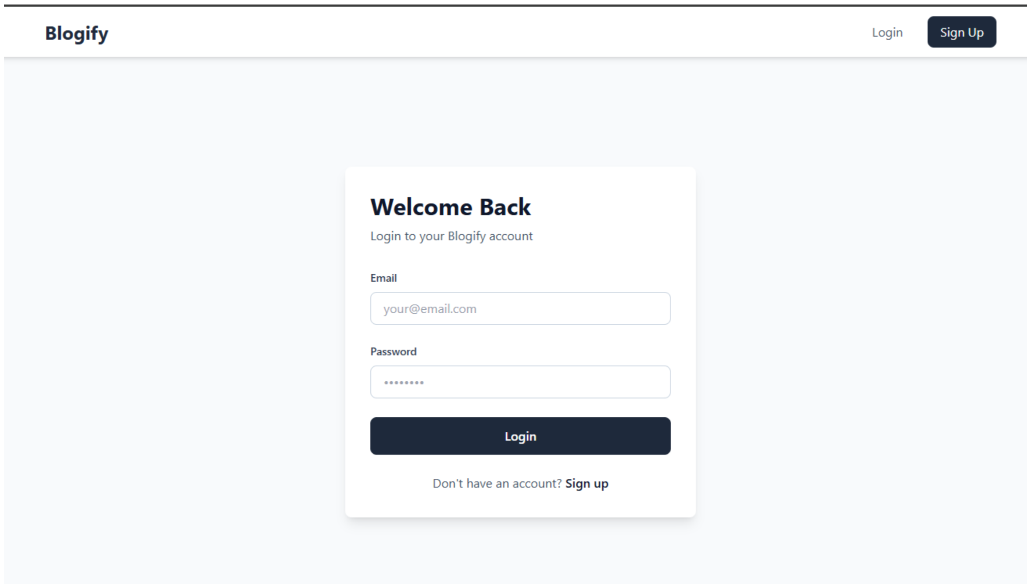

Blogify

Login

Sign Up

Welcome Back

Login to your Blogify account

Email

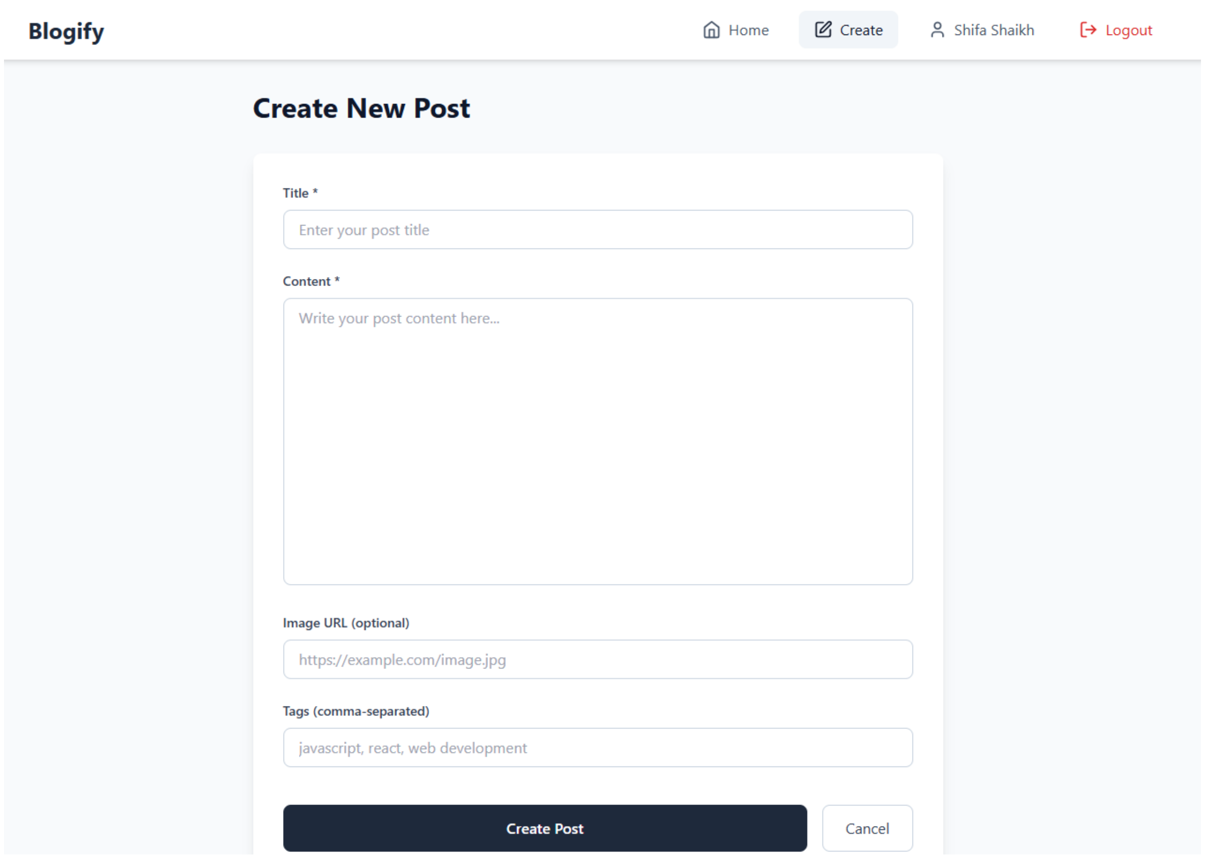
your@email.com

Password

••••••••

Login

Don't have an account? **Sign up**


Blogify

Home

Create

Shifa Shaikh

Logout

Create New Post

Title *

Enter your post title

Content *

Write your post content here...

Image URL (optional)

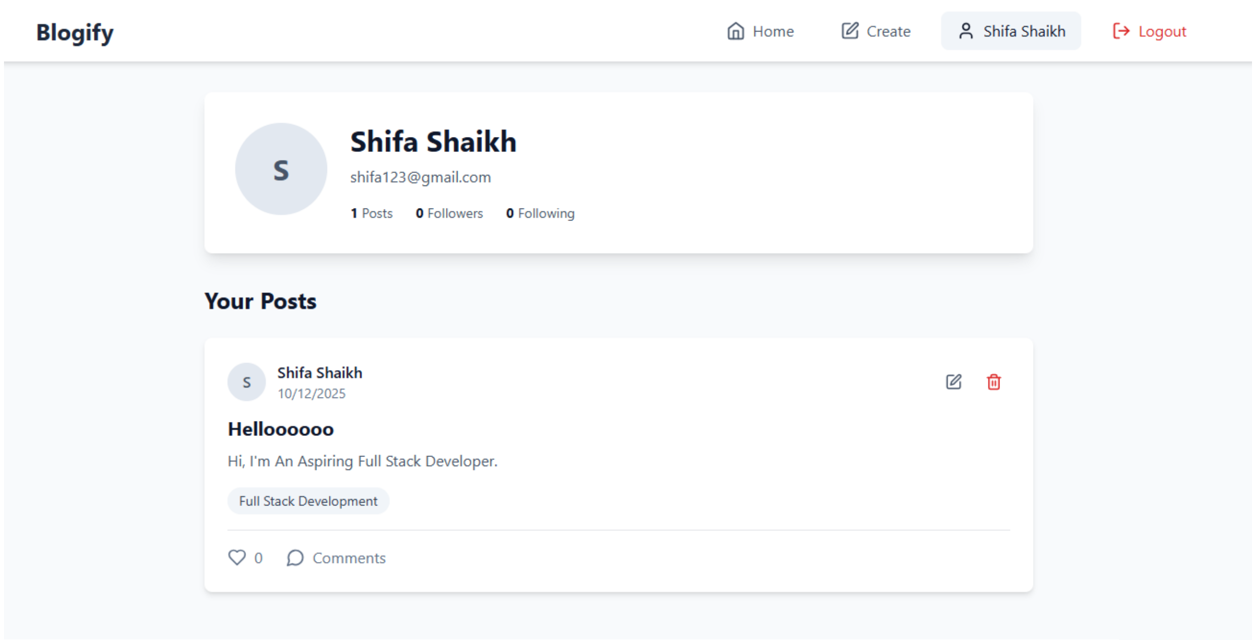
https://example.com/image.jpg

Tags (comma-separated)

