Lab Report for Group Sekhmet

Hospital Management System

Srishty Gandhi(20CS30052), Yatindra Indoria (20CS30060)
Chirag Ghosh (20CS10020), Sarita Singh (20CS10053)
Shreya Agrawal (20CS10058)

DBMS LABORATORY
Department of Computer Science and Engineering

March 20, 2023

Hospital Management Lab Mini Project

Introduction

This report outlines the implementation of Flask, Flask Bcrypt, and JWT-based authentication for data security in hospital management system. The system registers patient, schedules appointments with doctors, maintains patient information about diagnostic tests and treatments administered, maintains information about doctors/healthcare professionals, and stores admit/discharge information about the patient. The intended users of the system are front desk operators, data entry operators, doctors and database administrators. Here is the link for our working project.

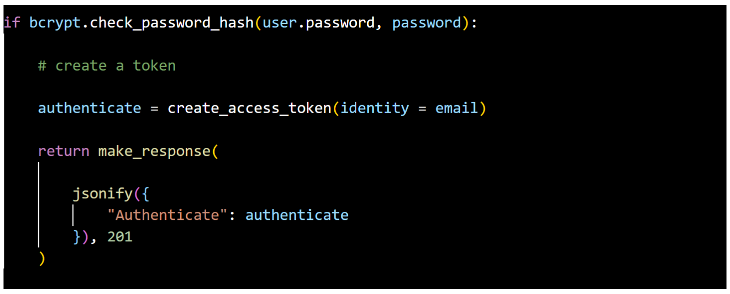
Backend

1. **Backend written in Flask**
   The backend of the hospital management system has been implemented using Flask, a popular Python-based web framework. Flask provides a lightweight and flexible web application framework, making it an ideal choice for this project.
   Flask has been used to create the endpoints for registering patients, scheduling appointments with doctors, maintaining patient information about diagnostic tests and treatments administered, maintaining information about doctors/healthcare professionals, and storing admit/discharge information about the patients.

```
@app.route('/user', methods=['POST', 'GET'])
def user():

@app.route('/user/login', methods=['POST', 'GET'])
def user_login():

@app.route('/patient', methods=['POST', 'GET'])
def patient():

@app.route('/patient/<int:ssn>')
def patient_ssn(ssn):

@app.route('/patient/<int:ssn>/appointment', methods=['POST', 'GET'])
def patient_ssn_appointment(ssn):

@app.route('/patient/<int:ssn>/test', methods=['POST', 'GET'])
def patient_ssn_test(ssn):

@app.route('/physician', methods=['POST', 'GET'])
def physician():
```

```
@app.route('/physician/<int:id>')
def physician_id(id):

@app.route('/physician/engagements')
def physician_engagements()

@app.route('/procedure')
def procedure():

@app.route('/procedure/<int:id>', methods=['POST'])
def procedure_id(id)

@app.route('/medication')
def medication():

@app.route('/appointment/<int:id>', methods=['PATCH', 'GET'])
def appointment_id(id):

@app.route('/notify')
def notify():

@app.route('/stats')
def stats():
```

2. **Flask-Bcrypt used for password hashing**
   Flask-Bcrypt has been used for password hashing, which is an essential step in data security. Flask-Bcrypt is a library that provides hashing utilities for Flask, by instantiating a 'Bcrypt' object, making it easy to generate and validate secure password hashes.



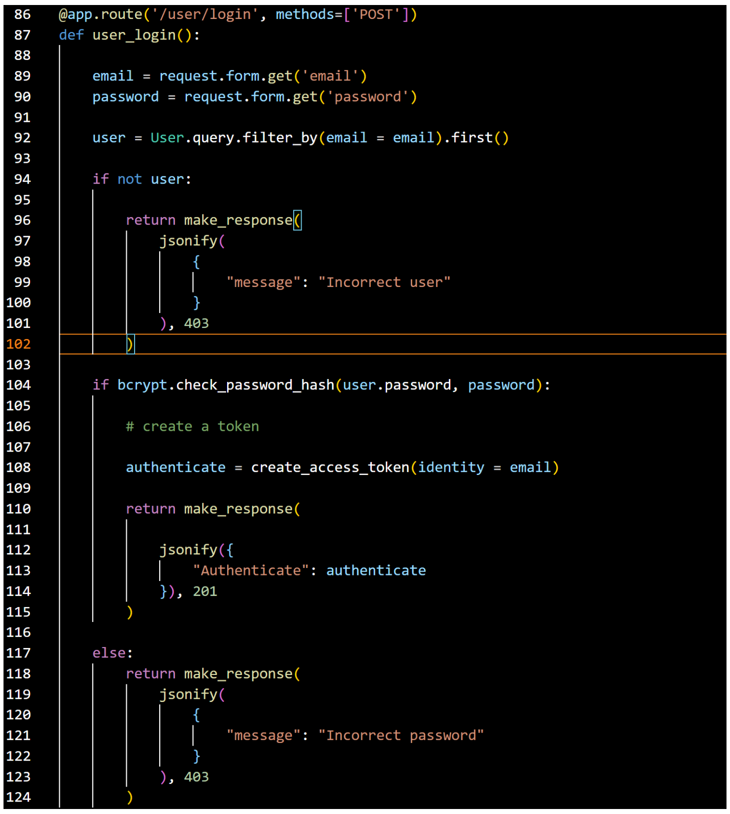
```
if bcrypt.check_password_hash(user.password, password):

    # create a token

    authenticate = create_access_token(identity = email)

    return make_response(
        jsonify({
            "Authenticate": authenticate
        }), 201
    )
```

Figure 1: Password Hashing using Bcrypt

   Passwords for all users including front desk operators, data entry operators, doctors, and database administrators are hashed before being stored in the database.

