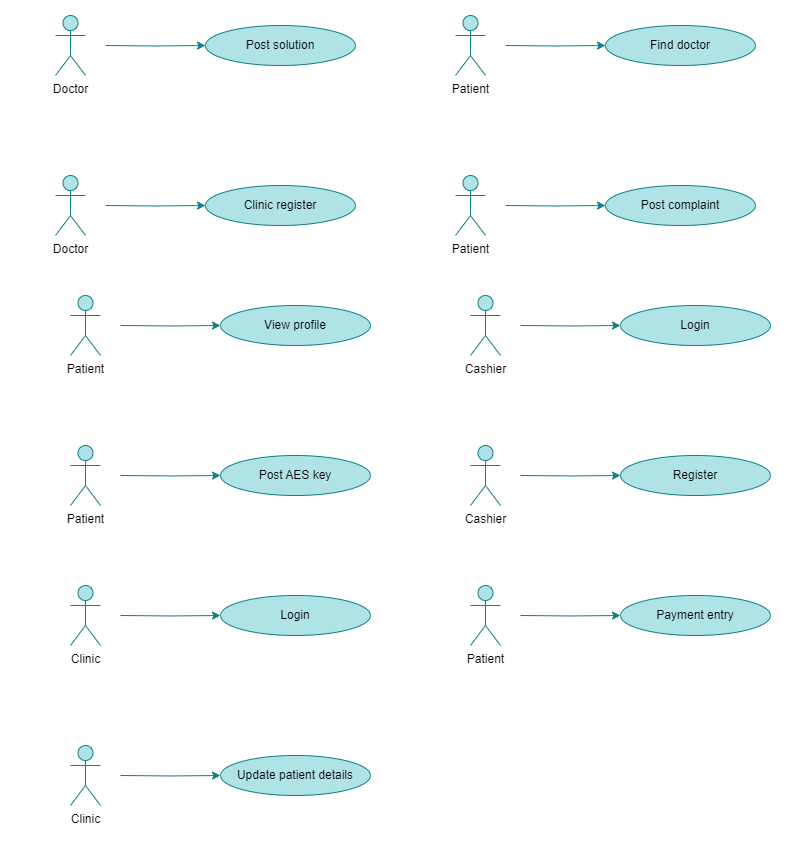


Fig. Patient post AES key use case

**Clinic**

* Login: Access the system.

| LOGIN Use Case | |
| --- | --- |
| Name: | Login |
| Actor/Role: | Clinic |
| Description: | Describes the process to login by clinic. |
| Successful completion: | 1. Clinic staff access the login page. 2. Clinic staff enter their username and password. 3. Clinic staff successfully log into the system. |
| Alternative: |  |
| Precondition: | Clinic staff should have an account. |
| Postcondition: | Clinic staff should be able to log in. |
| Assumptions: |  |

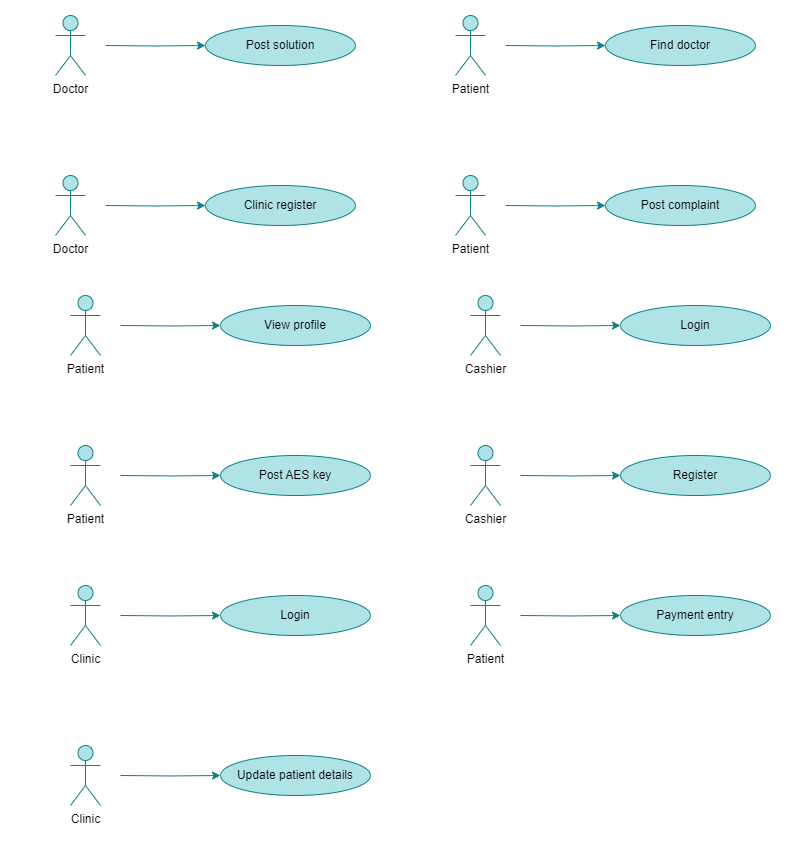


Fig. Clinic login use case

* Update Patient Details: Modify patient information as needed.

| UPDATE PATIENT DETAILS Use Case | |
| --- | --- |
| Name: | Update patient details |
| Actor/Role: | Clinic |
| Description: | Describes the process to update the patient details. |
| Successful completion: | 1. Clinic staff logs into the system and accesses the "Update Patient Details" section. 2. Clinic staff modify the patient information as needed. 3. The system successfully updates the patient details. |
| Alternative: |  |
| Precondition: | Clinic should have updated patient details. |
| Postcondition: | Clinic is able to update patient details |
| Assumptions: |  |

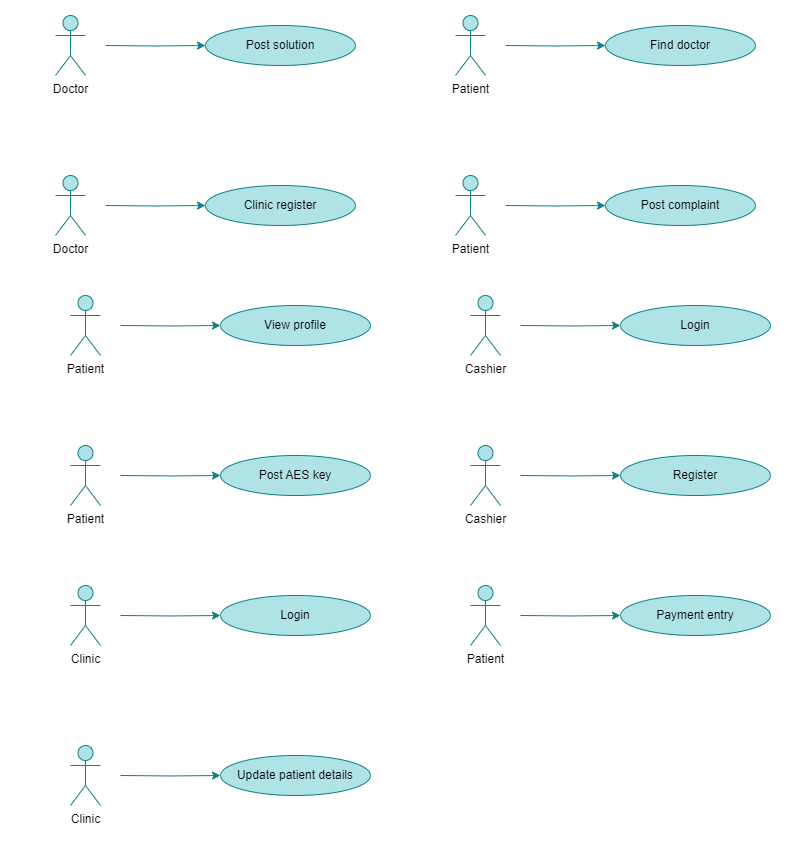
****

Fig. Clinic update patient details use case

**Cashier**

* Login: Access the system.

