CS 144 Final Project Writeup

Fruits & Vegetables Explorer
Namet: Xingbo Wu
Project: Individual

Specification Requirements Compliance

☑ 1. Semantic HTML5 & Required APIs

Implementation: Used semantic HTML5 elements (`<header>`, `<nav>`, `<article>`, `<section>`) throughout React components. Implemented **Drag and Drop API** using @dnd-kit library for comparing produce items side-by-side.
Location: `src/pages/Detail.jsx`, `src/components/DraggableProduceCard.jsx`, `src/components/ComparisonDropZone.jsx`

☑ 2. Responsive Design (320px minimum)

Implementation: Flexible grid layouts that adapt from 1 column (mobile) to 4+ columns (desktop). Comprehensive media queries ensure functionality down to 320px width.
Location: `src/index.css` with @media queries for various screen sizes

☑ 3. Progressive Web App (PWA) - Offline Functionality

Implementation: Service Worker caches resources and provides offline functionality. Shows appropriate messages and contents when offline. Web App Manifest enables installation.
Location: `public/sw.js`, `public/manifest.json`, `src/utils/registerSW.js`, `public/offline.html`

☑ 4. HTTPS

Implementation: All Google Cloud App Engine services use HTTPS by default. Both frontend and backend `app.yaml` files specify `secure: always` for all routes.
Location: Both `app.yaml` configuration files

☑ 5. Single Page Application

Implementation: React Router handles all navigation client-side without page reloads. Content scrolls within components, no horizontal scrolling.
Location: `src/App.jsx` with React Router configuration

☑ 6. Aesthetic Design

Implementation: Clean, modern interface with consistent green color scheme, gradients, and professional styling. Prioritized functionality while maintaining good UX.
Location: Comprehensive styling in `src/index.css`

☑ 7. CSS Processing

Implementation: Modern CSS with flexbox, grid, gradients, and custom properties. Chose custom CSS over preprocessors for better learning and control.
Location: `src/index.css` with organized, maintainable styles

☑ 8. Authentication with Cookies

Implementation: JWT tokens stored in HTTP-only cookies for secure authentication. Cookie banner informs users of cookie usage and rights.
Location: `fruit-veggie-api/routes/userRoutes.js`, `src/components/CookieBanner.jsx`

☑ 9. Security (SQL Injection, CSRF, XSS)

Implementation: MongoDB with Mongoose ODM prevents NoSQL injection. HTTP-only cookies prevent XSS on auth tokens. Server-side input validation on all endpoints. CSRF protection through cookie-based auth.
Location: `fruit-veggie-api/middleware/authMiddleware.js`, all route handlers with validation

☑ 10. Database and Caching Layer

Implementation: MongoDB as primary database using Mongoose ODM. Redis as optional caching layer for API performance improvement.
Location: `fruit-veggie-api/models/` (schemas), `fruit-veggie-api/config/redis.js`

☑ 11. Node.js and Express Backend

Implementation: Express.js RESTful API with modular architecture. Separated routes, models, services, and middleware for maintainability.
Location: `fruit-veggie-api/server.js`, `fruit-veggie-api/routes/`

☑ 12. PWA with Service Worker

Implementation: Service Worker handles caching and offline functionality. Manifest allows installation on desktop and mobile. Shows appropriate offline messages.
Location: `public/sw.js`, `public/manifest.json` with install prompts

☑ 13. WebAssembly Module (Extra Credit)

Implementation: AssemblyScript compiled to WebAssembly for nutrition scoring algorithms and serving size calculations. Provides near-native performance for complex mathematical operations.
Location: `assembly/index.ts`, `src/services/wasmService.js`

☑ 14. AI Services Integration

Implementation: Google Vertex AI integration using Gemini 2.0 Flash Lite model for health insights and translations. AI responses are cached to reduce costs and improve performance.
Location: `fruit-veggie-api/services/vertexAIService.js`, `fruit-veggie-api/routes/aiRoutes.js`

☑ 15. Frontend Framework

Implementation: React 18 with modern hooks, Context API for state management (user auth, comparison features), and functional component patterns.