3. **JWT-based authentication implemented for data security**
   To further improve data security, JWT-based authentication has been implemented. JWT-based authentication allows users to authenticate themselves without sending

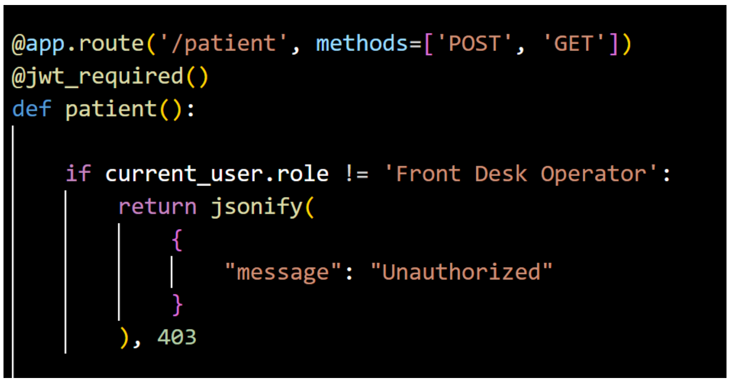
their passwords over the network, reducing the risk of password theft. JWTs are cryptographically signed tokens that can be used to authenticate users. Only authenticated users can perform certain operations such as scheduling appointments, entering patient data about tests and treatments and querying patient information



```
@app.route('/user/login', methods=['POST'])
def user_login():

    email = request.form.get('email')
    password = request.form.get('password')

    user = User.query.filter_by(email = email).first()

    if not user:

        return make_response(
            jsonify(
                {
                    "message": "Incorrect user"
                }
            ), 403
        )

    if bcrypt.check_password_hash(user.password, password):

        # create a token

        authenticate = create_access_token(identity = email)

        return make_response(

            jsonify({
                "Authenticate": authenticate
            }), 201
        )

    else:
        return make_response(
            jsonify(
                {
                    "message": "Incorrect password"
                }
            ), 403
        )
```

Figure 2: Role-based login using JWT

4. **Role-based authentication**
   Role-based authentication has been added to the frontend, allowing users to authenticate themselves based on their roles. This adds an additional layer of security to the application, ensuring that only authorized users can access sensitive data. Front desk operators can only register, admit, and discharge patients, while data entry operators can only enter patient data about tests and treatments. Doctors can query patient information, but cannot modify or delete it. Database administrators have the authority to add or delete users



```
@app.route('/patient', methods=['POST', 'GET'])
@jwt_required()
def patient():

    if current_user.role != 'Front Desk Operator':
        return jsonify(
            {
                "message": "Unauthorized"
            }
        ), 403
```

Figure 3: Role-based authentication for a route

5. **Changes made to database**
   The undergoes table in the database has been modified to include two new attributes - results and artifact. Results attribute is used to store the result of the procedure, while the artifact attribute is used to store images such as X-rays related to the procedure. This modification was made to provide a comprehensive record of the patient's diagnosis and treatment.

6. **Triggers implemented**
   A trigger has been added to automate the patient registration process. When a patient is added to the patient table, the trigger automatically looks up for an empty room from the room table and updates it as occupied. Along with that, it updates the stay table as well. This modification was made to ensure that the patient's stay in the hospital is recorded accurately, reducing the risk of human error and ensuring that the patient is assigned a room quickly.

Frontend

1. **Role based UI**
   We implemented different UI on the basis of roles like Doctor, Admin, Front Desk Operator and Data Entry Operator. We ensured that a person cannot access unauthorized data by simply changing the URLs. As the backend will return a '403: Unauthorized Error' in case of a user-role mismatch. The same is ensured by keeping the Role state of the 'current-user'.
   Here are the screenshots attached displaying the UI for different roles.