| LOGIN Use Case | |
| --- | --- |
| Name: | Login |
| Actor/Role: | Cashier |
| Description: | Describes the process to login by cashier. |
| Successful completion: | 1. Cashier accesses the login page. 2. Cashier enters their username and password. 3. Cashier successfully logs into the system. |
| Alternative: |  |
| Precondition: | Cashier should have an account. |
| Postcondition: | Cashier should be able to log in. |
| Assumptions: | Admin created the cashier account. |

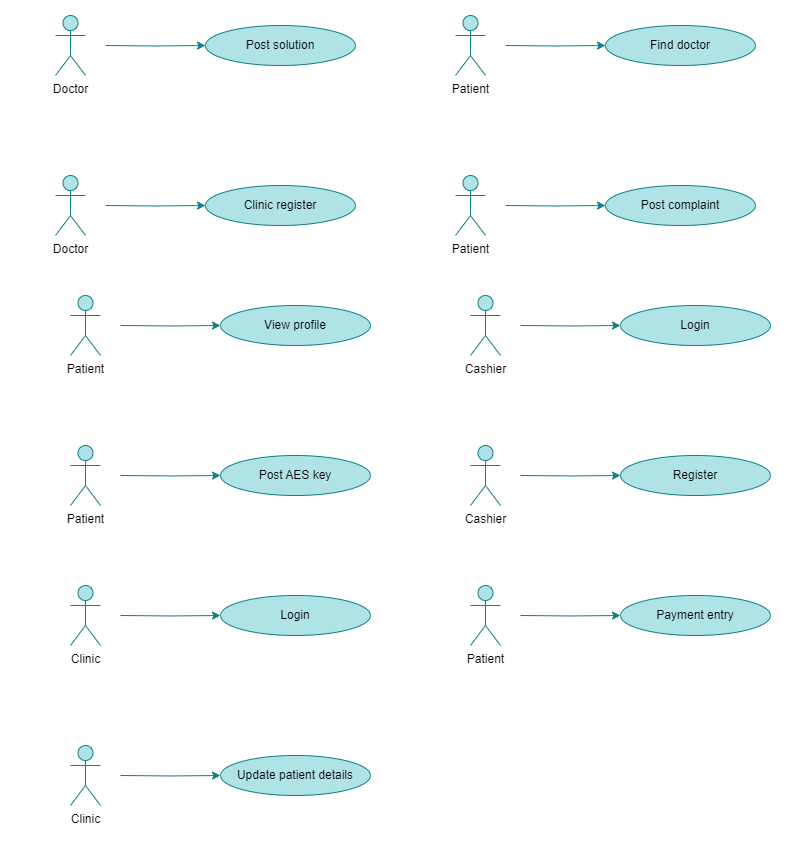


Fig. Cashier login use case

* Payment Entry: Record payment information.

| PAYMENT ENTRY Use Case | |
| --- | --- |
| Name: | Payment entry |
| Actor/Role: | Cashier |
| Description: | Describes the process on how to process payment entry. |
| Successful completion: | 1. Cashier logs into the system and accesses the "Payment Entry" section. 2. Cashier enters the payment information into the system. 3. The system successfully records the payment entry. |
| Alternative: |  |
| Precondition: | Cashiers have a payment entry to be made. |
| Postcondition: | Cashier is able to make a payment entry into the system. |
| Assumptions: |  |

