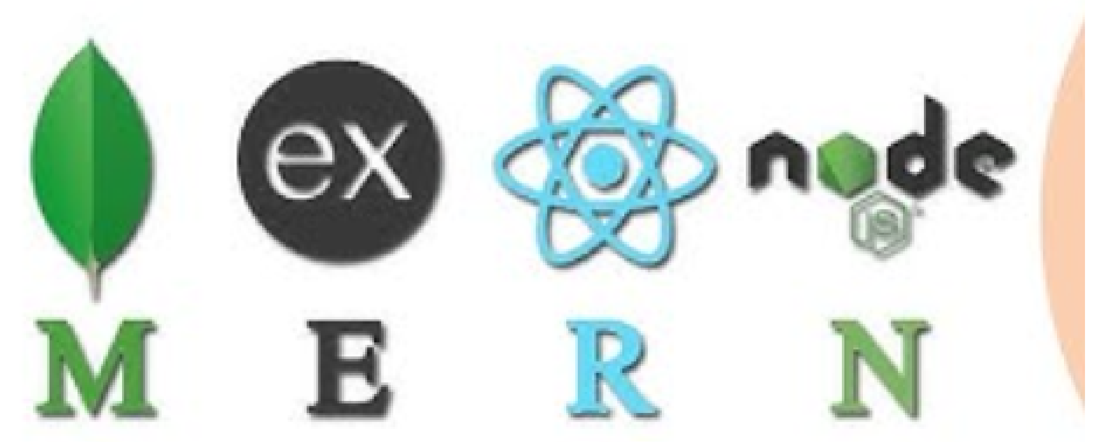
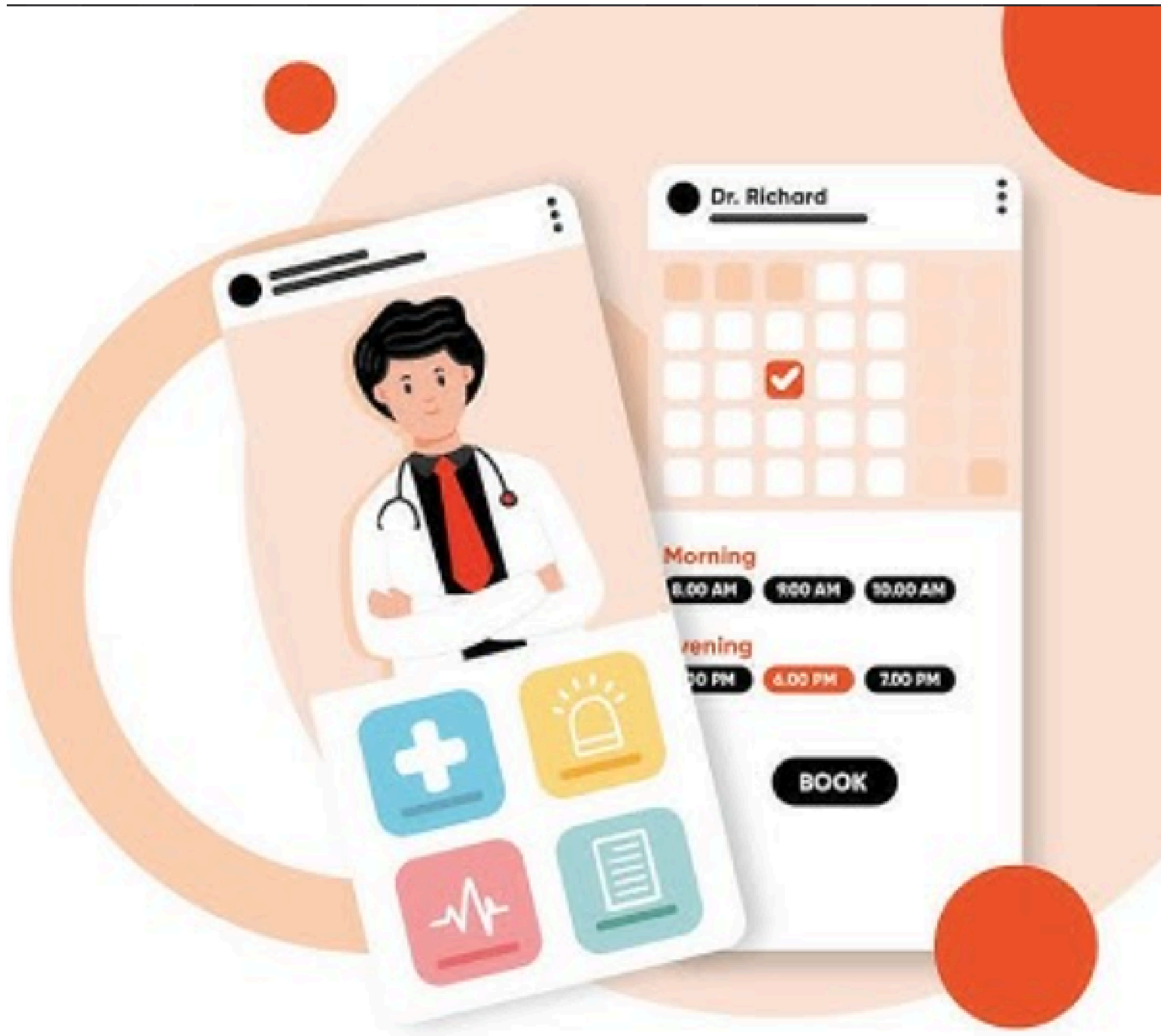
BOOK A DOCTOR USING MERN