javascript, react, web development

Create Post

Cancel



S

Shifa Shaikh

shifa123@gmail.com

1 Posts **0** Followers **0** Following

Your Posts

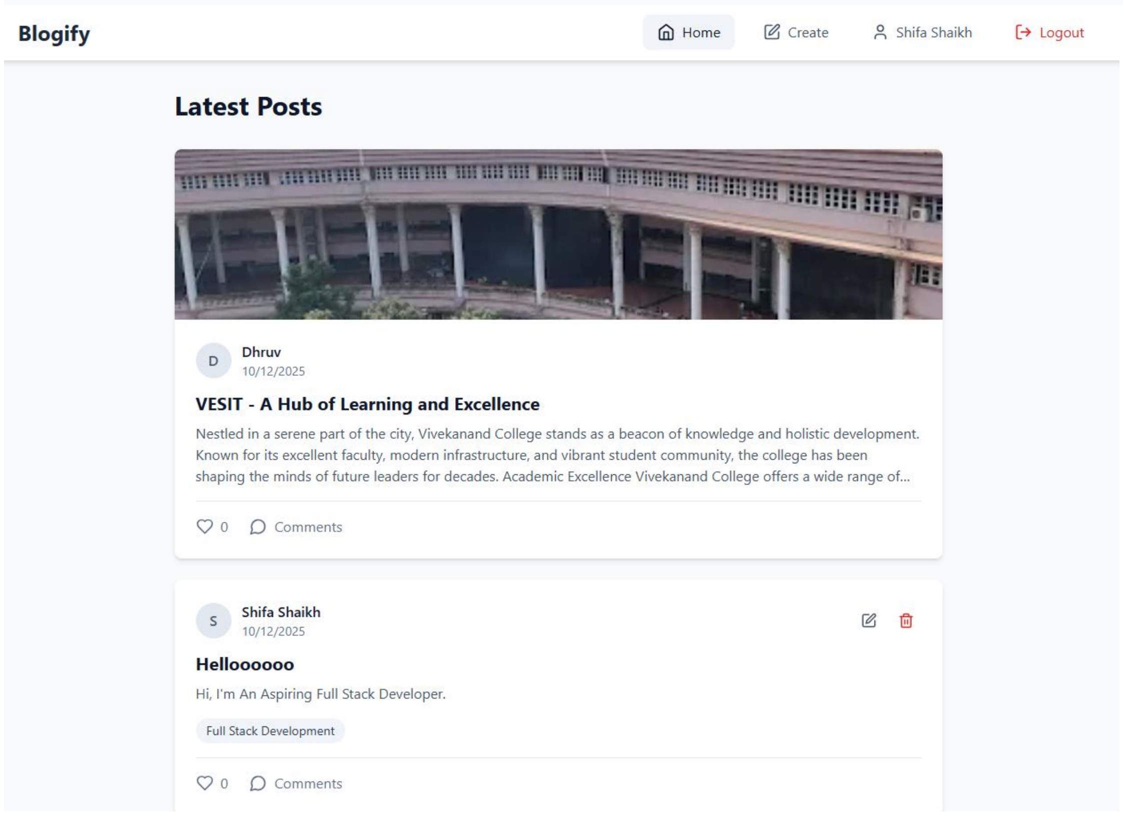
S Shifa Shaikh
10/12/2025

Hellooooo

Hi, I'm An Aspiring Full Stack Developer.

Full Stack Development

0 Comments



Latest Posts

D Dhruv
10/12/2025

VESIT - A Hub of Learning and Excellence

Nestled in a serene part of the city, Vivekanand College stands as a beacon of knowledge and holistic development. Known for its excellent faculty, modern infrastructure, and vibrant student community, the college has been shaping the minds of future leaders for decades. Academic Excellence Vivekanand College offers a wide range of...

0 Comments

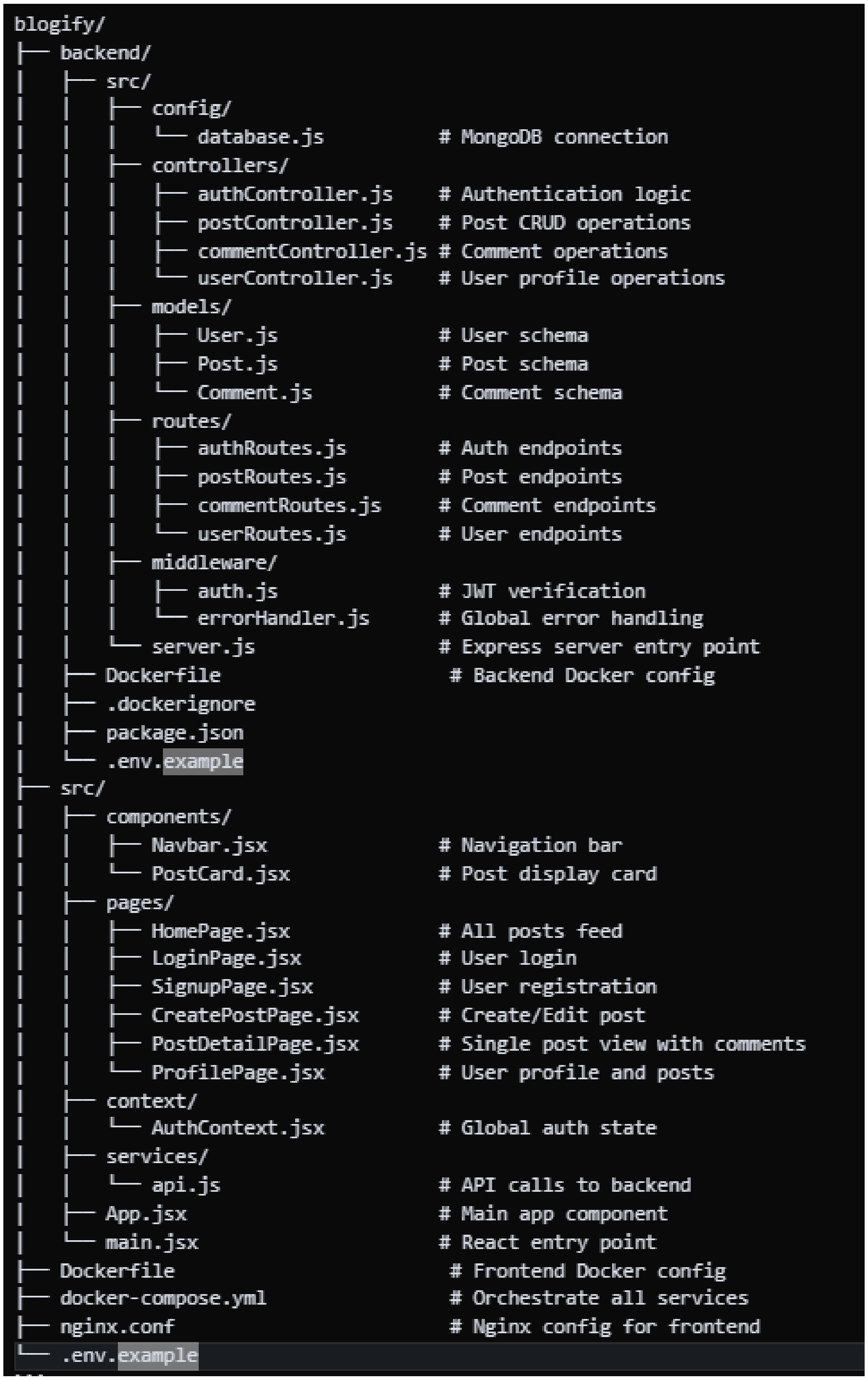
S Shifa Shaikh
10/12/2025

Hellooooo

Hi, I'm An Aspiring Full Stack Developer.

Full Stack Development

0 Comments



```
blogify/
├── backend/
│   ├── src/
│   │   ├── config/
│   │   │   └── database.js          # MongoDB connection
│   │   ├── controllers/
│   │   │   ├── authController.js    # Authentication logic
│   │   │   ├── postController.js    # Post CRUD operations
│   │   │   ├── commentController.js # Comment operations
│   │   │   └── userController.js    # User profile operations
│   │   ├── models/
│   │   │   ├── User.js              # User schema
│   │   │   ├── Post.js              # Post schema
│   │   │   └── Comment.js           # Comment schema
│   │   ├── routes/
│   │   │   ├── authRoutes.js        # Auth endpoints
│   │   │   ├── postRoutes.js        # Post endpoints
│   │   │   ├── commentRoutes.js     # Comment endpoints
│   │   │   └── userRoutes.js        # User endpoints
│   │   ├── middleware/
│   │   │   ├── auth.js              # JWT verification
│   │   │   └── errorHandler.js      # Global error handling
│   │   └── server.js                # Express server entry point
│   ├── Dockerfile                    # Backend Docker config
│   ├── .dockerignore
│   ├── package.json
│   └── .env.example
├── src/
│   ├── components/
│   │   ├── Navbar.jsx               # Navigation bar
│   │   └── PostCard.jsx             # Post display card
│   ├── pages/
│   │   ├── HomePage.jsx             # All posts feed
│   │   ├── LoginPage.jsx            # User login
│   │   ├── SignupPage.jsx           # User registration
│   │   ├── CreatePostPage.jsx       # Create/Edit post
│   │   ├── PostDetailPage.jsx       # Single post view with comments
│   │   └── ProfilePage.jsx          # User profile and posts
│   ├── context/
│   │   └── AuthContext.jsx          # Global auth state
│   ├── services/
│   │   └── api.js                   # API calls to backend
│   ├── App.jsx                      # Main app component
│   └── main.jsx                     # React entry point
├── Dockerfile                        # Frontend Docker config
├── docker-compose.yml                # Orchestrate all services
├── nginx.conf                        # Nginx config for frontend
└── .env.example
```

```
```

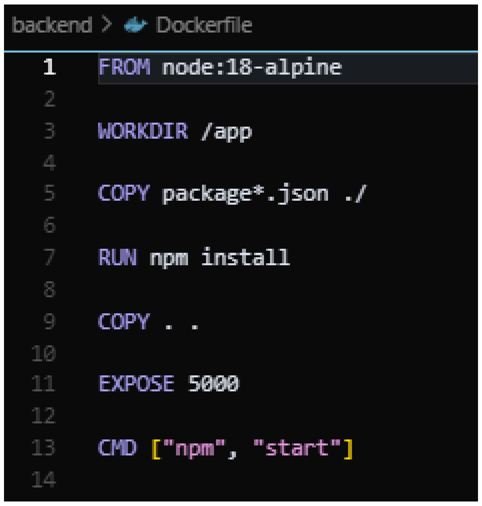
Step 1: Backend Dockerfile

Location: `backend/Dockerfile`

Code:

```
FROM node:18
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 5000
CMD ["npm", "run", "dev"]
```

Explanation: - Base image, working directory, install dependencies, expose port, start server.



