Patient management and ICU/ Ventilator Emergency Prediction System

HASHAM KHAN BCS173029
AHSAN KHAN BCS173042
MOMIN KHAN BCS173122

Spring-2021

Supervised By
Dr. Abdul Basit Siddiqui

Department of Computer Science

Capital University of Science & Technology, Islamabad

Submission Form for Final-Year

PROJECT REPORT

Version	V 4.0	NUMBER OF MEMBERS	3

TITLE	Patient Management and ICU/ Ventilator Emergency Prediction System

SUPERVISOR NAME	Dr. Abdul Basit Siddiqui

MEMBER NAME	REG. NO.	EMAIL ADDRESS
Hasham Khan	BCS173029	hashamkhancust@gmail.com
Ahsan Khan	BCS173042	ahsanpathan543@gmail.com
Momin Khan	BCS173122	mominkakar184@gmail.com

Member's Signatures

_____

_____

_____

Supervisor's Signature

Note 1: This paper must be signed by your supervisor
Note 2: The soft-copies of your project report, source codes, schematics, and executable should be delivered in a CD

APPROVAL CERTIFICATE

This project, entitled as “Patient management and ICU/ Ventilator Emergency Prediction System” has been approved for the award of

Bachelors of Science in Software Engineering

Committee Signatures:

Supervisor: _____

(Dr. Abdul Basit Siddiqui)

Project Coordinator: _____

(Mr. Ibrar Arshad)

Head of Department: _____

(Dr. Nayyer Masood)

DECLARATION

We, hereby, declare that “No portion of the work referred to, in this project has been submitted in support of an application for another degree or qualification of this or any other university/institute or other institution of learning”. It is further declared that this undergraduate project, neither as a whole nor as a part thereof has been copied out from any sources, wherever references have been provided.

MEMBERS’ SIGNATURES

ACKNOWLEDGEMENTS

First and foremost, we would like to express our sincere gratitude to our project supervisor Dr. Abdul Basit Siddiqui for his patience, rational ideas, valuable suggestions, helpful information and supporting us throughout the final year project. His vast knowledge, and expertise in Databases and especially in Machine Learning enabled us to complete this project successfully. We thank him for his precious time in guiding us, answering our questions and correcting us.

Secondly, we wish to express our gratitude to Dr. Nisar Ahmad, Assistant Professor, FCPS, MRCS (UK) of department of surgery Khyber Medical college and Khyber Teaching Hospital Peshawar for his guidance and assistance throughout the project.

Thirdly, thanks to all faculty members of FYP 203 P_06 i.e., Mr. Hafiz Anas Bilal, Ms. Tayyeba Zaheer, Ms. Asifa Kanwal and Ms. Maryam Zahid for correcting and guiding us throughout the project, without their guidance and endless help, the completion of this project would never have been possible.

Lastly, we thank Capital University of Science and Technology, its administration, its faculty members especially faculty of computing and project coordinator Mr. Ibrar Arshad for providing us such a platform which enabled us to achieve our goals and strive to have a successful career. May the Almighty Allah fill this organization with more peace and let it be source of blessing for the youth of Pakistan.

DEDICATION

It is with genuine gratitude and warm regard that we wholeheartedly dedicate this project to our parents for they have been praying for our success all the time, supporting us financially and keeping us motivated throughout the project.

EXECUTIVE SUMMARY

During the current world-wide pandemic, we lost great lives due to improper management of our hospitals. The issues such as unavailability of ICUs and doctors, contributed a lot to the increasing mortality rate.

This project aims at providing best patient management system to hospitals. The hospital administrators are able to register their hospitals on this system after which they will be provided their login credentials. Hospital administration can then add their doctors (each hospital their own doctors). Patients can be registered on the system to book appointment in any hospital with a certain doctor of specific specialty by choosing the city of their interest.

The doctor can then use their login credentials to sign in to the system and add their own schedules (single slot for single patient) and attend patients. The doctor can admit the patient on a bed, allot an ICU for the patient, if there be need. The doctor can request the hospital administration to reserve an ICU for the patient in any external hospital, if the current hospital have none vacant ICUs. The patient admitted on bed is under strict monitoring, as soon as changes occur in the patient's condition, the system issues a notification message that patient condition is critical. This will help the doctors and management to arrange an ICU for the patient in time.

Hospital administration are able to entertain incoming and outgoing ICU requests. As soon as ICU request is initiated in a hospital, the admin is able to forward the request to any nearby hospital having vacant ICU. The receiving hospital can either accept or reject the request by verifying the health condition of the patient.

Table of Contents

List of Tables

List of Figures

Chapter 1

1. Introduction

1.1. Project Introduction:

Major role is played by technology in our daily life. In such circumstances, in order to let health organizations, work more efficiently and minimize mortality rate, a system based on Machine Learning and enhanced information technology must be developed. The proposed system aims at providing best medical facilities to Health centers worldwide.

As during the current covid-19 pandemic we observed that doctors faced many issues. One of the issues was that there was no system that uses machine learning to predict and alert the doctors about the urgent transfer of a patient to an ICU for Ventilator. The delay in ventilators and ICU Transfers, resulted in a higher mortality rate. The other problem that contributed into increasing mortality rate consisted of unavailability of ventilators and ICU in a current hospital with no centralized system to see other hospitals where the patient is to be shifted to get ventilator and ICU facilities.

The proposed system aims at providing solutions to these problems by developing a web-based application for patient management System. The system based on machine learning to use the dataset of a hospital consisting of predictive variables associated with the patient. The system will use the predictive variables to predict whether the patient's condition is normal or an ICU transfer is needed. Based on these predictions performed on a dataset, the system will notify the doctors about the condition of the patient. Upon the unavailability of ICU in current hospital, the doctors can also request the hospital administration for ICU transfer in other hospitals. Hospital administration can see other nearby hospitals and request to reserve an ICU prior to a patient transfer in order to avoid the demise of a patient.

1.2. Existing Examples / Solutions:

The following table includes some examples of existing examples and a comparison with the existing project.

Table 1.1: Existing Solutions features comparison

Sr no	Characteristics	Shifa International Islamabad	Lady Reading Peshawar	Proposed System
1	Public Appointment Scheduling	✔	✔	✔
2	PATIENT REGISTRATION PROCESS	✔	✔	✔
4	PATIENT RECORD MANAGEMENT	✔	✔	✔
5	Doctors' registration	✔	✔	✔
6	ICU Transfer Prediction			✔
7	Alerting Doctors when ICU Transfer is needed			✔
8	Reserving ICU in other Hospitals if current hospital has none available			✔
9	Hospitals Registration			✔

1.3. Business Scope:

As during the current pandemic, the world witnessed the increase in need and demand of computerized system in health sector. The doctors and all other staff are fed up of the manual entries during the processing of a patient in a hospital. The proposed system can be implemented in worldwide Health organizations, clinics, pharmacies, laboratories and all other health related units. With the implementation of this system, the exponentially increasing mortality rate of patients can be controlled.

1.4. Useful Tools and Technologies:

1.4.1 Visual Studio:

Web Application will be developed using visual studio.

Figure 1.1: Visual Studio

1.4.2 Notepad++:

Notepad ++ will be used to develop the Web Application.

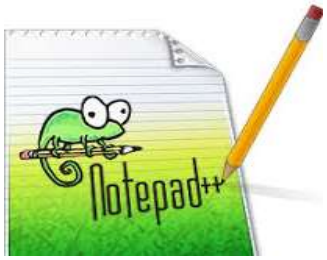

Figure 1.2: Notepad++

1.4.3 SQL Server Management Studio:

SQL Server Management Studio will be used for database.

Figure 1.3: MSSQL Server

1.5. Project Work Break Down:

The following figure shows the project work breakdown of our project.

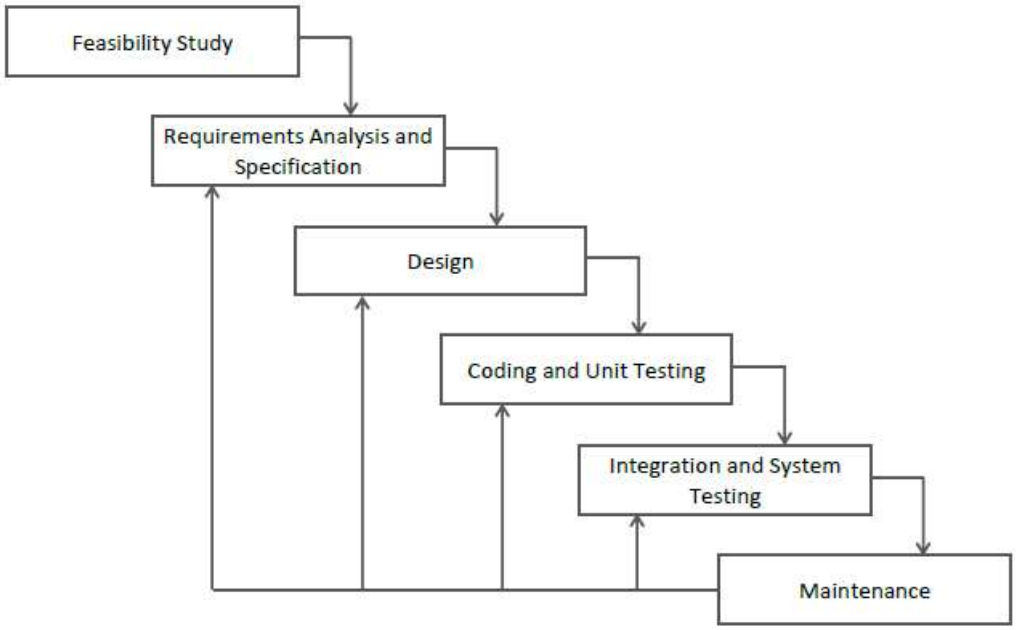

Figure 1.4: Project Work Breakdown Structure

1.6. Project Time Line:

The proposed project timeline is shown in the following the figure.

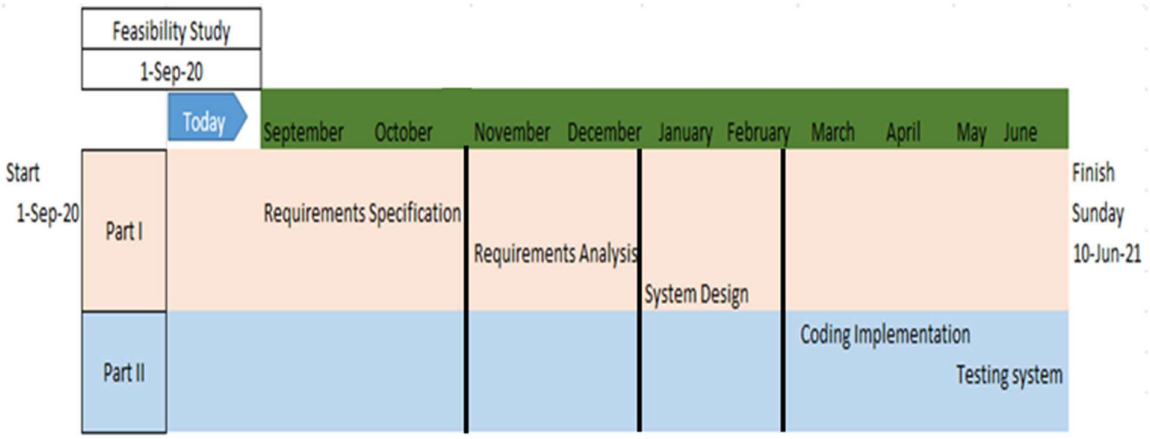

Figure 1.5: Project Timeline

The above timeline will be followed throughout the project.

Chapter 2

2. Requirement Specification and Analysis

Requirement's analysis is a process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. In Chapter 2 we will enlist the functional and non-functional requirements and model functional requirements in the form of use case model.

2.1. Functional Requirements:

A functional requirement defines a function of a system or its component. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish.

Table 2.1: Functional Requirements

S.No	Functional Requirement	Type	Status
1.	The Hospital admin can sign up.	Core	Done
2.	Hospital admin can sign into the system.	Core	Done
3.	Hospital admin can view profile.	Intermediate	Done
4.	Hospital admin can view current appointments.	Intermediate	Done
5.	Hospital admin can view appointments history.	Intermediate	Done

6.	Hospital admin can add doctors.	Core	Done
7.	Hospital admin can add ICU.	Intermediate	Done
8.	Hospital admin can add beds.	Intermediate	Done
9.	Hospital admin can view doctors.	Core	Done
10.	Hospital admin can view inpatient ICU.	Core	Done
11.	Hospital admin can view inpatient beds.	Intermediate	Done
12.	Hospital admin can view inpatient ICU history.	Core	Done
13.	Hospital admin can view inpatient beds history.	Intermediate	Done
14.	Hospital admin can view other hospitals.	Core	Done
15.	Hospital admin can request ICU reservation in other hospitals.	Core	Done
16.	Hospital admin can accept ICU reservation from other hospitals.	Core	Done
17.	Hospital admin can sign out from the system.	Intermediate	Done
18.	Doctor can log into the system using credentials provided by hospital admin.	Core	Done
19.	Doctor can view profile.	Intermediate	Done
20.	Doctor can view his current appointments.	Core	Done
21.	Doctor can view his appointments history.	Intermediate	Done
22.	Doctor can attend a patient.	Core	Done
23.	Doctor can add patient's health record.	Core	Done

24.	Doctor can edit patient health record.	Intermediate	Done
25.	Doctor can specify schedule.	Core	Done
26.	Doctor can admit patient.	Intermediate	Done
27.	Doctor can view inpatient ICU.	Core	Done
28.	Doctor can view inpatient beds.	Core	Done
29.	Doctor can view inpatient ICU history.	Intermediate	Done
30.	Doctor can view inpatient beds history.	Intermediate	Done
31.	Doctor can refer a patient to ICU.	Core	Done
32.	Doctor can transfer patient from bed to ICU.	Core	Done
33.	Doctor can sign out from the system	Intermediate	Done
34.	Patient can visit the website.	Core	Done
35.	Patient can view profile.	Intermediate	Done
36.	Patient can search Hospitals.	Intermediate	Done
37.	Patient can book an appointment with doctor in specified hospital.	Core	Done
38.	Patient can see current appointments	Core	Done
39.	Patient can see appointments history.	Core	Done
40.	System admin can log into the system online.	Core	Done
41.	System admin can view profile.	Intermediate	Done
42.	System admin can verify the hospital details.	Core	Done

43.	System admin can verify a hospital registration.	Core	Done
44.	System admin can view all registered hospitals.	Intermediate	Done

2.2. Non-Functional Requirements:

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions.

Table 2.2: Non-Functional Requirements

S. No.	Non-Functional Requirements	Category
1	The system should give possible suggestions if the user enters wrong input.	Usability
2	Logon ID: Any users who make use of the system need to hold a Logon ID and password.	Security
3	Modifications: Any modifications like insert, delete, update, etc. for the database can be synchronized quickly and executed only by the Hospital administrator.	Security
4	Doctors: Doctors can view any data in the system, add new patients record to the system's database but they don't have any rights to alter any data in it.	Security
5	Administrator rights: The administrator can view as well as alter any information in the System.	Security

6	Response Time: The system provides acknowledgment in just one second once the 'patient's information is checked.	Performance
7	Capacity: The system needs to support at least 500 people at once.	Performance
8	User-Interface: The user interface acknowledges within four seconds.	Performance
9	Conformity: The system needs to ensure that the guidelines of the Microsoft accessibilities are followed.	Performance
10	Back-Up: The system offers the efficiency for data backup.	Maintainability
11	Availability: The system is available all the time.	Reliability
12	The system must be able to grow its users without having negative influence on its performance.	Scalability

2.3. Selected Functional Requirements:

Following is the list of the requirements selected for the current iteration.

Table 2.3: Selected set of Requirements for current Operation

S.No	Selected Functional Requirement	type
1.	Hospital admin can view profile.	New
2.	Hospital admin can view current appointments.	New
3.	Hospital admin can view appointments history.	New
4.	Hospital admin can add doctors.	New

5.	Hospital admin can add ICU.	New
6.	Hospital admin can add beds.	New
7.	Hospital admin can view doctors.	New
8.	Hospital admin can view inpatient ICU.	New
9.	Hospital admin can view inpatient beds.	New
10.	Hospital admin can view inpatient ICU history.	New
11.	Hospital admin can view inpatient beds history.	New
12.	Hospital admin can view other hospitals.	New
13.	Hospital admin can request ICU reservation in other hospitals.	New
14.	Hospital admin can accept ICU reservation from other hospitals.	New
15.	Doctor can view profile.	New
16.	Doctor can view his current appointments.	New
17.	Doctor can view his appointments history.	New
18.	Doctor can attend a patient.	New
19.	Doctor can add patient's health record.	New
20.	Doctor can edit patient health record.	New
21.	Doctor can specify schedule.	New
22.	Doctor can admit patient.	New
23.	Doctor can view inpatient ICU.	New

24.	Doctor can view inpatient beds.	New
25.	Doctor can view inpatient ICU history.	New
26.	Doctor can view inpatient beds history.	New
27.	Doctor can refer a patient to ICU.	New
28.	Doctor can transfer patient from bed to ICU.	New
29.	Doctor can sign out from the system	New
30.	Patient can view profile.	New
31.	Patient can search Hospitals.	New
32.	Patient can book an appointment with doctor in specified hospital.	New
33.	Patient can see current appointments	New
34.	Patient can see appointments history.	New
35.	System admin can view profile.	New

2.4. System Use Case Modeling:

A use case is a list of actions or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system, to achieve a goal. The actor can be a human or other external system. User's use cases are shown in the following the figure.

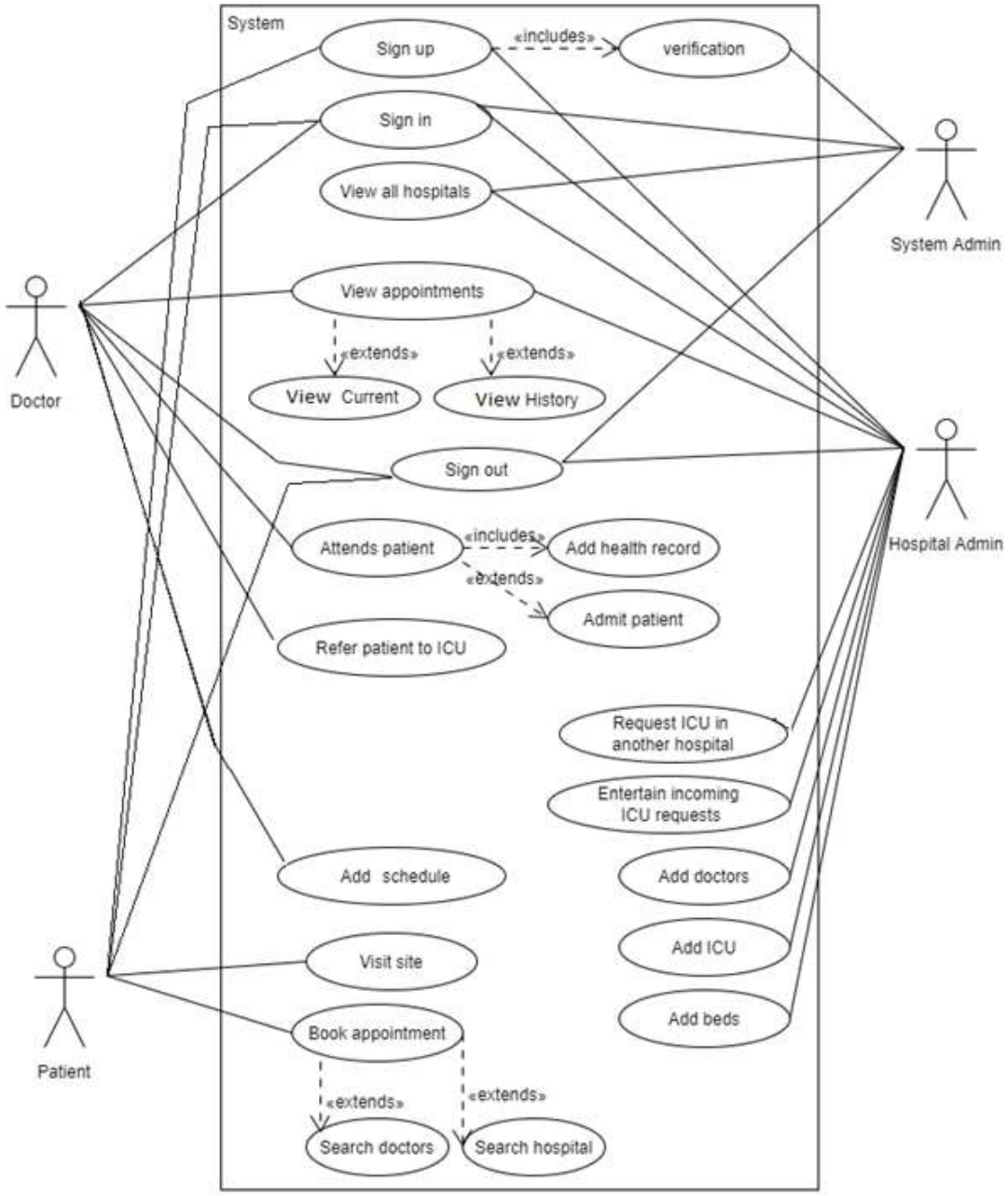

Figure 2.1: System's use case Diagram

2.4.1 Sign up:

Hospital Admin can sign up by providing username, password, hospital name, hospital address and mobile number.

Table 2.4: Sign up

Use Case ID	UC1		
USE Case Name	Sign up		
Created By	Hasham Khan	Last updated by:	
Date Created	11/25/2020	Last Revision Date:	11/25/2020
Actors	Hospital Admin		
Description	Hospital Admin can sign up by providing username, password, hospital name, hospital address and mobile number.		
Trigger	Signup button		
Preconditions	Hospital Admin provides username, password, hospital name, hospital address and mobile number to sign up and click on sign up button.		
Postconditions	Hospital Admin will be signed up to the system and now he/she will able to use the system after the system admin verifies the hospital.		
Normal Flow	Hospital Admin	**System**	
	1. Hospital admin clicks signup button to request for sign up.	2. The system provides sign-up form	
	3. Hospital admin fills in form by providing username, password, address, hospital name and mobile number.	4. System signs up the Hospital Admin.	
Alternate Flows	Hospital Admin cancels the current form.		
Exceptions	1. Database is not responding. 2. Hospital Admin has not filled the form correctly. 3. System is not responding.		

2.4.2 Verification:

System Admin requests the system admin to verify the hospital details.

Table 2.5: Verification

Use Case ID	UC2		
USE Case Name	Verification		
Created By	Ahsan Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	System Admin		
Description	System Admin requests the system admin to verify the hospital details.		
Trigger	Hospital admin signup		
Preconditions	Hospital is signed up.		
Post conditions	Hospital registration is verified.		
Normal Flow	**System**	**System admin**	
	1. System will require the system admin to verify the hospitals details.	2. System admin will verify the hospital and accept its registration.	
Alternate Flows	None		
Exceptions	1. Database is not responding. 2. System is not responding.		

2.4.3 Sign in:

Doctor, System Admin, Hospital Admin will sign into the system by providing Username and password.

Table 2.6: Sign in

Use Case ID	UC3		
USE Case Name	Sign in		
Created By	Momin Khan	Last updated by:	11/04/202
Date Created	11/05/2020	Last Revision Date:	11/04/2020
Actors	Doctor, System Admin, Hospital Admin		
Description	Actors will sign into the system by providing Username and password.		
Trigger	Sign in Button		
Preconditions	The Doctor, System Admin, Hospital Admin provides username, password and then click on the sign in button.		
Post conditions	The Doctor, System Admin, Hospital Admin will be signed in to the system.		
Normal Flow	**Doctor, System Admin, Hospital Admin**	**System**	
	1. Doctor, System Admin or Hospital Admin will click sign in button to request for sign in.	2. The system will provide Doctor, System Admin, Hospital Admin sign in form.	
	3. Doctor, System Admin, Hospital Admin will fill form by providing username, password	4. System will allow Doctor, System Admin, Hospital Admin to log in to the system.	
Alternate Flows	Doctor, System Admin, Hospital Admin will cancel the current form.		

Exceptions	1. Database is not responding. 2. Doctor, System Admin, Hospital Admin has not filled the form correctly.

2.4.4 View all Hospitals:

System Admin, Hospital Admin can view all registered hospitals.

Table 2.7: View all Hospitals

Use Case ID	UC4		
USE Case Name	View all Hospitals		
Created By	Momin Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	System Admin, Hospital Admin		
Description	Actor can view all registered hospitals.		
Trigger	View all Hospitals Button.		
Preconditions	System Admin, Hospital Admin will be signed in.		
Post conditions	System Admin, Hospital Admin can view all registered hospitals.		
Normal Flow	**System Admin, Hospital Admin**	**System**	
	1. System Admin, Hospital Admin clicks view all hospitals button.	2. System will display list of all registered hospitals.	
Alternate Flows	None		
Exceptions	1. Database is not responding. 2. System is not responding.		

2.4.5 View Appointments (Hospital Admin):

Hospital Admin can view appointments of patients with respective doctors

Table 2.8: View Appointments (Hospital Admin)

Use Case ID	UC5		
USE Case Name	View Appointments		
Created By	Momin Khan	Last updated by:	11/04/202
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Hospital Admin		
Description	Actor can view appointments of patients with respective doctors.		
Trigger	View all appointments Button.		
Preconditions	Hospital Admin is logged in.		
Post conditions	Hospital Admin viewed all appointments.		
Normal Flow	**Hospital Admin**	**System**	
	1. Hospital admin clicks View all appointments button	2. System displays all current appointments with their respective doctors.	
Alternate Flows	1. Hospital admin clicks history button	2. System displays history of all appointments with their respective doctors.	
Exceptions	1. Database is not responding. 2. System is not responding.		

2.4.6 View Appointments (Doctor):

Doctor can view patients' appointments with respective doctors.

Table 2.9: View Appointments (Doctor)

Use Case ID	UC6		
USE Case Name	View Appointments		
Created By	Ahsan Khan	Last updated by:	11/04/202
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Doctor		
Description	Doctor can view patients' appointments with respective doctors.		
Trigger	View appointments Button.		
Preconditions	Doctor is logged in.		
Post conditions	Doctor viewed all appointments.		
Normal Flow	**Doctor**	**System**	
	1. Doctor clicks View appointments button	2. System displays all current appointments of the doctor.	
Alternate Flows	1. Doctor clicks history button.	2. System displays history of all appointments of the doctor.	
Exceptions	1. Database is not responding. 2. 1. System is not responding.		

2.4.7 Sign out:

System Admin, Hospital Admin, Doctor and patient can log out of the system.

Table 2.10: Sign out

Use Case ID	UC7		
USE Case Name	Sign out		
Created By	Momin Khan	Last updated by:	11/04/202
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	System Admin, Hospital Admin, Doctor, patient		
Description	Actors can log out of the system.		
Trigger	Log out Button		
Preconditions	System Admin, Hospital Admin, Doctor is logged in.		
Post conditions	System Admin, Hospital Admin, Doctor is logged out of the system.		
Normal Flow	Hospital Admin	**System**	
	1. System Admin, Hospital Admin, Doctor clicks log out button	2. System will process System Admin's, Hospital Admin's or Doctor's request and allow them to sign out.	
Alternate Flows	None		
Exceptions	1. Database is not responding. 2. 1. System is not responding.		

