ZenFitAnalyzer

Project Overview

ZenFitAnalyzer is a comprehensive health and fitness tracking application built with a React frontend and Express.js backend. The platform offers various features to help users track their health, fitness, nutrition, and wellness journey.

Technology Stack

Frontend

- **Framework**: React.js with Vite
- **Routing**: React Router Dom v7
- **Styling**: Tailwind CSS, PostCSS
- **State Management**: Context API
- **UI Components**: React Icons
- **HTTP Client**: Axios
- **Data Visualization**: Recharts
- **Notifications**: React Toastify
- **Video Input**: React Webcam

Backend

- **Runtime**: Node.js
- **Framework**: Express.js
- **Database**: MongoDB with Mongoose
- **Authentication**: JWT (JSON Web Token), bcrypt
- **File Handling**: Multer, Cloudinary
- **Email Service**: Nodemailer
- **Real-time Communication**: Socket.io
- **AI Integration**: TensorFlow.js
- **Third-party APIs**: Spotify Web API

Key Features

1. **User Authentication**
    - Registration and login system
    - Profile management
    - JWT-based authentication
2. **Meal Planning & Nutrition**
    - Meal tracking and logging
    - Personalized meal recommendations
3. **Workout Tracking**

- Customized fitness routines
- Exercise tracking
- Workout tutorials

4. **Body Progress Analysis**
    - Weight tracking
    - Body measurements
    - BMI calculation
5. **Sleep Tracking**
    - Sleep pattern monitoring
    - Sleep quality analysis
6. **Activity Tracking**
    - GPS-based tracking for running, cycling, walking
    - Real-time location updates
7. **Habit Tracking**
    - Daily habit monitoring
    - Habit streak tracking
8. **Music Integration**
    - Spotify integration for workout music
9. **Social Features**
    - Community feed
    - Challenges and group workouts
    - Social sharing
10. **AI-Powered Features**
    - Chatbot assistance for fitness guidance
11. **Marketplace**
    - ZenFit marketplace for fitness products
    - Shopping cart functionality
12. **Health & Wellness Content**
    - Articles on fitness, nutrition, self-care, and wellness
    - Health tips and advice

Technical Implementation Details

Authentication Flow

1. **Registration Process**
    - User submits registration form
    - Server validates input data
    - Password is hashed using bcrypt
    - User document is created in MongoDB
    - JWT token is generated and returned
2. **Login Process**
    - User submits login credentials
    - Server validates credentials
    - Password is verified against hashed value
    - JWT token is generated and returned
    - Token is stored in localStorage
3. **Token Verification**
    - JWT token is included in Authorization header
    - Server middleware verifies token validity
    - User ID is extracted from token payload
    - Protected routes access is granted

Data Management

1. **MongoDB Schema Design**
    - User schema with profile information
    - Activity schema for workout tracking
    - Body progress schema for measurements
    - Sleep schema for sleep tracking
    - Challenge schema for fitness challenges
2. **Data Relationships**
    - User references in all related schemas
    - One-to-many relationship between users and activities
    - One-to-many relationship between users and body progress records

Frontend Architecture

1. **Component Structure**
    - Reusable UI components in Components directory
    - Page components in Pages directory
    - Context providers for state management
2. **Responsive Design**
    - Mobile-first approach with Tailwind CSS
    - Custom breakpoints for different device sizes

Project Structure

Frontend Structure

```
frontend/
├── public/                # Static files
├── src/
│   ├── assets/            # Images, icons, etc.
│   ├── Components/        # Reusable UI components
│   │   ├── ActivityTracker/  # Activity tracking components
│   │   ├── Advice/        # Health & wellness content components
│   │   ├── Common/        # Shared UI components
│   │   ├── Community/     # Social features components
│   │   ├── Home/          # Homepage components
│   │   ├── Layout/        # Layout components
│   │   ├── Music/         # Music integration components
│   │   ├── Profile/       # User profile components
│   │   ├── Progress/      # Body progress components
│   │   ├── SleepComponent/ # Sleep tracking components
│   │   └── Tutorial.jsx/  # Tutorial components
│   ├── Context/           # React context providers (includes CartContext)
│   ├── Pages/             # Application pages
│   │   ├── Advices/       # Health and wellness content pages
│   │   ├── HomeFooter/    # Main app sections
│   │   ├── Musics/        # Music integration pages
│   │   ├── Onboarding/    # User onboarding flow
│   │   └── Register/      # Authentication pages
│   ├── App.jsx            # Main application component
│   ├── index.css          # Global styles
│   └── main.jsx           # Application entry point
```

