Beans

Local Food Buying Groups made EASY

Table of Contents

Version history

Version	Date	Notes
1	Dec 2019	Initial brief & planning document
2	11th May 2020	Update while part way through ‘Scaffolding’ phase. Structure, modularity, state management, routing, initial component development and initial document-store schemas completed.

Beans

Local Food Buying Groups made EASY

Brief

Eating well and living mindfully leads away from supermarkets with their long supply chains, squeezing of farmers and chasing of shareholder profits toward small groups of people cooperating to buy from trusted farmers in their area. To source organic food locally and reliably though is quite complex especially on the scale of 10 - 30 members of a typical food group. What is needed is an app which takes away much of the pain of organising custom orders of regular local ethical food.

A sophisticated and intuitive app which enables the many facets of a food group to work smoothly could also then be a major help in others wishing to adopt the local food group approach. This is not a lucrative business opportunity: there is not more than $50 per week per group for this kind of app. However for those who want the freshest most nutritious food grown in a way that builds soil and gives local farmers a fair return this app would soon become essential.

Russell (document author & budding web developer) has been involved in such a group for 8 years, using a locally developed online web app. While ensuring $3000+ of custom orders per fortnight within a group of 40 people would be impossible without it, there are many places where friction could be reduced. ‘Beans’ is thus a re-think and extension of existing software.

The development of ‘Beans’ also is a means for me (Russell) to learn and showcase skills in web development with a strong focus in Vue JS.

Goals – project

1. An easy-to-administer management tool for food group organisers
2. An easy-to-use ordering and reminder tool for food group members
3. A fast, mobile, easy to use app that works well even with flaky internet
4. An app which assists admin & members as follows:
    1. Sophisticated import & processing of supplier availability files
    2. Intuitive ordering and shopping cart experience
    3. Managing orders to suppliers
    4. Generated printouts for delivery, packing and pickup
    5. Timely notifications re ordering, pickup, packing rosters, etc
    6. In-app tracking of invoices & payments for both suppliers and members
    7. API/s to allow integration with 3rd party invoicing, chat and ordering software
    8. Flexible array of real-time reporting features

Goals – development skills

Demonstrate web development proficiency across the following areas.

To view commits and code in Github, let me know your username so I can invite you as collaborator on private repository.

Progress as at May 2020:

● ○ ○	preliminary research
● ● ○	basic implementation
● ● ●	thorough implementation

1. Planning (this document)
    1. concept articulation & development ● ● ○
    2. user stories, sketches & wireframes ● ● ○
    3. data modelling ● ● ●
2. Setup
    1. Vue CLI ● ● ●
    2. version control (Git) ● ● ○
    3. Test coverage ○ ○ ○
    4. Deploy to web ● ○ ○
3. Vue
    1. folder structure ● ● ○
    2. naming convention ● ● ○
    3. modular & elegant code ● ● ○
    4. well commented code ● ● ●
    5. appropriate data structures ● ● ○
    6. logical routing system ● ● ○
    7. simple navigation system ● ● ○
    8. scalable state management ● ● ○
    9. fast & reactive app ● ○ ○
    10. mix of libraries, JS, HTML & CSS ● ○ ○
4. Persistent storage
    1. in-app first (via PouchDB) ● ○ ○
    2. synced to cloud (via CloudDB) ● ○ ○

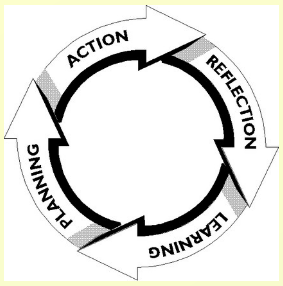
Beans

Local Food Buying Groups made EASY

Features

1. Admin
    1. calendar based order cycle management
    2. Role based management of packing teams & others
    3. Import and translate supplier lists to standard ‘beans’ format
    4. Fast multi-level sorting, culling, modifying & replacement of items
    5. Flexible pricing structures
    6. Print-ready exports of various lists (delivery checklist, member orders, packing sheets)
    7. API for interaction with other ordering systems
2. Finance
    1. Auto invoicing with tracking for member balances
    2. Real time reports: sales|surcharge|member-history per order cycle|quarter|year
    3. Bank reconciliation
    4. API for integration with accounting package (Xero etc)
3. Members
    1. Easy online ordering
    2. Reminders & notifications (to order, buy extras, news, call for packers)
    3. ‘Demo mode’ when outside ordering window
    4. Customisable recurring orders
    5. invoice, payment & balance history on demand
4. General
    1. authentication & role-based permissions for access to ordering/admin/finance functionality
    2. onboarding and setup wizards

Approach



- Independent study and development effort.
- Skill up using online training courses.
- Augment courses with independent research & practice.
- Implement patterns within this code project, with copious comments.
- From time to time, step back to reflect and re-orient
- Emergent planning – plan & execute one component or element at a time and from that vantage point, plan the next one (or refactor existing).

- Version control – commit early and often, simply stay on master branch.
- Documentation – update as the project progresses.
- Effort – 3 to 4 days a week ongoing.
- Roadmap – progress through the phases below. No deadlines (skill acquisition & understanding is currently more important than throughput).

Tech stack & environment

This is a Vue-centric application. It's Vue through and through.

