Project Report

for

CS 837 - Healthcare Application Development

Project 8

HIS with user-friendly Doctor App

Milestone 4

Prepared by Team 30:

MT2023005	Bhumika Jindal
MT2023017	Abhipsa Panda
MT2023024	Kakunuri Himarishitha Reddy
MT2023086	Charvy Koshta
MT2023119	Deepanjali Ghosh

Planned Scope For the Project:

For a healthcare application that involves multiple user roles such as Doctor, Nurse, Receptionist, Pharmacy, and Admin, the planned scope of the project would comprehensively cover functionalities tailored to each role, ensuring efficient workflow, data privacy, and enhanced patient care. Here’s a detailed breakdown of the scope for each module:

1. Admin Module

- **User Management**: Capability to add, modify, and deactivate user accounts for doctors, nurses, receptionists, and pharmacy staff.
- **Role-Based Access Control:** Define and manage access permissions for different user roles to ensure data security and operational efficiency.

2. Receptionist Module

- **Patient Registration and Management:** Register new patients, update existing patient profiles, and manage patient data.
- **Appointment Scheduling:** Schedule, reschedule, and cancel appointments. Manage appointment calendars for doctors.
- **Consent Management:** Obtain and manage patient consents as detailed earlier, including verification through OTP and handling of consent withdrawal or modification.

3. Nurse Module

- **Login Validation Process:** Nurse will be able to log in only during their scheduled hours otherwise rejected login.
- **Patient Vitals and Symptoms Recording:** Input and update patient vitals, symptoms, and other relevant health indicators.
- **Medical History Management:** Update and maintain detailed patient medical histories, including past treatments and outcomes.
- **Test Result Entry:** Enter and manage test results for various diagnostics performed, as prescribed by doctors.

4. Doctor Module

- **Patient Consultation and Diagnosis:** Access patient vitals, symptoms, and medical history to diagnose conditions.
- **Treatment and Prescription Management:** Prescribe medications and recommend tests. Review test results to adjust treatment plans as necessary.
- **Patient Rounds Management:** Manage and record details of patient rounds, including patient progress notes and future care planning.

5. Pharmacy Module

- **Medication Dispensing:** Process prescriptions from doctors, dispense medications to patients, and manage medication inventory.

API Endpoints:

Admin Module:

1.**"/admin/addDoctor/{specialization}**"

a) **Input**: name, age, sex, qualification,department,contact, photo in Request body and specialization as path variable .

b) **Output**: Doctor Object.

c) **Description:** This API is used to register a new doctor.

2. "**/admin/addNurse**"

a) **Input:** name,age,sex,contact number, List<NurseSchedule> in request body

b) **Output:** Nurse object

c) **Description:** This API is used to register new nurse.

3. "**/admin/viewDoctors/{department}**"

a) Input:

b) **Output:** List of all the Doctors according to the department selected.

c) **Description:** This API is used to view the list of doctors working in the hospital.

4. “**/admin/viewNurses**”

a) **Input:**
b) **Output:** Nurse details of all the nurses.
c) **Description:** This API is used to view all the nurses working in the hospital.

5.”**/admin/addPharmacy**”

a) **Input:** name,address,contact,active,licenseNumber as request body.
b) **Output:**Pharmacy Object
c) **Description:** This API is used to register a new Pharmacy.

6. **“/admin/viewPharmacies**”

a) **Input :**
b) **Output :** List of all the pharmacies in the hospital.
c) **Description:** This API is used to view the list of pharmacies in the hospital .

7. ”**/admin/addReceptionist**”

a) **Input:** name,age,sex,email,contact,active,photo as request body.
b) **Output:** Receptionist object
c) **Description:** This API is used to register a new Receptionist.

8. **“/admin/viewReceptionists**”

a) Input:
b) **Output:** List of all the receptionists working in the hospital.
c) **Description:** This API is used to view the list of receptionists working in the hospital.

9. **“/admin/patientCount**”

a) Input:
b) **Output:** Count of all the patients registered to the hospital.
c) **Description:** This API is used to view the count of all the patients registered to the hospital.

10**. “/admin/nurseCount”**

a) Input:
b) **Output:** Count of all the nurses registered to the hospital.
c) **Description:**This API is used to view the count of all the nurses registered to the hospital and are active.

11. **“/admin/doctorCount”**
   a) **Input:**
   b) **Output:** Count of all the patients registered to the hospital.
   c) **Description:** This API is used to view the count of all the patients registered to the hospital and are active.

12. **“/admin/editDoctor/{did}”**
   a) **Input:** name, age, sex, qualification, specialization,department,contact, photo as request body and DoctorID as path variable
   b) **Output:** Doctor object
   c) **Description :** This API lets us edit the existing doctor’s details having the doctorId as did path variable.

13. **“/admin/editNurse/{nid}”**
   a) **Input:** name,age,sex,contact number,List<NurseSchedule> as request body and NurseID as path variable.
   b) **Output:** Nurse object.
   c) **Description:** This API lets us edit the existing nurse’s details having the nurseId as nid path variable.

14. **“/admin/editReceptionist/{rid}”**
   a) **Input:** name,age,sex,email,contact,photo as request body and receptionistID as path variable.
   b) **Output:** Receptionist object
   c) **Description:** This API lets us edit the existing receptionist’s details having the receptionistId as rid path variable.

15**. “/admin/editPharmacy/{phid}**”
   a) **Input:** name,address,contact,active,licenseNumber as request body and pharmacyID as path variable.
   b) **Output**:Pharmacy Object
   c) **Description :** his API lets us edit the existing pharmacy’s details having the pharmacyId as phid path variable.

16. **“/admin/deactivateDoctor/{doctorId}**
   a) **Input:** DoctorID as path variable
   b) **Output:** Success or failure message of the operation.

c) **Description:** This API is used to deactivate a doctor’s ID once he/she leaves the hospital.

17. **“/admin/deactivateNurse/{nurseId}”**
    a) **Input:** NurseID as path variable.
    b) **Output:** Success or failure message of the operation.
    c) **Description:** This API is used to deactivate a nurse’s ID once he/she leaves the hospital.

18. **“/admin/deactivateReceptionist/{recepId}”**
    a) **Input:** ReceptionistID as path variable.
    b) **Output:** Success or failure message of the operation.
    c) **Description:** This API is used to deactivate a receptionist’s ID once he/she leaves the hospital.

19. **“/admin/deactivatePharmacy/{pharmaId}”**
    a) **Input:** PharmacyID as path variable.
    b) **Output:** Success or failure message of the operation.
    c) **Description:** This API is used to deactivate a pharmacy’s ID once he/she leaves the hospital.

20. **“/admin/login”**
    a) **Input:** email,password as request body.
    b) **Output:** String indicating successful login or failure.
    c) **Description:** This API is used for admin authentication. It takes the admin’s login credentials and verifies them.

21. **“/admin/addAdmin”**
    a) **Input:** admin name,password,email as request body.
    b) **Output:** Admin object.
    c) **Description:** This API is used for adding the admin.

22. **“/admin/viewSpecialization”**
    a) **Input:**
    b) **Output:** List of Specializations.
    c) **Description**: This API is used to get all specializations of doctors.

23. **“/admin/addSpecializations”**
    a) **Input:**Specialization name in Request Body.
    b) **Output**: Specialization Object
    c) **Description:** This API is used to add new specialization.

24. **“/admin/viewDoctor/{doctorId}”**
    a) **Input:** DoctorId as path variable
    b) **Output**: Doctor object
    c) **Description**: This API lets us view a particular doctor's details.

25. **“/admin/viewReceptionist/{receptionistId}”**
    a) **Input**: receptionistID as path variable
    b) **Output**: Receptionist object
    c) **Description**: This API lets us view a particular receptionist’s details.

26. **“/admin/viewNurse/{nurseId}”**
    a) **Input**: nurseId as path variable
    b) **Output**: Nurse object
    c) **Description**: This API lets us view a particular nurse’s details.

27. **“/admin/viewPharmacy/{pharmacyId}”**
    a) **Input**: pharmacyID as path variable
    b) **Output**: Pharmacy object
    c) **Description**: This API lets us view a particular pharmacy’s details.

28. **“/admin/getHospitalDetails”**
    a) **Input:**
    b) **Output**: hospital object
    c) **Description**:This API lets us view the hospital’s details.

29. **“/admin/logout/{email}”**
    a) **Input**:
    b) **Output**: String stating logout logout status.
    c) **Description**: Logouts the user having email passed as path variable.
30. **“/admin/pharmacyCount”**
    a) Input:
    b) **Output:** Count of all the nurses registered to the hospital.

c) **Description:**This API is used to view the count of all the pharmacy registered to the hospital and are active.

31. "/admin/receptionistCount"
    a) Input:
    b) **Output:** Count of all the nurses registered to the hospital.
    c) **Description:**This API is used to view the count of all the receptionist registered to the hospital and are active.

Nurse Module:

1. **"/nurse/login"**
    a. **Input** : Nurse's email id and password
    b. **Output**: String stating login failure or successful
    c. **Description**: This API checks if email and password entered by nurse matches with the record in backend or not and checks whether nurse is active and is trying to login in scheduled time or not,if all requirements met then nurse is allowed to login.
2. **"/nurse/getNurseDetailsByEmail/{email}"**
    a. **Input** : Email is sent as path Variable
    b. **Output**: Nurse Object
    c. **Description**: This API retrieves a specific nurse object based on the nurse email provided.
3. **"/nurse/getEmergencyPatients"**
    a. **Input** : No content
    b. **Output**: A list of patient Objects
    c. **Description**: This API retrieves a list of all emergency patients who are currently not discharged from the hospital
4. **"/nurse/getAllPatients"**
    a. **Input** : No content
    b. **Output**: A list of Patient Objects
    c. **Description**: This API retrieves a list of all normal patients who are currently not discharged from the hospital
5. **"/nurse/getPatientDetailsById/{patientId}"**
    a. **Input** : PatientId is sent as a path variable
    b. **Output**: Patient Object
    c. **Description**: This API retrieves a specific patient object based on the patientId provided

6. **“/nurse/addVitals/{patientId}”**
    a. **Input**: weight,height,blood pressure,SPO2,pulse,temperature in request body and patientId as pathvariable
    b. **Output**: Vital Added successfully
    c. **Description**: This API add vital for a specific patient based on patientId provided
7. **“/nurse/editVitals/{vitalid}”**
    a. **Input**: Updated weight,height,blood pressure,SPO2,pulse,temperature in request body and vitalid as path variable
    b. **Output**: Vitals Edited Successfully
    c. **Description**: This API allows nurse to edit a specific vital entry
8. **“/nurse/viewVitals/{patientId}/{consenttoken}”**
    a. **Input**: PatientId,consent token is sent as a path variable
    b. **Output**: a Vital Object
    c. **Description**: This API allows nurse to view the vitals of a specific patient if the consent token is valid
9. **“/nurse/deleteVitals/{vitalid}”**
    a. **Input**:VitalId as path variable
    b. **Output**: No content
    c. **Description**: This API allows nurse to delete an incorrect vitals entry
10. **“/nurse/viewVitalsById/{patientId}/{vitalId}”**
    a. **Input:** PatientId and VitalId is sent as Path Variable
    b. **Output**:The previously existed vitals detailed for the patient
    c. **Description** :This API helps in displaying previously entered vitals details for the patient while editing the previously existed vitals details.
11. **“/nurse/addSymptoms/{patientId}”**
    a. **Input** : symptom1,symptom2,symptom3,symptom4,symptom5 as request body and patientId as a path variable
    b. **Output**: Symptom Added successfully
    c. **Description**: This API allows nurse to add symptoms for a specific patient based on patientId provided
12. **“/nurse/editSymptoms/{symptomid}”**
    a. **Input**: Updated symptom1,symptom2,symptom3,symptom4,symptom5 as request body and symptomid as path variable
    b. **Output**: Symptom Edited successfully
    c. **Description**: This API allows nurse to edit a specific symptom entry
13. **“/nurse/viewSymptoms/{patientId}/{consenttoken}”**

   a. **Input** : PatientId and Consent Token is provided as path variable
   b. **Output**: a Symptom Object
   c. **Description**: This API allows nurse to see symptoms of a particular patient if the consent token is valid
14. **“nurse/deleteSymptoms/{symptomid}”**
   a. **Input**: Symptomid is provided as a path variable
   b. **Output**: No content
   c. **Description**:This API allows nurse to delete incorrect symptom entry
15. **“nurse/viewSymptomsById/{symptomid}/{patientId}”**
   a. Input : Symptomid and PatientId are sent as path variable
   b. **Output**: previously existed Symptom Object
   c. **Description**: This API helps in displaying previously entered symptoms details for the patient while editing the previously existed vitals details.
16. **“/nurse/addPastHistory/{patientId}”**
   a. **Input**: disease,medicine,dosage,remarks,recordedAt(local date) as request body and patientId as a path variable
   b. **Output**: Past History Added successful
   c. **Description**: This API allows nurse to add past history of a specific new patient(who previously had treatment in other hospital) based on the patientId provided
17. **“/nurse/editPastHistory/{historyId}”**
   a. **Input** : Updated disease,medicine,dosage,remarks,recordedAt(local date) as request body and patientId as a path variable
   b. **Output**: Past History edited successfully
   c. **Description**: This API allows nurse to edit a specific past history
18. **“/nurse/viewPastHistory/{patientId}/{consenttoken}”**
   a. **Input**: PatientId is sent as path variable
   b. **Output**: A list of past history objects
   c. **Description**: This API allows nurse to view all past history (added by him/her)of the specific patient if the consent token is valid
19. **“/nurse/deletePastHistory/{historyId}”**
   a. **Input**: historyId is sent as path variable
   b. **Output**: No content
   c. **Description**: This API allows nurse to delete an incorrect past history entry
20. **“nurse/viewPastHistoryById/{historyId}/{patientId}”**
   a. **Input:** HistoryId and PatientId are sent as path variable
   b. **Output:** Previously existed Past History Object

    c. **Description :** This API helps in displaying previously entered past history details for the patient while editing the previously existed vitals details.

21. **"/nurse/addSymptomImages/{patientId}"**
    a. **Input** : description,image as request body and patientId as path variable
    b. **Output**: Symptom Image added successfully
    c. **Description**:This API allows nurse to add symptom Image successfully for a specific patient
22. **"/nurse/editSymptomImages/{id}"**
    a. **Input** : description,image as request body and symptomId as path variable
    b. **Output**: Symptom Image edited successfully
    c. **Description**: This API allows nurse to edit the specific symptom image based on id provided
23. **"/nurse/viewSymptomImages/{patientId}/{consenttoken}"**
    a. **Input**: PatientId is sent as a path variable
    b. **Output**: A list of Symptom Image Objects of a specific patient
    c. **Description**: This API allows nurse to view all symptom image of a specific patient based on patientId provided if the consent token is valid
24. **"nurse/deleteSymptomImages/{id}"**
    a. **Input** : Symptom Image id is sent as path variable
    b. **Output**:No content
    c. **Description**: This API allows nurse to delete a specific symptom image based on symptom image id provided
25. **"/nurse/viewSymptomImagesById/{id}/{patientId}"**
    a. **Input:** Symptom Image id and patientId sent as path variable
    b. **Output :**Previously Existed Symptom Image object
    c. **Description :** This API helps in displaying previously entered symptom image details for the patient while editing the previously existed details.
26. **"/nurse/addPastImages/{historyId}"**
    a. **Input** : Past image is sent as request body and past history id as path variable
    b. **Output**: Past Image added successfully
    c. **Description**: This API allows nurse to add past image for a particular past history based on the past history id provided
27. **"/nurse/editPastImages/{imgId}"**
    a. **Input** : Updated Past image is sent as request body and past image id as path variable
    b. **Output**: Past Image edited successfully

    c. **Description**: This API allows nurse to edit the past image

28. **“/nurse/viewPastImages/{historyId}/{consenttoken}”**
    a. **Input** : Past History Id is sent as path variable
    b. **Output**: A list of past image object for a specific past history
    c. **Description**: This API allows nurse to view all past images of a specific past history if consent token is valid
29. **“/nurse/deletePastImages/{imgId}”**
    a. **Input** : Past Image id is sent as path variable
    b. **Output**: No content
    c. **Description**: This API allows nurse to delete a specific past image entry
30. **“/nurse/viewPastImagesById/{imgId}/{patientId}”**
    a. **Input:** Past Image Id and Patient Id are sent as path variable
    b. **Output :** Specific previously existed Past Image Object
    c. **Description** : This API helps in displaying previously entered past image details for the patient while editing the previously existed vitals details.
31. **“/nurse/viewTestName/{patientId}”**
    a. **Input**: Patient Id is sent as path variable
    b. **Output** : A list of test objects
    c. **Description**: This API allows nurse to retrieve all tests(whose results are not added) prescribed by doctor for a particular patient’s current visit
32. **“/nurse/addTestResult/{id}”**
    a. **Input** : result is sent as request body and test id as path variable
    b. **Output**: Test Result added successfully
    c. **Description**: This API allows nurse to add test result for a specific test based on test id provided
33. **“/nurse/editTestResult/{id}”**
    a. **Input** : Updated result is sent as request body and test id as path variable
    b. **Output**: Test Result edited successfully
    c. **Description**: This API allows nurse to edit test result for a specific test based on test id provided
34. **“/nurse/viewTest/{patientId}/{consenttoken}”**
    a. **Input** : PatientId is sent as path variable
    b. **Output**: A list of test Objects of a particular patient
    c. **Description**: This API allows nurse to view test results added by her for a particular patient’s current visit if consent token is valid
35. **“/nurse/viewTestById/{id}/{patientId}”**
    a. **Input:** Test Id and patientId are sent as path variable

   b. **Output: s**pecific Test Object
   c. **Description** : This API helps in displaying previously entered test result details for the patient while editing the previously existed details

36. **“/nurse/deleteTestResult/{id}”**
    a. **Input**: Test id is sent as path variable
    b. **Output**: No content
    c. **Description**: This API allows nurse to delete test result for a specific test

37. **“/nurse/addTestImages/{id}”**
    a. **Input**: test image is sent as request body and test id as path variable
    b. **Output**: Test Image added successfully
    c. **Description**: This API allows nurse to added test image for a particular test

38. **“/nurse/editTestImages/{testimageid}”**
    a. **Input**:Updated test image is sent as request body and test image id as path variable
    b. **Output**: Test Image edited successfully
    c. **Description**: This API allows nurse to edit test image for a particular test

39. **“/nurse/viewTestImages/{id}/{consenttoken}”**
    a. **Input**: Test id is sent as path variable
    b. **Output**: A list test image object for a particular test
    c. **Description**: This API allows nurse to view all test images for a particular test

40. **“/nurse/viewTestImagesById/{testimageid}/{patientId}”**
    a.**Input** : Test Image Id and Patient Id are sent as path variable
    b.**Output**: The specific Test Image previously existed
    c.**Description:** This API helps in displaying previously entered symptom image details for the patient while editing the previously existed vitals details.

41. **“/nurse/passwordChange”**
    a.**Input**: Email and password of the nurse
    b.**Output:**Password Changed successfully
    c.**Description**: This API allows to change the existing password for the nurse

42. .**“/nurse/deleteTestImages/{testimageid}”**
    a. **Input** : Test Image id is sent as path variable
    b. **Output**: No content
    c. **Description**: This API allows nurse to delete an incorrect test image entry

43. **“/nurse/logout/{email}”**
    a. Input : Email is sent as path variable

    b. **Output :** String stating logout is successful or failure
    c. **Description :** This API allows nurse to logout successfully and deletes respective jwt token from backend

44. **“/nurse/getConsentToken/{pid}”**
    a. **Input**: patient id as path variable.
    b. **Output**: Consent token.
    c. **Description**: This API returns the consent token for the patient id if the consent token.
45. **“/nurse/sendOtpforpassword/{contact}”**
    a. **Input**: contact as path variable.
    b. **Output**: status of sending contact.
    c. **Description**: This API returns a string status of sending OTP.
46. **“/nurse/verifyOtpforpassword/{contact}/{otp}”**
    a. **Input**: contact and otp as path variable.
    b. **Output**: status of verification.
    c. **Description**: This API returns a string stating the status of otp verification.
47. **“/nurse/getContactfromEmail/{email}”**
    a. **Input**: email as path variable.
    b. **Output**: contact number.
    c. **Description**: This API returns the contact number for the specified email if it exists.
48. **“/nurse/passwordChange”**
    a. **Input**: Email and password of the nurse
    b. **Output:**Password Changed successfully
    c. **Description**: This API allows us to change the existing password for the nurse.

Pharmacy Module:

1. **“/pharmacy/login”**
    a. **Input**: Email and password as request body.
    b. **Output**: String indicating successful login or failure.
    c. **Description**: This API is used for pharmacy authentication. It takes the pharmacy’s login credentials and verifies them.

2. **“/pharmacy/logout/{email}**
    a. Input: email as path variable.
    b. Output: String stating logout is successful or failure.
    c. Description: This API allows pharmacy to logout successfully and deletes respective jwt token from backend
3. **“pharmacy/home/{email}”**
    a. **Input:** email as path variable.
    b. Output: pharmacy object
    c. Description: This API lets us fetch the pharmacy details of the pharmacy as email as prop.
4. **“ /pharmacy/viewpharmacy/{pharmacyId}”**
    a. **Input**: Pharmacy ID as a path variable.
    b. **Output**: displays pharmacy details
    c. **Description**: This API retrieves detailed information about a specific pharmacy based on the provided pharmacy ID.
5. **“/pharmacy/view medication/{patientId}/{consenttoken}”**
    a. **Input**: PatientID and consent token as a path variable.
    b. **Output**:List of medications.
    c. **Description**: This API retrieves the recent medication table of a specific patient based on the provided patient ID.
6. **“/pharmacy/serve/{medicationId}”**
    a. **Input**: MedicationId as a path variable.
    b. **Output**: Updates serve attribute.
    c. **Description**: This API updates the serve attribute based on the provided medicationID
7. **“/pharmacy/passwordChange”**
    a. **Input**: Email and password of the pharmacy
    b. **Output:**Password Changed successfully
    c. **Description**: This API allows to change the existing password for the pharmacy
8. **“/pharmacy/viewCanvas/{patientId}/{consenttoken}”**
    a. **Input**: patient id and consent token as path variable.
    b. **Output**: canvas object
    c. **Description**: This API fetches the canvas for the patient id if the consent token is not expired.
9. **“/pharmacy/serveCanvas/{canvasid}”**
    a. **Input**: canvas id as path variable.

b. **Output**: updates serve attribute.
c. **Description**: This API updates the serve attribute based on the provided canvas id.

10. **“/pharmacy/total-served”**
    a. **Input**:
    b. **Output**:Count of patients served and unique medicines served.
    c. **Description**:This API Count of patients served and unique medicines served.
11. **“/pharmacy/getConsentToken/{pid}”**
    a. **Input**: patient id as path variable.
    b. **Output**: Consent token.
    c. **Description**: This API returns the consent token for the patient id if the consent token.
12. **“/pharmacy/sendOtpforpassword/{contact}”**
    a. **Input**: contact as path variable.
    b. **Output**: status of sending contact.
    c. **Description**: This API returns a string status of sending OTP.
13. **“/pharmacy/verifyOtpforpassword/{contact}/{otp}”**
    a. **Input**: contact and otp as path variable.
    b. **Output**: status of verification.
    c. **Description**: This API returns a string stating the status of otp verification.
14. **“/pharmacy/getContactfromEmail/{email}”**
    a. **Input**: email as path variable.
    b. **Output**: contact number.
    c. **Description**: This API returns the contact number for the specified email if it exists.
15. **“/pharmacy/passwordChange”**
    d. **Input**: Email and password of the pharmacy
    e. **Output:**Password Changed successfully
    f. **Description**: This API allows us to change the existing password for the pharmacy.

Receptionist Module:

1. **“/receptionist/login”**
    a. **Input** : **receptionist**’s email id and password
    b. **Output**: String stating login failure or successful

   c. **Description**: This API checks if email and password entered by receptionist matches with the record in backend or not and checks whether receptionist is active and is trying to login in scheduled time or not,if all requirements met then receptionist is allowed to login.
2. **“/receptionist/logout/{email}”**
   d. **Input :** Email is sent as path variable
   e. **Output :** String stating logout is successful or failure
   f. **Description :** This API allows nurse to logout successfully and deletes respective jwt token from backend
3. **“/receptionist/bookAppointmentForExistingPatient/{email}/{pid}”**
   a. **Input:** Patient ID and Receptionist email as path variables. Visit details in the request body in the format of appointment DTO .
   b. **Output:** Created Visit object.
   c. **Description:** This API allows a receptionist to book an appointment for an existing patient. Email indicates that consent was taken by an authorized receptionist.
4. **“/receptionist/bookAppointmentForNewPatient/{email}”**
   a. **Input:** Receptionist email as a path variable. Visit details in the request body.
   b. **Output:** Created Visit object.
   c. **Description:** This API allows a receptionist to book an appointment for a new patient with a specific doctor. Email indicates that consent was taken by an authorized receptionist.
5. **“/receptionist/bookEmergencyAppointment/{did}”**
   a. **Input:** Doctor ID as a path variable. Emergency visit details in the request body.
   b. **Output:** Created Visit object.
   c. **Description:** This API allows a receptionist to book an emergency appointment for a patient with a specific doctor.
6. **“/receptionist/getPatientDetails/{pid}/{consenttoken}”**
   a. **Input:** Patient ID and consent token as a path variable.
   b. **Output:** Patient object.
   c. **Description:** This API retrieves details of a specific patient based on the provided Patient ID If the consent token is valid.
7. **“/receptionist/addPatient”**
   a. **Input:** Patient details in the request body.
   b. **Output:** Create a Patient object.

    c. **Description:** This API allows a receptionist to add a new patient to the system.

8. **“/receptionist/updatePatient/{pid}”**
    a. **Input:** Patient ID as a path variable. Updated patient details in the request body.
    b. **Output:** Updated Patient object.
    c. **Description:** This API allows a receptionist to update details of a specific patient.
9. **“/receptionist/deletePatientPII/{pid}”**
    a. **Input:** Patient ID as a path variable.
    b. **Output:** Updated Patient object with sensitive information removed.
    c. **Description:** This API allows a receptionist to delete personally identifiable information (PII) of a specific patient.
10. **“/receptionist/deletePatientRecords/{pid}”**
    a. **Input:** Patient ID as a path variable.
    b. **Output:** No content.
    c. **Description:** This API is used to delete all medical records associated with a specific patient.
11. **“/receptionist/getAllPatients”**
    a. **Output:** Map containing patient count and a list of patients.
    b. **Description:** This API retrieves the list of all patients in the system.
12. **“/receptionist/getIndoorPatients”**
    a. **Output:** Map containing patient count and a list of indoor patients.
    b. **Description:** This API retrieves the list of indoor patients in the system.
13. **“/receptionist/getOutdoorPatients”**
    a. **Output:** Map containing patient count and a list of outdoor patients.
    b. **Description:** This API retrieves the list of outdoor patients in the system.
14. **“/receptionist/getAllDoctors”**
    a. **Output:** Map containing doctor count and a list of doctors.
    b. **Description:** This API retrieves the list of all doctors in the system.
15. **“/receptionist/getIndoorDoctors”**
    a. **Output:** Map containing doctor count and a list of indoor doctors.
    b. **Description:** This API retrieves the list of indoor doctors in the system.
16. **“/receptionist/getOutdoorDoctors”**
    a. **Output:** Map containing doctor count and a list of outdoor doctors.
    b. **Description:** This API retrieves the list of outdoor doctors in the system.
17. **“/receptionist/getOutdoorDoctorsBySpecialization/{specialization}”**

a. **Input:** Doctor specialization as a path variable.
b. **Output:** Map containing doctor count and a list of outdoor doctors with the specified specialization.
c. **Description:** This API retrieves the list of outdoor doctors based on the provided specialization.

