**A Project Report On**

**Doctor Desk**

**Master of Computer Applications (2022)**

**Submitted To**

**Shri Chimanbhai Patel Post Graduate Institute of Computer Applications**



**Gujarat Technological University**



**Under Guidance of**

**Prof. Tinal Parikh**

**Submitted By:**

**Prajapati Arjun :- 225350694084**

**Prajapati Mit :- 225350694087**

**Thakkar Manthan :- 225350694121**

Acknowledgement

When reach the completion of the Project giving Credit becomes a must, as without the support of many People and their guidance, this project could not be able to reach its goal successfully.

First, we would like to thanks to all professors and all staff members of my college and special thanks to our Internal Guide “Prof. Tinal Parikh” for the co-operation and keep interest extended by them, throughout our degree course. It is the base that they have built, which sustains such good jobs.

Last but not least, my heartfelt appreciation goes to all those not named here, but who have rendered their co-operation, little or more, directly or indirectly in the course of development of this system. To all my friends and most importantly to myself.

Thanks, all

All Group Members

Index

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Content** | **Page Number** |
| 1 | Introduction  1.1. Existing System  1.2. Need for the New System  1.3. Objective of the New System  1.4. Problem Definition  1.5. Core Components  1.6. Project Profile  1.7. Assumptions and Constraints  1.8. Advantages and Limitations of the Proposed System | 1 |
| 2 | Requirement Determination & Analysis  2.1. Requirement Determination  2.2. Targeted Users | 5 |
| 3 | System Design  3.1 Use Case Diagram  3.2 Class Diagram  3.3 Interaction Diagram  3.4 Activity Diagram  3.5 Data Dictionary | 6 |
| 4 | Development    4.1 Coding Standards  4.2 Screen Shot | 19 |
| 5 | Agile Documentation  5.1 Agile Project Charter  5.2 Agile Roadmap/Schedule  5.3 Agile Project Plan  5.4 Agile User Story  5.5 Agile Release Plan  5.6 Agile Sprint Backlog  5.7 Agile Test Plan | 29 |
| 6 | Proposer Enhancements | 36 |
| 7 | Conclusion | 37 |
| 8 | Bibliography | 38 |

1. **Introduction**

* **Doctor Desk** is a user-friendly and secure web application that provide a platform for doctors and patients to interact efficiently and effectively.
* The app allows doctors to manage appointments, view patient and make Prescription for patient.
* Patients can book appointments, view doctor’s schedules , and also view Booked appointments.
* There are basically 3 modules in this project:
  + Admin
  + Doctor
  + Patient
* Patients can register and log in to the app, they can view the doctor's schedule and book an appointment.
  1. Existing System
* Limited Accessibility: With no patient side interface, patients may find it difficult to book appointments, view their medical records or communicate with their doctor. This can lead to a lack of patient engagement and limit the accessibility of healthcare services.
* Missed Appointments: Without a patient side interface, there is a higher risk of missed appointments. Patients may not receive reminders or be able to reschedule appointments, which can result in wasted time and resources for doctors and their clinic.
* Limited Data Sharing: The doctor app with no patient side interface may not facilitate the sharing of medical data between patients and their doctors. This can lead to incomplete medical histories and missed diagnoses, as doctors may not have access to all relevant patient information. This can also limit collaboration between healthcare providers and result in fragmented care for patients.

1.2 Need for the New System

* In this system doctor can easily maintain their clinic.
* Doctor can easily manage appointment, their patients and also make prescription for patients.
* Patients can easily book appointment and save their time and get treatment as soon as fast.
* Patients can get update for appointment via e-mail.

1.3 Objective of the New System

* With no patient side interface, patients may find it difficult to book appointments, view their medical records or communicate with their doctor.
* The Patient side interface enable to book their appointment with doctor in advance.
* This features also allow doctor’s schedules and patients appointment.

1.4 Problem Definition

* In the old system medication error caused by miscommunication between doctor and pharmacists due to illegible hand-writing.
* In the old system patient need to call the clinic or visit in person to schedule appointment which can time consuming and frustrating for both patient and healthcare provider.

1.5 Core Components

|  |  |
| --- | --- |
| Operating System:  Front End Tools:  Backend:  Documentation Tools:  Database:  Browser specification: | Windows 7 and above  React Js  Node Js  Visual Studio Code  Mongo Db  Mozilla Firefox, Chrome, Microsoft Edge |
|  |  |

1.6 Project Profile

|  |  |
| --- | --- |
| Group Member: | Prajapati Arjun  Prajapati Mit  Thakkar Manthan |
| Enrollment: | 225350694084  225350694087  225350694121 |
| Internal Guide: | Prof. Tinal Parikh |
| Front End: | React Js |
| Back End: | Node Js |

1.7 Assumptions and Constraints

Assumption:

* Hardware Limitations The limitation of that it requires RAM that cannot be less than 512 MB.
* The processor cannot be less than 2 GHz speed as recommended in the hardware requirements.

Constraints:

* Email id verify before submitting.
* Check validation in email that @ is occur in mail and mail id ends with gmail.com.
* Null values are not allowed in registration form.
* In numeric fields user cannot insert character values like mobile number and age fields.
* User have to insert password and confirm password before submitting.
* This password use for the future login.

1.8 Advantages and Limitations of the Proposed System

Advantages:

* By booking an appointment, you can choose a specific time slot that suits your schedule. This helps you plan your day effectively and reduces waiting time at the clinic or hospital.
* Appointments ensure that you receive priority access to healthcare professionals. Instead of waiting in long queues, you can bypass the crowd and be seen by the doctor at the designated time.
* With an appointment, healthcare providers can allocate sufficient time to understand your medical concerns, perform a thorough examination, and address any specific health issues or questions you may have. This personalized attention leads to a more comprehensive and satisfactory healthcare experience.
* Handwriting can lead to errors in medication. Our DoctorDesk project includes an E-Prescription feature to ensure clear and accurate communication of medications

Disadvantages:

* The app may experience technical issues or downtime, which can disrupt patient care and reduce efficiency.
* In some cases, technical issues may require manual scheduling or other administrative tasks, which can be time-consuming and frustrating.
* The app may collect and store sensitive patient data, raising privacy concerns.
* If the app is not properly secured, patient data may be vulnerable to cyberattacks or other breaches, which can lead to identity theft and other problems.
* The app may not be accessible to all patients or healthcare providers, particularly those who are not comfortable using technology or who do not have access to reliable internet or mobile devices.
* In addition, some healthcare providers may be resistant to adopting new technology, which can slow adoption and reduce the overall benefits of the app.

**2. Requirement Determination & Analysis**

2.1 Requirement Determination

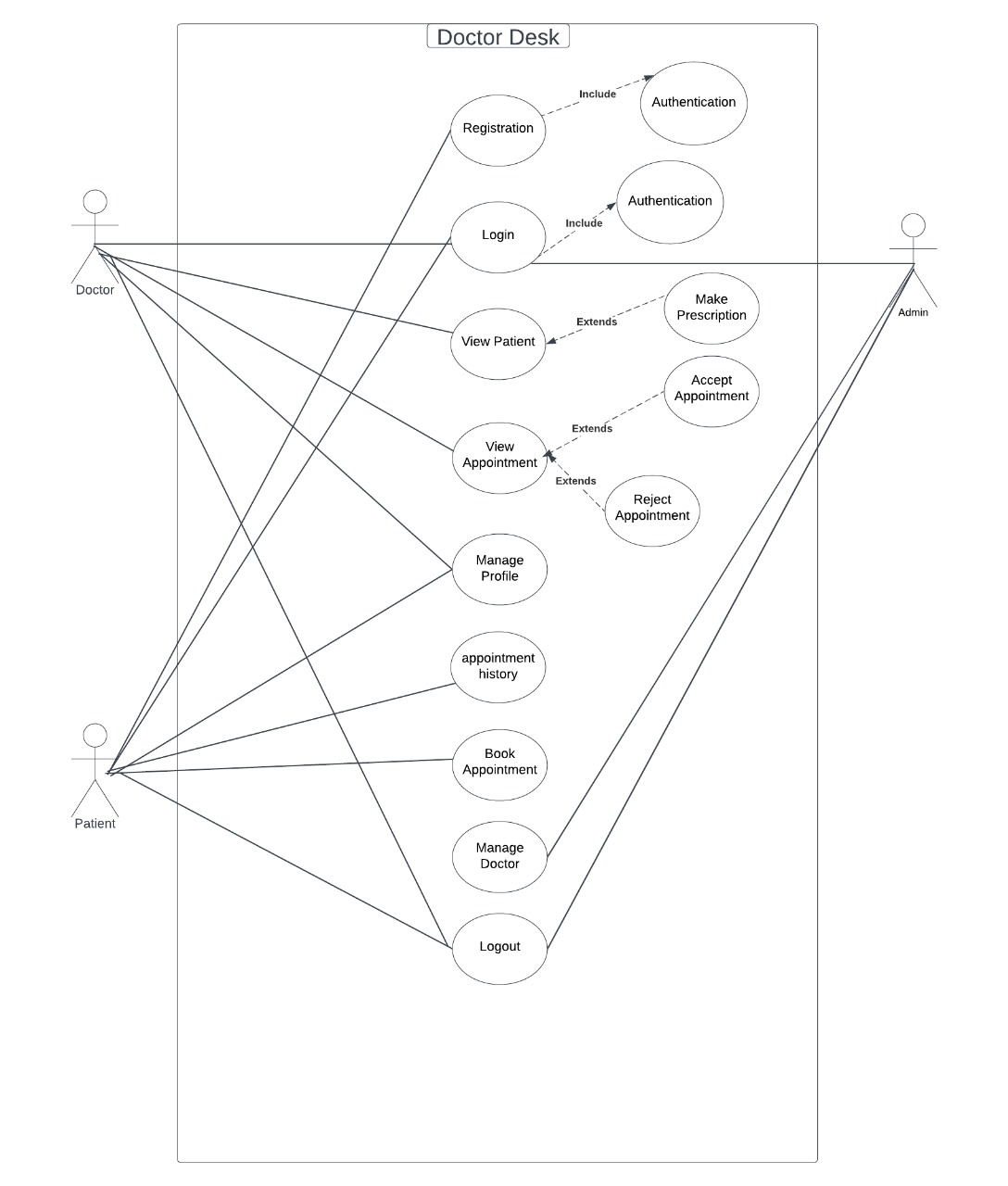
|  |  |
| --- | --- |
| Operating System:  Front End Tools:  Backend:  Documentation Tools:  Database:  Browser specification: | Windows 7 and above  Html, Bootstrap, CSS, React Js  Node.Js  Visual Studio Code  Mongo DB  Mozilla Firefox, Chrome, Microsoft Edge |