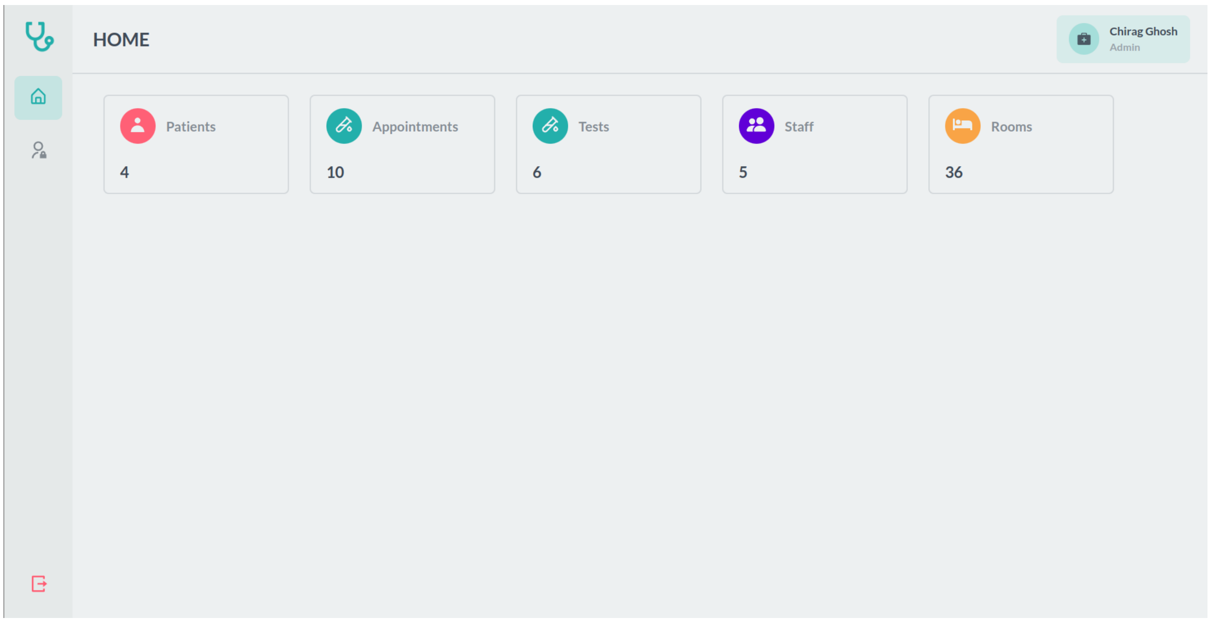
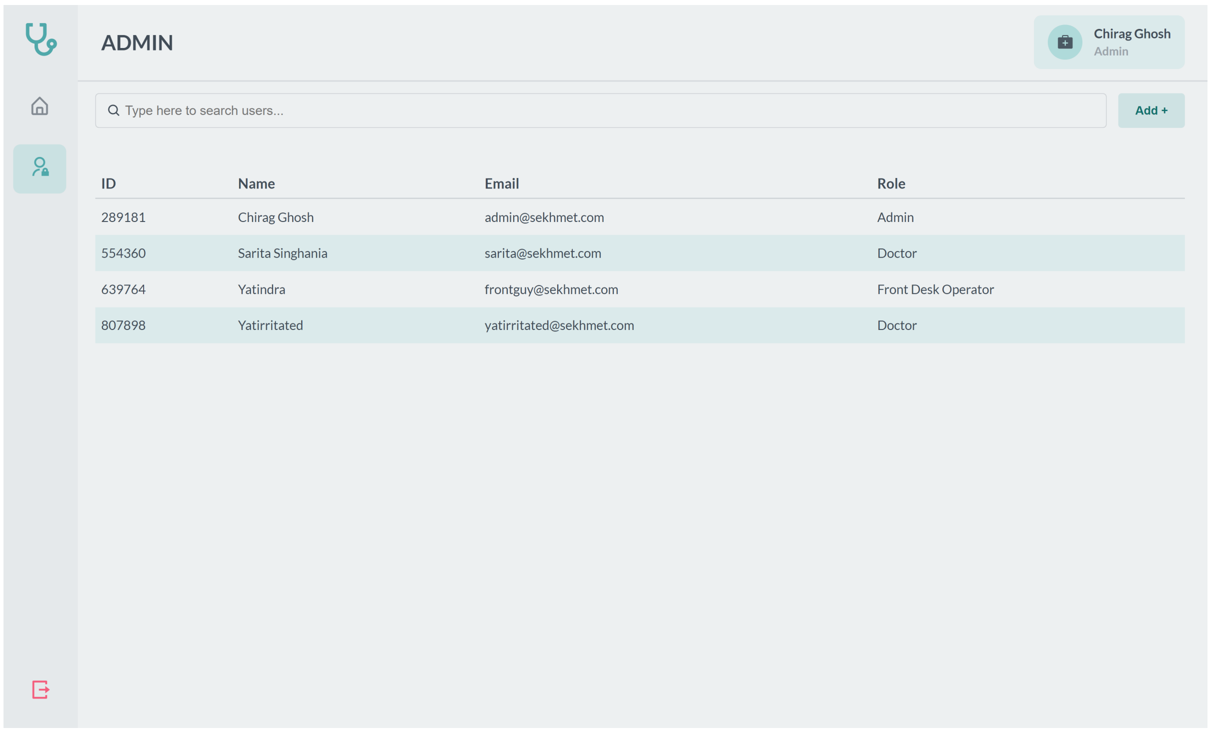
