# Name: Parth Patel & Shardul Panchal

Course: CIS 434

DATE: 4th December 2024

# Developer's Guide: Healthcare Patient Management Application (Appointer)

# 1] Introduction

### Purpose

This guide provides developers with the necessary information to understand, set up, and contribute to the healthcare patient management application. It includes detailed instructions on project setup, architecture, workflows, and testing.

### Overview

The application enables patients to register, book, and manage appointments with doctors. Admins can handle scheduling, confirmations, and cancellations. The project is built using modern web technologies like Next.js, Appwrite, TypeScript, and TailwindCSS.

### Target Audience

- Developers contributing to the project.  
- Administrators responsible for deploying and managing the application.

### Prerequisites

Ensure you have the following installed on your machine:  
- Git  
- Node.js  
- npm (Node Package Manager)

### Cloning the Repository

**git clone** https://github.com/Parth2861866/CIS434

### Installation

Install the project dependencies:  
**npm install**

### Environment Setup

Create a .env.local file in the root directory with the following content:  
  
#APPWRITE  
NEXT\_PUBLIC\_ENDPOINT=https://cloud.appwrite.io/v1  
PROJECT\_ID=  
API\_KEY=  
DATABASE\_ID=  
PATIENT\_COLLECTION\_ID=  
APPOINTMENT\_COLLECTION\_ID=  
NEXT\_PUBLIC\_BUCKET\_ID=  
  
NEXT\_PUBLIC\_ADMIN\_PASSKEY=123456  
  
Replace placeholder values with your actual Appwrite credentials.

### Run the Project

Start the development server: **npm run dev**  
  
Visit [http://localhost:3000](http://localhost:3000) in your browser.

## 2] Project Overview

### Tech Stack

- Frontend: Next.js, TypeScript, TailwindCSS, ShadCN.  
- Backend: Appwrite for backend services.

### Key Features

- User registration and profile management.  
- Appointment booking and management (admin and user).  
- File uploads with Appwrite storage.  
- Responsive UI for all devices.

## 3] Development Workflow

### Branching Strategy

Follow a Git Flow or feature-based branching model. Commit messages should follow a clear convention (e.g., `[feature]: add new booking page`).

### Code Standards

- Use TypeScript for all development.  
- Follow ESLint and Prettier for consistent formatting.  
- Adhere to the DRY (Don't Repeat Yourself) principle.

### Testing

Use a combination of unit tests and end-to-end tests. Testing tools: Jest, Cypress. Test cases should be regularly updated based on new features or updates.

## 4] API and Database

### API Details

RESTful API endpoints for appointments, users, and admin operations. Use secure tokens for authentication and authorization.

### Database Design

Collections:  
- Users: Stores patient and admin data.  
- Appointments: Tracks all appointment details.

## 5] Deployment

Note: It is not deployed on any platform, but below procedures shows how to do it

### Build and Deployment Steps

1. Run “npm run build” to generate the production build.  
2. Deploy using your preferred cloud provider (e.g., Vercel, AWS).

### CI/CD Pipeline

Use tools like GitHub Actions for continuous integration. Ensure all tests pass before deploying to production.

## 6] Troubleshooting

### Common Issues

- Environment variables missing: Ensure `.env.local` is configured properly.  
- Database errors: Ensure the Appwrite instance is configured correctly.

## 7] Testing Guide

### Test Strategy

Unit tests for individual components and APIs. Integration tests for end-to-end workflows.

### Test Cases

Refer to the testing templates provided for detailed scenarios.

## 8] Future Enhancements

- Integration with Electronic Health Records (EHR).  
- AI-powered appointment scheduling assistant.  
- Mobile applications for iOS and Android.

## 9] Appendix

### Glossary

- Next.js: A React framework for server-side rendering.  
- Appwrite: Backend platform for web and mobile apps.  
- TailwindCSS: Utility-first CSS framework for rapid UI development.