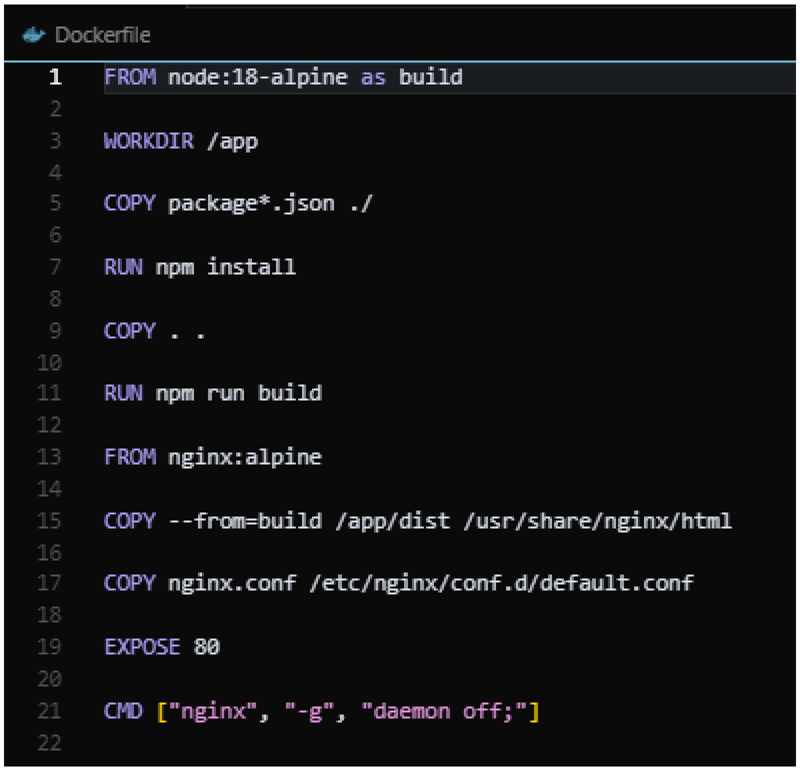
```
FROM node:18-alpine

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 5000

CMD ["npm", "start"]

```

Step 2: Frontend Dockerfile

Location: `frontend/Dockerfile`

Code:

```
FROM node:18
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 5173
CMD ["npm", "run", "dev"]
```



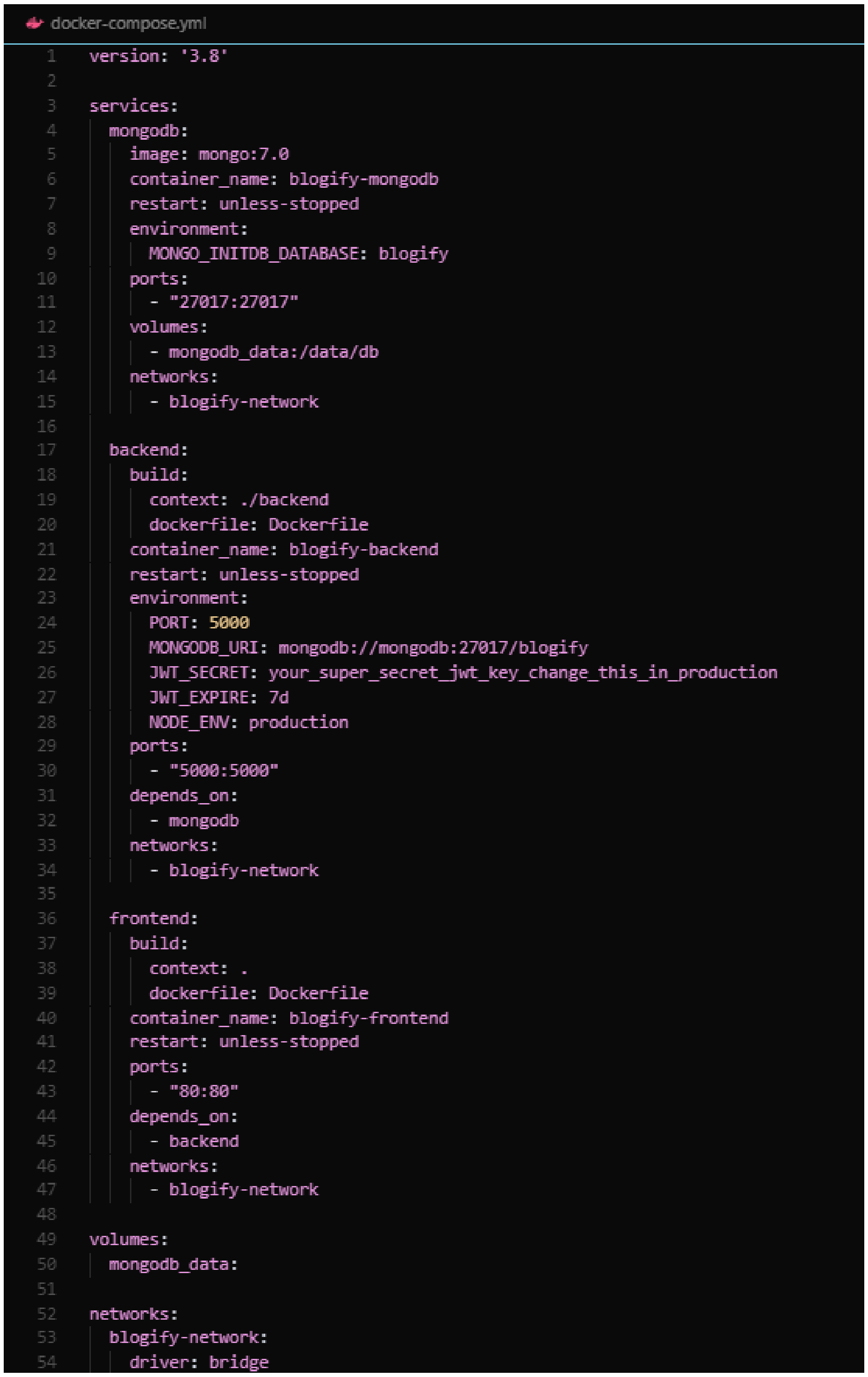
```
FROM node:18-alpine as build

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build

FROM nginx:alpine

COPY --from=build /app/dist /usr/share/nginx/html

COPY nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

Step 3: Docker Compose

Location: `docker-compose.yml`

Code:

```
version: '3.8'

services:
 mongodb:
  image: mongo:7.0
  container_name: blogify-mongodb
  restart: unless-stopped
  environment:
   MONGO_INITDB_DATABASE: blogify
  ports:
   - "27017:27017"
  volumes:
   - mongodb_data:/data/db
  networks:
    - blogify-network

  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    container_name: blogify-backend
    restart: unless-stopped
    environment:
      PORT: 5000
      MONGODB_URI: mongodb://mongodb:27017/blogify
      JWT_SECRET: your_super_secret_jwt_key_change_this_in_production
      JWT_EXPIRE: 7d
      NODE_ENV: production
    ports:
      - "5000:5000"
    depends_on:
      - mongodb
    networks:
      - blogify-network

  frontend:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: blogify-frontend
    restart: unless-stopped
    ports:
      - "80:80"
    depends_on:
      - backend
    networks:
      - blogify-network

volumes:
  mongodb_data:

networks:
  blogify-network:
    driver: bridge
```



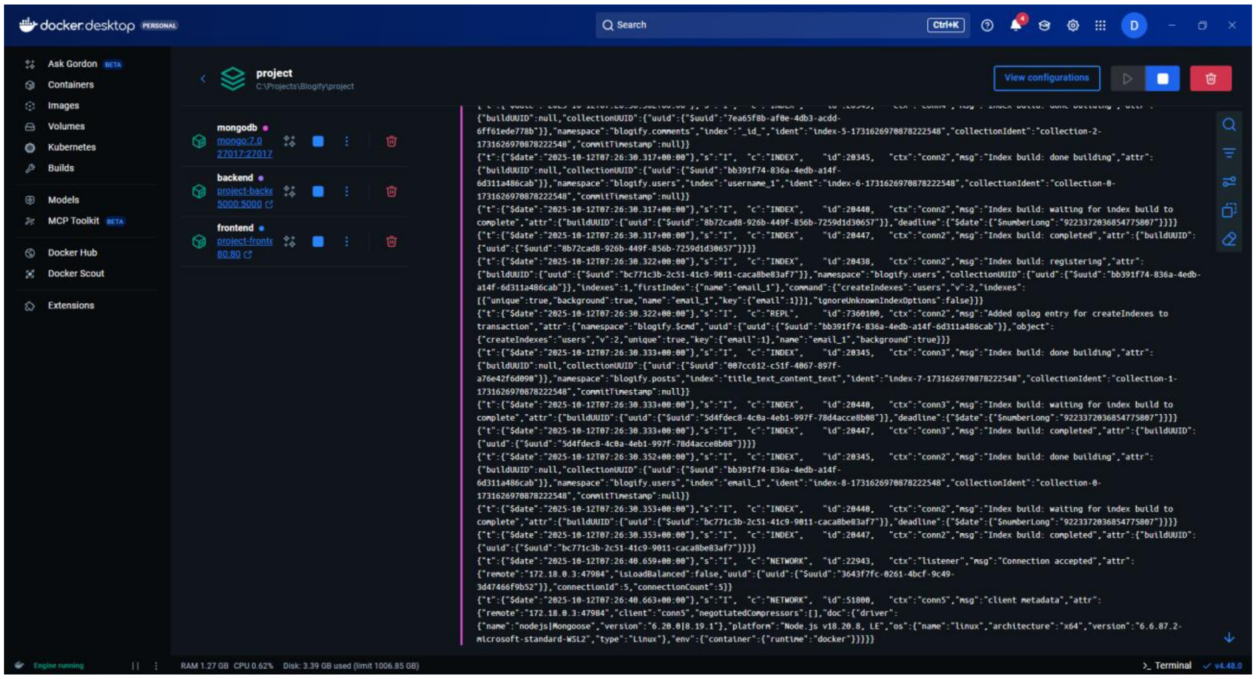
docker-compose.yml