18. **“/receptionist/getAllSpecializations”**
    a. **Input:**
    b. **Description:** This API retrieves the list of specializations in the system.
    c. **Output:** List of all specializations.
19. **“/receptionist/viewReceptionistScheduleById/{receptionistId}”**
    a. **Input:** Receptionist ID as a path variable.
    b. **Output:** List of receptionist schedule.
    c. **Description:** This API retrieves the list of schedules of a particular receptionist.
20. **“/receptionist/getReceptionistDetailsByEmail/{email}”**
    a. **Input:** Receptionist email as a path variable.
    b. **Output:** Receptionist object.
    c. **Description:** This API retrieves all the details of a particular receptionist.
21. **“/receptionist/sendOtp/{contact}”**
    a. **Input:** Contact number as a path variable.
    b. **Output:** Success / Failure message of sending otp.
    c. **Description:** This API sends otp to the provided contact number.
22. **“/receptionist/verifyOtp/{contact}/{otp}”**
    a. **Input:** Contact number as a path variable along with the typed in otp by user.
    b. **Output:** Success / Failure message of verification status of otp.
    c. **Description:** This API verifies the otp sent to a contact number.
23. **“/receptionist/getConsentToken/{pid}”**
    a. **Input:** Patient id as path variable.
    b. **Output:** Consent token of patient.
    c. **Description:** This API retrieves consent token of specified patient.

24. **“/receptionist/getAllAppointments/{pid}/{consenttoken}”**
    a. **Input:** Patient id as path variable along with consent token.
    b. **Output:** List of all doctors that that patient has appointments with.
    c. **Description:** This API retrieves list of all doctor objects that the patient has an appointment scheduled with.

25. **“/receptionist/deleteAppointments/{pid}/{did}”**
    a. **Input:** Patient id as path variable along with consent token.
    b. **Output:** Success message on appointment deletion**.**
    c. **Description:** This API deletes appointments of a patient with the specified doctor.
26. **“/receptionist/passwordChange”**
    a. **Input**: Email and password of the receptionist
    b. **Output:**Password Changed successfully
    c. **Description**: This API allows to change the existing password for the receptionist.

Doctor Module:

1. **"/doctor/login"**
    a. **Input:** Email and password.
    b. **Output:** String indicating successful login or failure.
    c. **Description:** This API is used for doctor authentication. It takes the doctor's login credentials and verifies them.
2. **"/doctor/home/{email}"**
    a. **Input:** Doctor's email as a path variable.
    b. **Output:** Doctor object.
    c. **Description:** This API retrieves the details of a doctor based on the provided email.
3. **"/doctor/viewPatients/{email}"**
    a. **Input:** Doctor's email as a path variable.
    b. **Output:** List of Patient objects.
    c. **Description:** This API retrieves the list of normal patients assigned to a specific doctor.
4. **"/doctor/viewEmergencyPatients/{email}”**
    a. **Input:** Doctor's email as a path variable.
    b. **Output:** List of emergency Patient objects.
    c. **Description:** This API retrieves the list of emergency patients assigned to a specific doctor.
5. **"/doctor/patientDetails/{pid}/{consenttoken}"**
    a. **Input:** Patient ID and consenttoken as a path variable.
    b. **Output:** Patient object.

   c. **Description:** This API retrieves detailed information about a specific patient based on the provided patient ID.
6. **"/doctor/patientVitals/{pid}/{consenttoken}"**
   a. **Input:** Patient ID and consent token as a path variable.
   b. **Output:** Vitals object.
   c. **Description:** This API retrieves the vitals of a specific patient.
7. **"/doctor/patientSymptoms/{pid}/{consenttoken}"**
   a. **Input:** Patient ID and consent token as a path variable.
   b. **Output:** Symptoms object.
   c. **Description:** This API retrieves the symptoms reported by a specific patient.
8. **"/doctor/symptomImages/{pid}/{consenttoken}"**
   a. **Input:** Patient ID and consent token as a path variable.
   b. **Output:** List of SymptomImages objects.
   c. **Description:** This API retrieves images related to symptoms reported by a specific patient.
9. **"/doctor/pastHistory/{pid}/{consenttoken}"**
   a. **Input:** Patient ID and consent token as a path variable.
   b. **Output:** List of PastHistory objects.
   c. **Description:** This API retrieves the past medical history of a specific patient.
10. **"/doctor/pastImages/{phid}/{pid}/{consenttoken}"**
   a. **Input:** Past History ID,patient ID and consent token as a path variable.
   b. **Output:** List of PastImages objects.
   c. **Description:** This API retrieves images related to the past medical history.
11. **"/doctor/pastMedications/{pid}/{consenttoken}"**
   a. **Input:** Patient ID and consent token as a path variable.
   b. **Output:** List of Medication objects.
   c. **Description**: This API retrieves past medications prescribed to a specific patient.
12. **"/doctor/pastTests/{pid}/{consenttoken}"**
   a. **Input:** Patient ID and consent token as a path variable.
   b. **Output:** List of Test objects.
   c. **Description:** This API retrieves past tests conducted on a specific patient.
13. **"/doctor/recordProgress/{pid}"**
   a. **Input:** Patient ID as a path variable, Progress status in the request body.
   b. **Output:** Progress object.

   c. **Description:** This API is used to record the progress of a patient. It takes the patient ID and progress details and returns the recorded progress.

14. **"/doctor/progressHistory/{pid}/{consenttoken}"**
   a. **Input:** Patient ID and consent token as a path variable.
   b. **Output:** List of Progress objects.
   c. **Description:** This API retrieves the progress history of an admitted patient.
15. **"/doctor/viewMedications/{pid}/{consenttoken}"**
   a. **Input:** Patient ID and consent token as a path variable.
   b. **Output:** List of Medication objects.
   c. **Description:** This API retrieves the current medications prescribed to a specific patient.
16. **"/doctor/getMedication/{pid}/{mid}"**
   a. **Input:** Patient ID and Medication ID as path variables.
   b. **Output:** Medication object.
   c. **Description:** This API allows a doctor to get medication for a specific patient.
17. **"/doctor/editMedication/{pid}/{mid}"**
   a. **Input:** Patient ID as a path variable, Edited Medicine patient.
   b. **Output**: Updated medication.
   c. **Description**: This API allows to update the medication and return updated medication.
18. **"/doctor/addMedication/{pid}/{email}"**
   a. **Input**:Patient ID and email as a path variable, Medicine name,dosage,frequency,duration and special Instructions in the request body.name,dosage,frequency,duration and special Instructions in the request body.
   b. **Output:** Updated Medication object.
   c. **Description:**This API allows a doctor to edit details of a previously recommended medication for a patient.
19. **"/doctor/deleteMedication/{pid}/{mid}"**
   a. **Input:** Patient ID and Medication ID as path variables.
   b. **Output:** No content.
   c. **Description:** This API is used to delete a specific medication prescribed to a patient..
20. **"/doctor/viewTests/{pid}/{consenttoken}/{email}"**
   a. **Input:** Patient ID, consent token and doctor email as a path variable.
   b. **Output:** List of Test objects.

   c. **Description:** This API retrieves the current tests scheduled for a specific patient.

21. **"/doctor/getTest/{pid}/{tid}"**
    a. **Input:** Patient ID and Test ID as path variables.
    b. **Output:** Test object.
    c. **Description:** This API retrieves detailed information about a specific medical test recommended for a particular patient.
22. **"/doctor/addTest/{pid}/{email}"**
    a. **Input:** Patient ID and doctor email as a path variable, Test Name in the request body.
    b. **Output:** Added Test object.
    c. **Description:** This API allows a doctor to recommend a new medical test for a patient.
23. **"/doctor/editTest/{pid}/{tid}"**
    a. **Input:** Patient ID and Test ID as path variables, Updated Test Name in the request body.
    b. **Output:** Updated Test object.
    c. **Description:** This API allows a doctor to edit details of a previously recommended medical test for a patient.
24. **"/doctor/deleteTest/{pid}/{tid}"**
    a. **Input:** Patient ID and Test ID as path variables.
    b. **Output:** No content.
    c. **Description:** This API allows a doctor to remove a recommended medical test from a patient's treatment plan.
25. **"/doctor/testImage/{tid}/{pid}/{consenttoken}"**
    a. **Input:** Test ID , patient ID and consent token as a path variable.
    b. **Output:** List of TestImages objects.
    c. **Description:** This API retrieves images associated with a specific medical test.
26. **"/doctor/setDisease/{pid}/{disease}/{email}"**
    a. **Input:** Patient ID , disease name and doctor’s email as path variables.
    b. **Output:** Updated Visit object.
    c. **Description:** This API allows a doctor to record a disease for a patient during a visit.
27. **"/doctor/changetoIP/{pid}/{did}/{email}"**
    a. **Input:** Patient ID , Doctor ID and doctor’s email as path variables.
    b. **Output:** Updated Patient object.

c. **Description:** This API allows a doctor to change a patient to "IP" (Inpatient) and assigns a specific doctor,bed to the patient.

28. **"/doctor/discharge/{pid}/{email}"**
    a. **Input:** Patient ID and email as a path variable.
    b. **Output:** Updated Patient object.
    c. **Description:** This API allows a doctor to discharge a patient. It marks the end of the treatment for the patient, updates the patient's status, and may free up a bed in case of IP.
29. **"/doctor/admittedCount/{email}"**
    a. **Input:** Doctor's email as a path variable.
    b. **Output**: Count of admitted patients for the doctor.
    c. **Description**: This API retrieves the count of patients admitted under a specific doctor.
30. **"/doctor/treatedCount/{email}"**
    a. **Input**: Doctor's email as a path variable.
    b. **Output**: Count of treated patients for the doctor.
    c. **Description**: This API retrieves the count of patients treated by a specific doctor based on the provided email.
31. **"/doctor/getSpecializationDoctors/{specialization}"**
    a. **Input**: Doctor's specialization as a path variable.
    b. **Output**: List of Doctor objects.
    c. **Description**: This API retrieves a list of available doctors with a specific specialization.
32. **“/doctor/exitDutyDoctor/{email}”**
    a. **Input**: doctor email as path variable.
    b. **Output**:
    c. **Description**: This API sets the availability as false.
33. **“/doctor/viewDetails/{email}/{pid}/{consenttoken}”**
    a. **Input**: doctor email, patient id, consent token as path variable.
    b. **Output**: patient entity.
    c. **Description**. This API returns the patient having the given patient id and provided consent token which is not expired.
34. **“/doctor/viewAdmitted/{bid}”**
    a. **Input**: bed id as path variable.
    b. **Output**: Patient ID .
    c. **Description**: This API returns the Patient ID of the admitted patient to that bed.

35. **“/doctor/getConsentToken/{pid}”**
    a. **Input**: patient id as path variable.
    b. **Output**: Consent token.
    c. **Description**: This API returns the consent token for the patient id if the consent token.
36. **“/doctor/checkPatient/{pid}/{email}”**
    a. **Input**: patient id and doctor email as path variable.
    b. **Output**:
    c. **Description**:This API gets the patient check status.
37. **“/doctor/addCanvas/{pid}/{email}”**
    a. **Input**: patient id and doctor email as path variable.
    b. **Output**: Initializes a canvas for that visit.
    c. **Description**: This API adds and returns canvas for that patient.
38. **“/doctor/editCanvas/{pid}/{email}”**
    a. **Input**: patient id and doctor email as path variable.
    b. **Output**: Updated Canvas.
    c. **Description**: This API helps to edit the canvas.
39. **“/doctor/deleteCanvas/{pid}/{email}”**
    a. **Input**: patient id and doctor email as path variable.
    b. **Output**:
    c. **Description**: This API is used to delete the canvas.
40. **“/doctor/viewCanvas/{pid}/{consenttoken}/{email}”**
    a. **Input**: pid, consent token and email as path variable
    b. **Output**: canvas
    c. **Description**: This API fetches the canvas of the patient’s recent visit (not discharged) which is assigned to the doctor
41. **“/doctor/pastCanvas/{pid}/{consenttoken}”**
    a. **Input**: patient id and consent token as path variable.
    b. **Output**: List of canvas.
    c. **Description**: This API returns a list of canvases for patient having pid when consent token is not expired.
42. **“/doctor/fetchNotification/{email}”**
    a. **Input**: doctor email as path variable.
    b. **Output**: List of messages.
    c. **Description**: This API returns a list of messages or notifications available for the doctor having email as his/her email.
43. **“/doctor/sendOtpforpassword/{contact}”**

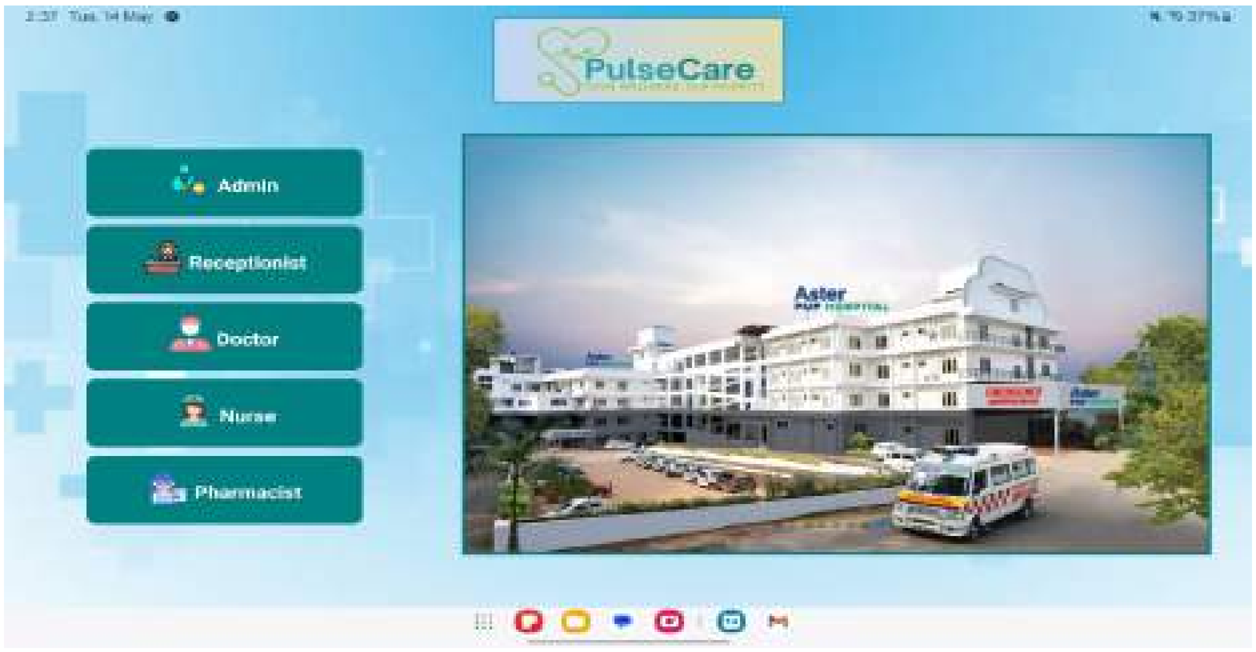
a. **Input**: contact as path variable.
b. **Output**: status of sending contact.
c. **Description**: This API returns a string status of sending OTP.

44. **“/doctor/verifyOtpforpassword/{contact}/{otp}”**
    a. **Input**: contact and otp as path variable.
    b. **Output**: status of verification.
    c. **Description**: This API returns a string stating the status of otp verification.
45. **“/doctor/getContactfromEmail/{email}”**
    a. **Input**: email as path variable.
    b. **Output**: contact number.
    c. **Description**: This API returns the contact number for the specified email if it exists.
46. **“/doctor/passwordChange”**
    g. **Input**: Email and password of the doctor
    h. **Output:**Password Changed successfully
    i. **Description**: This API allows to change the existing password for the doctor

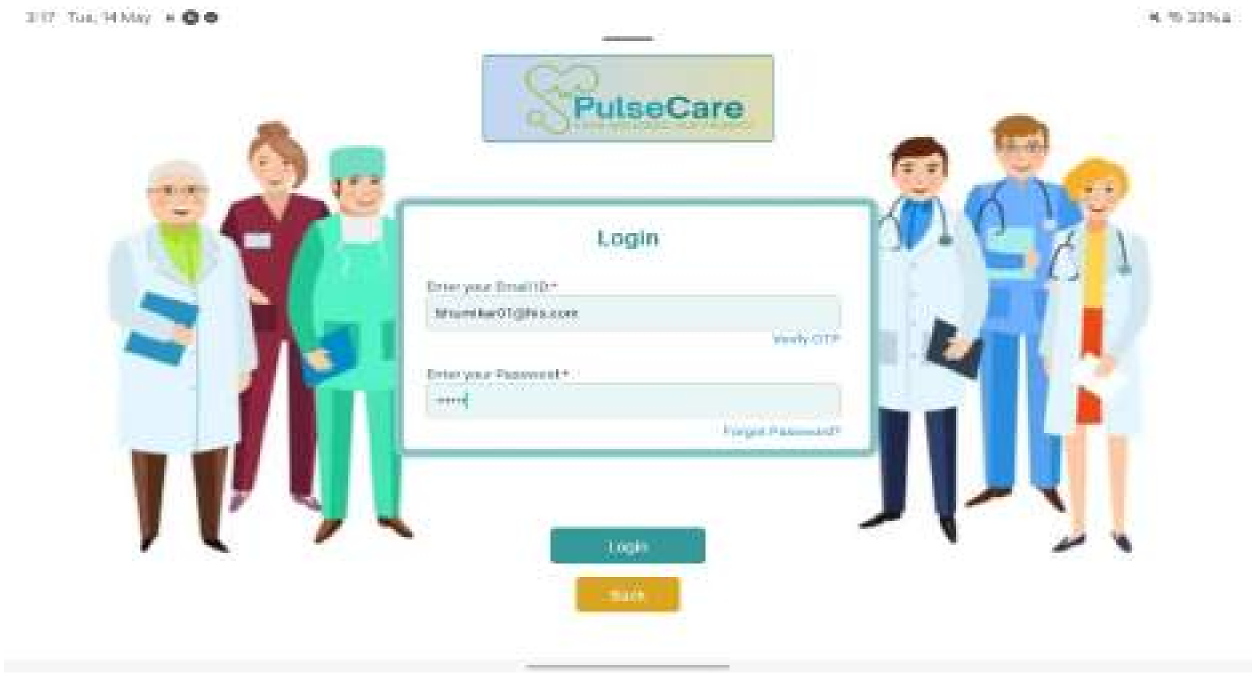
47. **“/doctor/logout/{email}”**
    a. **Input:** email as path variable.
    b. **Output:** String stating logout is successful or failure
    c. **Description:** This API allows doctor to logout successfully and deletes respective jwt token from backend.

DEMONSTRATION OF ACHIEVED FUNCTIONALITIES

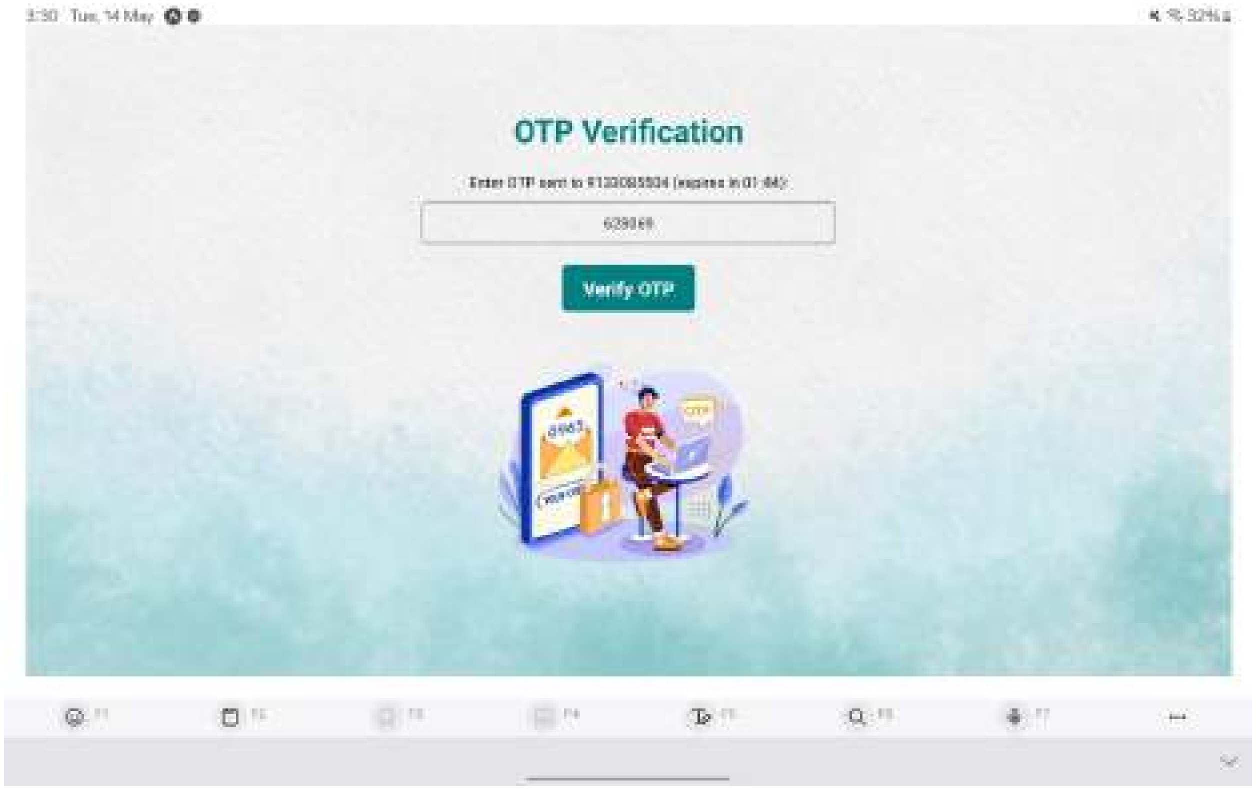
1. Home Page (common for all modules)



2. Login Page (common to all modules)

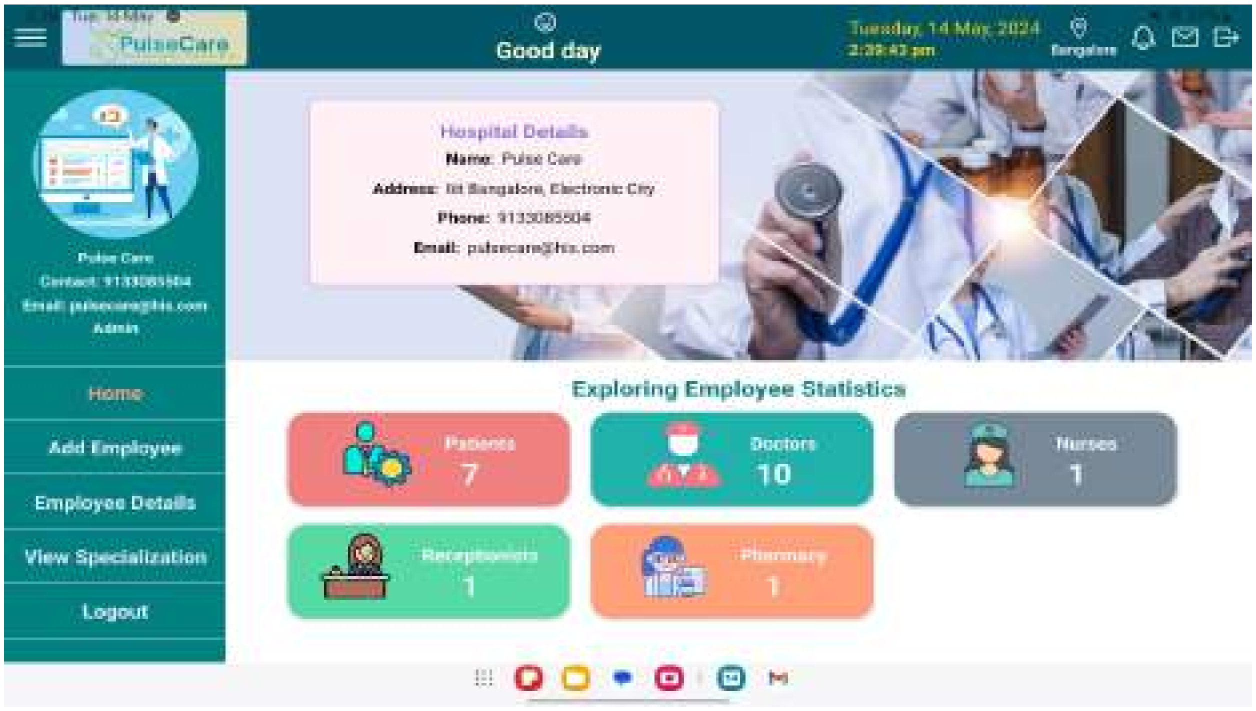


3.Login OTP Verification



Admin Module

1. Admin Dashboard



Hospital Details

Name: Pulse Care

Address: Bangalore, Electronic City

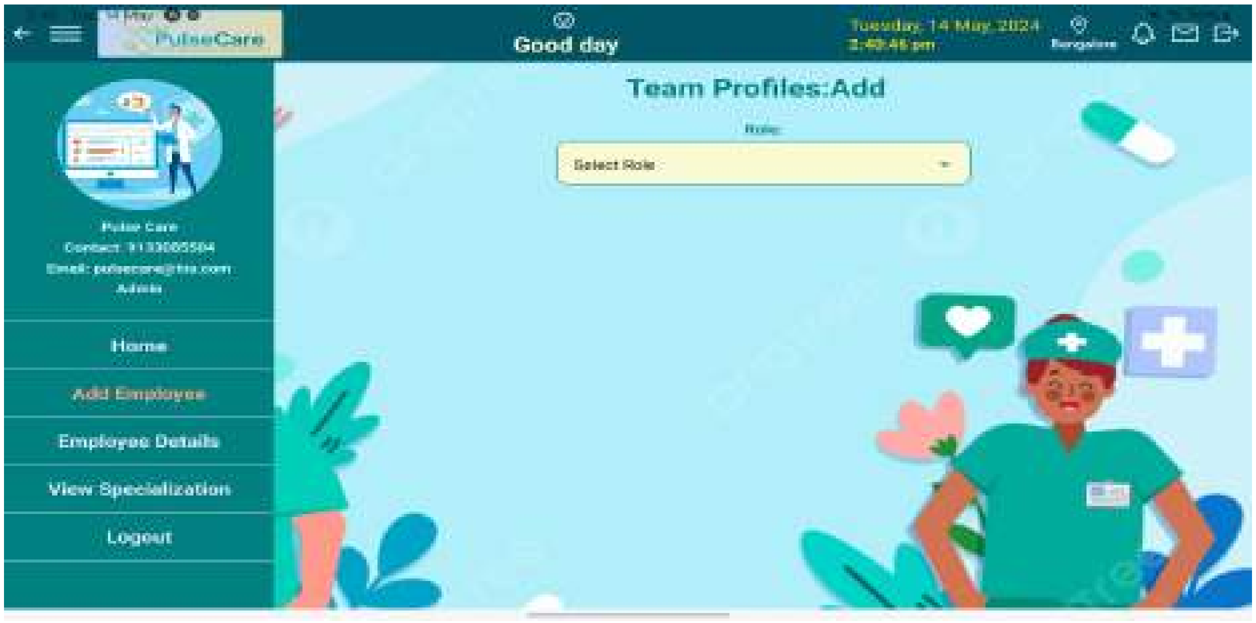
Phone: 9133085504

Email: pulsecare@his.com

Exploring Employee Statistics

Patients	7
Doctors	10
Nurses	1
Receptionists	1
Pharmacy	1

2. Add Employee

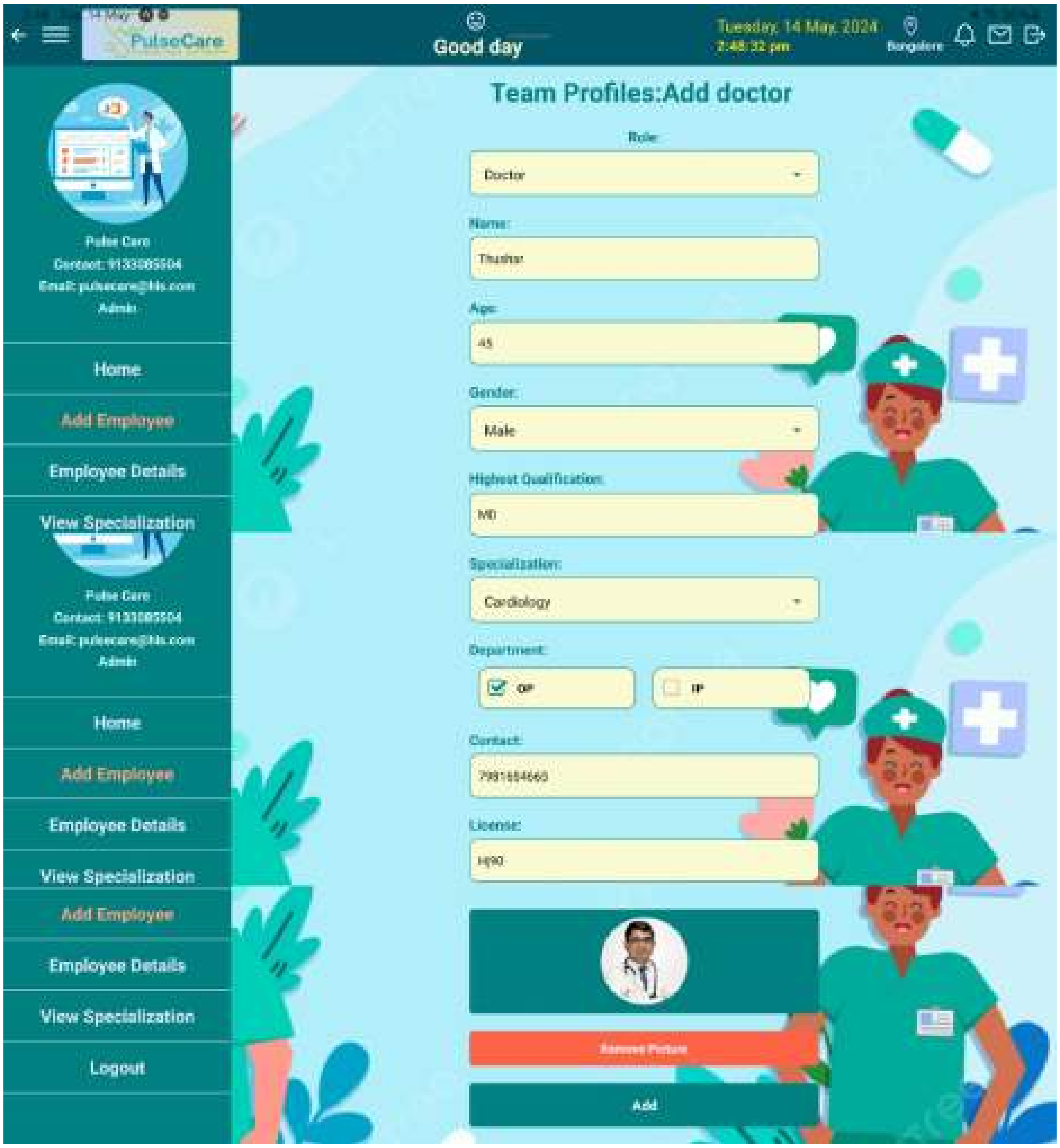


Team Profiles:Add

Role:

Select Role

a. Doctor



Team Profiles:Add doctor

Role: Doctor

Name: Thushar

Age: 45

Gender: Male

Highest Qualification: MD

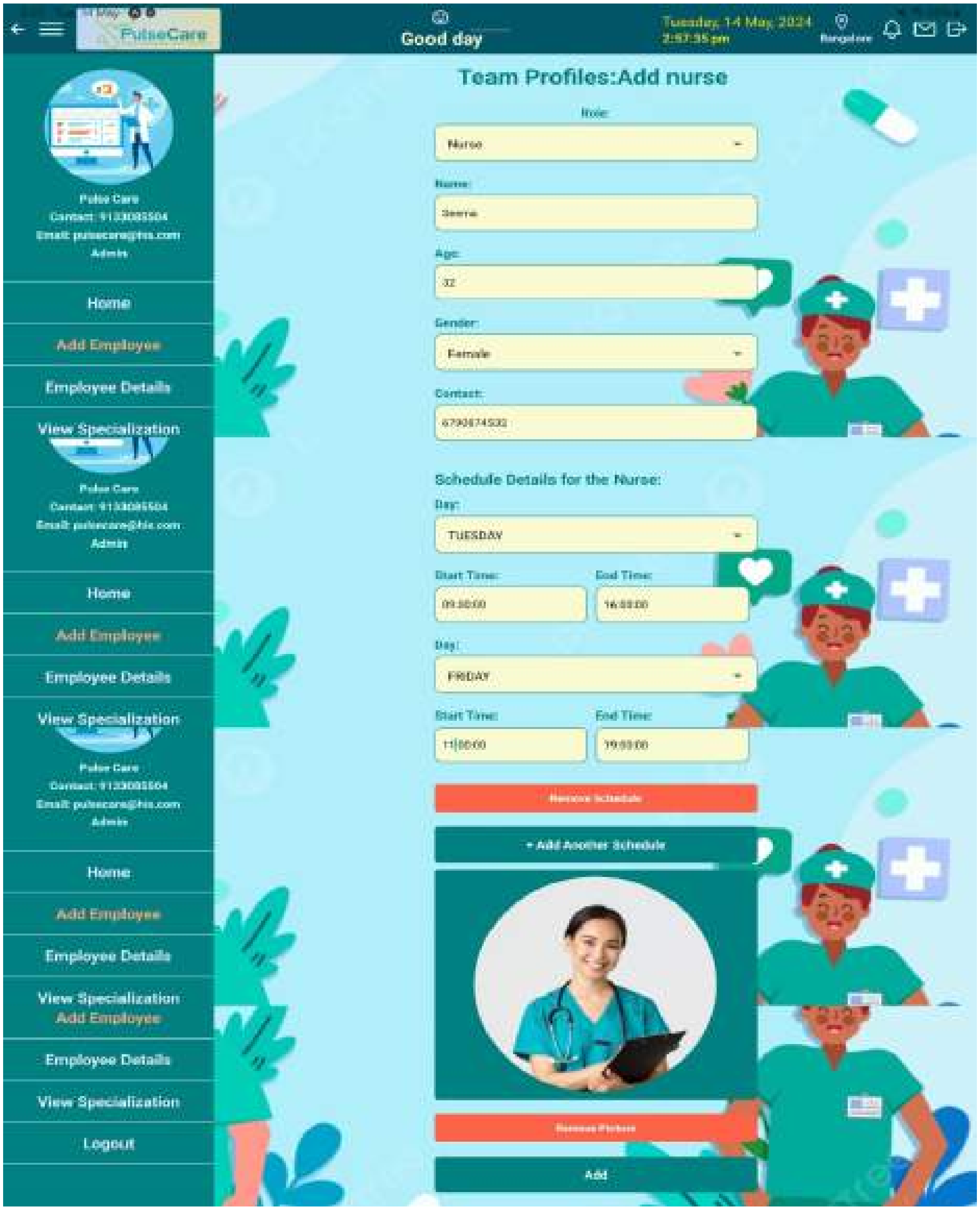
Specialization: Cardiology

Department: [✓ OP] IP

Contact: 7981654660

License: HJ90

b. Nurse



Team Profiles:Add nurse

Role: Nurse

Name: Seema

Age: 32

Gender: Female

Contact: 6790874532

Schedule Details for the Nurse:

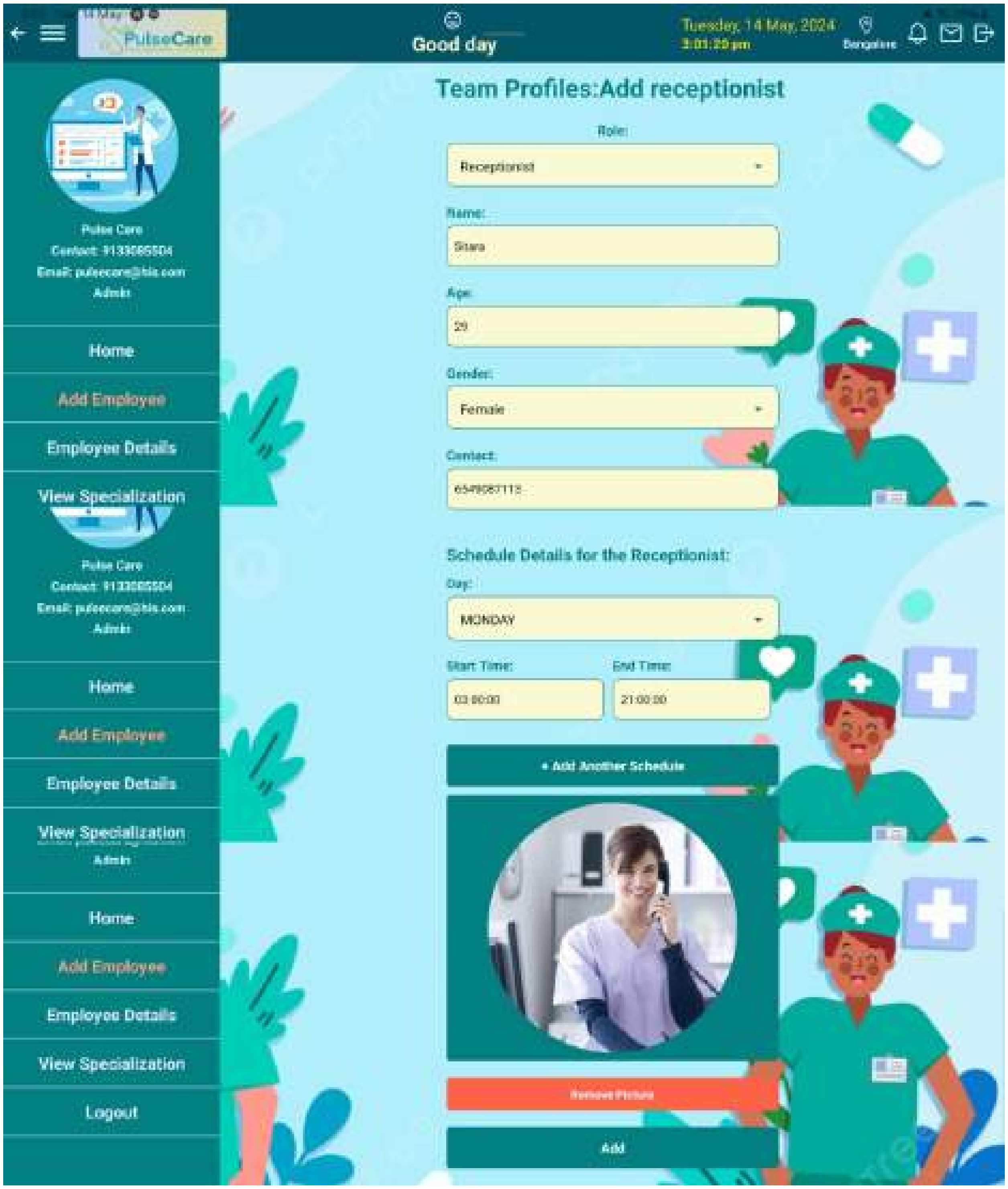
Day: TUESDAY

Start Time: 09:30:00 End Time: 16:30:00

Day: FRIDAY

Start Time: 11:30:00 End Time: 19:30:00

c. Receptionist



Team Profiles:Add receptionist

Role:
Receptionist

Name:
Sitara

Age:
29

Gender:
Female

Contact:
6549087112

Schedule Details for the Receptionist:

Day:
MONDAY

Start Time: 03:00:00

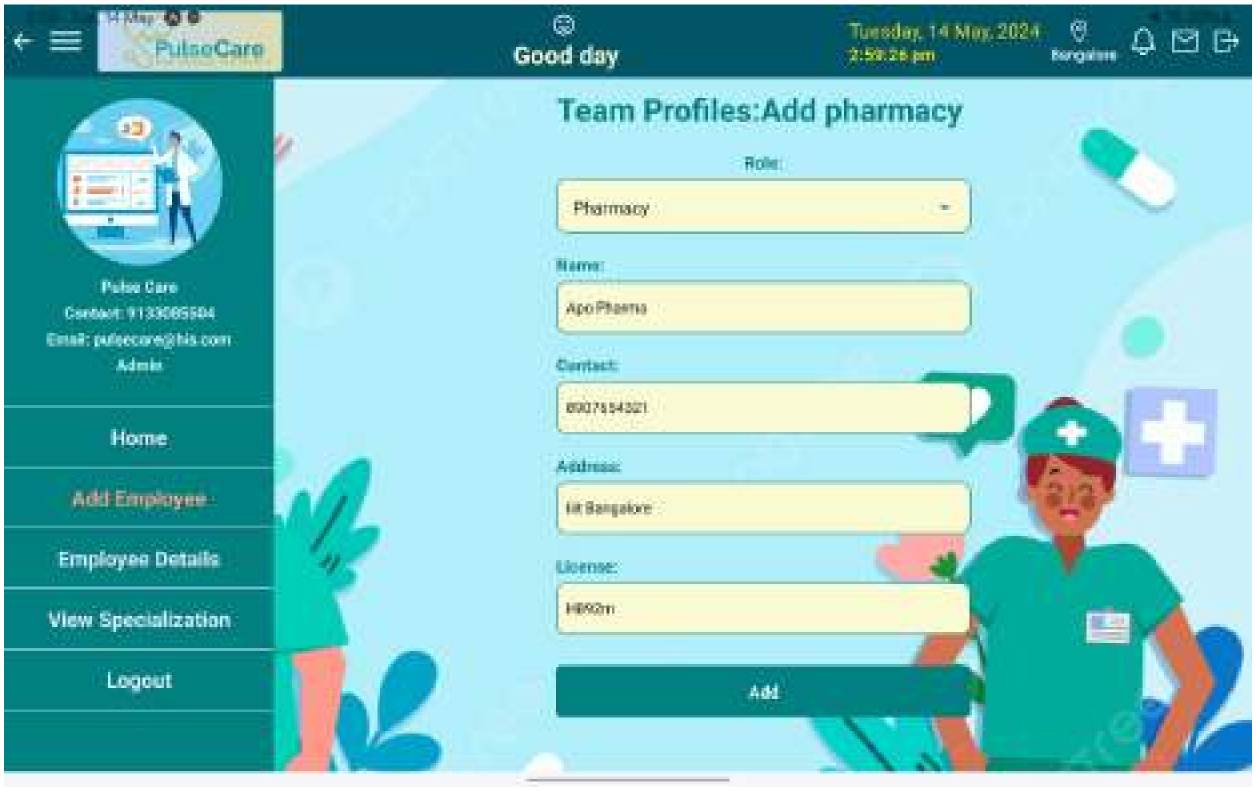
End Time: 21:00:00

+ Add Another Schedule

Remove Picture

Add

d. Pharmacy



Team Profiles:Add pharmacy

Role: Pharmacy

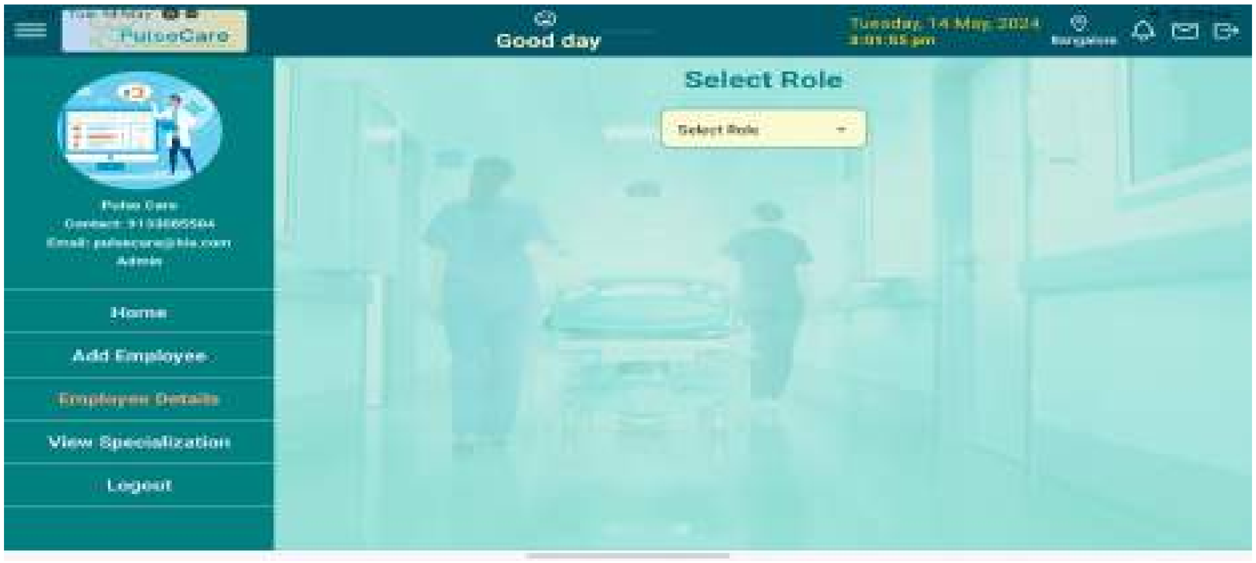
Name: Apo Pharma

Contact: 8907654321

Address: IIit Bangalore

License: H892m

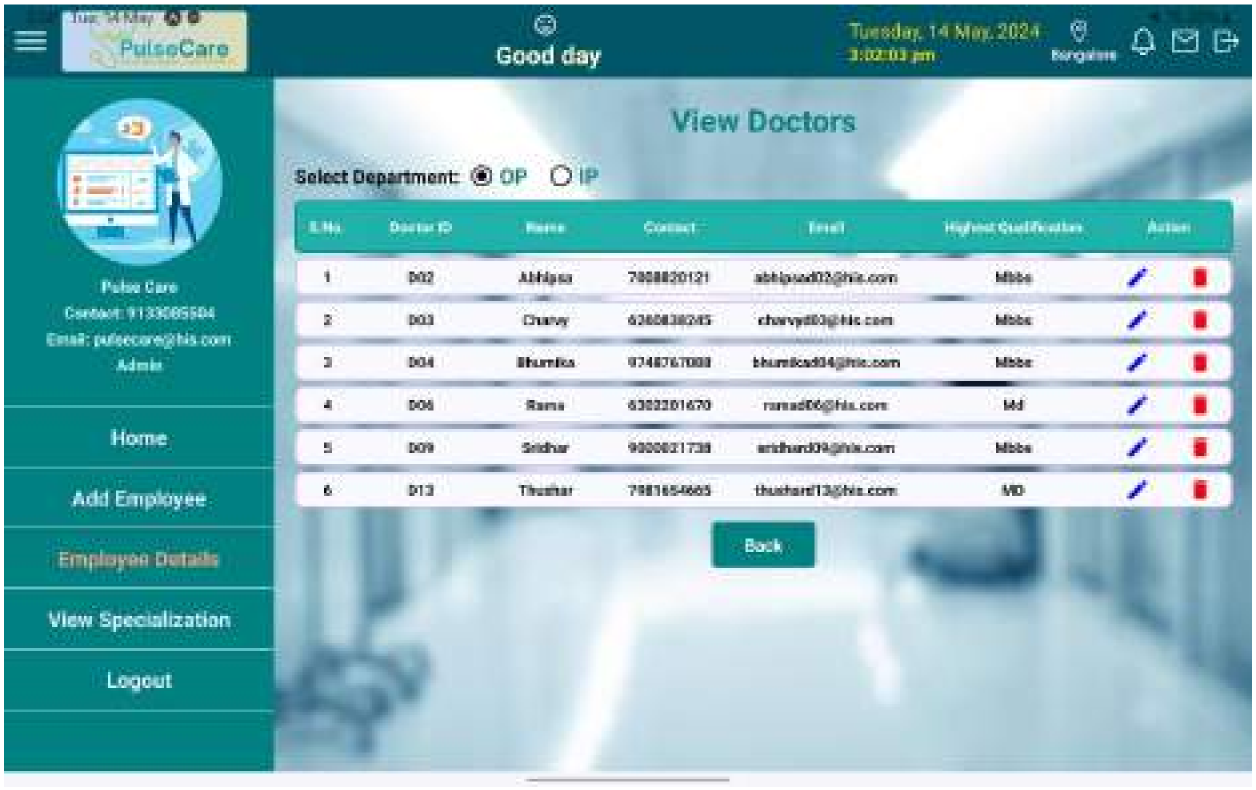
3. View all employees



Select Role

Select Role

a. Doctor



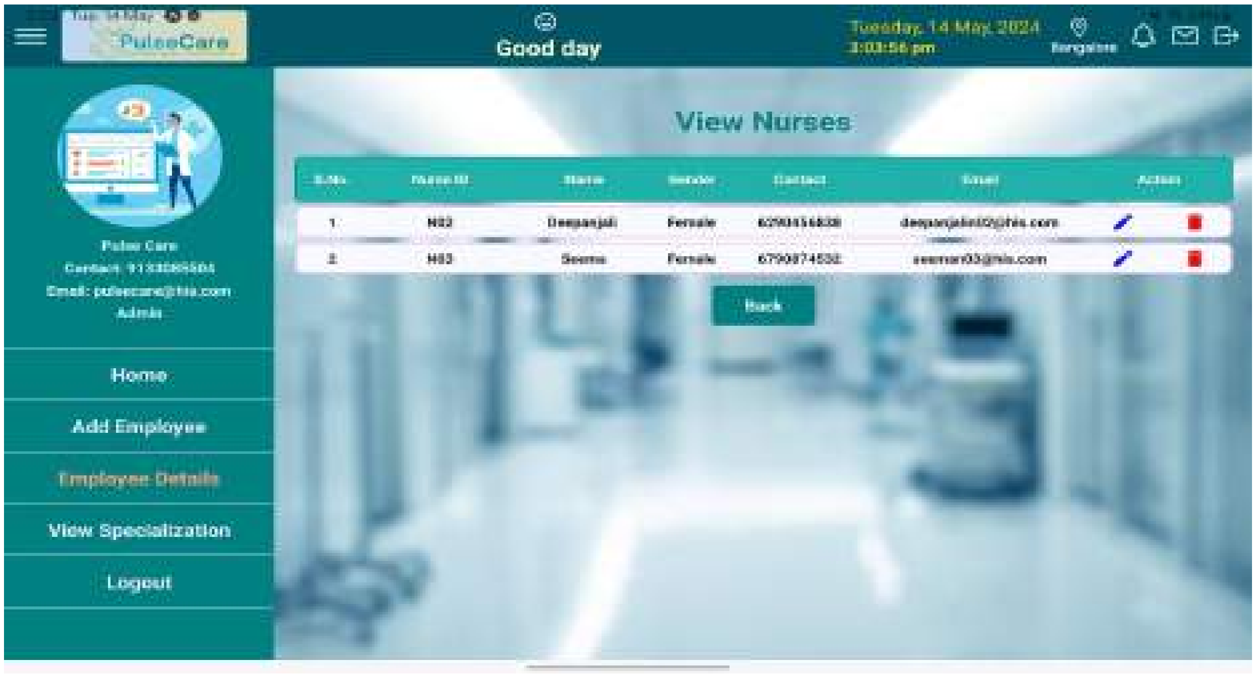
Pulse Care
Contact: 9133085504
Email: pulsecare@his.com
Admin

View Doctors

Select Department: [✓ OP] IP

S.No	Doctor ID	Name	Contact	Email	Highest Qualification	Action
1	D02	Abhipsa	7008820121	abhipsad02@his.com	Mbbs	
2	D03	Charvy	6360838245	charvyd03@his.com	Mbbs	
3	D04	Bhumika	9748767008	bhumikad04@his.com	Mbbs	
4	D06	Rama	6302201670	ramad06@his.com	Md	
5	D09	Sridhar	9000021738	sridhard09@his.com	Mbbs	
6	D13	Thushar	7981654665	thushard13@his.com	MD	

b. Nurse

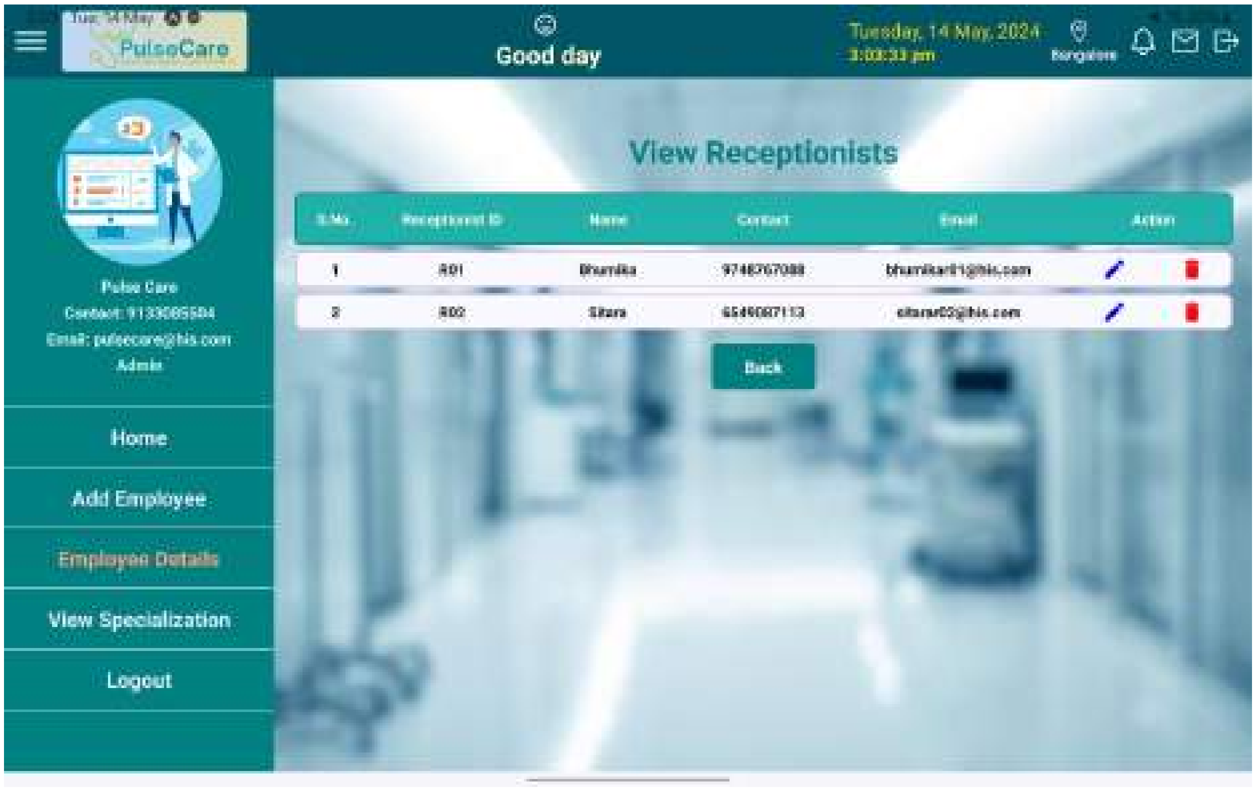


Pulse Care
Contact: 9133085504
Email: pulsecare@his.com
Admin

View Nurses

S.No	Nurse ID	Name	Gender	Contact	Email	Action
1	N02	Deepanjali	Female	6790354838	deepanjalin02@his.com	
2	N03	Seema	Female	6790874532	seeman03@his.com	

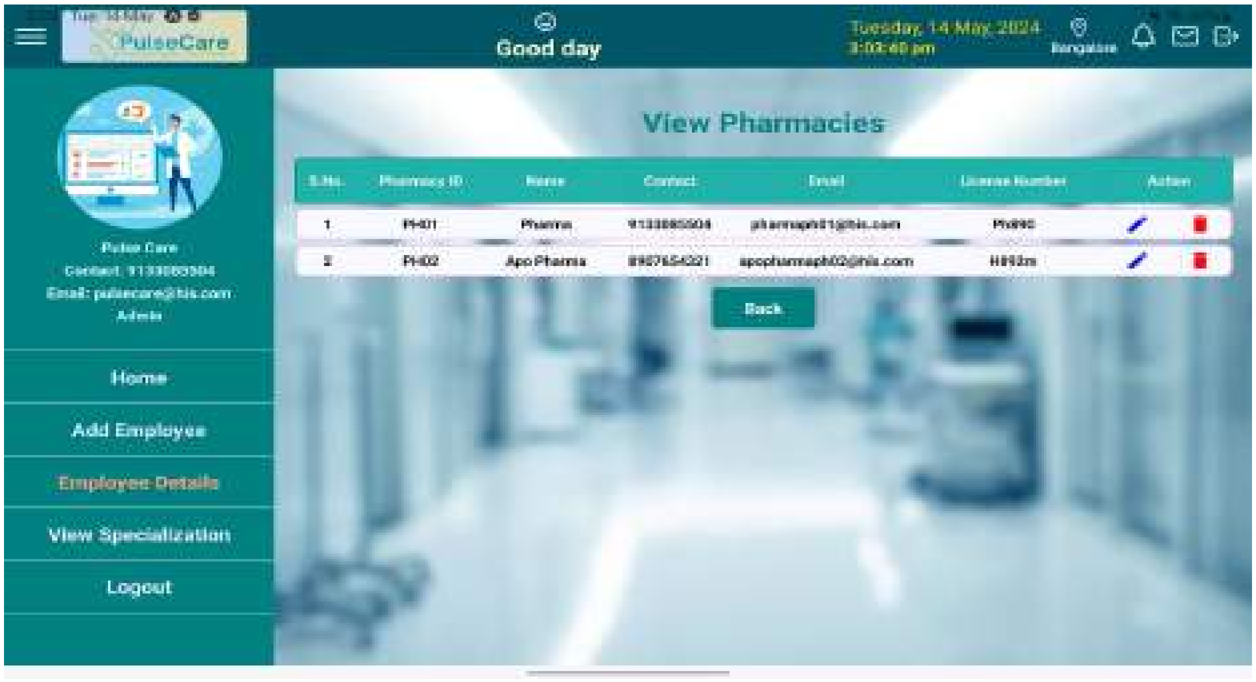
c. Receptionist



View Receptionists

S.No.	Receptionist ID	Name	Contact	Email	Action
1	R01	Bhumika	9748767088	bhumikar01@his.com	
2	R02	Sitara	6549087113	sitarar02@his.com	

d. Pharmacy



View Pharmacies

S.No.	Pharmacy ID	Name	Contact	Email	License Number	Action
1	PH01	Pharma	9133085504	pharmaph01@his.com	Ph890	
2	PH02	Apo Pharma	8907654321	apopharmaph02@his.com	H892m	

4. Edit Employees can be performed for :-

a. Doctor
b. Nurse
c. Receptionist
d. Pharmacy

From the view details page on clicking the edit icon we’ll be redirected to the edit page, where the form will come with the already filled details. Then we can change the details and then after successful updation, we’ll be redirected to the corresponding view page.

