Swasthya Setu IHWP

A comprehensive healthcare platform built with MERN stack (React frontend, Node.js/Express backend, MongoDB) plus an admin panel — aimed at delivering seamless signup/login, user interactions, and admin management.

Table of Contents

About the Project

Swasthya Setu IHWP is designed to provide a one-stop digital solution for health-care management. Users can sign up, login, access features via the frontend; admins can manage data via an admin panel. The backend connects both and uses MongoDB for data persistence.
It was developed in the context of a Indian Health Wellness And Psychology(IHWP) Project for Academics.

Key Features

- Ayurvedic Integration
    - Traditional dosha assessment methodology
    - Personalized recommendations based on constitution
    - Holistic wellness approach
- Comprehensive Wellness Tracking
    - Multi-dimensional health monitoring
    - Integrated task and health management
    - Progress visualization and reporting
    - User Experience
        - Responsive design across devices
        - Intuitive navigation and interface
        - PDF report generation
        - Real-time data synchronization
- Administrative Control
    - Complete system oversight
    - User behavior analytics
    - Health trend monitoring
    - Data-driven insights
- Security & Privacy
    - Secure authentication system
    - Protected API endpoints
    - Data encryption and validation
    - Role-based access control

1. Main Frontend Application (swasthyasetu)

- **Technology Stack:** React.js, Material-UI, Axios, HTML2Canvas, jsPDF
- **Core Features:**
- **Authentication System**

    - User registration and login
    - JWT token-based authentication
    - Protected routes
- **Dosha Assessment Module**
    - 12-question Ayurvedic constitution quiz
    - Real-time progress tracking
    - Detailed results with Vata, Pitta, Kapha percentages
    - Personalized recommendations for diet, lifestyle, and daily schedule
    - PDF report generation and download
    - Assessment history storage
    - Observation table for detailed analysis
- **Todo Manager & Health Tracking**
    - Task management with categories (water, exercise, food, meditation, sleep, general)
    - Priority levels (low, medium, high)
    - Health metrics tracking (water intake, exercise minutes, sleep hours, mood)
    - Daily notes and observations
    - Progress reports (daily, weekly, monthly)
    - Wellness insights and suggestions
- **User Profile Management**
    - Personal information display
    - Assessment history viewing
    - Progress tracking dashboard
- **Resources & Information**
    - Ayurveda educational content
    - Wellness tips and guidance

    - About page with project information

2. Admin Panel (swasthyasetu-admin)

- **Technology Stack:** React.js, Chart.js, React-ChartJS-2, Material-UI
- **Administrative Features:**
- **Dashboard Analytics**
    - User statistics overview
    - Assessment completion metrics
    - Todo and report analytics
    - Visual charts (Dosha distribution, daily assessments)
- **User Management**
    - Complete user database view
    - Individual user profile details
    - User assessment history
    - Todo tracking per user
    - Health data monitoring
    - Generated reports overview
- **Assessment Monitoring**
    - All assessments database
    - Dosha distribution analysis
    - Assessment completion tracking
    - Detailed assessment results viewing
- **Todo & Health Data Management**
    - System-wide todo monitoring
    - Health tracking data analysis
    - User wellness progress oversight
- **Report Management**

- All user reports viewing
- Report analytics and insights
- System-wide wellness trends

3. Backend API (swasthyasetu-backend)

- **Technology Stack:** Node.js, Express.js, MongoDB, Mongoose, JWT, Bcrypt
- **API Modules:**
- **Authentication Module**
  - User registration with password hashing
  - Secure login with JWT tokens
  - Password encryption using bcrypt
- **Assessment Module**
  - Dosha assessment data storage
  - Results calculation and storage
  - User assessment history retrieval
  - Assessment analytics for admin
- **Todo Management Module**
  - CRUD operations for todos
  - Category and priority management
  - Task completion tracking
  - Health data integration
- **Health Tracking Module**
  - Daily health metrics storage
  - Water intake, exercise, sleep tracking
  - Mood and notes recording
  - Health data analytics
- **Report Generation Module**

    - Automated report creation (daily, weekly, monthly)
    - Wellness insights generation
    - Progress analysis and suggestions
    - Report management (create, read, delete)
- **Admin Module**
    - Admin authentication
    - User management APIs
    - System statistics generation
    - Chart data preparation
    - Comprehensive data analytics
- **Database Models:**
    - **User Model:** Personal information, authentication
    - **Assessment Model:** Dosha quiz results and analysis
    - **Todo Model:** Task management with categories
    - **HealthTracking Model:** Daily wellness metrics
    - **Report Model:** Generated wellness reports
    - **Admin Model:** Administrative access control
- Admin API: System statistics and user management