```yaml
version: '3.8'

services:
  mongodb:
    image: mongo:7.0
    container_name: blogify-mongodb
    restart: unless-stopped
    environment:
      MONGO_INITDB_DATABASE: blogify
    ports:
      - "27017:27017"
    volumes:
      - mongodb_data:/data/db
    networks:
      - blogify-network

  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    container_name: blogify-backend
    restart: unless-stopped
    environment:
      PORT: 5000
      MONGODB_URI: mongodb://mongodb:27017/blogify
      JWT_SECRET: your_super_secret_jwt_key_change_this_in_production
      JWT_EXPIRE: 7d
      NODE_ENV: production
    ports:
      - "5000:5000"
    depends_on:
      - mongodb
    networks:
      - blogify-network

  frontend:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: blogify-frontend
    restart: unless-stopped
    ports:
      - "80:80"
    depends_on:
      - backend
    networks:
      - blogify-network

volumes:
  mongodb_data:

networks:
  blogify-network:
    driver: bridge
```

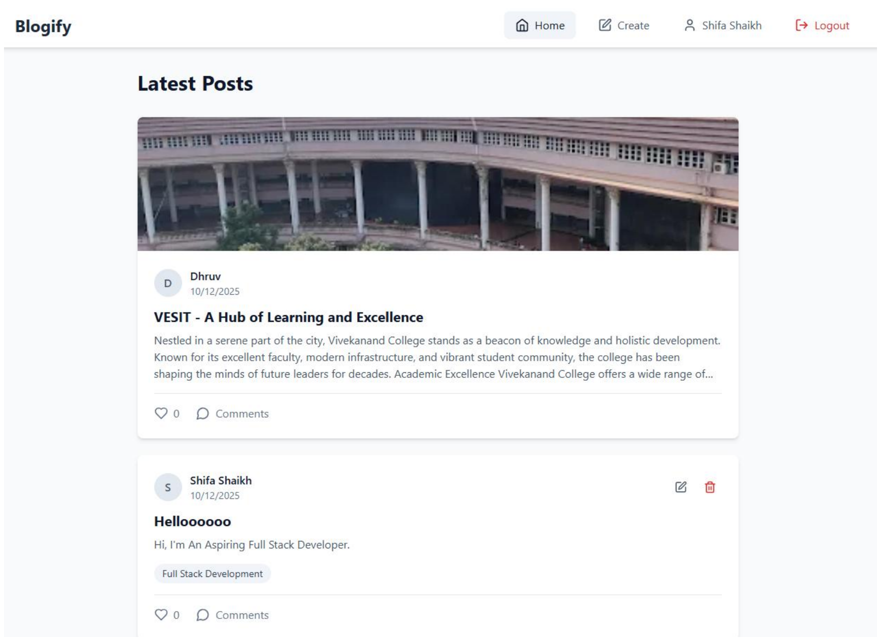
Step 4: Build and Run Containers

Commands:

```
docker-compose up --build
# OR to run in background
docker-compose up -d
# Check running containers
docker ps
```