2.4.8 Attends patient:

Doctor can entertain the patients who booked an appointment.

Table 2.11: Attends patient

Use Case ID	UC8		
USE Case Name	Attends patient		
Created By	Momin Khan	Last updated by:	11/04/202
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Doctor		
Description	Actor can entertain the patients who booked an appointment.		
Trigger	Attend button.		
Preconditions	Doctor is logged in. Doctor opened the page of view all appointments.		
Post conditions	Doctor has successfully entertained a patient.		
Normal Flow	Hospital Admin	**System**	
	1. Doctor clicks on attend button.	2. System displays a health record form	
	3. Doctor fill the health record form after examining the patient.	4. System adds the patient health record into database.	
Alternate Flows	Doctor cancels the appointment.		
Exceptions	1. Database is not responding. 2. Doctor has not filled the patient health record the form correctly. 3. System is not responding.		

2.4.9 Admit Patient:

Doctor can admit a patient if his/her condition is worse.

Table 2.12: Admit Patient

Use Case ID	UC9		
USE Case Name	Admit Patient		
Created By	Ahsan Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Doctor		
Description	Actor can admit a patient if his/her condition is worse.		
Trigger	Admit button.		
Preconditions	Doctor is logged in. Doctor attends the patient.		
Post conditions	Patient is successfully admitted in the hospital.		
Normal Flow	**Doctor**	**System**	
	1. Doctor clicks on admit button.	2. System allocates a bed for the patient.	
Alternate Flows	Doctor clicks on admit button.	System is unable to allocate a bed for the patient as there are no vacant beds available.	
Exceptions	1. Database is not responding. 2. System is not responding.		

2.4.10 Refer Patient to ICU:

Doctor can refer a patient to ICU

Table 2.13: Refer Patient to ICU

Use Case ID	UC10		
USE Case Name	Refer Patient to ICU		
Created By	Momin Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Doctor		
Description	Actor can refer a patient to ICU		
Trigger	Refer to ICU button		
Preconditions	Doctor is logged in. Doctor attends the patient.		
Post conditions	Request for ICU shifting is sent to hospital admin.		
Normal Flow	**Doctor**	**System**	
	1. Doctor clicks on shift to ICU button.	2. System prompts hospital admin to reserve ICU for the patient.	
Alternate Flows	None		
Exceptions	1. Database is not responding. 2. System is not responding.		

2.4.11 Reserve ICU

Hospital Admin can reserve ICU for patients.

Table 2.14: Reserve ICU

Use Case ID	UC11		
USE Case Name	Reserve ICU		
Created By	Momin Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Hospital Admin		
Description	Actor can reserve ICU for patients.		
Trigger	Reserve button		
Preconditions	Hospital Admin is logged in.		
Post conditions	Patient is shifted to ICU.		
Normal Flow	**Hospital Admin**	**System**	
	1. Hospital Admin clicks on ICU.	2. System displays all incoming ICU request	
	3. Hospital admin clicks on a request.	4. System displays details of the patient ICU request.	
	5. Hospital Admin clicks on reserve button.	6. System allocates an ICU for the patient.	
Alternate Flows	Hospital Admin cancels the request.		
Exceptions	1. Database is not responding. 2. System is not responding.		

2.4.12 Request ICU in another hospital:

Hospital Admin can reserve ICU for patients in another hospitals.

Table 2.15: Request ICU in another hospital

Use Case ID	UC12		
USE Case Name	Request ICU in another hospital		
Created By	Ahsan Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Hospital Admin		
Description	Actor can reserve ICU for patients in another hospitals.		
Trigger	Request ICU button		
Preconditions	Hospital Admin is logged in.		
Post conditions	ICU reservation request is sent to other hospital.		
Normal Flow	**Hospital Admin**	**System**	
	1. Hospital Admin clicks on ICU.	2. System displays all incoming ICU request	
	3. Hospital admin clicks on a request.	4. System displays details of the patient ICU request.	
	5. Hospital Admin clicks on request other hospitals button.	6. System displays all the hospitals with respective vacant ICU.	
	7. Hospital admin clicks on Request ICU button.	8. System prompts the requested hospital to accept or reject the request.	

Alternate Flows	Hospital Admin cancels the request.
Exceptions	1. Database is not responding. 2. System is not responding.

2.4.13 Entertain incoming ICU requests:

Hospital Admin can accept or reject incoming requests for ICU from other hospitals.

Table 2.16: Entertain incoming ICU requests

Use Case ID	UC13		
USE Case Name	Entertain incoming ICU requests		
Created By	Hasham Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Hospital Admin		
Description	Actor can accept or reject incoming requests for ICU from other hospitals.		
Trigger	Accept button.		
Preconditions	Hospital Admin is logged in.		
Post conditions	Incoming requests for ICU from other hospital is accepted or rejected.		
Normal Flow	Hospital Admin	**System**	
	1.Hospital Admin clicks on ICU.	2. System displays all incoming ICU requests from other hospitals.	
	3.Hospital admin clicks on a request.	4.System displays details of the patient ICU request.	

	5.Hospital admin clicks on accept Request button.	6.System adds patient record to current hospitals ICU database.
Alternate Flows	Hospital Admin rejects the request.	
Exceptions	1. Database is not responding. 2. System is not responding.	

2.4.14 Add Doctor:

Hospital Admin can add their hospital doctor record in database.

Table 2.17: Add Doctors

Use Case ID	UC14		
USE Case Name	Add Doctors		
Created By	Ahsan Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Hospital Admin		
Description	Actor can add their hospital doctor record in database.		
Trigger	Add Doctor button		
Preconditions	Hospital Admin is logged in.		
Post conditions	A doctor added to hospital doctor's database.		
Normal Flow	Hospital Admin	**System**	
	1. Hospital Admin visits doctors page.	2. System displays all registered doctors and an option of doctor's registration form.	

	3. Hospital Admin fills the form and clicks add doctor button.	4. System adds the doctor record to current hospital's doctor database.
Alternate Flows	Hospital Admin cancels the form.	
Exceptions	1. Database is not responding. 2. System is not responding. 3. Hospital Admin has not filled the doctor registration form correctly.	

2.4.15 Add ICU:

Hospital Admin can add beds in hospital database.

Table 2.18: Add ICU

Use Case ID	UC15		
USE Case Name	Add ICU		
Created By	Hasham Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Hospital Admin		
Description	Actor can add beds in hospital database.		
Trigger	Add ICU button		
Preconditions	Hospital Admin is logged in.		
Post conditions	A bed is added to hospital database.		
Normal Flow	Hospital Admin	**System**	

	1. Hospital Admin visits ICU page.	2. System displays all ICUs and an option to add ICU.
	3. Hospital Admin clicks add ICU button.	4. System adds the doctor record to current hospital's doctor database.
Alternate Flows	Hospital Admin cancels the form.	
Exceptions	1. Database is not responding. 2. System is not responding. 3. Hospital Admin has not filled the doctor registration form correctly.	

2.4.16 Add Doctor schedule:

Doctor can add his schedule.

Table 2.19: Add Doctor schedule

Use Case ID	UC16		
USE Case Name	Add Doctor schedule		
Created By	Hasham Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Doctor		
Description	Actor adds his schedule.		
Trigger	Schedule button.		
Preconditions	Doctor is logged in.		
Post conditions	A doctor schedule is specified.		

Normal Flow	Doctor	System
	1. Doctor visit Schedule page.	2. System displays all registered Schedules.
	3. Doctor clicks on add schedule button.	4. System adds the schedule in database.
Alternate Flows	None.	
Exceptions	1. Database is not responding. 2. System is not responding.	

2.4.17 Add bed:

Hospital Admin can add beds in hospital database.

Table 2.20: Add bed

Use Case ID	UC17		
USE Case Name	Add bed		
Created By	Hasham Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Hospital Admin		
Description	Actor can add beds in hospital database.		
Trigger	Add bed button		
Preconditions	Hospital Admin is logged in.		
Post conditions	A bed is added to hospital database.		
Normal Flow	Hospital Admin	**System**	
	1.Hospital Admin visits beds page.	2. System displays all beds and an option to add bed.	

	3.Hospital Admin clicks add bed button.	4.System adds the bed record to current hospital's database.
Alternate Flows	None	
Exceptions	1. Database is not responding. 2. System is not responding.	

2.4.18 Visit site:

Patient can visit the website.

Table 2.21: Visit site

Use Case ID	UC18		
USE Case Name	Visit site		
Created By	Ahsan Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Patient		
Description	Actor can visit the website.		
Trigger	URL of website		
Preconditions	None		
Post conditions	Patient has visited the site.		
Normal Flow	Patient	**System**	
	1.Patient visit website's URL.	2. Website is opened.	
Alternate Flows	None.		
Exceptions	None.		

2.4.19 Book appointment:

Patient can book an appointment.

Table 2.22: Book appointment

Use Case ID	UC19		
USE Case Name	Book appointment		
Created By	Ahsan Khan	Last updated by:	
Date Created	11/04/2020	Last Revision Date:	11/04/2020
Actors	Patient		
Description	Actor can Book appointment.		
Trigger	Book appointment button		
Preconditions	None		
Post conditions	Patient appointment is booked.		
Normal Flow	**Customer**	**System**	
	1. Patient clicks on book appointment button.	2. System displays the list of hospitals.	
	3. Patient chooses hospital and clicks next.	4. System asks the user about department.	
	5. Patient chooses the department.	6. System displays all the doctors of that department.	
	7.Patient chooses a doctor and clicks next.	8. System displays the timings of the doctor	
	9. Chooses the time schedule and clicks book appointment button.	10. System adds the patient record to appointments table in database.	
Alternate Flows	Patient cancels the booking.		

Exceptions	1. Database is not responding. 2. System is not responding. 3. 

2.5. System Sequence diagram:

System sequence diagram (SSD shows, for a particular scenario of a use case, the events that external actors generate their order, and possible inter-system events.

2.5.1 Patient appointment booking:

Patient requests for booking appointment by choosing certain doctor in specific hospital.

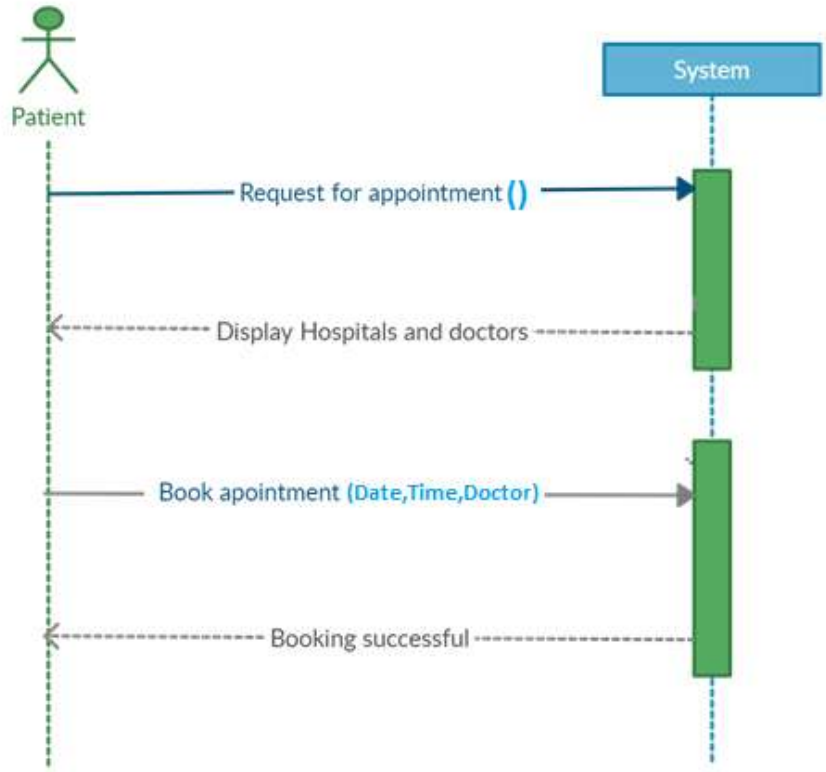

Figure 2.2: SSD (Patient appointment booking)

2.5.2 Hospital Admin Signup:

Hospital admin can register the respective hospital by requesting signup and providing the respective details after the system displays the signup form.

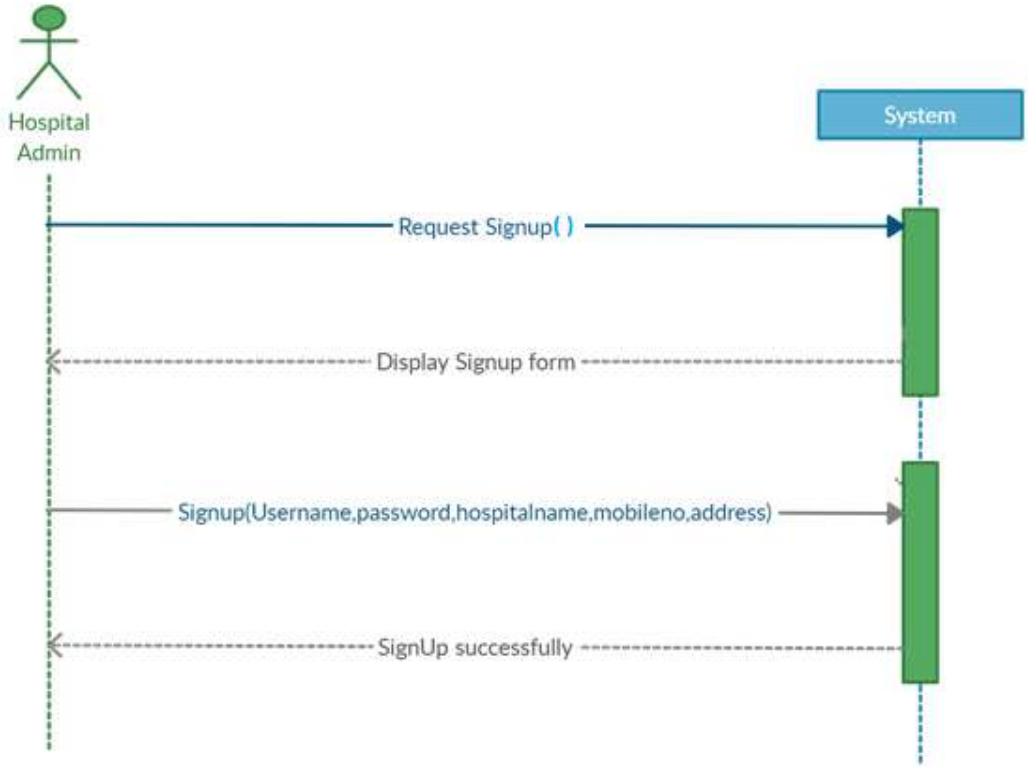

Figure 2.3: SSD (Hospital Admin Signup)

2.5.3 Hospital Admin Login:

Hospital admin uses his credentials to request login to the system. The system verifies the credentials and allow the user to proceed further.

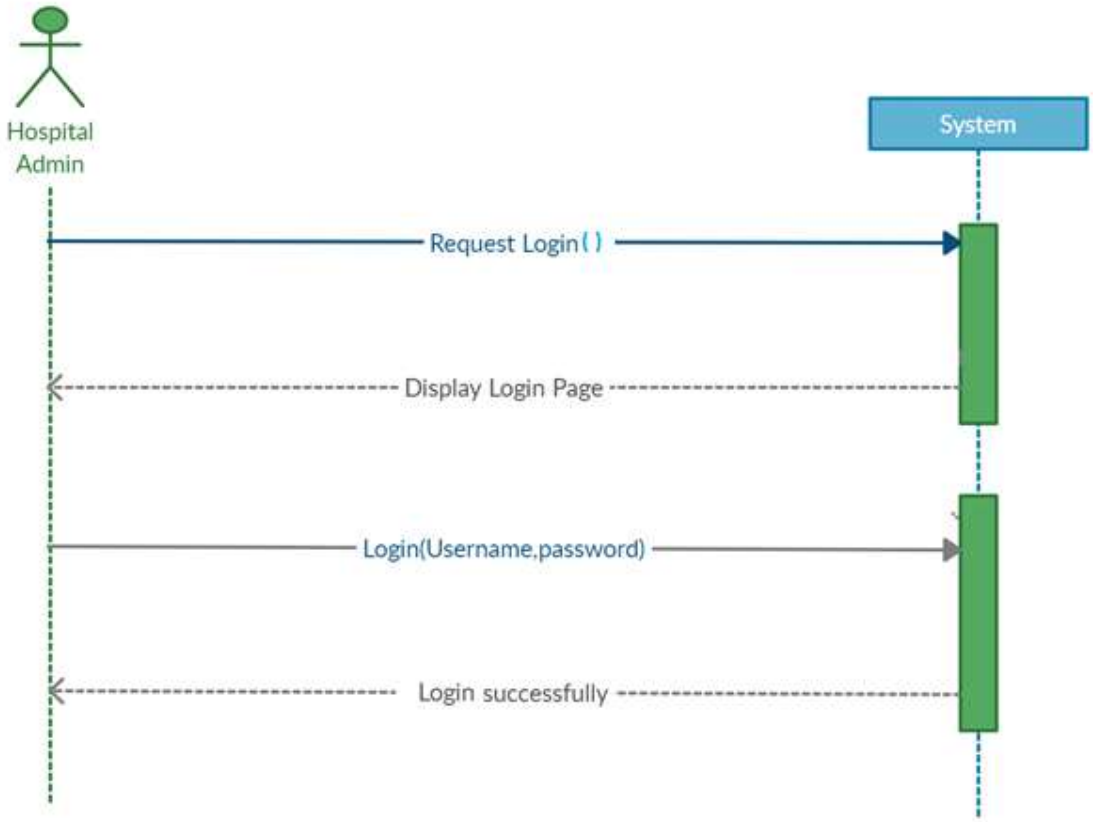

Figure 2.4: SSD (Hospital Admin Login)

2.5.4 Hospital Admin sign out:

Hospital Admin uses his credentials to request sign out to the system. The system confirm the credentials and Display all doctors.

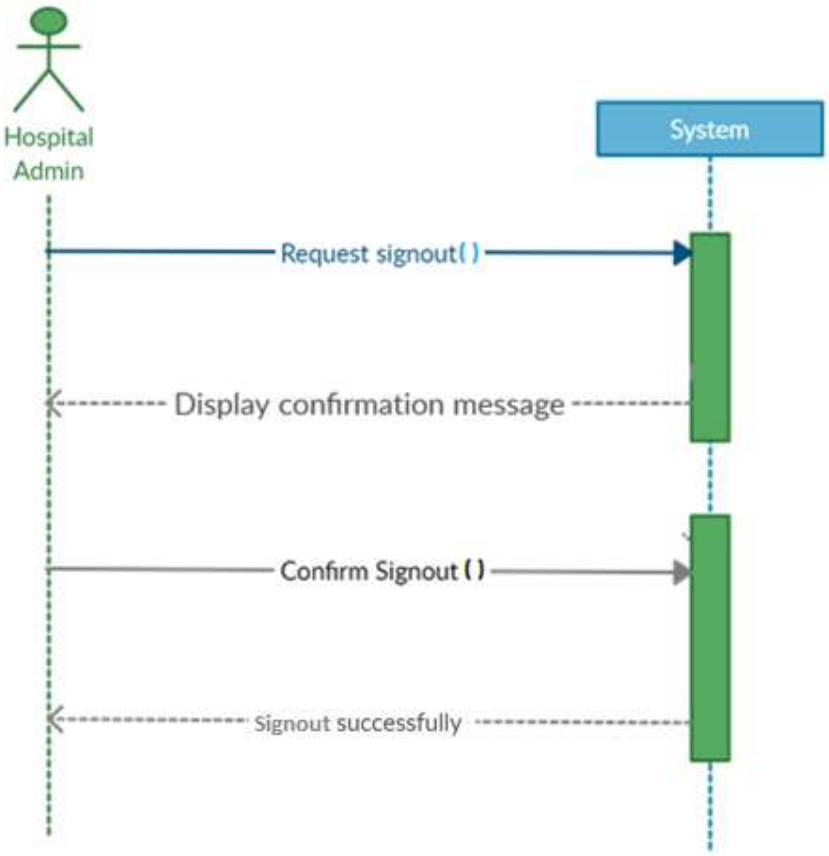

Figure 2.5: SSD (Hospital Admin sign out)

2.5.5 Doctor Login:

Doctor uses his credentials to request login to the system. The system verifies the credentials and allow the user to proceed further.

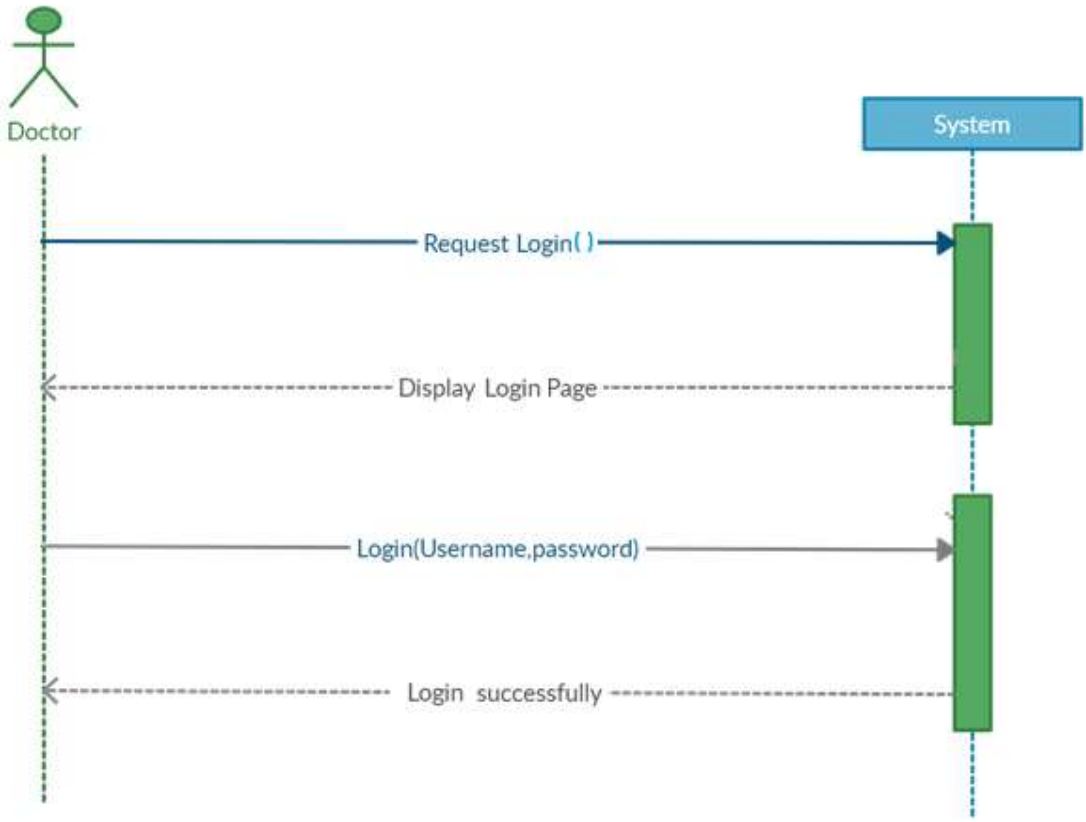

Figure 2.6: SSD (Doctor Login)

2.5.6 Doctor view appointments:

Doctor uses his credentials to request login to view appointments. The system verifies the credentials and display all appointments.

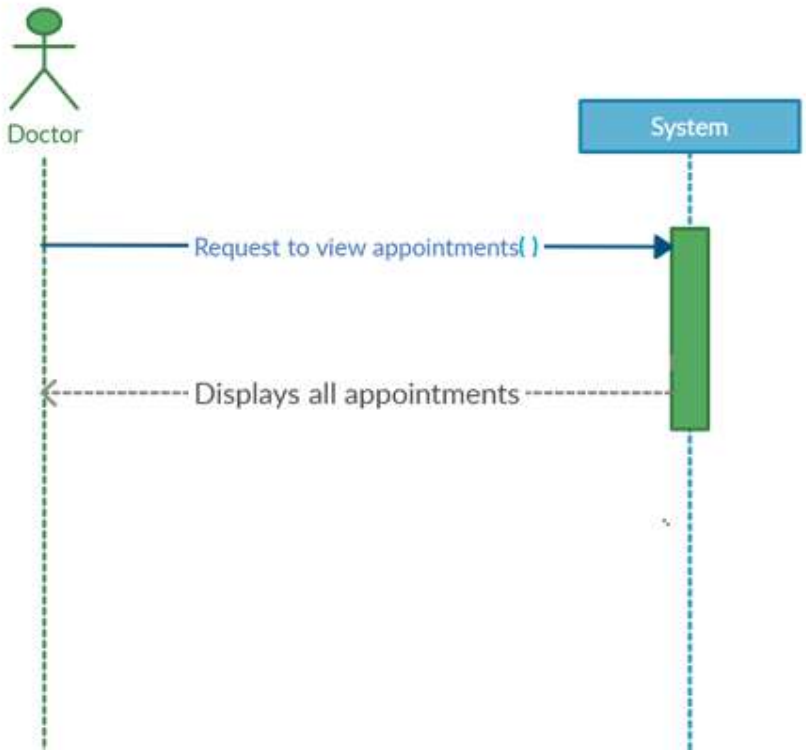

Figure 2.7: SSD (Doctor view appointments)

2.5.7 Doctor attends patient:

Doctor uses his credentials to request attend patient to the system. The system displays patient health form. Doctor adds health record of patient. The system attended patient sucessfully. Doctor request to admit patient to the system. The system send request to the hosptital admin.

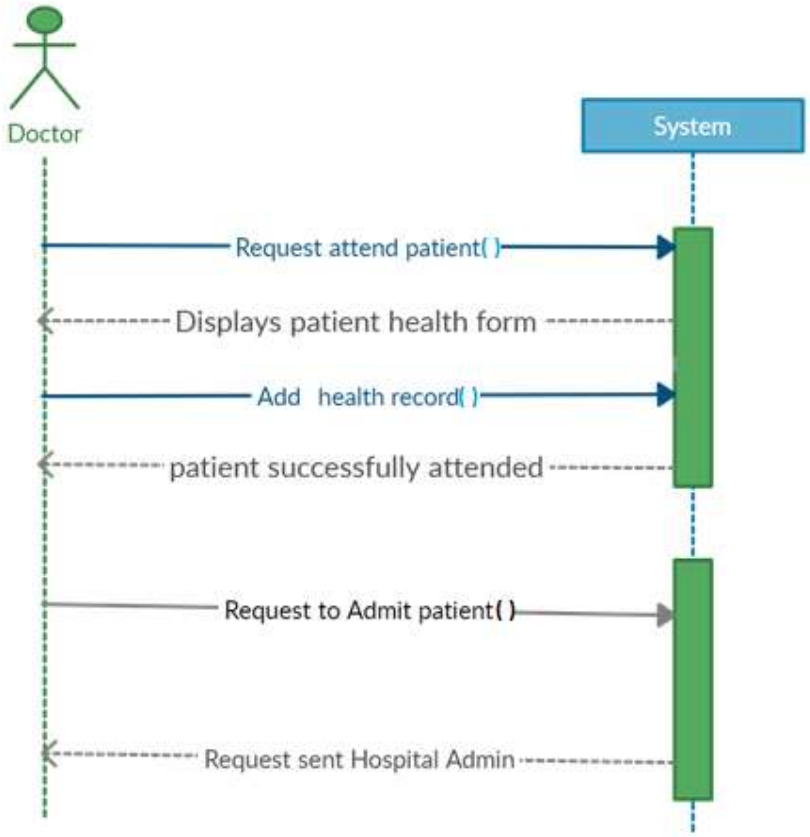

Figure 2.8: SSD (Doctor attends patient)

2.5.8 Doctor request ICU transfer:

Doctor uses his credentials to request ICU transfer to the system. The system send request to the hospital admin.