2.2 Targeted Users

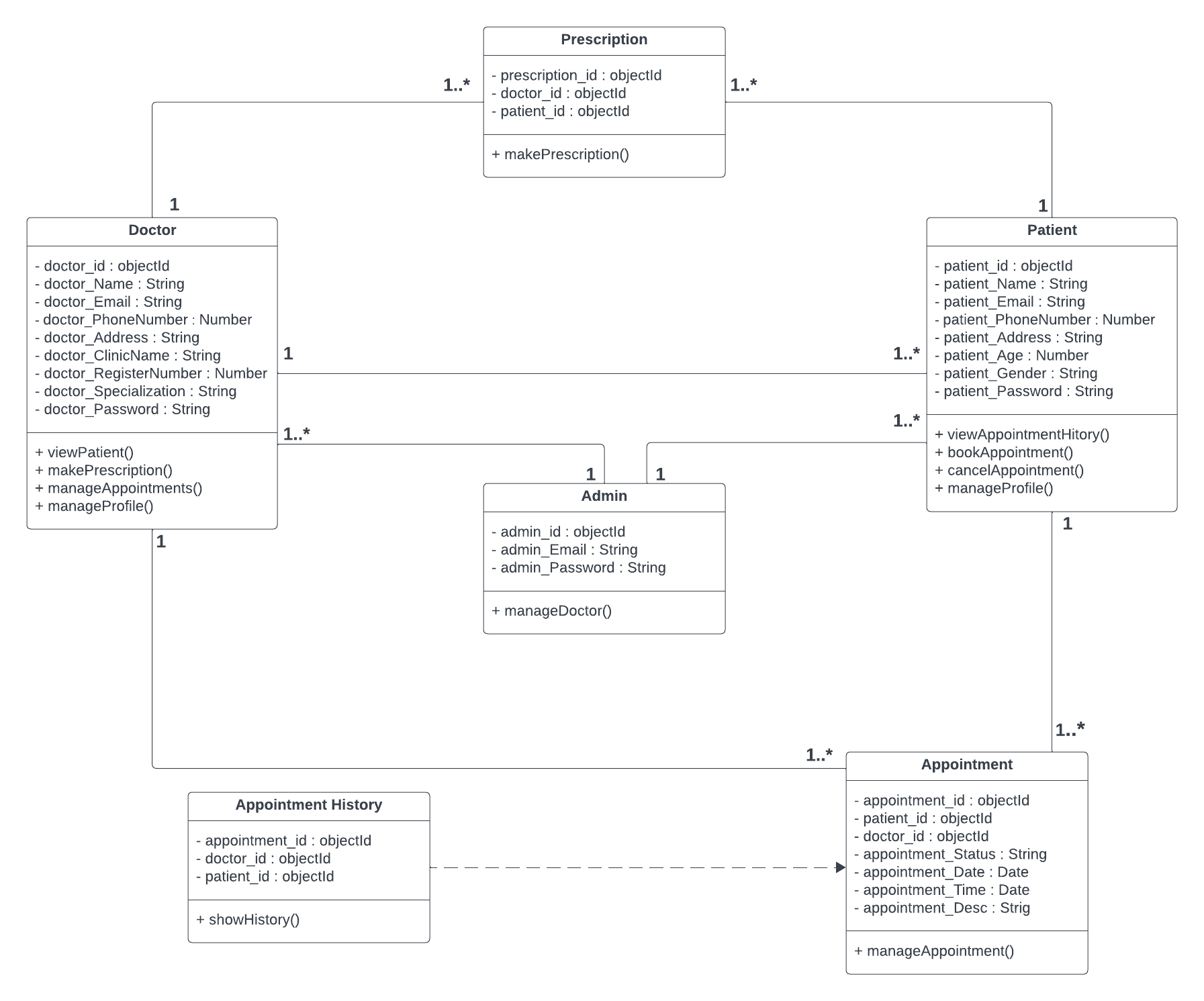
* Modules:
  + The application comprises the following major modules:
    - Doctor Module
    - Patient Module
    - Admin Module
* Doctor Module:
  + This module provides functionalities for doctors.
  + This module provides to maintain patient history.
  + And appointment management and Patient Prescription.
* Patient Module:
  + This module can provide functionalities for appointment booking.
  + This module can provide manage profile functionalities.
  + This module can provide past patient history
* Admin Module:
  + This module can provide functionalities for view Patients and Doctors.
  + This module can provide View appointments functionalities.