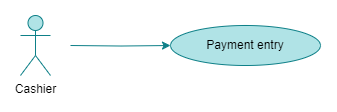


Fig. Cashier payment entry use case

2.3.1 Use Cases

**Admin**

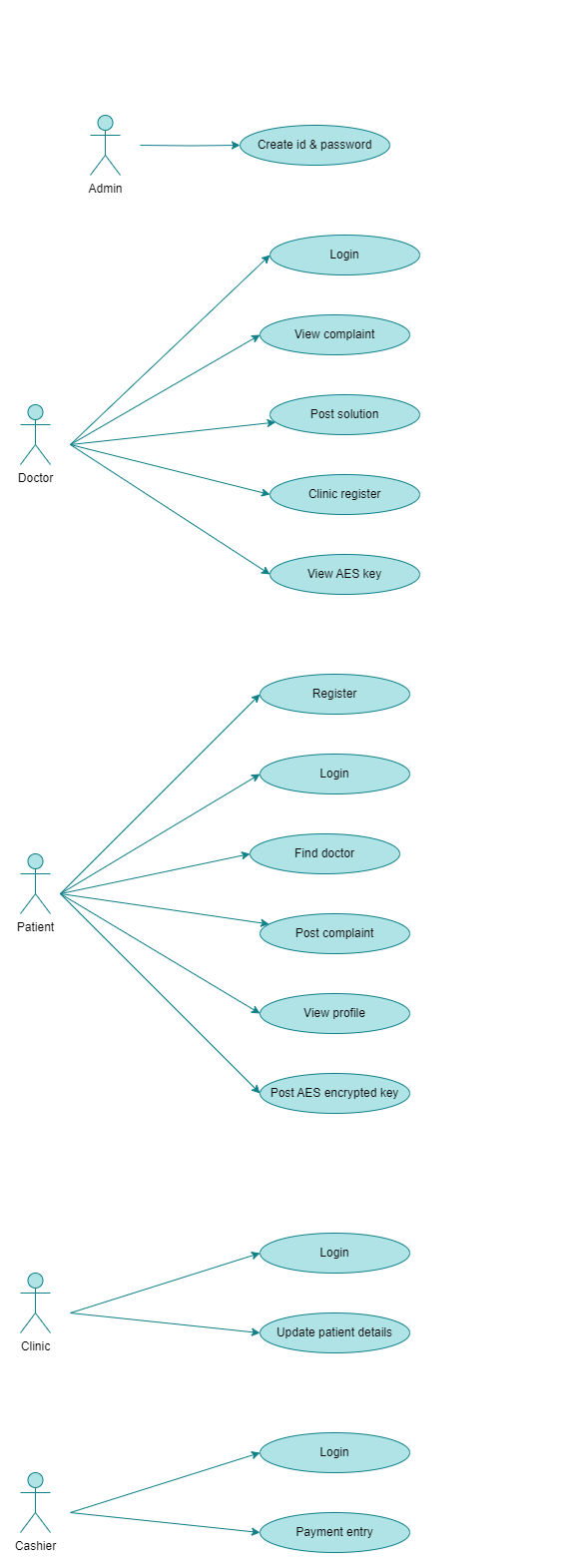


Fig. Admin use cases

**Doctor**

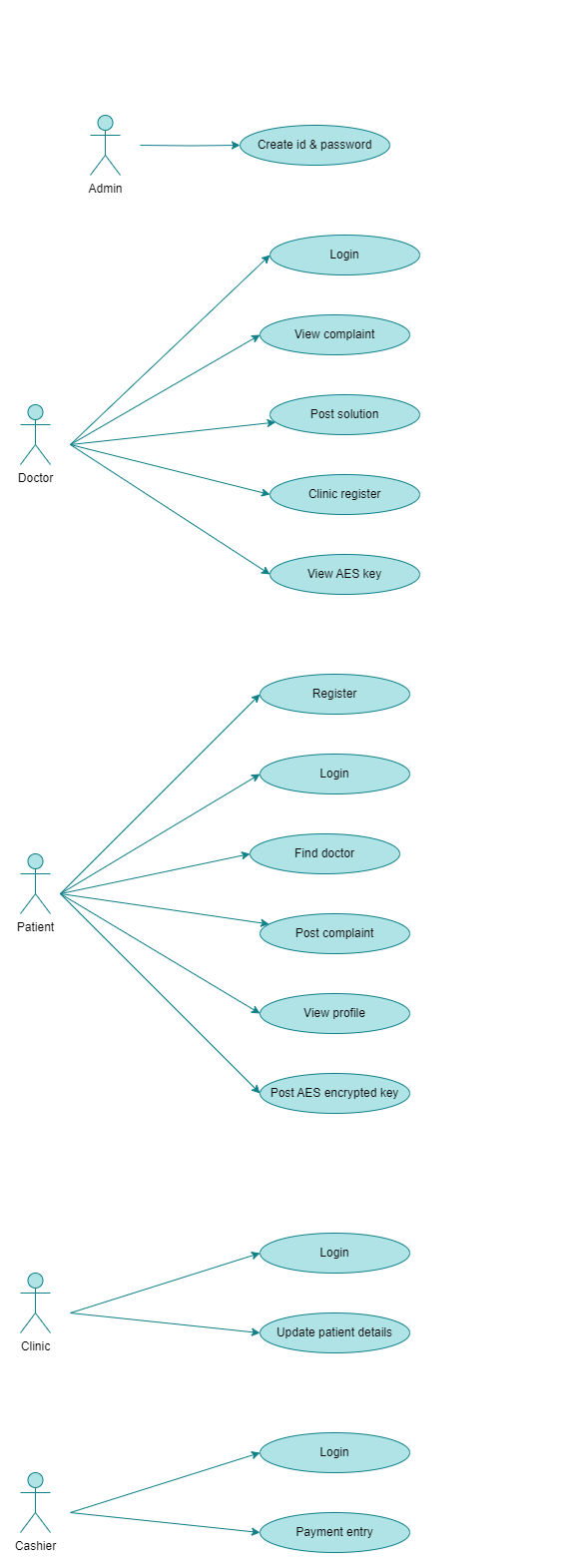


Fig. Doctor use cases

**Patient**

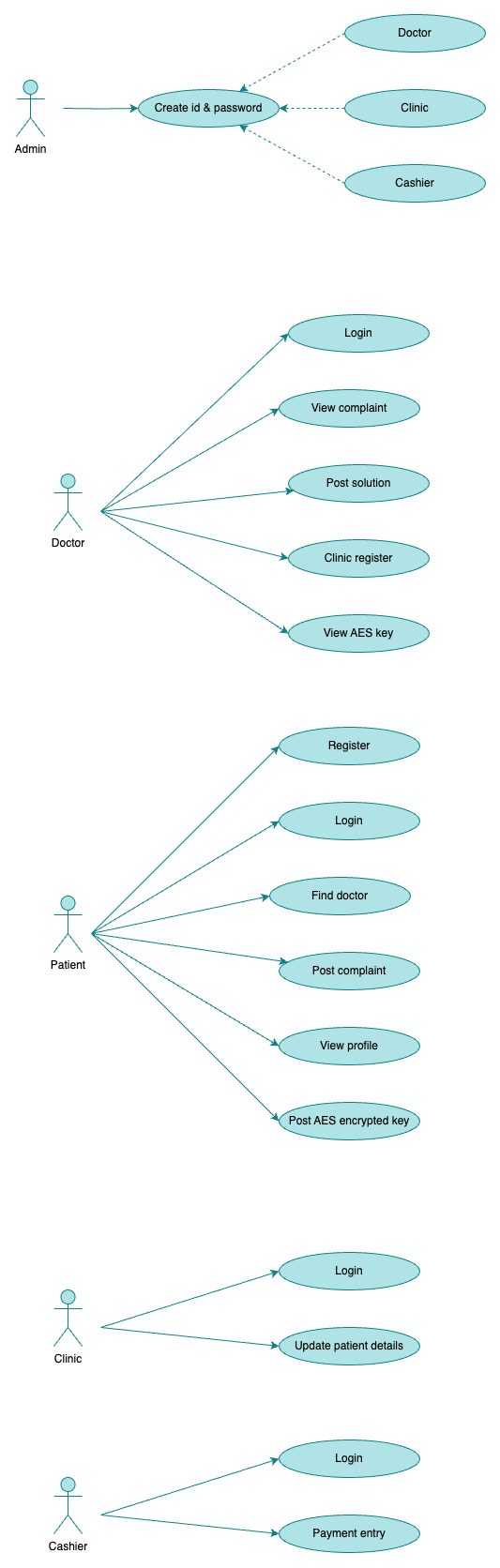


Fig. Patient use cases

**Clinic**

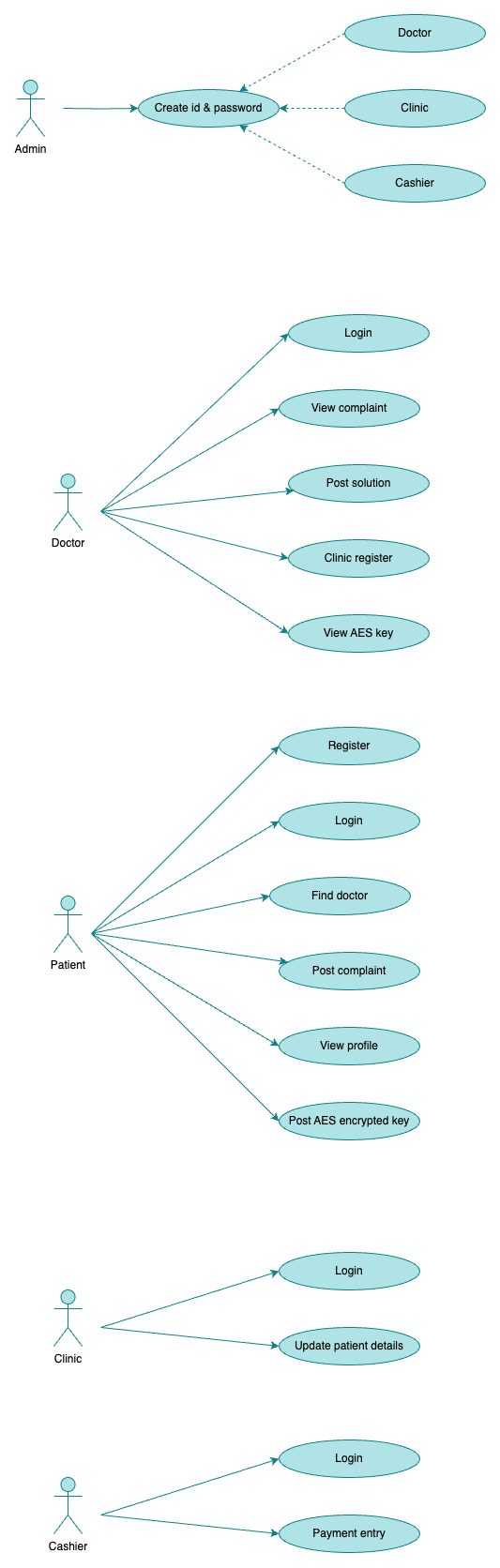
****

Fig. Clinic use cases

**Cashier**

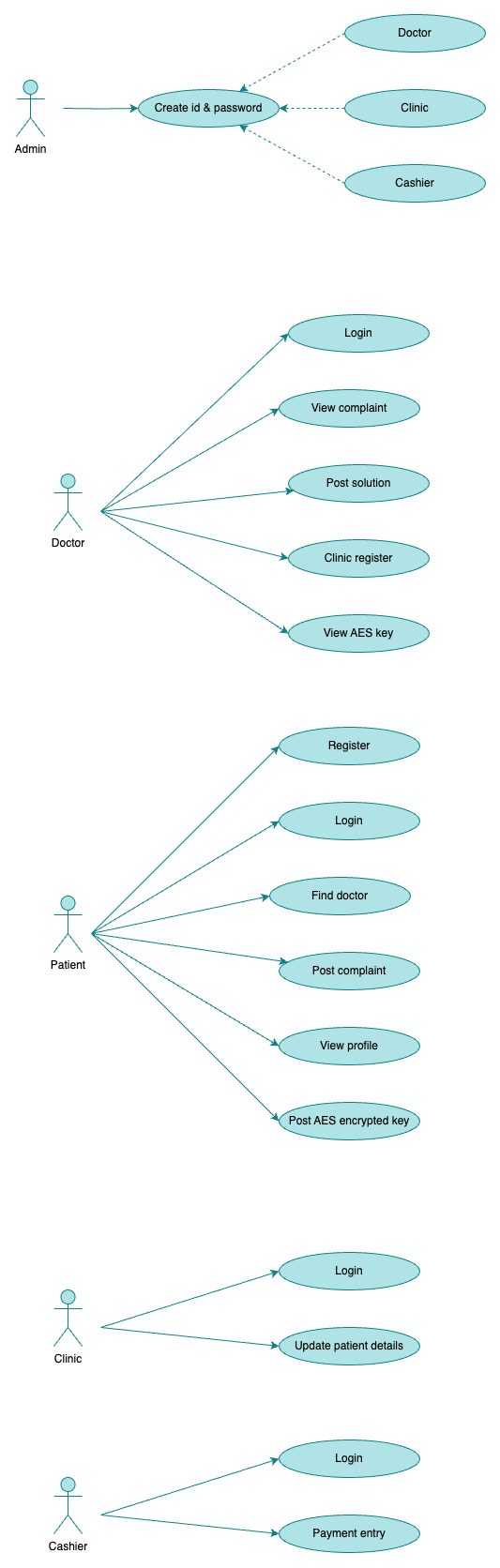


Fig. Cashier use cases

# **3. Database Design**

## **3.1. Entity Relationship Diagram (ERD)**

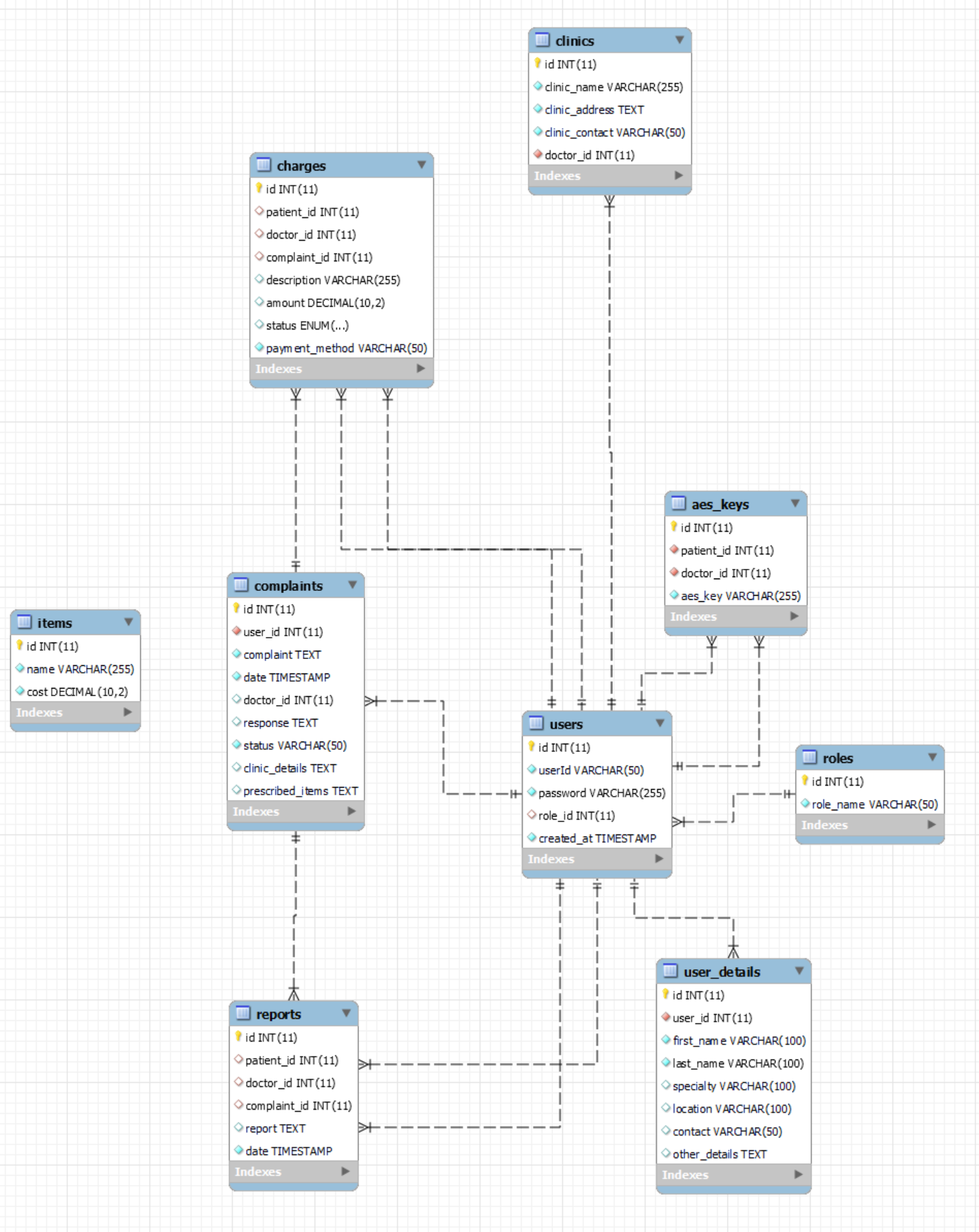


Fig. ER Diagram

## **3.2. Data Flow Diagram (DFD)**

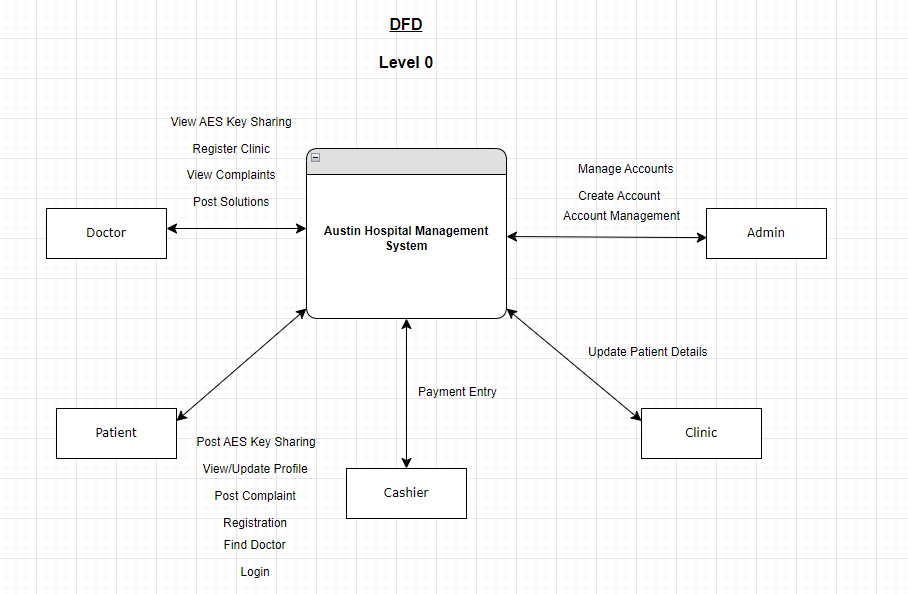


Fig. Data Flow Diagram Level 0

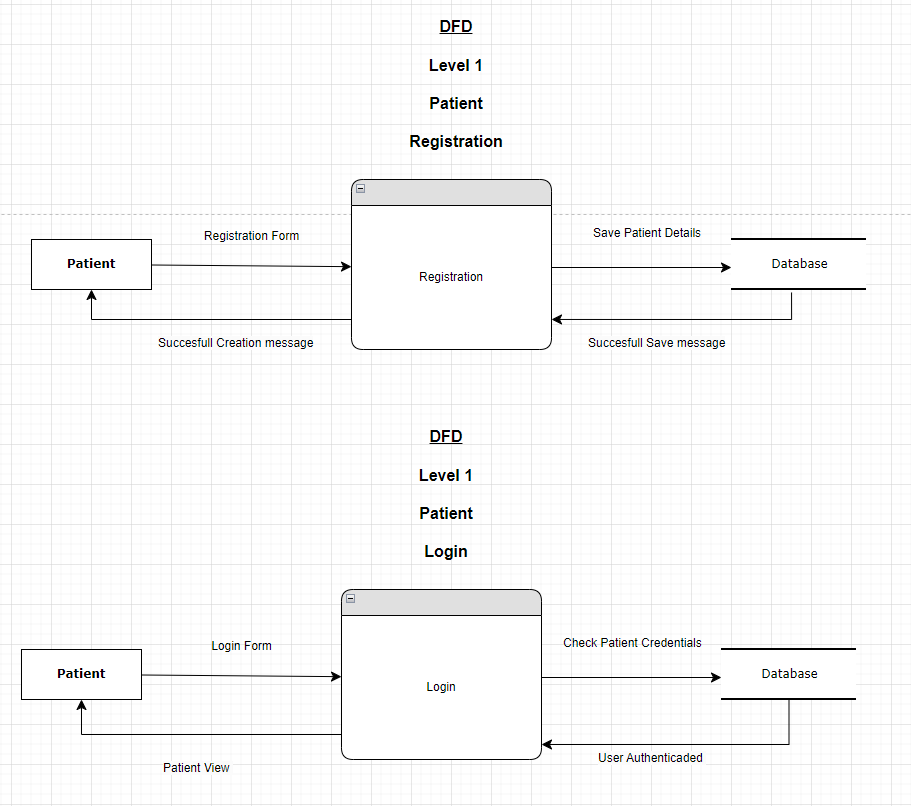


Fig. Data Flow Diagram Level 1 - Patient Registration and Login

