# **Lecture Video Tracker**

**Lecture Video Tracker Documentation**

**Overview**

The **Lecture Video Tracker** is a web application designed to track user engagement with lecture videos. It provides features such as user authentication, video progress tracking, and an admin dashboard for analytics. The project is divided into two main parts: the **client** (frontend) and the **server** (backend).

**Table of Contents**

1. Technologies Used
2. Project Structure
3. Features
4. Setup Instructions
5. API Endpoints
6. Frontend Components
7. Backend Components
8. Environment Variables
9. Screenshots

10. Future Enhancements

**Technologies Used**

**Frontend**

* **React**: For building the user interface.
* **Vite**: For fast development and build tooling.
* **Chart.js**: For rendering analytics charts.
* **Framer Motion**: For animations.
* **React Router**: For routing.

**Backend**

* **Node.js**: For server-side scripting.
* **Express.js**: For building RESTful APIs.
* **MongoDB**: For storing user and video progress data.
* **Mongoose**: For MongoDB object modeling.
* **JWT**: For user authentication.
* **bcrypt.js**: For password hashing.

**Project Structure**

**Client**

Located in d:\Lecture Video Tracker\client:

client/

|── public/ # Static assets (e.g., videos, images)

|── src/ # Source code

│ |── components/ # Reusable React components

│ |── pages/ # Page-level components

│ |── services/ # API service functions

│ |── App.jsx # Main application component

│ |── main.jsx # Entry point

│ |── index.css # Global styles

|── package.json # Frontend dependencies and scripts

|── vite.config.js # Vite configuration

└── index.html # HTML template

**Server**

Located in d:\Lecture Video Tracker\server:

server/

|── controllers/ # Business logic for routes

|──middleware/ # Middleware functions (e.g., authentication)

|── models/ # Mongoose schemas

|──routes/ # API route definitions

|── .env # Environment variables

|──server.js # Entry point for the backend

|── package.json # Backend dependencies and scripts

**Features**

**User Features**

* **Authentication**: Users can sign up, log in, and log out.
* **Video Progress Tracking**: Tracks watched intervals and calculates progress percentage.
* **Resume Playback**: Automatically resumes videos from the last watched position.

**Admin Features**

* **Analytics Dashboard**: Displays user engagement and active user trends using charts.

**Setup Instructions**

**Prerequisites**

* Node.js (v16+)
* MongoDB (local or cloud instance)

**Backend Setup**

1. Navigate to the [server](vscode-file://vscode-app/d:/Visua%20studio%20code/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) directory:

cd d:\Lecture Video Tracker\server

1. Install dependencies:

npm install

3. Create a [.env](vscode-file://vscode-app/d:/Visua%20studio%20code/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) file with the following variables:

MONGO\_URI=<your-mongodb-uri>

JWT\_SECRET=<your-jwt-secret>

4. Start the Server:

npm start

**Frontend Setup**

1. Navigate to the [client](vscode-file://vscode-app/d:/Visua%20studio%20code/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) directory:

cd d:\Lecture Video Tracker\client

2. Install dependencies:

npm install

1. Start the development server:

npm run dev

**API Endpoints**

**Authentication**

* **POST** /api/auth/signup: User signup.
* **POST** /api/auth/login: User login.

**Video Progress**

* **GET** /api/progress/get: Fetch video progress.
* **POST** /api/progress/save: Save video progress.

**Analytics**

* **GET** /api/analytics: Fetch analytics data (requires authentication).

**Frontend Components**

**Key Components**

1. **Login.jsx**: Handles user login and signup.
2. **Home.jsx**: Displays the video player and logout button.
3. **VideoPlayer.jsx**: Tracks and displays video progress.
4. **AdminDashboard.jsx**: Displays analytics charts for admins.
5. **ProtectedRoute.jsx**: Restricts access to authenticated users.

**Services**

* **authService.jsx**: Handles authentication-related API calls.
* **videoService.jsx**: Handles video progress-related API calls.

**Backend Components**

**Models**

1. **User.js**: Schema for user data.
2. **Progress.js**: Schema for video progress data.

**Controllers**

1. **authController.js**: Handles user signup and login.
2. **progressController.js**: Handles video progress saving and retrieval.
3. **analyticsController.js**: Provides analytics data.

**Middleware**

* **authMiddleware.js**: Verifies JWT tokens for protected routes.