**3.System Design**

3.1 Use Case Diagram

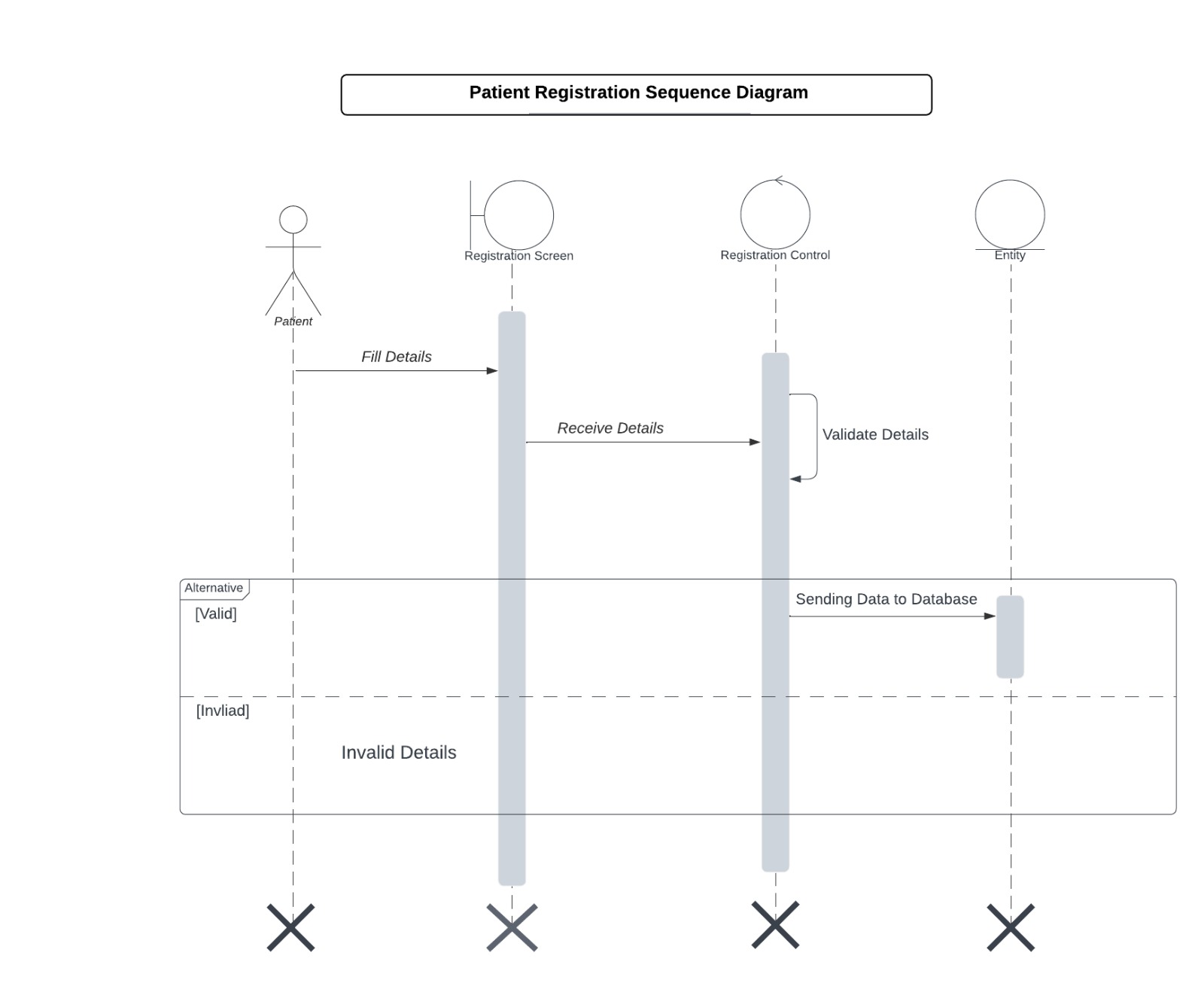


3.2 Class Diagram

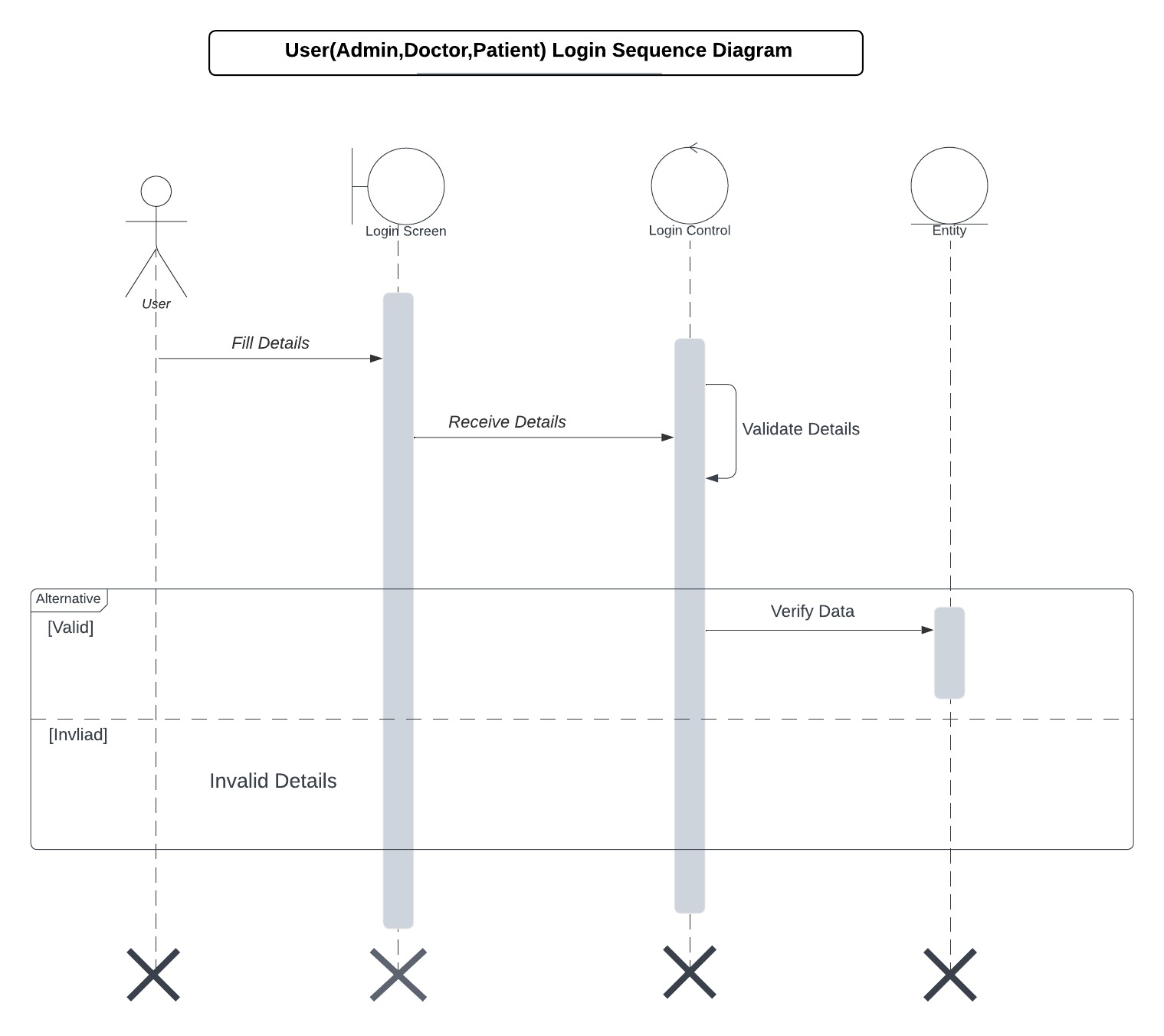


3.3 Sequence Diagram

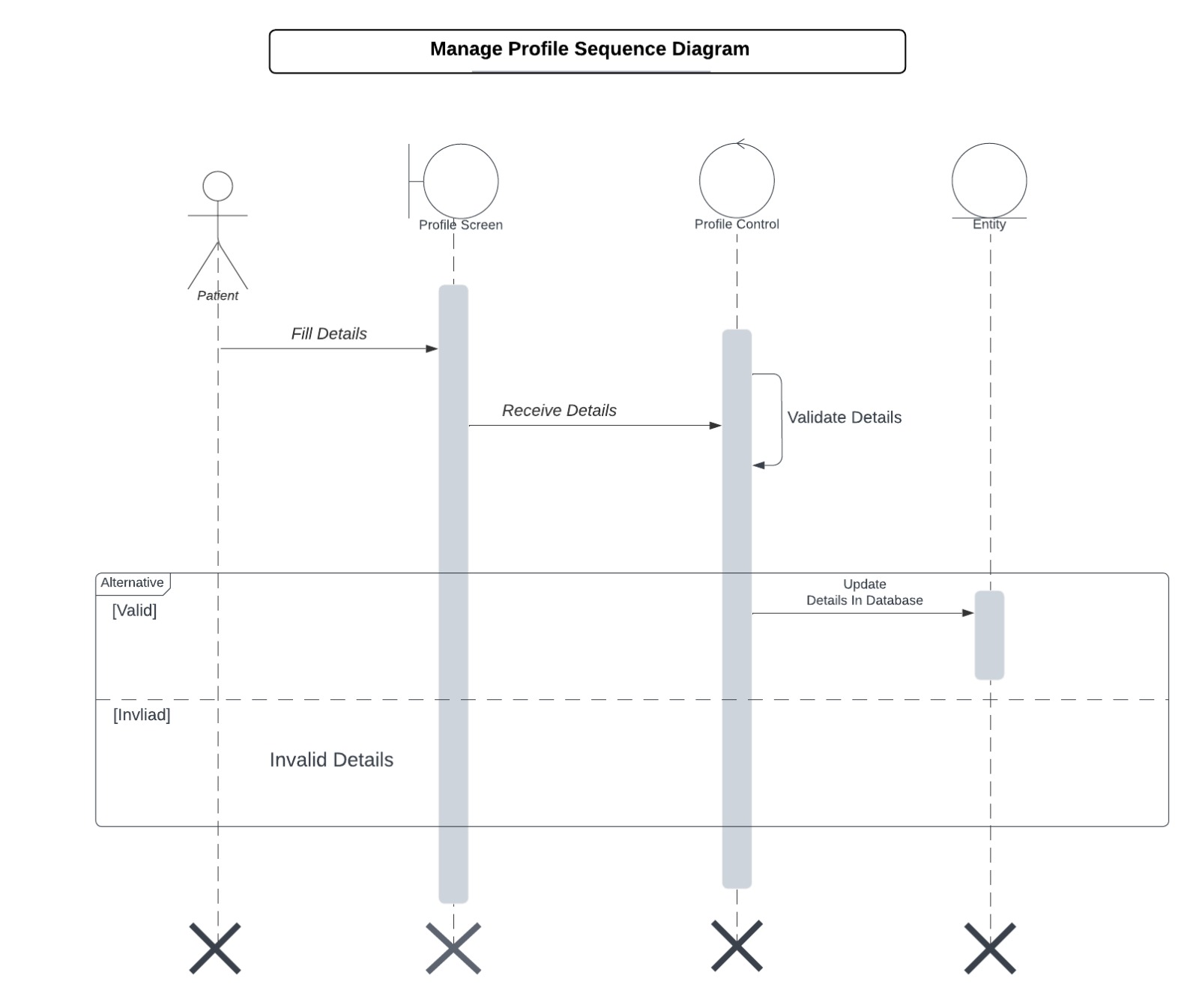
1.Registration



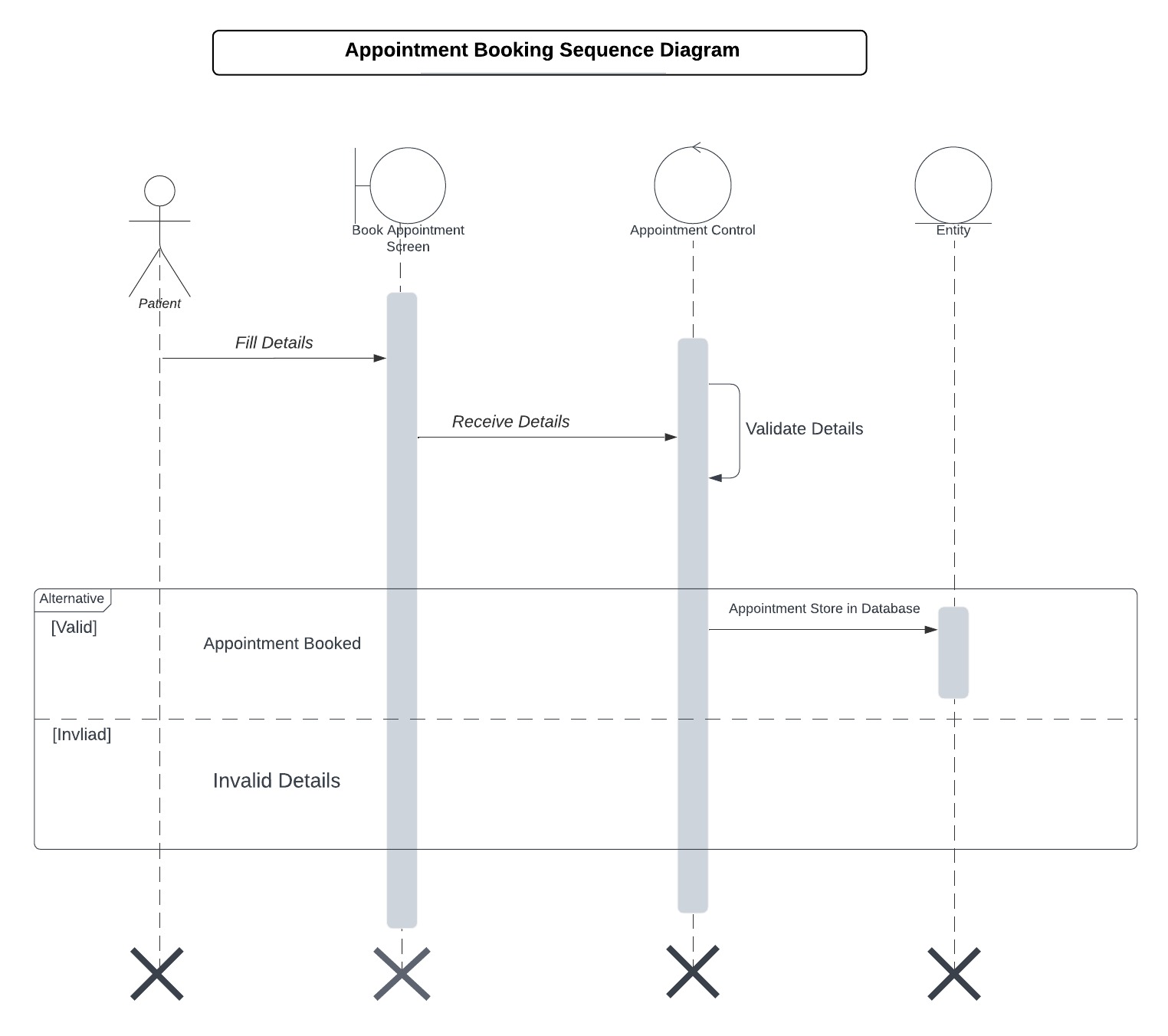
1. Login



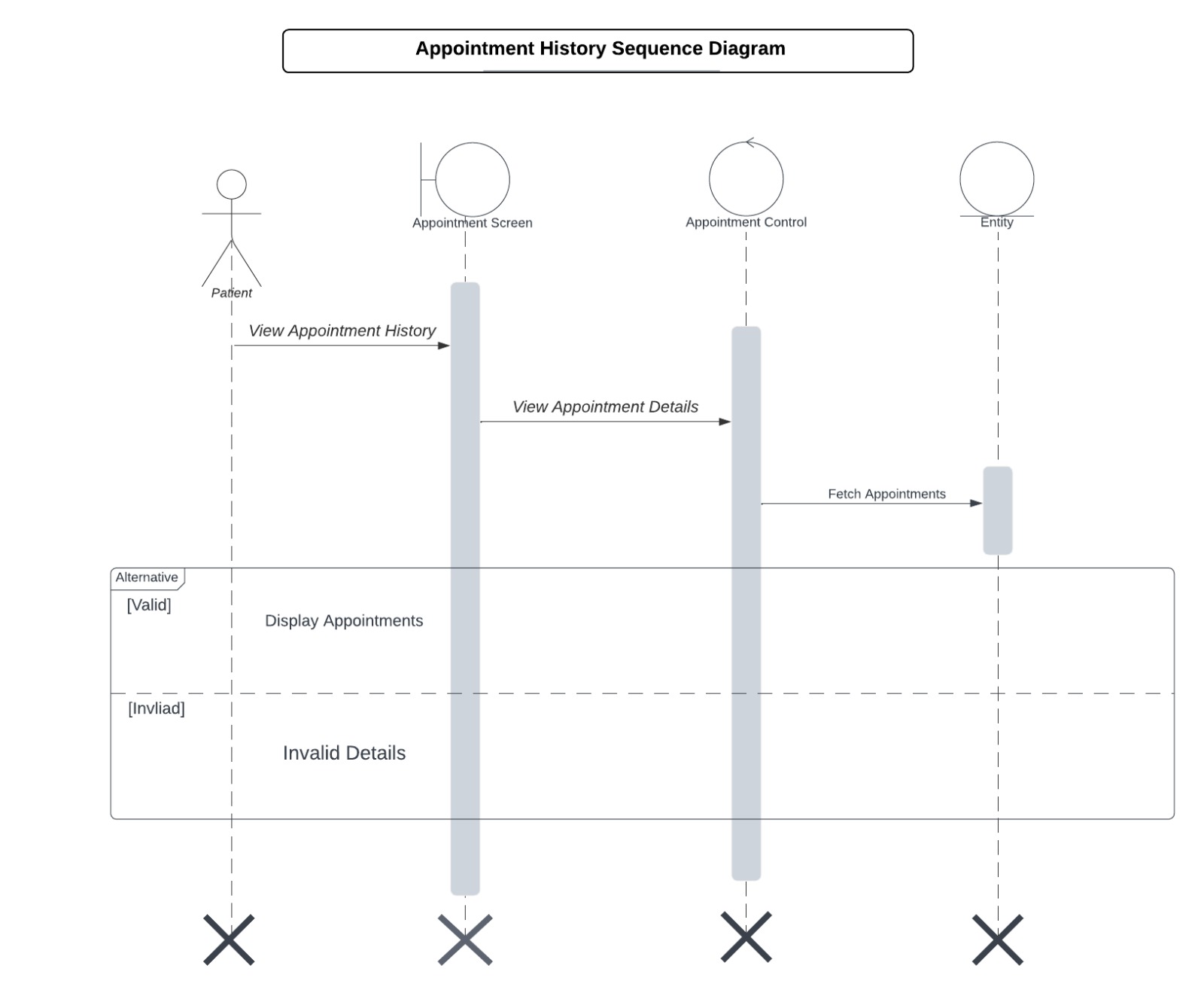
1. Manage Profile



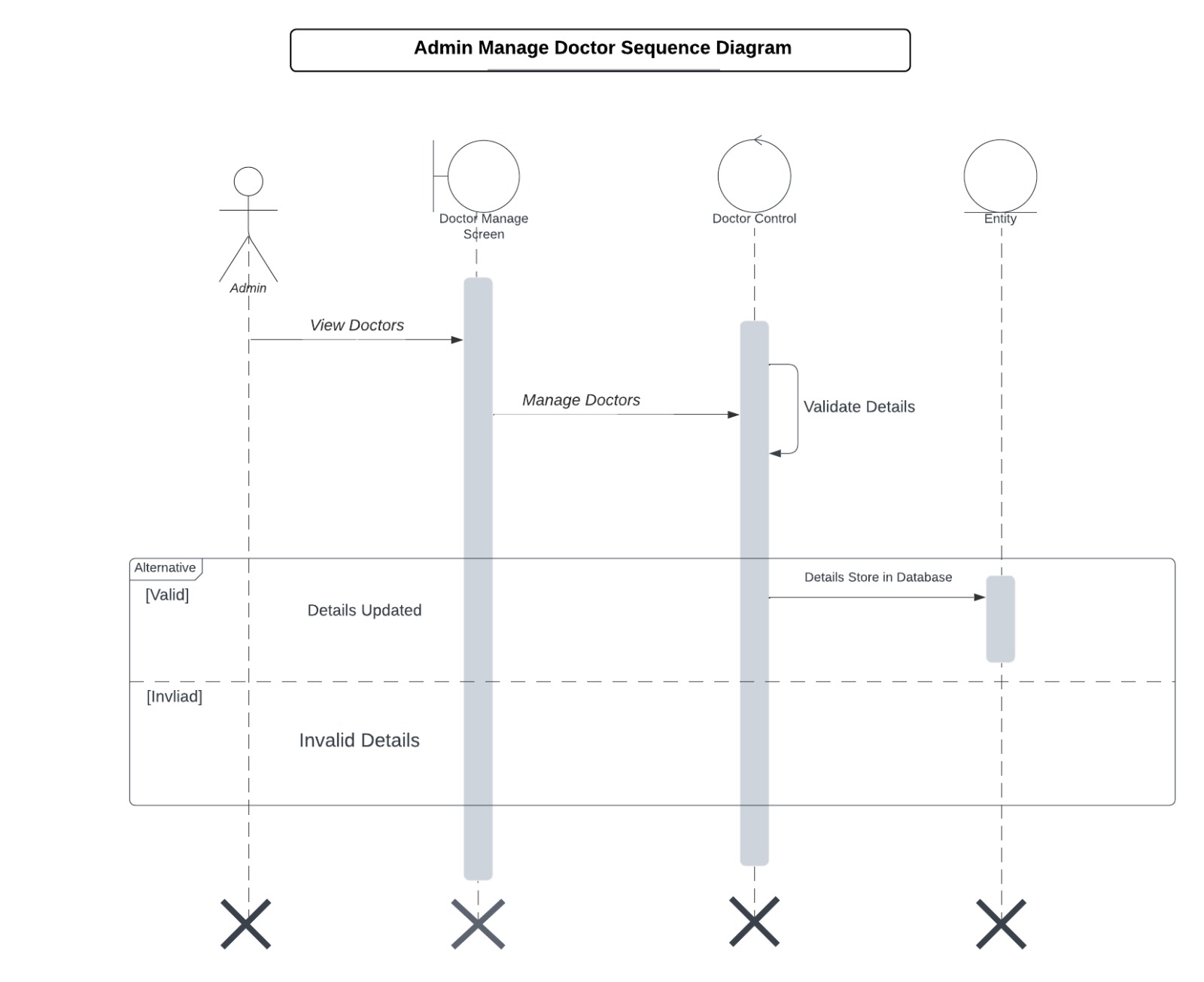
1. Appointment Booking



1. Appointment History

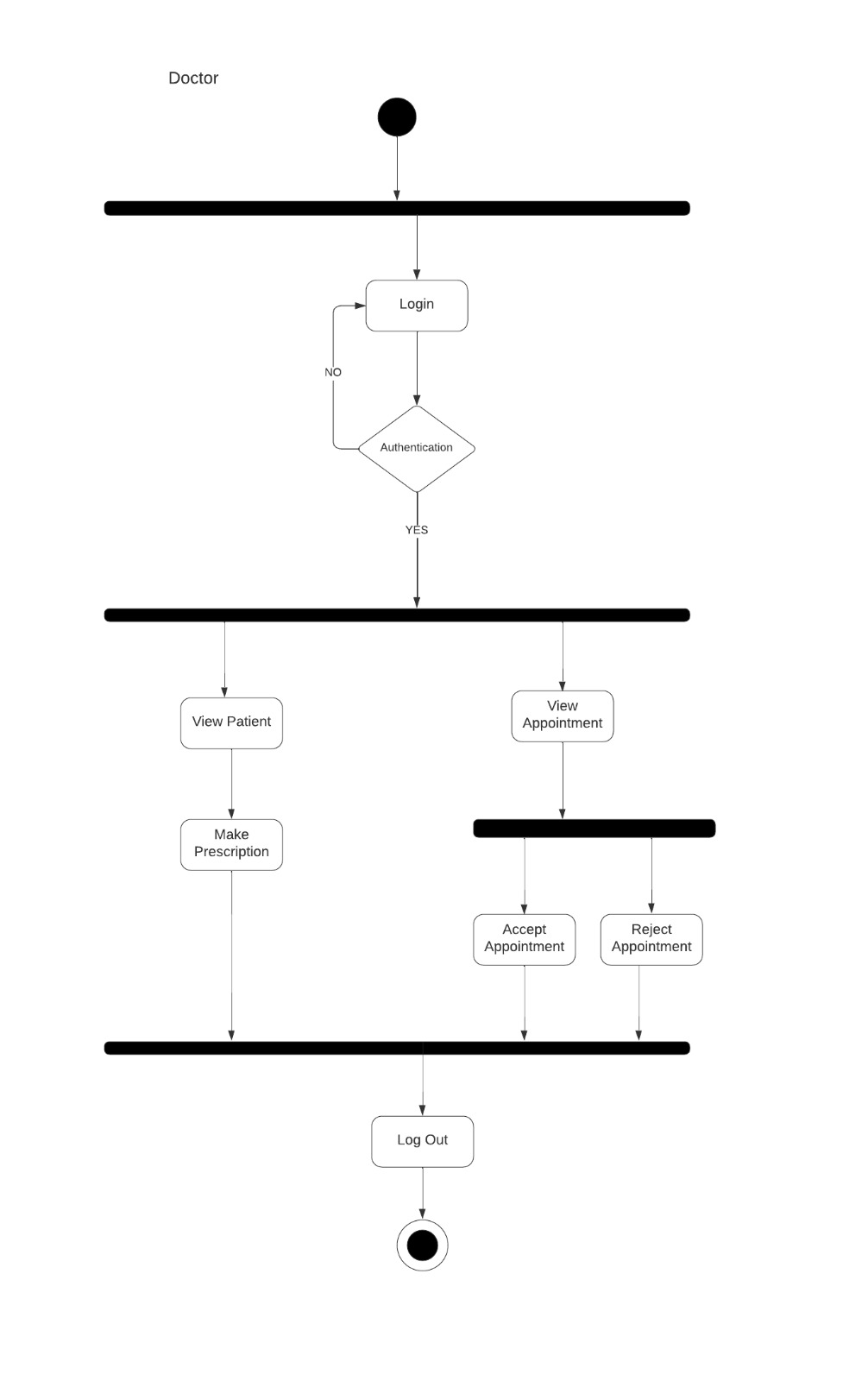


1. Manage Doctor

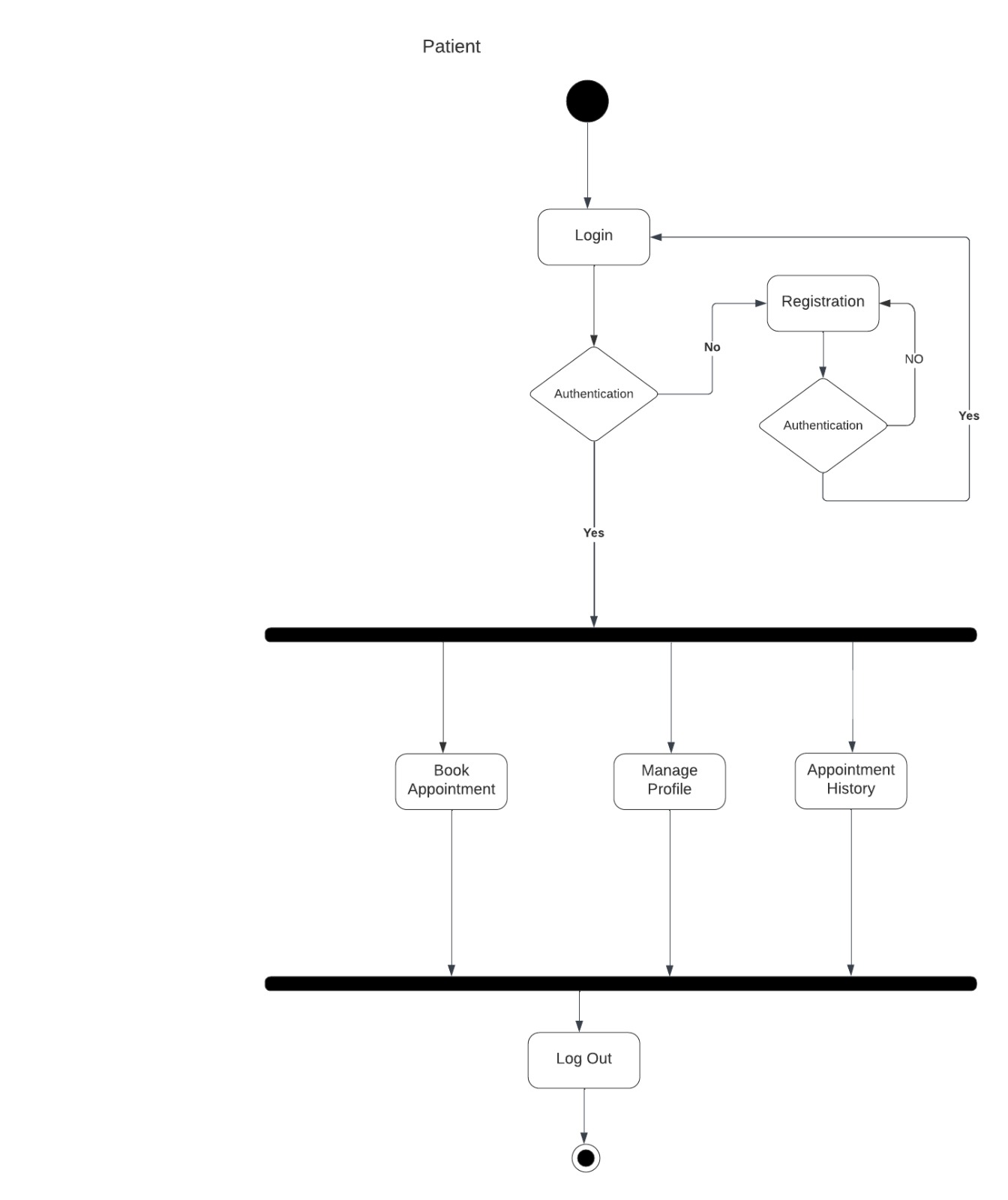


3.4 Activity Diagram

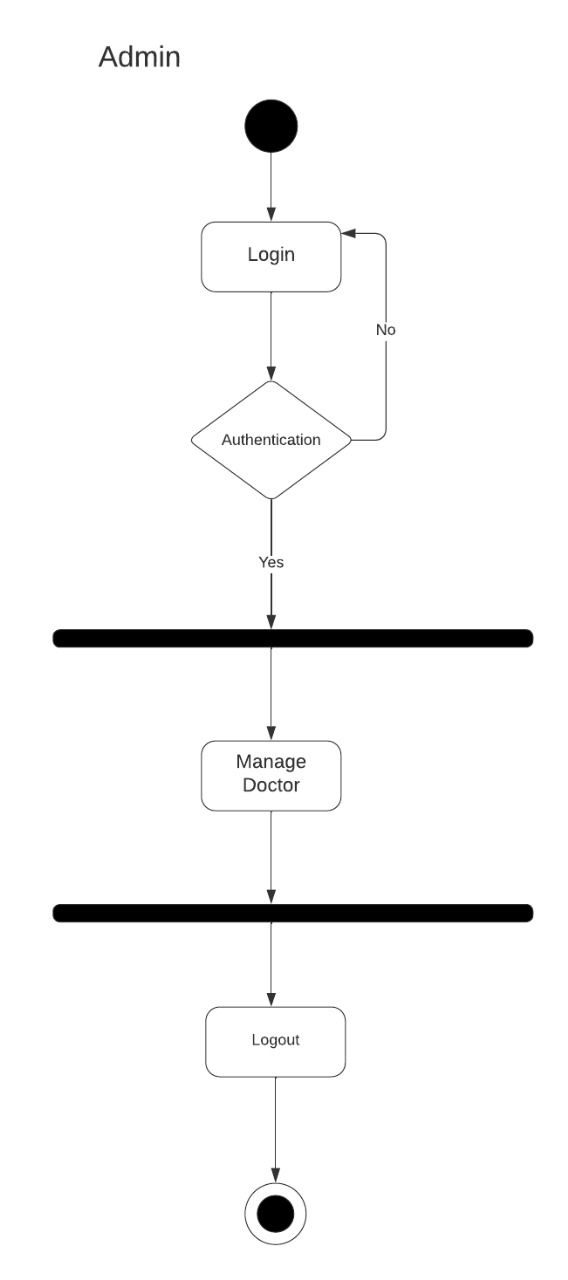
1. Doctor



2. Patient



1. Admin



* 1. Data Dictionary
* Doctor Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| Doctor\_Id | ObjectId | 30 | Primary Key | Store Doctor’s Id |
| Doctor\_Name | String | 20 | Not Null | Store Doctor’s Name |
| Doctor\_Phonenumber | Number | 10 | Not Null | Store Doctor’s Phone Number |