By Farha Ansari
Ankur Jha
Priyanshu Sharma
Ryan Joseph

INTRODUCTION

Booking a doctor's appointment has never been easier. With our convenient online platform, you can quickly and effortlessly schedule your appointments from the comfort of your own home. No more waiting on hold or playing phone tag with busy receptionists. Our user-friendly interface allows you to browse through a wide range of doctors and healthcare providers, making it simple to find the perfect match for your needs. With our advanced booking system, you can say goodbye to the hassle of traditional appointment booking. Our platform offers real-time availability, allowing you to choose from a range of open slots that fit your schedule. Whether you prefer early morning, evening, or weekend appointments, we have options to accommodate your needs.





PROJECT GOALS

- Simplify Doctor Appointment Booking

  -Allow users to find and book available doctors quickly and easily.
- Role-Based Access

  -Implement different functionalities for patients, doctors, and optionally an admin.
- Real-Time Availability

  -Display doctor availability dynamically and prevent booking conflicts.
- Secure Authentication

  -Enable secure login and registration using JWT (JSON Web Token).
- User-Friendly Interface

-Design a responsive and intuitive UI for a smooth user experience.

TECH STACK USED

- MERN Stack Breakdown:

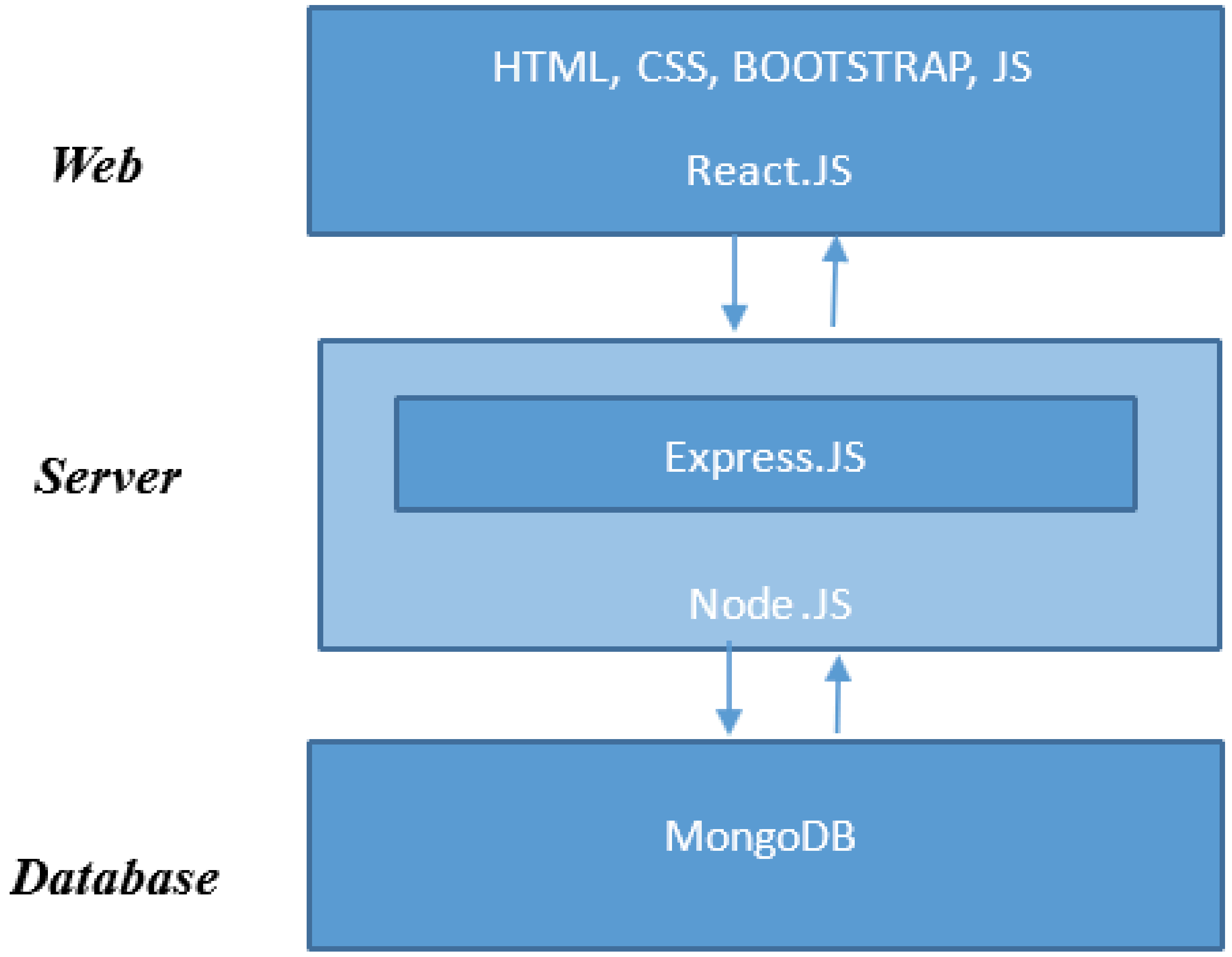
1. MongoDB – Database
2. Express.js – Backend framework
3. React.js – Frontend library
4. Node.js – Runtime environment