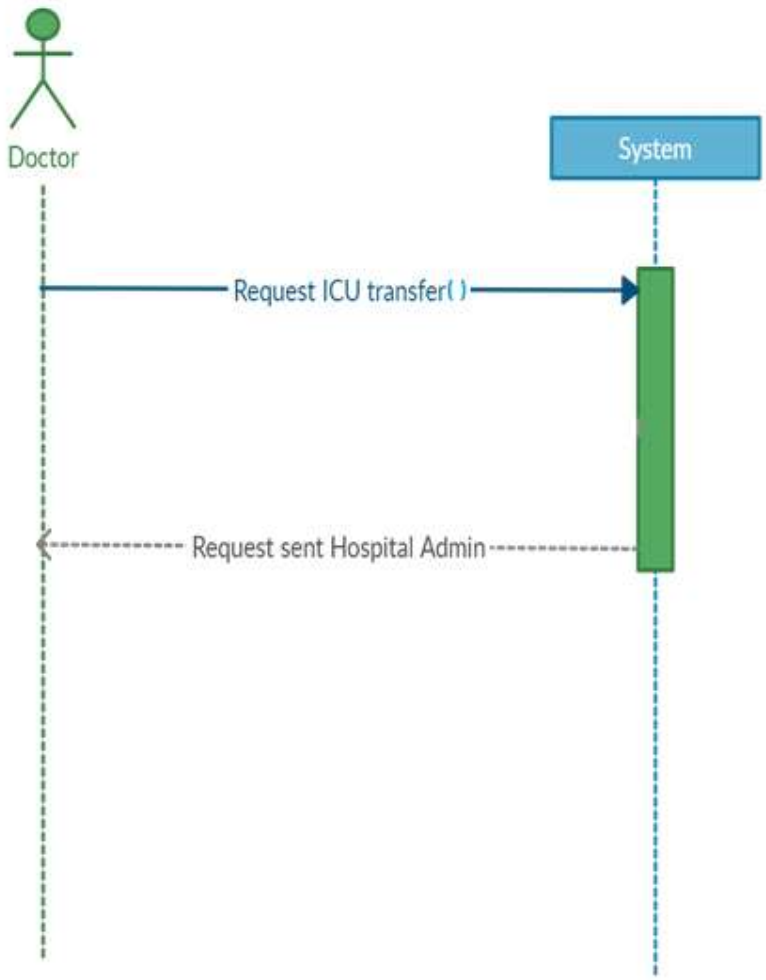

Figure 2.9: SSD (Doctor request ICU transfer)

2.5.9 Doctor sign out:

Doctor press the logout button and the user is logged out; thus, the session is ended.

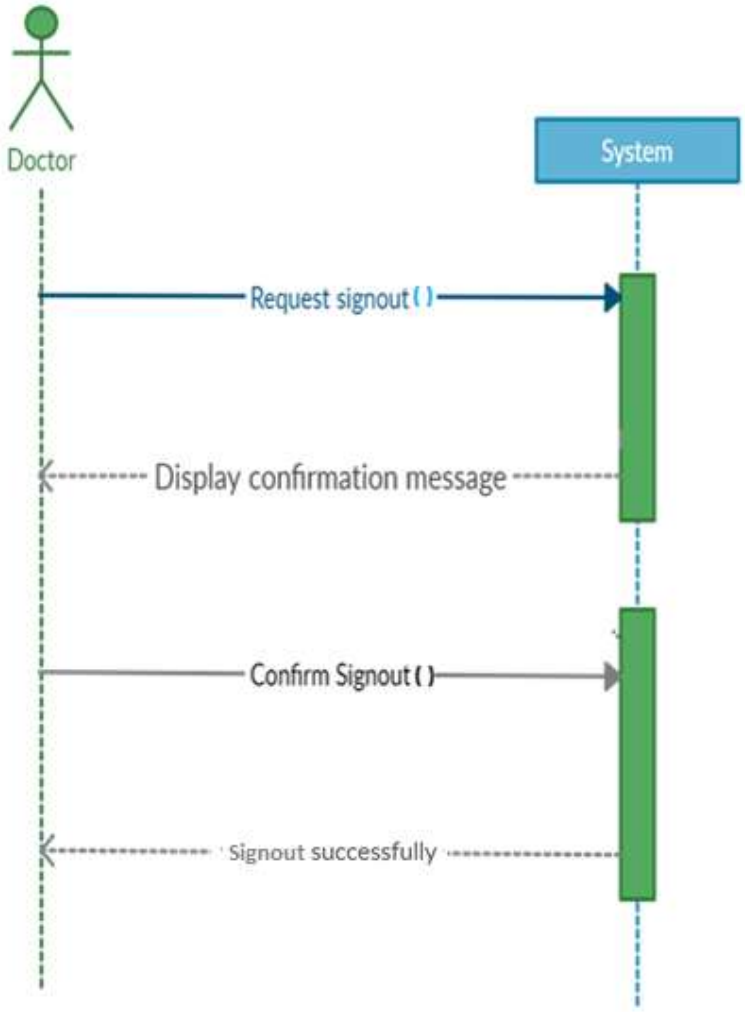

Figure 2.10: SSD (Doctor sign out)

2.5.10 System Admin Login:

System admin uses his credentials to request login to the system. The system verifies the credentials and allow the user to proceed further.

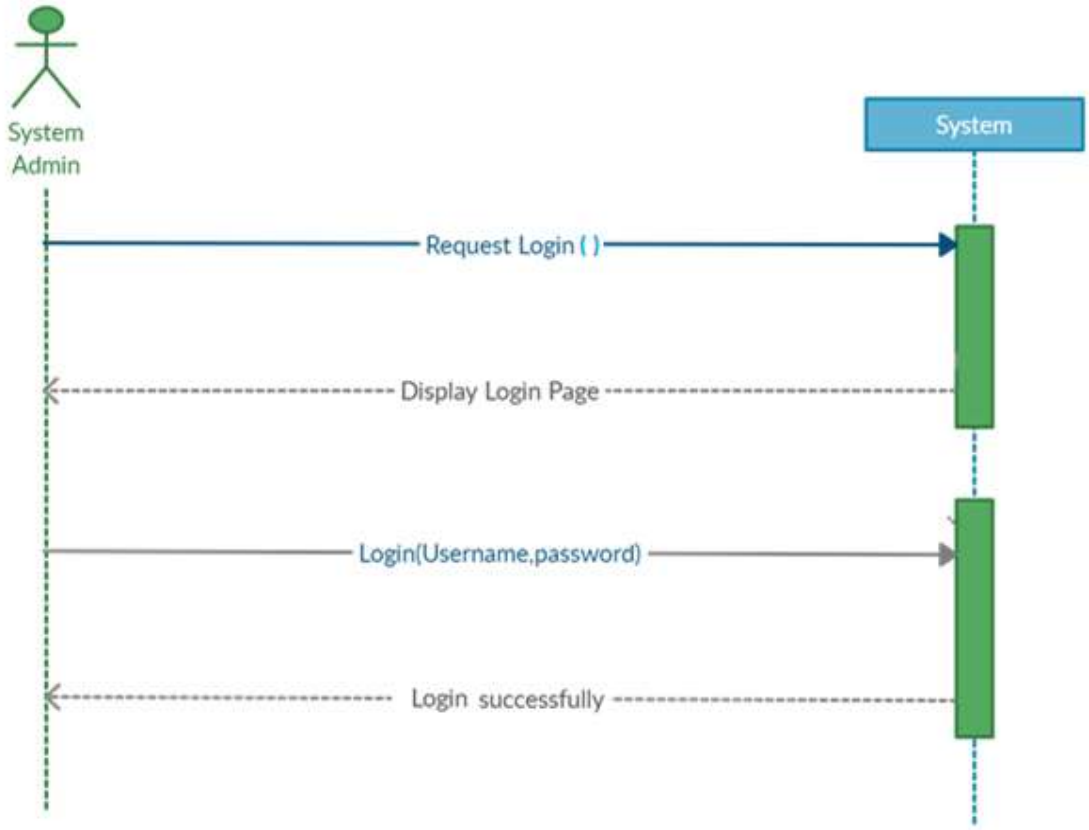

Figure 2.11: SSD (System Admin Login)

2.5.11 System Admin View all hospitals:

System admin can view all hospitals. The system will display all hospitals. System admin can view all doctors. The system will display all doctors.

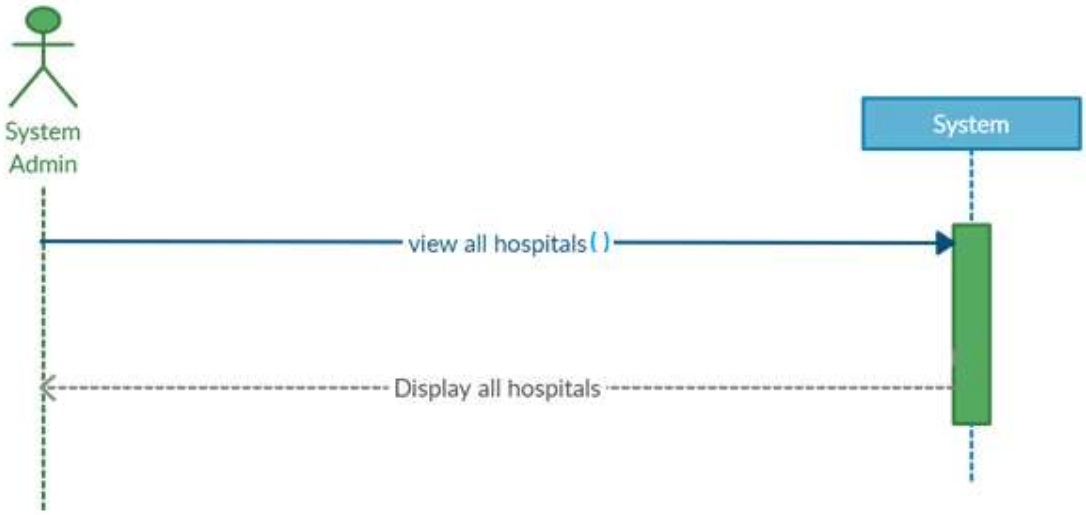

Figure 2.12: SSD (System Admin View all hospitals)

2.5.12 System Admin Verification:

System sends request to system admin for verification of new hospital. System admin verify hospital.

Figure 2.13: SSD (System Admin Verification)

2.5.13 Hospital admin entertain incoming ICU requests:

System requests to hospital admin for accept incoming ICU reservation. Hospital admin accept reservation request.

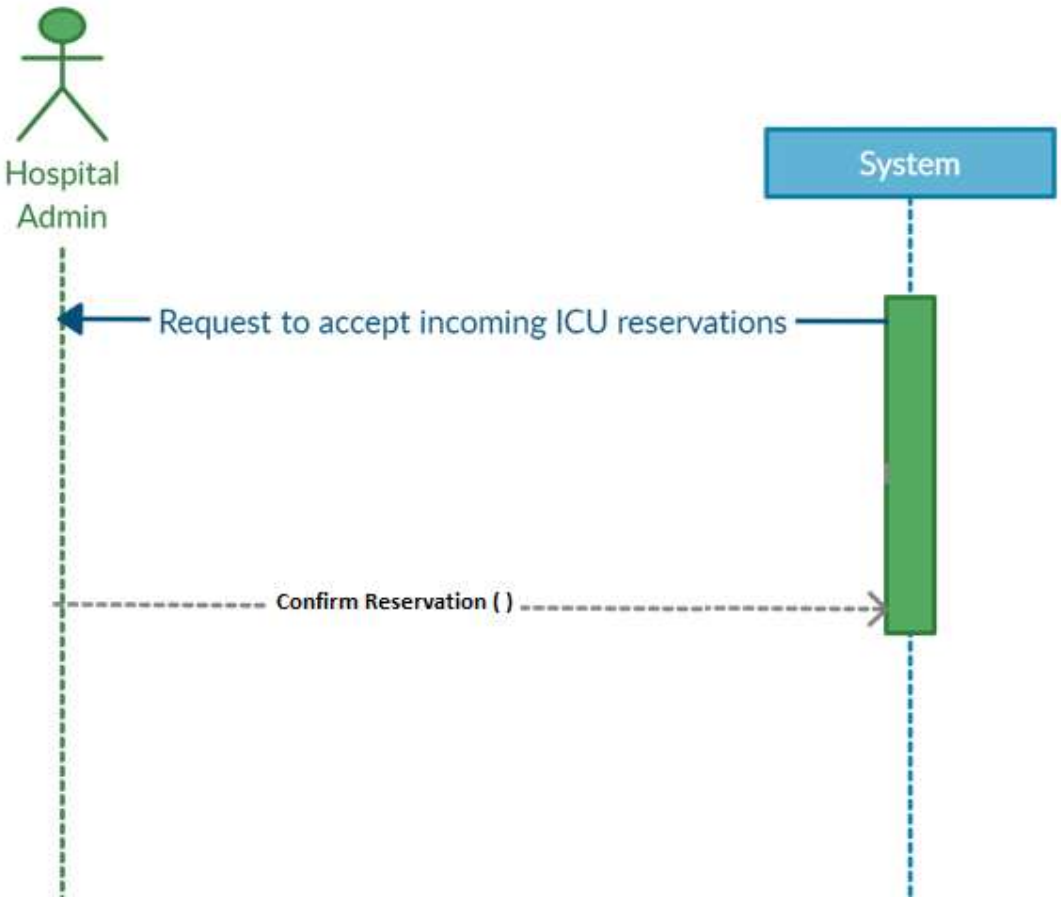

Figure 2.14: SSD (Hospital admin entertain incoming ICU requests)

2.5.14 Hospital admin add doctor:

Hospital admin uses his credentials to request to add doctor to the system. The system display doctor registration form. Hospital admin request to add schedule to the system. System display schedule form to hospital admin.

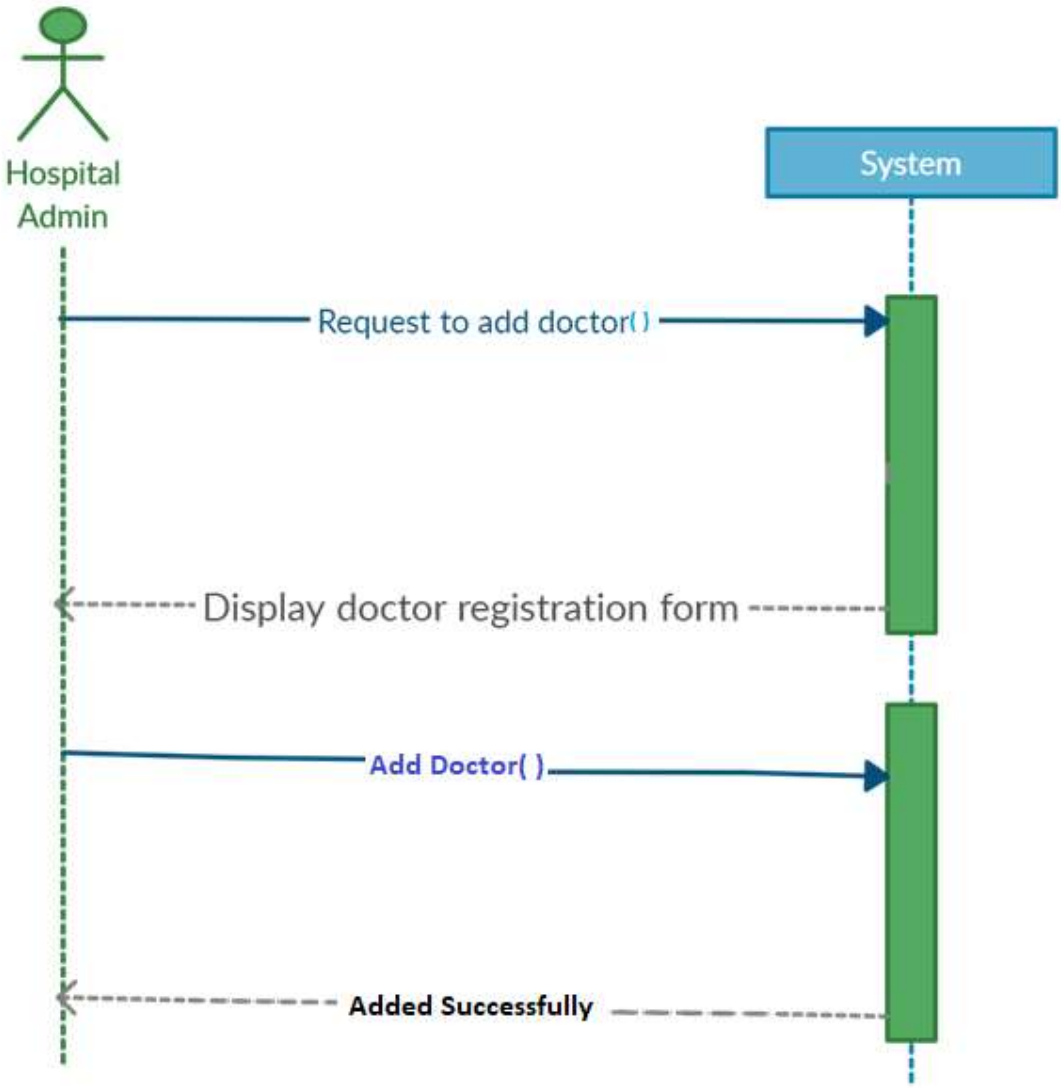

Figure 2.15: SSD (Hospital admin add doctor)

2.5.15 Hospital admin add bed:

Hospital admin request to add bed to the hospital. The system will add bed and display bed details.

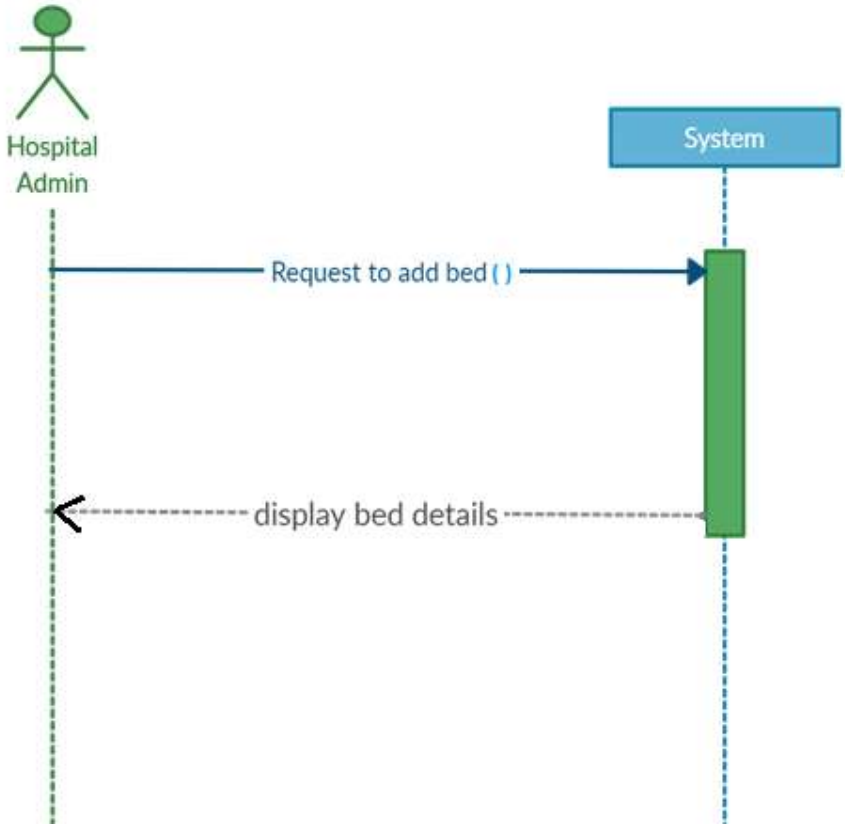

Figure 2.16: SSD (Hospital admin add bed)

2.5.16 Hospital admin add ICU:

Hospital admin request to add ICU to the hospital. The system will add ICU and display ICU details.

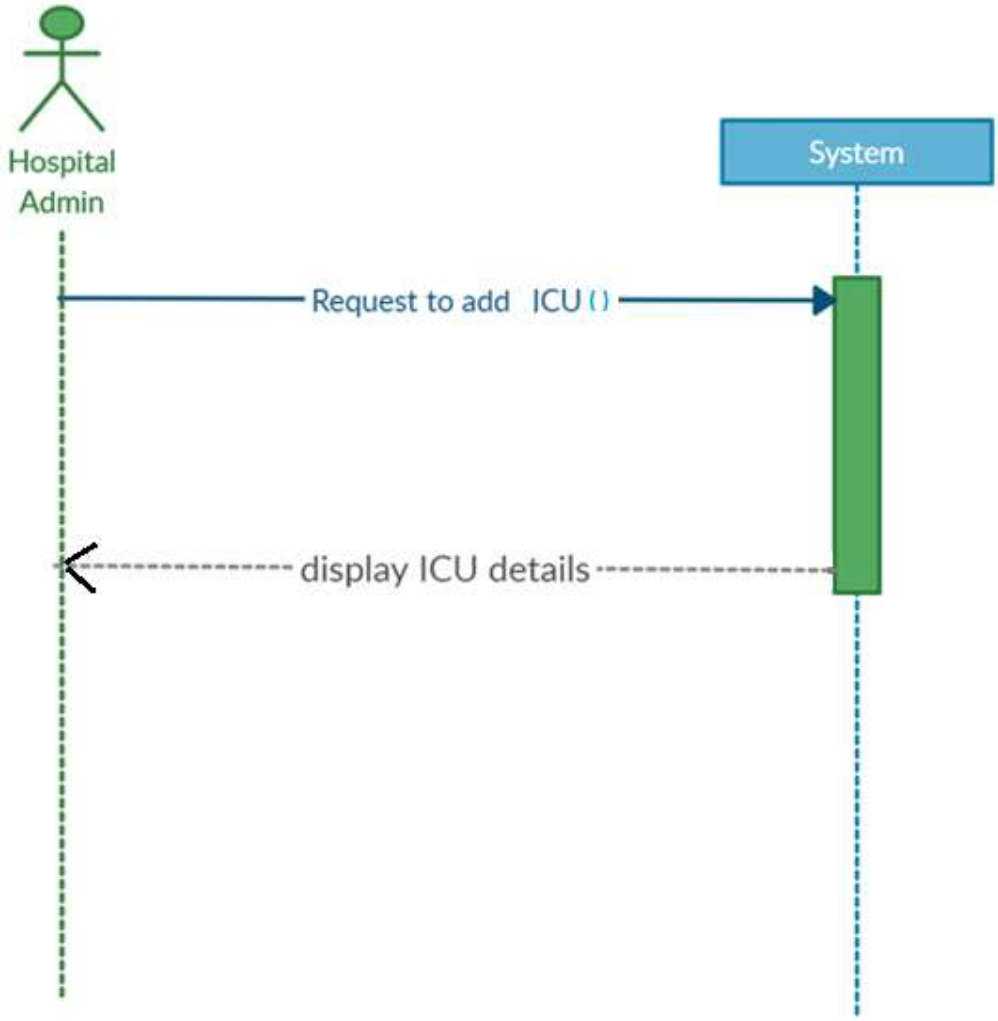

Figure 2.17: SSD (Hospital admin add ICU)

2.5.17 Hospital admin reserve ICU:

Hospital admin uses his credentials to request to reserve ICU to the system. The system will do reservation sucessfully. Hospital admin will do request to request ICU reservation in other hospital. System will reserved ICU sucessfully.

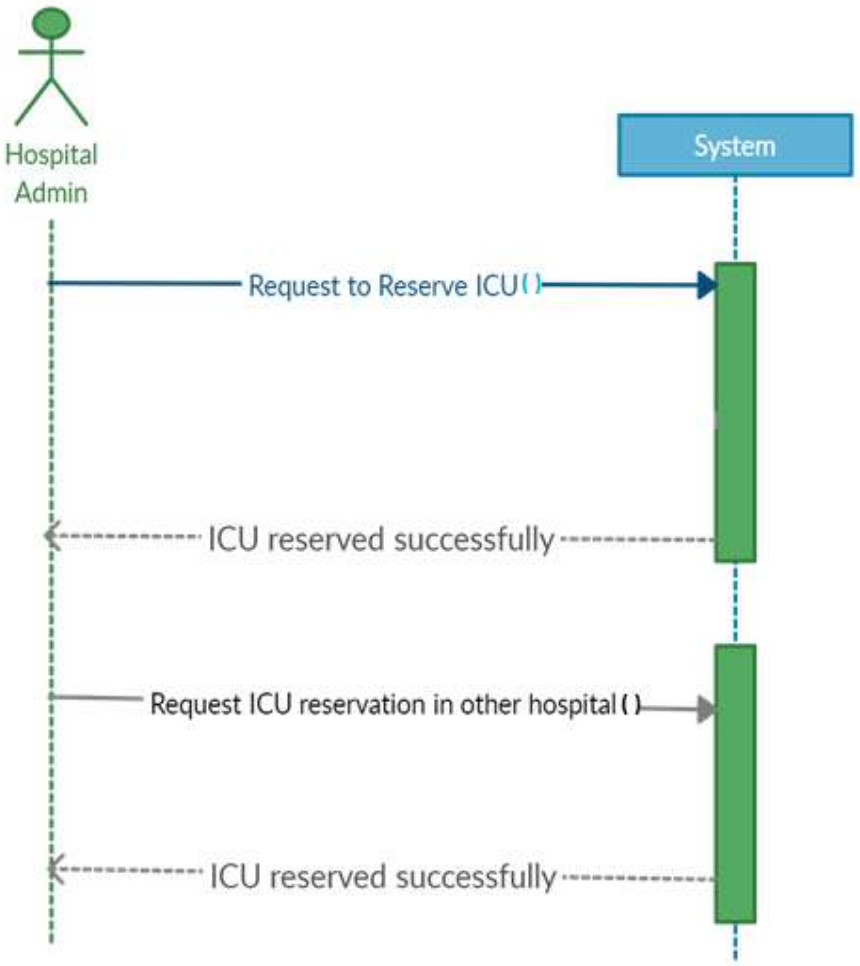

Figure 2.18: SSD (Hospital admin reserve ICU)

2.5.18 Hospital admin view all appointments:

Hospital admin can view all appointments that the patients booked in the hospital with any of the doctor. The system displays all the appointments.