```
{"buildUUID":null,"collectionUUID":{"uuid":{"$uuid":"7ea65f8b-af0e-4db3-acdd-
6ff61ede778b"}},"namespace":"blogify.comments","index":"_id_","ident":"index-5-1731626970878222548","collectionIdent":"collection-2-
1731626970878222548","commitTimestamp":null}}
{"t":{"$date":"2025-10-12T07:26:30.317+00:00"},"s":"I",  "c":"INDEX",    "id":20345,   "ctx":"conn2","msg":"Index build: done building","attr":
{"buildUUID":null,"collectionUUID":{"uuid":{"$uuid":"bb391f74-836a-4edb-a14f-
6d311a486cab"}},"namespace":"blogify.users","index":"username_1","ident":"index-6-1731626970878222548","collectionIdent":"collection-0-
1731626970878222548","commitTimestamp":null}}
{"t":{"$date":"2025-10-12T07:26:30.317+00:00"},"s":"I",  "c":"INDEX",    "id":20440,   "ctx":"conn2","msg":"Index build: waiting for index build to
complete","attr":{"buildUUID":{"uuid":{"$uuid":"8b72cad8-926b-449f-856b-7259d1d30657"}},"deadline":{"$date":{"$numberLong":"9223372036854775807"}}}}
{"t":{"$date":"2025-10-12T07:26:30.317+00:00"},"s":"I",  "c":"INDEX",    "id":20447,   "ctx":"conn2","msg":"Index build: completed","attr":{"buildUUID":
{"uuid":{"$uuid":"8b72cad8-926b-449f-856b-7259d1d30657"}}}}
{"t":{"$date":"2025-10-12T07:26:30.322+00:00"},"s":"I",  "c":"INDEX",    "id":20438,   "ctx":"conn2","msg":"Index build: registering","attr":
{"buildUUID":{"uuid":{"$uuid":"bc771c3b-2c51-41c9-9011-caca8be83af7"}},"namespace":"blogify.users","collectionUUID":{"uuid":{"$uuid":"bb391f74-836a-4edb-
a14f-6d311a486cab"}},"indexes":1,"firstIndex":{"name":"email_1"},"command":{"createIndexes":"users","v":2,"indexes":
[{"unique":true,"background":true,"name":"email_1","key":{"email":1}}],"ignoreUnknownIndexOptions":false}}}
{"t":{"$date":"2025-10-12T07:26:30.322+00:00"},"s":"I",  "c":"REPL",     "id":7360100, "ctx":"conn2","msg":"Added oplog entry for createIndexes to
transaction","attr":{"namespace":"blogify.$cmd","uuid":{"uuid":{"$uuid":"bb391f74-836a-4edb-a14f-6d311a486cab"}},"object":
{"createIndexes":"users","v":2,"unique":true,"key":{"email":1},"name":"email_1","background":true}}}
{"t":{"$date":"2025-10-12T07:26:30.333+00:00"},"s":"I",  "c":"INDEX",    "id":20345,   "ctx":"conn3","msg":"Index build: done building","attr":
{"buildUUID":null,"collectionUUID":{"uuid":{"$uuid":"007cc612-c51f-4067-897f-
a76e42f6d090"}},"namespace":"blogify.posts","index":"title_text_content_text","ident":"index-7-1731626970878222548","collectionIdent":"collection-1-
1731626970878222548","commitTimestamp":null}}
{"t":{"$date":"2025-10-12T07:26:30.333+00:00"},"s":"I",  "c":"INDEX",    "id":20440,   "ctx":"conn3","msg":"Index build: waiting for index build to
complete","attr":{"buildUUID":{"uuid":{"$uuid":"5d4fdec8-4c0a-4eb1-997f-78d4acce8b08"}},"deadline":{"$date":{"$numberLong":"9223372036854775807"}}}}
{"t":{"$date":"2025-10-12T07:26:30.333+00:00"},"s":"I",  "c":"INDEX",    "id":20447,   "ctx":"conn3","msg":"Index build: completed","attr":{"buildUUID":
{"uuid":{"$uuid":"5d4fdec8-4c0a-4eb1-997f-78d4acce8b08"}}}}
{"t":{"$date":"2025-10-12T07:26:30.352+00:00"},"s":"I",  "c":"INDEX",    "id":20345,   "ctx":"conn2","msg":"Index build: done building","attr":
{"buildUUID":null,"collectionUUID":{"uuid":{"$uuid":"bb391f74-836a-4edb-a14f-
6d311a486cab"}},"namespace":"blogify.users","index":"email_1","ident":"index-8-1731626970878222548","collectionIdent":"collection-0-
1731626970878222548","commitTimestamp":null}}
{"t":{"$date":"2025-10-12T07:26:30.353+00:00"},"s":"I",  "c":"INDEX",    "id":20440,   "ctx":"conn2","msg":"Index build: waiting for index build to
complete","attr":{"buildUUID":{"uuid":{"$uuid":"bc771c3b-2c51-41c9-9011-caca8be83af7"}},"deadline":{"$date":{"$numberLong":"9223372036854775807"}}}}
{"t":{"$date":"2025-10-12T07:26:30.353+00:00"},"s":"I",  "c":"INDEX",    "id":20447,   "ctx":"conn2","msg":"Index build: completed","attr":{"buildUUID":
{"uuid":{"$uuid":"bc771c3b-2c51-41c9-9011-caca8be83af7"}}}}
{"t":{"$date":"2025-10-12T07:26:40.659+00:00"},"s":"I",  "c":"NETWORK",  "id":22943,   "ctx":"listener","msg":"Connection accepted","attr":
{"remote":"172.18.0.3:47984","isLoadBalanced":false,"uuid":{"uuid":{"$uuid":"3643f7fc-0261-4bcf-9c49-
3d47466f9b52"}},"connectionId":5,"connectionCount":5}}
{"t":{"$date":"2025-10-12T07:26:40.663+00:00"},"s":"I",  "c":"NETWORK",  "id":51800,   "ctx":"conn5","msg":"client metadata","attr":
{"remote":"172.18.0.3:47984","client":"conn5","negotiatedCompressors":[],"doc":{"driver":
{"name":"nodejs|Mongoose","version":"6.20.0|8.19.1"},"platform":"Node.js v18.20.8, LE","os":{"name":"linux","architecture":"x64","version":"6.6.87.2-
microsoft-standard-WSL2","type":"Linux"},"env":{"container":{"runtime":"docker"}}}}}
```



Latest Posts

D Dhruv
10/12/2025

VESIT - A Hub of Learning and Excellence

Nestled in a serene part of the city, Vivekanand College stands as a beacon of knowledge and holistic development. Known for its excellent faculty, modern infrastructure, and vibrant student community, the college has been shaping the minds of future leaders for decades. Academic Excellence Vivekanand College offers a wide range of...

0 Comments

S Shifa Shaikh
10/12/2025

Hellooooooo

Hi, I'm An Aspiring Full Stack Developer.

Full Stack Development

0 Comments

Step 5: Stop Containers

Command:

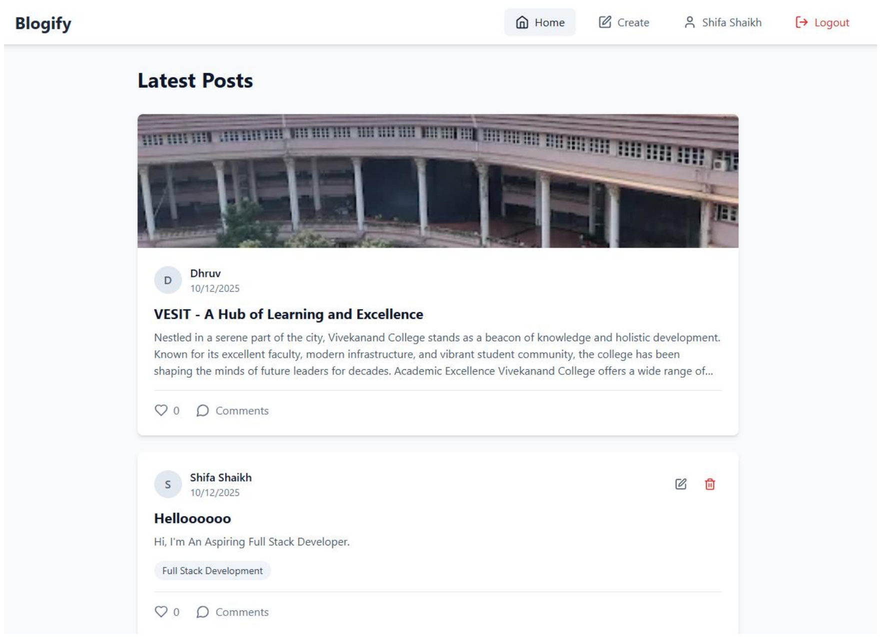
```
docker-compose down
```

- Stops and removes containers.

- Optional: `docker-compose down -v` to remove volumes and clear database.

5. Observations

- Docker containers isolate backend, frontend, and database.
- Using Docker Compose, the entire application can be started/stopped with a single command.
- Containerization ensures environment consistency.
- MongoDB Atlas works with containerized backend without needing a local MongoDB container.



Latest Posts

Dhruv
10/12/2025

VESIT - A Hub of Learning and Excellence

Nestled in a serene part of the city, Vivekanand College stands as a beacon of knowledge and holistic development. Known for its excellent faculty, modern infrastructure, and vibrant student community, the college has been shaping the minds of future leaders for decades. Academic Excellence Vivekanand College offers a wide range of...

0 Comments

Shifa Shaikh
10/12/2025

Helloooooo

Hi, I'm An Aspiring Full Stack Developer.

Full Stack Development

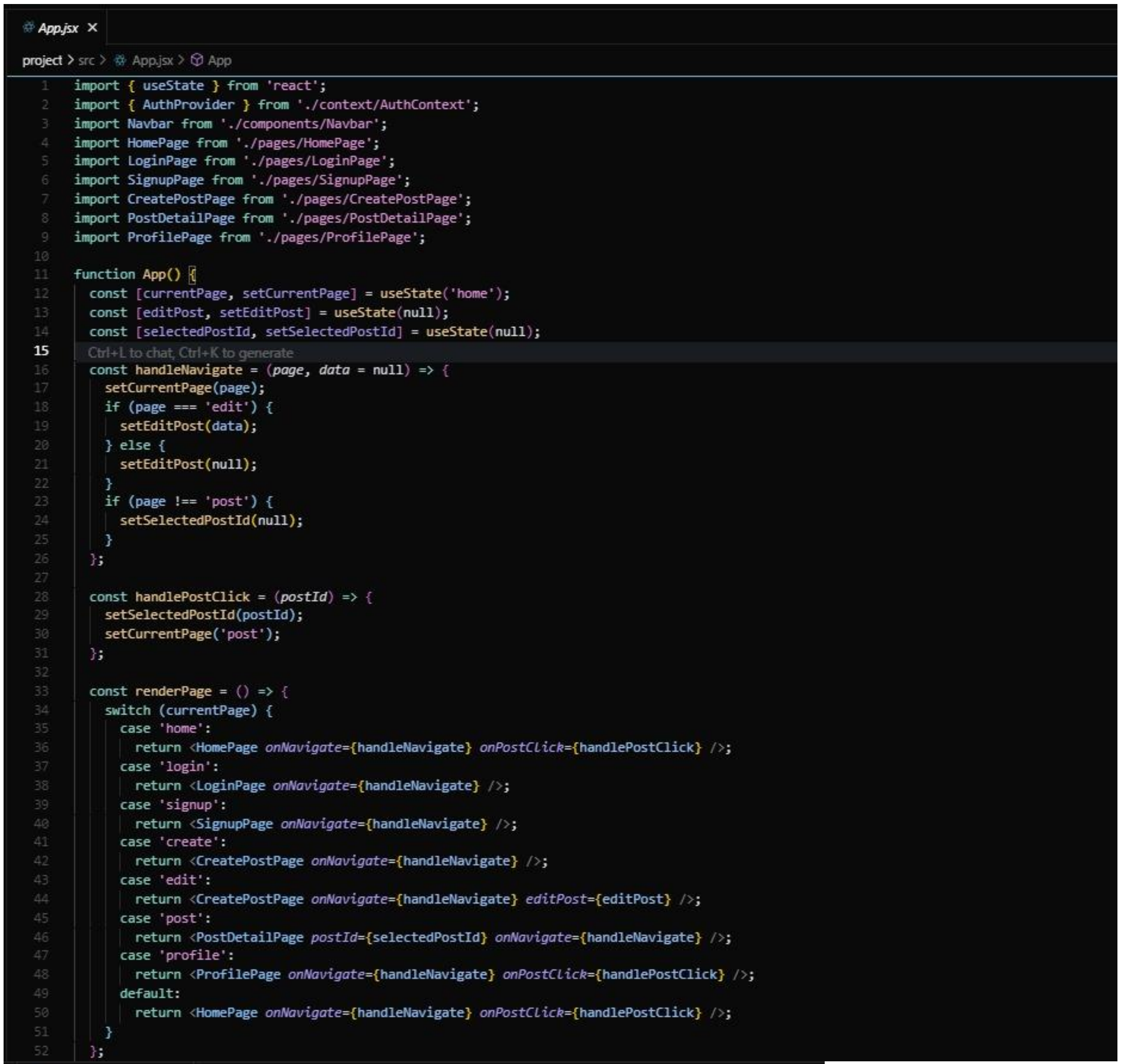
0 Comments

---

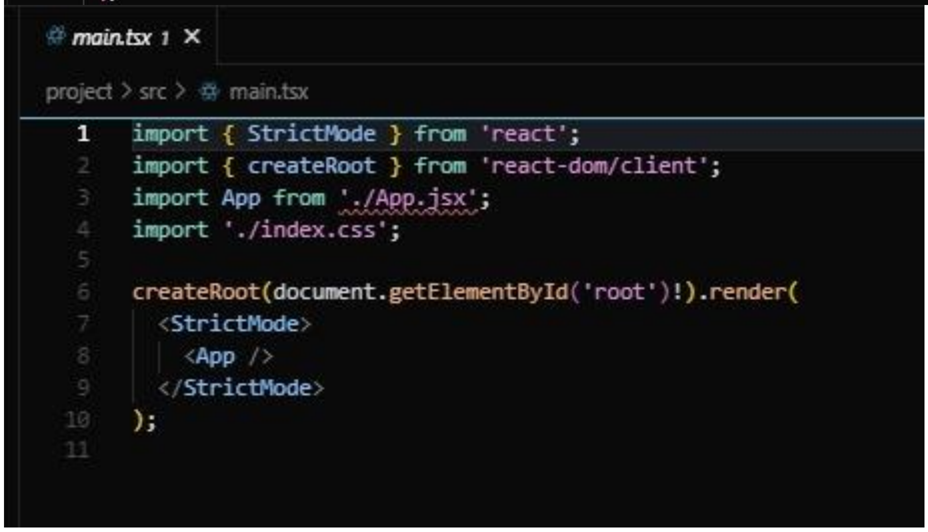
6. Conclusion

- The experiment successfully containerized the Blogify MERN application using Docker.
- Backend, frontend, and database can run independently in containers.
- Docker simplifies deployment and ensures portability.

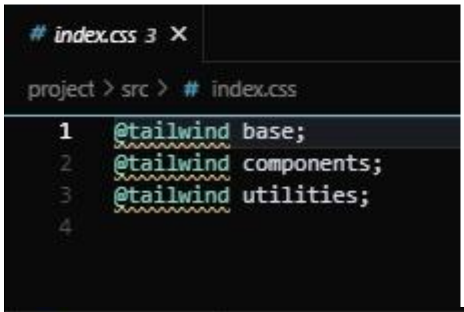
- Containerization is essential for modern software development and scalable applications.


App.jsx ×
project > src > App.jsx > App
Ctrl+L to chat, Ctrl+K to generate

