SynapShare Project Documentation

1. Project Structure

```
SynapShare/
│
├── synapshare-backend/   # Node.js/Express backend with MongoDB & Firebase
│
└── synapshare-frontend/  # React.js frontend with Tailwind CSS
```

2. Backend: synapshare-backend

Main Technologies

- Node.js with Express.js for REST API
- MongoDB (via mongoose) for data storage
- Firebase Admin SDK for authentication
- Multer for file uploads
- Nodemailer for sending emails (e.g., OTPs)
- dotenv for environment variables

Key Files & Directories

- **server.js** — Main Express server, all routes and logic
- **models/** — (Likely) contains Mongoose schemas (User, Note, Discussion, Node, SavedPost)
- **routes/** — (Likely) contains modularized route handlers
- **functions/** — Firebase Cloud Functions setup
- **uploads/** — Stores uploaded files
- **.env** — Environment variables (not public)

Core Features

- **User Authentication**: Uses Firebase tokens, verifies users, manages usernames and email verification.
- **Notes, Discussions, Nodes**: CRUD operations for each, including file uploads and voting (upvote/downvote).
- **Admin Features**: Admins can delete any notes, discussions, or nodes.
- **File Uploads**: Supports images, PDFs, and MP4/WebM videos (max 50MB).

- **Search**: Full-text search across notes, discussions, and nodes.
- **Voting & Comments**: Users can upvote/downvote and comment on items.
- **Password Reset & OTP**: OTP-based password reset via email.
- **API Endpoints**: `/api/notes`, `/api/discussions`, `/api/nodes`, `/api/user`, `/api/search`, etc.

Security

- **JWT Verification**: Middleware checks Firebase tokens for protected routes.
- **Role-based Access**: Admin routes protected by isAdmin middleware.
- **File Type & Size Validation**: Enforced via Multer.

3. Frontend: synapshare-frontend

Main Technologies

- React.js (functional components, hooks)
- React Router for navigation
- Axios for API requests
- Firebase for authentication
- Tailwind CSS for styling
- React Icons for UI icons
- Particles.js for visual effects

Key Files & Directories

- **src/App.js** — Main application component, routing, theme management
- **src/pages/** — Contains main pages: Home, Login, Notes, Discussions, Nodes, News, Search, Admin
- **src/components/** — Reusable components (e.g., Footer, Logo)
- **public/** — Static assets and index.html
- **tailwind.config.js** — Tailwind CSS configuration

Core Features

- **Authentication**: Login/register with email/password or Google, username selection after login.
- **Theming**: Light/Dark mode toggle, persisted via localStorage.
- **Notes, Discussions, Nodes**: UI for creating, editing, deleting, commenting, voting, and viewing.
- **Admin Panel**: Special admin page for managing all content and users.

- **Search**: Search bar to find notes, discussions, and nodes.
- **Responsive Design**: Modern, mobile-friendly UI.
- **Error Handling**: User-friendly error messages and loading states.
- **News Page**: (Likely) fetches and displays latest news.

4. Data Models (from backend schemas)

- **User**: uid, username, email, isEmailVerified, etc.
- **Note**: title, fileUrl, uploadedBy, subject, voting, comments, etc.
- **Discussion**: title, content, fileUrl, postedBy, voting, comments, etc.
- **Node**: title, description, codeSnippet, fileUrl, postedBy, voting, comments, etc.
- **SavedPost**: Links user to saved notes/discussions/nodes.

5. Security & Best Practices

- **Environment Variables**: Sensitive info in .env
- **Access Control**: Token-based, role-based for admin actions
- **Input Validation**: On both frontend (forms) and backend (API)
- **File Handling**: Secure upload directory, file type/size checks

6. Setup & Deployment

- **Backend**: npm install, node server.js (requires MongoDB, Firebase credentials)
- **Frontend**: npm install, npm start (runs on port 3000 by default)
- **Firebase Functions**: Configured for deployment via firebase.json and functions/

7. Notable Features

- Full-stack authentication and content management
- Integrated voting and commenting system
- Admin dashboard for moderation
- Modern, responsive, and visually appealing UI
- Secure file uploads and user data management
- Extensible architecture for future features

8. Recommendations

- **Documentation**: Add more usage and setup details in README.md for both frontend and backend.

- **Testing**: Add unit/integration tests for backend and frontend.
- **Deployment**: Consider Dockerizing for easier deployment.
- **Monitoring**: Add logging/monitoring for production use.

Summary

SynapShare is a robust, full-stack web application for sharing and discussing notes and ideas, with strong authentication, admin controls, and a modern user interface. It leverages React and Tailwind on the frontend, and Node.js/Express, MongoDB, and Firebase on the backend. The architecture is modular and follows best practices for security and scalability.