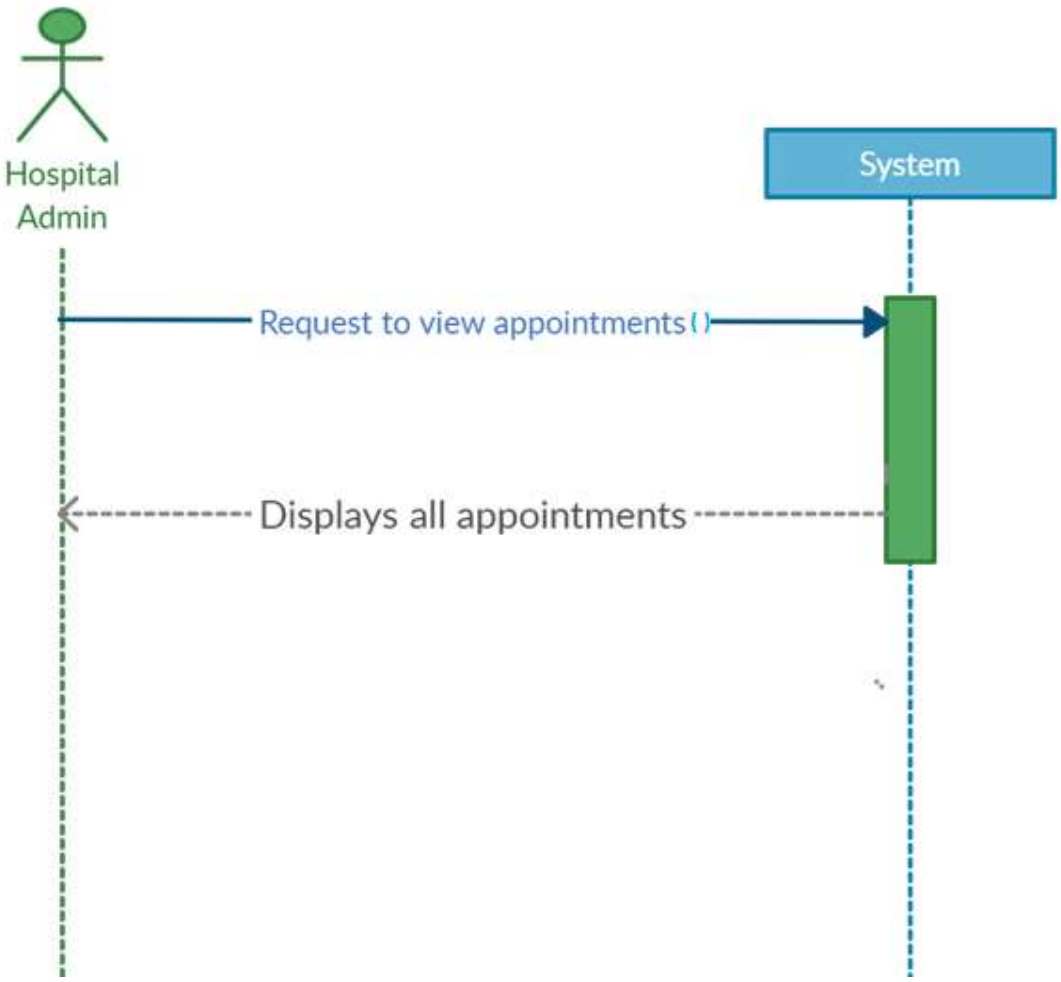

Figure 2.19: SSD (Hospital admin view all appointments)

2.5.19 System Admin sign out:

Hospital admin press the logout button and the user is logged out; thus, the session is ended.

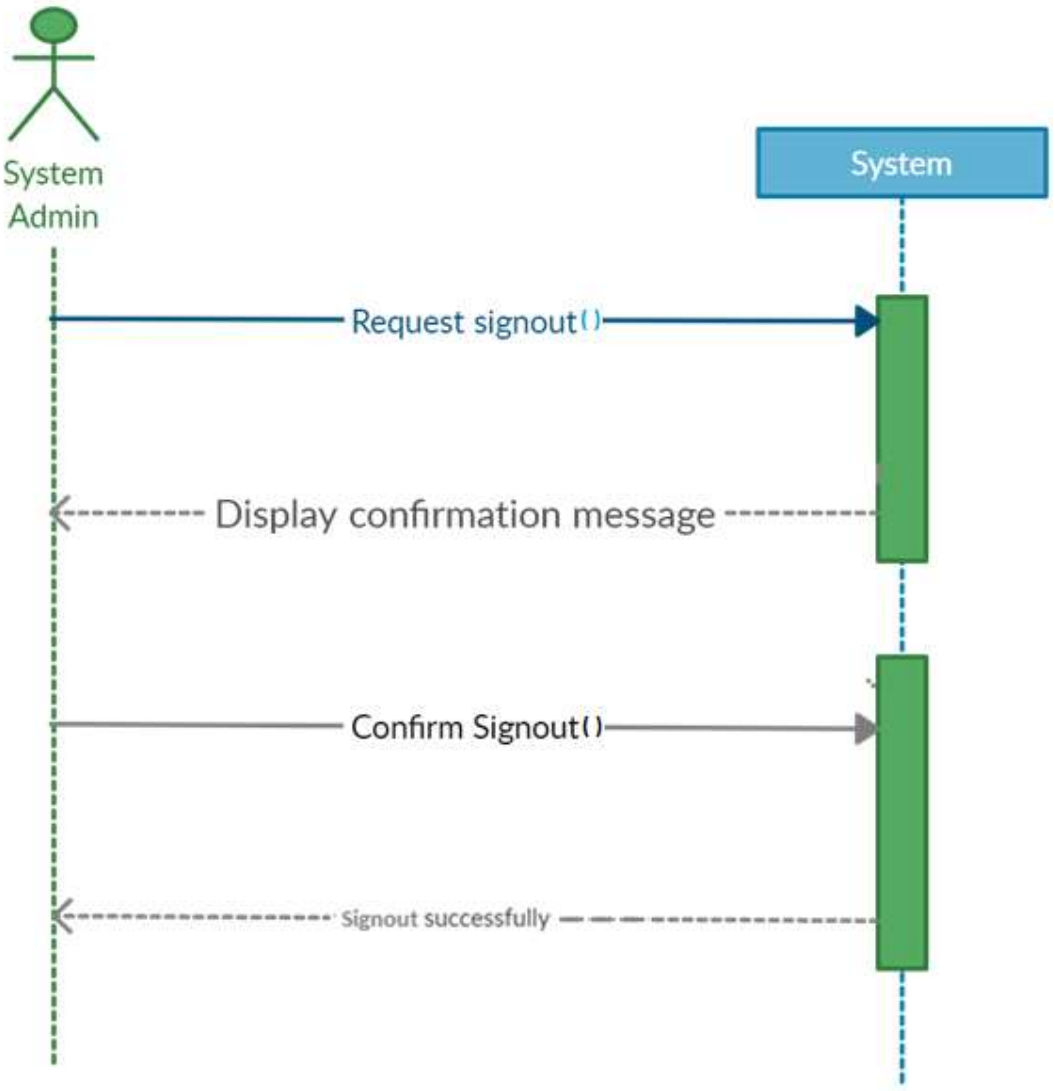

Figure 2.20: SSD (System Admin sign out)

2.6. Domain Model:

A domain model is a visual representation of conceptual classes or real-world objects in a domain of interest.

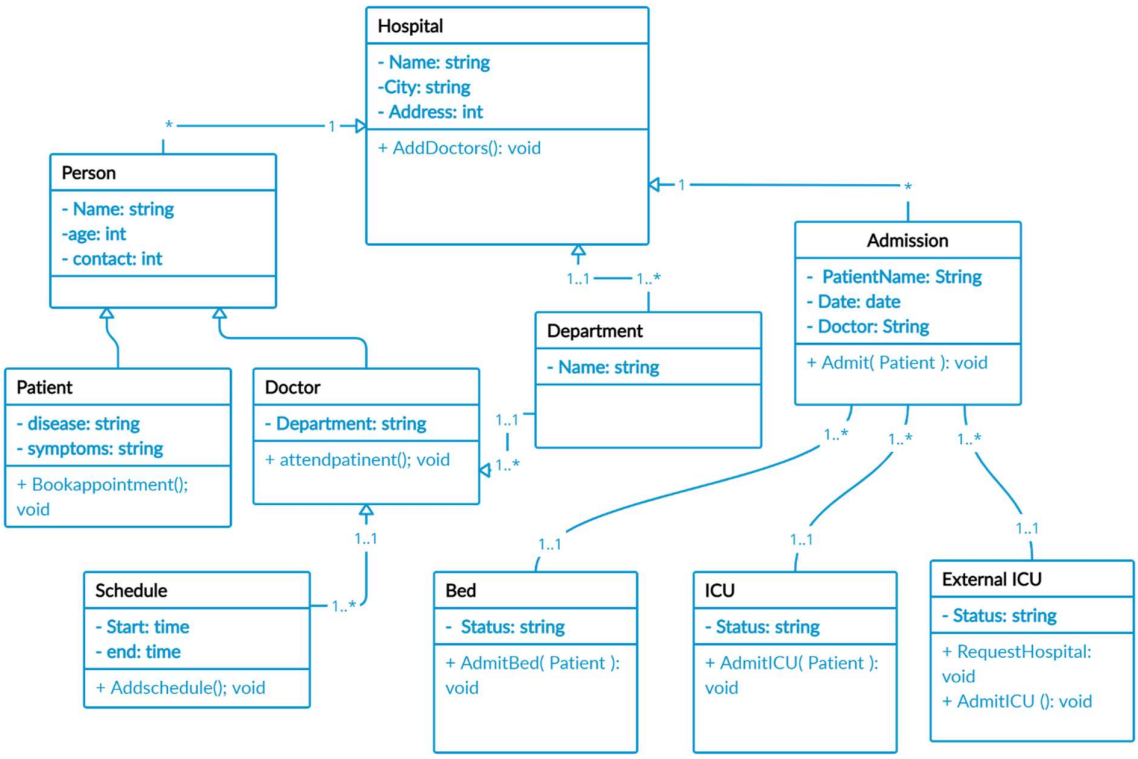

Figure 2.21: Domain Mode

Chapter 3

3. System Design

The purpose of this chapter is to provide information that is complementary to the development phase. Without an adequate design, that delivers required function as well as quality attributes, the project will fail. However, communicating architecture to its stakeholders is as important a job as creating it in the first place.

3.1. Layer Definition

Presentation, Business and Database layers are discussed below.

Table3.1: Layers Definition

Layers	Description
Presentation Layer	This layer will be used for the interaction with the user through a graphical user interface.
Business Logic Layer	This layer contains the business logic. All the constraints and majority of the functions reside under this layer.
Database Layer	This layer contains the database of the application being developed.

3.1.1 Presentation Layer:

Occupies the top level and displays information related to services available on a website. This tier communicates with other tiers by sending results to the browser and other tiers in the network.

3.1.2 Business Logic Layer:

Application Layer also called the middle tier, logic tier, business logic or logic tier, this tier is pulled from the presentation tier. It controls application functionality by performing detailed processing.

3.1.3 Database Layer:

Database layer includes database servers where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic.

3.2. Software Architecture:

Software architecture is described as the organization or structure of a system, where the system represents a collection of components that accomplish a specific function or set of functions. Below is the architecture diagram of the system:

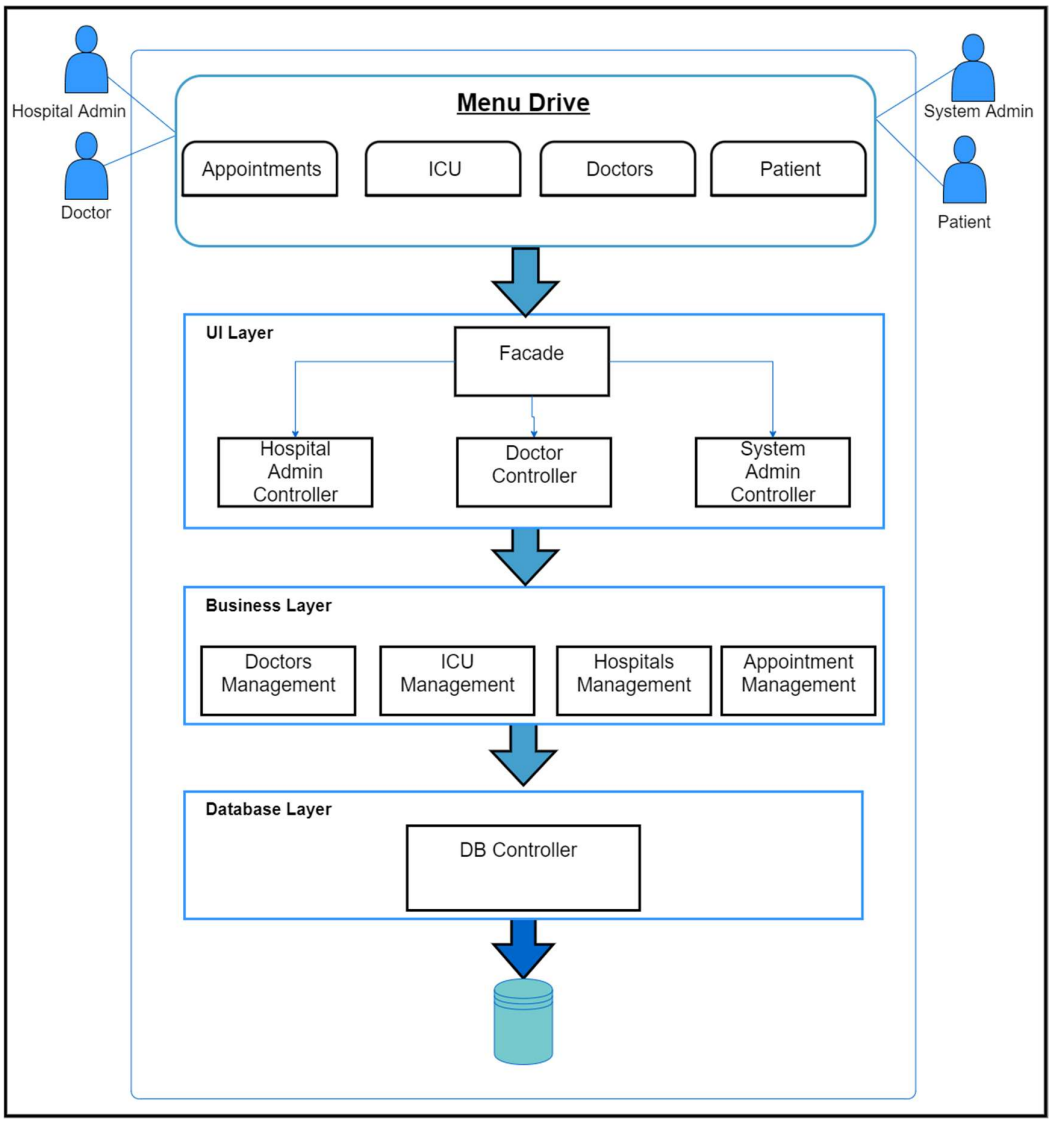

Figure 3.1: Software Architecture Diagram

3.3. Class Diagram

The class diagram describes the attributes and operations of a class and the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages:

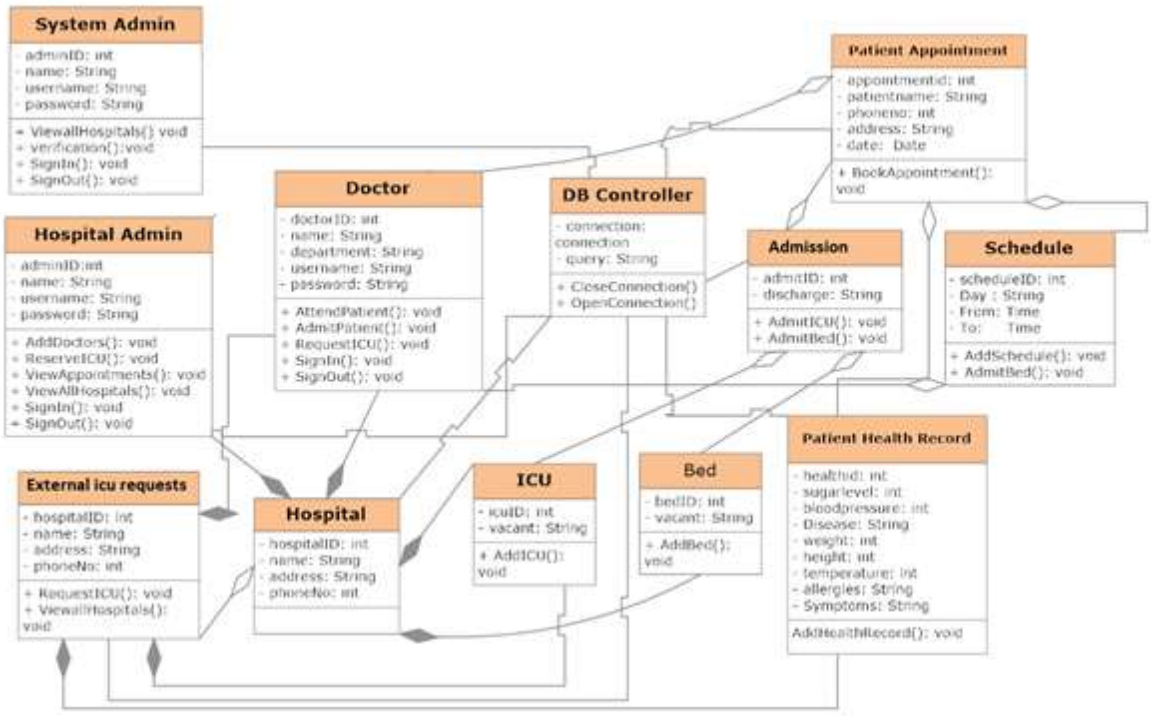

Figure 3.2: UML Class Diagram

3.4. Sequence Diagram

Sequence Diagram model the flow of logic within your system in a visual manner enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes

3.4.1 System Admin:

In this sequence diagram system admin can do request for sign in to façade. Facade forward sign in request to user controller. User controller will display sign in form to system admin. System admin providing data to façade. Facade send data to user controller. User controller send data to DB controller for data validation. DB controller validate data and send to the user controller. User controller send format response to facade. Facade display response message to system admin.

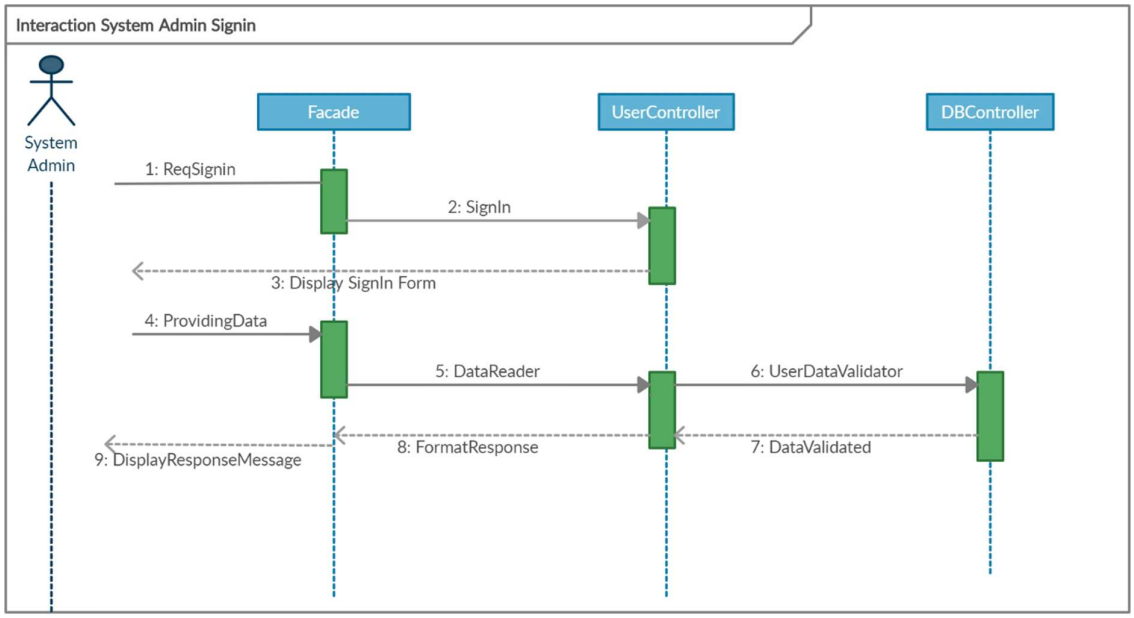

Figure 3.3: SD System Admin Sign In

3.4.2 Hospital Admin:

In below sequence diagram hospital admin can do request for sign up to façade. Façade forward request for sign up to user controller. User controller display sign up form to hospital admin. Hospital admin send user information to façade. Façade submit form to user controller. User controller write form data to DB controller. DB controller written yes and send to user controller. User controller send confirmation message to hospital admin.

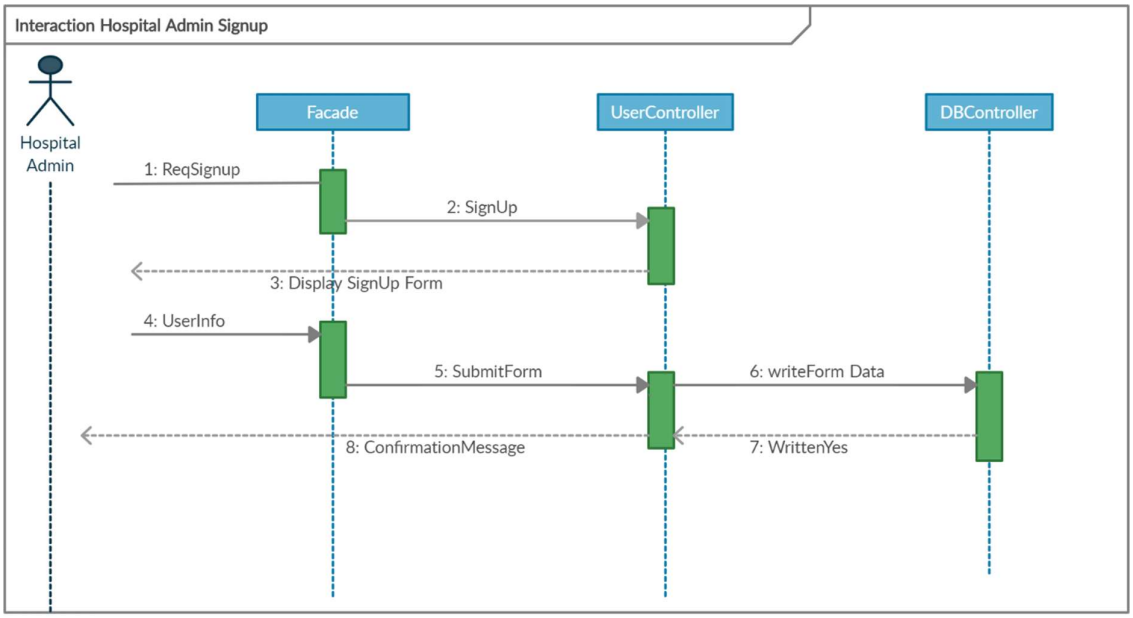

Figure 3.4: SD Hospital Admin Sign Up

In below sequence diagram hospital admin can do request for sign in to façade. Facade forward sign in request to user controller. User controller will display sign in form to hospital admin. hospital admin providing data to façade. Facade send data to user controller. User controller send data to DB controller for data validation. DB controller validate data and send to the user controller. User controller send format response to facade. Facade display response message to hospital admin.

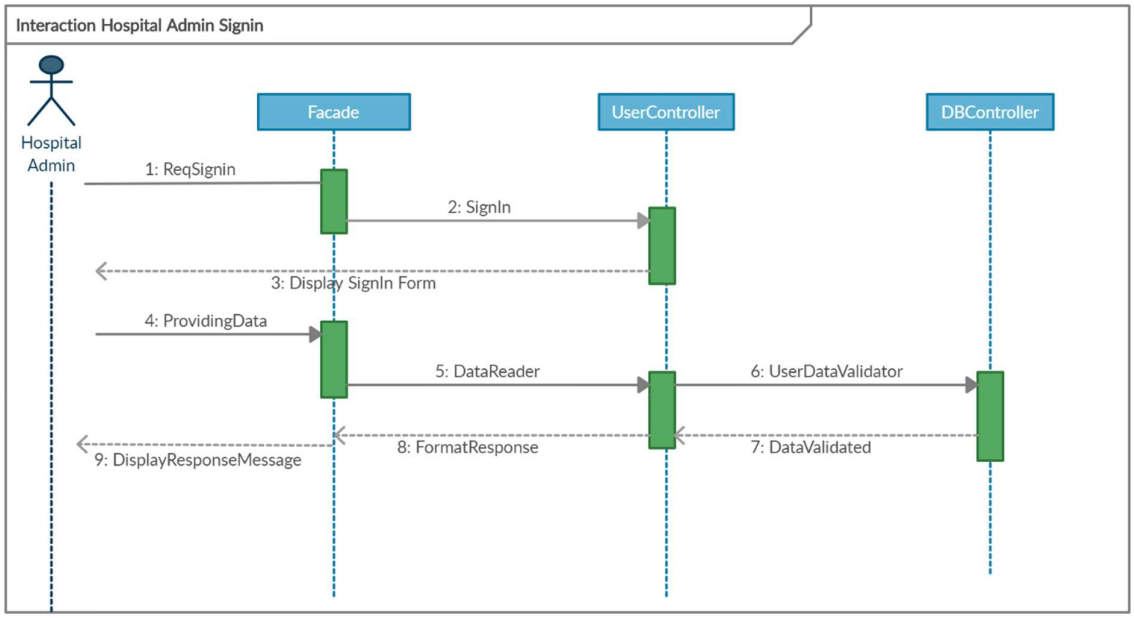

Figure 3.5: SD Hospital Admin Sign In

3.4.3 Doctor:

In this sequence diagram doctor can do request for sign in to façade. Facade forward sign in request to user controller. User controller will display sign in form to doctor. Doctor providing data to façade. Facade send data to user controller. User controller send data to DB controller for data validation. DB controller validate data and send to the user controller. User controller send format response to facade. Facade display response message to doctor.

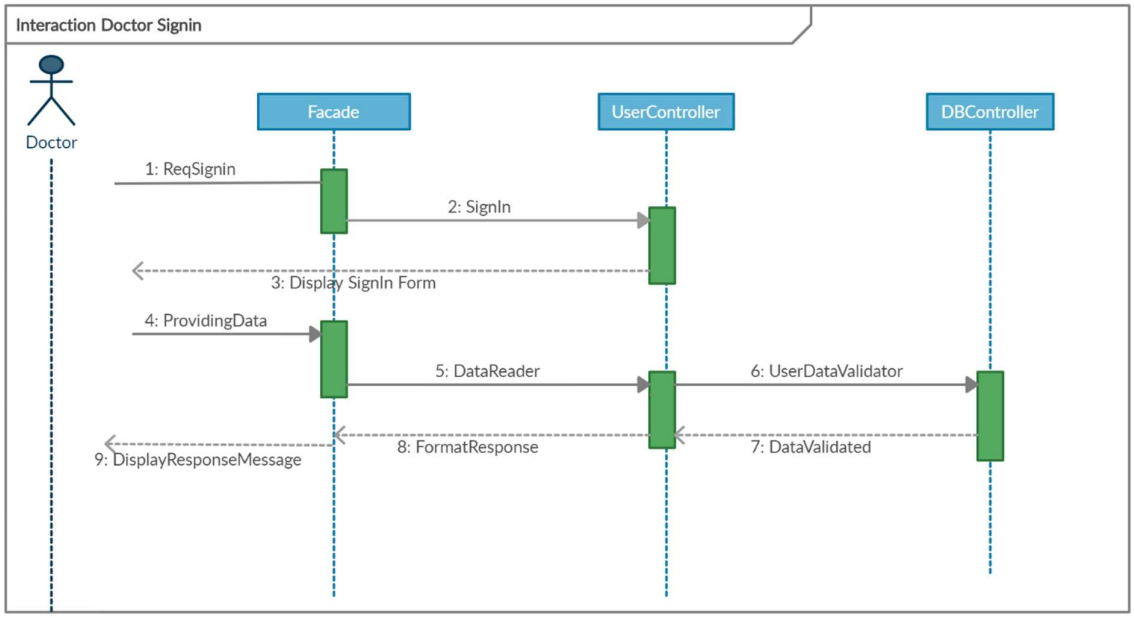

Figure 3.6: SD Doctor Sign In

3.5. Entity Relationship Diagram

Entity relationship define the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.