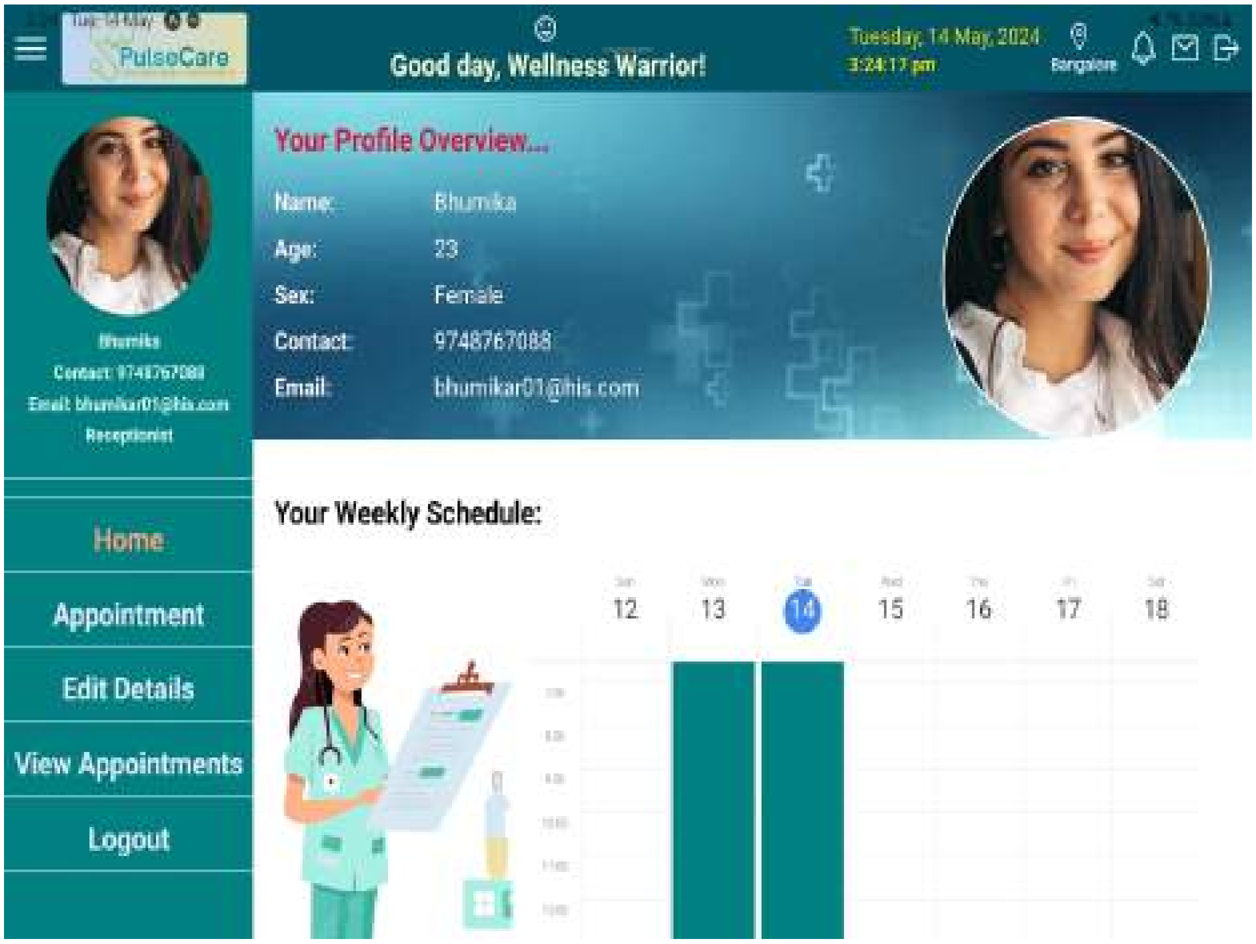
5. Deactivate employees can be performed for :-

a. Doctor
b. Nurse
c. Receptionist
d. Pharmacy

From the view details page on clicking the delete icon then after successful deactivation, we’ll be redirected to the corresponding view page.

RECEPTIONIST

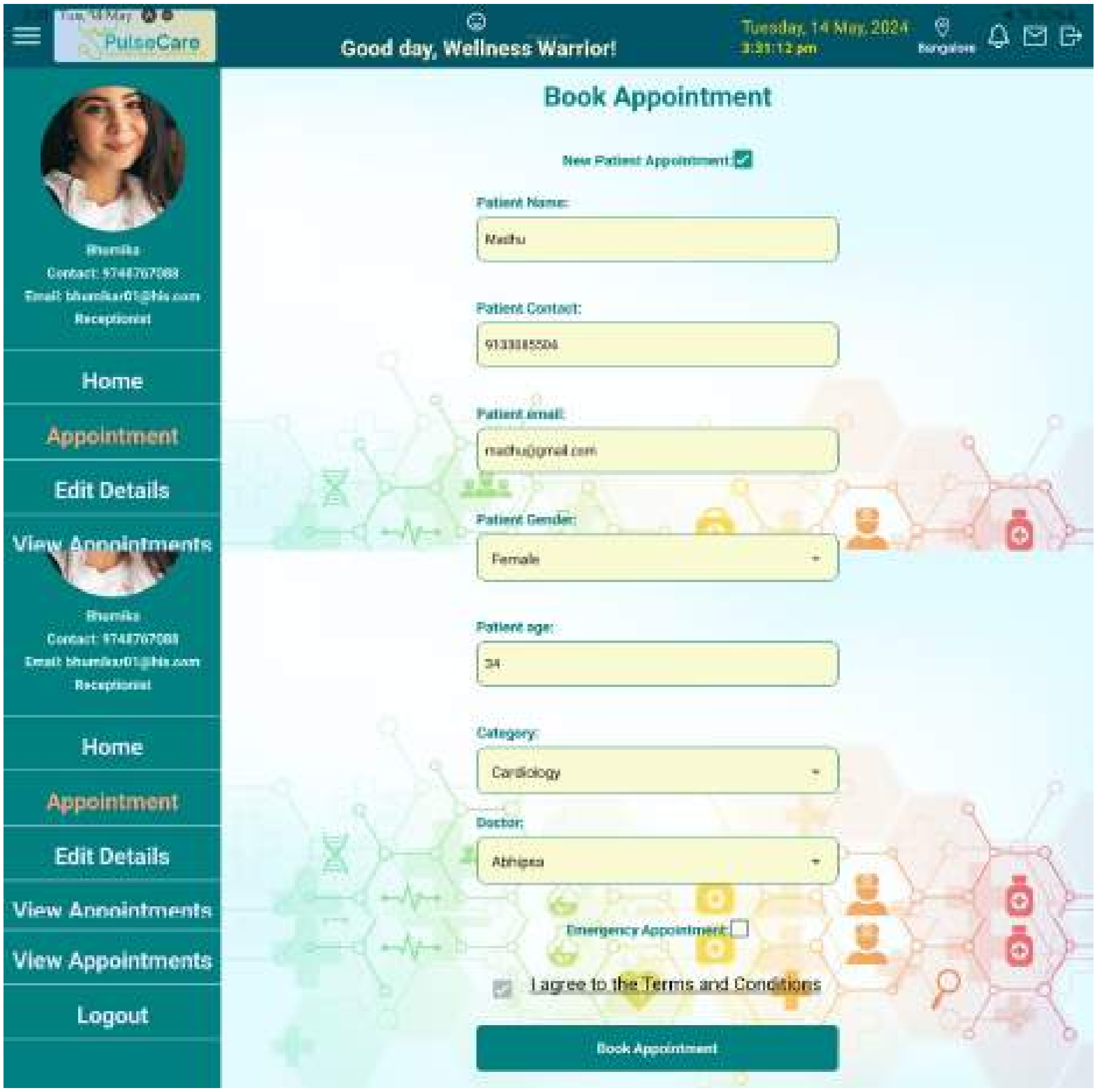
1. Dashboard



Your Profile Overview...

Name:	Bhumika
Age:	23
Sex:	Female
Contact:	9748767088
Email:	bhumikar01@his.com

2. Book appointment for new patient



Book Appointment

New Patient Appointment: ✓

Patient Name: Madhu

Patient Contact: 9133845504

Patient email: madhu@gmail.com

Patient Gender: Female

Patient age: 34

Category: Cardiology

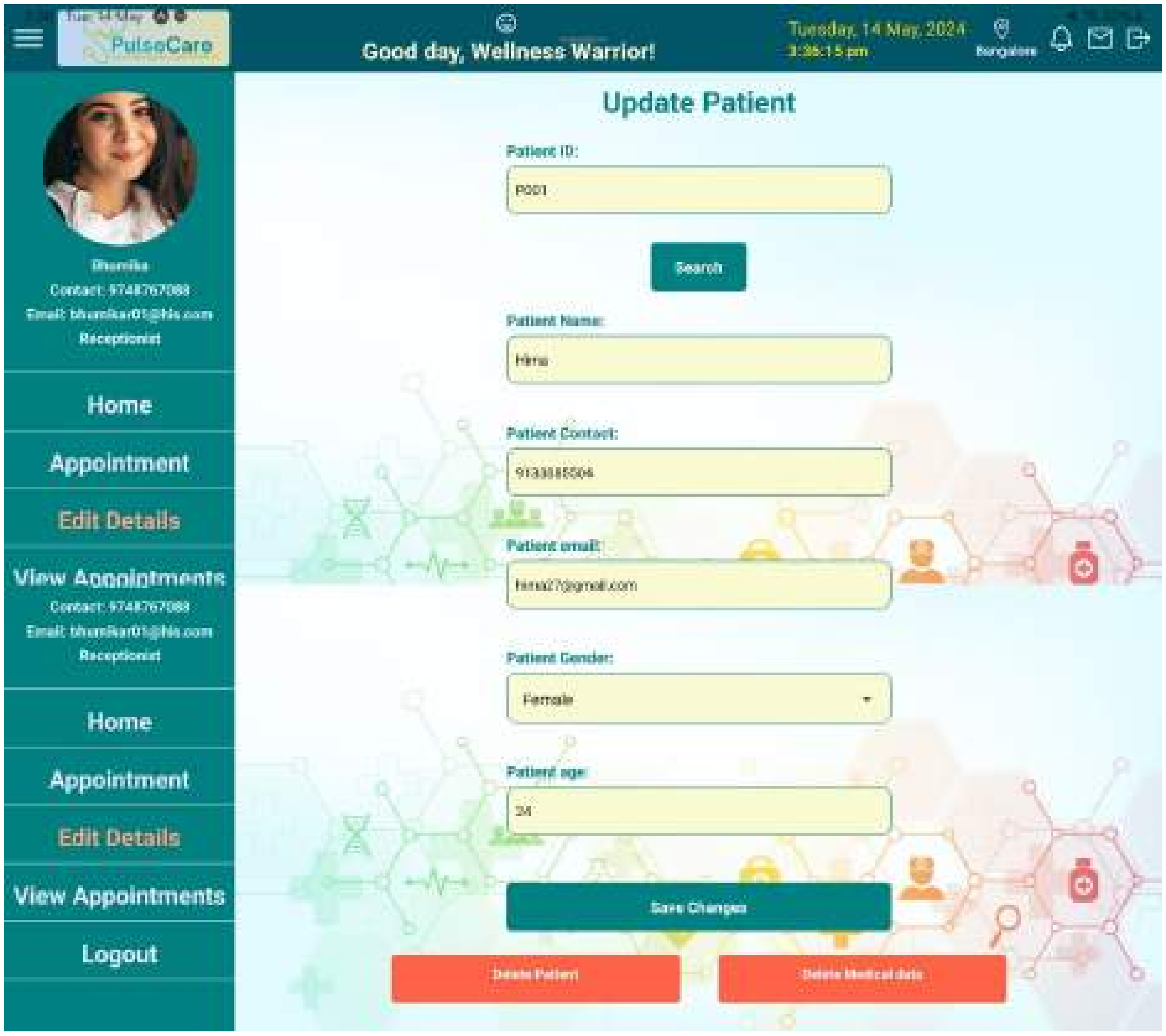
Doctor: Abhipsa

Emergency Appointment: ☐

✓ I agree to the Terms and Conditions

Book Appointment

3. Update patient details



Bhumika
Contact: 9748767088
Email: bhumikar01@his.com
Receptionist

Update Patient

Patient ID: P001

Patient Name: Hima

Patient Contact: 9133885504

Patient email: hima27@gmail.com

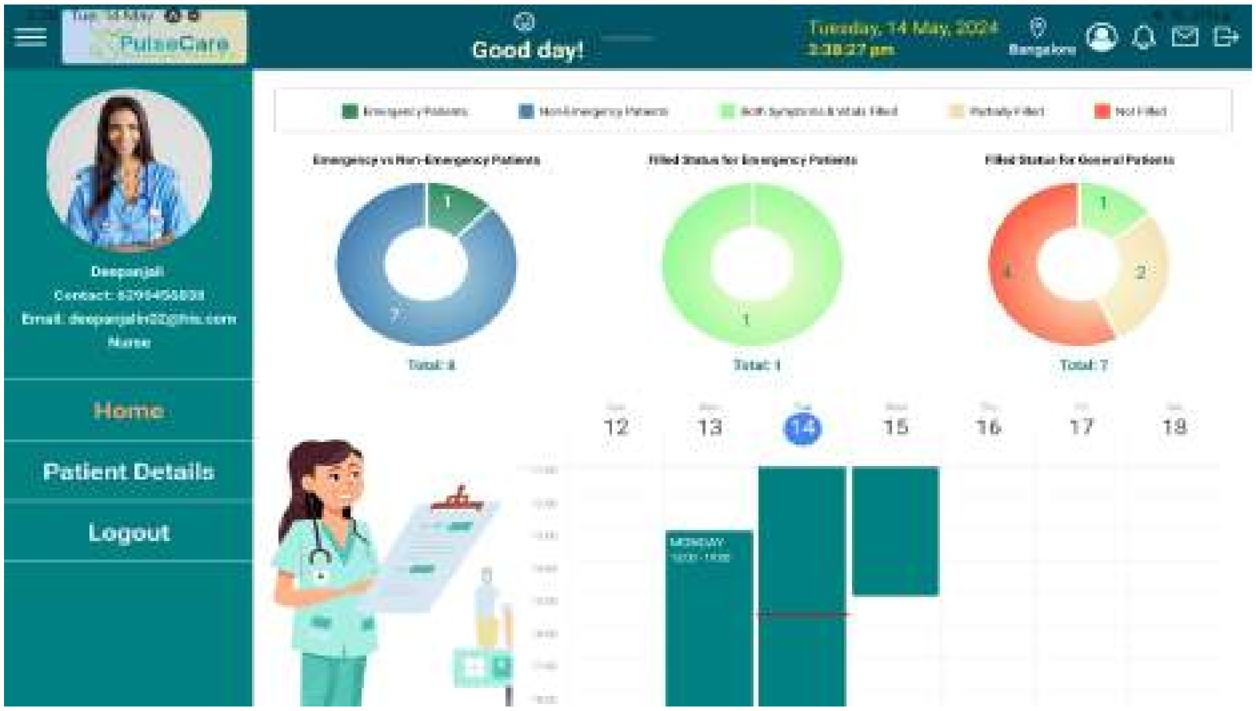
Patient Gender: Female

Patient age: 24

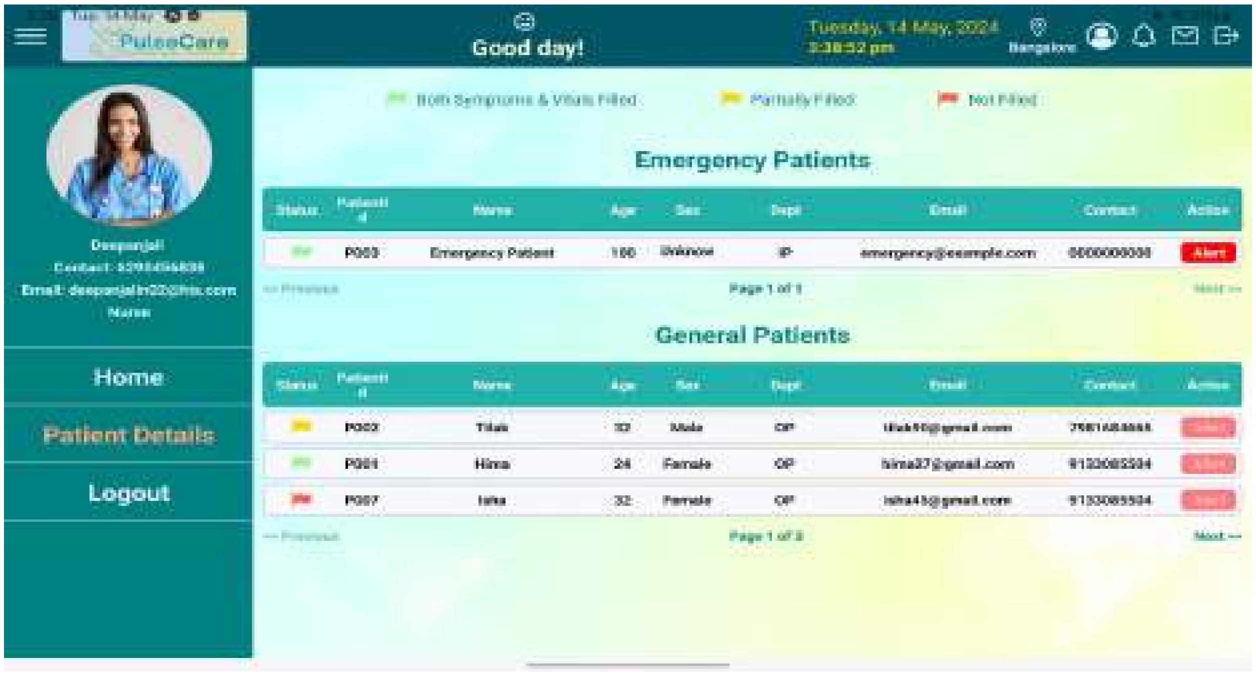
We can also book appointment for already existing patients.

NURSE

1. Nurse Dashboard.



2. Patient List



🟩 Both Symptoms & Vitals Filled 🟨 Partially Filled 🟥 Not Filled

Emergency Patients

Status	Patient Id	Name	Age	Sex	Dept	Email	Contact	Action
🟩	P003	Emergency Patient	100	Unknown	IP	emergency@example.com	0000000000	Alert

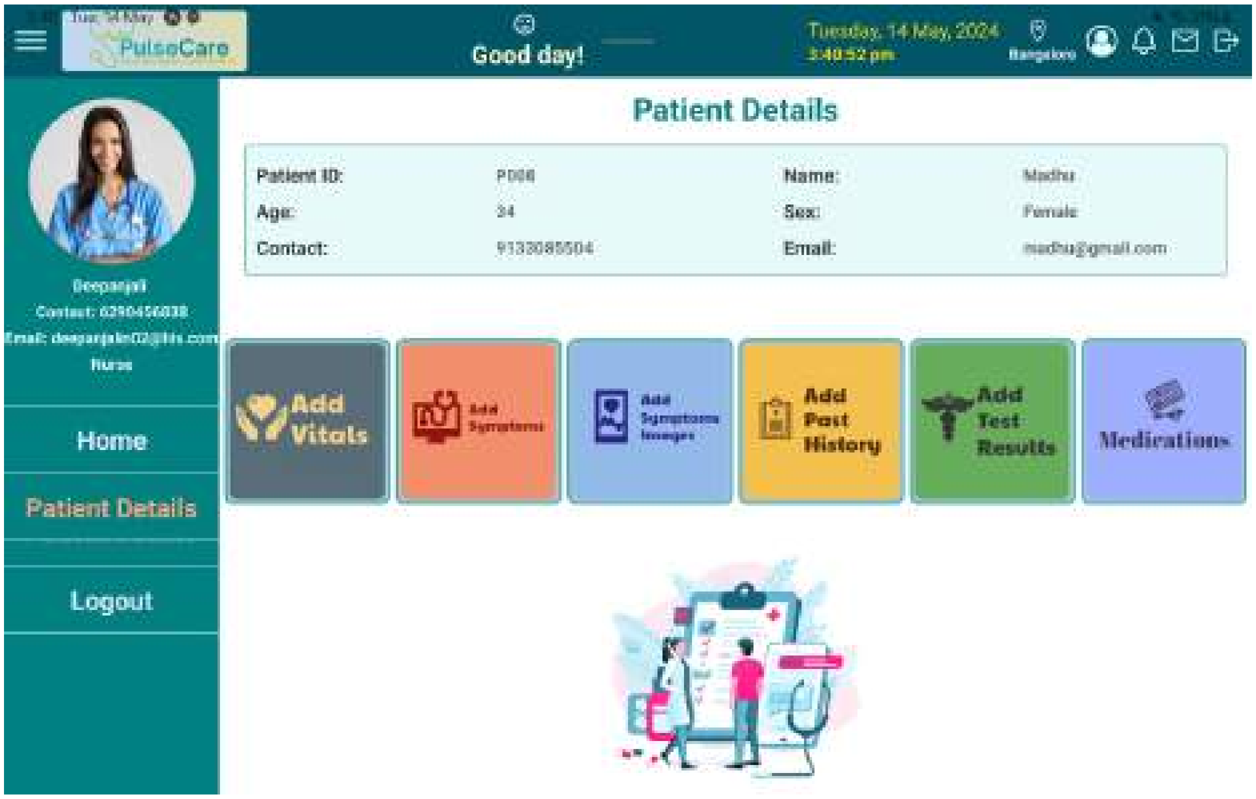
← Previous Page 1 of 1 Next →

General Patients

Status	Patient Id	Name	Age	Sex	Dept	Email	Contact	Action
🟨	P002	Tilak	32	Male	OP	tilak10@gmail.com	7981464665	Alert
🟩	P001	Hima	24	Female	OP	hima27@gmail.com	9133085504	Alert
🟥	P007	Isha	32	Female	OP	isha45@gmail.com	9133085504	Alert

← Previous Page 1 of 3 Next →

3. Patient details dashboard


PulseCare

Good day!

Tuesday, 14 May, 2024
3:40:52 pm

Bangalore

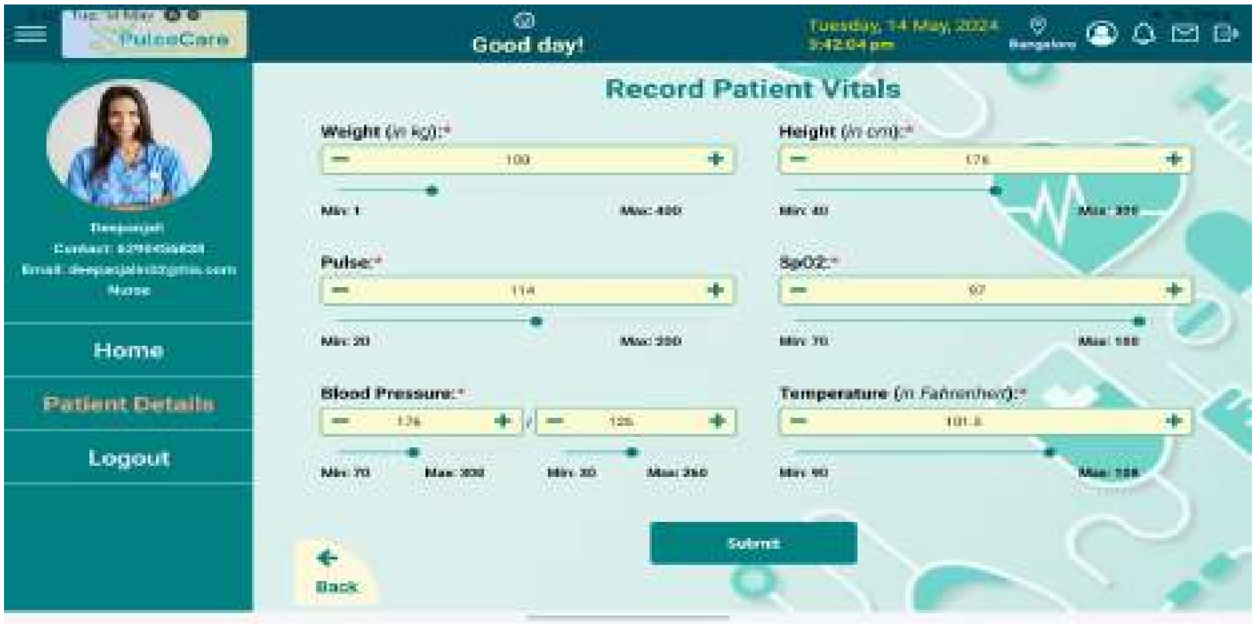
Deepanjali
Contact: 6290456038
Email: deepanjalin02@hls.com
Nurse

Home
Patient Details
Logout

Patient Details

Patient ID:	P006	Name:	Madhu
Age:	34	Sex:	Female
Contact:	9133085504	Email:	madhu@gmail.com

4. Add Vitals


PulseCare

Good day!

