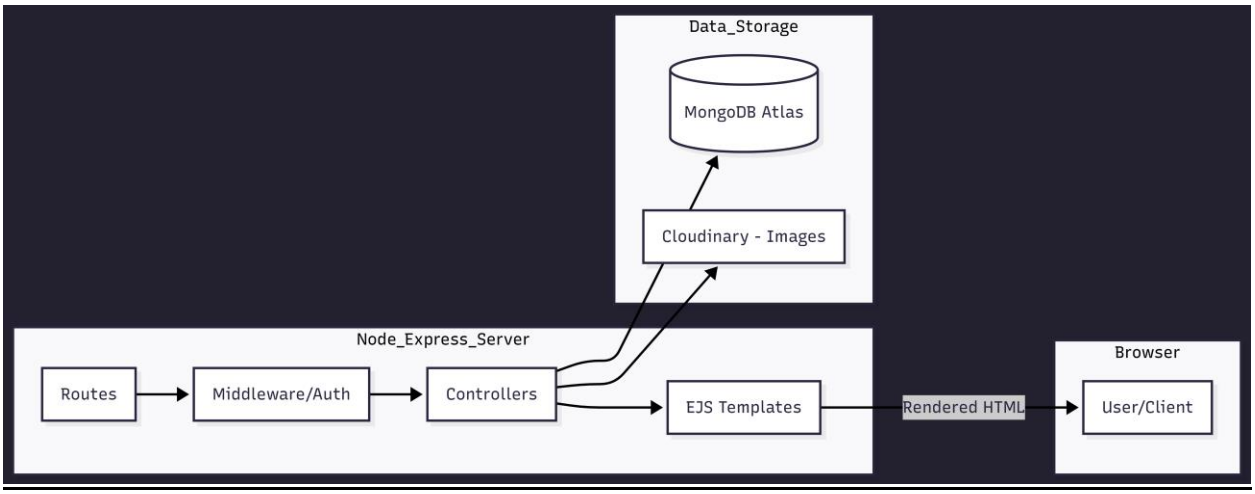
🏠 StayNest

Developed **StayNest**, a full-stack rental marketplace built on the MERN stack. I focused on implementing a complete CRUD lifecycle for property listings, integrated with Cloudinary for cloud image hosting and Passport.js for secure user sessions. I prioritized a modular MVC architecture to ensure system scalability and clean data flow between the Node/Express backend and MongoDB Atlas.

Built using Node.js, Express.js, MongoDB, and EJS.

Production-Grade Architecture Flow



Engineering Highlights

- **Robust Data Modeling:** Designed a normalized schema using **Mongoose** to manage complex relationships between Users, Listings, and Reviews.

- **Secure Authentication:** Implemented **Passport.js** for session-based authentication, ensuring protected routes for property management and user data privacy.

- **Asset Management:** Integrated **Multer** and **Cloudinary** for an optimized image upload pipeline, reducing server-side storage load by offloading media to the cloud.

- **MVC Architecture:** Followed the **Model-View-Controller** pattern to ensure the codebase remains maintainable, modular, and scalable.

Technical Tradeoffs

- **Database Choice:** Selected **MongoDB** for its flexible schema, allowing for rapid iteration of property attributes without complex migrations.

- **Template Engine:** Used **EJS** for server-side rendering to prioritize SEO-friendly property pages and faster initial load times for listing details.

Output Demo:

https://github.com/user-attachments/assets/71f1c8a0-1a79-465e-bb10-8533bdbad419

Project Status

This project is now finished. 🎉

⚡ Note: This project was built as part of the Apna College Delta Web Development Course. The main goal was to practice full-stack development concepts. I extended/customized some parts to improve my learning.

Features

🔐 User registration & login

🏘️ View all property listings

📝 Create, update, and delete listings (CRUD)

📸 Upload property images (via Multer)

🔒 Authentication & session management (via Passport.js)

Lessons Learned/Challenge

Optimized MongoDB aggregation pipelines and implemented indexing on property categories, ensuring <200ms response times for high-traffic listing pages.

Architected a normalized Mongoose schema to maintain data integrity across 10+ concurrent relationships (Users, Listings, Reviews), reducing data redundancy and API payload size

Author:

Jaya Rani.Y.S

Github: https://github.com/jayalloyd/StayNest