Data Analytics with Tableau

1. Introduction:

• Project Title: Heritage Treasures: An In-Depth Analysis of UNESCO World Heritage Sites in Tableau

• Team Members:

1. A.M. Dheekshitha –ideation phase

--requirement analysis

--project planning phase

2.K.Bhanu Prakash —project design phase

--performance testing

2. Project Overview:

### 1. Purpose:

### To provide a comprehensive, interactive dashboard in Tableau that showcases global heritage data, enabling users to discover, compare, and analyze cultural and natural sites recognized by UNESCO.

**> Goals:**

* To **visualize** the distribution of World Heritage Sites by region, country, and classification (cultural, natural, mixed).
* To **identify trends** over time such as the number of sites added per decade.
* To **highlight at-risk sites** and their conservation status.
* To **enable comparisons** of heritage richness between countries and continents.
* To **raise awareness** about the importance of preserving global heritage through accessible data storytelling.

**2.Features:**

* Interactive Tableau dashboards showcasing UNESCO World Heritage Site data
* Filters by country, continent, category, and year of inscription
* Detailed statistics (e.g., cultural vs. natural sites, endangered sites)
* User authentication for saving dashboard views
* Admin panel to upload or update data
* API access for third-party integration

3.Architecture:

#### • Frontend (React)

* Built using React JS with functional components
* React Router for client-side routing
* Axios for API communication
* Tableau JS API embedded for dashboard interactivity
* Context API for user state management

#### • Backend (Node.js + Express.js)

* REST full API architecture
* Express router modules for users, sites, and analytics
* JWT-based authentication
* Middleware for error handling and request validation

#### • Database (Mongo DB)

* Collections:
  + users: Stores user profiles and roles
  + sites: UNESCO site data (name, location, criteria, status)
  + views: User-saved dashboard states
* Mongoose ODM for schema and model definitions
* Aggregation queries for analysis

4.Setup Instructions:

#### • Prerequisites

* Node.js (v18+)
* MongoDB (local or Atlas)
* Git
* Tableau Public/Server access (optional for demo)

#### • Installation

bash

CopyEdit

# Clone the repository

git clone https://github.com/username/heritage-treasures.git

cd heritage-treasures

# Install dependencies

cd server

npm install

cd ../client

npm install

# Setup environment variables

# In server/.env

PORT=5000

MONGO\_URI=mongodb://localhos

**5.Folder Structure:**

**->Client:**

/client

│

├── /src

│ ├── /components # Reusable UI components

│ ├── /pages # Dashboard, Login, Admin

│ ├── /services # API calls via Axios

│ ├── /context # Auth provider

│ └── /App.js

│

├── /public

└── package.json

**🡪server:**

/server

│

├── /controllers # Route logic

├── /models # Mongoose schemas

├── /routes # API endpoints

├── /middleware # Auth and error handlers

├── /utils # Helper functions

├── server.js

└── package.json

6. Running the Application:

**• Frontend:**

cd client

npm start

**• Backend:**

cd server

npm start

7. API Documentation:

| **Endpoint** | | | **Method** | **Description** |
| --- | --- | --- | --- | --- |
| /api/auth/register | | | POST | Register a new user |
| /api/auth/login | | | POST | Login and receive token |
| /api/sites | | | GET | Fetch all heritage sites |
| /api/sites/:id | | | GET | Get site by ID |
| /api/views/save | | | POST | Save a dashboard state (auth needed) |
| 7. ****Authentication****  * **Method**: JWT (JSON Web Token) * **Flow**:   + On login, server returns a signed JWT token   + Client stores token in localStorage   + Token attached to protected API calls via Authorization header * **Authorization**: Role-based access for Admin and Users  8. ****User Interface****  * Clean, responsive UI with data visualizations * Filter panel (country, year, type) * Tableau iframe embedding * Light/dark mode toggle * User login modal   **9. Testing**   * **Frontend**: Jest + React Testing Library * **Backend**: Mocha + Chai + Supertest * **Coverage**: API route tests, component rendering, and form validation   **10. Screenshots / Demo**   * Demo Tableau Dashboard     **11. Known Issues**   * Tableau Public iframe embedding may not work on some browsers without third-party cookie access * Limited search/filter performance on large datasets * No email verification for registration (planned)   **12. Future Enhancements**   * Add support for multi-language dashboards * Offline export of Tableau views as PDF * Real-time data sync with UNESCO API (if available) * Role-based analytics (e.g., per country admin) | | |  |  |
|  |  |