Tuesday, 14 May, 2024
3:42:04 pm

Bangalore

Deepanjali
Contact: 6290456038
Email: deepanjalin02@hls.com
Nurse

Home
Patient Details
Logout

Record Patient Vitals

Weight (*in kg*):* 100 — Min: 1, Max: 400

Height (*in cm*):* 176 — Min: 40, Max: 300

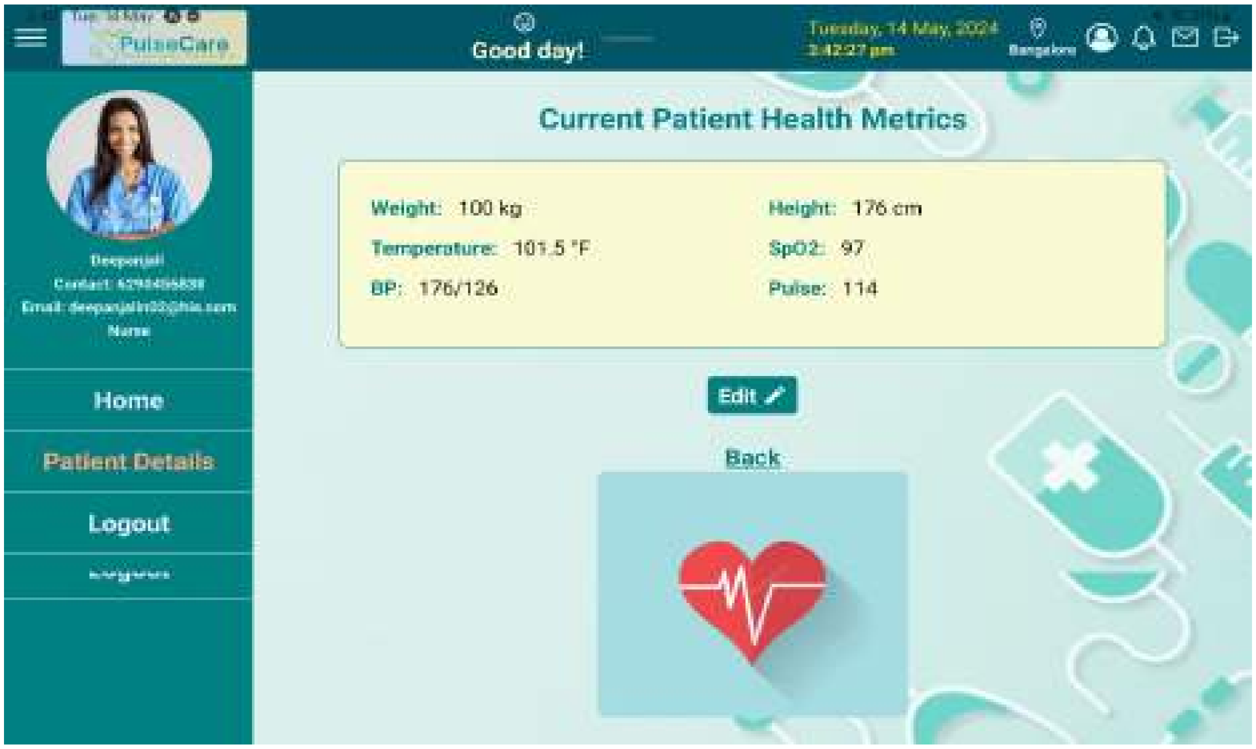
Pulse:* 114 — Min: 20, Max: 200

SpO2:* 97 — Min: 70, Max: 100

Blood Pressure:* 176 / 126 — Min: 70, Max: 300; Min: 30, Max: 260

Temperature (*in Fahrenheit*):* 101.8 — Min: 90, Max: 108

5. View Vitals (once added)


Good day!

Tuesday, 14 May, 2024 3:42:27 pm

Bangalore

Deepanjali
Contact: 6290456838
Email: deepanjalin22@hie.com
Nurse

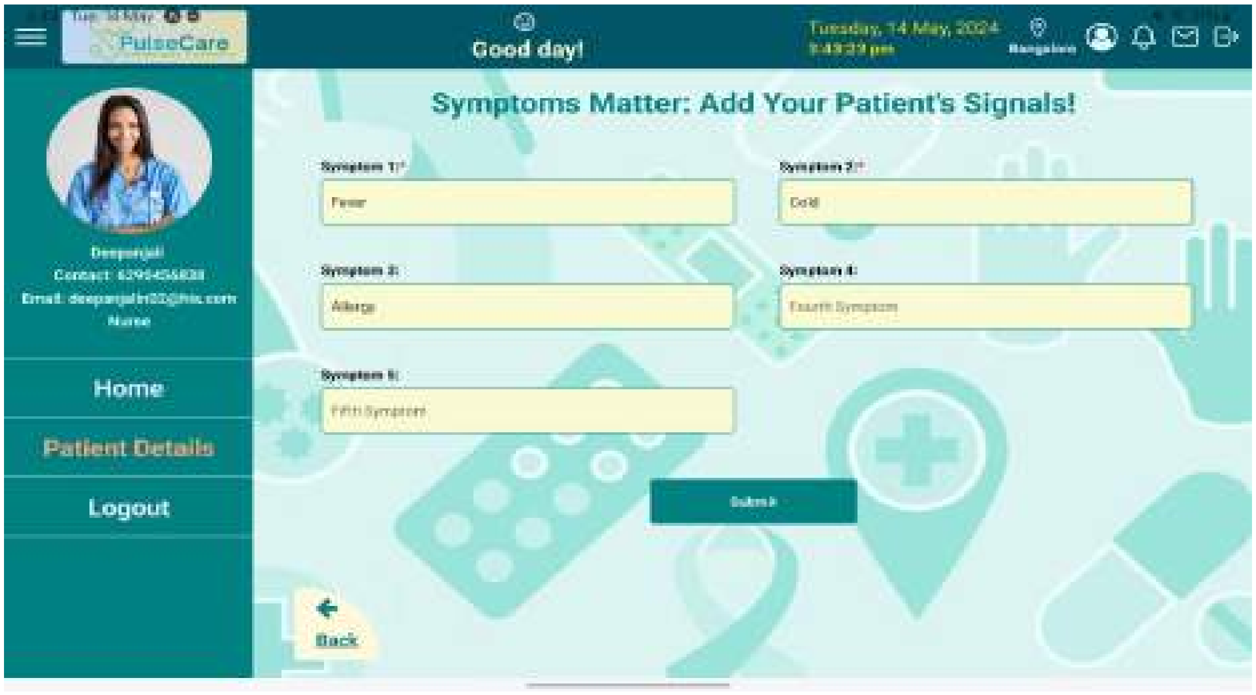
Home
Patient Details
Logout

Current Patient Health Metrics

Weight: 100 kg
Height: 176 cm
Temperature: 101.5 °F
SpO2: 97
BP: 176/126
Pulse: 114

Edit
Back

6. Add Symptoms


Good day!

Tuesday, 14 May, 2024 3:43:23 pm

Bangalore

Deepanjali
Contact: 6290456838
Email: deepanjalin22@hie.com
Nurse

Home
Patient Details
Logout

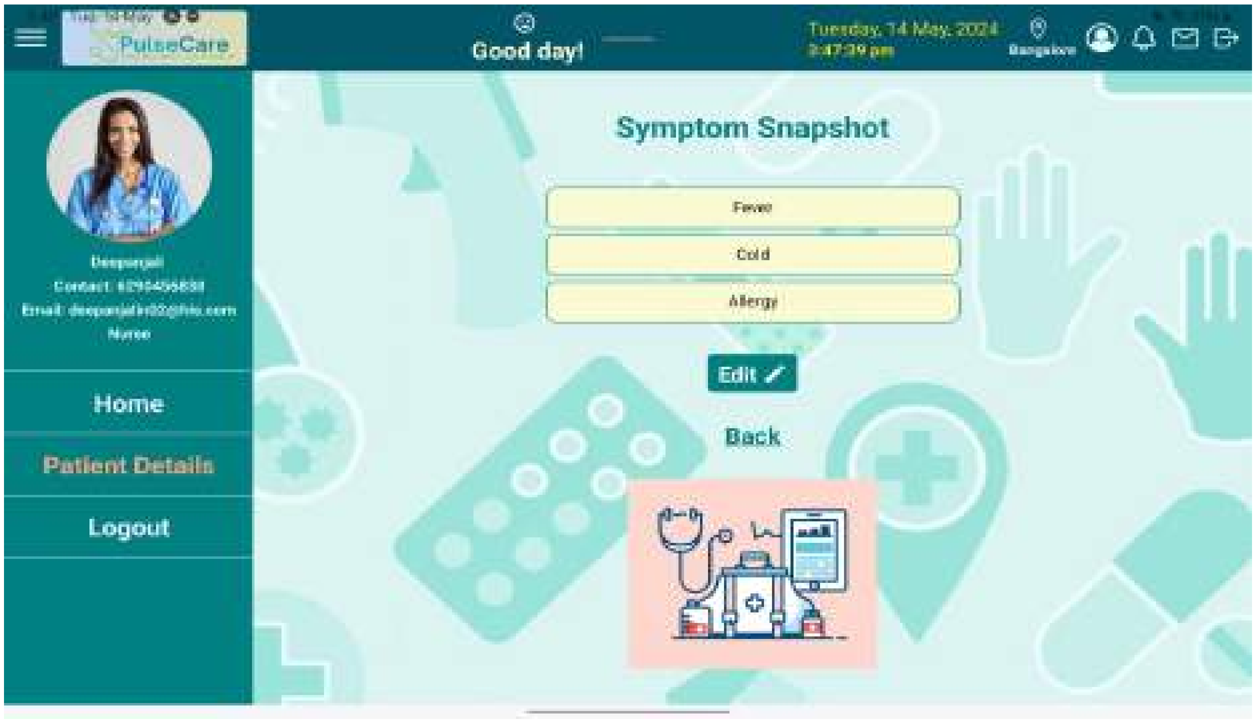
Symptoms Matter: Add Your Patient's Signals!

Symptom 1:* Fever
Symptom 2:* Cold
Symptom 3: Allergy
Symptom 4: Fourth Symptom
Symptom 5: Fifth Symptom

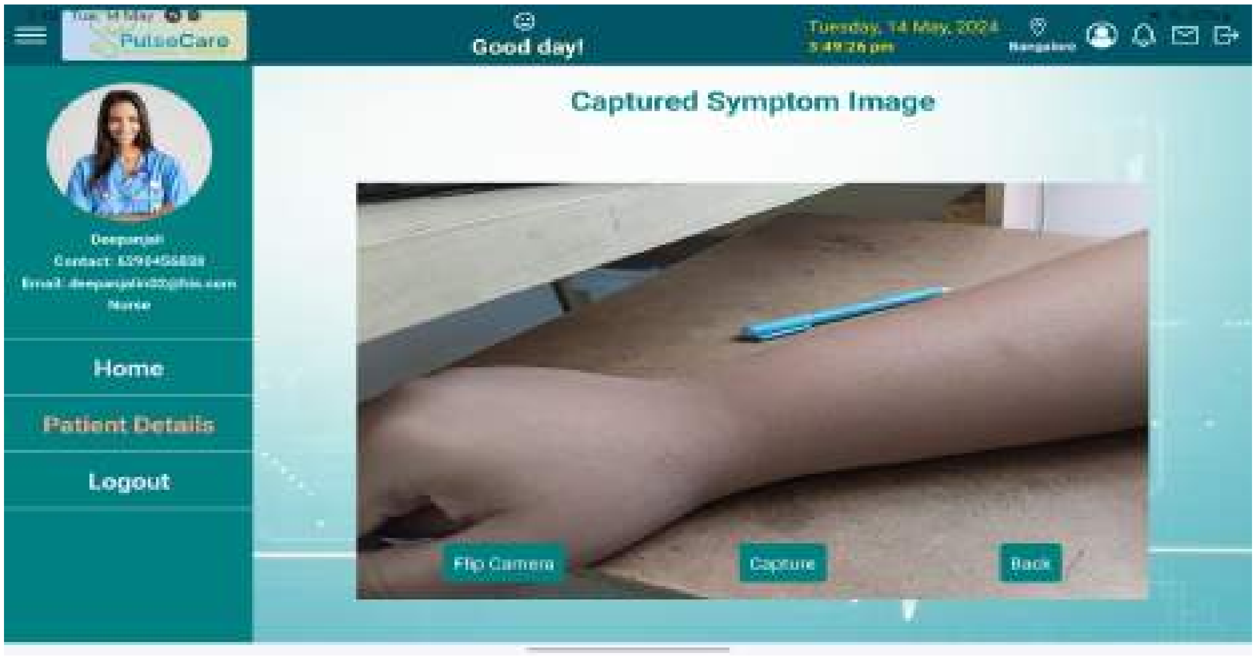
Submit

Back

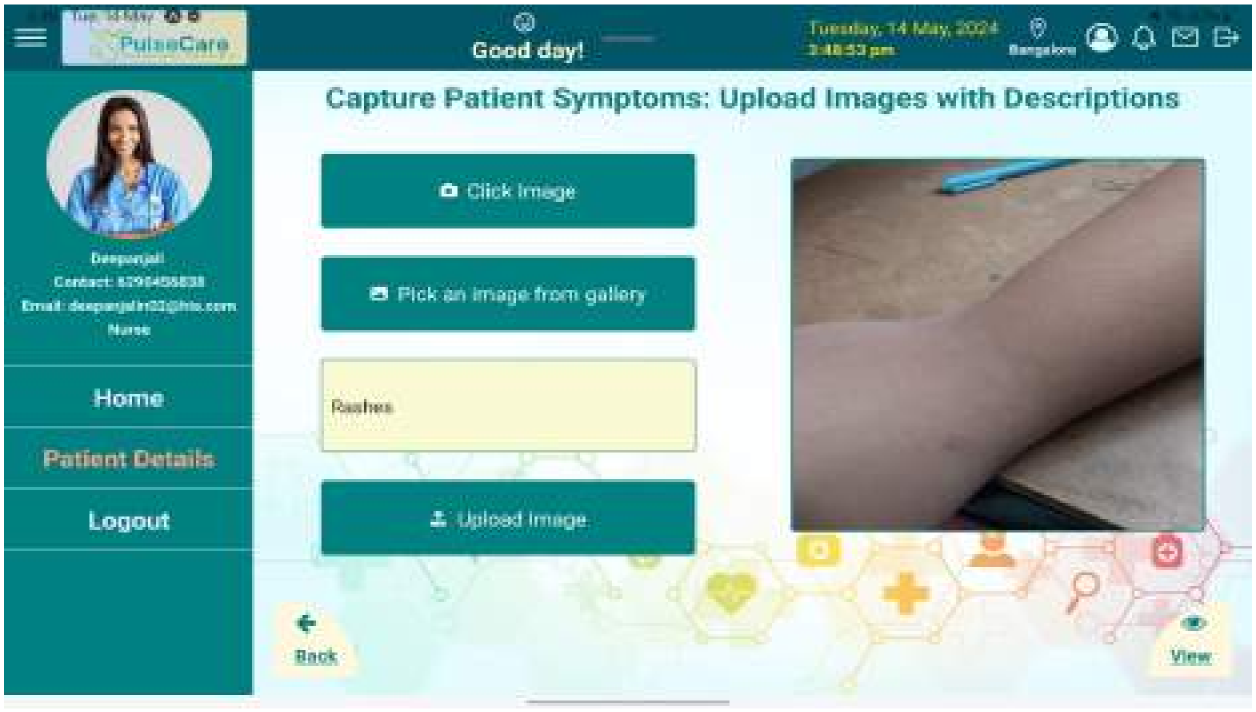
7. View symptoms



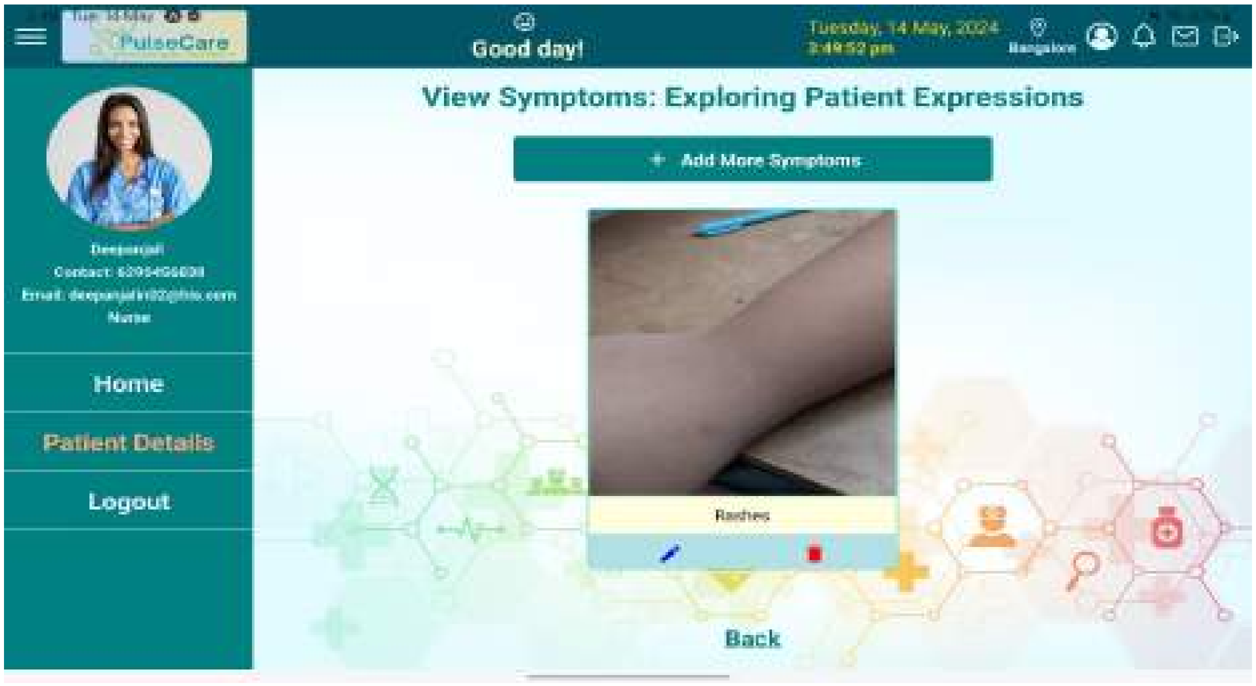
8. Add Symptom images
    a. Capture image



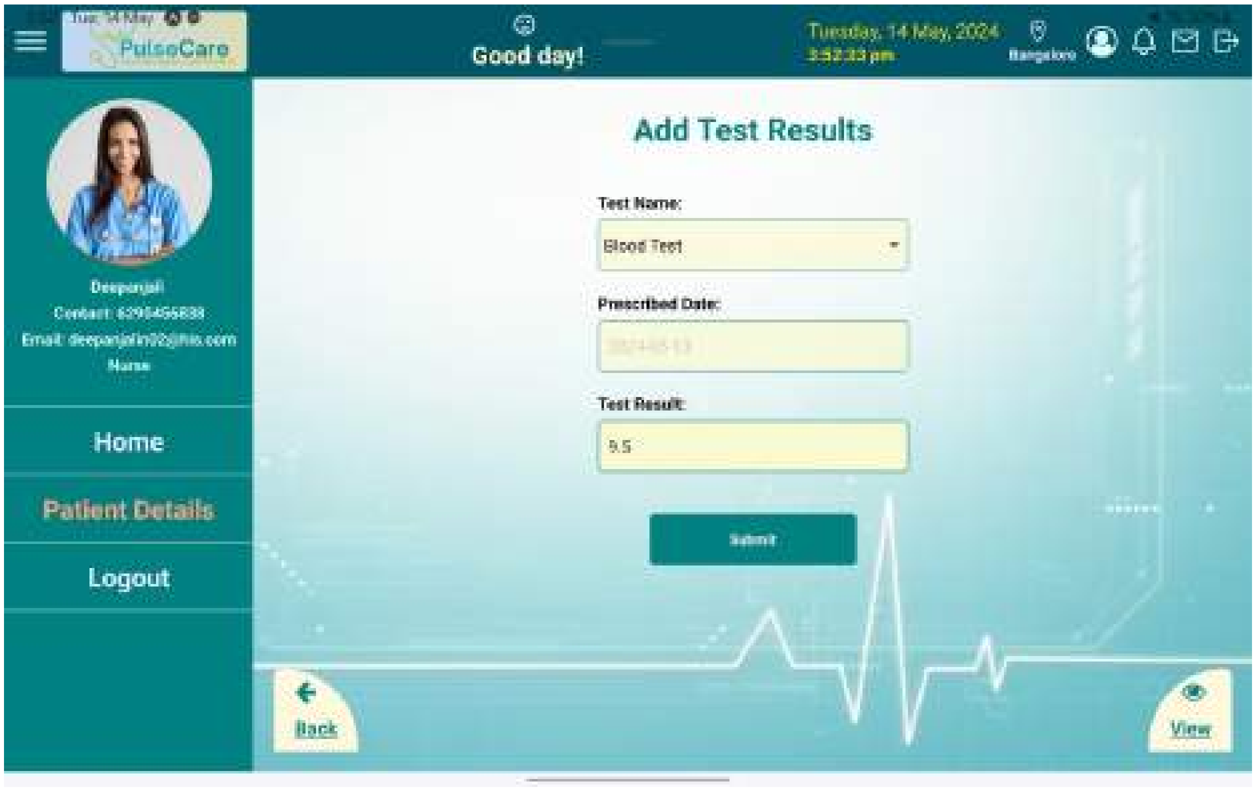
    b. Upload image



9. View Added Symptom images



10. Add Test Results



Deepanjali
Contact: 6295456838
Email: deepanjalin02@his.com
Nurse

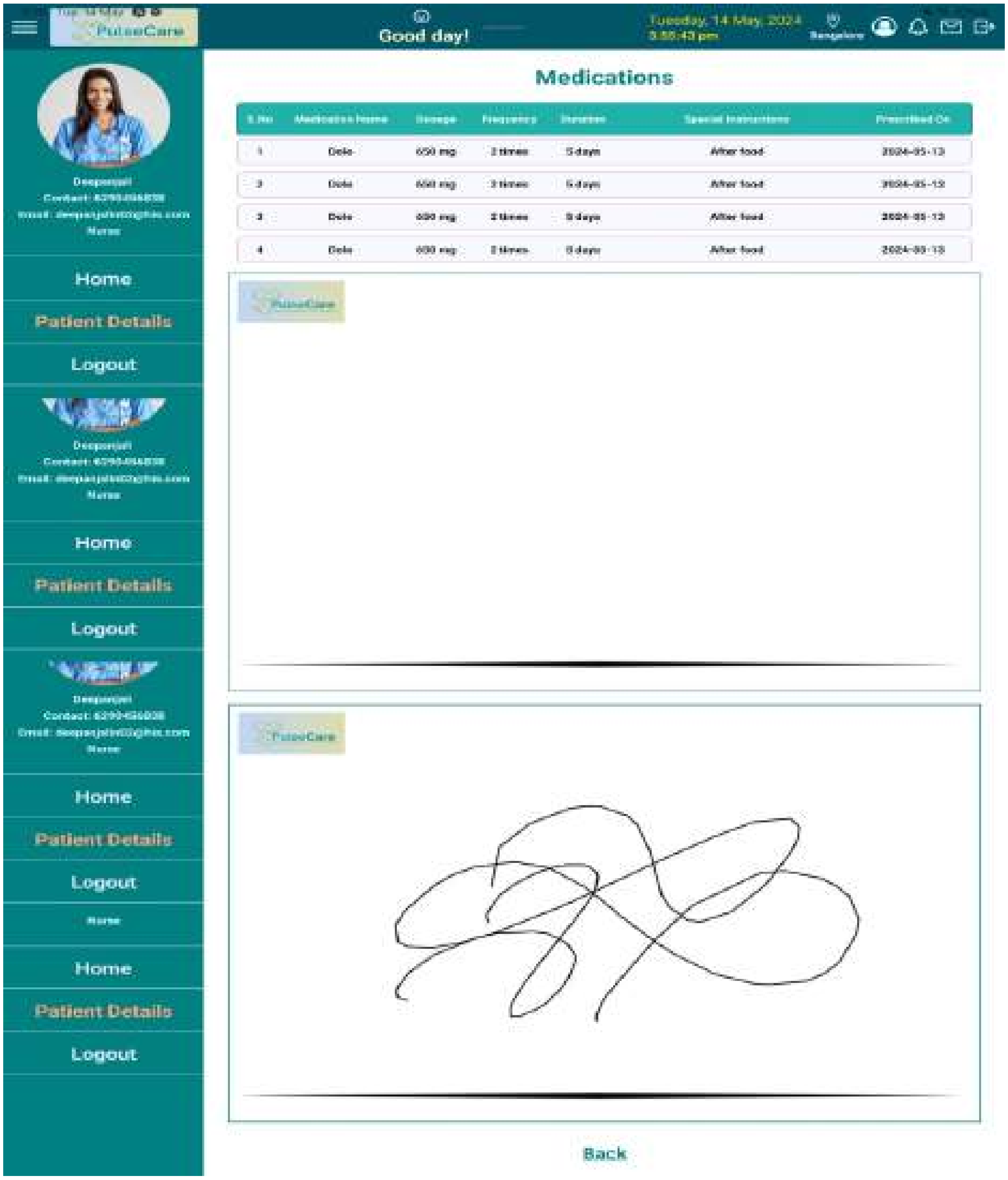
Add Test Results

Test Name: Blood Test

Prescribed Date: 2024-05-13

Test Result: 9.5

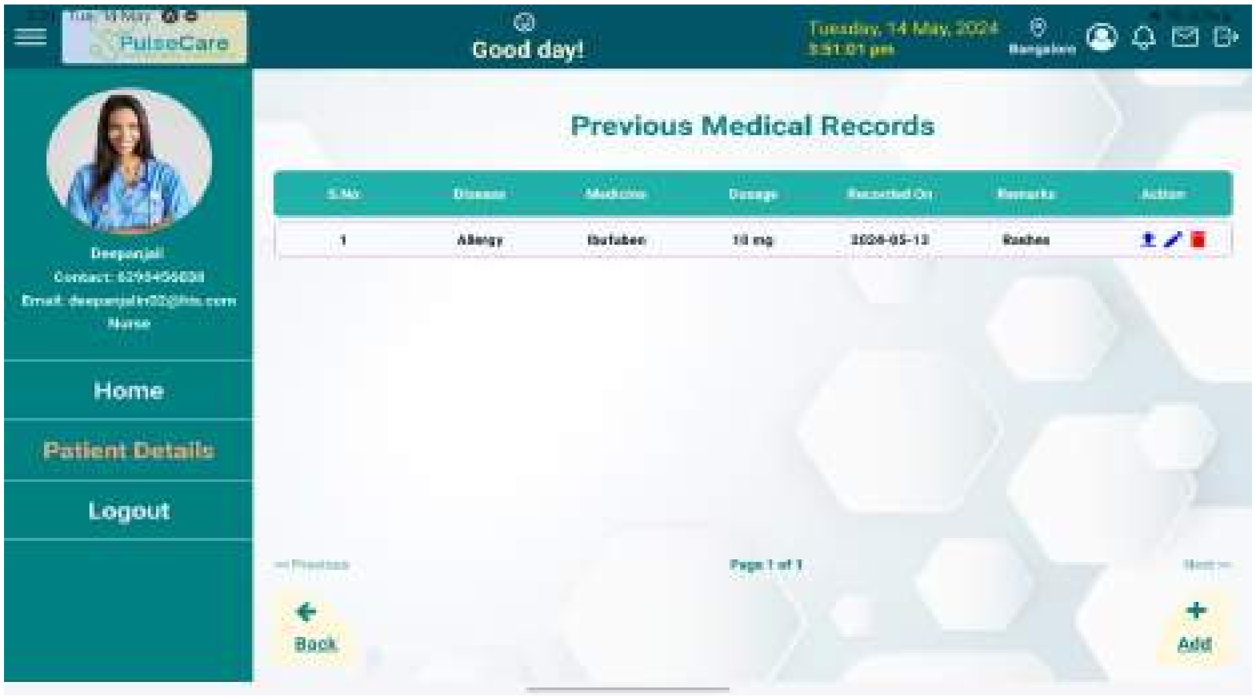
11. View Medications



Medications

S.No	Medication Name	Dosage	Frequency	Duration	Special Instructions	Prescribed On
1	Dolo	650 mg	2 times	5 days	After food	2024-05-13
2	Dolo	650 mg	2 times	5 days	After food	2024-05-13
3	Dolo	650 mg	2 times	5 days	After food	2024-05-13
4	Dolo	650 mg	2 times	5 days	After food	2024-05-13

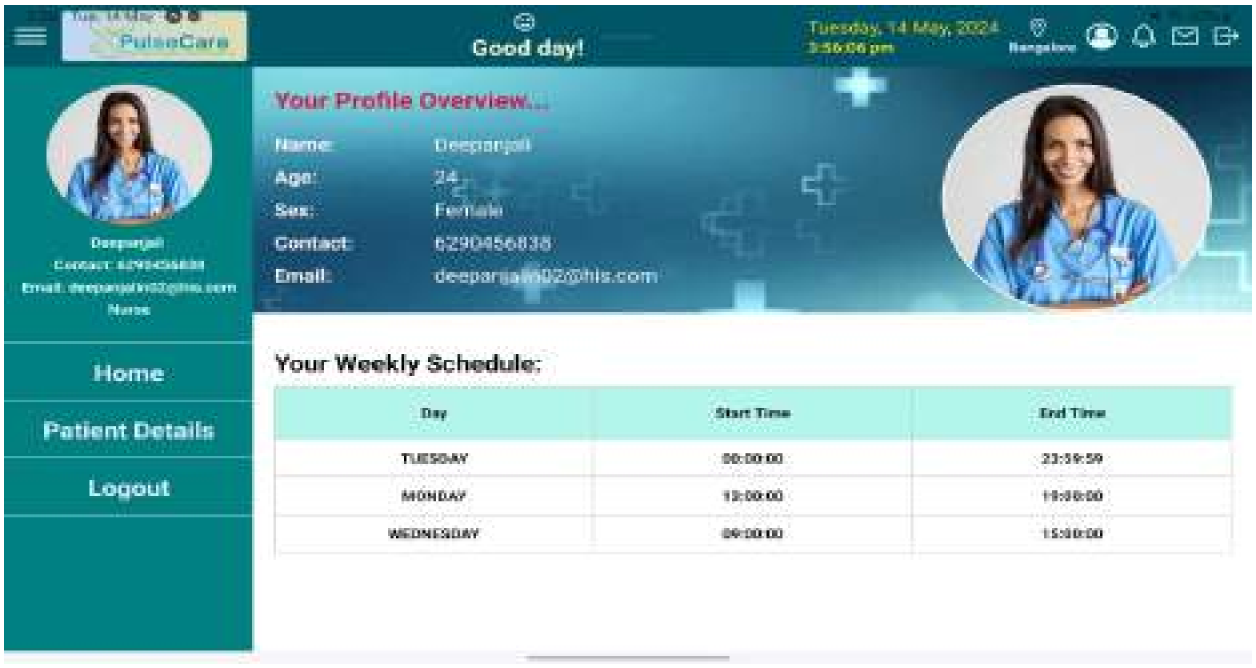
12. View Past Records



Previous Medical Records

S.No	Disease	Medicine	Dosage	Recorded On	Remarks	Action
1	Allergy	Ibufuben	10 mg	2024-05-13	Rashes	

13. Nurse Profile



Your Profile Overview...