**Environment Variables**

The backend requires the following environment variables in the [.env](vscode-file://vscode-app/d:/Visua%20studio%20code/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) file:

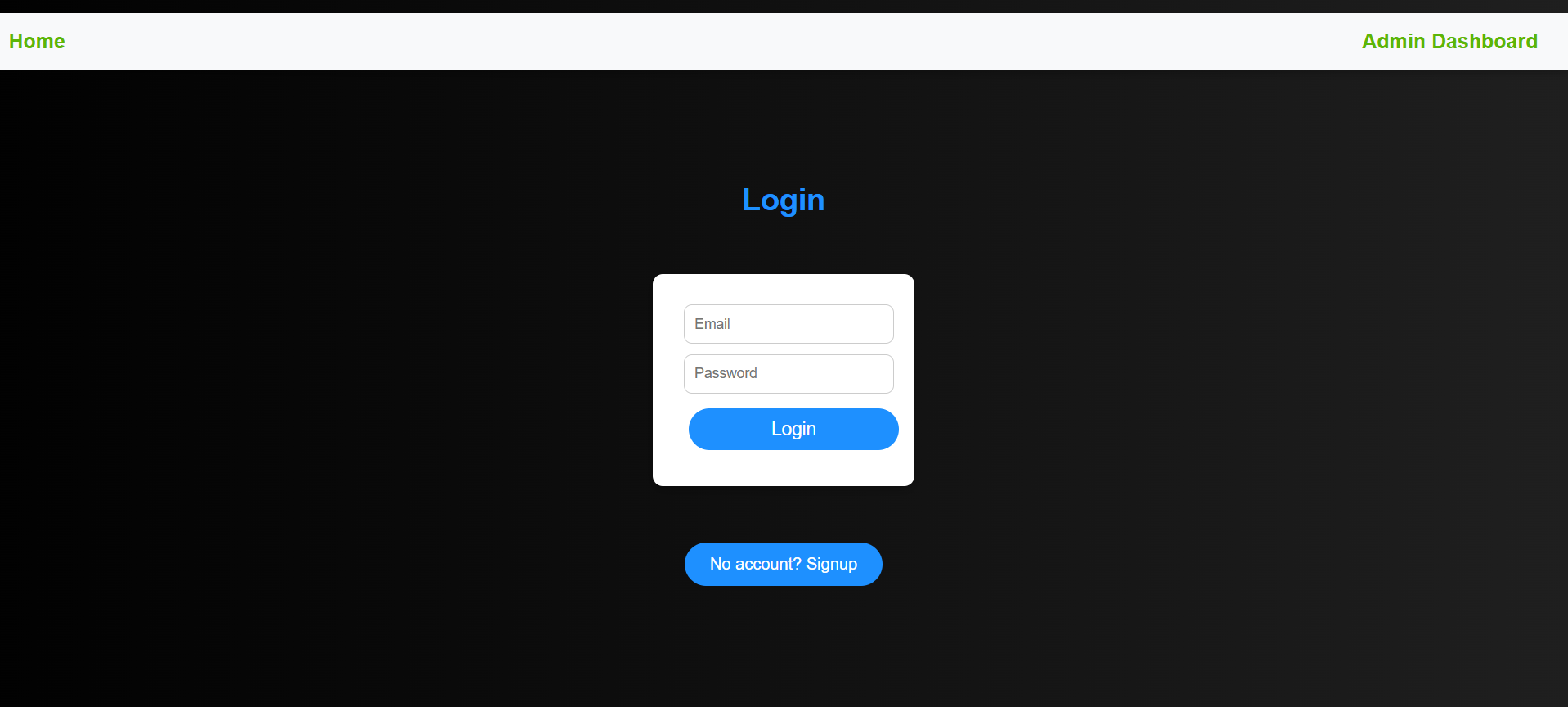
* [MONGO\_URI](vscode-file://vscode-app/d:/Visua%20studio%20code/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): MongoDB connection string.
* [JWT\_SECRET](vscode-file://vscode-app/d:/Visua%20studio%20code/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Secret key for JWT token generation.

