Samuel Cousin
5371 Waverly
Montreal, QC H2T2X8
hello@samuelcousin.com

DTLJ / FOOD SESSIONS APP

19th April 2016

OVERVIEW

Application that allows users to record short textual entries of a meal in real time while they listen to a curated soundscape. These entries will be saved to a database, and available to be displayed and updated in real time. An admin panel will allow control over the entries (delete, edit), enable DTLJ to add new restaurants with associated URLs and content, search & filter entries and survey the app stats.

KEY TAKEAWAYS

1. 168 to 210 work hours
2. A minimum of 5 releases milestones and two QA milestone
3. A minimum of 13 feature sets and about 49 draft features
4. Suggest to use Heroku as the app platform

GOALS

1. Maintainability (i.e. documentation, best practices, scalability and simplicity)
2. RESTfulness (i.e. incorporate an API with CRUD endpoints)
3. Reactivity (i.e. views update quickly)
4. Responsivity (i.e. device agnostic)

SPECIFICATIONS

An effort will be made to keep the app codebase nimble and lean. The core components of the Javascript back-end will be **Node.js**, **MongoDB**, **Express** and **Socket.io**

For the front-end, **Bootstrap** will be used along with **Socket.io** and **jQuery**. If the need arise, **Angular** will be used to create a richer user interface.

The build pipeline will be simplified but will include dependencies installation, versioning, linting, concatenation and browser synchronization. This will depend on mostly on **Gulp**, **Git**, **npm**. If time permit, this will be further streamlined by using **Webpack** and **Browserify.**

Once the stack will be firmly decided, a document will be created with specs and instructions for development, testing & deployment. The development process will try to follow a Gitflow methodology and adhere to a semantic versioning scheme.

Below is a provisional list of frameworks that are expected to be used or seriously considered:

Front-end

- Angular (~1.5.X)*
- Bootstrap (~3.3.6)
- Socket.io client (~1.4.5)
- jQuery (~2.2.0)
- Handlebars (~4.0.X)

Back-end

- Node.js (~6.2.X)
  - Express (~4.X.X)
  - Socket.io server (~1.4.5)
  - Mongoose (~4.4.X)
  - Morgan (~1.6.X)
  - lodash (~4.12.X)*
  - Passport (~0.3.X)
- MongoDB (~3.2.X)
- Mocha (~2.4.X)*
- Chai (~3.5.0)*

Tooling

- Gulp (~3.9.0)
- npm (~3.5.X)
- Webpack (~1.13.X)*
- Browserify (~13.0.X)*
- SASS (~3.4.X)*
- Git (~2.5.X)