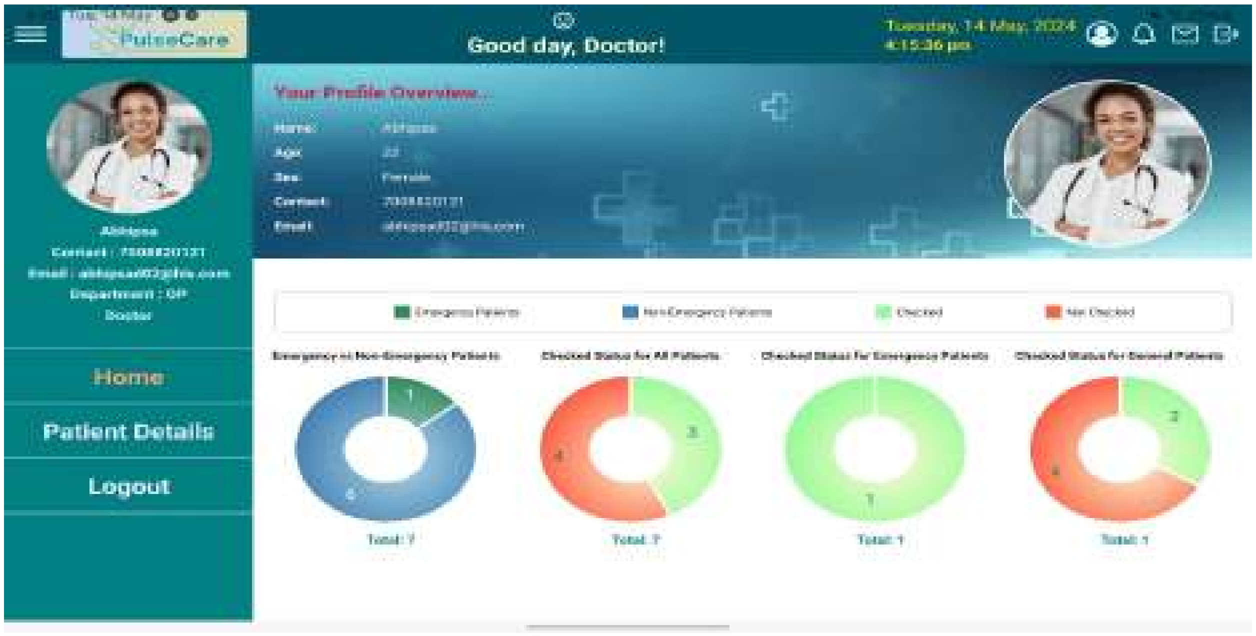
Name:	Deepanjali
Age:	24
Sex:	Female
Contact:	6290456838
Email:	deepanjalin02@his.com

Your Weekly Schedule:

Day	Start Time	End Time
TUESDAY	00:00:00	23:59:59
MONDAY	12:00:00	18:00:00
WEDNESDAY	09:00:00	15:00:00

4. DOCTOR

1. Dashboard
   a. OP doctor



PulseCare

Good day, Doctor!

Tuesday, 14 May, 2024
4:15:36 pm

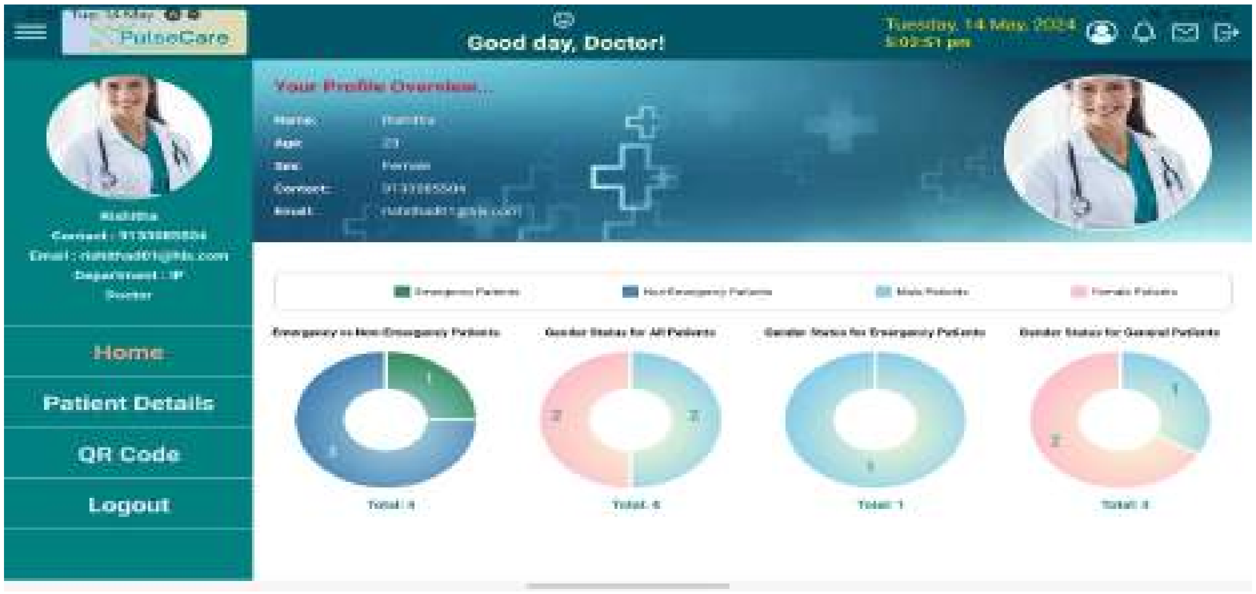
Your Profile Overview...

Name: Abhipsa
Age: 22
Sex: Female
Contact: 7008820121
Email: abhipsa02@his.com

Abhipsa
Contact : 7008820121
Email : abhipsa02@his.com
Department : OP Doctor

Home
Patient Details
Logout
Emergency Patients
Non-Emergency Patients
Checked
Not Checked
Emergency vs Non-Emergency Patients
Checked Status for All Patients
Checked Status for Emergency Patients
Checked Status for General Patients
Total: 7
Total: 7
Total: 1
Total: 1

   b. IP doctor



PulseCare

Good day, Doctor!

Tuesday, 14 May, 2024
5:02:51 pm

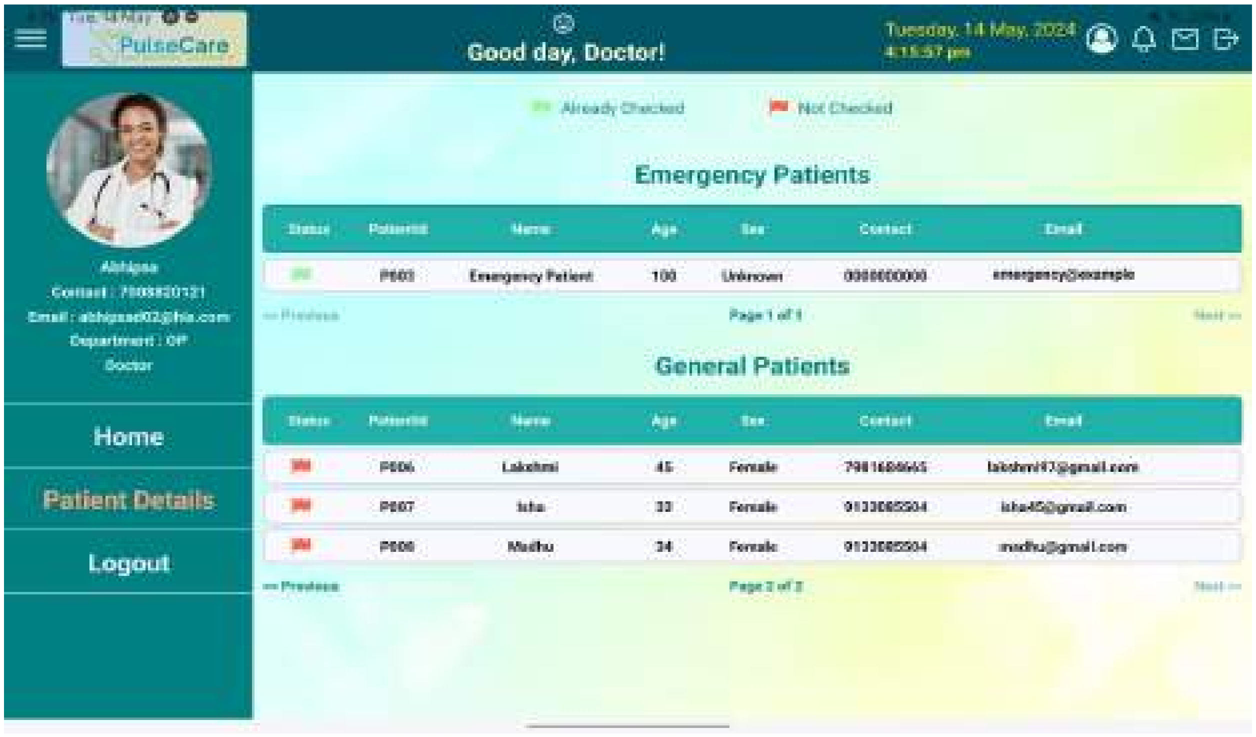
Your Profile Overview...

Name: Rishitha
Age: 21
Sex: Female
Contact: 9133985504
Email: rishithad01@his.com

Rishitha
Contact : 9133985504
Email : rishithad01@his.com
Department : IP Doctor

Home
Patient Details
QR Code
Logout
Emergency Patients
Non-Emergency Patients
Male Patients
Female Patients
Emergency vs Non-Emergency Patients
Gender Status for All Patients
Gender Status for Emergency Patients
Gender Status for General Patients
Total: 4
Total: 4
Total: 1
Total: 3

2. Patient list



Abhipsa
Contact : 7008820121
Email : abhipsad02@hia.com
Department : OP
Doctor

🟩 Already Checked 🟥 Not Checked

Emergency Patients

Status	PatientId	Name	Age	Sex	Contact	Email
🟩	P603	Emergency Patient	100	Unknown	0000000000	emergency@example

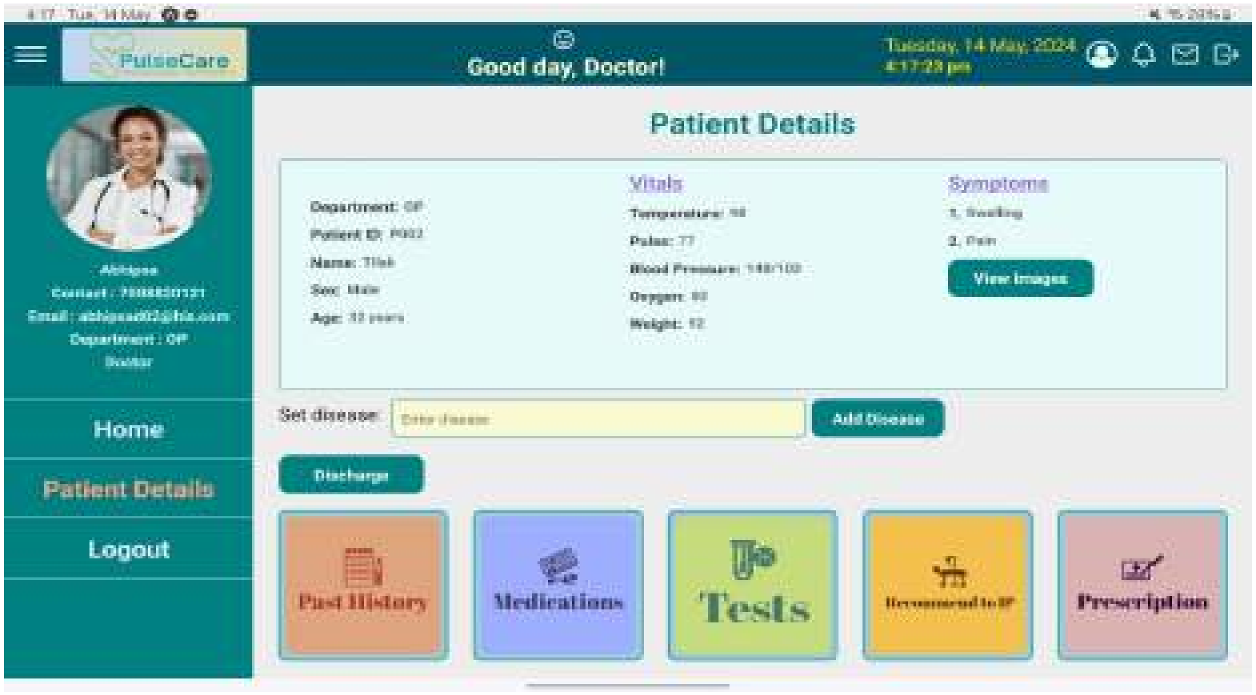
<< Previous Page 1 of 1 Next >>

General Patients

Status	PatientId	Name	Age	Sex	Contact	Email
🟥	P606	Lakshmi	45	Female	7981680665	lakshmi47@gmail.com
🟥	P607	Isha	33	Female	9133085504	isha45@gmail.com
🟥	P608	Madhu	34	Female	9133085504	madhu@gmail.com

<< Previous Page 2 of 2 Next >>

3. Patient dashboard



Abhipsa
Contact : 7008820121
Email : abhipsad02@his.com
Department : OP
Doctor

Patient Details

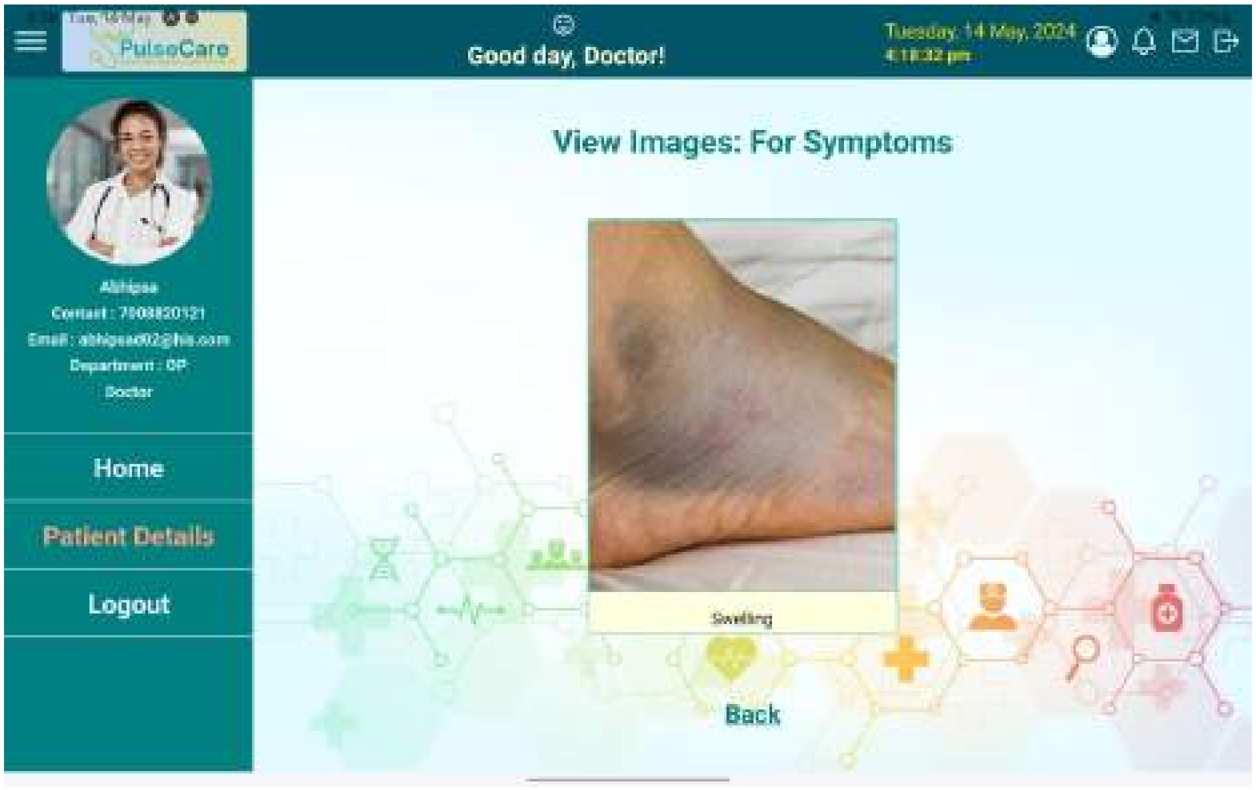
Department: OP
Patient ID: P002
Name: Tilak
Sex: Male
Age: 32 years

Vitals
Temperature: 98
Pulse: 77
Blood Pressure: 148/102
Oxygen: 92
Weight: 72

Symptoms
1. Swelling
2. Pain

Set disease: Enter disease

4. View Symptom Images

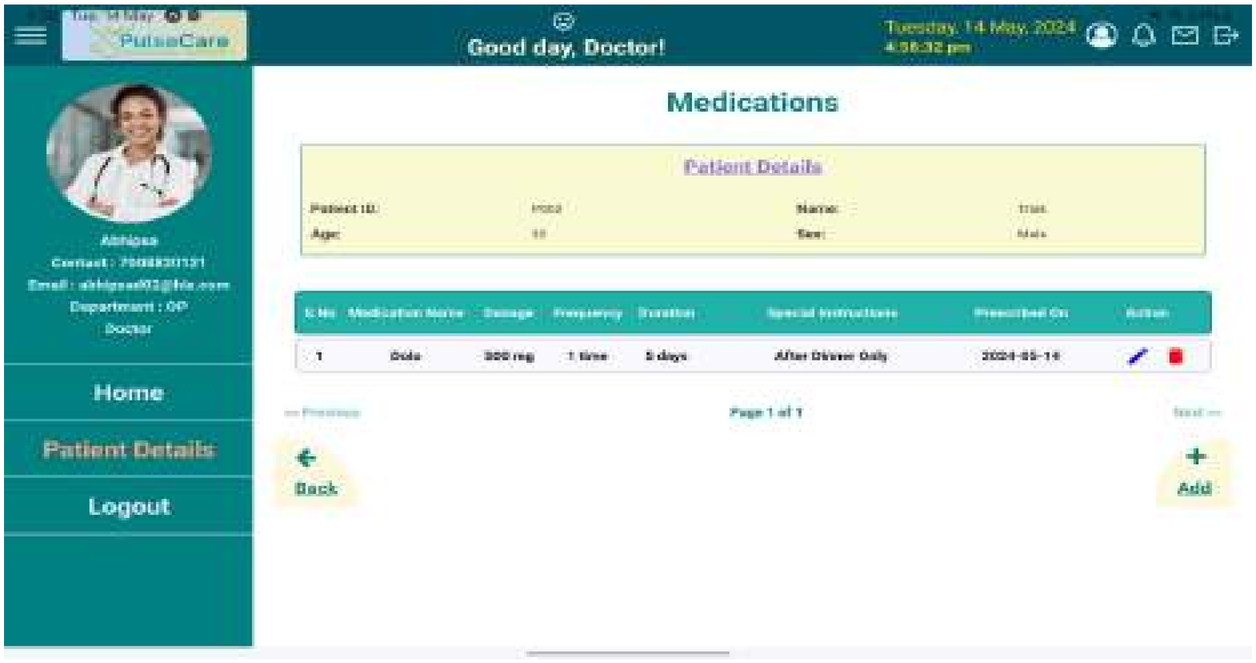


Abhipsa
Contact : 7008820121
Email : abhipsad02@his.com
Department : OP
Doctor

View Images: For Symptoms

Swelling

5. View Medications


PulseCare

Good day, Doctor!

Tuesday, 14 May, 2024
4:58:32 pm

Abhipsa
Contact : 7008820121
Email : abhipsa002@fda.com
Department : OP
Doctor

Home

Patient Details

Logout

Medications

Patient Details

Patient ID:	P002	Name:	Trial
Age:	33	Sex:	Male

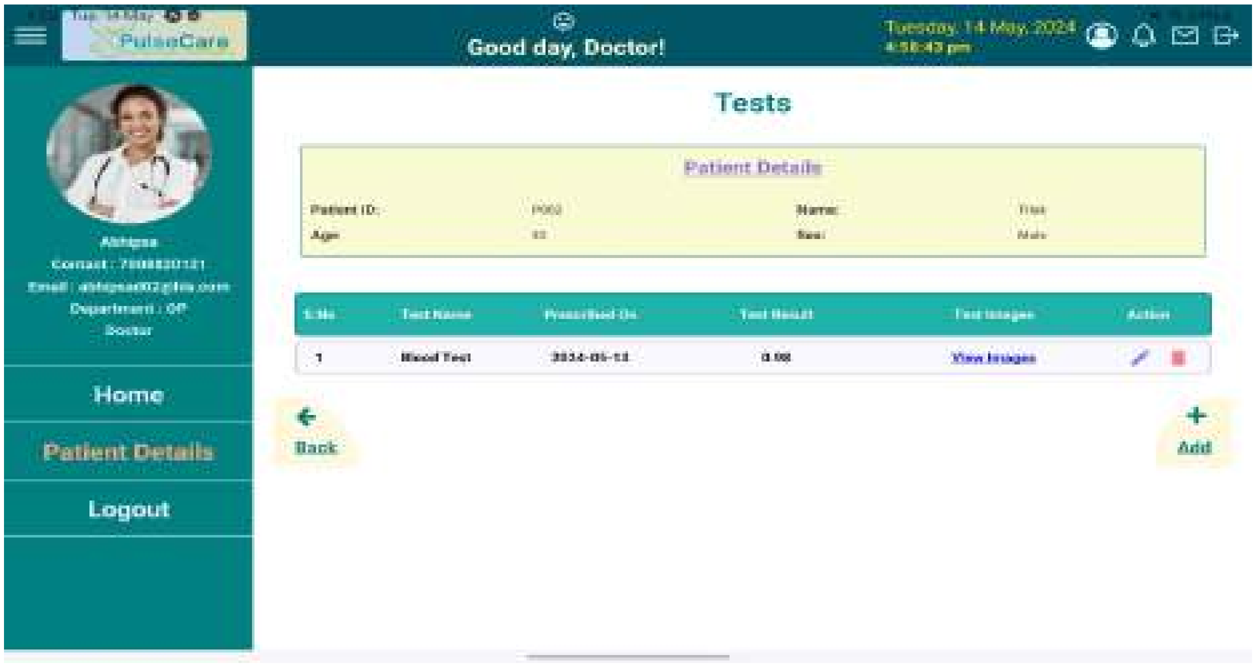
S.No	Medication Name	Dosage	Frequency	Duration	Special Instructions	Prescribed On	Action
1	Dolo	500 mg	1 time	5 days	After Dinner Only	2024-05-14	

← Previous Page 1 of 1 Next →

Back

Add

6. View Tests


PulseCare

Good day, Doctor!