**Future Enhancements**

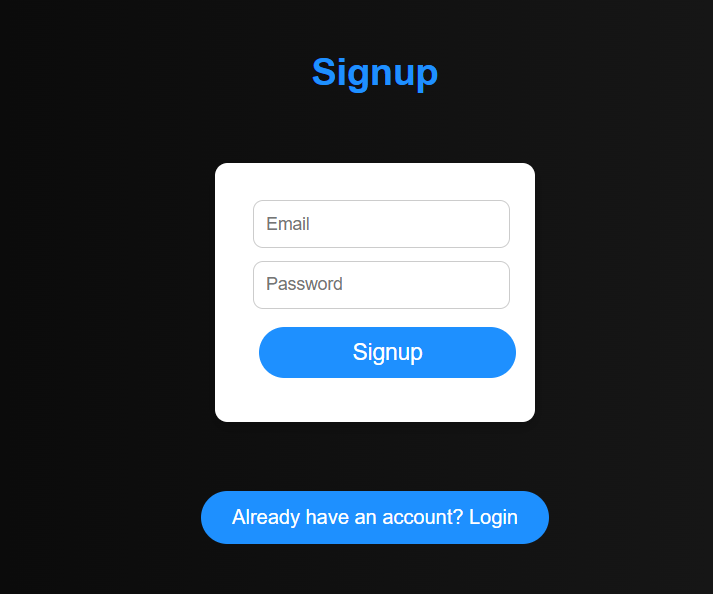
1. **User Profiles**: Add support for user profiles and preferences.
2. **Video Library**: Allow users to browse and select videos.
3. **Advanced Analytics**: Provide more detailed insights into user engagement.
4. **Mobile Support**: Optimize the UI for mobile devices.