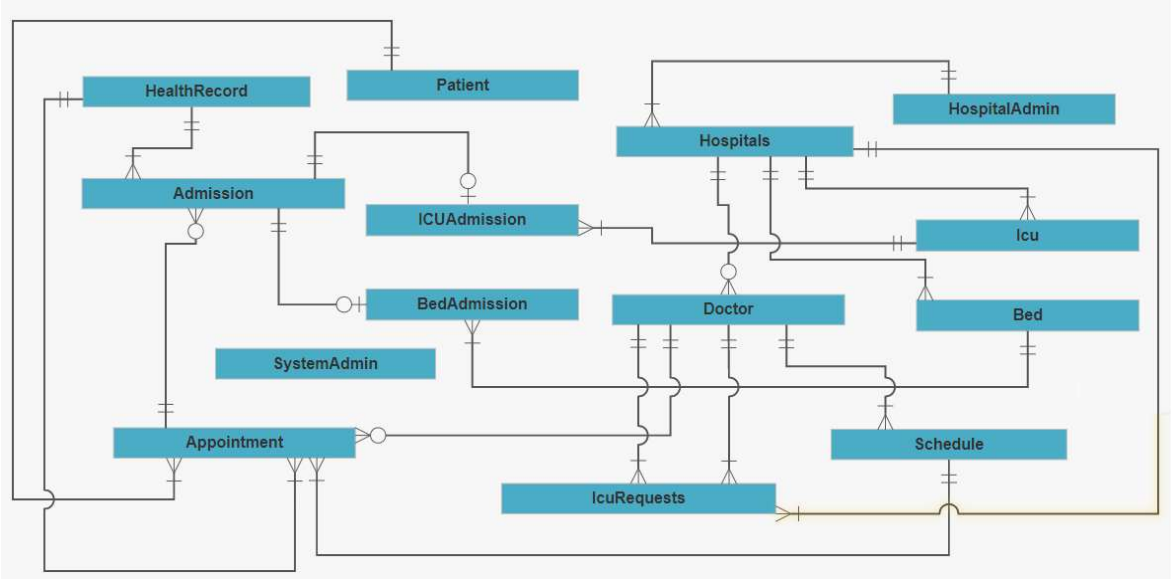

Figure 3.7: Entity Relationship Diagram

3.6. Database Schema

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

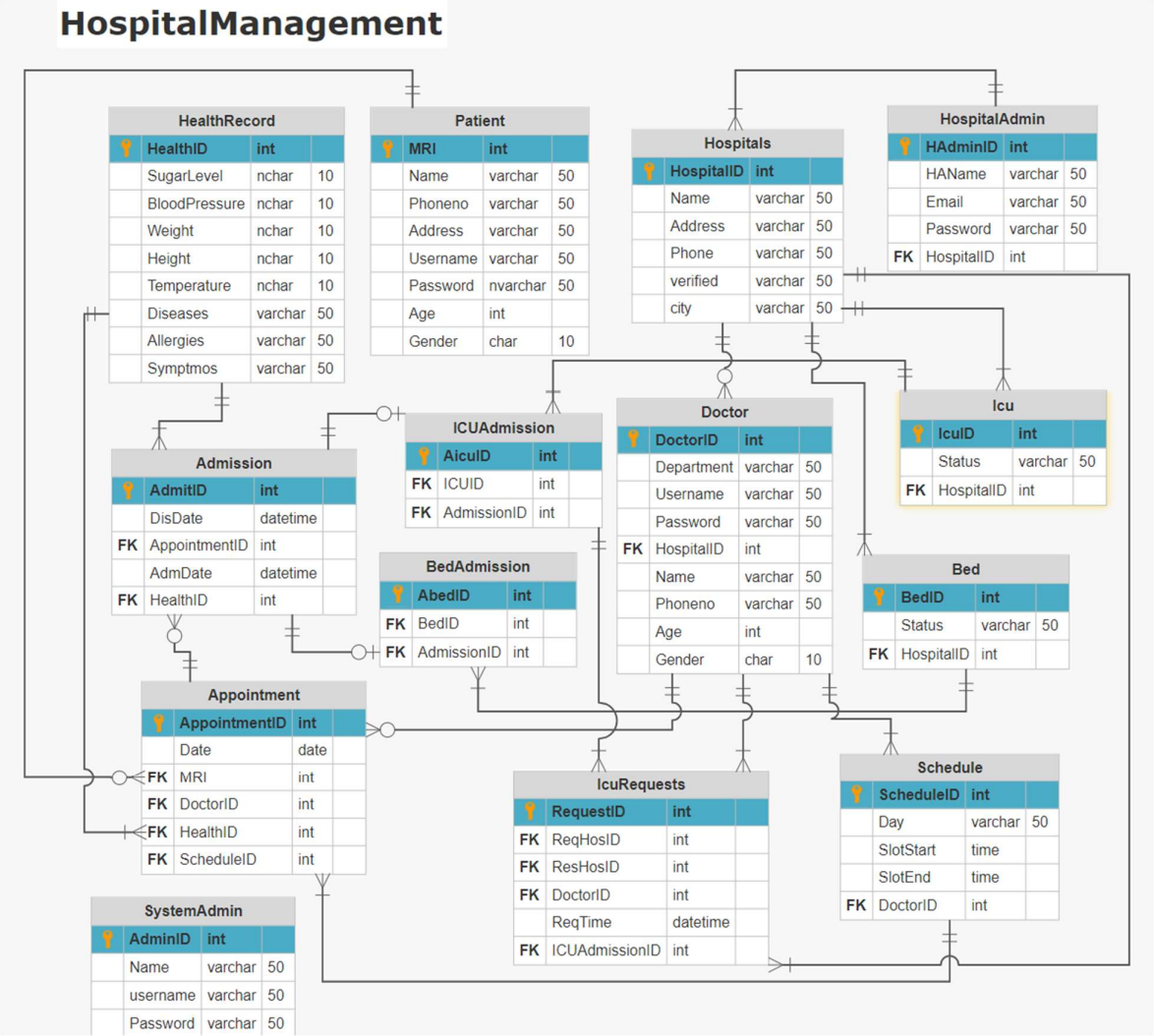

Figure 3.8: Database Schema

3.7. User Interface Design

User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

3.7.1 Login Options

Here patient, doctor, hospital admin and system admin can sign in.

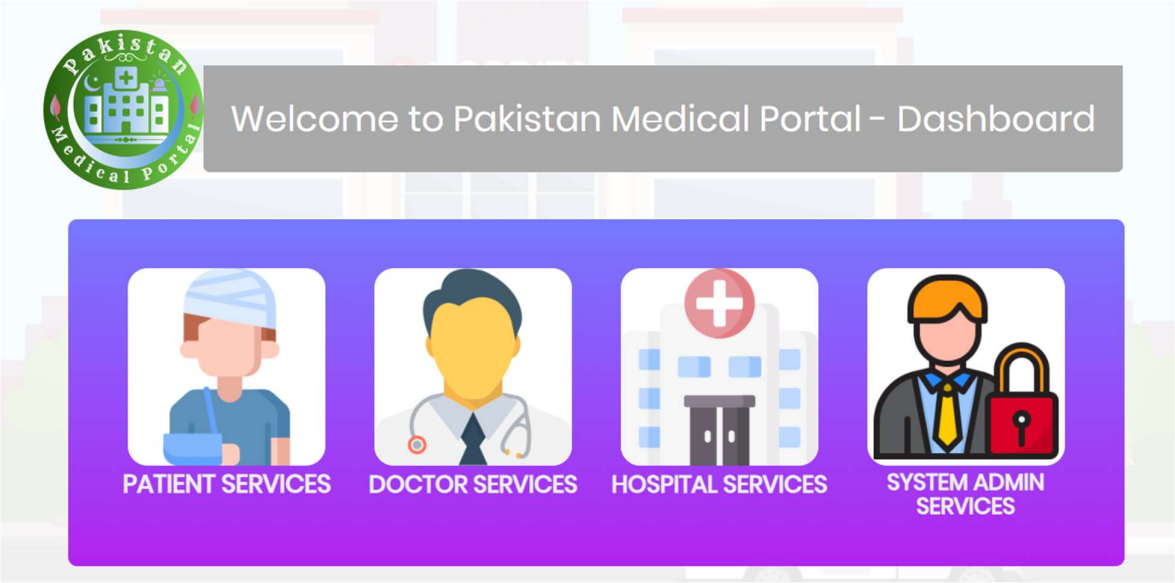

Figure 3.9: Sign in Dashboard GUI

3.7.2 Hospital Admin

Hospital admin can log in here. Admin enter your username and password

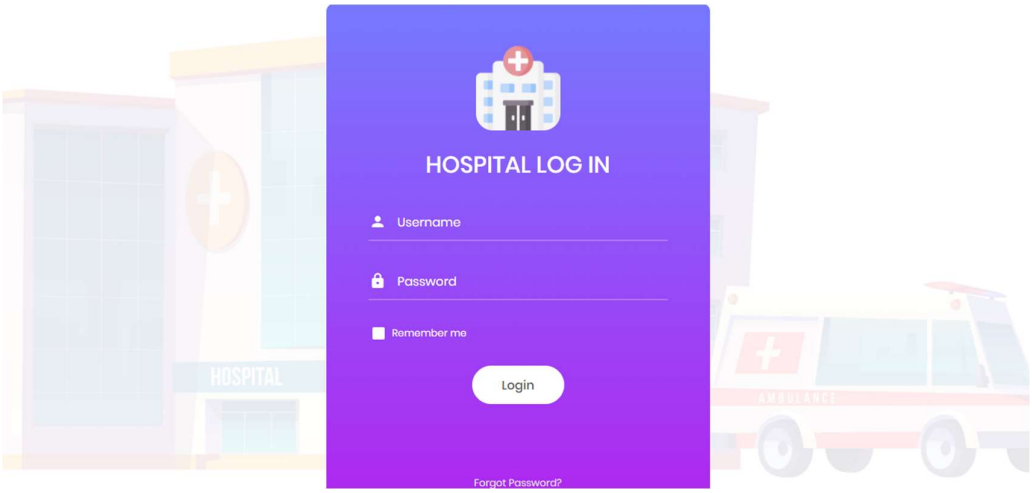

Figure 3.10: Sign in Hospital Admin GUI

3.7.3 System Admin

System admin can log in here. Admin enter your username and password.

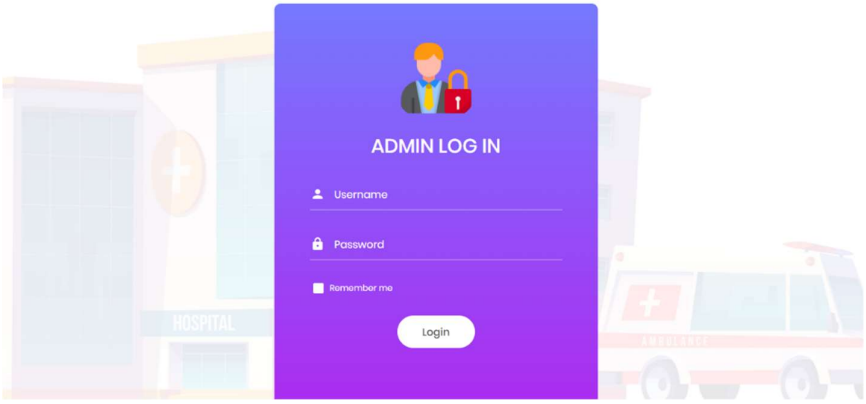

Figure 3.11: Sign in System Admin GUI

3.7.4 Doctor:

Doctor can log in here. Doctor enter your username and password.

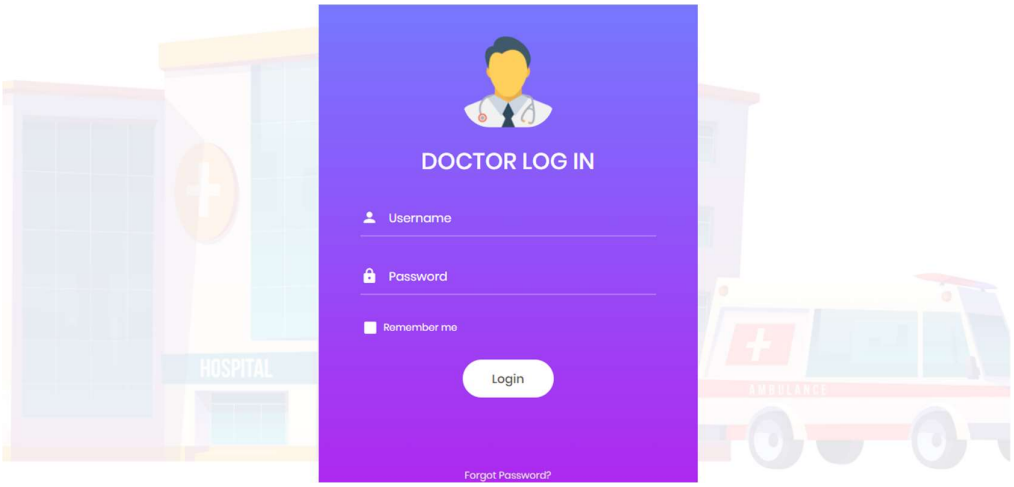

Figure 3.12: Sign in Doctor GUI

3.7.5 Patient

Here patient enter your username and password for log in.

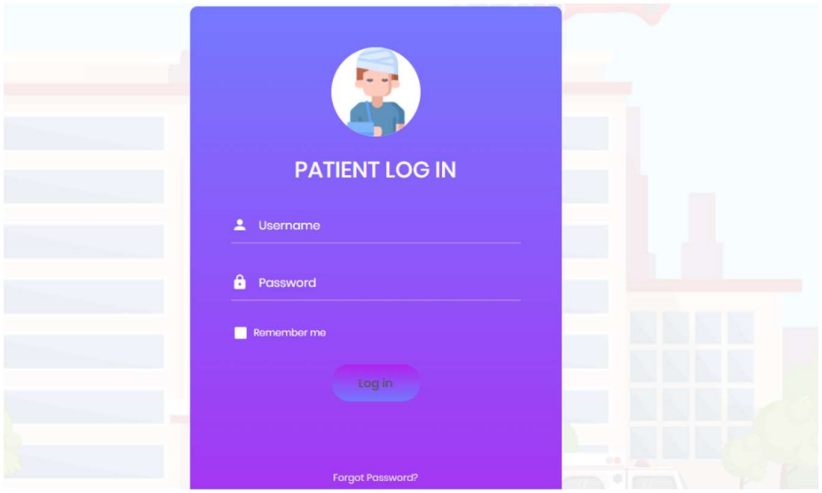

Figure 3.12: Sign in Patient GUI

Chapter 4

4. Software Development

This chapter will provide the details about the coding standards that has been adopted during implementation phase.

4.1. Coding Standards:

The adopted coding standards are discussed in the following subsections.

4.1.1 Indentation

Four spaces are used as the unit of indentation. The indentation pattern should be consistently followed throughout.

4.1.2 Declaration

One declaration per line is used to enhances the clarity of code. The order and position of declaration is as follows:

- First the static/class variables are placed in the sequence: First public class variables, protected,
- package/default level i.e., with no access modifier and then the private. As far as possible static or class fields are explicitly instantiated.
- Instance variables are placed in the sequence: First public instance variables, protected,
- package level with no access modifier and then private.
- Next the class constructors are declared.
- Class methods are grouped by functionality rather than by scope or accessibility to make reading and understanding the code easier.

4.1.3 Statement Standards

Each line contains at most one statement. While compound statements are statements that contain lists of statements enclosed in braces. The enclosed statements are indented one more level than the compound statement. The opening brace at the end of the line that begins the compound statement. The closing brace to begin a line and be indented to the beginning of the compound statement. Braces are used around all statements, even single statements, when they are part of a control structure, such as an if-else or for statement. A Boolean expression / function is compared to a Boolean constant.

4.1.4 Naming Convention

Naming conventions make programs more understandable by making them easier to read. Following conventions are followed while naming a class or a member:

- We used full English descriptors that accurately describe the variable, method or class.
- Terminology applicable to the domain is used.
- Mixed case is used to make names readable with lower case letters in general capitalizing the first letter of class names and interface names.

4.2. Development Environment

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.
Microsoft Visual Studio, like any other IDE, includes a code editor that supports syntax highlighting and code completion using IntelliSense for variables, functions, methods,

loops, and LINQ queries. IntelliSense is supported for the included languages, as well as for XML, Cascading Style Sheets, and JavaScript when developing web sites and web applications. Autocomplete suggestions appear in a modeless list box over the code editor window, in proximity of the editing cursor. In Visual Studio 2008 onwards, it can be made temporarily semi-transparent to see the code obstructed by it.

4.3. Database management System

Microsoft SQL Server:

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network.

SQL Server Management Studio is a GUI tool included with SQL Server 2005 and later for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools that work with objects and features of the server. SQL Server Management Studio replaces Enterprise Manager as the primary management interface for Microsoft SQL Server since SQL Server 2005. A version of SQL Server Management Studio is also available for SQL Server Express Edition, for which it is known as SQL Server Management Studio Express (SSMSE).

4.4. Software Description:

Main modules of our project are:

- Hospital's registration.

- Appointment booking.
- Admitting patient.
- Entertaining external and internal ICU requests
- ICU Transfer Prediction.

4.4.1 Hospitals Registration:

4.4.1.1 Input:

The hospital administration will sign up on the system and request to register their hospital. The details of the admin and hospital is provided.

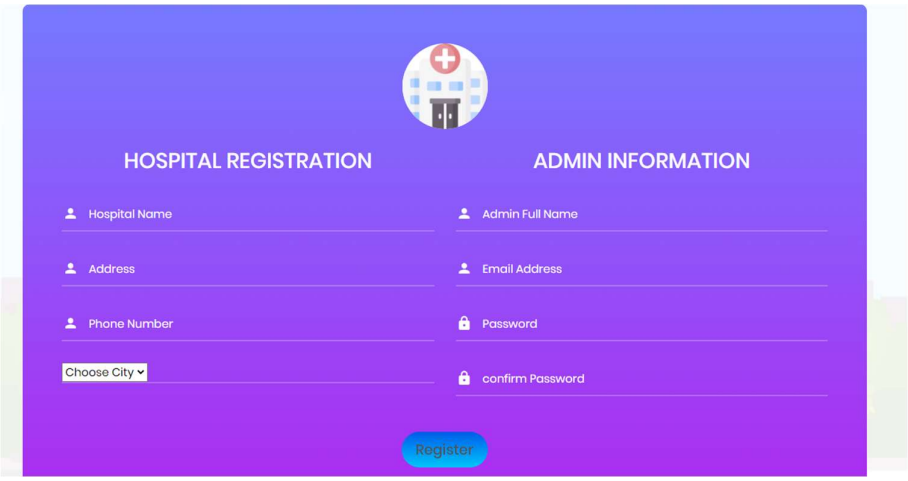

Figure 4.1: Hospital Registration

4.4.1.2 Output:

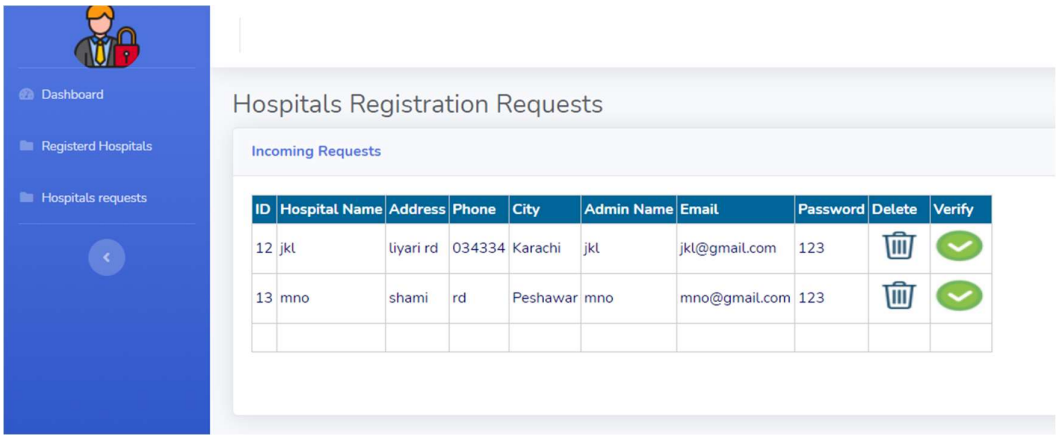

Figure 4.2: Hospital registration requests

4.4.1.3 Code:

The system administration will see the request of the specific hospital and accept or reject the request after verifying the details.

```
public partial class sysadmdashhospreq : System.Web.UI.Page
{
readonly string constring = "Data Source=DESKTOP-OB3RU32;Initial Catalog=HospitalManagement;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;Multi SubnetFailover=False";

protected void Page_Load(object sender, EventArgs e)
{

if (Session["username"] == null)
Response.Redirect("systemadminlogin.aspx");
lblUserDetails.Text = "Username : " + Session["username"];
if (!IsPostBack)
{
PopulateGridview();
}



}

protected void btnLogout_Click(object sender, EventArgs e)
{
Session.Abandon();
Response.Redirect("systemadminlogin.aspx");
}
```

```
protected void lnkSelect_Click(object sender, EventArgs e)
{
int HospitalID = Convert.ToInt32((sender as LinkButton).CommandArgument);
}


void PopulateGridview()
{
DataTable dtbl = new DataTable();


string query = "SELECT * FROM HospitalAdmin join Hospitals on HospitalAdmin.HospitalID = Hospitals.HospitalID WHERE  Hospitals.Verified = 'NO'";


using (SqlConnection sqlCon = new SqlConnection(constring))
{
sqlCon.Open();
SqlDataAdapter sqlDa = new SqlDataAdapter(query, sqlCon);
sqlDa.Fill(dtbl);
}
if (dtbl.Rows.Count > 0)
{
MyDataTable.DataSource = dtbl;
MyDataTable.DataBind();
}
else
{
dtbl.Rows.Add(dtbl.NewRow());
MyDataTable.DataSource = dtbl;
MyDataTable.DataBind();
MyDataTable.Rows[0].Cells.Clear();
MyDataTable.Rows[0].Cells.Add(new TableCell());
MyDataTable.Rows[0].Cells[0].ColumnSpan = dtbl.Columns.Count;
MyDataTable.Rows[0].Cells[0].Text = "No Hospitals Requests ..!";
MyDataTable.Rows[0].Cells[0].HorizontalAlign = HorizontalAlign.Center;
}



}


protected void MyDataTable_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
try
{
using (SqlConnection sqlCon = new SqlConnection(constring))
{
sqlCon.Open();
```

```
string query = "UPDATE  Hospitals set Hospitals.Verified='YES' from HospitalAdmin join Hospitals on HospitalAdmin.HospitalID = Hospitals.HospitalID  WHERE  Hospitals.Verified = 'NO' and Hospitals.HospitalID=@Hid";
SqlCommand sqlCmd = new SqlCommand(query, sqlCon);
sqlCmd.Parameters.AddWithValue("@Hid",
Convert.ToInt32(MyDataTable.DataKeys[e.RowIndex].Value.ToString()));

sqlCmd.ExecuteNonQuery();

PopulateGridview();
lblSuccessMessage.Text = "Hospital  Successfully  Verified";
lblErrorMessage.Text = "";
}
}
catch (Exception ex)
{
lblSuccessMessage.Text = "";
lblErrorMessage.Text = ex.Message;
}
}

protected void MyDataTable_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
try
{
using (SqlConnection sqlCon = new SqlConnection(constring))
{
sqlCon.Open();
string query = "Delete  Hospitals  from Hospitals inner join HospitalAdmin on HospitalAdmin.HospitalID = Hospitals.HospitalID  WHERE  Hospitals.Verified = 'NO' and Hospitals.HospitalID=@Hid";
SqlCommand sqlCmd = new SqlCommand(query, sqlCon);
sqlCmd.Parameters.AddWithValue("@Hid",
Convert.ToInt32(MyDataTable.DataKeys[e.RowIndex].Value.ToString()));
sqlCmd.ExecuteNonQuery();
PopulateGridview();
lblSuccessMessage.Text = "Hospital Request Denied ";
lblErrorMessage.Text = "";
}
}
catch (Exception ex)
{
lblSuccessMessage.Text = "";
lblErrorMessage.Text = ex.Message;
}
}
```

4.4.2 Appointment booking:

4.4.2.1 Input:

The patient searches for doctors in a certain hospital and chooses the desired appointment.

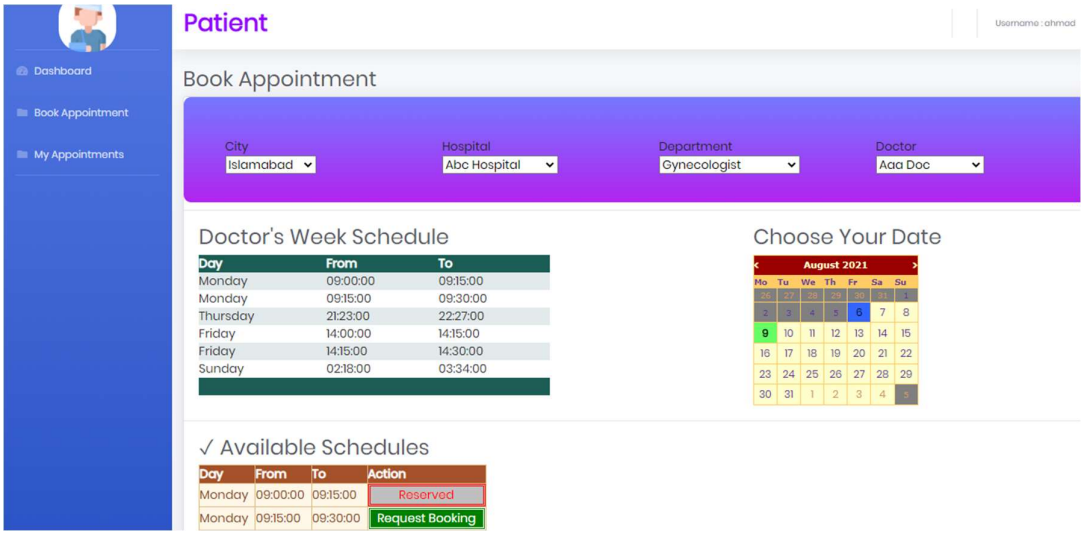

Figure 4.3: Appointment Booking

4.4.2.2 Output:

The selected appointment is sent to the specific hospital and the concerned doctor, specifying the chosen date and time.