Backend Structure

```
Backend/
├── config/                # Configuration files
├── controllers/           # Request handlers
│   ├── user.controller.js         # User management
│   ├── bodyProgress.controller.js # Body progress tracking
│   ├── sleep.controller.js        # Sleep tracking
│   ├── habitController.js         # Habit tracking
│   ├── challenge.controller.js    # Fitness challenges
│   ├── social.controller.js       # Social features
│   └── chatbot.controller.js      # AI chatbot
├── db/                    # Database connection
├── middlewares/           # Express middlewares
├── Models/                # Mongoose schemas
│   ├── user.model.js          # User schema
│   ├── bodyProgress.model.js  # Body progress schema
│   ├── sleep.model.js         # Sleep tracking schema
│   ├── habit.js               # Habit tracking schema
```

```
│   ├── challenge.model.js     # Challenges schema
│   ├── post.model.js          # Social posts schema
│   ├── gpsActivity.js         # GPS tracking schema
│   ├── Song.js                # Music integration schema
│   ├── chat.model.js          # Chatbot conversations schema
│   └── blacklistToken.model.js # Token blacklisting for logout
├── routes/                # API routes
│   ├── users.js               # User routes
│   ├── bodyProgress.routes.js # Body progress routes
│   ├── sleep.routes.js        # Sleep tracking routes
│   ├── habitRoutes.js         # Habit tracking routes
│   ├── challenge.routes.js    # Challenges routes
│   ├── social.routes.js       # Social features routes
│   ├── gps.js                 # GPS tracking routes
│   └── chatbot.js             # Chatbot routes
├── uploads/               # File uploads
└── app.js                 # Main application file
```

API Endpoints

User Authentication & Management

Endpoint	Method	Description	Authentication Required
`/users/register`	POST	Register a new user	No
`/users/login`	POST	Login an existing user	No
`/users/profile`	GET	Get user profile data	Yes
`/users/logout`	GET	Logout user	Yes
`/users/update-profile`	PUT	Update user profile information	Yes
`/users/water-intake`	GET/POST	Update and retrieve water intake	Yes
`/users/bmi`	GET/POST	Calculate and retrieve BMI	Yes
`/users/upload-photo`	POST	Upload profile photo	Yes
`/users/change-password`	PUT	Change user password	Yes
`/users/contact`	GET/POST	Send and retrieve contact messages	Yes

Body Progress

Endpoint	Method	Description	Authentication Required
`/body-progress/add`	POST	Add body progress record	Yes
`/body-progress/user`	GET	Get all body progress records	Yes

Endpoint	Method	Description	Authentication Required
`/body-progress/stats`	GET	Get summarized progress statistics	Yes

Sleep Tracking

Endpoint	Method	Description	Authentication Required
`/sleep/add`	POST	Add sleep record	Yes
`/sleep/user`	GET	Get user's sleep records	Yes

Habit Tracking

Endpoint	Method	Description	Authentication Required
`/habit/add`	POST	Add habit record	Yes
`/habit/user`	GET	Get user's habits	Yes

Challenge & Social Features

Endpoint	Method	Description	Authentication Required
`/challenge/create`	POST	Create a new challenge	Yes
`/challenge/join/:id`	POST	Join a challenge	Yes
`/challenge/active`	GET	Get active challenges	Yes
`/social/posts`	GET	Get social feed posts	Yes
`/social/posts`	POST	Create a new post	Yes
`/social/posts/:id/like`	POST	Like a post	Yes
`/social/posts/:id/comment`	POST	Comment on a post	Yes

Chatbot

Endpoint	Method	Description	Authentication Required
`/chatbot/chat`	POST	Get chatbot response	Yes

Music Integration

Endpoint	Method	Description	Authentication Required
`/api/songs`	GET	Get recommended songs for workout	Yes

Testing Procedures

Backend Testing

1. **Manual API Testing**
    - Test each endpoint using Postman or similar tools
    - Verify expected responses for valid and invalid requests
    - Check authentication and authorization
2. **Database Testing**
    - Verify data persistence
    - Test data relationships
    - Validate schema constraints

Frontend Testing

1. **Component Testing**
    - Test rendering of components
    - Verify user interactions
    - Check responsive design
2. **Integration Testing**
    - Test complete user flows
    - Verify data fetching and display
    - Test form submissions
3. **Cross-browser Testing**
    - Test on Chrome, Firefox, Safari, Edge
    - Check mobile responsiveness

Installation & Setup

Prerequisites

- Node.js (v14 or later)
- MongoDB
- npm or yarn

Frontend Setup

```
cd frontend
npm install
npm run dev
```

Backend Setup

```
cd Backend
npm install
```

```
nodemon
```

Deployment

Backend Deployment

1. **Prepare for Production**

```
cd Backend
npm install
```

2. **Set Environment Variables**
   - Create a `.env` file with production values
   - Ensure MongoDB connection string points to your production database