```jsx
import { useState } from 'react';
import { AuthProvider } from './context/AuthContext';
import Navbar from './components/Navbar';
import HomePage from './pages/HomePage';
import LoginPage from './pages/LoginPage';
import SignupPage from './pages/SignupPage';
import CreatePostPage from './pages/CreatePostPage';
import PostDetailPage from './pages/PostDetailPage';
import ProfilePage from './pages/ProfilePage';

function App() {
  const [currentPage, setCurrentPage] = useState('home');
  const [editPost, setEditPost] = useState(null);
  const [selectedPostId, setSelectedPostId] = useState(null);

  const handleNavigate = (page, data = null) => {
    setCurrentPage(page);
    if (page === 'edit') {
      setEditPost(data);
    } else {
      setEditPost(null);
    }
    if (page !== 'post') {
      setSelectedPostId(null);
    }
  };

  const handlePostClick = (postId) => {
    setSelectedPostId(postId);
    setCurrentPage('post');
  };

  const renderPage = () => {
    switch (currentPage) {
      case 'home':
        return <HomePage onNavigate={handleNavigate} onPostClick={handlePostClick} />;
      case 'login':
        return <LoginPage onNavigate={handleNavigate} />;
      case 'signup':
        return <SignupPage onNavigate={handleNavigate} />;
      case 'create':
        return <CreatePostPage onNavigate={handleNavigate} />;
      case 'edit':
        return <CreatePostPage onNavigate={handleNavigate} editPost={editPost} />;
      case 'post':
        return <PostDetailPage postId={selectedPostId} onNavigate={handleNavigate} />;
      case 'profile':
        return <ProfilePage onNavigate={handleNavigate} onPostClick={handlePostClick} />;
      default:
        return <HomePage onNavigate={handleNavigate} onPostClick={handlePostClick} />;
    }
  };
```



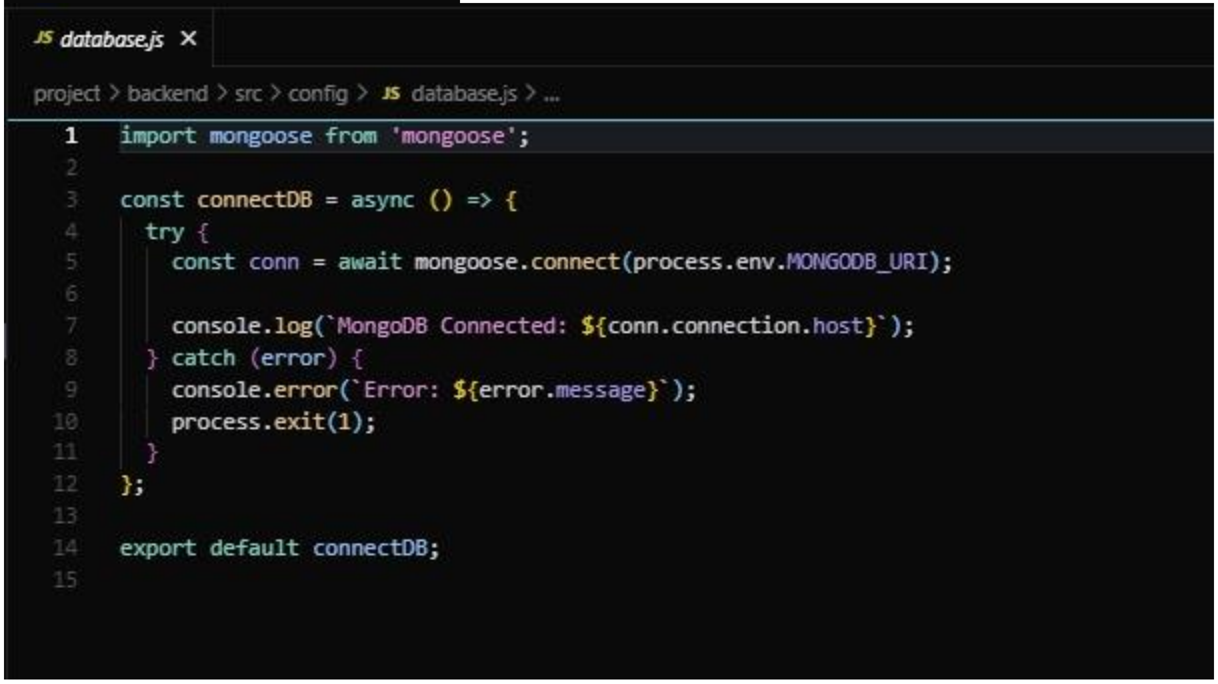
```tsx
import { StrictMode } from 'react';
import { createRoot } from 'react-dom/client';
import App from './App.jsx';
import './index.css';