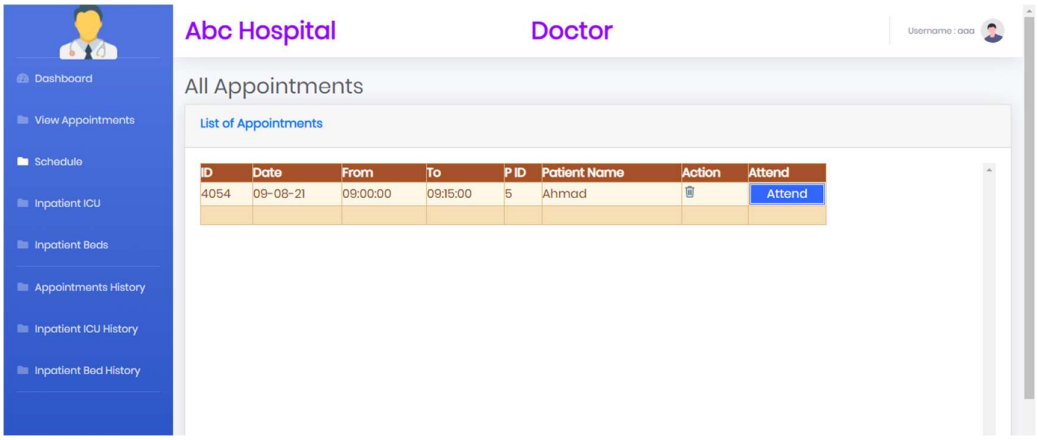

Figure 4.4: Appointment List

4.4.2.3 Code:

public partial class pbookappointment : System.Web.UI.Page

```
{
readonly string constring = "Data Source=DESKTOP-OB3RU32;Initial Catalog=HospitalManagement;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;Multi SubnetFailover=False";
string query = "select  distinct city from hospitals where verified='YES' ";


protected void Page_Load(object sender, EventArgs e)
{
if (Session["username"] == null)
Response.Redirect("patientsignin.aspx");
lblUserDetails.Text = "Username : " + Session["username"];


if (!IsPostBack)
{
Dayschedule.Visible = false;
MyDataTable.Visible = false;
Mycalendar.Visible = false;
using (SqlConnection sqlCon = new SqlConnection(constring))
{

SqlCommand cmd = new SqlCommand(query, sqlCon);
sqlCon.Open();
SqlDataReader rdr = cmd.ExecuteReader();
DropDownCity.DataTextField = "City";
DropDownCity.DataSource = rdr;
DropDownCity.DataBind();
DropDownCity.Items.Insert(0, new ListItem("Choose City"));
DropDownCity.SelectedIndex = 0;
}


}
}

protected void btnLogout_Click(object sender, EventArgs e)
{
Session.Abandon();
Response.Redirect("patientsignin.aspx");
}

protected void DropDownCity_TextChanged(object sender, EventArgs e)
{
using (SqlConnection sqlCon = new SqlConnection(constring))
{

//Displaying Hospitals in dropdown
query = "Select  Name from hospitals where verified='yes' and city=@city";
SqlCommand cmd = new SqlCommand(query, sqlCon);
sqlCon.Open();
```

```
cmd.Parameters.AddWithValue("@city", DropDownCity.SelectedValue);
SqlDataReader rdr = cmd.ExecuteReader();
DropDownHospital.DataTextField = "Name";
DropDownHospital.DataSource = rdr;
DropDownHospital.DataBind();

DropDownHospital.Items.Insert(0, new ListItem("Choose Hospital"));
DropDownHospital.SelectedIndex = 0;

DropDownDepartment.Items.Clear();
DropDownDepartment.Items.Insert(0, new ListItem("Choose Department"));
DropDownDepartment.SelectedIndex = 0;

DropDownDoctor.Items.Clear();
DropDownDoctor.Items.Insert(0, new ListItem("Choose Doctor"));
DropDownDoctor.SelectedIndex = 0;

}
}

protected void DropDownHospital_TextChanged(object sender, EventArgs e)
{
using (SqlConnection sqlCon = new SqlConnection(constring))
{
//Displaying Departments  in dropdown
query = "Select distinct doctor.department from doctor join hospitals on
hospitals.HospitalID=Doctor.HospitalID where Hospitals.verified='yes' and
Hospitals.Name=@HName";
SqlCommand cmd = new SqlCommand(query, sqlCon);
sqlCon.Open();
cmd.Parameters.AddWithValue("@HName", DropDownHospital.SelectedValue);
SqlDataReader rdr = cmd.ExecuteReader();
DropDownDepartment.DataTextField = "Department";
DropDownDepartment.DataSource = rdr;
DropDownDepartment.DataBind();
DropDownDepartment.Items.Insert(0, new ListItem("Choose Department"));
DropDownDepartment.SelectedIndex = 0;

DropDownDoctor.Items.Clear();
DropDownDoctor.Items.Insert(0, new ListItem("Choose Doctor"));
DropDownDoctor.SelectedIndex = 0;


}
}

protected void DropDownDepartment_SelectedIndexChanged(object sender, EventArgs e)
{

using (SqlConnection sqlCon = new SqlConnection(constring))
{
//Displaying Doctors  in dropdown
```

```
query = "Select doctor.doctorid , doctor.Name from doctor join hospitals on hospitals.HospitalID=Doctor.HospitalID where Hospitals.verified='yes'and Doctor.Department=@Dep and Hospitals.Name=@HName";
SqlCommand cmd = new SqlCommand(query, sqlCon);
sqlCon.Open();
cmd.Parameters.AddWithValue("@Dep", DropDownDepartment.SelectedValue);
cmd.Parameters.AddWithValue("@HName", DropDownHospital.SelectedValue);
SqlDataReader rdr = cmd.ExecuteReader();
DropDownDoctor.DataTextField = "Name";
DropDownDoctor.DataValueField = "Doctorid";
DropDownDoctor.DataSource = rdr;
DropDownDoctor.DataBind();
DropDownDoctor.Items.Insert(0, new ListItem("Choose Doctor"));
DropDownDoctor.SelectedIndex = 0;

}
}

protected void DropDownDoctor_SelectedIndexChanged(object sender, EventArgs e)
{


if (DropDownDoctor.SelectedItem.Equals("Choose Doctor"))
{
Dayschedule.DataSource = null;
Dayschedule.DataBind();
MyDataTable.DataSource = null;
MyDataTable.DataBind();
Dayschedule.Visible = false;
Mycalendar.Visible = false;
}
else
{
MyDataTable.Visible = true;
filltable();
Mycalendar.Enabled = true;
Mycalendar.Visible = true;
}

}

void filltable()
{
DataTable dtbl = new DataTable();

string query = "select Schedule.ScheduleID,Schedule.Day,schedule.SlotStart,schedule.SlotEnd FROM schedule join doctor on schedule.doctorid =doctor.doctorid where schedule.doctorid=@Did and schedule.doctorid is not null order by CASE WHEN Day = 'Sunday' THEN 7 WHEN Day = 'Monday' THEN 1 WHEN Day = 'Tuesday' THEN 2 WHEN Day = 'Wednesday' THEN 3 WHEN Day = 'Thursday' THEN 4 WHEN Day = 'Friday' THEN 5 WHEN Day = 'Saturday' THEN 6 END ASC , SlotStart asc";
```

```
using (SqlConnection sqlCon = new SqlConnection(constring))
{
sqlCon.Open();


SqlCommand cmd = new SqlCommand(query, sqlCon);

cmd.Parameters.AddWithValue("@Did", DropDownDoctor.SelectedValue);
SqlDataAdapter sqlDa = new SqlDataAdapter(cmd);


sqlDa.Fill(dtbl);
}
if (dtbl.Rows.Count > 0)
{
MyDataTable.DataSource = dtbl;
MyDataTable.DataBind();
}
else
{
dtbl.Rows.Add(dtbl.NewRow());
MyDataTable.DataSource = dtbl;
MyDataTable.DataBind();
MyDataTable.Rows[0].Cells.Clear();
MyDataTable.Rows[0].Cells.Add(new TableCell());
MyDataTable.Rows[0].Cells[0].ColumnSpan = dtbl.Columns.Count;
MyDataTable.Rows[0].Cells[0].Text = "No Schedules Available   ..!";
MyDataTable.Rows[0].Cells[0].HorizontalAlign = HorizontalAlign.Center;
}


}

protected void Cal_DayRender(object sender, DayRenderEventArgs e)
{
if(e.Day.Date.CompareTo(DateTime.Today)<0)
{
e.Day.IsSelectable = false;
e.Cell.BackColor = System.Drawing.Color.Gray;
}
if (e.Day.Date.CompareTo(DateTime.Today.AddDays(6)) > 0)
{
e.Day.IsSelectable = false;
e.Cell.BackColor = System.Drawing.Color.Gray;
}

}

protected void Mycalendar_SelectionChanged(object sender, EventArgs e)
{
Label3.Visible = false;
if (DropDownDoctor.SelectedItem.Equals("Choose Doctor"))
```

```
{
Label2.Text = "Choose Doctor";
Label2.Visible = true;
}
else
{
Dayschedule.Visible = true;
MyDataTable.Visible = true;
Mycalendar.Visible = true;
FillDayschedule();

}
}


void FillDayschedule()
{
DataTable dtbl = new DataTable();
//  string query = "select distinct schedule.scheduleid,  schedule.day, schedule.slotstart, schedule.slotend from  doctor join schedule  on Schedule.DoctorID =doctor.doctorid  full outer join  Appointment  on  Appointment.ScheduleID  =  Schedule.ScheduleID  where Doctor.DoctorID=@Did  and  Schedule.Day=@Day  and  (Appointment.Date!=@Date  or Appointment.Date is NULL)  order by  SlotStart asc  ";
string query = "select  schedule.scheduleid,  schedule.day, schedule.slotstart, schedule.slotend from  doctor  join  schedule  on  Schedule.DoctorID =doctor.doctorid  full  outer  join Appointment  on  Appointment.ScheduleID  =  Schedule.ScheduleID  where Doctor.DoctorID=@Did  and  Schedule.Day=@Day  and  (Appointment.Date <@Date  or appointment.date is null  ) order by  SlotStart asc  ";
using (SqlConnection sqlCon = new SqlConnection(constring))
{
sqlCon.Open();


SqlCommand cmd = new SqlCommand(query, sqlCon);

cmd.Parameters.AddWithValue("@Date", Mycalendar.SelectedDate);
cmd.Parameters.AddWithValue("@Did", DropDownDoctor.SelectedValue);
cmd.Parameters.AddWithValue("@Day",  Enum.GetName(typeof(DayOfWeek), Mycalendar.SelectedDate.DayOfWeek));
SqlDataAdapter sqlDa = new SqlDataAdapter(cmd);


sqlDa.Fill(dtbl);
}
if (dtbl.Rows.Count > 0)
{
Dayschedule.DataSource = dtbl;
Dayschedule.DataBind();
}
else
{
dtbl.Rows.Add(dtbl.NewRow());
```

```
Dayschedule.DataSource = dtbl;
Dayschedule.DataBind();
Dayschedule.Rows[0].Cells.Clear();
Dayschedule.Rows[0].Cells.Add(new TableCell());
Dayschedule.Rows[0].Cells[0].ColumnSpan = dtbl.Columns.Count;
Dayschedule.Rows[0].Cells[0].Text = "No Schedules Available   ..!";
Dayschedule.Rows[0].Cells[0].HorizontalAlign = HorizontalAlign.Center;
}


}
protected void lnkSelect_Click(object sender, EventArgs e)
{
int ScheduleID = Convert.ToInt32((sender as LinkButton).CommandArgument);
}

protected void Dayschedule_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
try
{
using (SqlConnection sqlCon = new SqlConnection(constring))
{


sqlCon.Open();
SqlCommand sqlCmd = new SqlCommand("AddApp", sqlCon);
sqlCmd.CommandType = CommandType.StoredProcedure;

sqlCmd.Parameters.AddWithValue("@Date",Mycalendar.SelectedDate );
sqlCmd.Parameters.AddWithValue("@DID", DropDownDoctor.SelectedValue);
sqlCmd.Parameters.Add(new SqlParameter("@PUser", Session["username"].ToString()));
sqlCmd.Parameters.AddWithValue("@SID",
Convert.ToInt32(Dayschedule.DataKeys[e.RowIndex].Value.ToString()));

sqlCmd.ExecuteNonQuery();


Label3.Text = "Appointment booked successfully";
Label3.Visible = true;
FillDayschedule();
}
}
catch (Exception ex)
{
lblSuccessMessage.Text = "";
lblErrorMessage.Text = ex.Message;}
```

4.4.3 Admitting patient:

4.4.3.1 Input:

After the checkup of the patient, the doctor can decide to admit the patient whether in ICU or on bed.

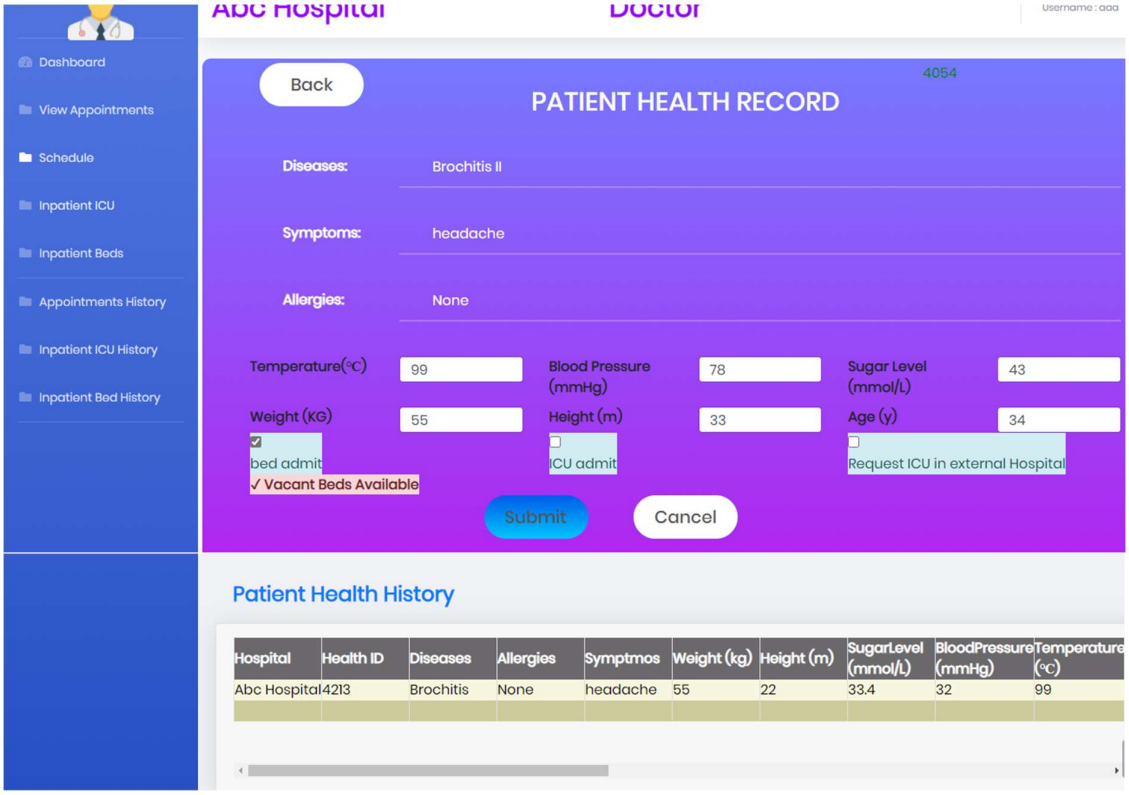

Figure 4.5: Attending Patient

4.4.3.2 Output:

ICU or bed is successfully assigned to a patient.

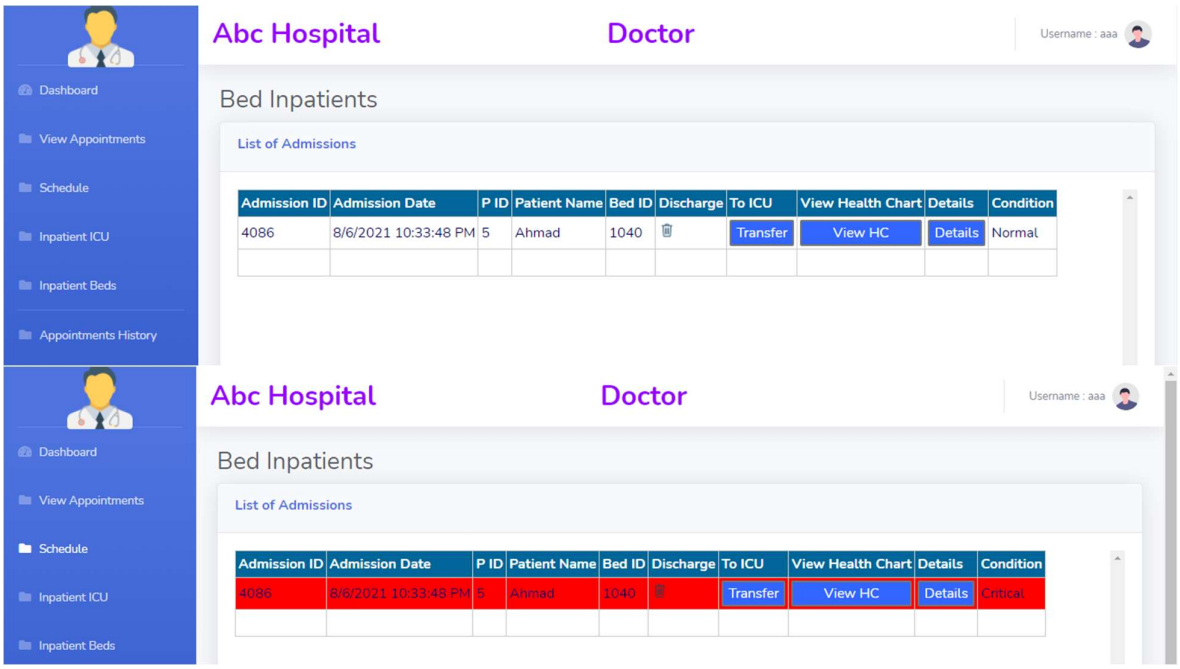

Figure 4.6: Admission of patient

4.4.3.3 Code:

```
protected void healthrecordsubmit_Click(object sender, EventArgs e)
{

if(appid.Text!="")
{
try
{
using (SqlConnection sqlCon = new SqlConnection(constring))
{
if (Session["check"].Equals(1))
{

Session["icuid"] = -1;
}
else if (Session["check"].Equals(0))
{
Session["bedid"] = -1;

}
else
```

```
{
Session["bedid"] = -1;
Session["icuid"] = -1;
}

sqlCon.Open();
SqlCommand sqlCmd = new SqlCommand("Addhealthrec", sqlCon);
sqlCmd.CommandType = CommandType.StoredProcedure;

sqlCmd.Parameters.AddWithValue("@AID", appid.Text);
sqlCmd.Parameters.AddWithValue("@Sug", SugarLevel.Text);
sqlCmd.Parameters.AddWithValue("@BP", Bloodpressure.Text);
sqlCmd.Parameters.AddWithValue("@D", Disease.Text);
sqlCmd.Parameters.AddWithValue("@w", Weight.Text);
sqlCmd.Parameters.AddWithValue("@h", Height.Text);
sqlCmd.Parameters.AddWithValue("@t", Temperature.Text);
sqlCmd.Parameters.AddWithValue("@a", Allergies.Text);
sqlCmd.Parameters.AddWithValue("@s", Symptoms.Text);

sqlCmd.Parameters.AddWithValue("@check", Session["check"]);
sqlCmd.Parameters.AddWithValue("@icuid", Session["icuid"]);
sqlCmd.Parameters.AddWithValue("@bedid", Session["bedid"] );

sqlCmd.Parameters.AddWithValue ("@Duser", Session["username"].ToString());


sqlCmd.ExecuteNonQuery();

lblSuccessMessage.Text = "Health record entered successfully";
lblSuccessMessage.Visible = true;


if (Session["check"].Equals(1) )
lblSuccessMessage.Text= "Health record entered successfully and Bed is allocated to patient
successfully";
else if (Session["check"].Equals(0))
lblSuccessMessage.Text = "Health record entered successfully and ICU is allocated to patient
successfully";
else if (Session["check"].Equals(2))
lblSuccessMessage.Text = "Health record entered and request of external hospital's ICU
allocation sent to hospital Admin  successfully";


bedlbl.Text=iculbl.Text= MRI.Text = appid.Text = Date.Text = SugarLevel.Text =
Disease.Text = Bloodpressure.Text = Weight.Text = Height.Text = Symptoms.Text =
Allergies.Text = "" ;
healthrecord.Visible = false;
PopulateGridview();

}
}
catch (Exception ex)
```

```
{
lblSuccessMessage.Text = "";
lblErrorMessage.Text = ex.Message;
}
}
else
{
lblErrorMessage.Text = "Enter Data";
}
}

protected void bedcb_CheckedChanged(object sender, EventArgs e)
{
if (bedcb.Checked == true)
{
icucb.Checked = false;
reqcb.Checked = false;
using (SqlConnection sqlCon = new SqlConnection(constring))
{
//Displaying Doctors  in dropdown
query = "SELECT top(1) bed.bedid FROM Bed join Hospitals  on Bed.HospitalID = Hospitals.HospitalID join Doctor on Doctor.HospitalID=Hospitals.HospitalID where Doctor.Username=@Duser and bed.status='Vacant' order by bed.BedID";
SqlCommand cmd = new SqlCommand(query, sqlCon);
sqlCon.Open();
cmd.Parameters.AddWithValue("@Duser", Session["username"].ToString());

SqlDataReader rdr = cmd.ExecuteReader();

// int bedid = rdr.;
if (rdr.HasRows)
{
using (SqlConnection sqlCon2 = new SqlConnection(constring))
{
query = "SELECT top(1) bed.bedid FROM Bed join Hospitals  on Bed.HospitalID = Hospitals.HospitalID join Doctor on Doctor.HospitalID=Hospitals.HospitalID where Doctor.Username=@Duser and bed.status='Vacant' order by bed.BedID";
SqlCommand cmd2 = new SqlCommand(query, sqlCon2);
sqlCon2.Open();
cmd2.Parameters.AddWithValue("@Duser", Session["username"].ToString());

Session["bedid"] = (int)cmd2.ExecuteScalar();


bedlbl.Text = " Vacant Beds Available ";

Session["check"] = 1;

Session["icuid"] = -1;
}
}
else
```

```
{
bedlbl.Text = "No Vacant Beds";
Session["check"] = -1;
Session["bedid"] = -1;
Session["icuid"] = -1;
}

}

}
else
{
bedlbl.Text = "";
}
}

protected void icuch_CheckedChanged(object sender, EventArgs e)
{
if (icucb.Checked == true)
{
bedcb.Checked = false;
reqcb.Checked = false;

using (SqlConnection sqlCon = new SqlConnection(constring))
{
//Displaying Doctors  in dropdown
query = "SELECT top(1) icu.icuid FROM icu join Hospitals  on icu.HospitalID = Hospitals.HospitalID join Doctor on Doctor.HospitalID=Hospitals.HospitalID where Doctor.Username=@Duser and icu.status='Vacant' order by icu.icuid";
SqlCommand cmd = new SqlCommand(query, sqlCon);
sqlCon.Open();
cmd.Parameters.AddWithValue("@Duser", Session["username"].ToString());

SqlDataReader rdr = cmd.ExecuteReader();
// int bedid = rdr.;
if (rdr.HasRows)
{

using (SqlConnection sqlCon2 = new SqlConnection(constring))
{
query = "SELECT top(1) icu.icuid FROM icu join Hospitals  on icu.HospitalID = Hospitals.HospitalID join Doctor on Doctor.HospitalID=Hospitals.HospitalID where Doctor.Username=@Duser and icu.status='Vacant' order by icu.icuid";
SqlCommand cmd2 = new SqlCommand(query, sqlCon2);
sqlCon2.Open();
cmd2.Parameters.AddWithValue("@Duser", Session["username"].ToString());

Session["icuid"] = (int)cmd2.ExecuteScalar();


iculbl.Text = " Vacant ICU Available ";
```

```
Session["check"] = 0;

Session["bedid"] = -1;
}




}
else
{
iculbl.Text = "No Vacant ICU";
Session["check"] = -1;
Session["bedid"] = -1;
Session["icuid"] = -1;
}

}

}
else
{
iculbl.Text = "";
}
}
```

4.4.4 Entertaining external and internal ICU requests:

4.4.4.1 Input:

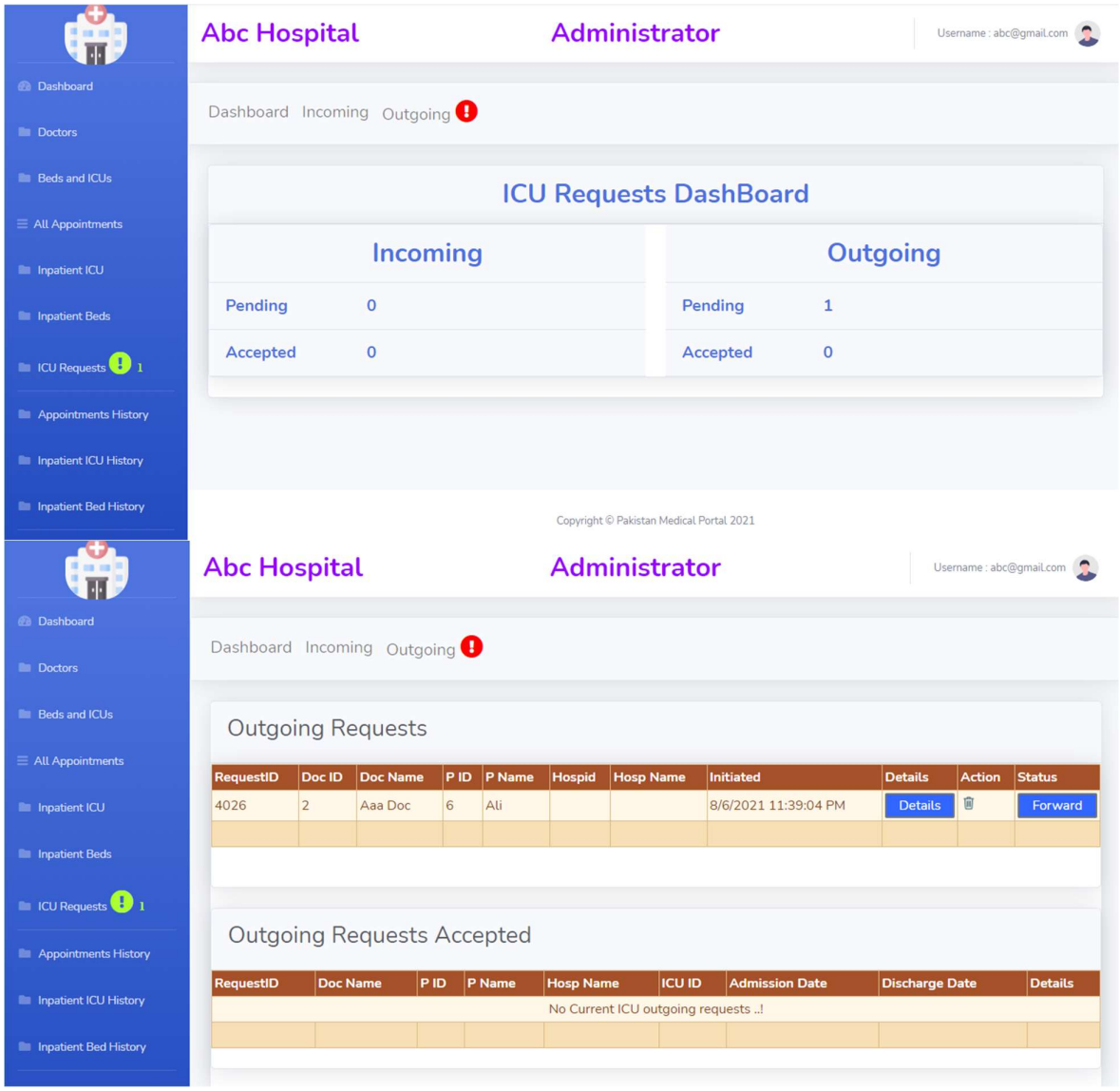

Figure 4.7: External ICU Requests Outgoing

Upon the unavailability of the ICU in current hospital, the doctor requests for ICU transfer in other nearest hospitals.