3. **Deploy to Server**
   - **Using PM2**:

```
npm install -g pm2
pm2 start app.js --name "zenfit-backend"
```

   - **Using Docker**:

```
docker build -t zenfit-backend .
docker run -p 4000:4000 -d zenfit-backend
```

Frontend Deployment

1. **Build for Production**

```
cd frontend
npm run build
```

2. **Deploy the Build Folder**
   - **Using Netlify/Vercel**:
     - Connect your GitHub repository
     - Configure build settings:
       - Build command: `npm run build`
       - Publish directory: `dist`

        - Set environment variables
    - **Using Traditional Hosting**:
        - Upload the contents of the `dist` folder to your web server
3. **Configure Environment Variables**
    - Ensure VITE_BASE_URL points to your production backend

Environment Variables

Backend (.env)

```
PORT=4000
MONGODB_URI=your_mongodb_connection_string
JWT_SECRET=your_jwt_secret
CLOUDINARY_CLOUD_NAME=your_cloudinary_name
CLOUDINARY_API_KEY=your_cloudinary_api_key
CLOUDINARY_API_SECRET=your_cloudinary_api_secret
EMAIL_USER=your_email@gmail.com
EMAIL_PASS=your_email_password
SPOTIFY_CLIENT_ID=your_spotify_client_id
SPOTIFY_CLIENT_SECRET=your_spotify_client_secret
```

Frontend (.env)

```
VITE_BASE_URL=http://localhost:4000
VITE_SPOTIFY_CLIENT_ID=your_spotify_client_id
VITE_SPOTIFY_REDIRECT_URI=http://localhost:5173/musichome
```

Troubleshooting

Common Issues

Backend Issues

1. **MongoDB Connection Failed**
    - Verify MongoDB is running
    - Check connection string in `.env` file
    - Ensure network allows connection to MongoDB server
2. **JWT Authentication Errors**
    - Verify JWT_SECRET is set correctly
    - Check token expiration settings
    - Clear browser cookies and localStorage

3. **File Upload Issues**
    - Verify Cloudinary credentials
    - Check permissions on uploads directory
    - Ensure file size limits are appropriate

Frontend Issues

1. **API Connection Errors**
    - Verify VITE_BASE_URL is set correctly
    - Check network for CORS issues
    - Ensure backend is running and accessible

2. **Authentication Flow Problems**
    - Clear localStorage and try again
    - Check browser console for errors
    - Verify that token is being stored properly

Frequently Asked Questions (FAQ)

General

Q: What is ZenFitAnalyzer?

A: ZenFitAnalyzer is a comprehensive health and fitness tracking application that helps users monitor various aspects of their wellness journey, including nutrition, workouts, sleep, body progress, and more.

Q: Is ZenFitAnalyzer free to use?

A: The application offers a free tier with basic features. Premium features may be available in future updates.

Technical

Q: How is my data protected?

A: ZenFitAnalyzer implements industry-standard security measures including password hashing, JWT authentication, HTTPS encryption, and secure database storage to protect user data.

Q: Can I export my data from ZenFitAnalyzer?

A: Currently, data export functionality is planned for future updates.

Features

Q: Can I connect my fitness wearable to ZenFitAnalyzer?

A: Currently, direct integration with fitness wearables is in development. We plan to support major devices in upcoming releases.

Q: How does the AI-powered workout feedback work?

A: Our AI system analyzes your workout patterns, progress, and goals to provide personalized recommendations and feedback to help optimize your fitness routine.

Security Considerations

- All user passwords are hashed using bcrypt before storage
- JWT tokens are used for authentication with appropriate expiration
- Input validation is implemented on all API endpoints
- CORS is configured to restrict access to trusted domains
- Authentication middleware protects sensitive routes

Roadmap

- Mobile applications for iOS and Android
- Wearable device integrations
- Advanced analytics dashboard
- Enhanced AI-powered personalized recommendations
- Group challenges with real-time leaderboards

Version History

v1.0.0 (Initial Release)

- Core authentication system
- Basic profile management
- Meal tracking functionality
- Simple workout tracking

v1.1.0

- Added body progress tracking
- Implemented BMI calculator
- Enhanced meal tracking

v1.2.0 (Current)

- Added sleep tracking
- Integrated Spotify for workout music
- Implemented social feed
- Added challenges feature
- AI-powered chatbot for assistance

Acknowledgements

- **Open Source Libraries**
    - React, Express, MongoDB, TensorFlow.js
    - Tailwind CSS, React Router, Axios

- **APIs and Services**
  - Spotify API for music integration
  - Cloudinary for image hosting

Contributing

Contributions are welcome! Please feel free to submit a Pull Request.

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes (`git commit -m 'Add some amazing feature'`)
4. Push to the branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

License

This project is licensed under the MIT License.