| Doctor\_Email | String | 30 | Not Null | Store Doctor’s Email |
| Clinic\_address | String | 100 | Not Null | Store Doctor’s Clinic Address |
| Clinic\_Name | String | 20 | Not Null | Store Doctor’s Clinic Name |
| Doctor\_RegistrationNumber | Number | 12 | Not Null | Store Doctor’s Registration |
| Specialization | String | 20 | Not Null | Store Doctor’s Specialization |
| Doctor\_Password | String | 20 | Not Null | Store Doctor’s Password |

* Patient Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| Patient\_Id | ObjectId | 30 | Primary Key | Store Patient’s Id |
| Patient\_Name | String | 20 | Not Null | Store Patient’s Name |
| Patient Phonenumber | Number | 10 | Not Null | Store Patient’s Phone Number |
| Patient\_Email | String | 30 | Not Null | Store Patient’s Email |
| Patient\_Address | String | 100 | Not Null | Store Patient’s Address |
| Patient\_Age | String | 3 | Not Null | Store Patient’s Age |
| Patient\_Gender | String | 6 | Not Null | Store Patient’s Gender |
| Patient\_Password | String | 20 | Not Null | Store Patient’s Password |

* Appointme Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| Appointment\_Id | ObjectId | 30 | Primary Key | Store Appointment’s Id |
| Patient\_Id | ObjectId | 30 | Foreign Key | Store Patient’s Id |
| Doctor\_Id | ObjectId | 30 | Foreign Key | Store Doctor’s Id |
| Appointment\_Status | String | 10 | Not Null | Store Appointment Status |
| Appointment\_Description | String | 200 | - | Store Appointment Description (Problem) |
| Appointment\_Date | String | 10 | Not Null | Store Appointment Date |
| Appointment\_Time | String | 10 | Not Null | Store Appointment Time |

* Appointment History Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| Appointment\_Id | Number | 5 | Foreign Key | Store Patient’s Id |
| Doctor\_Id | Number | 5 | Foreign Key | Store Doctor’s Id |
| Patient\_Id | Number | 5 | Foreign Key | Store Patient’s Id |