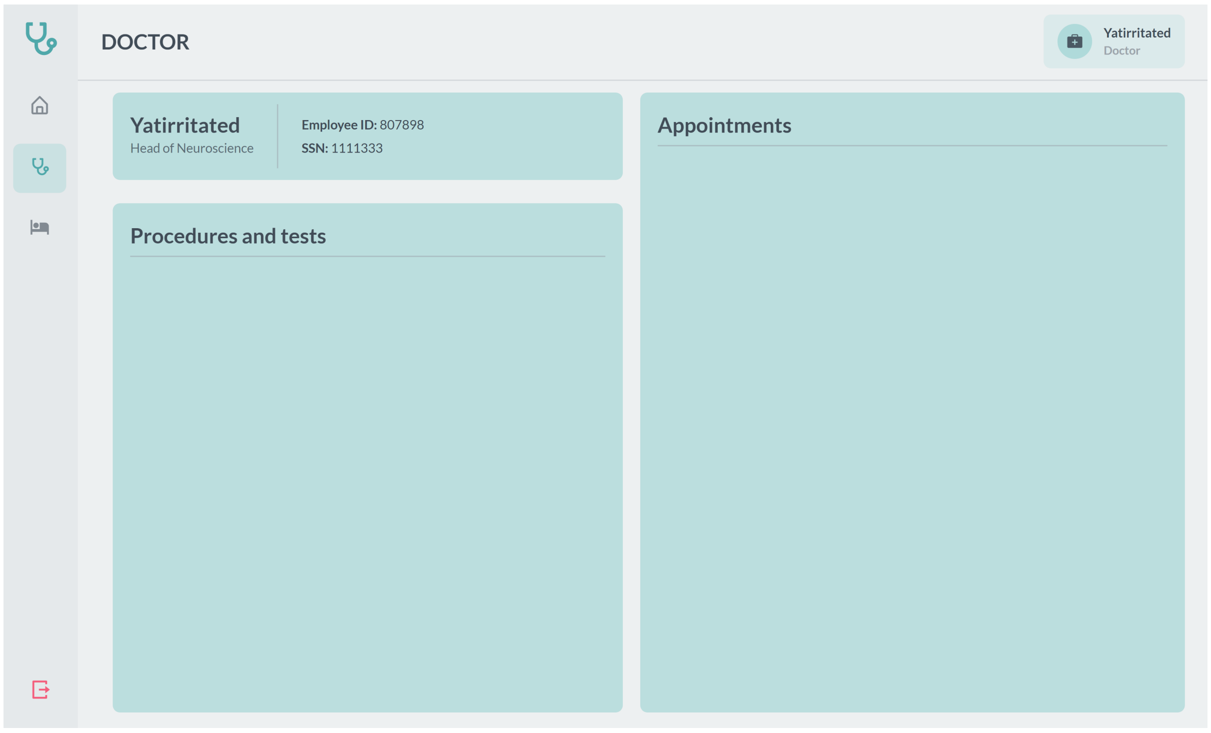