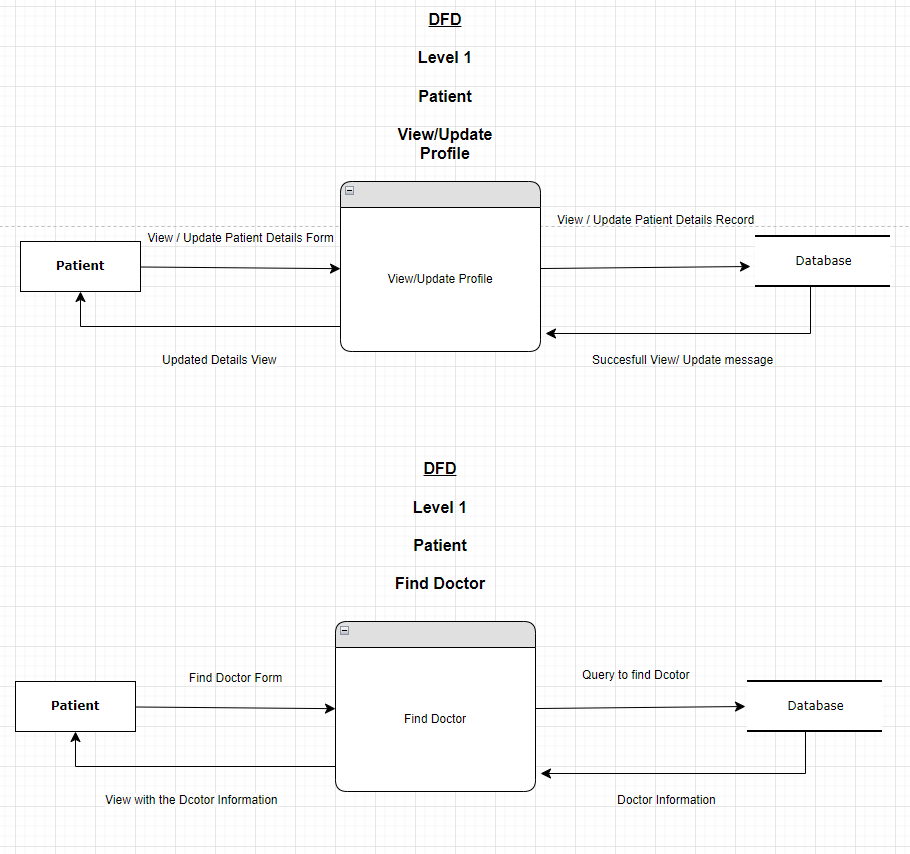


Fig. Data Flow Diagram Level 1 - Patient View/Update profile and Find doctor

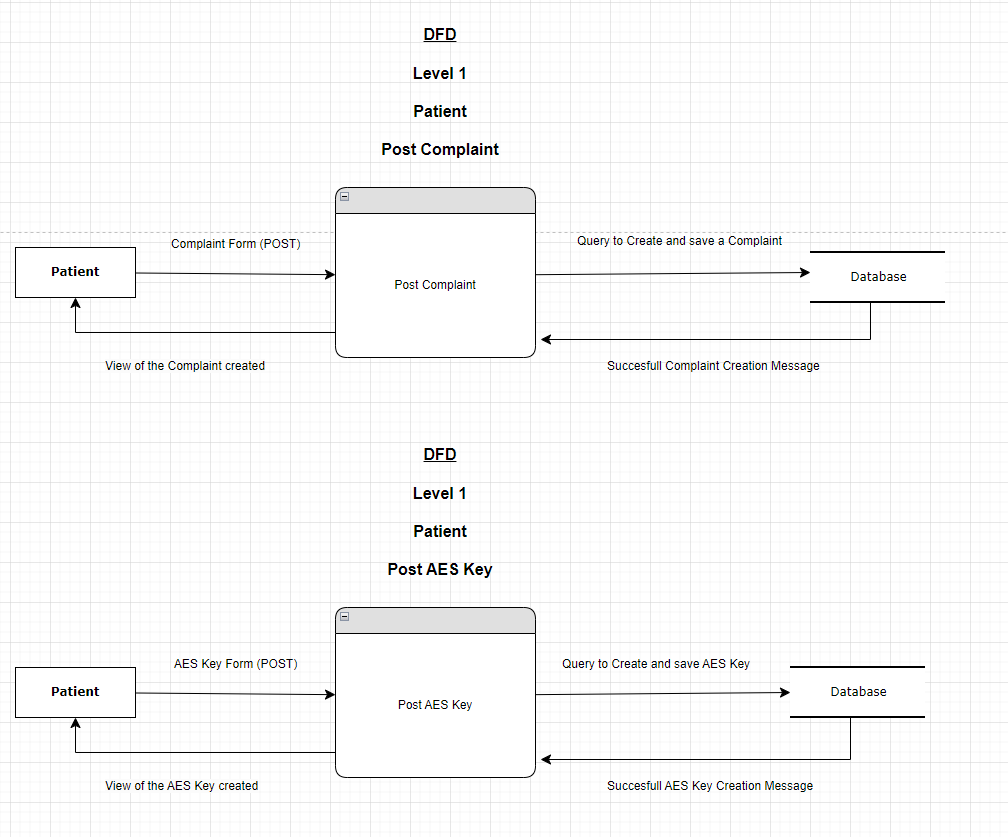


Fig. Data Flow Diagram Level 1 - Post complaint and AES Key

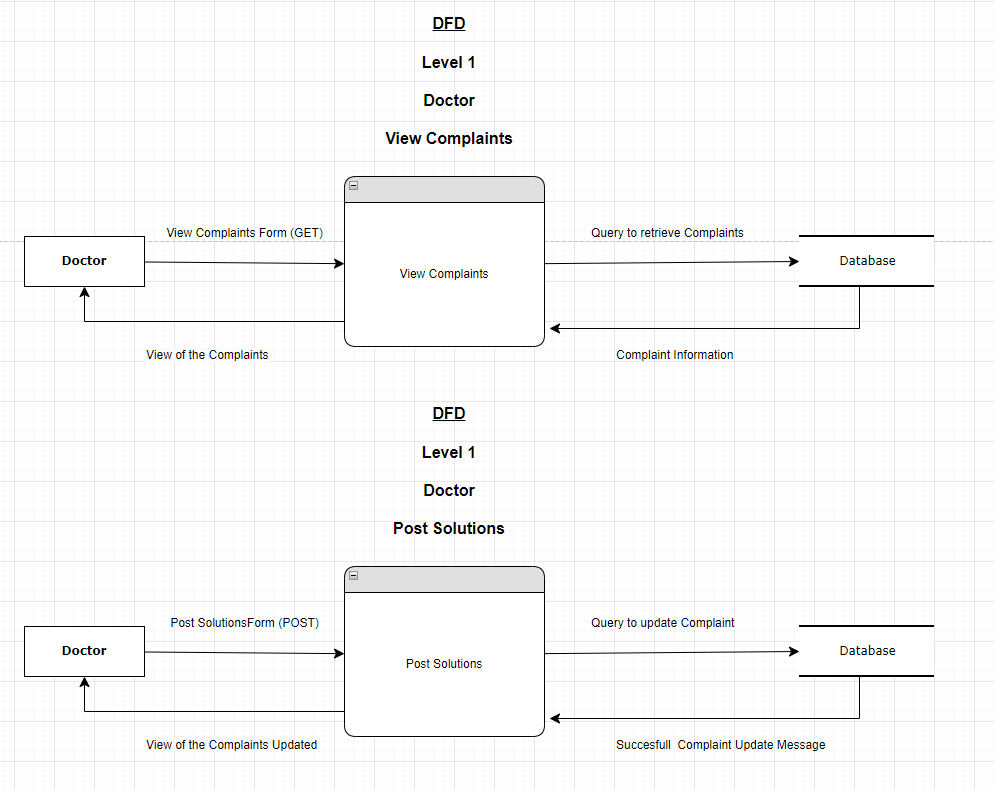


Fig. Data Flow Diagram Level 1 - Doctor View complaint and Post solution

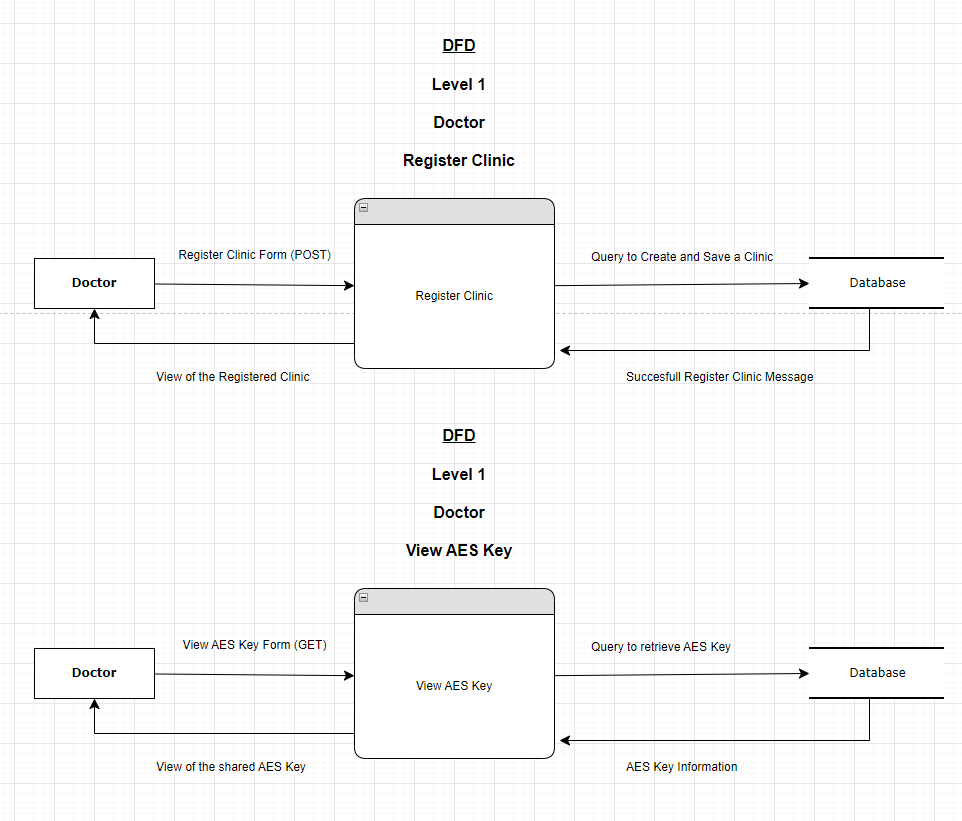


Fig. Data Flow Diagram Level 1 - Doctor Register Clinic and View AES Key

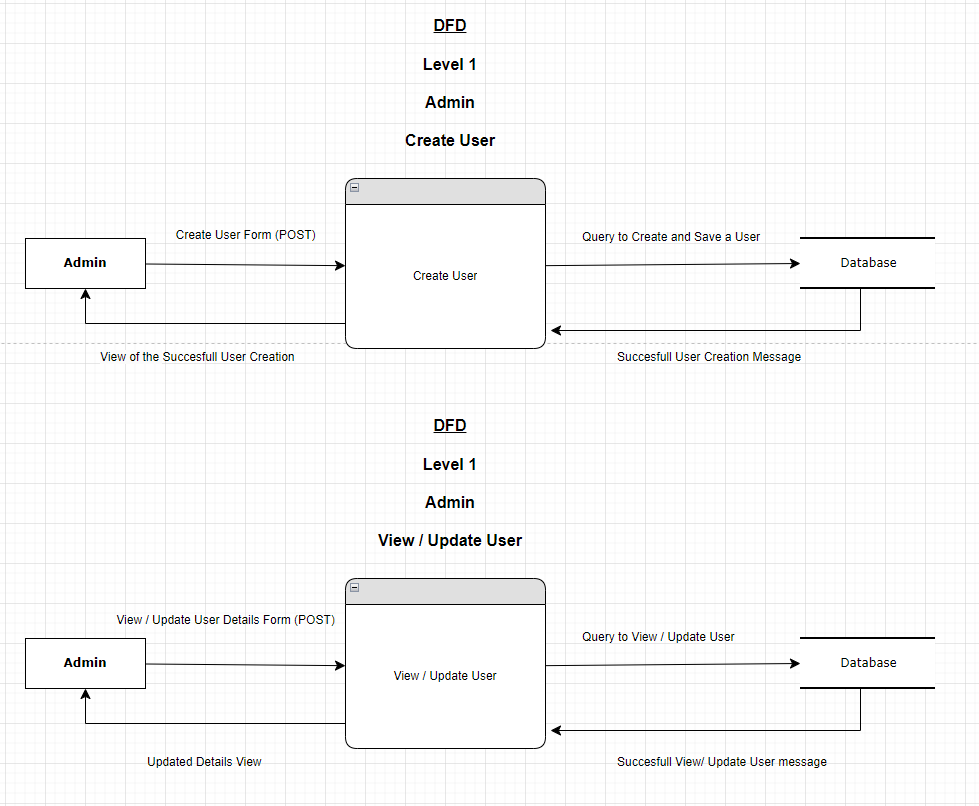


Fig. Data Flow Diagram Level 1 - Admin Create and View/Update user

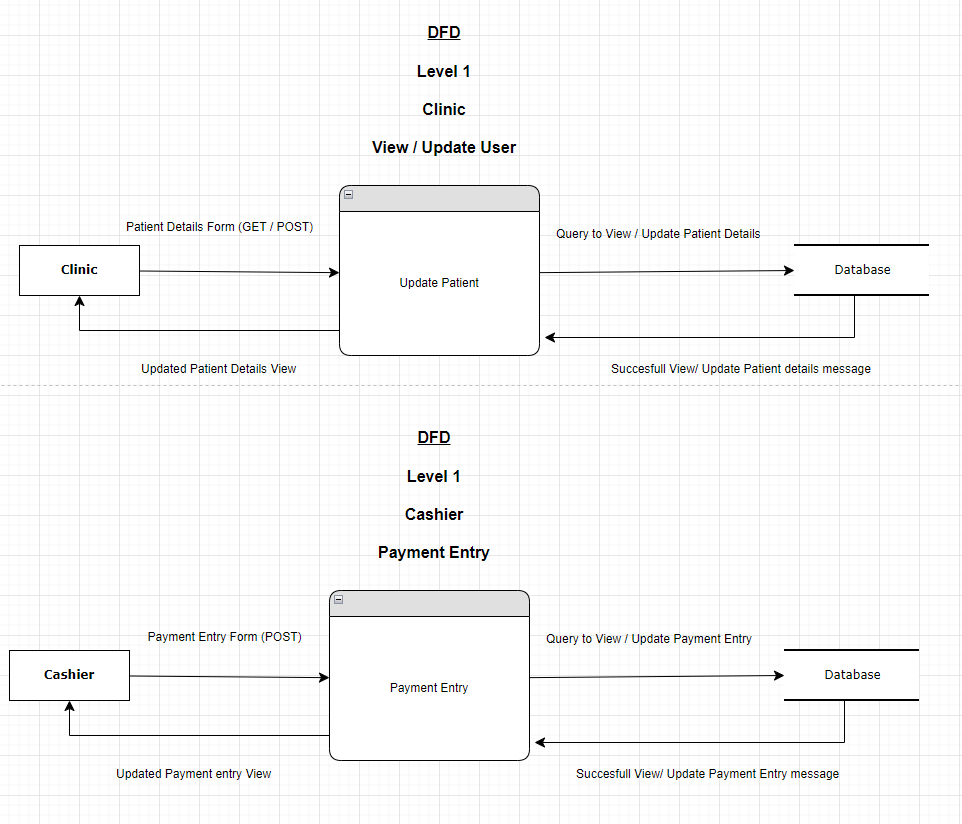


Fig. Data Flow Diagram Level 1 - Clinic and Cashier

## **3.3. Security Implementation**

**Hashing Passwords**

Hashing is an important technique to secure user passwords that are stored in a database. To hash is a one-way function