4.4.4.2 Output:

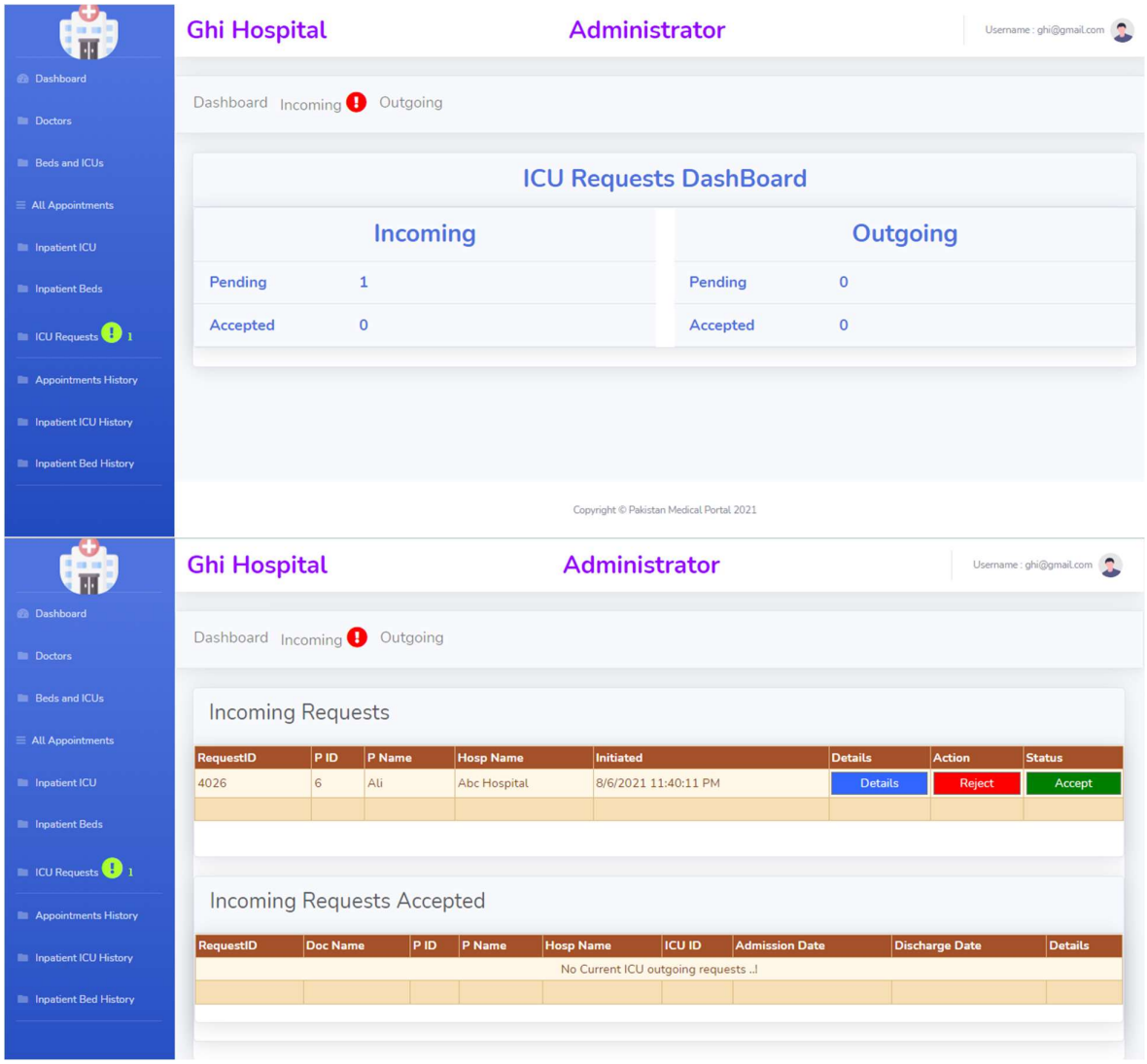

Figure 4.8: External ICU Requests Incoming

ICU for the patient is successfully allocated after the external hospital accepts the request of current hospital administration.

4.4.4.3 Code:

```
void PopulateGridview()
{
DataTable dtbl = new DataTable();

using (SqlConnection sqlCon = new SqlConnection(constring))
```

```
{
sqlCon.Open();

string query = @"select IcuRequests.RequestID, IcuRequests.MRI, IcuRequests.HealthID,
IcuRequests.DoctorID,
hospres.hospitalid as hid,hospres.name as hname,
doctor.Name as dname, Patient.Name as pname from IcuRequests
full outer join doctor on doctor.DoctorID=IcuRequests.DoctorID
join Patient on Patient.MRI=IcuRequests.MRI
join Hospitals hospreq on IcuRequests.ReqHosID=hospreq.HospitalID
full outer join Hospitals hospres on IcuRequests.ResHosID=hospres.HospitalID
join HospitalAdmin on HospitalAdmin.HospitalID=hospreq.HospitalID
where HospitalAdmin.Email=@Hemail and IcuRequests.IcuID is  null";
SqlCommand cmd = new SqlCommand(query, sqlCon);
cmd.Parameters.Add(new SqlParameter("@Hemail", Session["username"].ToString()));

SqlDataAdapter sqlDa = new SqlDataAdapter(cmd);


sqlDa.Fill(dtbl);
}
if (dtbl.Rows.Count > 0)
{
MyDataTable.DataSource = dtbl;
MyDataTable.DataBind();
}
else
{
dtbl.Rows.Add(dtbl.NewRow());
MyDataTable.DataSource = dtbl;
MyDataTable.DataBind();
MyDataTable.Rows[0].Cells.Clear();
MyDataTable.Rows[0].Cells.Add(new TableCell());
MyDataTable.Rows[0].Cells[0].ColumnSpan = dtbl.Columns.Count;
MyDataTable.Rows[0].Cells[0].Text = "No Current ICU outgoing requests   ..!";
MyDataTable.Rows[0].Cells[0].HorizontalAlign = HorizontalAlign.Center;
}

}


protected void MyDataTable_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
try
{
using (SqlConnection sqlCon = new SqlConnection(constring))
{
sqlCon.Open();
string query = "DELETE FROM IcuRequests WHERE RequestID = @Rid";
SqlCommand sqlCmd = new SqlCommand(query, sqlCon);
sqlCmd.Parameters.AddWithValue("@Rid",
Convert.ToInt32(MyDataTable.DataKeys[e.RowIndex].Value.ToString()));
```

```
sqlCmd.ExecuteNonQuery();
PopulateGridview();
lblSuccessMessage.Text = "Selected Appointment Deleted";
lblErrorMessage.Text = "";
}
}
catch (Exception ex)
{
lblSuccessMessage.Text = "";
lblErrorMessage.Text = ex.Message;
}
}


void PopulateGridview2()
{
lblSuccessMessage.Text = "";
DataTable dtbl = new DataTable();

using (SqlConnection sqlCon = new SqlConnection(constring))
{
sqlCon.Open();

string query = @"select hospitals.HospitalID as hid,hospitals.name as hname,
count(icuid) as count
from hospitals join icu on Hospitals.HospitalID=icu.HospitalID
join HospitalAdmin on HospitalAdmin.HospitalID=Hospitals.HospitalID
where hospitals.city=@city and HospitalAdmin.Email !=@Hemail and Icu.Status='Vacant'
group by Hospitals.HospitalID ,hospitals.name";
SqlCommand cmd = new SqlCommand(query, sqlCon);
cmd.Parameters.Add(new SqlParameter("@Hemail", Session["username"].ToString()));
cmd.Parameters.AddWithValue("@city", DropDownCity.SelectedValue);
SqlDataAdapter sqlDa = new SqlDataAdapter(cmd);


sqlDa.Fill(dtbl);
}
if (dtbl.Rows.Count > 0)
{
MyDataTable2.DataSource = dtbl;
MyDataTable2.DataBind();
}
else
{
dtbl.Rows.Add(dtbl.NewRow());
MyDataTable2.DataSource = dtbl;
MyDataTable2.DataBind();
MyDataTable2.Rows[0].Cells.Clear();
MyDataTable2.Rows[0].Cells.Add(new TableCell());
MyDataTable2.Rows[0].Cells[0].ColumnSpan = dtbl.Columns.Count;
MyDataTable2.Rows[0].Cells[0].Text = "No ICUs available in current city  ..!";
MyDataTable2.Rows[0].Cells[0].HorizontalAlign = HorizontalAlign.Center;}}
```

4.4.5 ICU Transfer Prediction:

4.4.5.1 Input:

The patient health details are provided to the python code as input and based on this health data, the python program predict the ICU transfer prediction using logistic regression.

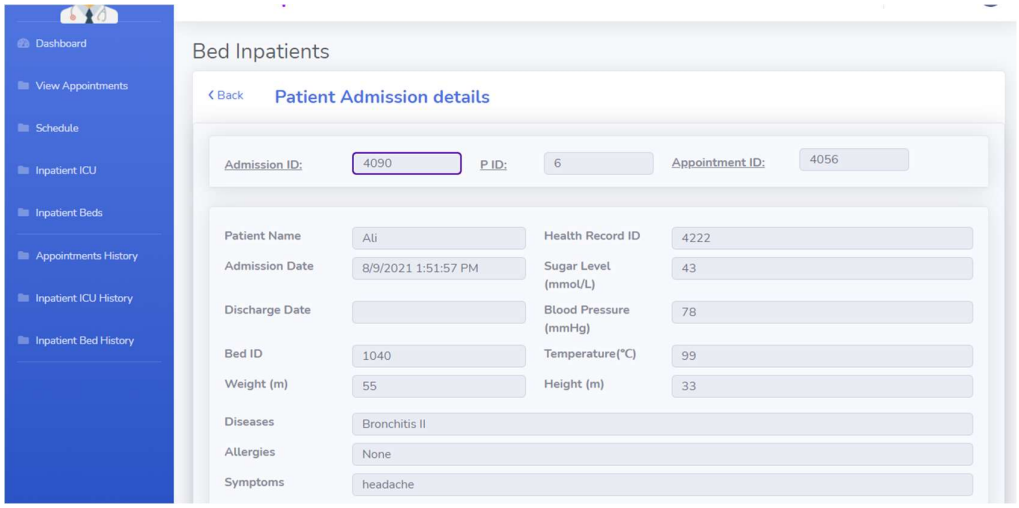

Figure 4.19: Patient Details ICU Prediction

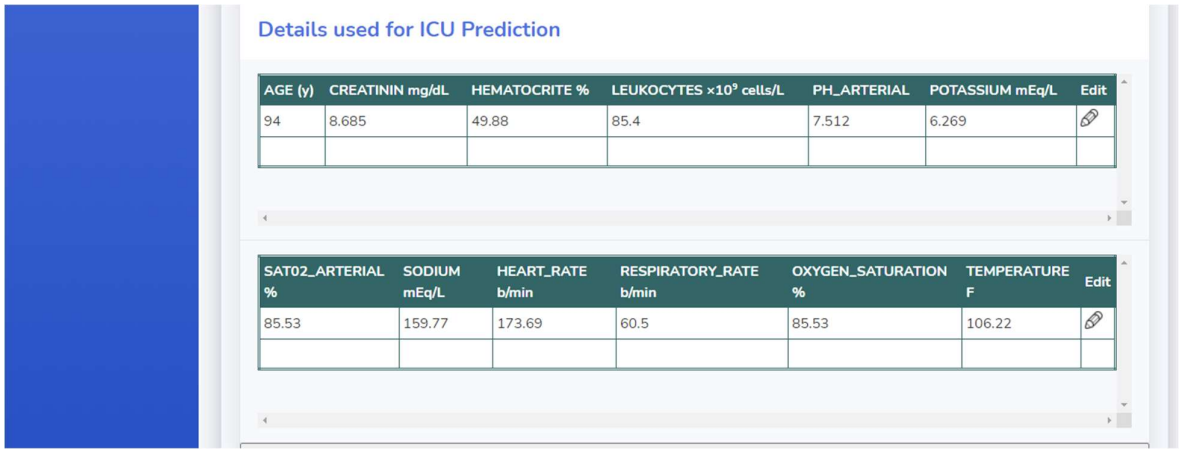

Figure 4.10: Input for ICU Prediction

4.4.5.2 Output:

The output of the python program is obtained and based on its output, the system issues a notification if the patient condition is critical or normal.

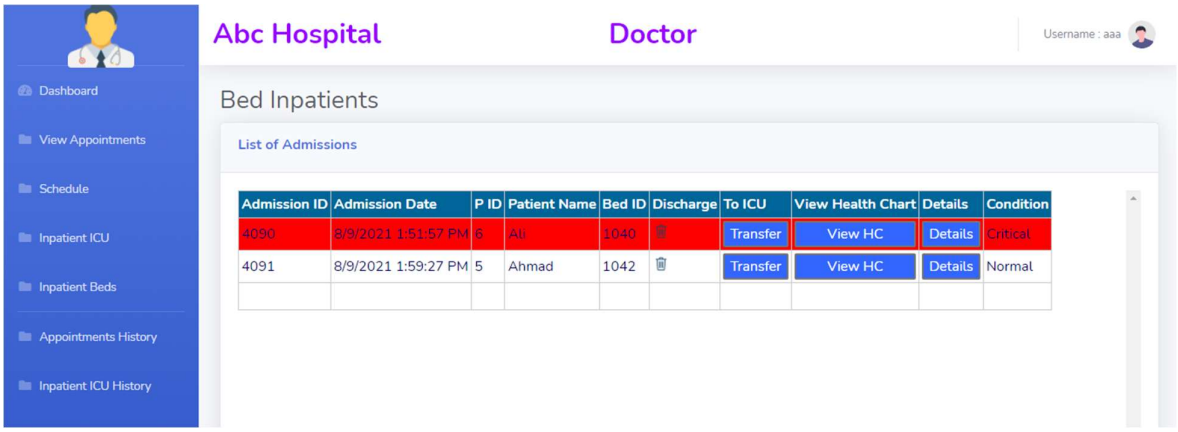

Figure 4.11: Prediction Result

4.4.5.3 Code:

Python Code:

The code python which is implemented in the project is attached below.

```
#!/usr/bin/env python
# coding: utf-8
#import pandas
import seaborn as sns
import pandas as pd
import sys
# import warnings filter
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
# load dataset
pima = pd.read_excel(r'C:\Users\hashi\OneDrive\Desktop\FYP\UnderGoing FYP Implementation\FYPv1\FYPv1\
#pima = pd.read_excel(r'dataset.xlsx')
#pima.fillna(pima.min()+pima.max()+pima.mean()+pima.median()/4, inplace=True)
pima.fillna(pima.mean(), inplace=True)
#split dataset in features and target variable
feature_cols = ['AGE_PERCENTIL','CREATININ_MEDIAN','HEMATOCRITE_MEDIAN','LEUKOCYTES_MEDIAN','PH_ARTER
X = pima[feature_cols] # Features
y = pima.ICU # Target variable
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=3)
```

Figure 4.12: Python code (Input)

```
23
24  AGE_PERCENTIL=sys.argv[1]
25  CREATININ_MEDIAN =sys.argv[2]
26  HEMATOCRITE_MEDIAN=sys.argv[3]
27  LEUKOCYTES_MEDIAN=sys.argv[4]
28  PH_ARTERIAL_MEDIAN=sys.argv[5]
29  POTASSIUM_MEDIAN =sys.argv[6]
30  SAT02_ARTERIAL_MEDIAN =sys.argv[7]
31  SODIUM_MEDIAN=sys.argv[8]
32  HEART_RATE_MEAN=sys.argv[9]
33  RESPIRATORY_RATE_MEAN=sys.argv[10]
34  OXYGEN_SATURATION_MEAN=sys.argv[11]
35  TEMPERATURE_MEDIAN=sys.argv[12]
36  # import the class
37  from sklearn.linear_model import LogisticRegression
38  # instantiate the model (using the default parameters)
39  logreg = LogisticRegression()
40  # fit the model with data
41  logreg.fit(X_train,y_train)
42
43  # define input
44  new_input = [[AGE_PERCENTIL, CREATININ_MEDIAN ,HEMATOCRITE_MEDIAN, LEUKOCYTES_MEDIAN, PH_ARTERIAL_MED
45  # get prediction for new input
46  new_output = logreg.predict(new_input)
47  # summarize input and output
48  myfinal=new_output[0]
49  print(myfinal)
50
```

Figure 4.13: Python code (Output)

```
In [13]: class_names=[0,1] # name  of classes
         fig, ax = plt.subplots()
         tick_marks = np.arange(len(class_names))
         plt.xticks(tick_marks, class_names)
         plt.yticks(tick_marks, class_names)
         # create heatmap
         sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
         ax.xaxis.set_label_position("top")
         plt.tight_layout()
         plt.title('Confusion matrix', y=1.1)
         plt.ylabel('Actual label')
         plt.xlabel('Predicted label')

Out[13]: Text(0.5, 257.44, 'Predicted label')
```

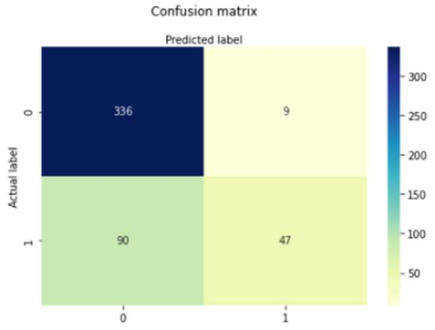

Figure 4.14: Python Confusion Matrix

```
In [14]: print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
         print("Precision:",metrics.precision_score(y_test, y_pred))
         print("Recall:",metrics.recall_score(y_test, y_pred))

         Accuracy: 0.7946058091286307
         Precision: 0.8392857142857143
         Recall: 0.34306569343065696
```

Figure 4.15: Python code Accuracy

C# Code:

```
public int run_cmd(float AGE, float CREATININ, float HEMATOCRITE, float LEUKOCYTES, float PH_ARTERIAL, float POTASSIUM, float SAT02_ARTERIAL, float SODIUM, float HEART_RATE, float RESPIRATORY_RATE, float OXYGEN_SATURATION, float TEMPERATURE_M)

{


// 1) Create Process Info

var psi = new ProcessStartInfo();

var script = Server.MapPath(ConfigurationManager.AppSettings["pypath"].ToString()) + "ICU.py";

psi.FileName = ConfigurationManager.AppSettings["pythonexepath"].ToString();


psi.Arguments = $"\"{script}\" \"{AGE}\" \"{CREATININ}\" \"{HEMATOCRITE}\" \"{LEUKOCYTES}\" \"{PH_ARTERIAL}\" \"{POTASSIUM}\" \"{SAT02_ARTERIAL}\" \"{SODIUM}\" \"{HEART_RATE}\" \"{RESPIRATORY_RATE}\" \"{OXYGEN_SATURATION}\" \"{TEMPERATURE_M}\" ";


// 3) Process configuration

psi.UseShellExecute = false;

psi.CreateNoWindow = true;

psi.RedirectStandardOutput = true;

psi.RedirectStandardError = true;


// 4) Execute process and get output

var errors = "";

int results = -1;


using (var process = Process.Start(psi))
```

```
{
// errors = process.StandardError.ReadToEnd();
results = Int32.Parse(process.StandardOutput.ReadLine());
}
// 5) Display output
return results;
}
void CheckICUNeed()
{
int admissionid, healthid;
Label emergency;
float AGE, CREATININ, HEMATOCRITE, LEUKOCYTES, PH_ARTERIAL, POTASSIUM, SAT02_ARTERIAL, SODIUM, HEART_RATE, RESPIRATORY_RATE, OXYGEN_SATURATION, TEMPERATURE_M;
AGE = CREATININ = HEMATOCRITE = LEUKOCYTES = PH_ARTERIAL = POTASSIUM = SAT02_ARTERIAL = SODIUM = HEART_RATE = RESPIRATORY_RATE = OXYGEN_SATURATION = TEMPERATURE_M = 0.00f;
try
{
if (countrecord !=0)
foreach (GridViewRow row in MyDataTable.Rows)
{
admissionid = Convert.ToInt32(MyDataTable.DataKeys[row.RowIndex].Value.ToString());

emergency = MyDataTable.Rows[row.RowIndex].FindControl("emergency") as Label;
//Checking availability
SqlConnection con = new SqlConnection(constring);

SqlCommand cmd = new SqlCommand("select Admission.HealthID from BedAdmission join Admission on Admission.AdmitID=BedAdmission.AdmissionID where Admission.AdmitID= @admissionid  ", con);
cmd.Parameters.AddWithValue("admissionid", admissionid);
```

```
con.Open();
SqlDataReader sdr ;
healthid = (int)cmd.ExecuteScalar();

con.Close();
con = new SqlConnection(constring);

cmd = new SqlCommand("select AGE , CREATININ , HEMATOCRITE , LEUKOCYTES ,PH_ARTERIAL,POTASSIUM , SAT02_ARTERIAL , SODIUM , HEART_RATE , RESPIRATORY_RATE , OXYGEN_SATURATION , TEMPERATURE_M from healthrecord where healthid=@hid", con);
cmd.Parameters.AddWithValue("hid", healthid);

con.Open();
sdr = cmd.ExecuteReader();
while (sdr.Read())
{
AGE = float.Parse(( sdr["AGE"].ToString()));
CREATININ = float.Parse( sdr["CREATININ"].ToString());
HEMATOCRITE = float.Parse(sdr["HEMATOCRITE"].ToString());
LEUKOCYTES = float.Parse( sdr["LEUKOCYTES"].ToString());
PH_ARTERIAL = float.Parse( sdr["PH_ARTERIAL"].ToString());
POTASSIUM = float.Parse( sdr["POTASSIUM"].ToString());
SAT02_ARTERIAL = float.Parse( sdr["SAT02_ARTERIAL"].ToString());
SODIUM = float.Parse( sdr["SODIUM"].ToString());
HEART_RATE = float.Parse( sdr["HEART_RATE"].ToString());
RESPIRATORY_RATE = float.Parse( sdr["RESPIRATORY_RATE"].ToString());
OXYGEN_SATURATION = float.Parse( sdr["OXYGEN_SATURATION"].ToString());
TEMPERATURE_M = float.Parse( sdr["TEMPERATURE_M"].ToString());
}
con.Close();
AGE = scaleBetween(AGE, 1,110);
```

```
CREATININ = scaleBetween(CREATININ,1,10);

HEMATOCRITE = scaleBetween(HEMATOCRITE,20,55);

LEUKOCYTES = scaleBetween(LEUKOCYTES,0.1f,100);

PH_ARTERIAL = scaleBetween(PH_ARTERIAL,7,7.6f);

POTASSIUM = scaleBetween(POTASSIUM,2,7);

SAT02_ARTERIAL = scaleBetween(SAT02_ARTERIAL,1,100);

SODIUM = scaleBetween(SODIUM,100,170);

HEART_RATE = scaleBetween(HEART_RATE,20,200);

RESPIRATORY_RATE = scaleBetween(RESPIRATORY_RATE,50,70);

OXYGEN_SATURATION = scaleBetween(OXYGEN_SATURATION,1,100);

TEMPERATURE_M = scaleBetween(TEMPERATURE_M,90,109);

if (run_cmd(AGE, CREATININ, HEMATOCRITE, LEUKOCYTES, PH_ARTERIAL, POTASSIUM,
SAT02_ARTERIAL, SODIUM, HEART_RATE, RESPIRATORY_RATE,
OXYGEN_SATURATION, TEMPERATURE_M )==1)

{

MyDataTable.Rows[row.RowIndex].BackColor = Color.Red;

//emergency.BackColor= Color.Red;

emergency.Text = "Critical";

}

else

{

//MyDataTable.Rows[row.RowIndex].BackColor = Color.Red;

emergency.Text = "Normal";

//emergency.BackColor = Color.Green;}}}

catch (Exception ex)

{

lblSuccessMessage.Text = "";

lblErrorMessage.Text = ex.Message;

lblErrorMessage.Visible = true;

lblSuccessMessage.Visible = false;}}
```

Chapter 5

5. Software Testing

This chapter provides a description about the adopted testing procedure. This includes the selected testing methodology, test suite and the test results of the developed software.

5.1. Testing Methodology

We have used black box testing in our testing phase. We used black box testing because it is very efficient and it contains following benefits. Black box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance. Black box unit testing is used in our project. Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use:

- Black box tests are reproducible.
- Find software bugs early.
- Facilitates change.
- The environment the program is running is also tested.
- The invested effort can be used multiple times.
- More effective on larger units of code than glass box testing.
- Tester needs no knowledge of implementation, including specific programming languages.
- Tests are done from a user's point of view.
- Will help to expose any ambiguities or inconsistencies in the specifications.

5.2. Testing Environment

Test Café Studio:

TestCafe allows to write tests using TypeScript or JavaScript (with its modern features like async/await). We get all the advantages of strongly-typed languages like rich coding assistance, painless scalability, check-as-you-type code verification, etc., by using TypeScript to write your TestCafe tests. To create a test, create a new .js or .ts file. This file must have a special structure - tests must be organized into fixtures.

Running the Test:

You can run the test from a command shell by calling a single command where you specify the target browser and file path.

testcafe chrome test1.js

Observing Page State

TestCafe allows you to observe the page state. For this purpose, it offers special kinds of functions that will execute your code on the client: Selector used to get direct access to DOM elements and ClientFunction used to obtain arbitrary data from the client side. You call these functions as regular async functions, that is, you can obtain their results and use parameters to pass data to them.

The selector API provides methods and properties to select elements on the page and get their state.

5.2.1 Test Case: System Admin Sign in:

The following is test case details for System admin Sign in.

Table 5.1: System Admin Sign in

Date: 24 Jan 2021	
System: System Admin	
Objective: System Admin Login	Test ID: 1
Version: 1	Test type: Unit Testing
Input: Username: hasham1998 Password: 1998	
Expected Result: System admin will be signed in	
Actual result: Passed	

Description:

In this figure, a test case is built using Test Café studio. Firstly, a fixture is created which records a test case, providing the user credentials for login. Then the test is run and its results are obtained deciding whether the test is passed or failed. The java script of the test can also be obtained.