Tuesday, 14 May, 2024
4:58:43 pm

Abhipsa
Contact : 7008820121
Email : abhipsa002@fda.com
Department : OP
Doctor

Home

Patient Details

Logout

Tests

Patient Details

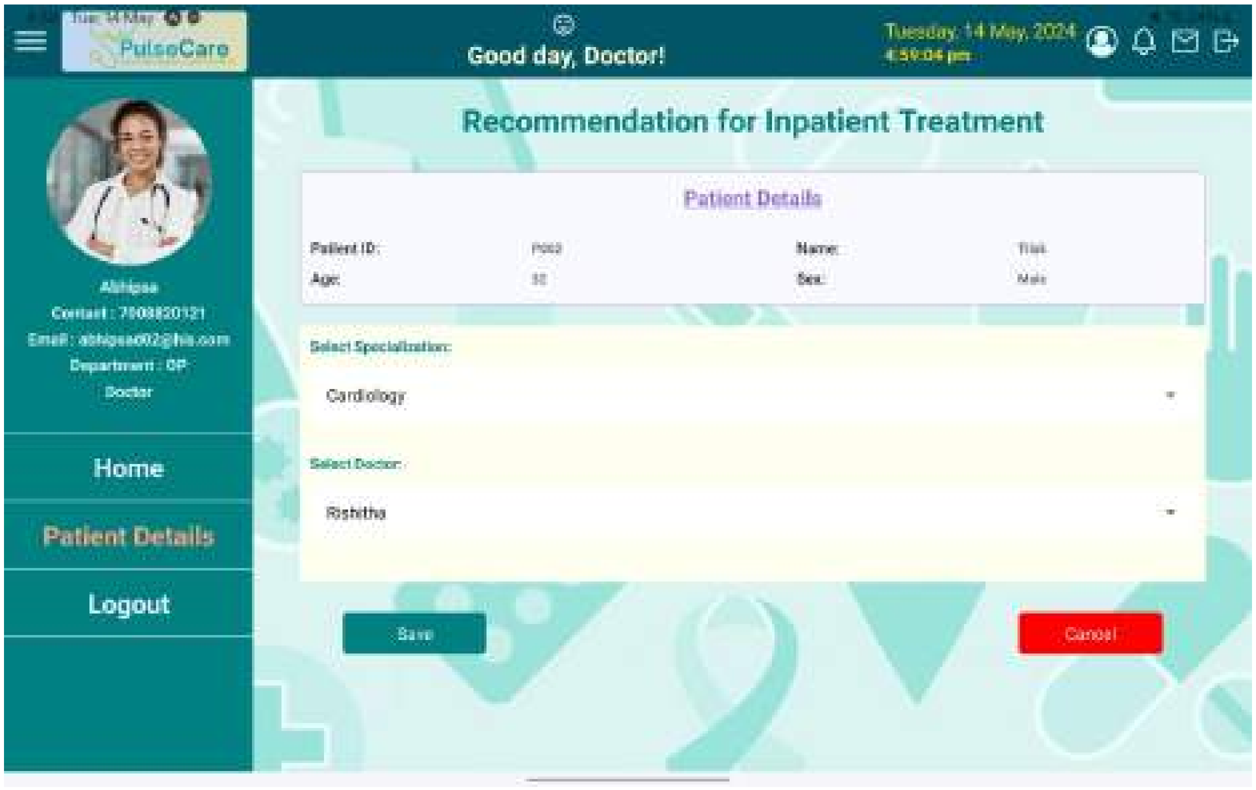
Patient ID:	P002	Name:	Trial
Age:	33	Sex:	Male

S.No	Test Name	Prescribed On	Test Result	Test Images	Action
1	Blood Test	2024-05-13	0.00	View Images	

Back

Add

7. Recommend to IP (OP doctor)



Abhipsa
Contact : 7008820121
Email : abhipsad02@his.com
Department : OP Doctor

Recommendation for Inpatient Treatment

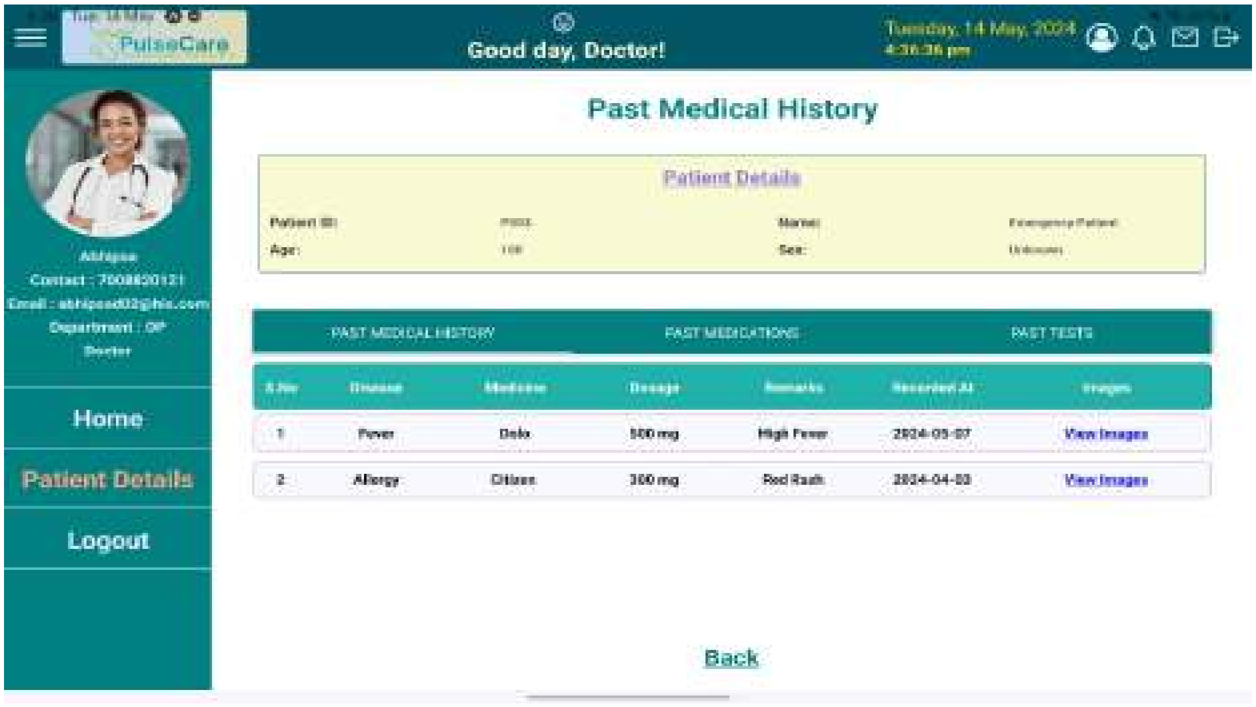
Patient Details

Patient ID:	P003	Name:	Tilak
Age:	32	Sex:	Male

Select Specialization:
Cardiology

Select Doctor:
Rishitha

8. Past Medical History



Abhipsa
Contact : 7008820121
Email : abhipsad02@his.com
Department : OP Doctor

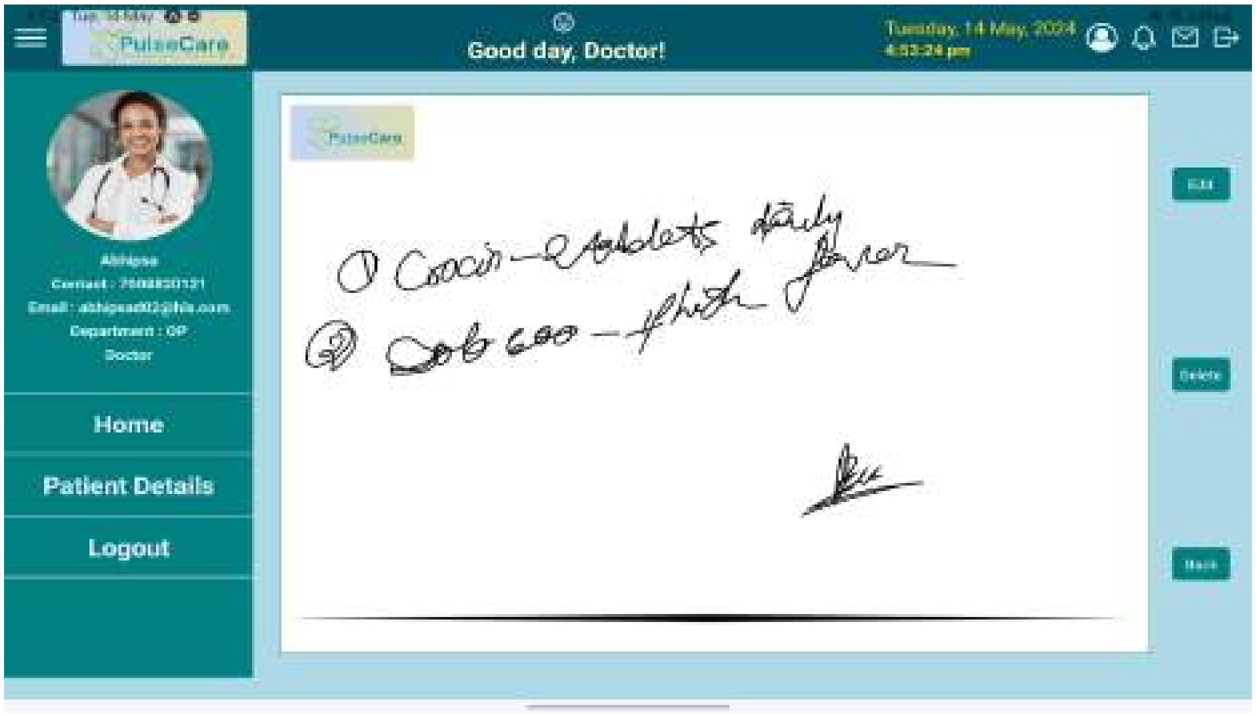
Past Medical History

Patient Details

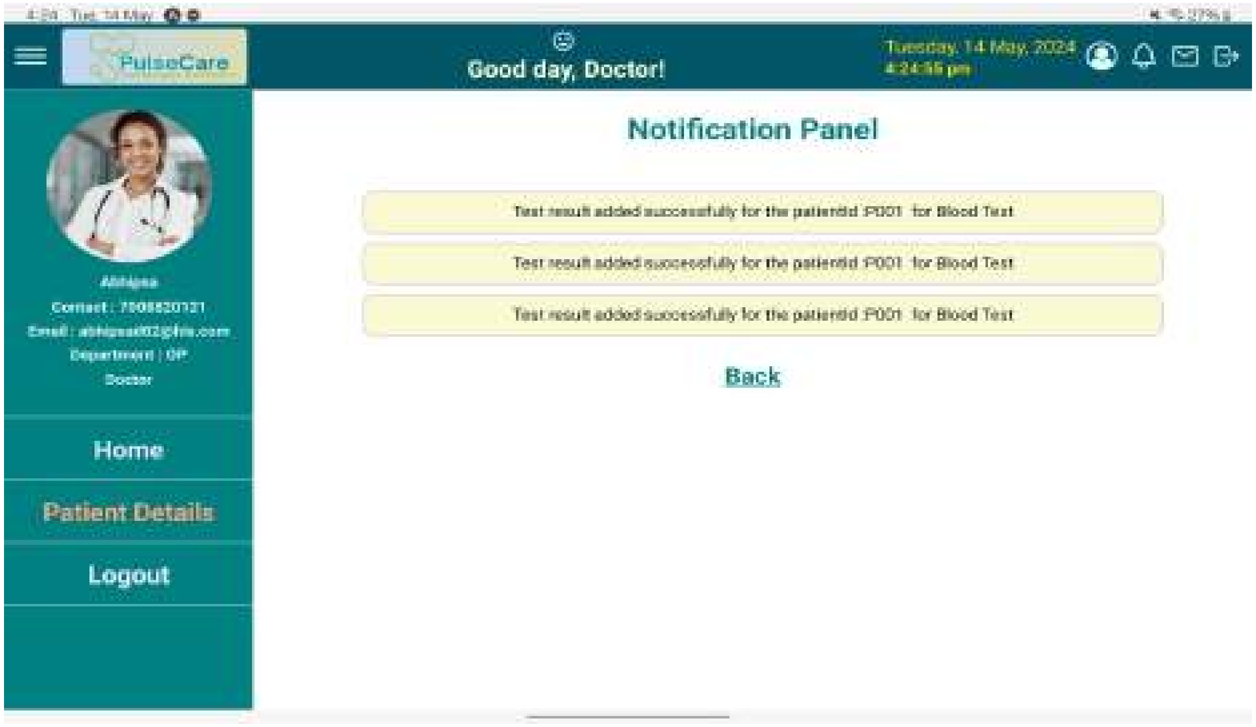
Patient ID:	P003	Name:	Emergency Patient
Age:	100	Sex:	Unknown

S.No	Disease	Medicine	Dosage	Remarks	Recorded At	Images
1	Fever	Dolo	500 mg	High Fever	2024-05-07	View Images
2	Allergy	Citizen	300 mg	Red Rash	2024-04-03	View Images

9. Add Prescription



10. Notification Panel



Notification Panel

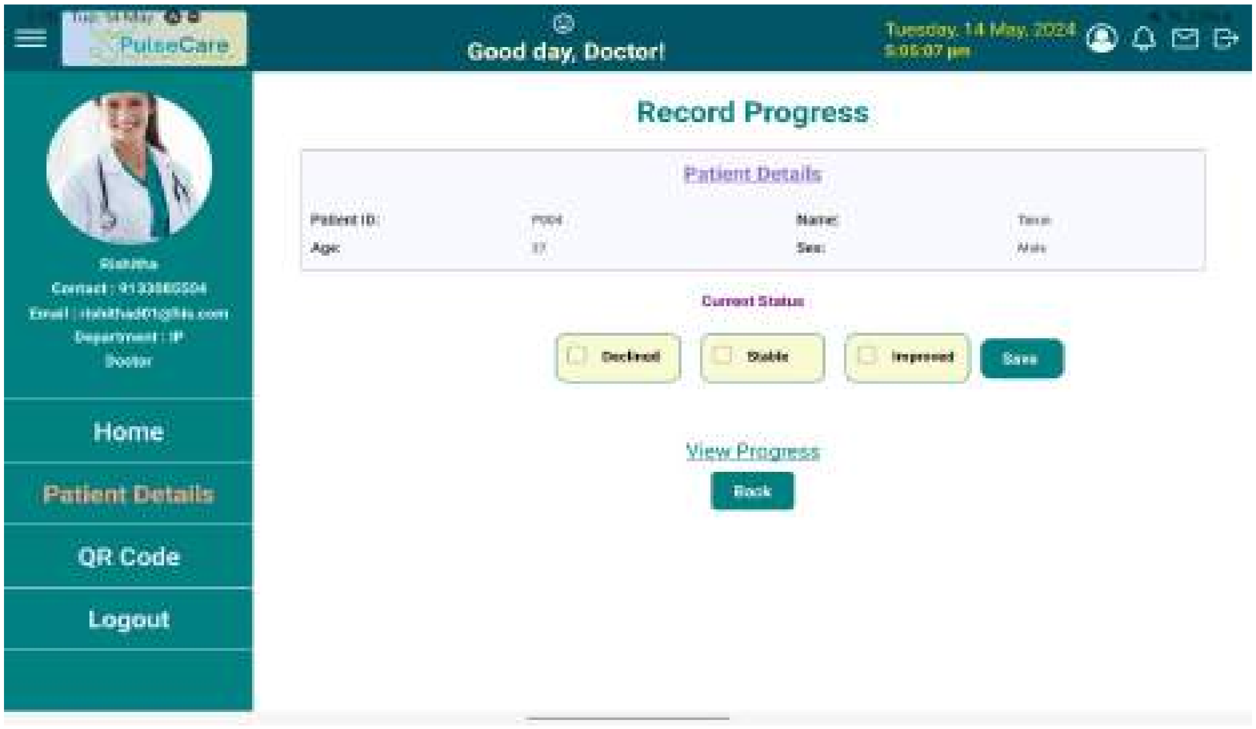
Test result added successfully for the patientId :P001 for Blood Test

Test result added successfully for the patientId :P001 for Blood Test

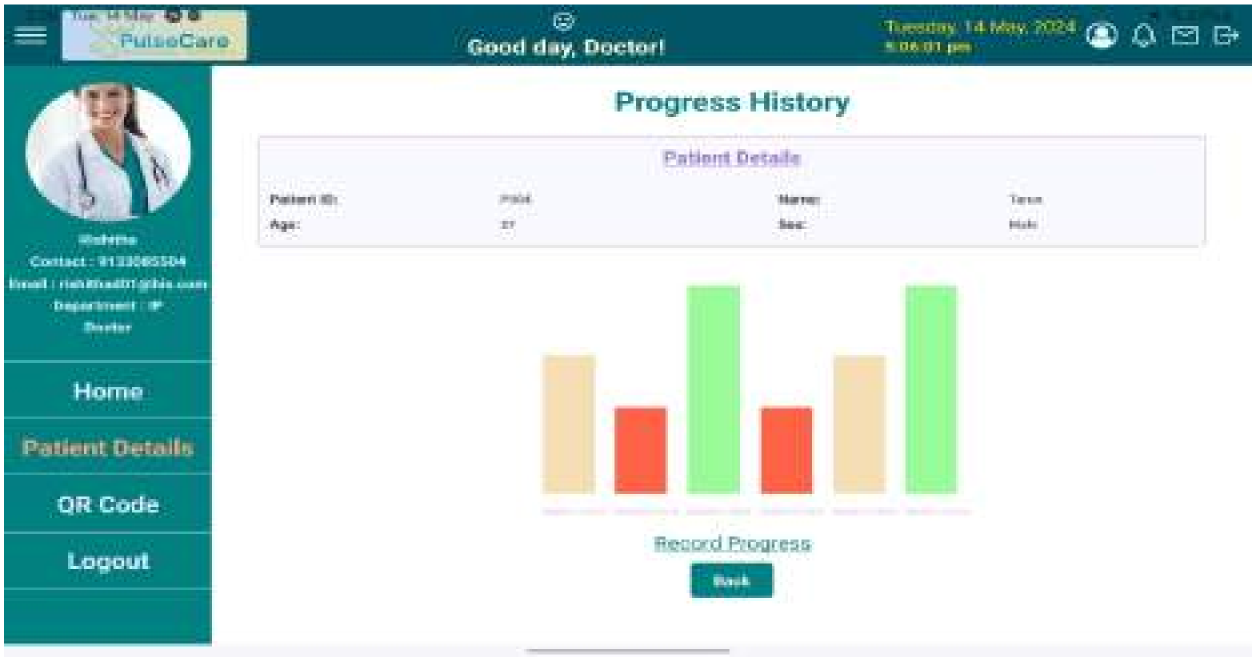
Test result added successfully for the patientId :P001 for Blood Test

Back

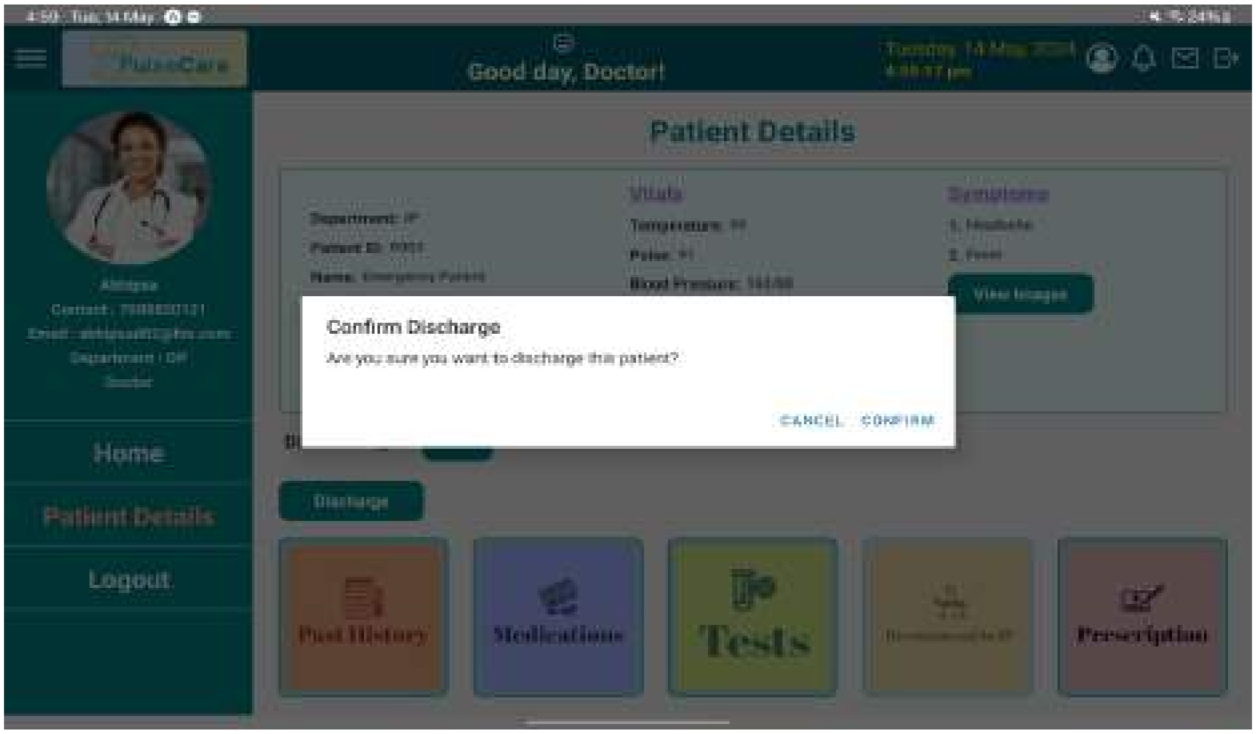
11. Record Progress (IP doctor)



12. Progress History (IP doctor)



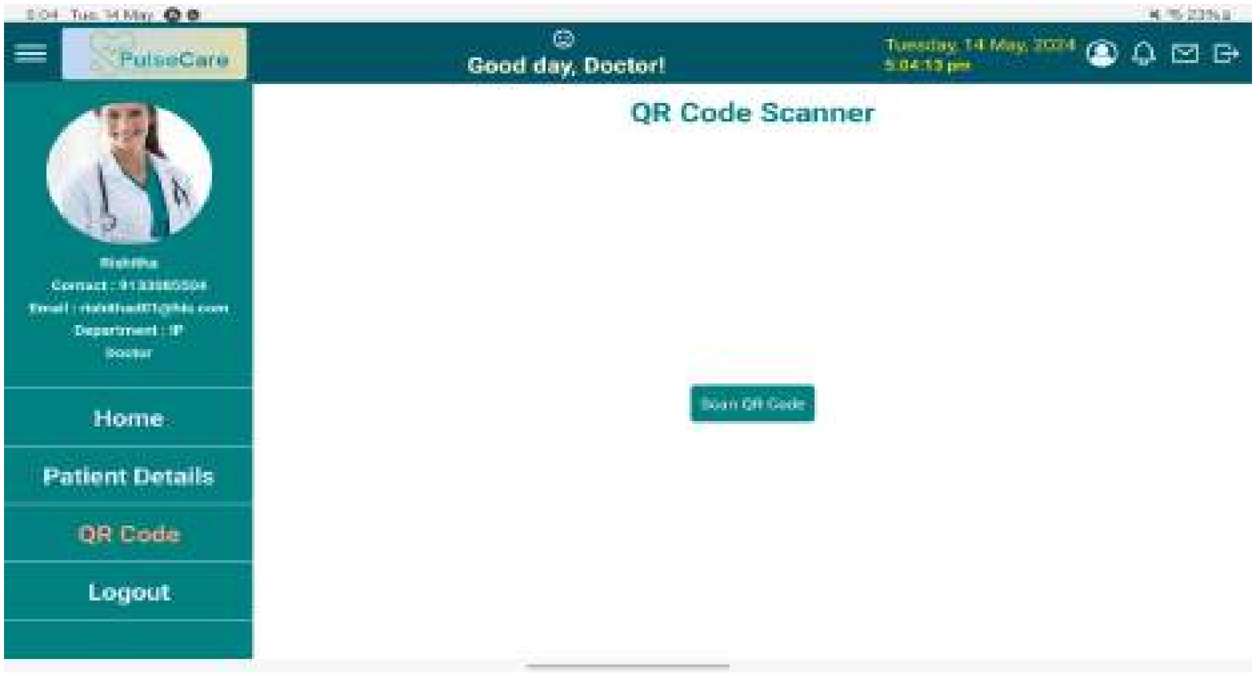
13. Discharge Patient



Department: IP
Name: Emergency Patient
Temperature:
Pulse:
Blood Pressure:
1. Headache
2. Fever

Confirm Discharge
Are you sure you want to discharge this patient?
CANCEL CONFIRM

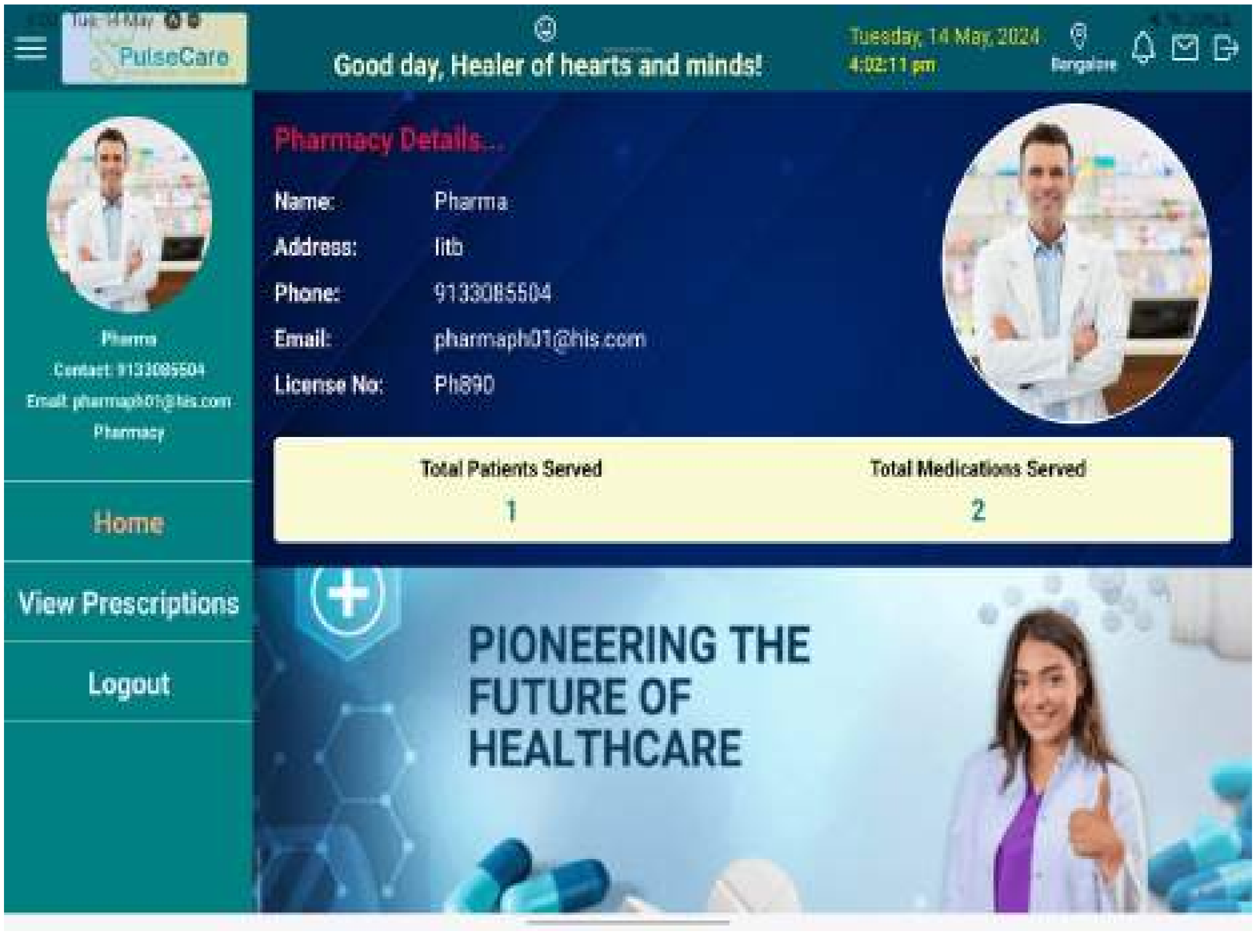
14. QR Scanner (IP doctor)



Department : IP
Doctor

5. PHARMACY

1. Pharmacy Dashboard



PulseCare

Good day, Healer of hearts and minds!

