Recipe Generator Web Application

A modern web application that helps users find or generate recipes based on their available ingredients. The app searches an internal Firebase database first, and if no matches are found, uses AI to generate custom recipes.

🚀 Features

- **Dynamic Ingredient Input**: Tag-style input system for easy ingredient management
- **Database Search**: Searches Firebase Firestore for matching recipes
- **AI Generation**: Uses Together AI's Llama model to generate custom recipes
- **Responsive Design**: Clean, modern interface with smooth animations
- **Loading Animation**: Custom knife-cutting animation while processing

📁 Project Structure

```
recipe-generator/
├── public/
│   └── index.html          # Frontend application
├── netlify/
│   └── functions/
│       └── recipe-api.py   # Serverless Flask API
├── netlify.toml            # Netlify configuration
├── requirements.txt        # Python dependencies
├── .env.example            # Environment variables template
└── README.md               # Project documentation
```

🛠️ Technology Stack

- **Frontend**: HTML5, CSS3, Vanilla JavaScript
- **Backend**: Python Flask (Serverless)
- **Database**: Firebase Firestore
- **AI Provider**: Together AI (Llama-3-8B-Instruct-Turbo)
- **Deployment**: Netlify

📋 Prerequisites

1. **Firebase Project**
   - Create a Firebase project at Firebase Console

- Enable Firestore Database
- Generate a service account key

2. **Together AI Account**
   - Sign up at Together AI
   - Get your API key
3. **Netlify Account**
   - Sign up at Netlify

🔧 Setup Instructions

1. Clone the Repository

```bash
git clone <your-repo-url>
cd recipe-generator
```

2. Project Structure Setup

Create the following directory structure:

```bash
mkdir -p public netlify/functions
```

Move the files to their correct locations:

- Place `index.html` in the `public/` directory
- Place `recipe-api.py` in the `netlify/functions/` directory
- Keep `netlify.toml` and `requirements.txt` in the root directory

3. Firebase Setup

1. Go to Firebase Console → Your Project → Project Settings → Service Accounts
2. Click "Generate New Private Key"
3. Save the downloaded JSON file securely
4. Create a Firestore collection called `recipes` with this structure:

```javascript
```

```
{
  title: "Recipe Name",          // String
  description: "Description",    // String
  ingredients: ["ing1", "ing2"],  // Array of lowercase strings
  instructions: ["Step 1", "Step 2"] // Array of strings
}
```

4. Environment Variables

1. Copy `.env.example` to `.env`:

   ```bash
   cp .env.example .env
   ```

2. Fill in your credentials in `.env`:
   - Firebase credentials from the service account JSON
   - Together AI API key

5. Local Development (Optional)

To test locally, you can run the Flask app:

```bash
pip install -r requirements.txt
python netlify/functions/recipe-api.py
```

Then open `public/index.html` in a browser and update the API_URL in the JavaScript to point to your local Flask server.

🚀 Deployment to Netlify

Method 1: Deploy with Git

1. Push your code to a GitHub repository
2. Log in to Netlify
3. Click "New site from Git"
4. Choose your repository
5. Configure build settings:
   - Build command: (leave empty)

- Publish directory: `public`

6. Add environment variables in Site Settings → Environment Variables
7. Deploy!

Method 2: Manual Deploy

1. Install Netlify CLI:

```bash
npm install -g netlify-cli
```

2. Login to Netlify:

```bash
netlify login
```

3. Initialize the site:

```bash
netlify init
```

4. Set environment variables:

```bash
netlify env:set FIREBASE_PROJECT_ID "your-project-id"
netlify env:set FIREBASE_PRIVATE_KEY_ID "your-key-id"
# ... set all other variables
```

5. Deploy:

```bash
netlify deploy --prod
```

🔑 Environment Variables Reference

Variable	Description	Required
`FIREBASE_PROJECT_ID`	Your Firebase project ID	Yes
`FIREBASE_PRIVATE_KEY_ID`	Private key ID from service account	Yes
`FIREBASE_PRIVATE_KEY`	Private key (include newlines as \n)	Yes
`FIREBASE_CLIENT_EMAIL`	Service account email	Yes

Variable	Description	Required
FIREBASE_CLIENT_ID	Client ID from service account	Yes
FIREBASE_CERT_URL	Certificate URL	Yes
TOGETHER_API_KEY	Your Together AI API key	Yes
SAVE_AI_RECIPES	Save AI recipes to database (true/false)	No

📝 Adding Sample Recipes to Database

Here's a sample recipe structure for your Firestore database:

```javascript
// Example recipe document
{
  title: "Simple Tomato Pasta",
  description: "A quick and delicious pasta dish with fresh tomatoes",
  ingredients: ["tomatoes", "pasta", "garlic", "olive oil", "basil"],
  instructions: [
    "Boil pasta according to package directions",
    "Dice tomatoes and mince garlic",
    "Heat olive oil in a pan",
    "Sauté garlic until fragrant",
    "Add tomatoes and cook for 5 minutes",
    "Toss with pasta and fresh basil",
    "Season with salt and pepper to taste"
  ]
}
```

🎨 Customization

Styling

- Modify the CSS in `index.html` to change colors, fonts, and layouts
- The current theme uses a purple gradient background

AI Prompt

- Edit the prompt template in `recipe-api.py` to change how recipes are generated
- Adjust the Together AI parameters (temperature, max_tokens) for different outputs

Database Schema

- Add additional fields like `cookingTime`, `difficulty`, `servings`, etc.
- Update the search logic accordingly

🐛 Troubleshooting

Common Issues

1. **CORS Errors**
   - Ensure the Flask-CORS is properly configured
   - Check Netlify headers configuration in `netlify.toml`
2. **Firebase Connection Issues**
   - Verify all environment variables are set correctly
   - Check that the private key has proper newline characters (\n)
3. **Together AI Rate Limits**
   - Check your API usage on the Together AI dashboard
   - Consider implementing rate limiting in the application
4. **Function Timeout**
   - Netlify functions have a 10-second timeout by default
   - Consider caching or optimizing the AI generation

📚 API Endpoints

POST `/.netlify/functions/recipe-api`

Request body:

```json
{
  "ingredients": ["tomato", "pasta", "cheese"],
  "forceAI": false
}
```

Response (Database match):

```json
```

```json
{
  "source": "database",
  "recipes": [
    {
      "id": "recipe_id",
      "title": "Recipe Title",
      "description": "Description",
      "ingredients": ["..."],
      "instructions": ["..."]
    }
  ]
}
```

Response (AI generated):

```json
{
  "source": "ai",
  "recipe": {
    "title": "AI Recipe Title",
    "description": "Description",
    "ingredients": ["..."],
    "instructions": ["..."]
  }
}
```

🤝 Contributing

1. Fork the repository
2. Create a feature branch
3. Commit your changes
4. Push to the branch
5. Open a Pull Request

📄 License

This project is open source and available under the MIT License.

🆘 Support

For issues or questions, please create an issue in the GitHub repository.

Built with ❤️ using Flask, Firebase, and AI