Travel & Tourism Platform — Full Documentation (Step-by-Step Setup)

This document contains **everything** from the beginning: idea, tech stack, installations, tools, architecture, features, phases, and next steps. Nothing is missed.

✅1. Project Concept & Purpose

A dedicated **Travel & Tourism Community Platform** where users can: - Share travel experiences & incidents - Post mistakes, safety tips, do's & don'ts - Share routes, budgets, packing lists - Add destination guides - Interact through likes & comments - Follow other travelers

---

✅2. Features List

User Features

- Register & Login
- Profile page
- Add travel story
- Upload photos/videos
- Comments & likes
- Save/bookmark posts
- Search destinations

Content Types

- Travel stories
- Incident experiences
- Destination guides
- Packing checklist
- Budget breakdown
- Safety tips

Admin Features

- Approve/remove content
- Manage users
- Manage destinations
- View analytics

✅3. Complete System Architecture

Frontend (User Interface)

- React.js
- Tailwind CSS
- React Router
- Axios

Backend (Server)

- Node.js
- Express.js
- JWT Authentication
- Multer (Image upload)
- Bcrypt (Password hashing)

Database

- MongoDB Atlas (recommended)

Storage

- AWS S3 or Cloudinary

Hosting

- Frontend → Vercel / Netlify
- Backend → Render / AWS EC2
- Database → MongoDB Atlas

✅4. Installations Needed

Install these tools before starting development.

1. Node.js

For React + Backend. - Website: nodejs.org - After install: `node -v`, `npm -v`

2. VS Code

Coding editor. Extensions: - ES7 React Snippets - Prettier - Tailwind CSS IntelliSense - Material Icon Theme

3. Git

Version control. - Website: git-scm.com - Check: `git --version`

4. MongoDB

Two options: - **MongoDB Atlas (cloud)** → Recommended - **MongoDB Community Server (local)**

5. Postman

To test backend APIs. Website: postman.com

6. Browser: Chrome

Recommended for debugging.

✅5. Folder Structure

We'll create:

```
travel-platform/
    backend/
    frontend/
```

✅6. Backend Setup Steps

Inside `backend/` folder:

Step 1: Initialize Node

```
npm init -y
```

Step 2: Install Needed Packages

```
npm install express mongoose cors dotenv multer jsonwebtoken bcrypt
```

Step 3: Create Files

```
backend/
  server.js
  .env
  /models
  /routes
  /middleware
  /uploads
```

Step 4: Backend Components

- server.js (main file)
- User model
- Story model
- Auth routes
- Story routes
- Auth middleware
- Multer config for image upload

✅7. Frontend Setup Steps

Inside `frontend/` folder:

Step 1: Create React App

```
npx create-react-app frontend
```

Step 2: Install Dependencies

```
npm install axios react-router-dom
```

Step 3: Install Tailwind CSS

```
npm install -D tailwindcss
npx tailwindcss init
```

Frontend Pages

- HomePage
- LoginPage

- RegisterPage
- AddStoryPage
- StoryPage
- ProfilePage
- ExplorePage

✅8. Integration Steps

Connect frontend with backend:

- Axios baseURL set
- API routes create
- Form submission for adding story
- Fetch & display stories

✅9. Deployment Plan

Frontend → Netlify / Vercel

- Connect GitHub repo
- Automatic build

Backend → Render

- Deploy Node server
- Add environment variables

Database → MongoDB Atlas

- Free cluster
- Connection string inside `.env`

✅10. Development Roadmap (Phase-wise)

PHASE 1 — Planning

- Requirements
- UI wireframe
- Database schema

PHASE 2 — Backend

- Auth system
- Story posting
- Image upload
- Comments API

PHASE 3 — Frontend

- Login/Register UI
- Add story UI
- Display stories
- Story details page

PHASE 4 — Testing

- API tests via Postman
- UI testing

PHASE 5 — Deployment

- Final hosting

✅11. What We Will Build First

1. Backend Auth System
2. Frontend Login/Register
3. Add Story Feature
4. Story Feed Page

After that: Comments → Likes → Profile → Followers → Filters → Tags.

✅12. Your Action Before Starting Coding

Install the following: - Node.js - VS Code - Git - Postman - MongoDB (Atlas is best)

Once done, say: **"Installation complete, start backend setup"**