- Vue, vue-router, vue-unstated & numerous other Vue libraries
- HTML, CSS, JavaScript are integrated, focussing on Bulma CSS framework
- PouchDB – for browser persistence
- CouchDB – for offline persistence & data syncing
- hosting environment to be determined
- Development on Ubuntu box, synced to laptop
- Visual Studio Code IDE
- Github repository

Phases

Progress as at May 2020:

Preliminaries: ● ● ●

- Broad Planning
- Architecture
- Tech stack
- Version Control
- Initial setup of project
- Basic folder structure
- Navigation model
- Routing model

Scaffolding:

- Look and feel ● ● ○
- State management pattern ● ● ○
- Persistent storage patterns ● ○ ○
- Component development: Making Available ● ○ ○

Development:

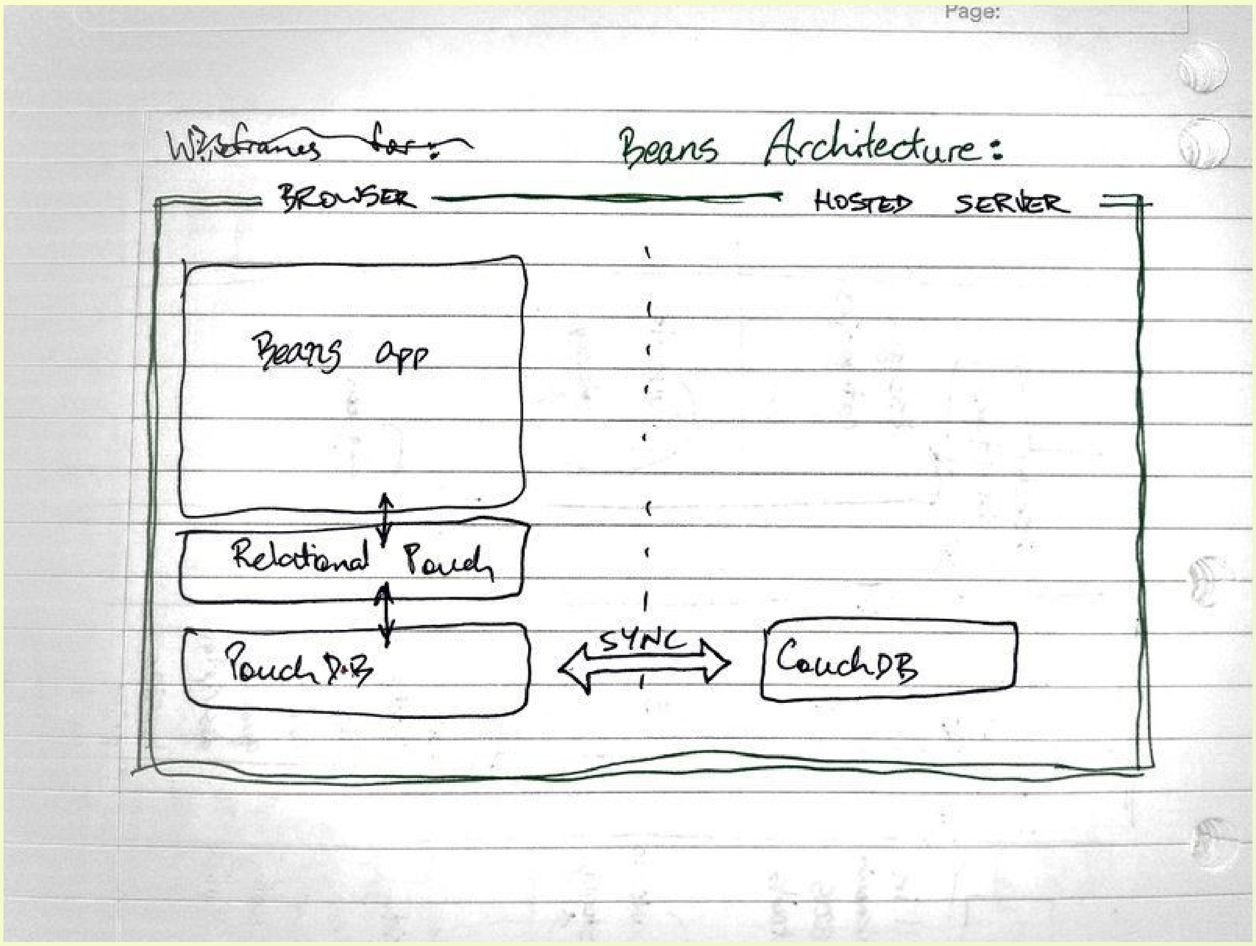
- Admin ○ ○ ○
  - calendar
  - calendar-based order cycle
  - main admin interface
- Authentication ○ ○ ○
  - authentication system with roles & permissions
  - notification system – based on calendar, roles, permissions
- Ordering ○ ○ ○
  - shopping cart for members
  - order → invoice tracking + member balances
  - ordering management via suppliers
- Accounting ○ ○ ○
  - track all transactions
  - export to csv
- APIs ○ ○ ○
  - notification API for integration with Slack or similar
  - availability API for suppliers to list items
  - API for integration with Xero etc

Production:

- Testing & Deployment ○ ○ ○
  - set up testing framework
  - set up hosted solution
  - deploy
- Maintenance ○ ○ ○

Architecture

Data persistence



Note during development the decision has been made to steer away from 'relational' and store data using a 'document store' format. See 'schema.txt' within 'beans' working directory to see what that looks like.

PouchDB is designed to store data in the browser, and to sync with CouchDB over the wire for 'eventual consistency'. This 'browser first' architecture means that once the app is loaded it can run without further connection to the web, but will also sync to a 'dumb' database when connection is available. The aim is to make the app accessible in a paddock or a packing venue even if there is questionable connectivity.

Code structure and patterns

The project is likely to eventually be configured as an SPA/PWA.

More info on folder structure, state management patterns, router info and document schemas are all found in relevant code, comments or text files in the codebase.

Wrap

The ‘beans’ project is an ongoing labour of love by which I hope to build a better app for our own food group, as well as acquire foundational skills for a career as Vue-centric web developer.

Call me if you wish to get in touch or would like to view the codebase.

Russell Austerberry
0423 860 848
rausterberry@gmail.com