**4.Development**

4.1 Coding Standards

* Registration Model:

const mongoose = require("mongoose");

const patientModel = new mongoose.Schema({

patientName:{

type:String,

require:true,

},

patientEmail:{

type:String,

require:true,

},

patientPhoneNumber:{

type:Number,

require:true,

},

patientAdd:{

type:String,

require:true,

},

patientAge:{

type:Number,

require:true,

},

patientGender:{

type:String,

require:true,

},

patientPassword:{

type:String,

require:true,

},

type:{

type:String,

default:"patient",

require:true,

}

})

const patient = new mongoose.model("Patient",patientModel);

module.exports= patient;

* Registration Controller:

const patientSignup = async (request, response) => {

try {

const data = request.body;

console.log("data", data);

const isExist = await Patient.findOne({ patientEmail: data.patientEmail })

if (isExist) {

response.status(400).json("User Already Exits");

}

else {

const newPatient = new Patient(data);

//hashing password

const hashedPassword = await hashPassword(newPatient);

newPatient.patientPassword = hashedPassword;

//save user in db

await newPatient.save()

//generate token

const token = await generateAuthToken(newPatient);

console.log("token: ", token);

newPatient.patientPassword = undefined;

const signupPatient = { newPatient, token };

// return { signupPatient };

response.status(201).json(signupPatient);

}

} catch (error) {

console.log("error", error);

    }

}

* Registration Page:

import "react-toastify/dist/ReactToastify.css";

import React, { useEffect, useState } from "react";

import axios from "axios";

import { NavLink, Navigate, useNavigate } from "react-router-dom";

import { ToastContainer, toast } from "react-toastify";

function PatientRegister({ updatePatient, updatePatientToken }) {

const navigate = useNavigate();

const [patient, setPatient] = useState({

patientName: "",

patientEmail: "",

patientPhoneNumber: null,

patientAdd: "",

patientAge: null,

patientGender: "",

patientPassword: "",

});

const referesh = (e) => {

e.preventDefault();

};

const setData = (e) => {

// console.log(e.target.value);

const { name, value } = e.target;

setPatient((preVal) => {

return {

...preVal,

[name]: value,

};

});

};

//register

const patientRegister = async () => {

console.log(patient);

const pass1 = document.getElementById("password").value;

const pass2 = document.getElementById("confirmPassword").value;

try {

if (

pass1 === "" ||

pass2 === "" ||

patient.patientName === "" ||

patient.patientEmail === "" ||

patient.patientPhoneNumber === "" ||

patient.patientAdd === "" ||

patient.patientAge === "" ||

patient.patientGender === "" ||

patient.patientPassword === ""

) {

return toast.error("please fill all the details!", {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "colored",

});

}

// console.log(patient);

if (pass1 === pass2) {

const response = await axios

.post("http://localhost:8888/patientSignup", patient)

.then((res) => {

console.log(res);

updatePatient(res.data.newPatient);

updatePatientToken(res.data.token);

navigate("/patientHome", { replace: true });

return toast.success("Register Successfully", {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "colored",

});

});

} else {

toast.success("Please Enter Both Password Same", {

position: "top-right",

autoClose: 5000,

hideProgressBar: false,

closeOnClick: true,

pauseOnHover: true,

draggable: true,

progress: undefined,

theme: "colored",

});

}

} catch (error) {

console.log("error", error);

}

};

return (

<div>

<nav className="navbar navbar-expand-lg bg-light fixed-top shadow-lg">

<div className="container">

<a className="navbar-brand mx-auto d-lg-none" href="index.html">

Medic Care

<strong className="d-block">Health Specialist</strong>

</a>

<button

className="navbar-toggler"

type="button"

data-bs-toggle="collapse"

data-bs-target="#navbarNav"

aria-controls="navbarNav"

aria-expanded="false"

aria-label="Toggle navigation"

>

<span className="navbar-toggler-icon"></span>

</button>

<div className="collapse navbar-collapse" id="navbarNav">

<ul className="navbar-nav mx-auto">

<a className="navbar-brand d-none d-lg-block">

Doctor Desk

<strong className="d-block">Health Specialist</strong>

</a>

<li className="nav-item">

<a className="nav-link" href="#contact">

Contact

</a>

</li>

<li className="nav-item">

<a className="nav-link">

<NavLink to="/">Login</NavLink>

</a>

</li>

</ul>

</div>

</div>

</nav>

<section className="h-100 h-custom">

<div className="container py-5 h-100">

<div className="row d-flex justify-content-center align-items-center h-100">

<div className="col-lg-8 col-xl-6">

<div className="card rounded-3">

<div className="card-body p-4 p-md-5">

<h3 className="mb-4 pb-2 pb-md-0 mb-md-5 px-md-2">

Patient Registration

</h3>

<form

className="px-md-2"

method="post"

onSubmit={(e) => referesh(e)}

>

<div className="form-outline mb-4">

<label style={{ marginRight: 470 }}>Name</label>

<input

type="text"

className="form-control"

placeholder="Enter Your name"

name="patientName"

value={patient.patientName}

onChange={setData}

required

/>

</div>

<div className="form-outline mb-4">

<label

className="form-label"

style={{ marginRight: 100 }}

htmlFor="form3Example1q"

>

Phone Number

</label>

<input

type="tel"

className="form-control"

placeholder="Enter your phone number"

name="patientPhoneNumber"

value={patient.patientPhoneNumber}

pattern="[6-9]\d{9}"

title="Incorrect Mobile Number"

required

maxlength="10"

onChange={setData}

/>

</div>

<div className="form-outline mb-4">

<label

style={{ marginRight: 470 }}

className="form-label"

htmlFor="form3Example1q"

>

Email

</label>

<input

type="email"

className="form-control"

placeholder="Enter Email Id"

name="patientEmail"

value={patient.patientEmail}

pattern="[a-zA-Z0-9.-\_]{1,}@[a-zA-Z.-]{2,}[.]{1}[a-zA-Z]{2,}"

onChange={setData}

required

/>

</div>

<div className="form-outline mb-4">

<label style={{ marginRight: 470 }}>Address</label>

<input

type="text"

className="form-control"

placeholder="Enter Your Address"

name="patientAdd"

value={patient.patientAdd}

onChange={setData}

required

/>

</div>

<div

className="border mb-4"

style={{ padding: "5px", marginRight: 100 }}

>

<label

className="form-label"

htmlFor="form3Example1q"

style={{ marginLeft: "5px", marginTop: "3px" }}

>

Gender:

</label>

<label>

<input

type="radio"

name="patientGender"

value="Male"

style={{ marginLeft: "25px" }}

onChange={setData}

/>

Male

</label>

<label>

<input

type="radio"

name="patientGender"

value="Female"

style={{ marginLeft: "25px" }}

onChange={setData}

/>

Female

</label>

</div>

<div className="form-outline mb-4">

<label

className="form-label"

style={{ marginRight: 470 }}

htmlFor="form3Example1q"

>

Age

</label>

<input

type="number"

className="form-control"

placeholder="Enter Age"

name="patientAge"

value={patient.patientAge}

onChange={setData}

required

/>

</div>

<div className="form-outline mb-4">

<label

className="form-label"

style={{ marginRight: 470 }}

htmlFor="form3Example1q"

>

Password

</label>

<input

id="password"

type="password"

className="form-control"

placeholder="Enter password"

name="patientPassword"

value={patient.patientPassword}

// pattern="(?=.\d)(?=.[a-z])(?=.[A-Z])(?=.[\W\_]).{8,}"

// title="Must contain at least one uppercase letter, one lowercase letter, one number, one special character, and be at least 8 characters long"

onChange={setData}

required

/>

</div>

<div className="form-outline mb-4">

<label

className="form-label"

style={{ marginRight: 30 }}

htmlFor="form3Example1q"

>

Confirm Password

</label>

<input

id="confirmPassword"

type="password"

className="form-control"

placeholder="Confirm password"

name="confirmPassword"

onChange={setData}

required

/>

</div>

<section className="text-center">

<button

type="submit"

className="btn btn-success btn-lg mb-1"