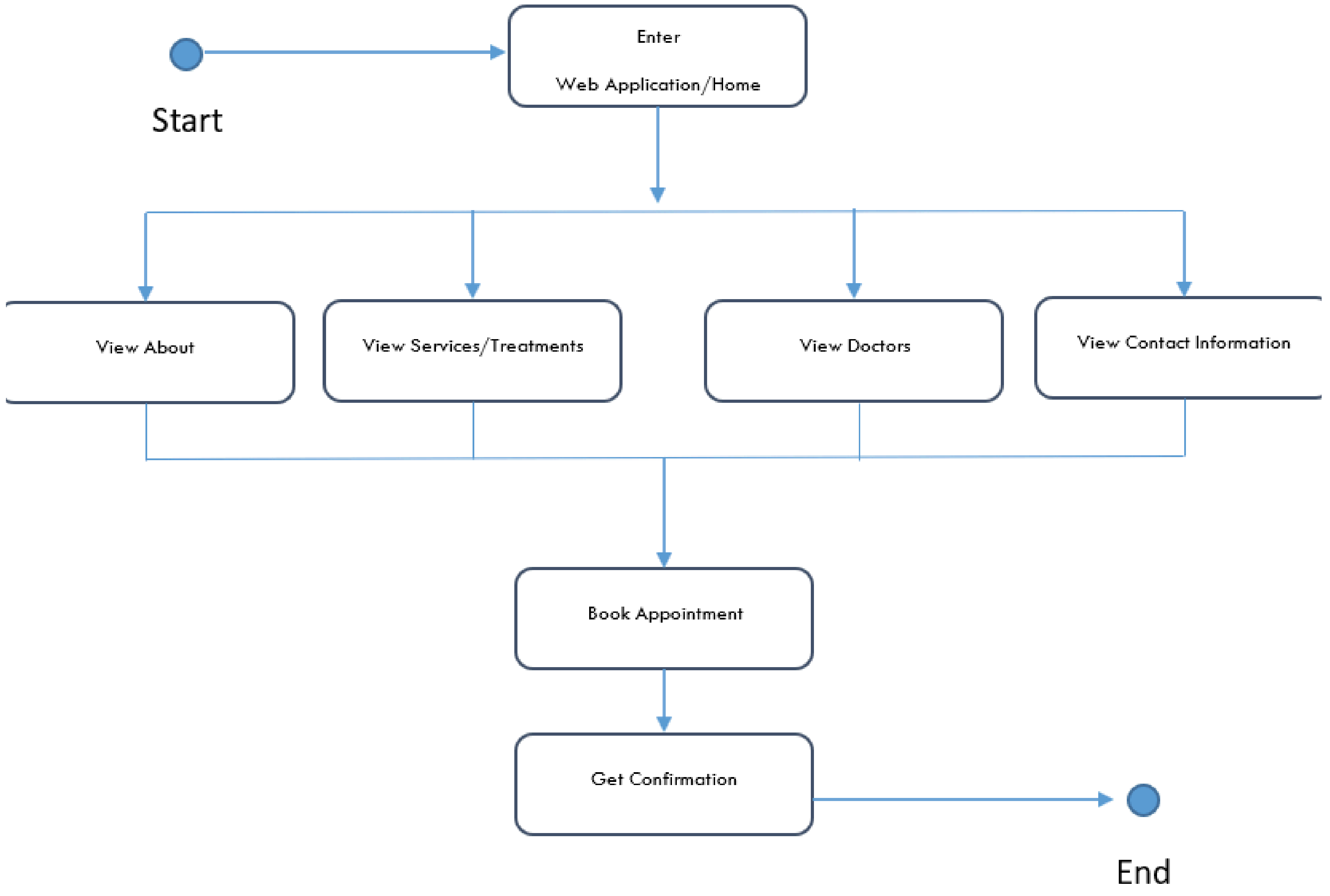
- Other Tools: (e.g., Postman, GitHub, Vercel/Render, Bootstrap/Tailwind)

