Backend Configuration

1. Rename .env.example to .env (use .env not .env.local it's only for backend).

2. Configure your MongoDB database, watch this video MongoDB, aer configuring you will find a mongo URI just put that on your .env file MONGO_URI variable.

3. The JWT_SECRET is just a random value for creating a secret token, you can use whatever you want but make sure it is secret.

4. You need an email and password for using email verification and forget the password opon. Use an email that you want to send messages to others when they register or request to forget the password. We use Nodemailer and the default email server for this. watch this video to create an app password for email app-password. aer that put your email in .env file EMAIL_USER and app password in the EMAIL_PASS variable.

Also, need to **Allow less secure apps to be ON, and access captcha for using in production environment, see this doc**

4. Use your local server URL in ADMIN_URL variable, when you run on the local server your URL will be http://localhost:3000,

Finally, your .env file will look like this:

PORT=5055
MONGO_URI=your mongodb uri
JWT_SECRET=alamsfdfsdfsdfsdrafdar!@#$0fdsfdsfdsfds
JWT_SECRET_FOR_VERIFY=lasjfr09ri09wrilfdjdj

SERVICE=gmail
EMAIL_USER=your email //change with your sender email
EMAIL_PASS=you email app password //change with your email app password
HOST=smtp.gmail.com
EMAIL_PORT=465

//use this when in dev/local server but when you will run on production/ live server then use that URL/domain in here and put that live URL on environment variable when using this backend

ADMIN_URL= http://localhost:3000

Once you successfully connect with MongoDB and configured .env then run "npm run data:import", it will run seed.js file and will import all demo data on the database. (You will find all demo data in the utils folder, change that data according to your need, also use staff email with real email for use forget password option) If everything is okay, then the backend configuration is done. Now you will find all demo data in your MongoDB database.

Admin Configuration

1. Rename .env.example to .env.local

2. Please watch this video for Cloudinary configuration Cloudinary configuration, (We use Cloudinary for image upload).

After Configure your .env.local file will look like this:

REACT_APP_API_BASE_URL=http://localhost:5055/api

REACT_APP_CLOUDINARY_URL=https://api.cloudinary.com/v1_1/your-cloudinary-user-name/image/upload

REACT_APP_CLOUDINARY_UPLOAD_PRESET=fg1vfge //your cloudinary upload preset

Deploy On Vercel

Here is your guide for deploying Dashtar on vercel:

1. Create a GitHub account, go to vercel and sign up with that GitHub

account.

2. Create two private repositories on GitHub, then push your backend code in one, and admin code in another repository.

3. Watch this video deploy on vercel, do according to.

4. When you import your GitHub repository on vercel by creating a project, you will see an opon for Environment Variables, just click on that and give you a local .env all variable with the value. then click on deploy.

Important :: First you have to deploy a backend project.

5. After the backend is deployed successfully, you will find a URL for your API route that will like this https://dashtar.vercel.app/, and now change that like this https://dashtar.vercel.app/api and use this as a REACT_APP_API_BASE_URL when you deploy your admin project.

6. Now create another project for admin deploy, same as backend project and put all .env.local variables before clicking on deploy button, then click deploy, it will take some time for build and after that build, you will see your live version of Dashtar admin.

7. If you do accordingly, then everything will be okay, for now when you make any changes on your local file, you just need to push your code on GitHub, vercel will automatically detect those changes and will redeploy your project with updated features.

You will find many videos on youtube and also articles on google about how to deploy React.js and express apps on vercel.