Figure 4: Common Landing page



ID	Name	Email	Role
289181	Chirag Ghosh	admin@sekhmet.com	Admin
554360	Sarita Singhania	sarita@sekhmet.com	Doctor
639764	Yatindra	frontguy@sekhmet.com	Front Desk Operator
807898	Yatirritated	yatirritated@sekhmet.com	Doctor

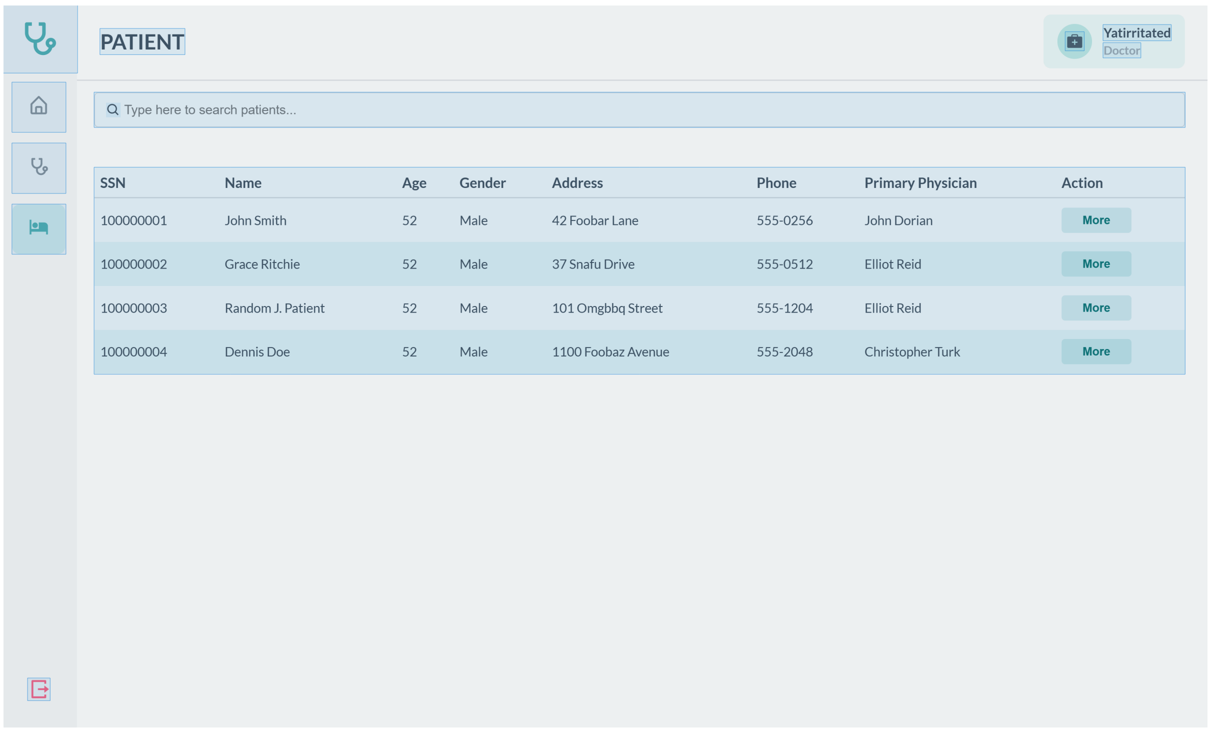
Figure 5: Admin Page



Yatirritated
Head of Neuroscience

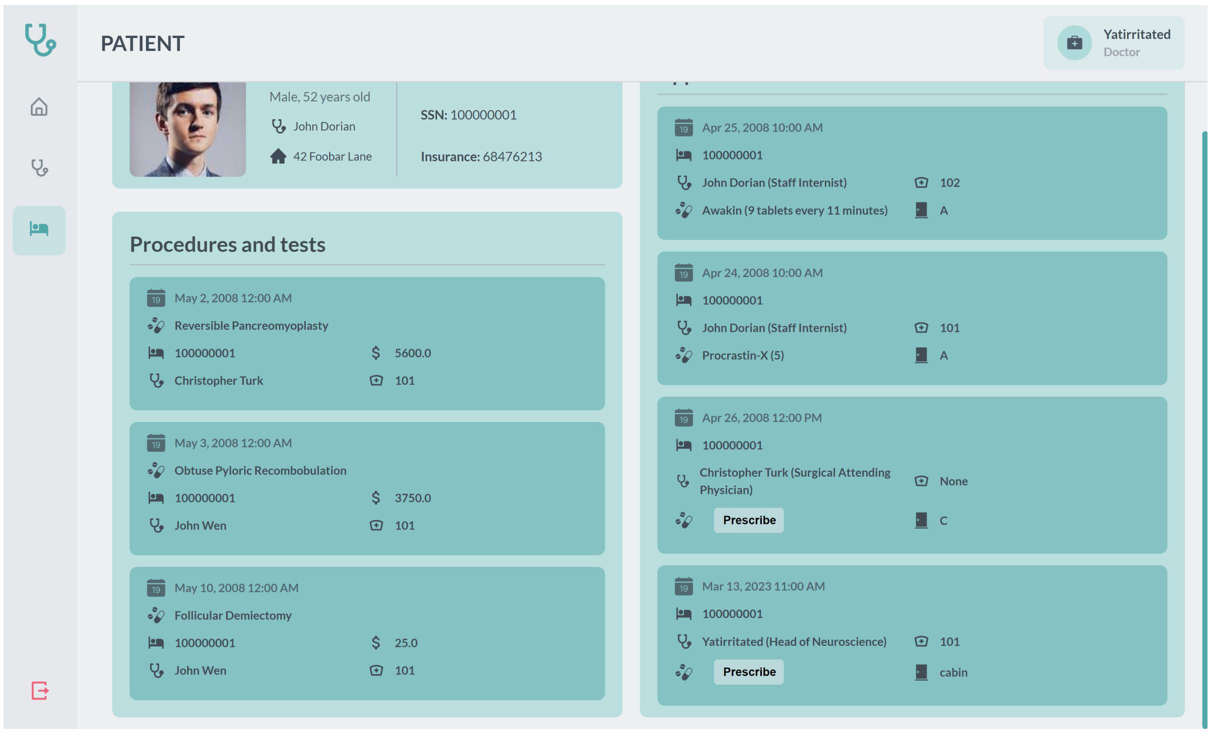
Employee ID: 807898
SSN: 1111333

Figure 6: New Doctor's Dashboard Page



SSN	Name	Age	Gender	Address	Phone	Primary Physician	Action
100000001	John Smith	52	Male	42 Foobar Lane	555-0256	John Dorian	More
100000002	Grace Ritchie	52	Male	37 Snafu Drive	555-0512	Elliot Reid	More
100000003	Random J. Patient	52	Male	101 Omgbbq Street	555-1204	Elliot Reid	More
100000004	Dennis Doe	52	Male	1100 Foobaz Avenue	555-2048	Christopher Turk	More

Figure 7: Patient Dashboard for Doctors Page


PATIENT

Yatirritated
Doctor

Male, 52 years old
John Dorian
42 Foobar Lane

SSN: 100000001
Insurance: 68476213

Procedures and tests

May 2, 2008 12:00 AM
Reversible Pancreomyoplasty
100000001 $ 5600.0
Christopher Turk 101