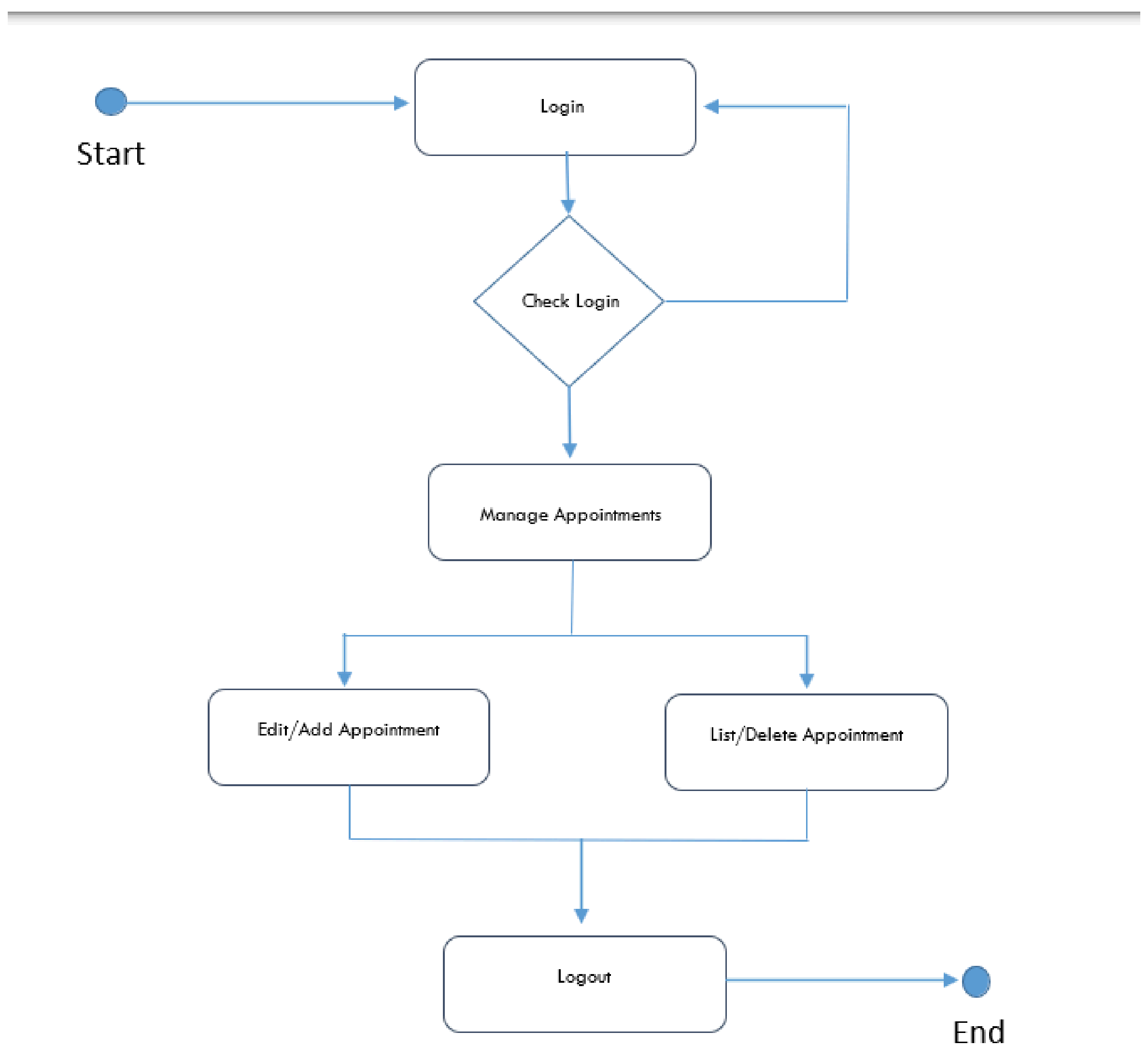
CORE FEATURES

- User Authentication (Login/Signup)
- Book Appointments
- View Available Doctors
- Doctor Dashboard (Appointments, Profile)
- Patient Dashboard (History, Cancel Booking)
- Admin Panel (optional)