createRoot(document.getElementById('root')!).render(
  <StrictMode>
    <App />
  </StrictMode>
);
```



index.css 3 ×

project > src > # index.css

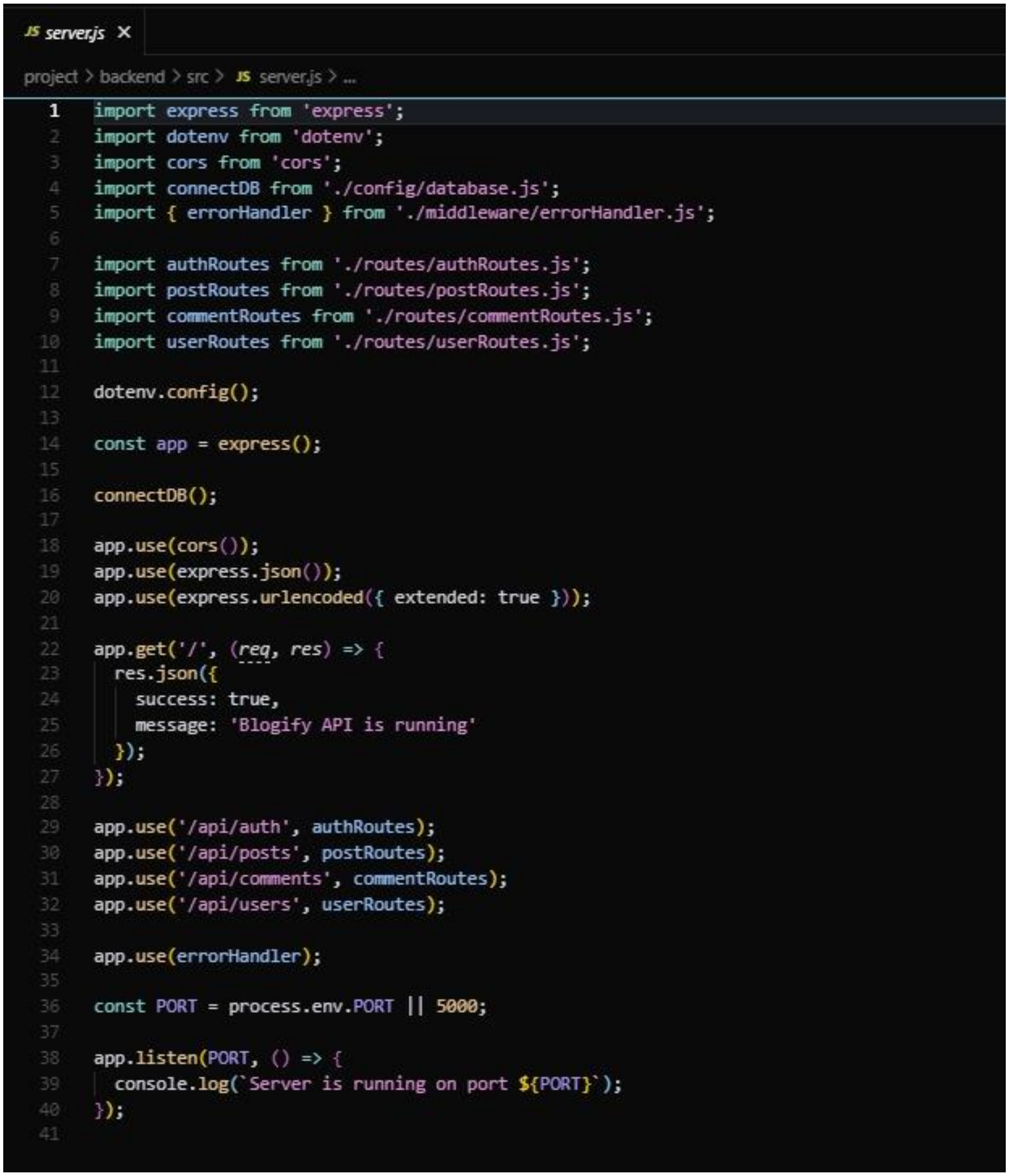
```css
@tailwind base;
@tailwind components;
@tailwind utilities;
```



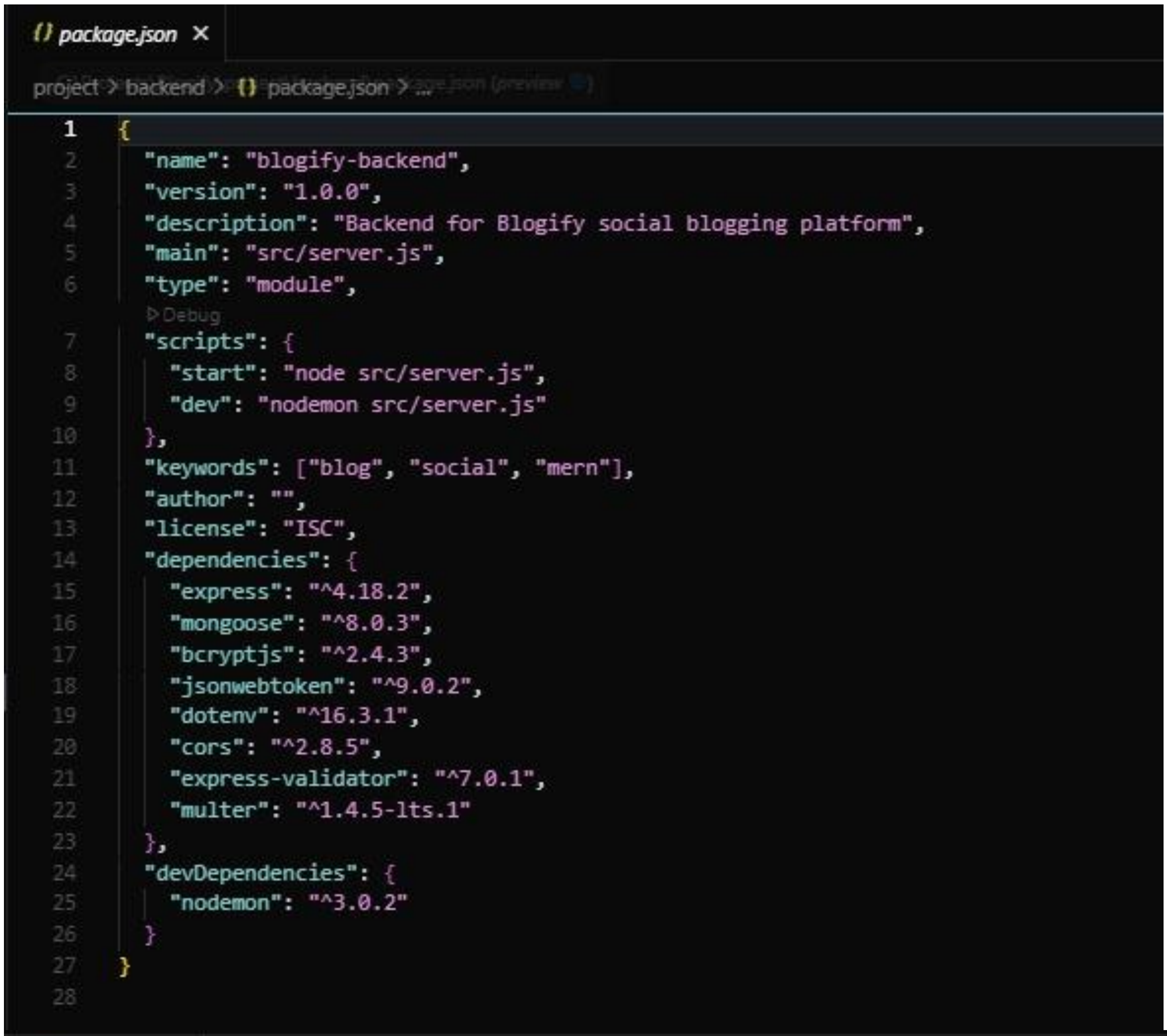
database.js ×

project > backend > src > config > database.js > ...

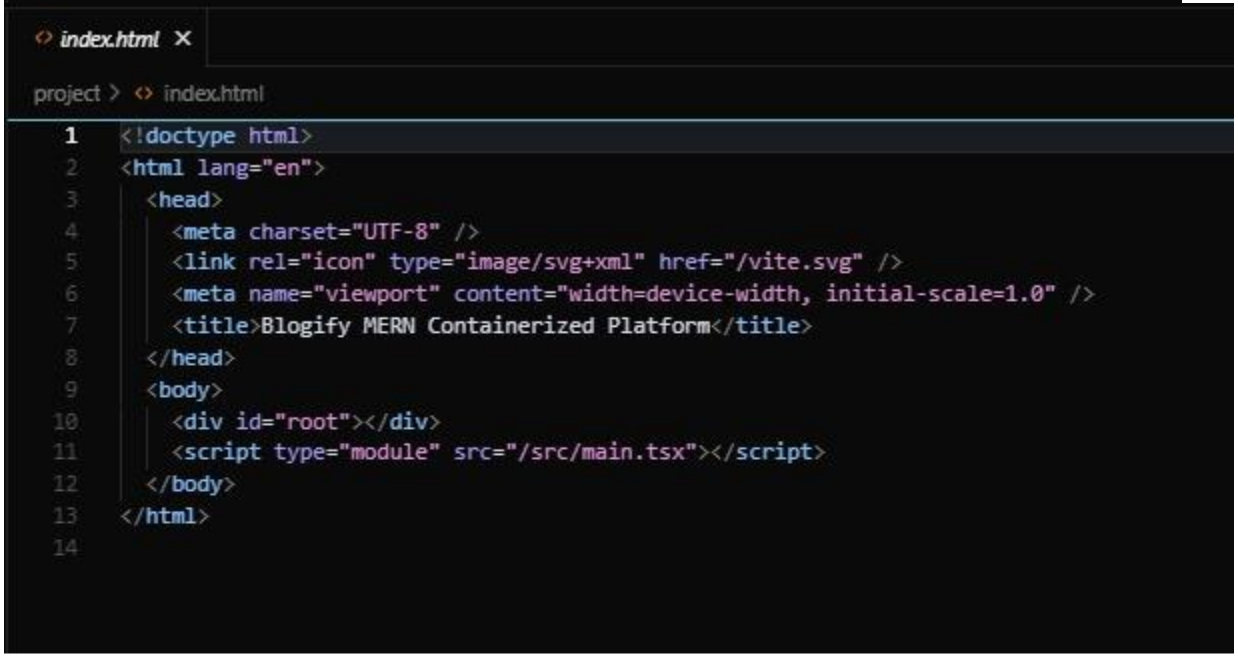
```javascript
import mongoose from 'mongoose';

const connectDB = async () => {
  try {
    const conn = await mongoose.connect(process.env.MONGODB_URI);

    console.log(`MongoDB Connected: ${conn.connection.host}`);
  } catch (error) {
    console.error(`Error: ${error.message}`);
    process.exit(1);
  }
};