Architecture & Tech Stack

Frontend:
- React (with routers, components, hooks)
- CSS any UI library (Material UI)

Backend:
- Node.js + Express for RESTful APIs
- Authentication via JWT
- bcrypt (or equivalent) for password hashing

Database:
- MongoDB (NoSQL)

Dev Tools:
- npm / yarn - Postman (for testing APIs)
- Git & GitHub for version control

Getting Started

Prerequisites

- Node.js & npm installed
- MongoDB instance (local or cloud e.g. MongoDB Atlas)
- Git (to clone the repo)

Installation

1. Clone the repo:

   git clone https://github.com/bendalejatin/Swasthya-Setu-IHWP.git

2. Navigate into each module (frontend, backend, admin) and install dependencies (open each module in different Terminal): bash cd swasthyasetu-backend npm install cd ../swasthyasetu npm install cd ../swasthyasetu-admin npm install

Running Locally

Backend: bash cd swasthyasetu-backend node server.js # or npm start

Frontend: bash cd swasthyasetu npm start

Admin Module: bash cd swasthyasetu-admin npm start —

Project Structure

- /swasthyasetu-backend

  /controllers
  /models
  /routes
  createAdmin.js
  test-server.js
  server.js

- /swasthyasetu

  /src
    /components
    /Auth
    /services
    App.js

- /swasthyasetu-admin

  /src
    /components
    /services
    App.js

- Each module has its own package.json and dependency setup.

Usage

User Flow (Frontend)

1. Authentication Flow Landing Page → Sign Up/Login → JWT Token → Dashboard Access

2. Main User Journey

Home Page
↓
Login/Register
↓
User Dashboard
- Dosha Assessment
  - 12 Questions Quiz
  - Results & Recommendations
  - PDF Download
- Todo Manager
  - Create Tasks (Categories: Water, Exercise, Food, Meditation, Sleep)
  - Set Priority (Low/Medium/High)
  - Mark Complete/Incomplete
  - Filter & Manage Tasks
- Health Tracking
  - Water Intake Counter
  - Exercise Minutes
  - Sleep Hours
  - Mood Selection
  - Daily Notes
- Reports
  - Generate Reports (Daily/Weekly/Monthly)
  - View Wellness Insights
  - Progress Analytics
  - Delete Reports
- Profile
  - Personal Info

- Assessment History
- Progress Overview

3. Detailed User Actions
    - Assessment: Take quiz → Get dosha results → Download PDF → View recommendations
    - Tasks: Add todo → Set category/priority → Track completion → Filter by status
    - Health: Log daily metrics → Track progress → View trends
    - Reports: Generate insights → Review suggestions → Monitor wellness journey

Admin Flow

1. Admin Authentication

    Admin Login Page → Admin Credentials → Admin Dashboard

2. Admin Dashboard Navigation

    Admin Dashboard
    - Statistics
        - Total Users Count
        - Total Assessments
        - Recent Activity (7 days)
        - Dosha Distribution Chart
        - Daily Assessment Trends
    - Users Management
        - View All Users List
        - Select User → Detailed Profile
        - Assessment History per User
        - Todo Activity per User
        - Health Data per User
        - Reports per User
    - Assessments Overview
        - All Assessment Results
        - Dosha Distribution Analysis
        - Assessment Completion Rates
        - Detailed Assessment View
    - Todo Management
        - System-wide Todo Overview
        - User Task Analytics
        - Category-wise Distribution
        - Completion Statistics
    - Reports Analytics
        - All Generated Reports
        - User Report Patterns

├── Wellness Trends Analysis
└── System Health Insights

3. Admin Capabilities
    - Monitor: Track all user activities and system health
    - Analyze: View charts, statistics, and wellness trends
    - Manage: Oversee user data and system performance
    - Insights: Generate system-wide analytics and reports

Data Flow Architecture

1. User Data Flow:
    - User Action → Frontend (React) → API Call → Backend (Express) → Database (MongoDB) → Response → UI Update
2. Admin Data Flow:
    - Admin Query → Admin Panel → Admin API → Database Aggregation → Charts/Analytics → Dashboard Display

