## **Web Application Documentation**

## **Overview**

## This web application is built using Node.js and Express.js for the backend, MongoDB for database operations, JWT (JSON Web Tokens) for user authentication, and bcrypt for password hashing. It includes user signup, login, and main page routes with form validation and token-based authentication.

## **Installation and Setup**

## download the project files.

## Install Node.js and MongoDB if not already installed.

## Navigate to the project directory in the terminal.

## Install dependencies using the command:

# **npm install**

# **Dependencies**

# **Express.js:** Web application framework for Node.js.

# **Mongoose:** MongoDB object modeling for Node.js.

# **bcrypt:** Library for hashing passwords.

# **jsonwebtoken (JWT):** Token-based authentication.

# **cookie-parser:** Middleware for handling cookies.

# **Configuration**

# **secretKey:** Change the secret key for JWT token generation to a secure random string.

# **mongoose.connect():** Update the MongoDB connection string with your database URL.

# port: Set the port number for the server to listen on.

# **File Structure**

# **app.js:** Main Express.js application file containing routes and middleware.

# **models/user.js:** MongoDB schema and model for the User collection.

# **views/:** Directory containing EJS templates for rendering HTML pages.

# **Routes**

# **GET /:** Renders the signup form.

# **GET /login:** Renders the login form.

# **POST /signup:**

# Accepts form data (username, email, password).

# Hashes the password using bcrypt.

# Generates a JWT token and stores it in a cookie.

# Redirects to the main page.

# **POST /login:**

# Accepts login credentials (email, password).

# Compares hashed password with the stored hash using bcrypt.

# Generates a JWT token and stores it in a cookie upon successful login.

# Redirects to the main page.

# GET /main (Protected Route, requires valid JWT token):

# Verifies the JWT token from the cookie.

# Retrieves user data from MongoDB based on the token.

# Renders the main page with user data.

# **Middleware**

# express.urlencoded(): Middleware for parsing URL-encoded form data.

# express.static(): Serves static files from the public directory.

# express.json(): Middleware for parsing JSON data.

# cookie-parser: Middleware for parsing cookies.

# verifyToken: Custom middleware to verify JWT tokens for protected routes.

# **Security Measures**

# Passwords are hashed using bcrypt before storing in the database.

# JWT tokens are signed and stored in HTTP-only cookies for enhanced security.

# Token expiration (expiresIn) is set to limit token validity duration.

# **Notes**

# Ensure to replace placeholders (like secretKey and MongoDB connection string) with actual values.

# Implement additional security measures like HTTPS for production deployments.

This is the link of the Website implemented in Hack the Tank 2.0(Finalist)