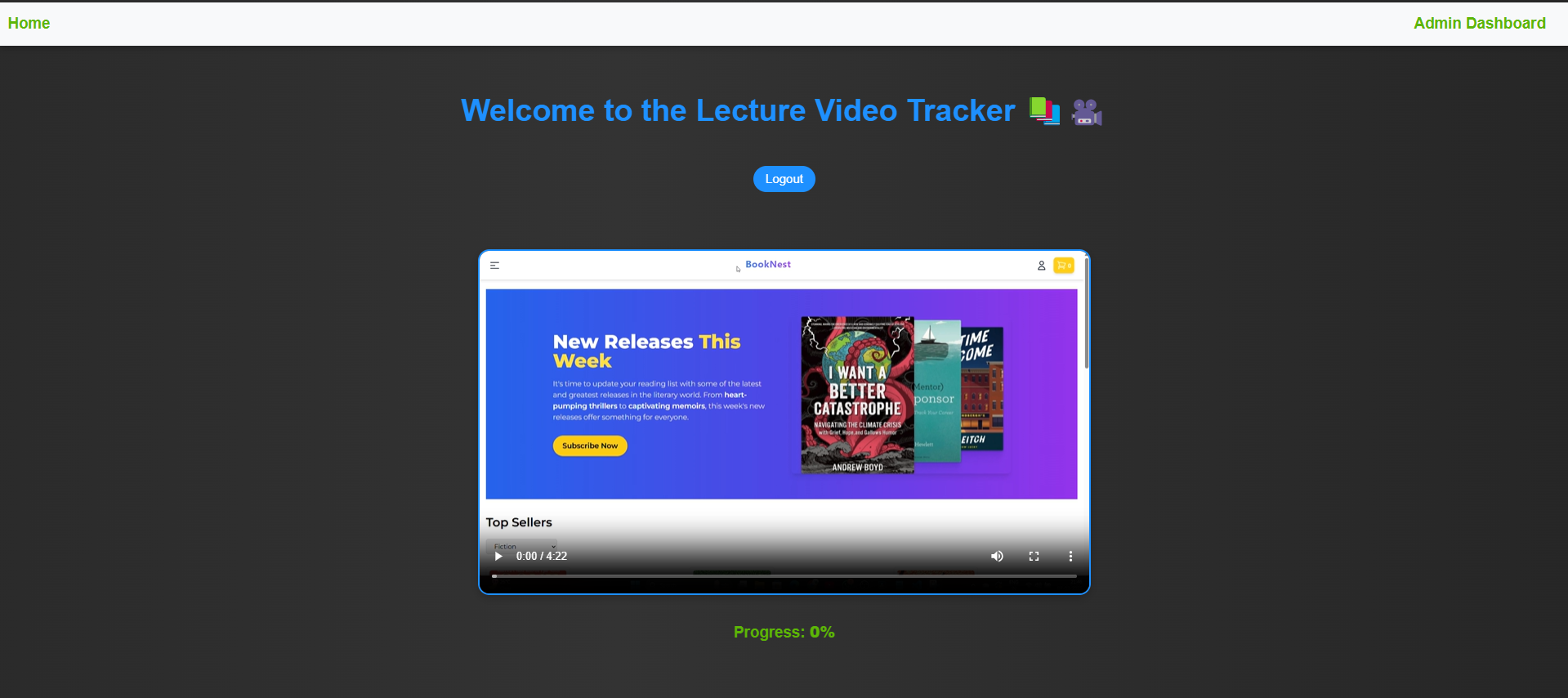
**Screenshots**

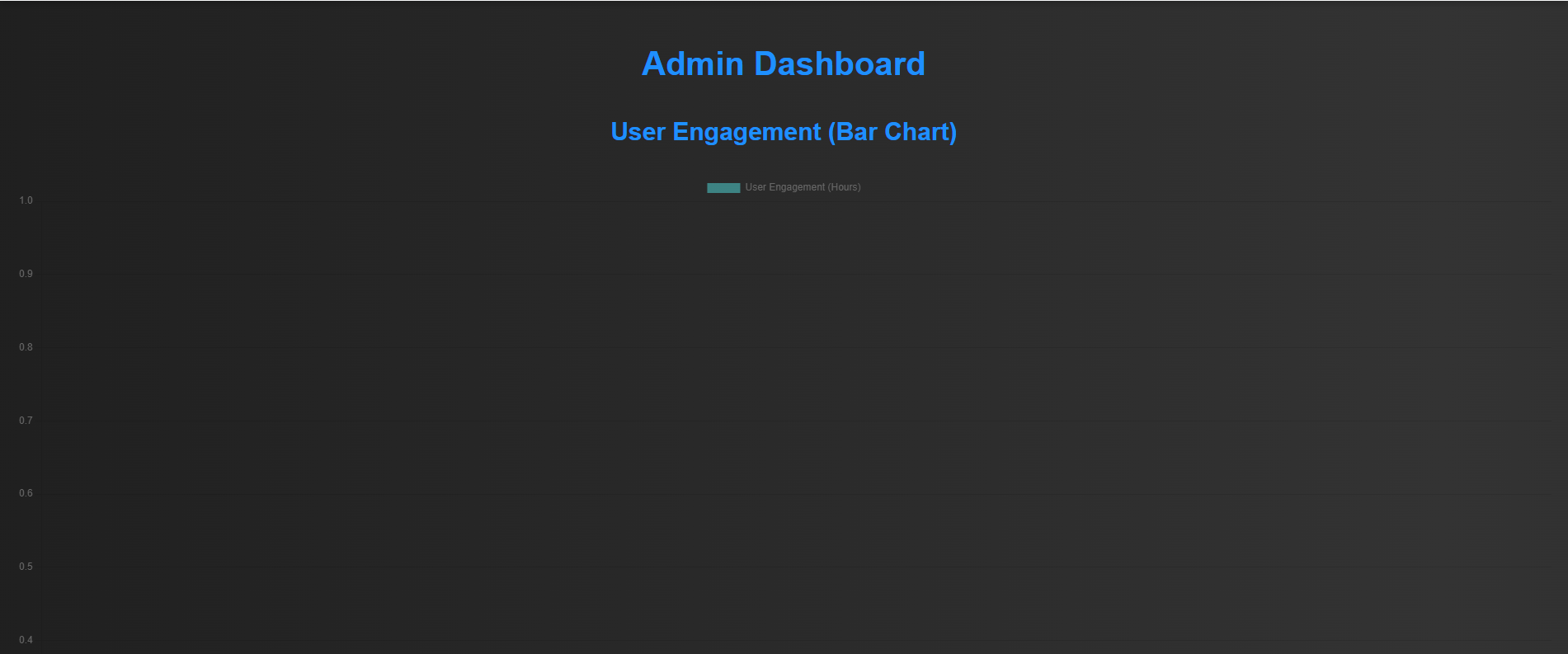
**Landing/Login page:**

****

**Signup page:**

****

**Video Player/Progress :  
**

**Admin Dashboard Page:  
**

**Demo video link:**[**https://drive.google.com/file/d/1xPPzNP8O6-0olVmbKD5tjWJ\_VOeS4Yi3/view?usp=sharing**](https://drive.google.com/file/d/1xPPzNP8O6-0olVmbKD5tjWJ_VOeS4Yi3/view?usp=sharing)

**GitHub Repository link:**[**https://github.com/Suryanarayana1269/Lecture-Video-Tracker**](https://github.com/Suryanarayana1269/Lecture-Video-Tracker)