export default connectDB;
```



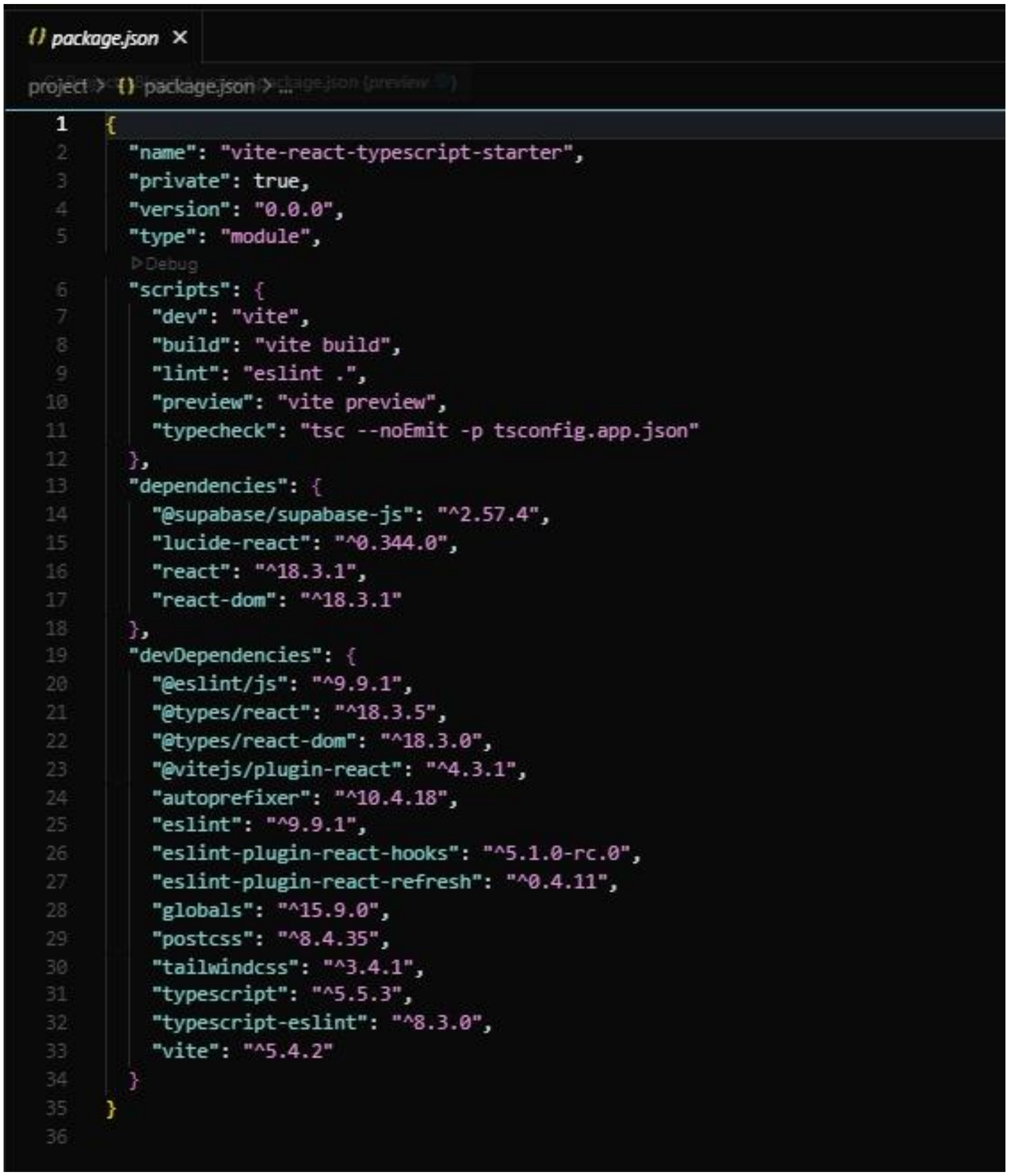
```javascript
import express from 'express';
import dotenv from 'dotenv';
import cors from 'cors';
import connectDB from './config/database.js';
import { errorHandler } from './middleware/errorHandler.js';

import authRoutes from './routes/authRoutes.js';
import postRoutes from './routes/postRoutes.js';
import commentRoutes from './routes/commentRoutes.js';
import userRoutes from './routes/userRoutes.js';

dotenv.config();

const app = express();

connectDB();

app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.get('/', (req, res) => {
  res.json({
    success: true,
    message: 'Blogify API is running'
  });
});

app.use('/api/auth', authRoutes);
app.use('/api/posts', postRoutes);
app.use('/api/comments', commentRoutes);
app.use('/api/users', userRoutes);

app.use(errorHandler);

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```



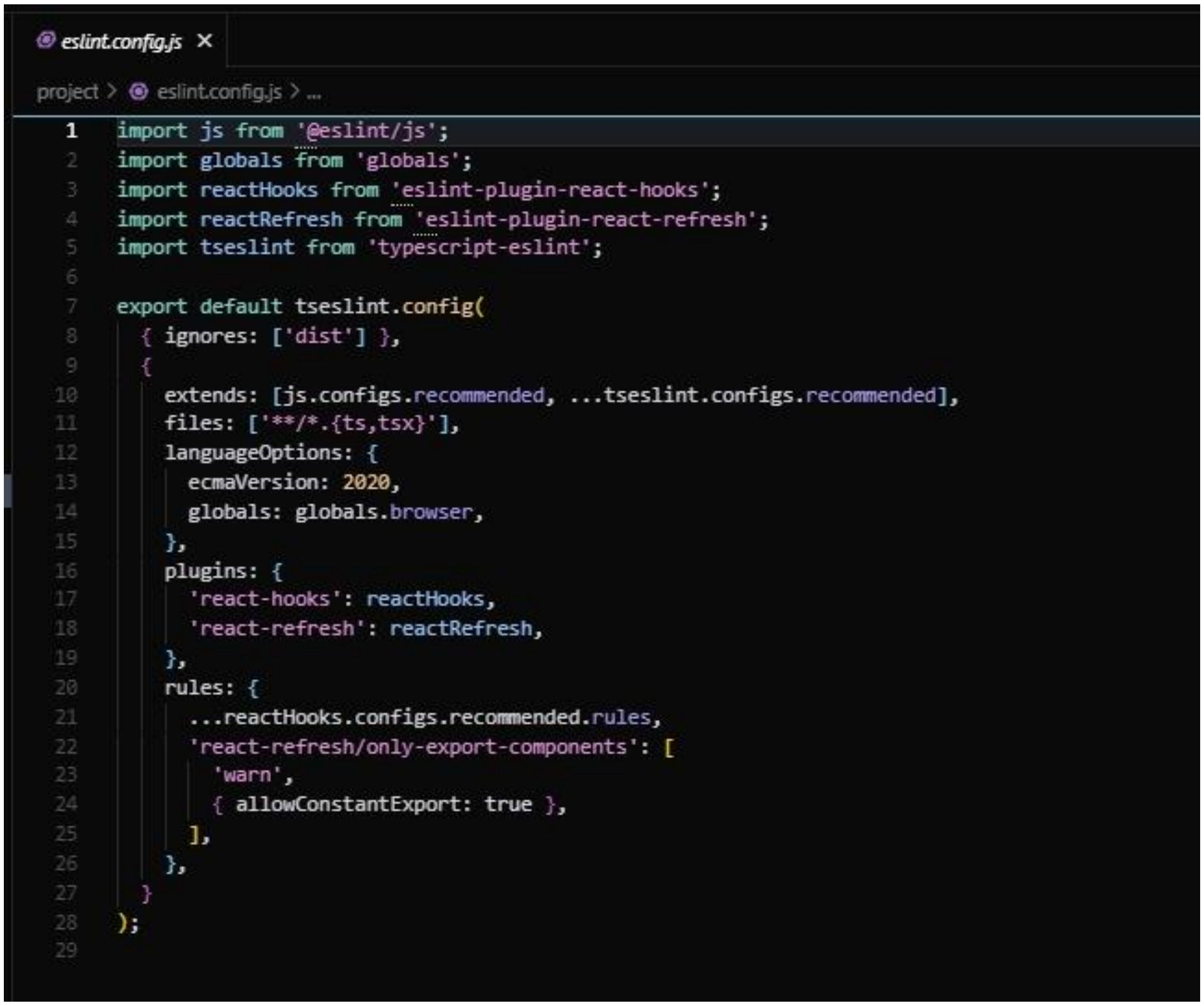
```json
{
  "name": "blogify-backend",
  "version": "1.0.0",
  "description": "Backend for Blogify social blogging platform",
  "main": "src/server.js",
  "type": "module",
  "scripts": {
    "start": "node src/server.js",
    "dev": "nodemon src/server.js"
  },
  "keywords": ["blog", "social", "mern"],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2",
    "mongoose": "^8.0.3",
    "bcryptjs": "^2.4.3",
    "jsonwebtoken": "^9.0.2",
    "dotenv": "^16.3.1",
    "cors": "^2.8.5",
    "express-validator": "^7.0.1",
    "multer": "^1.4.5-lts.1"
  },
  "devDependencies": {
    "nodemon": "^3.0.2"
  }
}
```



```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Blogify MERN Containerized Platform</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
```



```json
{
  "name": "vite-react-typescript-starter",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
    "preview": "vite preview",
    "typecheck": "tsc --noEmit -p tsconfig.app.json"
  },
  "dependencies": {
    "@supabase/supabase-js": "^2.57.4",
    "lucide-react": "^0.344.0",
    "react": "^18.3.1",
    "react-dom": "^18.3.1"
  },
  "devDependencies": {
    "@eslint/js": "^9.9.1",
    "@types/react": "^18.3.5",
    "@types/react-dom": "^18.3.0",
    "@vitejs/plugin-react": "^4.3.1",
    "autoprefixer": "^10.4.18",
    "eslint": "^9.9.1",
    "eslint-plugin-react-hooks": "^5.1.0-rc.0",
    "eslint-plugin-react-refresh": "^0.4.11",
    "globals": "^15.9.0",
    "postcss": "^8.4.35",
    "tailwindcss": "^3.4.1",
    "typescript": "^5.5.3",
    "typescript-eslint": "^8.3.0",
    "vite": "^5.4.2"
  }
}
```



```js
import js from '@eslint/js';
import globals from 'globals';
import reactHooks from 'eslint-plugin-react-hooks';
import reactRefresh from 'eslint-plugin-react-refresh';
import tseslint from 'typescript-eslint';

export default tseslint.config(
  { ignores: ['dist'] },
  {
    extends: [js.configs.recommended, ...tseslint.configs.recommended],
    files: ['**/*.{ts,tsx}'],
    languageOptions: {
      ecmaVersion: 2020,
      globals: globals.browser,
    },
    plugins: {
      'react-hooks': reactHooks,
      'react-refresh': reactRefresh,
    },
    rules: {
      ...reactHooks.configs.recommended.rules,
      'react-refresh/only-export-components': [
        'warn',
        { allowConstantExport: true },
      ],
    },
  }
);
```