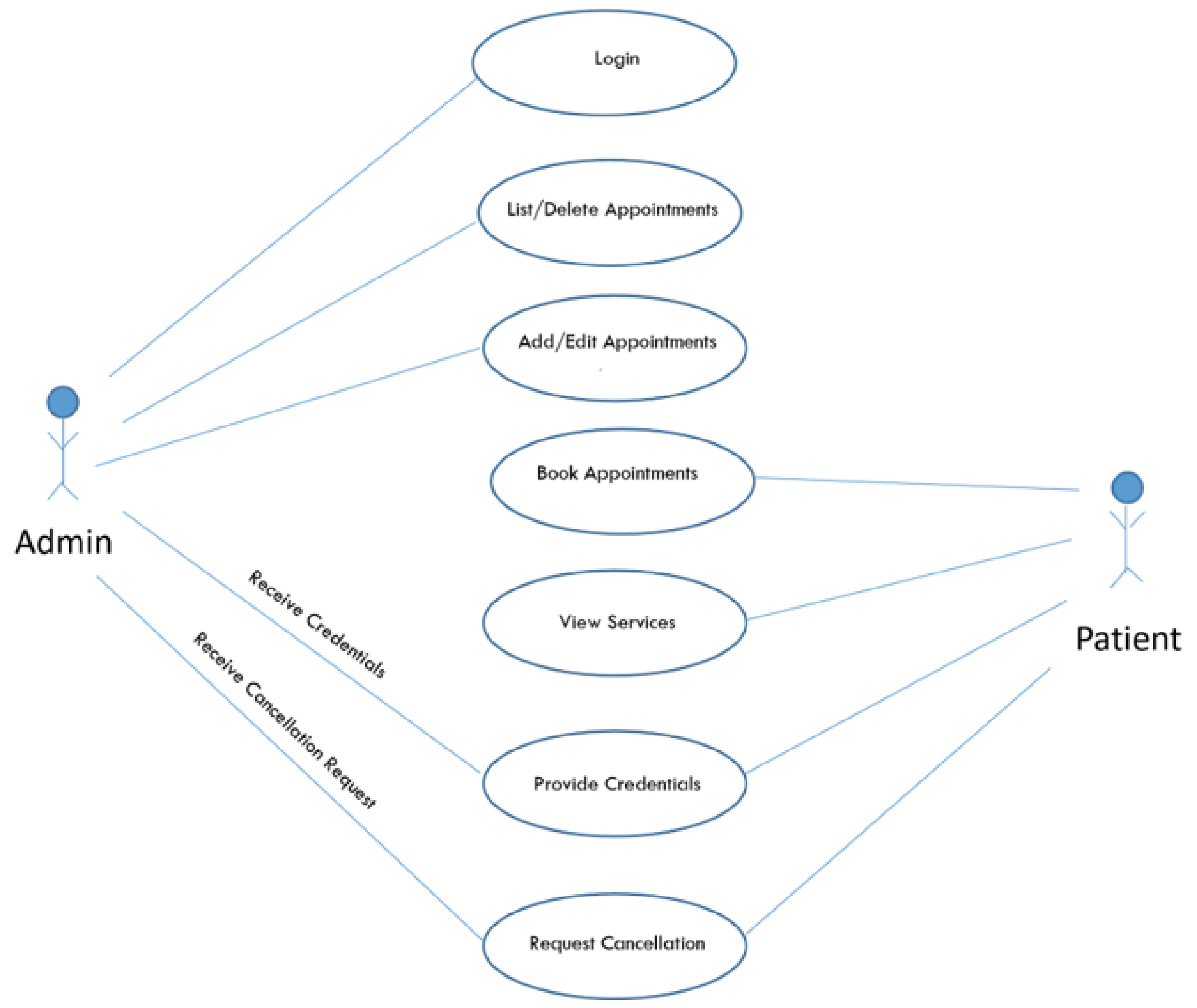
SYSTEM ARCHITECTURE





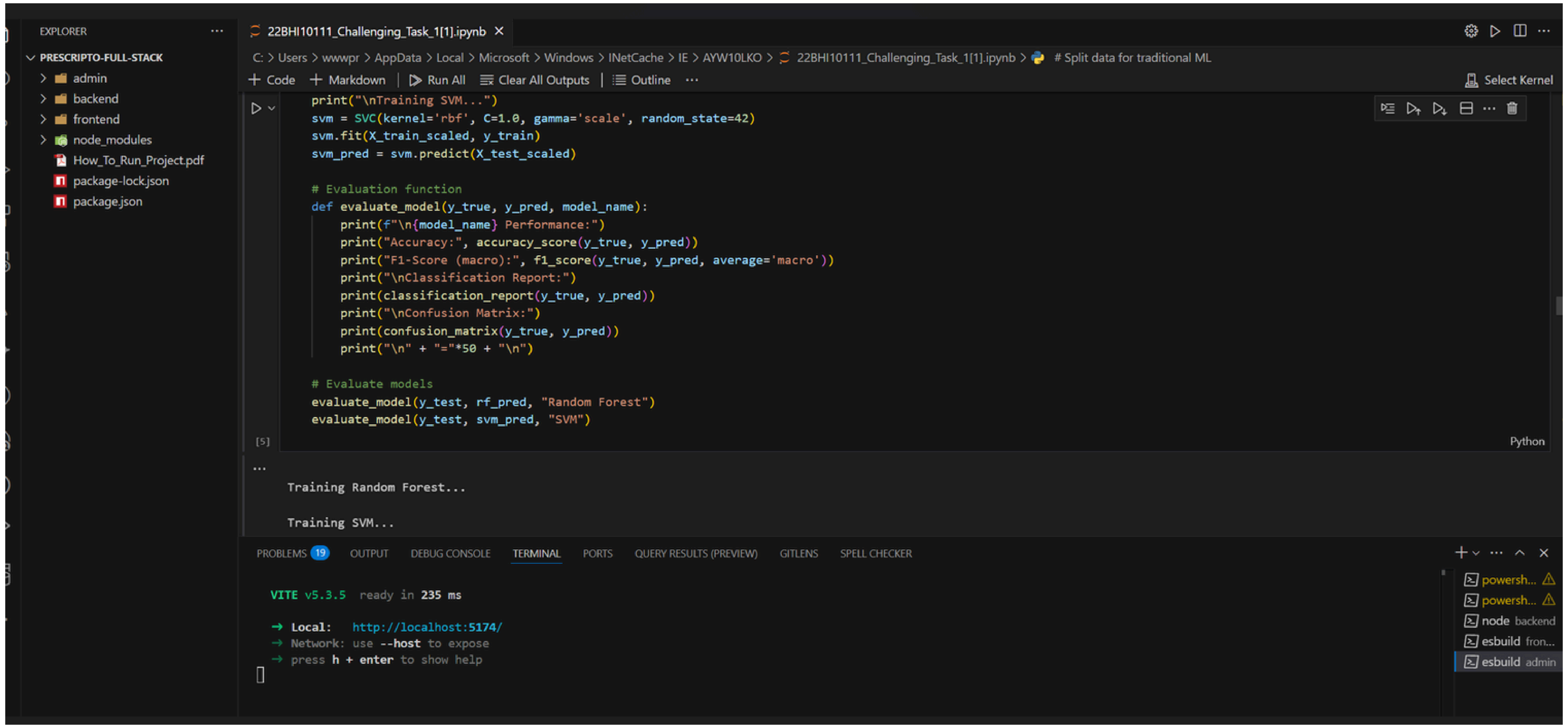


Admin Panel



Use Case Diagram

Admin panel



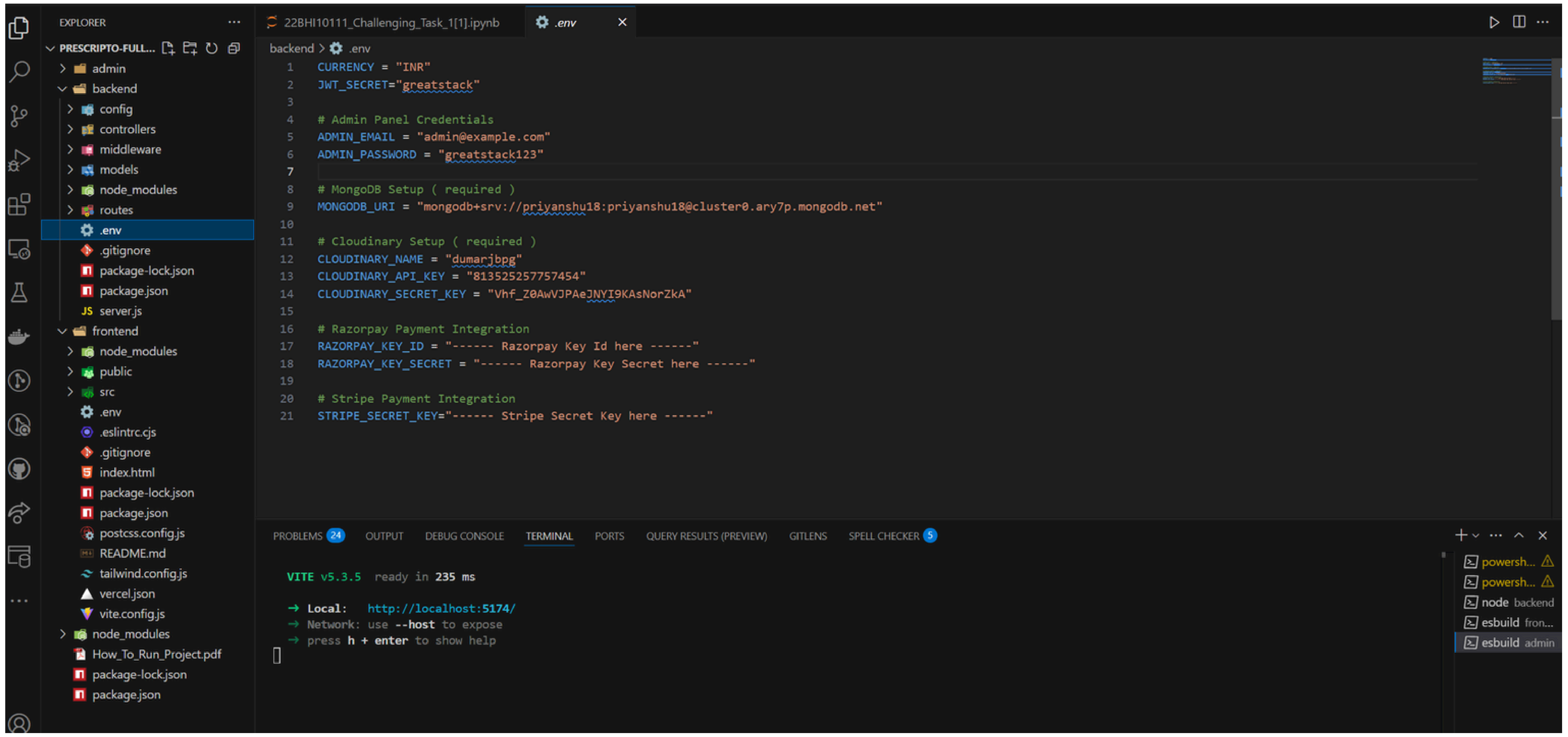
```python
print("\nTraining SVM...")
svm = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
svm.fit(X_train_scaled, y_train)
svm_pred = svm.predict(X_test_scaled)

# Evaluation function
def evaluate_model(y_true, y_pred, model_name):
    print(f"\n{model_name} Performance:")
    print("Accuracy:", accuracy_score(y_true, y_pred))
    print("F1-Score (macro):", f1_score(y_true, y_pred, average='macro'))
    print("\nClassification Report:")
    print(classification_report(y_true, y_pred))
    print("\nConfusion Matrix:")
    print(confusion_matrix(y_true, y_pred))
    print("\n" + "="*50 + "\n")

# Evaluate models
evaluate_model(y_test, rf_pred, "Random Forest")
evaluate_model(y_test, svm_pred, "SVM")
```

[5]

```
...
Training Random Forest...

Training SVM...
```

```
VITE v5.3.5  ready in 235 ms

➜  Local:   http://localhost:5174/
➜  Network: use --host to expose
➜  press h + enter to show help
```