Security Flow

- User: JWT token validation for protected routes
- Admin: Separate admin token for dashboard access
- API: Token verification middleware on all protected endpoints
- Data: Encrypted passwords with bcrypt, CORS protection

Navigation Structure

User App Routes

- / - Home Page
- /login - User Login
- /signup - User Registration
- /profile - User Dashboard
- /dosha-assessment - Ayurvedic Quiz
- /todo-manager - Task & Health Management
- /features - App Features
- /resources - Wellness Resources
- /about - About Page

Admin Panel Routes

- / - Admin Login
- /dashboard - Admin Dashboard with tabs:
    - Statistics
    - Users

- Assessments
- Todos
- Reports

Database & API Endpoints

Database Schema (MongoDB)

1. Users Collection

```
{
  _id: ObjectId,
  name: String (required),
  email: String (required, unique),
  password: String (required, hashed),
  phone: String (optional)
}
```

2. Assessments Collection

```
{
  _id: ObjectId,
  userId: ObjectId (ref: User, required),
  type: String (enum: ["dosha"], required),
  responses: [String] (required),
  results: {
    percentages: {
      Vata: Number (required),
      Pitta: Number (required),
      Kapha: Number (required)
    },
    dominant: String (required),
    secondary: String (required)
  },
  completedAt: Date (default: now)
}
```

3. Todos Collection

```
{
  _id: ObjectId,
  userId: ObjectId (ref: User, required),
  title: String (required),
  description: String,
  category: String (enum: ["general", "water", "exercise", "food", "meditation", "sleep"], default: "general
"),
  completed: Boolean (default: false),
  priority: String (enum: ["low", "medium", "high"], default: "medium"),
  dueDate: Date,
  createdAt: Date (default: now),
  completedAt: Date
}
```

4. HealthTracking Collection

```
{
  _id: ObjectId,
  userId: ObjectId (ref: User, required),
  date: Date (required),
  waterIntake: Number (default: 0),
  exerciseMinutes: Number (default: 0),
  meals: [{
    name: String,
    calories: Number,
    time: Date
  }],
  sleepHours: Number (default: 0),
  mood: String (enum: ["excellent", "good", "okay", "poor", "terrible"]),
  notes: String
}
```

5. Reports Collection

```
{
  _id: ObjectId,
  userId: ObjectId (ref: User, required),
  date: Date (required),
  type: String (enum: ["daily", "weekly", "monthly"], required),
  data: {
    completedTodos: Number,
    totalTodos: Number,
    waterIntake: Number,
    exerciseMinutes: Number,
    averageMood: String,
    sleepHours: Number
  },
  suggestions: [String],
  createdAt: Date (default: now)
}
```

6. Admins Collection

```
{
  _id: ObjectId,
  username: String (required, unique),
  password: String (required, hashed),
  email: String (required, unique),
  role: String (default: "admin")
}
```

API Endpoints

1. Authentication Routes (/)

   POST /signup - User registration
   POST /login - User login

2. Assessment Routes (/api)

   POST /api/assessment - Save dosha assessment
   GET /api/assessment/:userId - Get user assessments

3. Todo & Health Routes (/api)

   POST /api/todos - Create todo (🔒 Auth)
   GET /api/todos - Get user todos (🔒 Auth)
   PUT /api/todos/:id - Update todo (🔒 Auth)
   DELETE /api/todos/:id - Delete todo (🔒 Auth)
   POST /api/health - Update health data (🔒 Auth)
   GET /api/health - Get health data (🔒 Auth)
   GET /api/reports/generate - Generate report (🔒 Auth)
   GET /api/reports - Get user reports (🔒 Auth)
   DELETE /api/reports/:id - Delete report (🔒 Auth)

4. Admin Routes (/admin)

   POST /admin/login - Admin login
   GET /admin/users - Get all users
   GET /admin/users/:userId - Get user details
   GET /admin/assessments - Get all assessments
   GET /admin/todos - Get all todos
   GET /admin/reports - Get all reports
   GET /admin/stats - Get system statistics
   GET /admin/charts - Get chart data

Testing

- You can use Postman or similar tools to test APIs.
- Run frontend and admin locally and verify flows manually (login, signup, access control).
- Test with multiple users
- Verify database query performance
- Check API response times
- Monitor memory usage

License

Contact

- Author: Jatin Bendale

- GitHub: @bendalejatin