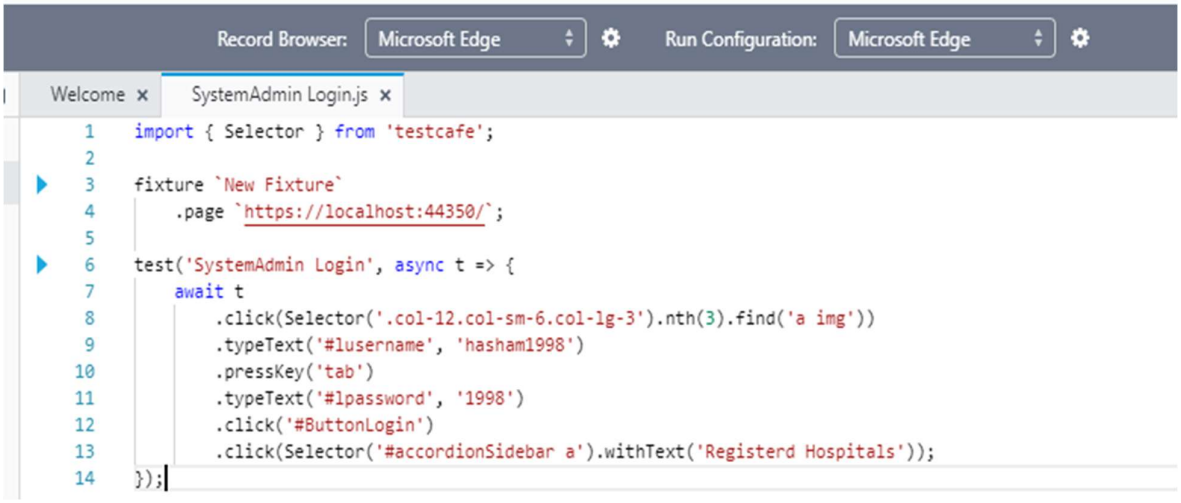

Figure 5.1: System Admin Sign in Java Script

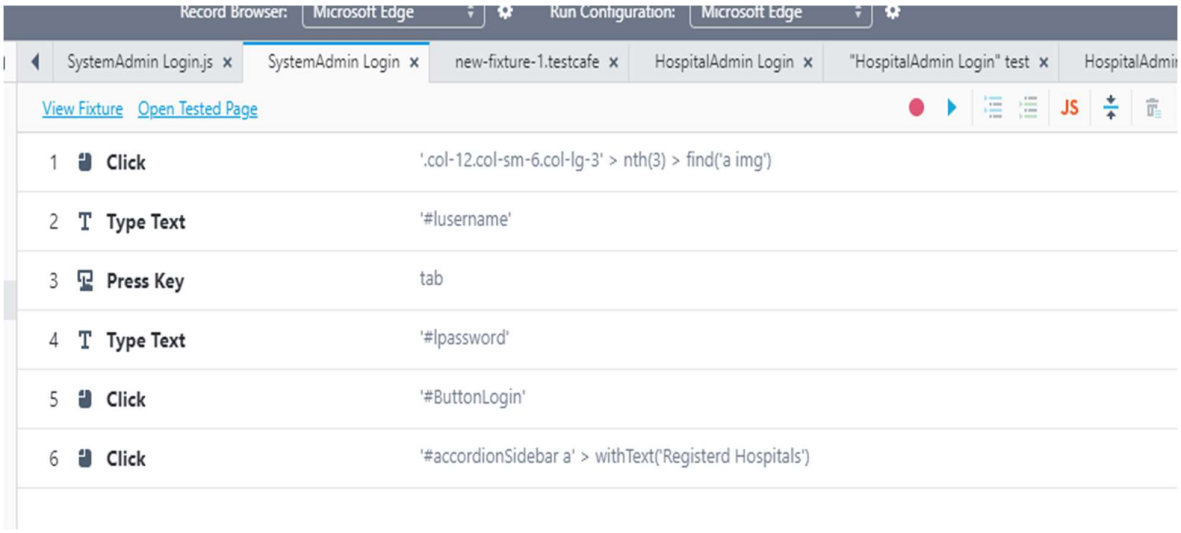

Figure 5.2: System Admin Sign in Test Case

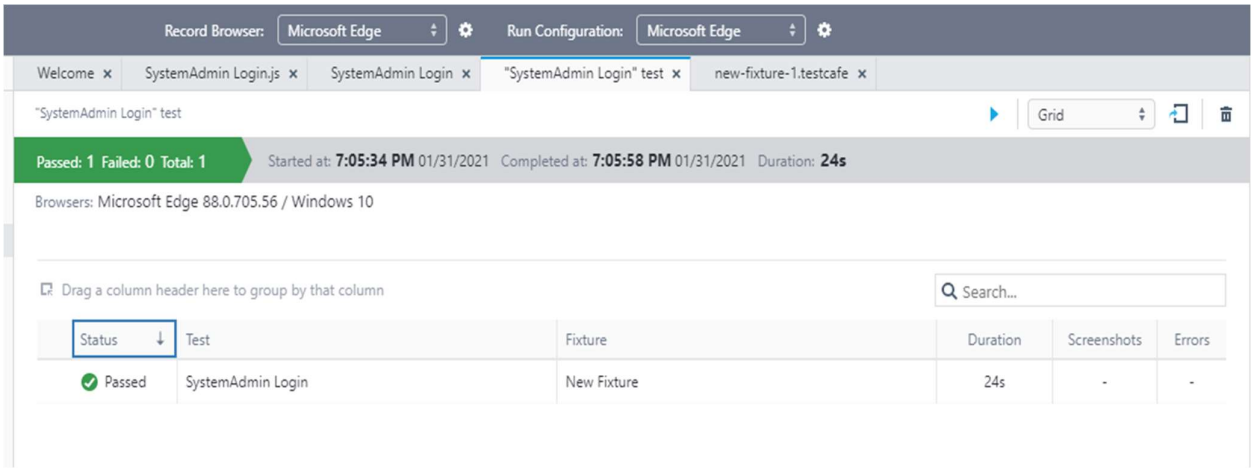

Figure 5.3: System Admin Sign in Passed

5.2.2 Test Case: Hospital Admin Sign in:

The following is test case details for Hospital admin Sign in.

Table 5.2: Hospital Admin Sign in

Date: 26 Jan 2021	
System: Hospital Administration	
Objective: Hospital Admin Login	Test ID: 2
Version: 1	Test type: Unit Testing
Input: Username: abc@gmail.com Password: 123	
Expected Result: Hospital admin will be signed in	
Actual result: Passed	

Description:

In this figure, a test case is built using Test Café studio. Firstly, a fixture is created which records a test case, providing the user credentials for login. Then the test is run and its results are obtained deciding whether the test is passed or failed. The java script of the test can also be obtained.

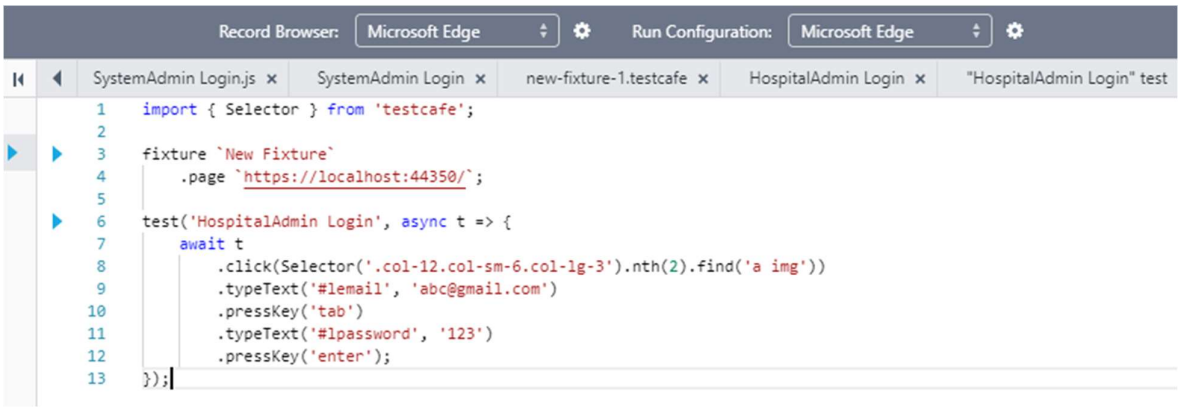

Figure 5.4: Hospital Admin Sign in Java Script

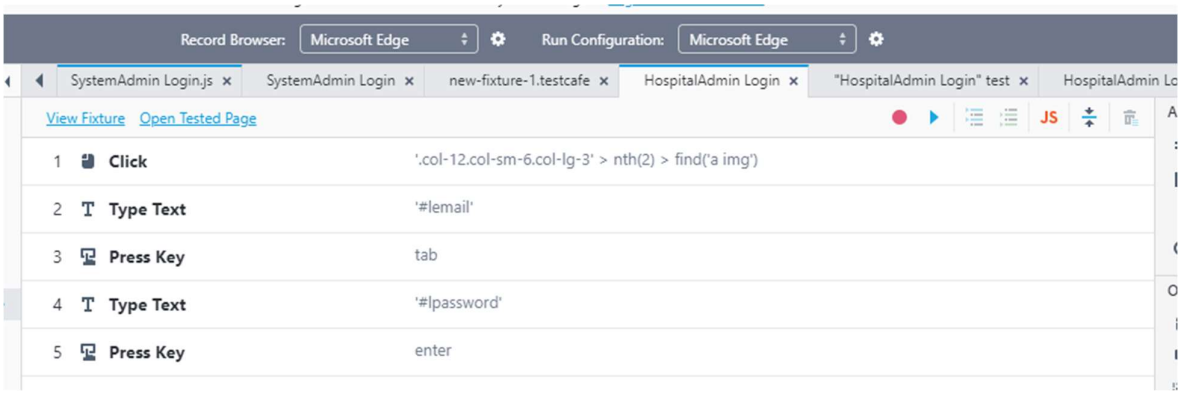

Figure 5.5: Hospital Admin Sign in Test case

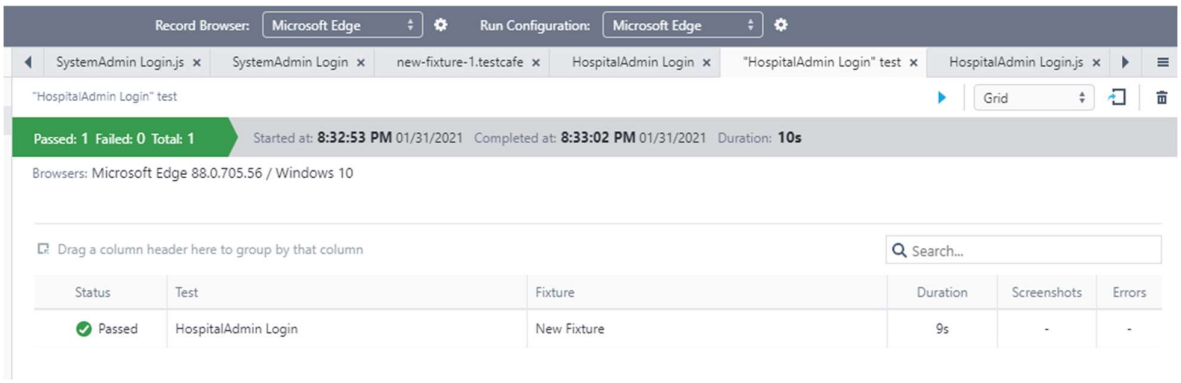

Figure 5.6: Hospital Admin Sign in Passed

5.2.3 Test Case: Doctor Sign in:

The following is test case details for Doctor Sign in.

Table 5.3: Doctor Sign in

Date: 28 Jan 2021	
System: Doctor sign in	
Objective: System Admin Login	Test ID: 3
Version: 1	Test type: Unit Testing

Input: Username: aaa Password: 123	
Expected Result: Doctor will be signed in	
Actual result: Passed	

Description:

In this figure, a test case is built using Test Café studio. Firstly, a fixture is created which records a test case, providing the user credentials for login. Then the test is run and its results are obtained deciding whether the test is passed or failed. The java script of the test can also be obtained.

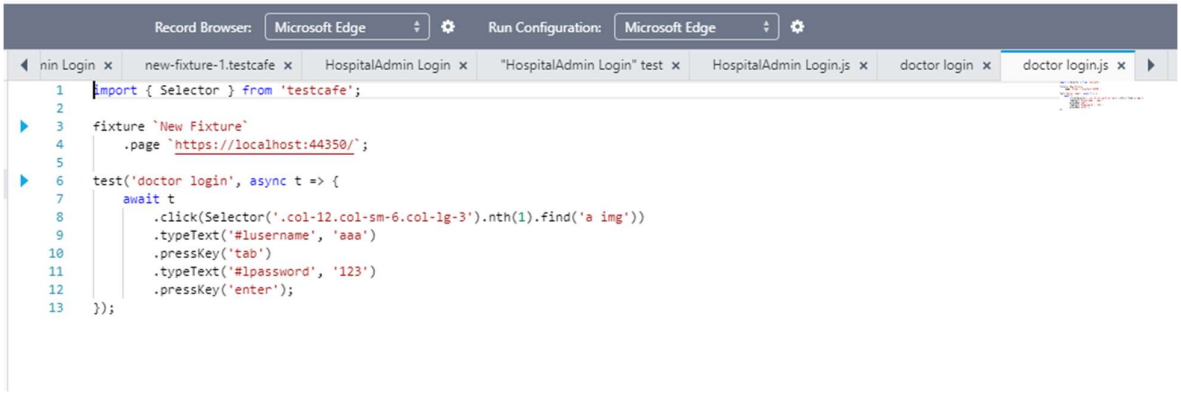

Figure 5.7: Doctor Sign in Java Script

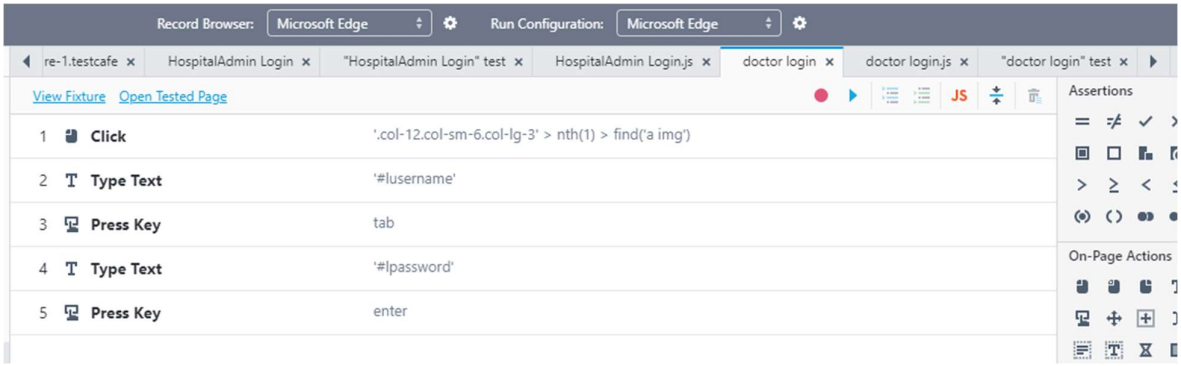

Figure 5.8: Doctor Sign in Test Case

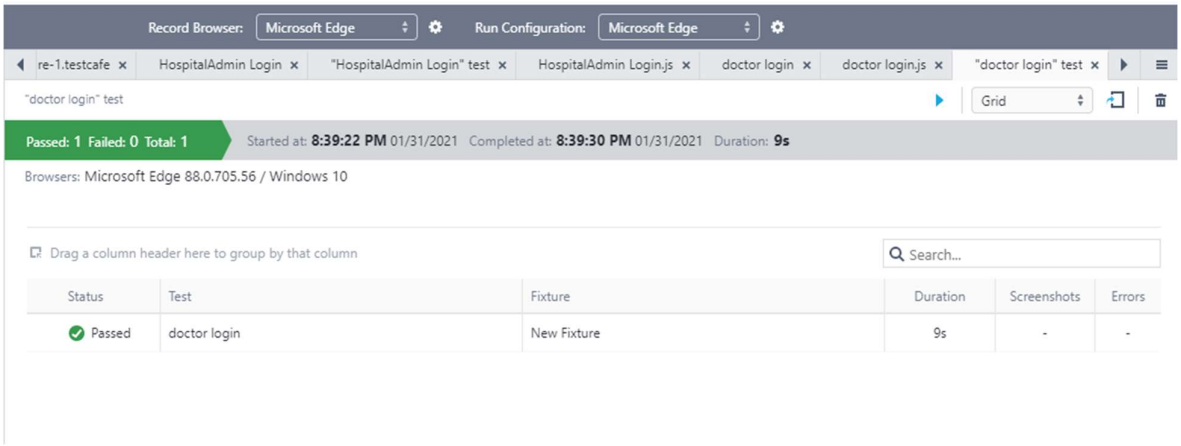

Figure 5.9: Doctor Sign in Passed

5.2.4 Test Case: Appointment Booking:

The following is test case details for Appointment Booking.

Table 5.4: Appointment Booking

Date: 30 Jan 2021	
System: Appointment booking	
Objective: System Admin Login	Test ID: 4
Version: 1	Test type: Unit Testing
Input: Username: qazi Password: 123 City: Islamabad Hospital: abc Department: Dentist Doctor: Ahmad Date: 6 jan 2021	
Expected Result: Patient's appointment will be fixed with doctor.	

Actual result: Passed	

Description:

In this figure, a test case is built using Test Café studio. Firstly, a fixture is created which records a test case, providing the user credentials for login of patient. The patient searches hospital and required department and chooses the desired doctor. Then the test is run and its results are obtained deciding whether the test is passed or failed. The java script of the test can also be obtained.

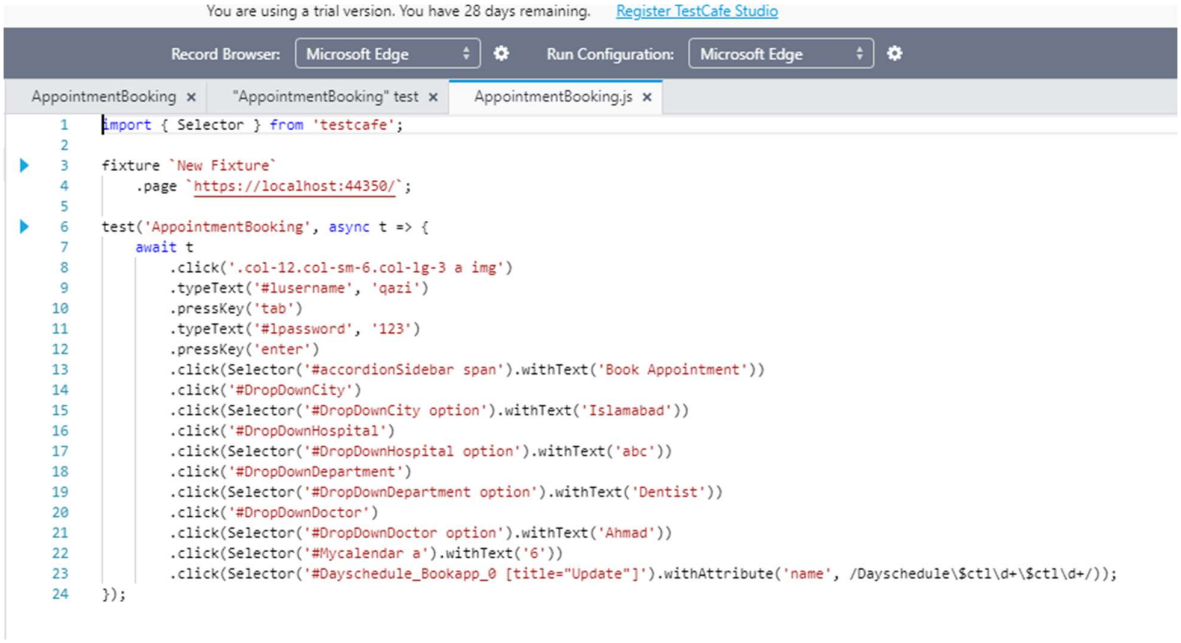

Figure 5.10: Appointment Booking Java Script

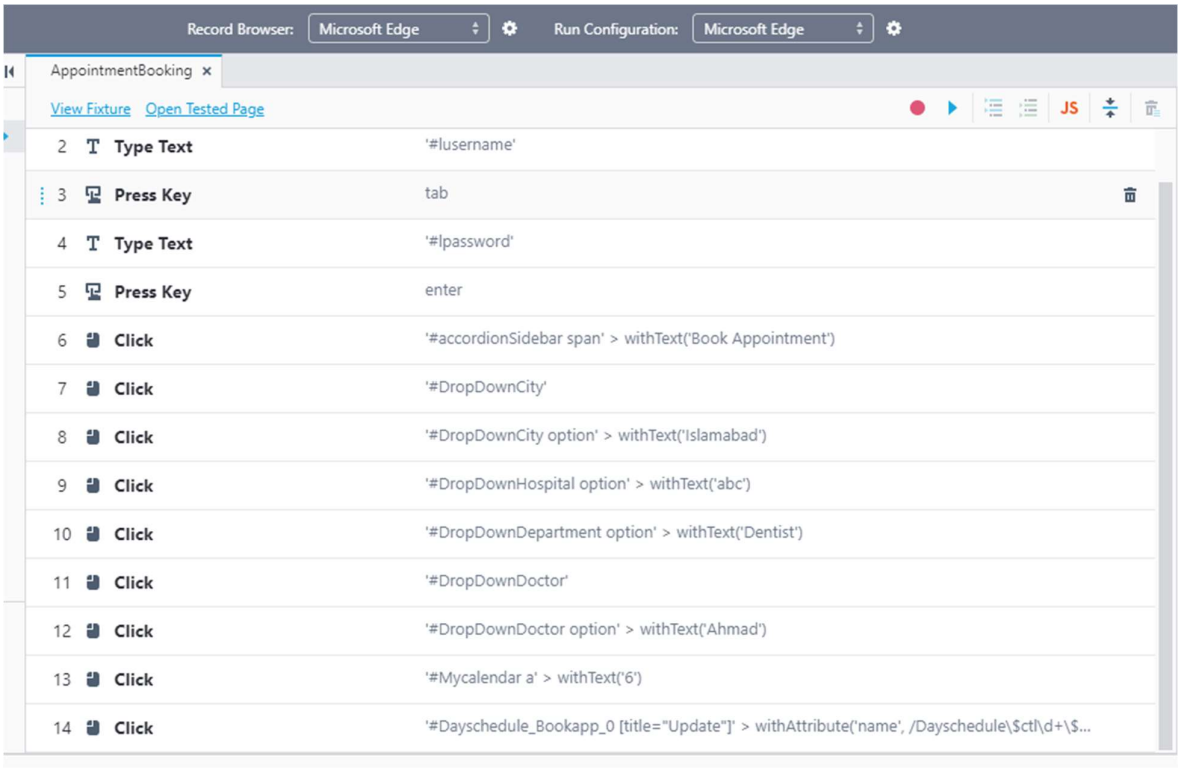

Figure 5.11: Appointment Booking Test Case

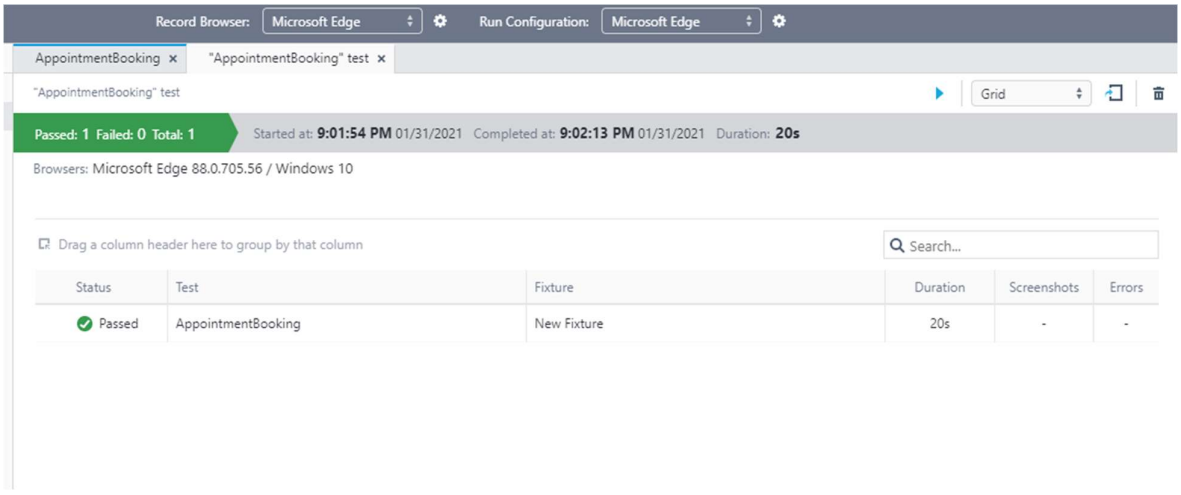

Figure 5.12: Appointment Booking Passed

Chapter 6

6. Software Deployment

6.1. Installation / Deployment Process Description:

For the deployment, we will provide the user a zipped file which contains all the published aspx web pages of the project. It will also contain the scripts (.sql) and backup (.bak) file of the database.

The user can use the zip file to host the project on any desired web server and get it running.

6.1.1 Web.config file:

The user must have to replace the connection string in web.config file according the domain on which the web application is being host.

Figure 6.1: Connection String to be replaced

6.1.2 New connection string:

This is where the user can get new connection string from the domain where the user intends to deploy the web application.

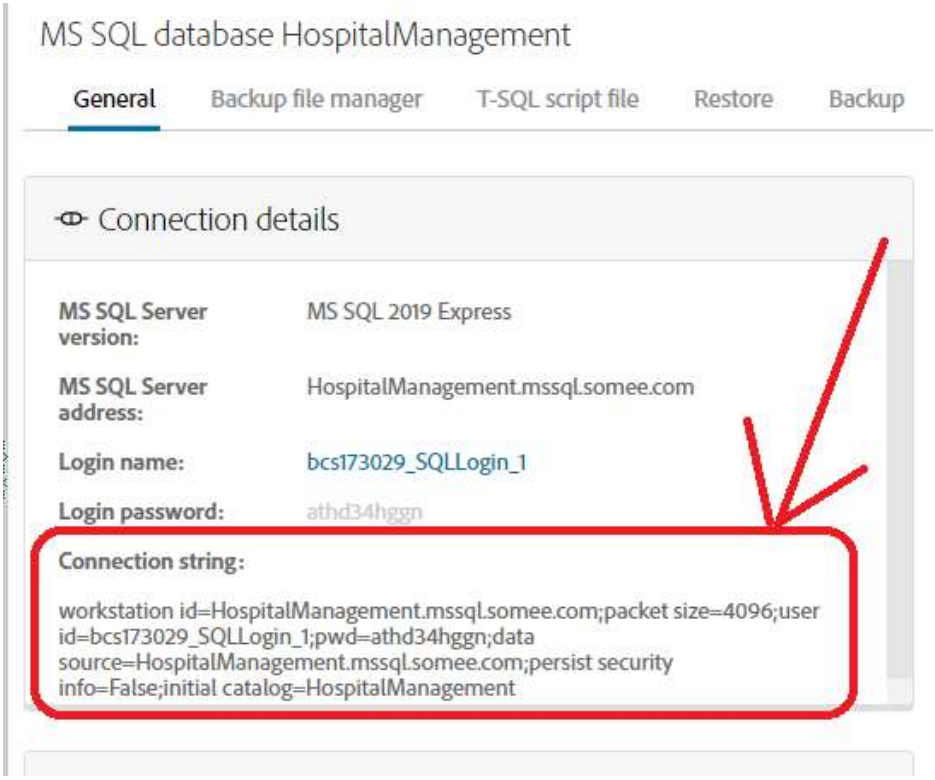

Figure 6.2: New Connection String (Domain Hosting site)

Chapter 7

7. Project Evaluation

7.1 Project Evaluation Report

Table 7.1: Evaluation Report

Examiner Name:	
S. No.	**Suggestion**

Other Comments (If any):

_____
_____
_____
_____
_____
_____
_____
_____

_____

Signature