May 3, 2008 12:00 AM
Obtuse Pyloric Recombobulation
100000001 $ 3750.0
John Wen 101

May 10, 2008 12:00 AM
Follicular Demiectomy
100000001 $ 25.0
John Wen 101

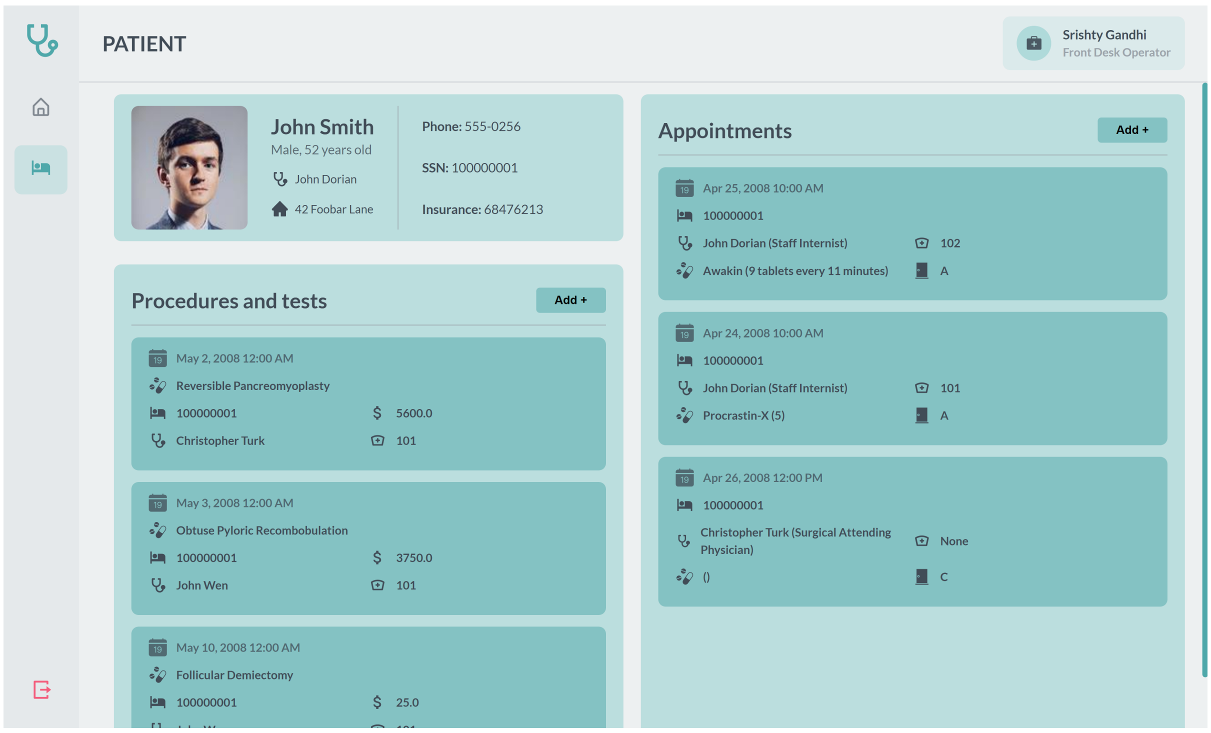
Apr 25, 2008 10:00 AM
100000001
John Dorian (Staff Internist) 102
Awakin (9 tablets every 11 minutes) A

Apr 24, 2008 10:00 AM
100000001
John Dorian (Staff Internist) 101
Procrastin-X (5) A

Apr 26, 2008 12:00 PM
100000001
Christopher Turk (Surgical Attending Physician) None
Prescribe C

Mar 13, 2023 11:00 AM
100000001
Yatirritated (Head of Neuroscience) 101
Prescribe cabin

Figure 8: Individual Patient as seen by Doctor


PATIENT

Srishty Gandhi
Front Desk Operator

John Smith
Male, 52 years old
John Dorian
42 Foobar Lane

Phone: 555-0256
SSN: 100000001
Insurance: 68476213

Procedures and tests Add +

May 2, 2008 12:00 AM
Reversible Pancreomyoplasty
100000001 $ 5600.0
Christopher Turk 101

May 3, 2008 12:00 AM
Obtuse Pyloric Recombobulation
100000001 $ 3750.0
John Wen 101

May 10, 2008 12:00 AM
Follicular Demiectomy
100000001 $ 25.0

Appointments Add +

Apr 25, 2008 10:00 AM
100000001
John Dorian (Staff Internist) 102
Awakin (9 tablets every 11 minutes) A

Apr 24, 2008 10:00 AM
100000001
John Dorian (Staff Internist) 101
Procrastin-X (5) A

Apr 26, 2008 12:00 PM
100000001
Christopher Turk (Surgical Attending Physician) None
() C