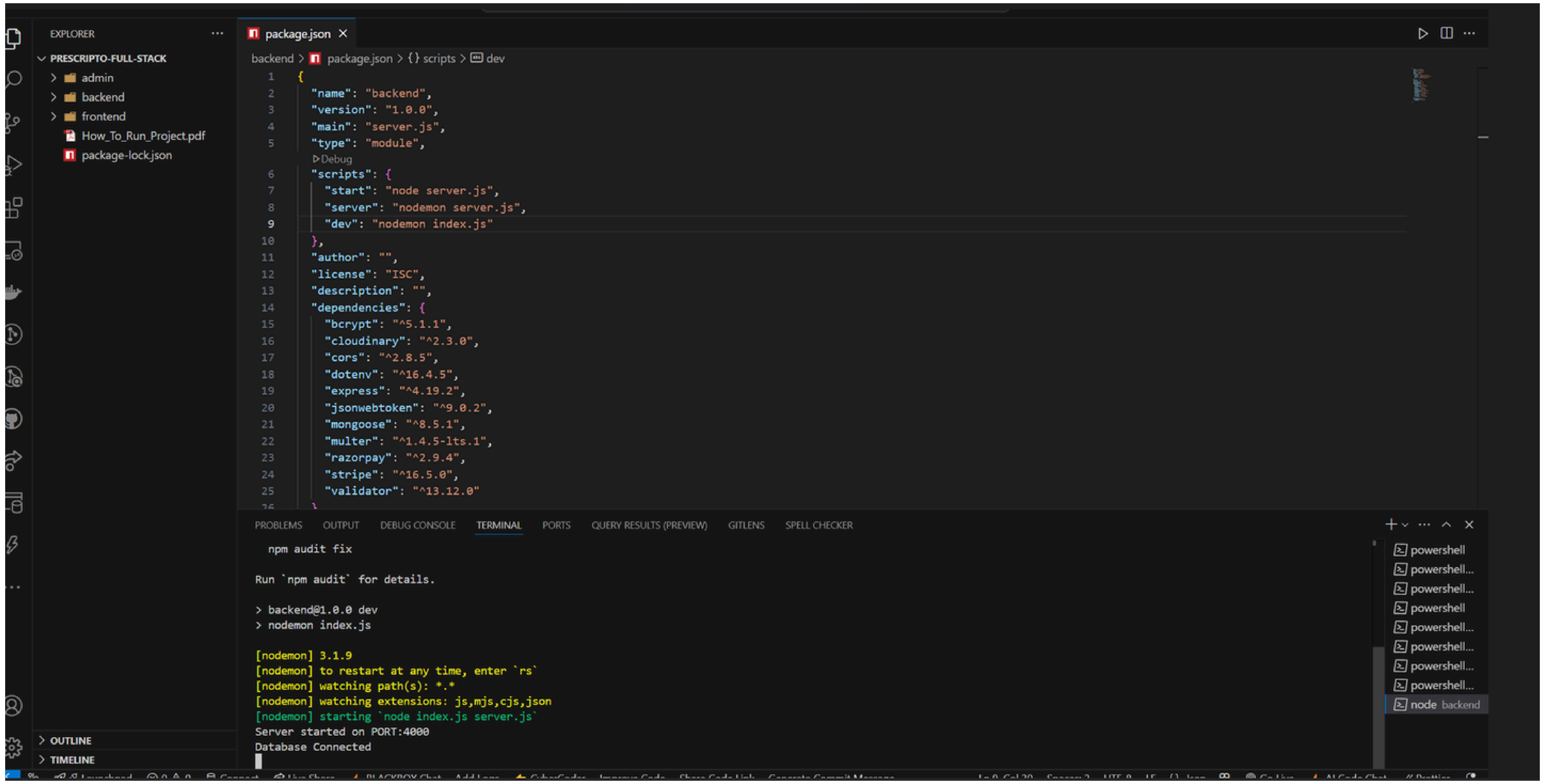
Credentials-Admin Panel,MongoDB,CLoudinary,Payment(Razorpay ,Stripe)



```
CURRENCY = "INR"
JWT_SECRET="greatstack"

# Admin Panel Credentials
ADMIN_EMAIL = "admin@example.com"
ADMIN_PASSWORD = "greatstack123"

# MongoDB Setup ( required )
MONGODB_URI = "mongodb+srv://priyanshu18:priyanshu18@cluster0.ary7p.mongodb.net"

# Cloudinary Setup ( required )
CLOUDINARY_NAME = "dumarjbpg"
CLOUDINARY_API_KEY = "813525257757454"
CLOUDINARY_SECRET_KEY = "Vhf_Z0AwVJPAeJNYI9KAsNorZkA"

# Razorpay Payment Integration
RAZORPAY_KEY_ID = "------ Razorpay Key Id here ------"
RAZORPAY_KEY_SECRET = "------ Razorpay Key Secret here ------"

# Stripe Payment Integration
STRIPE_SECRET_KEY="------ Stripe Secret Key here ------"
```

```
VITE v5.3.5  ready in 235 ms

➜  Local:   http://localhost:5174/
➜  Network: use --host to expose
➜  press h + enter to show help
```

Database connected



```
{
  "name": "backend",
  "version": "1.0.0",
  "main": "server.js",
  "type": "module",
  ▷Debug
  "scripts": {
    "start": "node server.js",
    "server": "nodemon server.js",
    "dev": "nodemon index.js"
  },
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "bcrypt": "^5.1.1",
    "cloudinary": "^2.3.0",
    "cors": "^2.8.5",
    "dotenv": "^16.4.5",
    "express": "^4.19.2",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^8.5.1",
    "multer": "^1.4.5-lts.1",
    "razorpay": "^2.9.4",
    "stripe": "^16.5.0",
    "validator": "^13.12.0"
```

```
  npm audit fix

Run `npm audit` for details.

> backend@1.0.0 dev
> nodemon index.js

[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js server.js`
Server started on PORT:4000
Database Connected
```

FUTURE SCOPE

- Online Payment Integration

Enable patients to pay consultation fees securely using payment gateways like Razorpay, Stripe, or PayPal.

- Video Consultation Feature

Allow virtual appointments through video calls for remote or follow-up consultations.

- Automated Reminders

Send SMS or email reminders for upcoming appointments to reduce no-shows.

- Medical Report Uploads

Let patients upload and share medical documents or reports with doctors securely.

- Rating & Review System

Allow patients to rate and review doctors to help others make informed choices.

CONCLUSION

- Book a Doctor is a full-stack web application built using the MERN stack that aims to streamline the process of booking medical appointments online. By integrating secure user authentication, dynamic scheduling, and a user-friendly interface, the platform ensures a smooth experience for both patients and doctors. This project not only demonstrates the practical application of modern web technologies but also addresses a real-world need for accessible and efficient healthcare services. Through this journey, I have gained hands-on experience in full-stack development, API integration, and managing real-time data—skills that are essential for building scalable and impactful applications.

THANK YOU