Tuesday, 14 May, 2024
4:02:11 pm

Bangalore

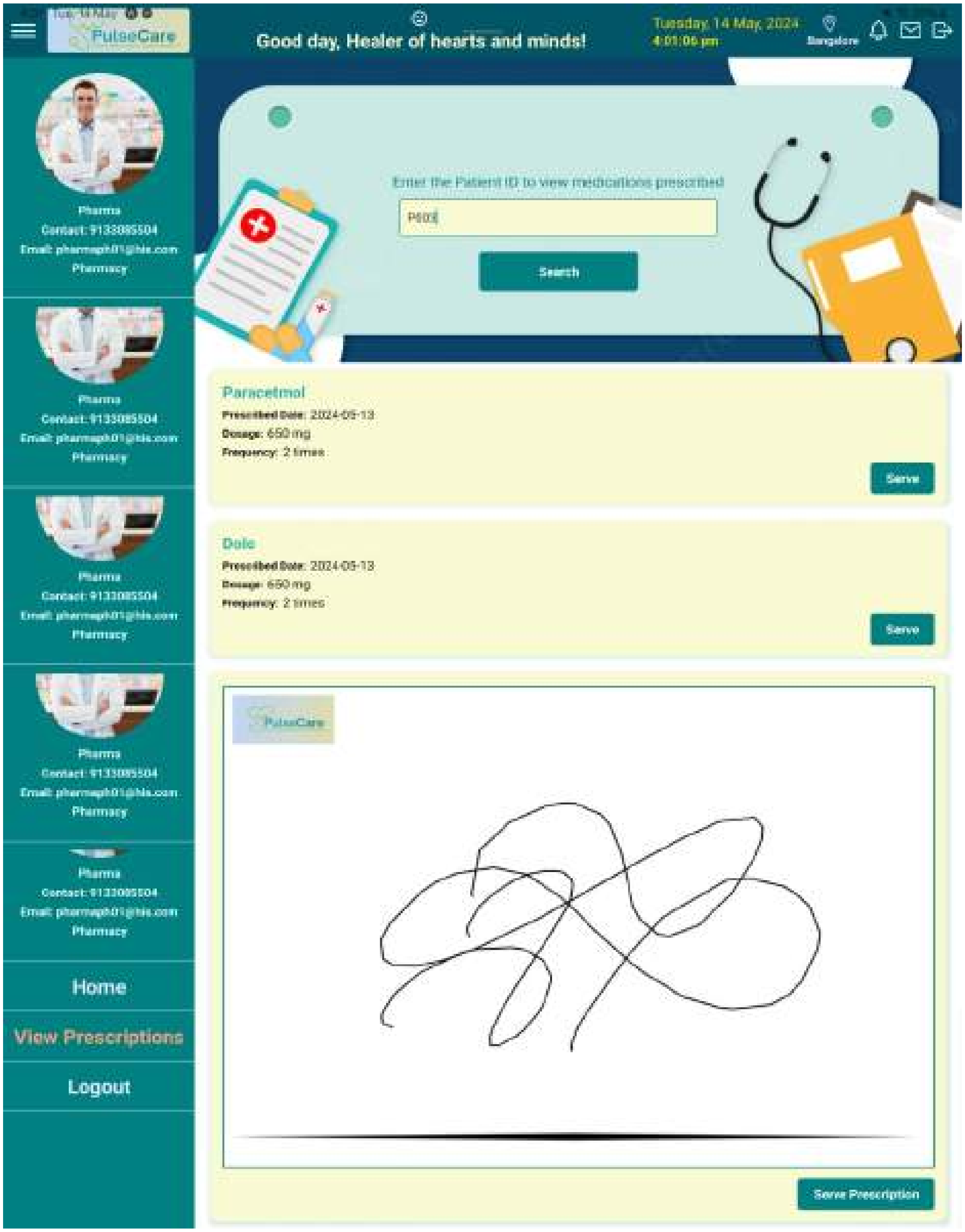
Pharma
Contact: 9133085504
Email: pharmaph01@his.com
Pharmacy

Pharmacy Details...

Name:	Pharma
Address:	Iitb
Phone:	9133085504
Email:	pharmaph01@his.com
License No:	Ph890

Total Patients Served	Total Medications Served
1	2

2. Pharmacy View Medications



PulseCare

Good day, Healer of hearts and minds!

Tuesday, 14 May, 2024
4:01:06 pm

Bangalore

Pharma
Contact: 9133085504
Email: pharmaph01@his.com
Pharmacy

Home

View Prescriptions

Logout

Enter the Patient ID to view medications prescribed

P003

Search

Paracetmol
Prescribed Date: 2024-05-13
Dosage: 650 mg
Frequency: 2 times

Serve

Dolo
Prescribed Date: 2024-05-13
Dosage: 650 mg
Frequency: 2 times

Serve

PulseCare

Serve Prescription

Figma Design(Front-End):

1. **ADMIN**

a. Design:
https://www.figma.com/file/Ffgl3ydMpUbkqf9IWlhuFl/Admin-Module?type=design&node-id=17-84&mode=design&t=lzQgDyAy5MwuzTBR-0

b. Prototype:
https://www.figma.com/proto/Ffgl3ydMpUbkqf9IWlhuFl/Admin-Module?type=design&node-id=17-46&t=lzQgDyAy5MwuzTBR-0&scaling=scale-down&page-id=0%3A1&starting-point-node-id=17%3A46

2. **IP Doctor**

a. Design:
https://www.figma.com/file/xACBHHZgVHV3E1S5hEt3V7/IP-Doctor?type=design&node-id=0-1&mode=design&t=6hyasGehnEUdWkbz-0

b. Prototype:
https://www.figma.com/proto/xACBHHZgVHV3E1S5hEt3V7/IP-Doctor?type=design&node-id=1-274&t=6hyasGehnEUdWkbz-0&scaling=scale-down&page-id=0%3A1

3. **OP Doctor**

a. Design:
https://www.figma.com/file/w2zMVUSJRwWVEwoEAOuNhe/OP-Doctor?type=design&node-id=0-1&mode=design&t=ASEDCWtVu2PDiU8G-0

b. Prototype:
https://www.figma.com/proto/w2zMVUSJRwWVEwoEAOuNhe/OP-Doctor?type=design&node-id=128-194&t=ASEDCWtVu2PDiU8G-0&scaling=scale-down&page-id=0%3A1

4. **Nurse**

a. Design:
https://www.figma.com/file/NekyNlZaaTeTWvX08pMFEr/Nurse-Module?type=design&node-id=0%3A1&mode=design&t=PgpEuVqneosvjeo5-1

b. Prototype:
https://www.figma.com/proto/NekyNlZaaTeTWvX08pMFEr/Nurse-Module?type=design node-id=30-2&t=8WbT4pbuEmxjngfv-1&scaling=scale-down&page-id=0%3A1&starting-point-node-id=30%3A2&show-proto-sidebar=1&mode=design

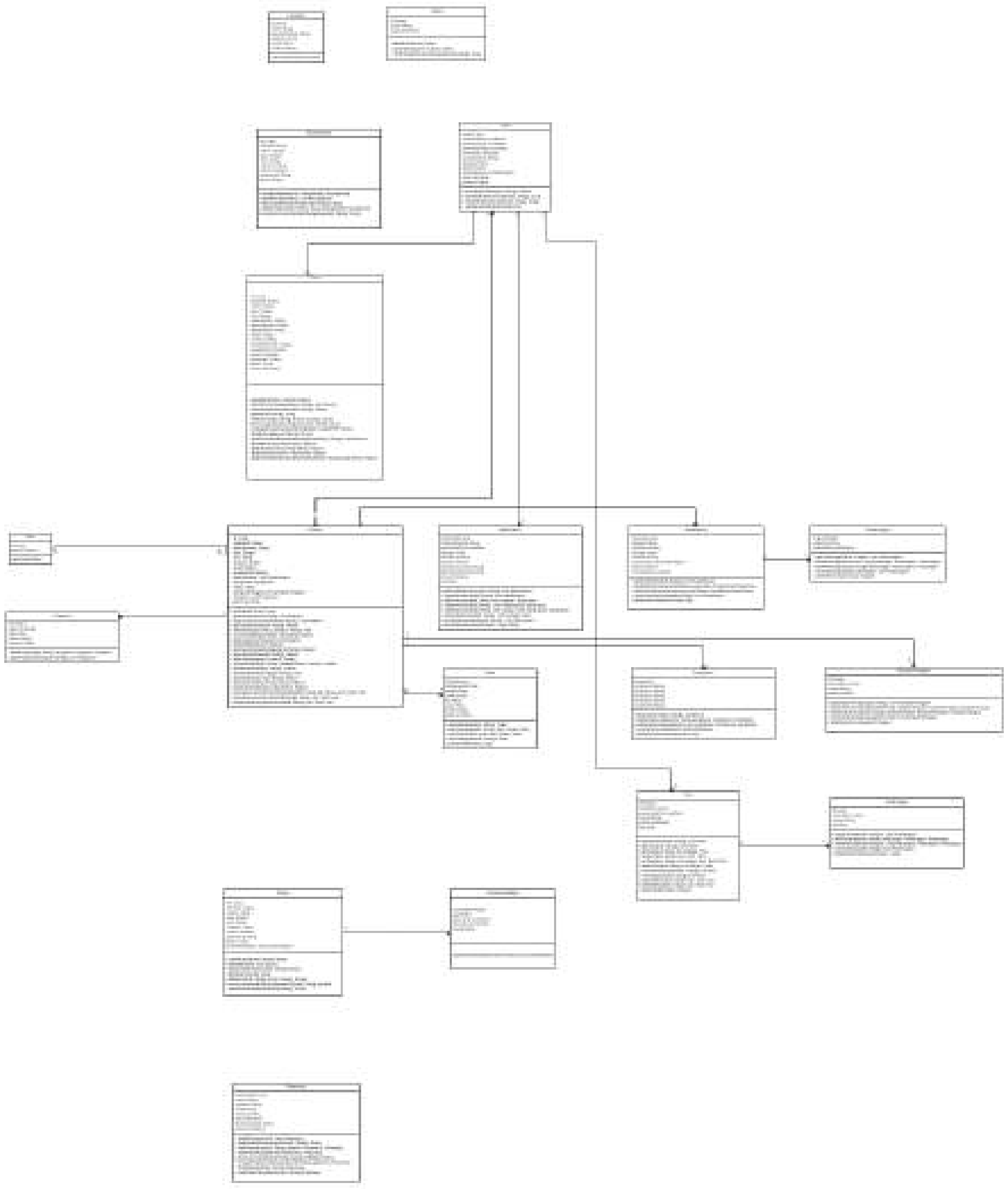
5. **<u>Receptionist</u>**
    a. Design:
    https://www.figma.com/file/WZ8QUxu9wqstokztKOIhqI/Receptionist?type=design&node-id=0%3A1&mode=design&t=cCdL1cSmr770pe0j-1

    b. Prototype:
    https://www.figma.com/proto/WZ8QUxu9wqstokztKOIhqI/Receptionist?type=design&node-id=11-2&t=tCm4GifQ1HxSuo3u-1&scaling=scale-down&page-id=0%3A1&starting-point-node-id=11%3A2&mode=design

6. **<u>Pharmacy</u>**
    a. Design:
    https://www.figma.com/file/S7W13ICs8fTNjAPl278NQK/Pharmacy?type=design&node-id=0-1&mode=design&t=mMfKNaRjUlbg3Q7Q-0

    b. Prototype:
    https://www.figma.com/proto/S7W13ICs8fTNjAPl278NQK/Pharmacy?type=design&node-id=33-8&t=mMfKNaRjUlbg3Q7Q-0&scaling=scale-down&page-id=0%3A1&starting-point-node-id=33%3A8

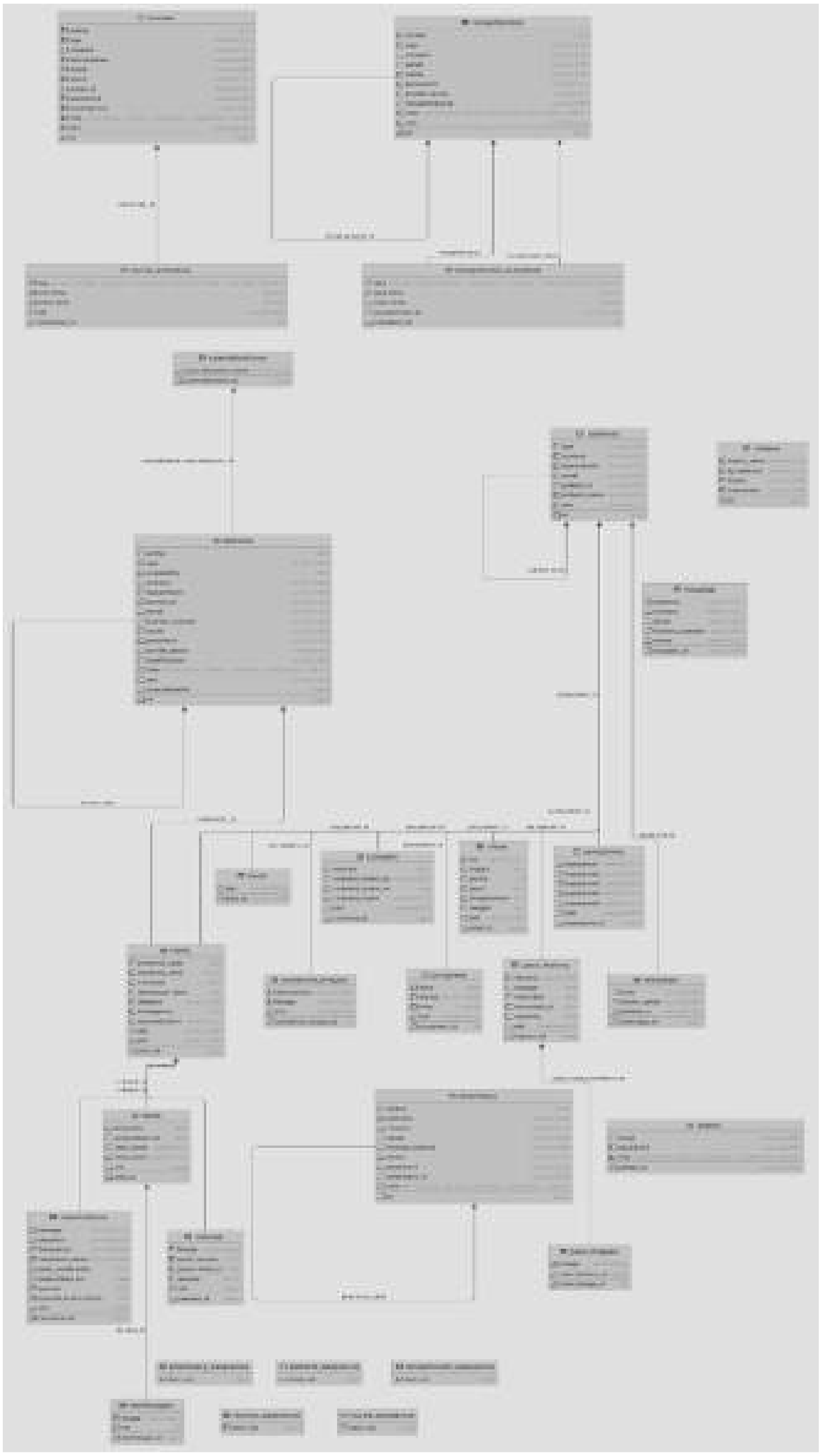
UML Class Diagram:



UML Class Diagram HAD.pdf

Please refer to the attached document.

Database Design:



Security:

1)JWT (JSON Web Tokens) Implementation:

JWTs are used to securely transmit information between parties as a JSON object, allowing us to verify the token's authenticity with a secret key. In our code, the JwtTokenProvider class is responsible for token generation, validation, and management:

1. **Token Generation**: Upon authentication, a token is generated using user details such as username(email) and expiration time. This token is signed using a secret key derived from a configured secret (jwtSecret).
2. **Token Validation**: When a request is received, the token is validated for integrity and expiration. This is handled by extracting claims such as username and comparing it against the user details stored in the system, ensuring that the token is both valid and corresponds to a legitimate user.
3. **Database Interaction**: Generated tokens are stored in a database with their expiration times, linking them to specific users. This allows additional validation layers, such as checking if a token is still stored in the database when accessed, providing a way to handle token revocation.

2)Role-Based Authentication:

Role-based authentication is implemented via the JwtAuthenticationFilter, extending OncePerRequestFilter to ensure it executes once per request. It performs the following operations:

1. **Token Extraction**: It first checks for a bearer token in the Authorization header of incoming requests.
2. **User Identification**: If a valid token is found, it extracts the username and validates the token by comparing it with the details fetched from the database through the loadUserByUsername method of a user details service.
3. **Role Verification**: It ensures that the token not only belongs to a valid user but also to a user with the correct role authorized to access the requested resource. This is achieved by matching user roles against the required roles for the resource being accessed.

3)Enhanced Login Security with OTP Verification:

This enhanced login process incorporates an OTP (One-Time Password) step to verify the user's identity through their registered contact number before granting access to the system.

1. **Email Entry**:
   - The user enters their email address on the login page.
2. **Extract Contact Information**:
   - The system retrieves the user's contact number associated with the entered email from the database.
3. **Send OTP**:
   - An OTP is generated and sent to the retrieved contact number.
   - This step ensures that the person attempting to log in has access to the contact number tied to the user's account.
4. **OTP Verification**:
   - The user is prompted to a new screen to enter the OTP they received on their contact number.
   - If the entered OTP matches the one sent, the login process proceeds; otherwise, access is denied.
5. **Access Granted**:
   - Upon successful OTP verification, the user is granted access to their account, ensuring that the login is secured through two-factor authentication.

4)AES Encryption For Secure Transmission Of Login Credentials:

To secure user credentials during transmission over the internet, AES (Advanced Encryption Standard) encryption can be employed at the frontend, and the corresponding decryption occurs at the backend. This method ensures that sensitive data like email addresses and passwords are protected from eavesdropping and man-in-the-middle attacks.

1. **User Inputs Credentials**:
   - The user enters their email and password into the login form on the frontend application.

2. **Encrypt Credentials**:
    - Before sending these credentials over the network, the frontend encrypts the data using AES encryption with a secure key.
    - The key for encryption and decryption is pre-shared between the frontend and the backend or derived through a secure key exchange mechanism.
3. **Transmission**:
    - The encrypted credentials are transmitted over the internet to the backend server.
4. **Decrypt Credentials**:
    - Upon receiving the encrypted data, the backend uses the AES key to decrypt the credentials.
    - The backend must ensure secure storage and handling of the decryption key to prevent unauthorized access.
5. **Verify Credentials**:
    - The backend then verifies the decrypted email and password against the stored credentials (which should also be securely hashed and stored in the database).
6. **Grant/Deny Access**:
    - If the credentials are verified successfully, the user is granted access. If not, access is denied.

5)Session Management :

1)IP Matching:

IP matching is a security measure to ensure that the user's session is being accessed from the same device or location as when the token was issued. This is crucial for preventing token misuse if stolen.

1.User Logs In :

- User logs into the system.
- System captures and records the user's IP address at the time of login.

2. Token and IP Storage:
    - A JWT token is generated and stored in the database.

    - The user's IP address is also stored in the database, linked to the token.
3. Request Validation:
    - Each request includes the JWT token and is made from a source IP.
    - The system retrieves the IP associated with the token from the database.
4. IP Match Verification:
    - If the source IP matches the IP stored in the database, the request proceeds.
    - If there's a mismatch, indicating potential token theft or misuse, access is denied, token gets invalidated and deleted from database and any user cannot make any further request using that token and a security alert is triggered.

<u>2)Concurrent Login Handling:</u>

1. **User Login Attempt**:
    - When a user attempts to log in, the system first checks if there is an existing active token for that username in the database.
2. **Check Existing Token**:
    - If an active token is found in the database for the username, the login request is denied.
    - The system responds with a message indicating that a session is already active, preventing concurrent logins.
3. **Successful Login**:
    - If no active token exists for the username, the system proceeds with the login process.
    - A new token is generated, recorded in the database along with the user's current IP address, and provided to the user.

<u>6) Logout Automatically After 15 Minutes of Inactivity Of User:</u>

To enhance the security of our application, we have implemented an automatic logout feature that triggers after 15 minutes of user inactivity. This feature is crucial for protecting user data, especially in scenarios where a user might leave their device unattended while logged into the application.

Mechanism:

- **Inactivity Detection:** The application utilizes the PanResponder API, which is part of the React Native framework, to detect user interactions such as taps, swipes, and other gestures. The PanResponder listens for touch events across the application to determine when the last user activity occurred.
- **Timer Management:** A timer is set to track the inactivity period. Each time the PanResponder detects user activity, the timer is reset to 15 minutes. If the timer expires without any further user interaction, the automatic logout process is triggered.
- **Logout Process:** Upon detecting the expiration of the inactivity timer, the application automatically logs the user out. This process involves clearing the user's session and any sensitive data cached on the device, followed by redirecting the user to the login screen.

Privacy Measures:

1)AES Encryption of Private and Sensitive Information In Database:

1. **AES Encryption**:
   a. AES (Advanced Encryption Standard) provides strong encryption and is widely supported. An appropriate key length (128 bits) is chosen based on our security requirements and regulatory compliance.
2. **Data Fields to Encrypt**:
   a. Identified which fields contain sensitive information that must be encrypted, such as personal identification numbers, medical records, contact details, etc.
3. **Method:**

The EncryptionUtil class in the Java application provides a utility for securing sensitive information using AES encryption with CBC mode and PKCS5 padding. It is designed to handle both encryption and decryption processes to protect data integrity and confidentiality.

Key Components:

- **Key and Initialization Vector (IV):** The class uses a predefined symmetric key (key) and an initialization vector (initVector), both of which are crucial for the AES algorithm's operation. The key and IV are hardcoded, which is not recommended for production environments due to security concerns but is sufficient for demonstration purposes.
- **Algorithm Specification:** It specifies the use of AES encryption in CBC mode with PKCS5 padding (AES/CBC/PKCS5PADDING). This mode of operation requires an IV for the encryption process, adding an additional layer of randomness and security.
- **Encryption Method:** The encrypt method takes a plaintext string as input, converts it into bytes, and encrypts it using AES with the specified key and IV. The encrypted data is then encoded into a Base64 string to ensure safe transmission or storage of the binary data.
- **Decryption Method:** Conversely, the decrypt method takes an encrypted Base64 string, decodes it into bytes, and decrypts it using the same key, IV, and algorithm configuration. It returns the original plaintext if decryption is successful.

2)Patient Consent Management in HealthCare:

In the modern healthcare setting, the management of patient consent is a critical component that ensures the privacy and autonomy of patients are respected. Our system has implemented a robust consent management process that aligns with these priorities, particularly when patients come for appointments.

Consent Acquisition Process: When a patient arrives for a doctor's appointment, the receptionist initiates the consent process. This involves the patient reading through specific terms and conditions that outline how their medical and personal data will be handled. To affirm their understanding and agreement, the patient verifies their consent through an OTP (One-Time Password) sent to their registered contact number. This verification process is crucial as it ensures that the consent is informed and explicitly given, aligning with legal and ethical standards.

Use and Validation of Consent: Once obtained, the consent is encapsulated within a token. This consent token has a defined validity period and is crucial for subsequent interactions within the healthcare system. Whenever a doctor or nurse needs to access the patient's details, they must first check the validity of this consent token. This step is essential to ensure that every access to the patient's data is authorized and compliant with the consent provided.

Revocation of Consent: Our system also provides patients with the capability to revoke their consent. This can be done in two distinct ways, depending on the patient's preference and the sensitivity of the data involved:

1. **Complete Removal of Data:** If a patient chooses to completely revoke consent, all medical data related to them will be removed from our systems. Furthermore, their personal details are anonymized, ensuring that the data cannot be traced back to them. This option is chosen when patients no longer wish their information to be used or stored in any form.
2. **Anonymization of Personal Details:** In cases where the patient decides to only make their personal details anonymous, the medical data is retained. This option is beneficial for ongoing research or statistical purposes where the medical information can still provide value, but the personal identity of the patient is concealed.

Security and Compliance: Our consent management process is designed to be secure, adhering to the latest standards in data protection and privacy laws. By implementing such a system, we ensure that all interactions with patient data are conducted transparently and ethically, thereby building trust and integrity in our healthcare services.

This dual approach to consent management allows patients to exercise their rights over their personal and health information actively, providing them with choices that respect their privacy while balancing the needs of the healthcare system to maintain essential medical records.

c)Automated Consent Management Implementation:

This system is designed to automatically update patient consent status to inactive one week after their last visit, thereby restricting access to their personal details by healthcare professionals unless renewed consent is obtained.

Implementation Details:

- **Scheduler Configuration:** Using Spring Boot's scheduling capabilities, we have configured a scheduled task that runs daily to assess the consent status of patient records.We employ the @Scheduled annotation to define the task, which uses a cron expression to perform daily checks on the consent validity. The task executes a database query to locate patient records due for consent status updates based on their last visit date.

Consent Update Mechanism: The scheduler identifies patients whose last visit occurred more than one week ago and automatically sets their consent status to false. This process is critical for managing the lifecycle of patient consent effectively.

- **Security and Compliance:** This automated process supports strict adherence to privacy laws and healthcare regulations, which mandate that patient data access must be contingent on active consent. By automatically updating consent statuses, we ensure that patient data is not accessed without explicit permission.

3)Automated Data Retention and Deletion Policy Implementation:

As part of our commitment to data privacy and regulatory compliance, we have implemented an automated system within our Spring Boot application to manage the retention and deletion of patient data. This system ensures that patient details are automatically removed from our database three years after their last recorded visit, aligning with our data retention policy.

Implementation Details:

- **Scheduler Configuration:** Utilizing Spring Boot's scheduling capabilities, we configured a scheduled task that periodically scans the database for patient records where the last visit date exceeds three years.
    - We use the @Scheduled annotation in Spring Boot to define the cron expression that triggers this task daily. This scheduler is responsible for initiating the deletion process.The task executes a query to identify patient records eligible for deletion based on the date criteria.
- **Deletion Mechanism:** Once identified, these records are securely and permanently deleted from the database. This automated process reduces the risk of human error and ensures timely adherence to our data retention policy.
- **Security and Compliance:** This feature supports compliance with healthcare regulations that mandate the secure handling and timely disposal of patient

information, thus protecting patient privacy and minimizing the risk of data breaches.

Future Scope:

1. Real-Time Communication Between Pharmacy and Doctors Using WebSockets

Objective: Establish a real-time communication channel between the pharmacy and doctors within the healthcare application using WebSocket technology. This enhancement aims to improve collaboration and responsiveness when dealing with medication availability issues.

Description: When a prescribed medication is unavailable, the pharmacy staff can immediately notify the prescribing doctor through a real-time chat system. This system will leverage WebSocket technology to facilitate an open communication channel that remains active during user sessions, allowing instant messaging without the need for repeated polling.

2. Voice-to-Text and NLP Integration for Medication and Test Entries

Objective: Incorporate voice-to-text capabilities and Natural Language Processing (NLP) into the system to streamline the process of adding medications and test orders.

Description: This feature allows healthcare providers to dictate medication orders and test requests verbally, which are then converted into structured text using voice recognition technology. The NLP component will analyze the text to understand and verify the content, ensuring accuracy before submission.

Conclusion:

The healthcare application is specifically designed to optimize the workflows of busy medical facilities, greatly enhancing the efficiency of doctors, nurses, receptionists, and pharmacy staff by automating routine tasks and centralizing patient data. This centralized approach ensures that all modules are seamlessly interconnected, facilitating a smoother workflow that enables healthcare providers, especially doctors who are often pressed for time, to deliver higher quality care more efficiently.

The implementation of this system is poised to significantly improve patient outcomes by providing doctors with more accurate and timely access to patient data, enhancing communication across different departments, and enabling quicker response times in both routine and emergency care scenarios. This is particularly beneficial for busy doctors who need to make quick, informed decisions. The consent management feature also plays a crucial role in this system, ensuring adherence to privacy laws and ethical standards, thereby enhancing patient trust and security.

LINKS

DRIVE LINK FOR DEMO VIDEO

https://drive.google.com/drive/folders/1aN6RcrAnKwic-P8HabNnfoDtTA1CUnYU?usp=drive_link

GITHUB LINKS

Frontend : https://github.com/Team30HAD/HISFrontend.git

Backend : https://github.com/Team30HAD/HISBackend.git

PRIVACY POLICY SUMMARY

https://drive.google.com/file/d/1f8AEQo3JkTMFSWQCUy_F_0DmSvUdhGCF/view?usp=drive_link