onClick={() => patientRegister()}

>

Register

</button>

</section>

</form>

</div>

</div>

</div>

</div>

</div>

</section>

{/\* Footer \*/}

<footer className="site-footer section-padding" id="contact">

<div className="container">

<div className="row">

<div className="col-lg-5 me-auto col-12">

<h5 className="mb-lg-4 mb-3">Opening Hours</h5>

<ul className="list-group list-group-flush">

<li className="list-group-item d-flex">

Monday - Friday <br></br>

9:00 AM - 12:30 PM<br></br>

4:00 PM - 7:30 PM

</li>

</ul>

</div>

<div className="col-lg-2 col-md-6 col-12 my-4 my-lg-0">

<h5 className="mb-lg-4 mb-3">Our Clinic</h5>

<p>Parasnagar</p>

</div>

<div className="col-lg-3 col-12 ms-auto mt-4 mt-lg-0">

<p className="copyright-text">Copyright © Docotr Desk 2023 </p>

</div>

</div>

</div>

<ToastContainer />

</footer>

</div>

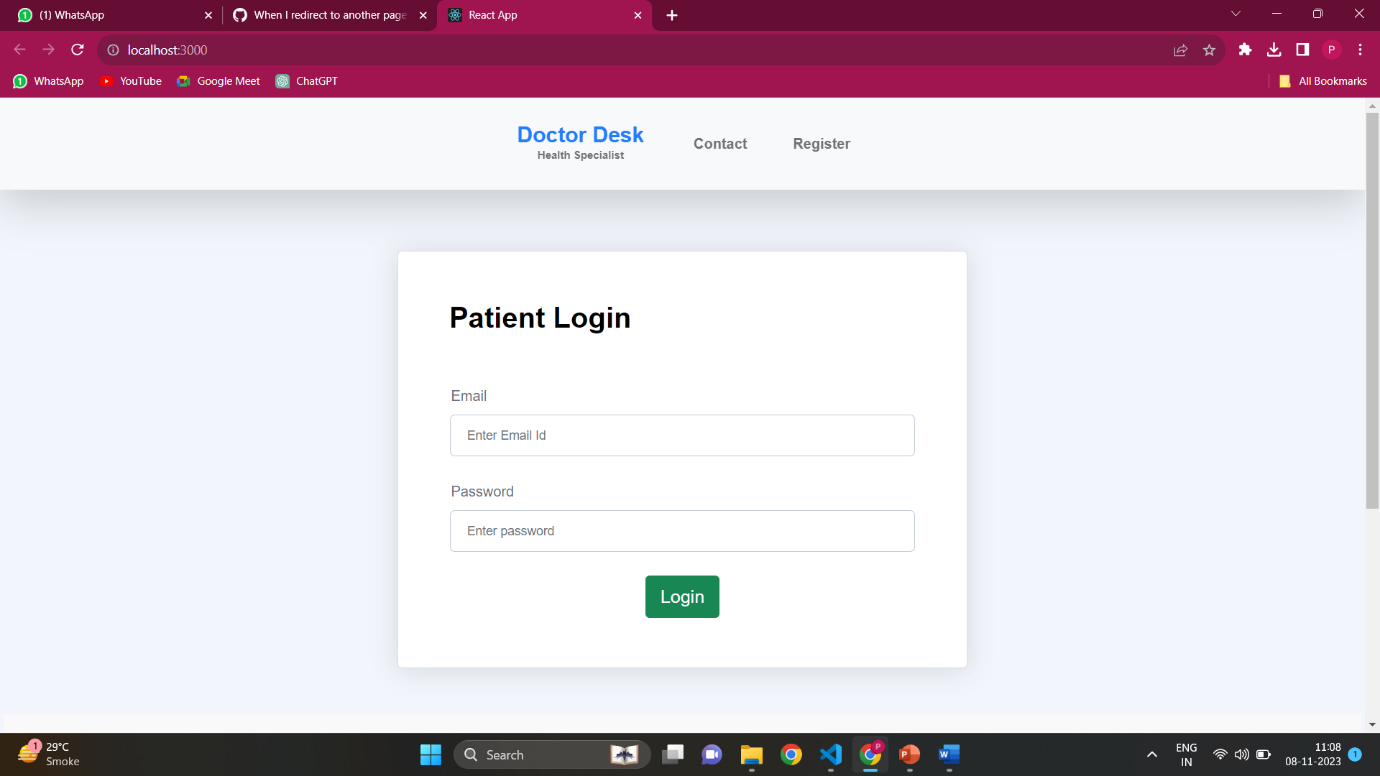
);

}

export default PatientRegister;