*to be confirmed

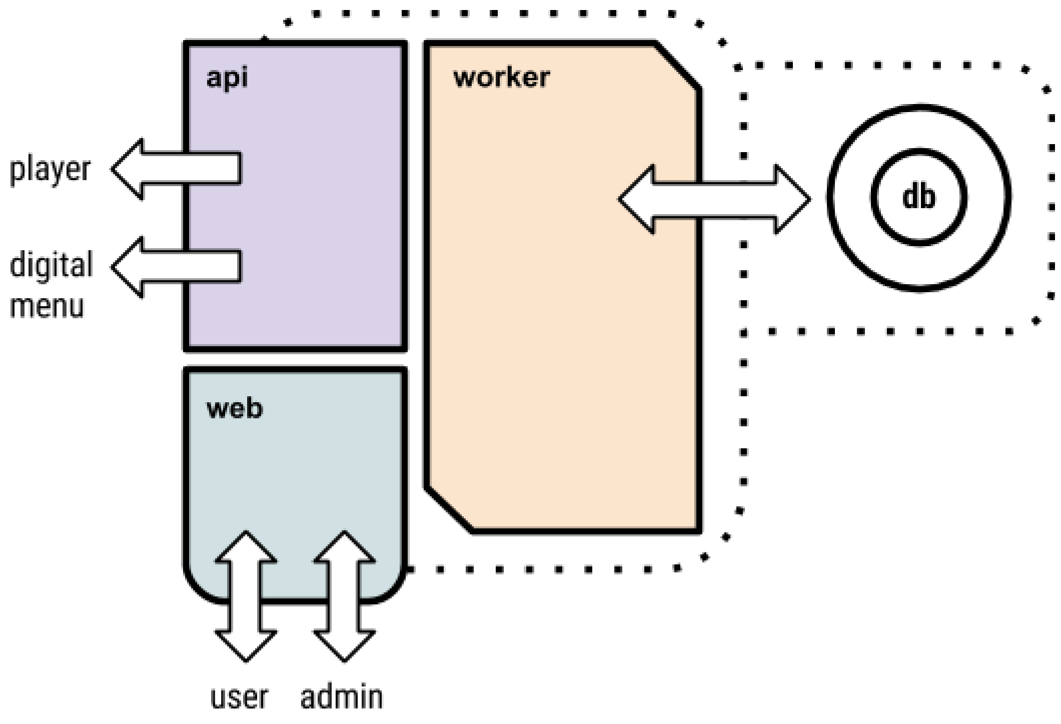
PLATFORM

The app can be hosted either on a regular server with a Node.js runtime environment or off a PaaS provider like Docker or Heroku. There is pros and cons to both solution and they are not mutually exclusive, as something that run in one environment would run on the other with minimal changes. **However, for this particular application I would suggest using Heroku for the following reasons:**

1. The cost would be relatively small ($7-25 USD per month) and prorated (you don't pay when the app is "off")
2. Integrated add-on for our planned database scheme (MongoDB)
3. We can readily test and develop the app on the free tier until we are ready for launch
4. Niceties like a commit to deploy from Git, easy scaling, and app metrics
5. No server/OS to manage, update and maintain
6. The application will not be locked to this platform, and could be run elsewhere if the need arises (EC2, Docker, OpenShift, self-managed server, etc)

As for most web applications that require more than merely rendering static pages, the benefits that come with full control & ownership of a server are offset by the burden of micromanaging and lack of automation.

DIAGRAM



FUNCTIONALITIES

API*

1. With a given location:
    a. Retrieve all entries
    b. Save an entry with its metadata & current state of soundscape
    c. Retrieve the last entry or group of entries
    d. Save the current state of the soundscape
    e. Retrieve the current state of the soundscape
2. Create a new location
3. Delete a location
4. Edit a location
5. Retrieve all locations with details
6. Given an entry id:
    a. Edit an entry
    b. Delete an entry

*API responses will be in JSON format and requests will use HTTP methods

WORKER

1. Apply CRUD (Create, Read, Update, Delete) operations to the DB
2. Sanitize inputs

WEB

1. Serve admin page
2. Serve user page
3. Serve API doc page
4. For development & testing:
    a. Serve digital menu page
    b. Serve player page

DB

1. Store admin credential
2. Store entries
3. Store locations with details

MILESTONES

Milestones (a.k.a. sprints) are there as indicators but their components are fluid and bound to reflect changes in direction, in resource limitation, and technical oversight. See the development board[1] for further details.

CALENDAR

SUN	MON	TUE	WEN	THU	FRI	SAT
			18	19	20	21
22	**23**	24	25	26	**27**	28
29	30	31	01	02	**03**	04
05	06	07	08	09	**10**	11
12	13	14	15	16	**17**	

0.1.0 (23/05)

Release 0.1.0; The app is online; the development environment is functional; the most basic tests are implemented;

0.2.0 (27/05)

Release 0.2.0; The database is connected to the API; the API allows to save entry to the database; the API allows to retrieve a list of all entry from the database; the user page is functional (one way only); development/test player is implemented; temporary digital menu is implemented;

>0.2.0 (31/05)

Soft deadline; Tests are implemented; QA phase on 0.2.0;

0.5.0 (03/06)

Release 0.5.0; The API endpoints are extended; the admin page is functional; the user page is functional;

[1] https://trello.com/b/cQA9OOXq

>0.5.0 (07/06)

Soft deadline; Tests are implemented; review of the app functionalities, performance and appearance; QA phase on 0.5.0;

0.8.0 (10/06)

Release 0.8.0; Add functionalities & sections to admin page; user authentication is implemented;

1.0.0 (17/06)

Release 1.0.0; App is running on production or staging server; final QA; ship it holla & congrats;

TIME ESTIMATE

See the development board[2] for a more granular estimation. Time is estimated in units of workdays that represent a value between 8h and 10h.

Milestone **0.1.0**	:	2 / 2
Milestone **0.2.0**	:	2 / 4
Milestone >**0.2.0**	:	1 / 5
Milestone **0.5.0**	:	3.5 / 8.5
Milestone >**0.5.0**	:	2 / 10.5
Milestone **0.8.0**	:	3 / 13.5
Milestone **1.0.0**	:	4 / 17.5
Overflow (20%)	:	3.5 workdays
Total	:	21 workdays
Minimum hours	:	**168 hours**
Maximum hours	:	**210 hours**

[2] https://trello.com/b/cQA9OOXq