Location: All components in `src/components/`, `src/pages/`, `src/contexts/`

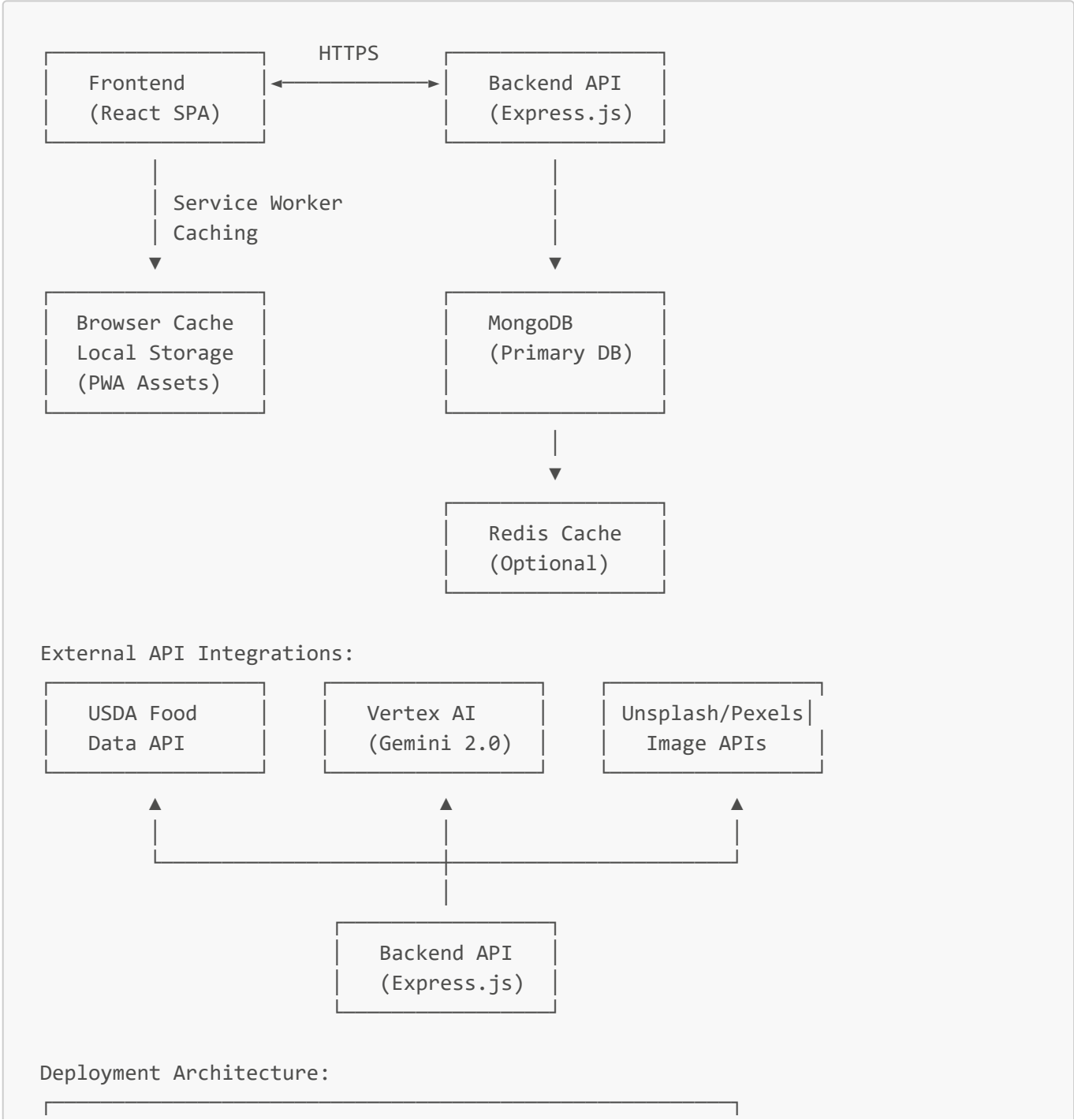
☑ 16. Accessibility

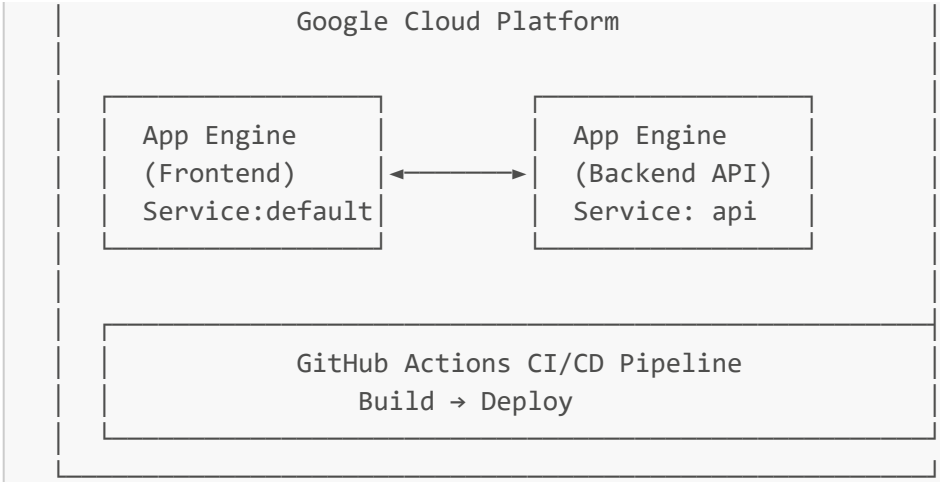
Implementation: Semantic HTML with proper heading hierarchy, ARIA labels for screen readers, high color contrast meeting WCAG guidelines.
Location: Throughout all components with `aria-label` attributes and semantic structure

☑ 17. Google Cloud Deployment with CI/CD

Implementation: Google App Engine deployment for both frontend and backend services. GitHub Actions automates CI/CD pipeline for build and deployment.
Location: `fruit-veggie-app/app.yaml`, `fruit-veggie-api/app.yaml`, `.github/workflows/`

System Architecture Diagram





Communication Flow:

1. **Frontend ↔ Backend:** HTTPS REST API calls with JSON payloads
2. **Backend ↔ Database:** Mongoose ODM with connection pooling
3. **Backend ↔ External APIs:** Authenticated HTTP requests to USDA, Vertex AI, image services
4. **Client ↔ Cache:** Service Worker manages cache strategy and offline functionality
5. **CI/CD:** GitHub Actions triggers automated deployment on code push

Security and Privacy Features

Authentication & Session Management

- **JWT with HTTP-only cookies:** Prevents XSS attacks by making tokens inaccessible to JavaScript
- **Secure cookie configuration:** `secure: true` in production, `sameSite` protection
- **Token expiration:** 30-day expiration with automatic cleanup

Data Protection

- **MongoDB with Mongoose ODM:** Built-in protection against NoSQL injection attacks through parameterized queries
- **Input validation:** Server-side validation on all user inputs before database operations
- **CORS configuration:** Restricted to specific origins, prevents unauthorized cross-origin requests

Privacy Compliance

- **Cookie banner:** GDPR-compliant notification informing users of cookie usage and rights
- **Minimal data collection:** Only collects essential user data (name, favorites)
- **Data encryption:** All communications encrypted via HTTPS

Infrastructure Security

- **Google Cloud App Engine:** Automatic security patches and DDoS protection
- **Environment variables:** Sensitive configuration stored securely, not in code
- **Service isolation:** Frontend and backend deployed as separate services

CI/CD Log - Successful Deployment

GitHub Actions Workflow Runs