Diet Guidelines Application

A comprehensive nutrition tracking application with webpack bundling.

Project Structure

```
project/
├── app.js                 # Node.js Express server
├── services.js            # API service handlers
├── lib/                   # Server-side JavaScript modules
│   └── db/                # Database files and schema
│       ├── *.js           # Database helper modules
│       ├── *.db           # SQLite database
│       └── *.sql          # Database schema
├── public/                # Source files (development)
│   ├── css/               # Stylesheets
│   └── js/                # Client-side JavaScript
│       └── client.js      # Main client entry point
├── index.html             # HTML template
├── dist/                  # Built files (production)
├── webpack.common.js      # Shared webpack config
├── webpack.dev.js         # Development webpack config
├── webpack.prod.js        # Production webpack config
└── package.json
```

Installation

```
bash

npm install
```

Development

Run both the Node.js server and webpack dev server concurrently:

```
bash

npm run dev
```

This will:

- Start the Node.js API server on `http://localhost:3000`

- Start the webpack dev server on `http://localhost:3001`
- Enable hot module replacement (HMR)
- Proxy API requests from port 3001 to 3000

Or run them separately:

```bash
# Terminal 1: Run Node.js server
npm run dev:server

# Terminal 2: Run webpack dev server
npm run dev:client
```

Production Build

Build the production bundle:

```bash
npm run build
```

This will:

- Bundle and minify JavaScript
- Minify CSS
- Generate content hashes for cache busting
- Create source maps
- Output to `dist/` directory

Run the production server:

```bash
NODE_ENV=production npm start
```

The server will serve the built files from the `dist/` directory.

Scripts

- `npm start` - Start the Node.js server (production mode)

- `npm start` - Start the Node.js server (production mode)
- `npm run dev` - Run both servers in development mode
- `npm run dev:server` - Run only the Node.js server with nodemon
- `npm run dev:client` - Run only the webpack dev server
- `npm run build` - Build production bundle
- `npm run clean` - Remove the dist directory

Webpack Configuration

webpack.common.js

- Entry point: `public/js/client.js`
- Output: `dist/js/[name].[contenthash].js`
- HTML injection via HtmlWebpackPlugin
- CSS file copying via CopyWebpackPlugin
- Babel transpilation for ES6+ support
- Code splitting for vendor libraries

webpack.dev.js

- Development mode with inline source maps
- Webpack dev server on port 3001
- Hot module replacement (HMR)
- API proxy to backend server (port 3000)

webpack.prod.js

- Production mode with optimizations
- Terser for JavaScript minification (removes console.log)
- CSS minification
- Source maps for debugging
- Performance hints

Environment Variables

Set `NODE_ENV=production` when running in production to:

- Serve static files from `dist/`
- Enable production optimizations
- Serve index.html for all routes (SPA support)

API Endpoints

All API routes are prefixed with `/api`:

- `GET /api/config` - Get application configuration
- `GET /api/kidney-stone-risk` - Get kidney stone risk levels
- `GET /api/daily-requirements` - Get daily nutritional requirements
- `GET /api/recipes` - List all recipes
- `GET /api/recipes/:id` - Get recipe details
- `GET /api/recipes/:id/full` - Get full recipe data
- `POST /api/recipes` - Create new recipe
- `PUT /api/recipes/:id` - Update recipe
- `DELETE /api/recipes/:id` - Delete recipe
- `GET /api/ingredients` - List all ingredients
- `GET /api/ingredients/:id` - Get ingredient details
- `GET /api/ingredients/:id/full` - Get full ingredient data
- `POST /api/ingredients` - Create new ingredient
- `PUT /api/ingredients/:id` - Update ingredient
- `DELETE /api/ingredients/:id` - Delete ingredient

Notes

- The webpack dev server proxies API requests to the Node.js server
- In development, access the app at `http://localhost:3001`
- In production, the Node.js server serves both API and static files
- CSS files are copied as-is (not processed through webpack)

- Client JavaScript is bundled and transpiled through webpack
- Database files in `lib/db/` are server-side only
- **All configuration files use ES modules** (`import`/`export`) since `package.json` has `"type": "module"`
- Webpack config files must use `.js` extension and include `.js` in import paths