designed to protect the user passwords from unauthorised access even if the database is compromised (Ferguson, Schneier & Kohno, 2010).

***Implementation Steps:***

1. Implement BCrypt hashing algorithm to store passwords and protect against brute-force and rainbow table attacks.
2. Salt Passwords to allow for password duplication and hash identification making it harder to crack the passwords.
3. Hash Passwords applied to the salted passwords to have a fixed-length hash value.
4. Store Hashes and Salts in the database, and never store the password in plain text.
5. Verify the users hash and salts that are saved and allow access on matches.

**Sharing AES Key Encryption**

AES (Advanced Encryption Standard) is an encryption algorithm that encrypts and decrypts data by securely sharing AES keys through OpenSSL & Passwords (Stallings, 2017). It is used to maintain and secure confidential data stores.

***Implementation Steps:***

1. Generate the AES Keys via secure key generation processes from libraries like OpenSSL.
2. Encrypt Keys with OpenSSL allows the decryption to occur only when it has its corresponding keys.
3. Distribute Keys securely via channels like HTTPS to prevent the keys going to malicious actors.
4. Password protection allows for unique and strong passwords, and ensures they are secured properly.

**Role Based Access Control (RBAC)**

RBAC is the process of implementing different interfaces and access to resources or functions depending on the role of the user that is logging in (Sandhu et al., 1996). This ensures users do not access areas of the system not related to their requirements.

***Implementation Steps:***

1. Define Roles by identifying the roles and determine the responsibilities, permissions and access required for each role.
2. Assign Roles to Users when a new registration is created to the system.
3. Implement RBAC logic when designing the application and enforce access permissions dependent on the role level.
4. Interface Customisation based on the different roles will have different views i.e. admin will have more navigation areas on the menus.
5. Audits and Monitors ensure that the system is kept up to date and in order, ensuring that no unauthorised accesses occur on the system.

# **4. User Interface Design**

## **4.1. User Interface Storyboard**

User interface storyboards illustrate how users are going to interact with the system. It shows a sequence of screens that represent this interaction. Here follows Austin Hospital Management System storyboard:

* **Doctor storyboard**



Fig. Doctor storyboard

* **Admin Storyboard**

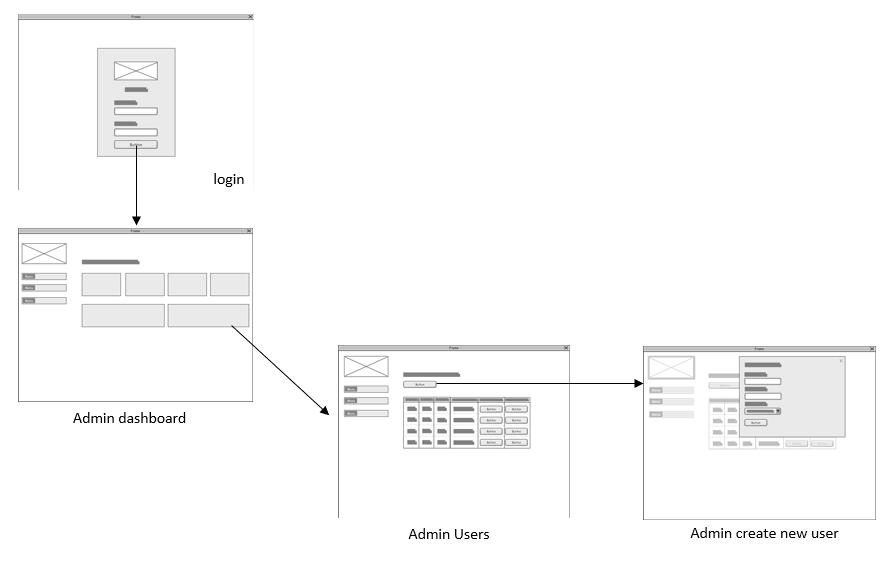
****

Fig. Admin storyboard

* **Clinic Storyboard**

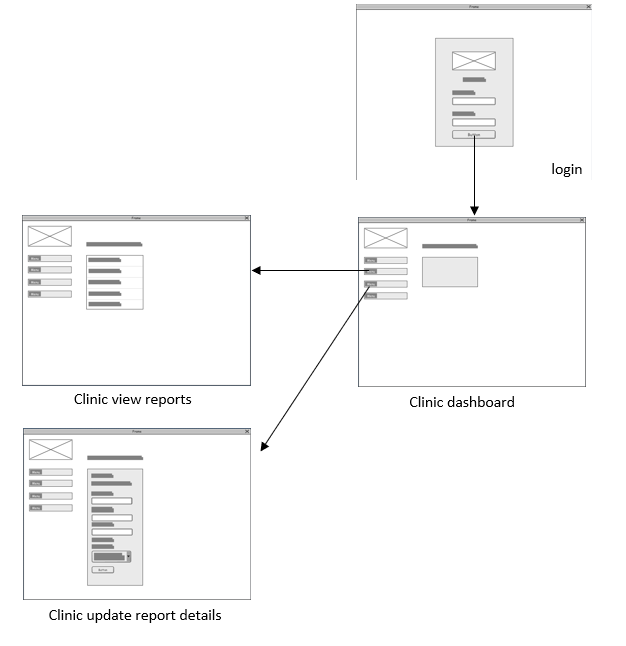
****

Fig. Clinic storyboard

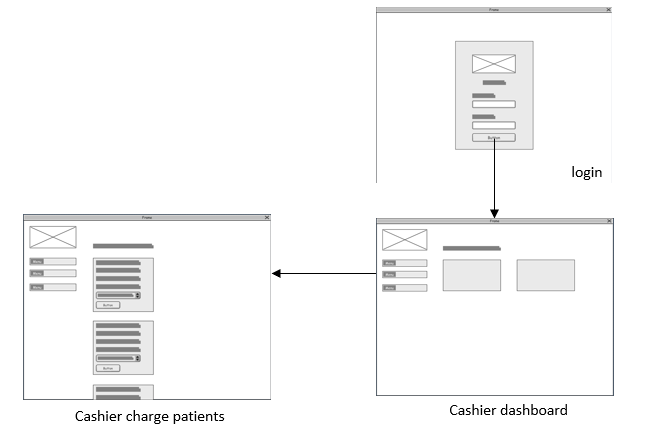
* **Cashier Storyboard**

Fig. Cashier storyboard

* **Patient Storyboard**

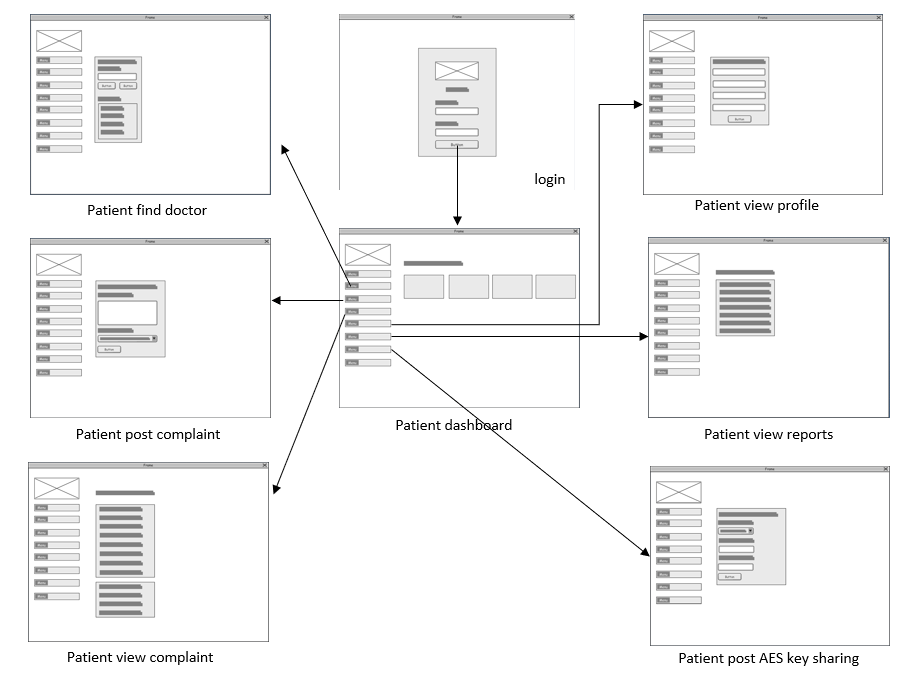
****

Fig. Patient storyboard

## **4.2. Input Data Forms**

Input data form is used to collect data from users to the system. Following AHMS system requirements we developed it to gather specific information:

**Patient post AES Key Sharing**

****

**Cashier charge patients**

****

**Clinic register**

****

**Patient post complaint**

****

**Clinic update report details**

****

**Create new role**

****

**Create new user**

****

**Update user details**

****

**Create procedures item numbers**

****

## **4.3. Output Report Forms**

Output report form shows data that was collected and processed by the system. Based on the previous collection (input data form) this is how we organise the information :

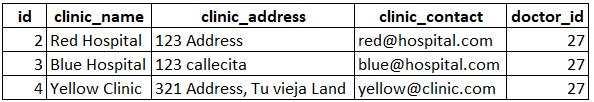
**Patient post AES Key Sharing**



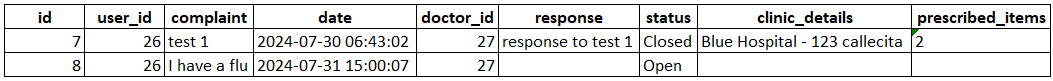
**Cashier charge patients**



**Clinic register**

****

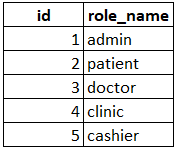
**Patient post complaint**

****

**Clinic update report details**

****

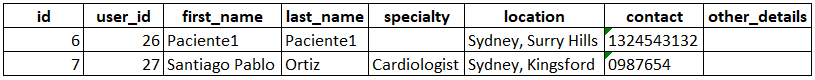
**Create new role**

****

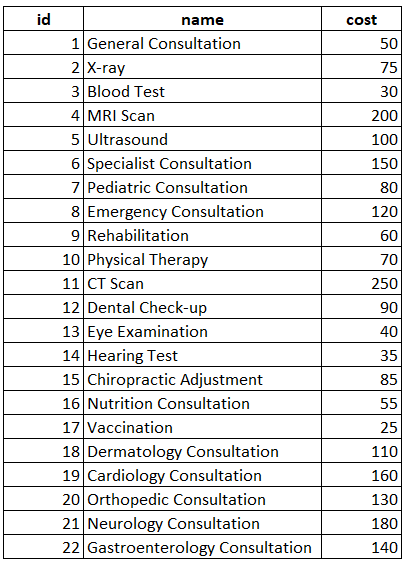
**Create new user**

****

**Update user details**

****

**Create procedures item numbers**

****

# **5. Test Plan & System Implementation Plan**

## **5.1 System Implementation Plan**

1. Initial Setup

Team Creation: Assemble the project team, comprising a management, developers, quality assurance testers, specialists in deployment, and support personnel.

Environment Configuration: Configure development, testing, and production environments together. Make sure backup plans and servers are set up.

Data Migration: Transfer current data to the new system through data migration.

2. Development and Integration

Development: Include features like user registration and incorporate data security measures like AES encryption.

Integration: Merge various system components, such as notification systems and appointment scheduling, to make sure they work well together.

3. Testing

First Testing: To verify individual parts and their interactions, run unit and integration tests.

Quality Assurance: Carry out thorough testing, including security and functionality inspections.

4. Deployment

Staging: To complete testing, deploy the system in a staging environment.

Instruction: Give people instruction on the new system.

Production Launch: Activate the system and track its functionality.

5. Post-Deployment

Support: Provide continuing assistance and deal with any new problems.

Feedback Gathering: Get input from users to guide future developments.

Documentation: Write user guides and technical documentation, then disseminate them.

6. Project Timeline

Weeks 1-2: Preparation

Weeks 3-6: Development and Integration

Weeks 7-8: Internal Testing

Week 9: Staging and Training

Week 10: Go-Live

Ongoing: Support and Feedback

## **5.2 Test Plan**

1. Testing Goals

Verify that every feature operates as intended.

Check that AES encryption is working properly.

Analyse the security and operation of the system.

2. Testing Stages

Unit testing: Examine separate parts, like scheduling and user registration.

Integrity testing: Verify the correct operation of various system components.

System Testing: Verify the general operation of the system.Examine the implementation of AES encryption and access controls as part of security testing.(Kim, Debois, Willis, and Humble 2016).

Performance testing: Evaluate the scalability and response times of the system.

User Acceptance Testing (UAT): UATis a phase of software development in which the software is tested in the real-world scenario by the end-users to ensure that it meets their needs and requirements (Patton, 2006).

3. Testing Schedule

Week 7: Unit Testing

Week 8: Integration Testing

Week 9: System and Security Testing

Weeks 9-10: Performance Testing and UAT

Week 10: Final Review

4. Testing Resources

Testing Environment: Make use of a staging environment for testing.

Tools: Make use of performance and automated testing resources.

Test Data: For thorough testing, use a variety of sample data.

5. Reporting

Track and document any issues discovered.

Incorporate feedback for final adjustments.

# **6. Feedback**

ER diagram: usage of subtype and supertype.

# **7. References**

Ferguson, N., Schneier, B. & Kohno, T. (2010). *Cryptography Engineering: Design Principles and Practical Applications*. Wiley Publishing, Indianapolis.

Stallings, W. (2017). Cryptography and Network Security: Principles and Practice. 7th edn, Pearson, Boston.

Sandhu, R., Coyne, E.J., Feinstein, H.L. & Youman, C.E. (1996). *Role-Based Access Control Models*. IEEE Computer Society Press, Los Alamitos.

DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations by Gene Kim, Patrick Debois, John Willis, and Jez Humble (2016)

Patton, R. (2006). *Software Testing*. 2nd edn, Sams Publishing, Indianapolis.