4.2 Screen shots

Login



Registration

A screenshot of a computer

Description automatically generated

Patient Home

A person in a white coat

Description automatically generated

Booking Appointment

A computer screen shot of a computer screen

Description automatically generated

Appointment History

A screenshot of a computer

Description automatically generated

Patient Profile

A screenshot of a computer

Description automatically generated

Doctor Login

A screenshot of a computer

Description automatically generated

Doctor Pending Appointment

A screenshot of a computer

Description automatically generated

Doctor Confirm Appointment

A screenshot of a computer

Description automatically generated

Doctor patient Appointment

A screenshot of a computer

Description automatically generated

Doctor Profile

A screenshot of a computer

Description automatically generated

Admin Login

A screenshot of a computer

Description automatically generated

All Doctor Admin

A screenshot of a computer

Description automatically generated

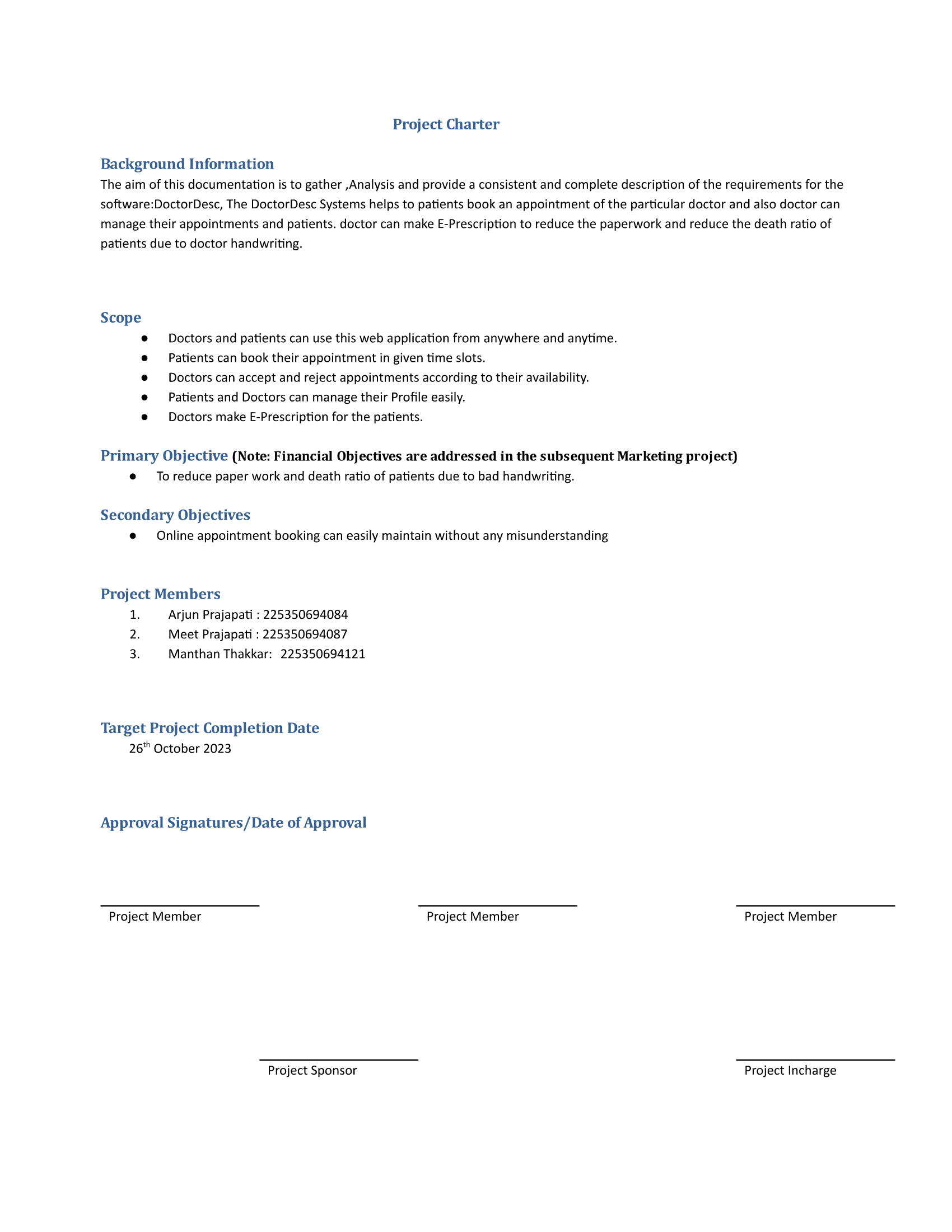
Add Doctor

A screenshot of a computer

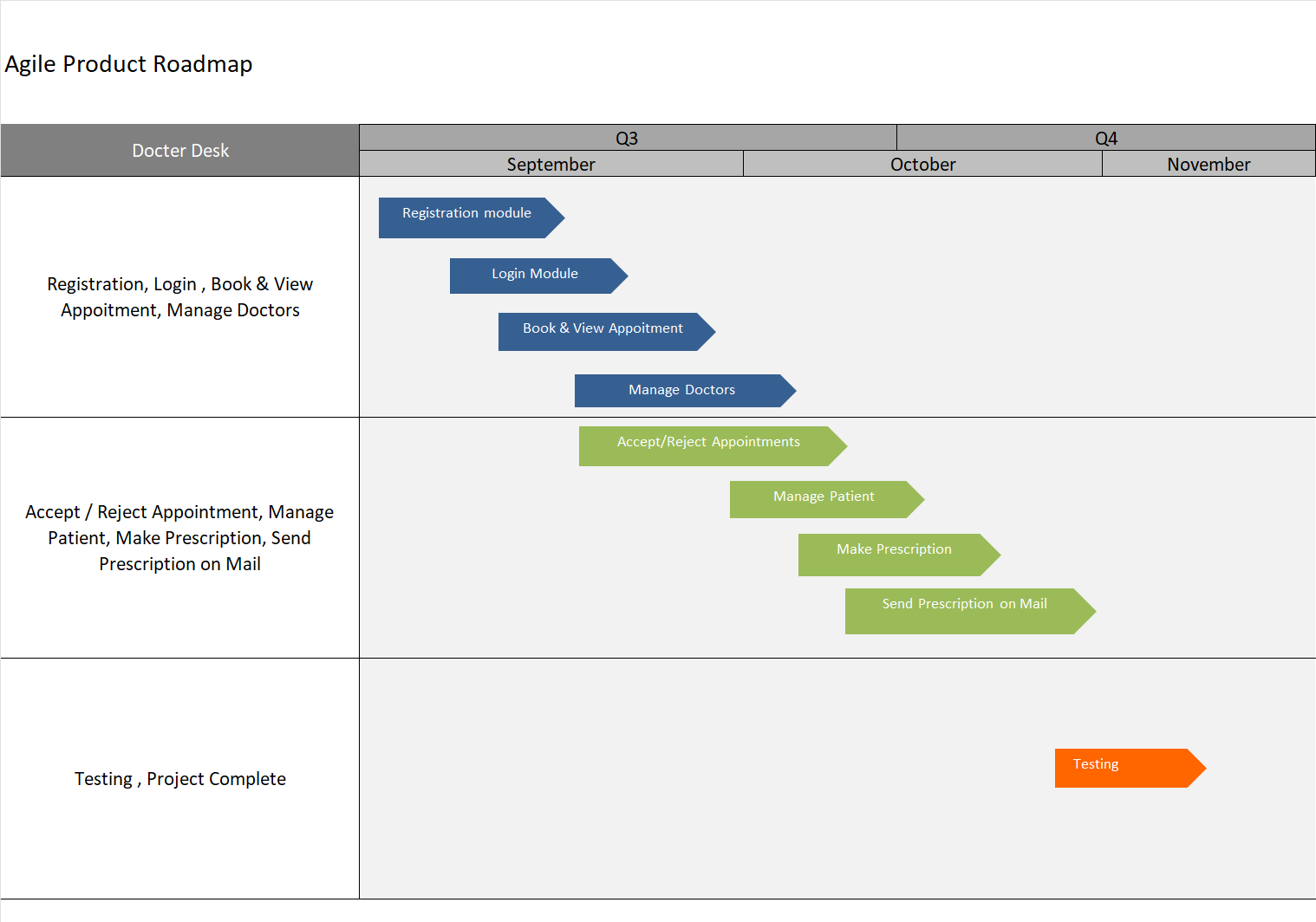
Description automatically generated

**5.Agile Documentation**

5.1 Agile Project Charter



5.2 Agile Roadmap/ Schedule



5.3 Agile Project Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Name | Project Manager | Start Date | End Date | Overall progress | Project Deliverable |
| Docter Desk | Prof . Tinal Parikh | 08-09-2023 | 26-10-2023 | 100% | Yes |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TASK NAME** | **RESPONSIBLE** | **START** | **FINISH** | **DURATION** in days | **STATUS** |
| Registration Module |  | 08-09-2023 | 09-09-2023 | 2 |  |
| Analysis | Arjun | 08-09-2023 | 08-09-2023 | 1 | Complete |
| Design | Meet | 08-09-2023 | 08-09-2023 | 1 | Complete |
| Coding | Meet | 09-09-2025 | 09-09-2025 | 1 | Complete |
| Login |  | 08-09-2023 | 09-09-2023 | 2 |  |
| Analysis | Arjun | 08-09-2023 | 08-09-2023 | 1 | Complete |
| Design | Meet | 08-09-2023 | 08-09-2023 | 1 | Complete |
| Coding | Meet | 09-09-2025 | 09-09-2025 | 1 | Complete |
| Manage Doctor |  | 08-09-2023 | 09-09-2023 | 2 |  |
| Analysis | Manthan | 08-09-2023 | 08-09-2023 | 1 | Complete |
| Design | Meet | 08-09-2023 | 08-09-2023 | 1 | Complete |
| Coding | Arjun | 09-09-2025 | 09-09-2025 | 1 | Complete |
| Manage Profile |  | 15-09-2023 | 18-09-2023 | 4 |  |
| Analysis | Manthan | 15-09-2023 | 15-09-2023 | 1 | Complete |
| Design | Meet | 16-09-2023 | 16-09-2023 | 1 | Complete |
| Coding | Arjun | 17-09-2023 | 18-09-2023 | 2 | Complete |
| Book Appointment |  | 15-09-2023 | 18-09-2023 | 4 |  |
| Analysis | Meet | 15-09-2023 | 15-09-2023 | 1 | Complete |
| Design | Arjun | 16-09-2023 | 16-09-2023 | 1 | Complete |
| Coding | Manthan | 17-09-2023 | 18-09-2023 | 2 | Complete |
| Appointment History |  | 18-09-2023 | 19-09-2023 | 2 |  |
| Analysis | Arjun | 18-09-2023 | 18-09-2023 | 1 | Complete |
| Design | Meet | 19-09-2023 | 19-09-2023 | 1 | Complete |
| Coding | Manthan | 19-09-2023 | 19-09-2023 | 1 | Complete |
| Manage Appointments |  | 24-09-2023 | 27-09-2023 | 4 |  |
| Analysis | Arjun | 24-09-2023 | 25-09-2023 | 2 | Complete |
| Design | Meet | 25-09-2023 | 25-09-2023 | 1 | Complete |
| Coding | Manthan | 26-09-2023 | 27-09-2023 | 2 | Complete |
| View Patient |  | 20-10-2023 | 21-10-2023 | 2 |  |
| Analysis | Arjun | 20-10-2023 | 20-10-2023 | 1 | Complete |
| Design | Meet | 20-10-2023 | 20-10-2023 | 1 | Complete |
| Coding | Arjun | 21-10-2023 | 21-10-2023 | 1 | Complete |
| Manage Patients |  | 23-10-2023 | 26-10-2023 | 4 |  |
| Analysis | Arjun | 23-10-2023 | 23-10-2023 | 1 | Complete |
| Design | Meet | 24-10-2023 | 24-10-2023 | 1 | Complete |
| Coding | Manthan | 25-10-2023 | 26-10-2023 | 2 | Complete |

5.4 Agile User Story

|  |  |  |  |
| --- | --- | --- | --- |
| No | TYPE OF USER | I WANT TO PERFORM | RESULT |
| 1 | Patient | Registration | I can do registration. |
| 2 | Patient | Login | I can access the system |
| 3 | Patient | Profile Update | I can update my profile. |
| 4 | Patient | Book Appointment | I can book an appointment. |
| 5 | Patient | Select Date and Time Slot | I can select date and time slot. |
| 6 | Patient | View Appointment History | I can view my appointment History. |
| 7 | Doctor | Login | I can access the system. |
| 8 | Doctor | Manage Appointment | I can accept and reject patient’s Appointment. |
| 9 | Doctor | View Patient | I can view my patient. |
| 10 | Doctor | Make Prescription | I can make prescription for patient. |
| 11 | Admin | Login | I can access the system. |
| 12 | Admin | Manage Doctors | I can manage doctors. |

5.4 Agile Release Plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint | Task | Start | End | Duration | Status | Release Date |
| Patient & Doctor & Admin Registration, Login and Manage Doctors | Registration | 08-09-2023 | 09-09-2023 | 1 | Released | 09-09-2023 |
| Login | 08-09-2023 | 09-09-2023 | 1 | Released | 09-09-2023 |
| Manage Doctors | 08-09-2023 | 09-09-2023 | 1 | Released | 09-09-2023 |
| Patient Manage Profile, Book Appointment, Doctor Check Appointments | Manage Profile | 15-09-2023 | 18-09-2023 | 4 | Released | 18-09-2023 |
| Book Appointment | 15-09-2023 | 18-09-2023 | 4 | Released | 18-09-2023 |
| Check Appointments | 24-09-2023 | 27-09-2023 | 4 | Released | 18-09-2023 |
| Manage Patient, View Patient, Make Prescription, Receive Prescription | View & Manage Patient | 20-10-2023 | 21-10-2023 | 2 | Released | 21-10-2023 |
| Manage Prescription | 23-10-2023 | 25-10-2023 | 3 | Released | 25-10-2023 |
| Receive Prescription | 25-10-2023 | 26-10-2023 | 2 | Released | 26-10-2023 |

5.6 Agile Sprint Backlog

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task Name | Responsible | Story | Sprint Ready | Days | Priority | Status |
| Registration | Arjun , Meet | Yes | Yes | 2 | Low | Complete |
| Login | Arjun , Meet | Yes | Yes | 2 | Low | Complete |
| Manage Profile | Manthan, Meet , Arjun | Yes | Yes | 4 | Medium | Complete |
| Appointment Booking | Manthan, Meet , Arjun | Yes | Yes | 4 | High | Complete |
| Appointment History | Manthan, Meet , Arjun | Yes | Yes | 2 | High | Complete |
| Manage Appointments | Manthan, Meet , Arjun | Yes | Yes | 4 | High | Complete |
| View Patients | Arjun, Meet | Yes | Yes | 2 | High | Complete |
| Make Prescription | Manthan | Yes | Yes | 5 | High | Complete |
| Manage Patient | Manthan, Meet , Arjun | Yes | Yes | 2 | High | Complete |
| Manage Doctors | Manthan, Meet , Arjun | Yes | Yes | 2 | Medium | Complete |

5.7 Agile Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Project Name | Doctor Desk | Browser: | Chrome |
| Test Case ID | 1 | **Version:** | 1 |
| Written By: |  | **Description:** |  |
| Tested By: |  | **Tested On:** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test No | Date | Action | Expected Results | Actual Results | Pass? |
| 1 | 09-09-2023 | Registration of Patient, Doctor & Admin | Should go to Registration Screen | User will get the Registration Screen |  |
| 2 | 09-09-2023 | Login of Patient, Doctor & Admin | Should go to the Login Screen | User will get the Login Screen |  |
| 3 | 19-09-2023 | Manage Profile | Should go to the update profile page | Patient will update his/her profile. |  |
| 4 | 19-09-2023 | Book Appointment | Should go to the booking page | Patient will book an appointment |  |
| 5 | 19-09-2023 | View History | Take him/her to Booking History Screen | Patient get the Booking History Screen |  |
| 6 | 28-09-2022 | Manage Appointment | Take Doctor to his/her Appointment Screen | Doctor will Accept and Reject the appointments |  |
| 7 | 21-10-2023 | View Patient | Should go to the View Patient Page | Doctor get his/her all patient list. |  |
| 8 | 25-10-2022 | Make Prescription | Should go to the Make Prescription Page | Doctor will make patient’s prescription |  |
| 9 | 26-10-2022 | Manage Doctors | Show him/her to inserted & delete doctor’s details | Admin get the doctors details |  |

**6.Proposed Enhancements**

* **Real-time Appointment Scheduling**: Implement a real-time scheduling system where patients can see doctors' availability and book appointments instantly. This can help reduce waiting times and improve the overall user experience.
* **Notifications and Reminders**: Set up automated reminders for upcoming appointments, prescription refills, or test results. These notifications can be sent via email, SMS, or push notifications to keep patients and doctors informed and engaged.
* **Online Prescription and Refill Requests**: Allow patients to request prescriptions or refills through the app, saving them time and effort. Doctors can review and approve these requests digitally, streamlining the process.
* Implement a broadcast messaging feature for doctors to send common announcements to all patients in your web app. This enables efficient and convenient communication of important updates, event notifications, and general health tips. By sending a single message to all patients simultaneously, doctors save time and ensure timely delivery of information, enhancing engagement and communication.

1. **Conclusion**

* The Doctor Desk Website is a user-friendly solution that enables patients to easily sign up, log in, and manage their profiles. With the site, patients can conveniently book appointments with doctor. Doctors can securely log in, manage appointments, and access patient information. Overall, the Doctor Desk site simplifies appointment management, improves communication, and enhances the patient experience.

**8. Bibliography**

* <https://dribbble.com/>
* <https://stackoverflow.com/>
* <https://www.w3schools.com/django/index.php>
* <https://www.mongodb.com/>
* <https://www.npmjs.com/>