Figure 9: Individual Patient as seen by Front-Desk Operator

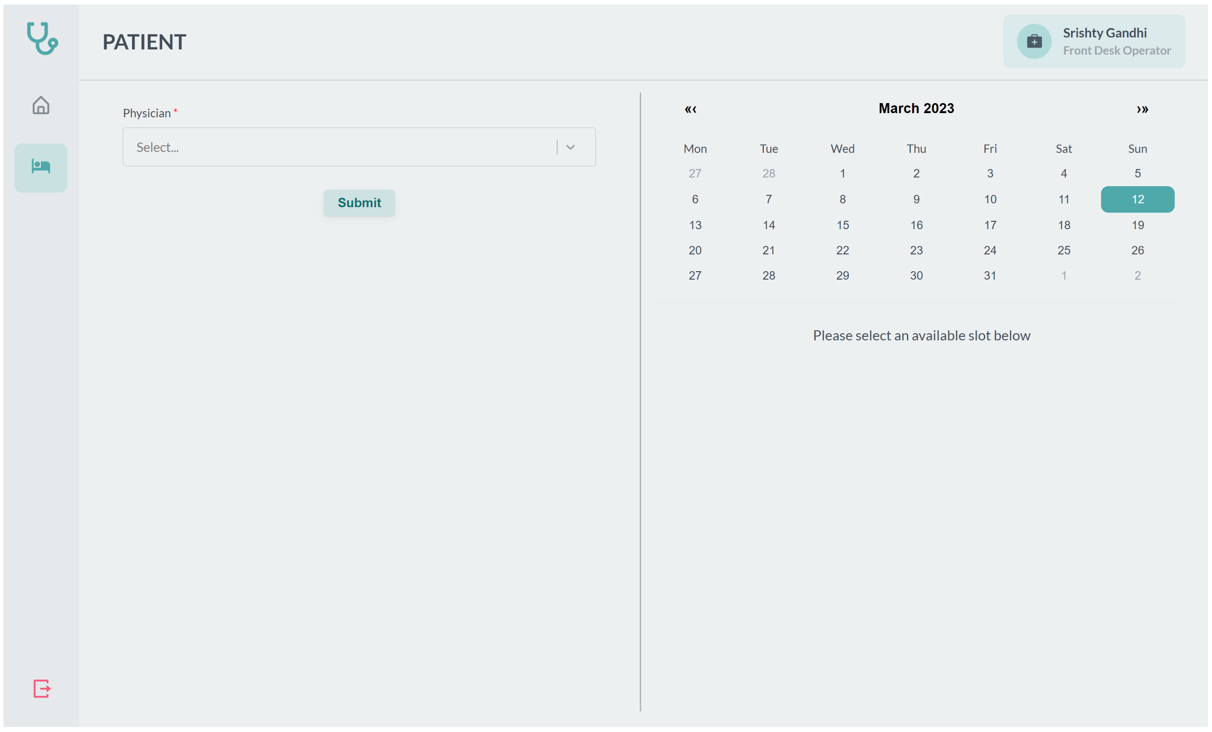


Figure 10: Appointment Scheduling by Front-Desk Operator

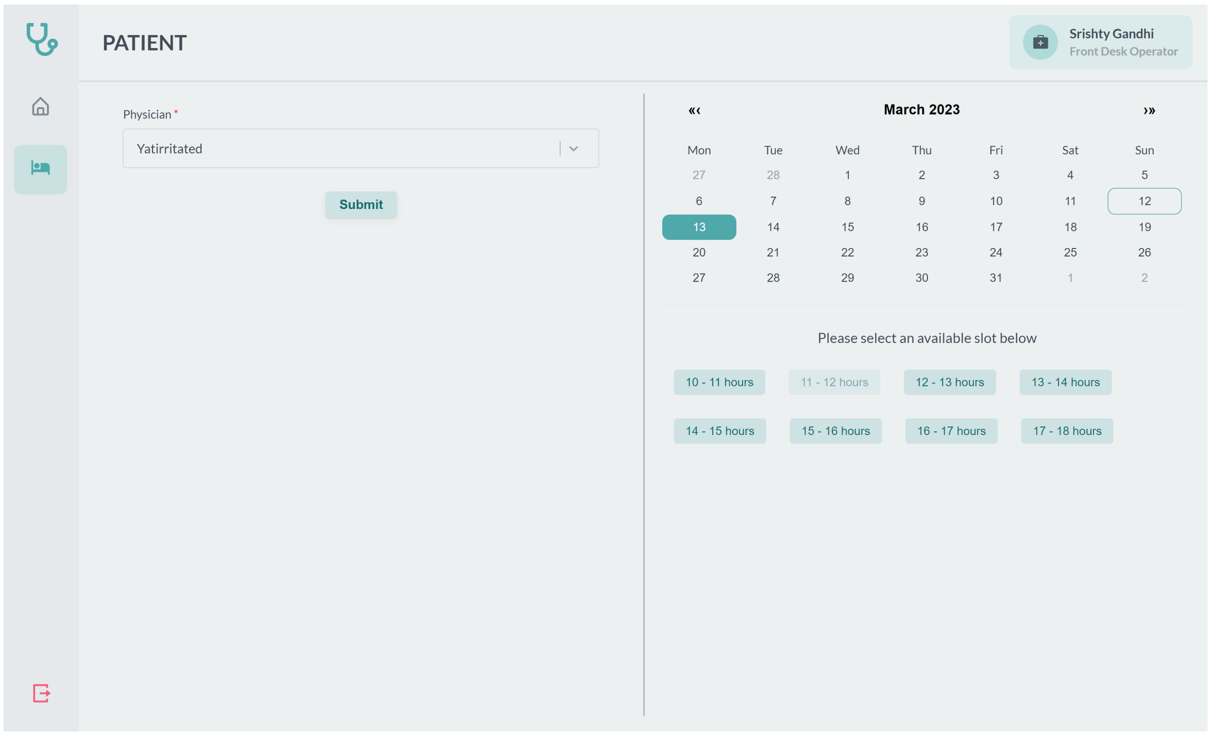
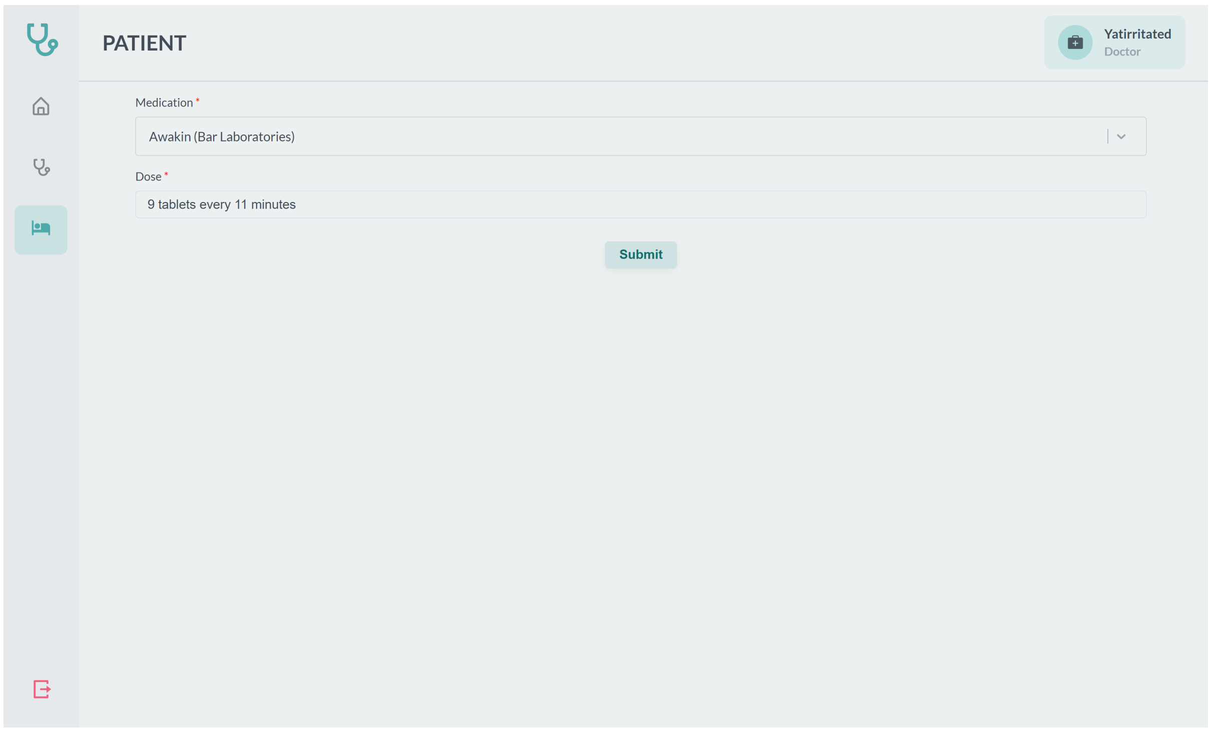


Figure 11: Appointment for a Doctor by Front-Desk Operator



Medication *
Awakin (Bar Laboratories)
Dose *
9 tablets every 11 minutes

Figure 12: Prescribes view for a Doctor

2. **Grid-Based Date Entry**
   In case of patient admission, appointment, and test scheduling, the 'Date' field is selected via a grid-UI. Hence no formatting, and logic errors can occur.

   Similarly for data consistency, if a 'Patient' or 'Date', which is not logically valid is selected, the system doesn't allow it. (For example, a patient, whose 'Stay' isn't ongoing, cannot have an appointment, or any appointment is unavailable or inconsistent (possibly a past time) then the system again doesn't allow it to happen.

3. **Selection via Drop-Downs**
   The fields which are important as Foreign Keys are made available via a dropdown to select. Hence making it impossible to violate the existing conditions. (For example, the medications for an appointment).

4. **In Case of Issues (Like Unavailable Room)**
   In case of issues that come apparent later during the process, an error is shown as an alert denoting the issue. (for example, a patient, when added, by logic is given a room automatically, in case none is available, we can acknowledge the same).