TODO IT ALL

The Job

Our customer wanted a to-do list web application. However, this wasn't to be any regular to-do list. The customer wanted the ability to assign each task a robust set of attributes for detailed filtering. They likewise wanted to share to-dos amongst multiple users and be able to use their to-do list offline.

User Stories

- As a business user, I want to capture ideas/tasks/etc on the go so that I can remember to deal with them all later
- As a business coach, I want to work through my client's braindump list so that we can identify important tasks and support them in making them happen
- As a user, I want to have a clear intention or reason for working on a task so that I am more likely to work on the "right" tasks and finish them
- As a coach, I want my client to have a clear intention or reason for working on a project so that I can help them focus on the tasks that matter
- As a productive tension trained coach, I want to be able to map tasks to the ChangeGrid(R) so that I can quickly see where my client's productive tension and engagement are so that I can recommend support activities so that my client can accomplish their important goals so that I can show clients the ROI from hiring me
- As a business user, I want to be able to filter my tasks quickly so that I can be more likely to spend time on tasks that matter so that I can reach my bigger goals so that I can get that promotion so that I can provide for my family so that they know how much I do for them
- As a mobile user, I want to be able to access my lists when online or off so that I can work from wherever and not fight with connectivity when I am adding to/reading from my list/s
- As a manager, I want to be able to delegate tasks and schedule items from within the app so that I can be more efficient with my time and only have to "handle" that task once
- As a manager, I want to be able to quickly follow up on delegated tasks so that nothing falls through the cracks

- As a user, I want to choose between opportunities so that I can leverage my resources by completing projects so that I can generate additional resources so that I can work on bigger and better projects so that I can help more people
- As a business user, I want to work on tasks that will give me the best ROI for resources invested so that I generate more value so that I can have a better life
- As a user, I want to be able to tell what tasks are high leverage so that I can get more "bang for my buck" so that I can get bigger results with less effort
- As a business user, I want to be able to recognize mission critical tasks so that I can focus on the tasks that are important for reaching my goals so that I can reach my goals more quickly and easily (at all!)
- As a project manager, I want to be able to track what tasks depend on other tasks so that I can prioritize and plan (and maybe draw Gantt-style charts)
- As a new user, I want to create an account and "projects" with lists of tasks so that I can use the platform and be able to group tasks and share those groups with others
- As a new user, I want to be guided through creating my first project and prioritizing task/s so that I can quickly understand how to use the platform and see the difference between it and another arbitrary tool
- As a project manager, I want to be able to move tasks from my default "project" list to other projects so that I can brain dump without worrying about where it belongs, then sort things into different projects/lists when I'm doing my other sorting/ prioritizing steps

Derived Tasks

- Persistent storage of tasks
- Set tasks as active/finished
- User
    - Roles
    - Permissions
    - Sharing
- Filters for tasks
    - Priority high, priority medium, priority low
    - Mission critical, mission incidental, mission optional
    - Income generating, income conserving, income consuming

Effort vs Value (0-8)

Ability vs Challenge (0-12)

- Generate recommended action for each task after filter is applied

  i.e., calculate expected returns for ROI calculations

- Ability to dump, delegate, schedule, and apply additional filters
- Multiple projects for each user with multiple tasks and multiple users
- Delegation of tasks by managers
- Offline first storage

  Both an online and local database.

The Process

Our process began with choosing tools. Since we were all relatively inexperienced with developing web applications, approximately the first 30 hours of our time was spent attempting to launch a basic to-do list with different tool sets. We tried Rails, Meteor, the MEAN stack, etc. By the end of the 30 hours, we had finally settled on using CouchDB for the back end, PouchDB for the local storage, and AngularJS coupled with Angular Material for the front end.

In choosing tools, we briefly spent time going over the supplied user stories. We divided features into individual tasks and gave each a priority. We then highlighted the most critical and made those our focus. To decide who did what, we first experimented with giving specific roles for each team member to fulfill. However, we ultimately ended up with a more loose structure with each of us choosing a specific aspect of the project to work on with occasional assistance from anyone who was willing. This led to a near split with half working on the back end and half working on the front end.

Once tools were picked and a rough plan of attack was in place, the next 30 hours were dedicated strictly to code generation. We announced the completion of features as they were implemented and routinely check in with others to learn of their progress and if there was anything they were stuck on. At certain milestones and as we got nearer to deployment, a “big bang” approach was used to bring features together and grow our application.

The Results

After 60 hours of solid work we have a persistent todo list with basic CRUD functionality. To-dos can be created, marked active or complete, deleted, and edited. All this can be done offline as well and be synced to the database once back online. The requested todo filters were likewise implemented with different views to sort by and edit

each. We had even started work on supporting multiple users but, unfortunately, didn't have time to finish. As it stands, our application is a rudimentary todo app with added fields to narrow down tasks with the greatest importance.

Though we did not meet all our client's requirements, this was expected from the beginning and was made well aware to the client. Our limited time combined with bootstrapping on new tools made reaching their goals extremely difficult. So while we may not have yet implemented everything they want, we feel we have at least completed the essential parts –persistent offline and online storage, CRUD functionally, and the application of task filters.

The Current Features

- A persistent to-do list with CRUD functionality
    - Create tasks
    - Mark tasks as active or completed
    - Delete tasks
    - Edit tasks
- Separate views for adding and viewing todos
- Offline storage
    - Local PouchDB storage syncing to CouchDB backend
- Filters
    - User set priority high, priority medium, priority low
    - User set mission critical, mission incidental, mission optional
    - User set income generating, conserving, consuming
    - User set ability and challenge levels
    - User set effort and value levels
- Separate views to sort by and edit each filter
- User log in
    - Only able to log in
    - No individual user storage

The Planned Features

- Individual user storage

- To-do delegation and sharing

    Permissions

- Projects

    Group to-dos

    Group users

- Comments on to-dos
- Dates for when to-dos were created and when they need to be completed by
- Other statuses for tasks

    i.e., “In Progress” rather than active or finished

- Additional, possibly user created, filters
- Filter actions

Final Thoughts

This project was a learning experience for all of us. Though we each left with different views of how the project went and what we’d do differently, there were some clear areas where we were in accordance. We needed more artifacts, more communication, and more formality.

More engineering artifacts would’ve better ensured that our work met the required specs and at the acceptable qualities. While we did have the user stories and a wireframe of the UI, these were scarcely updated, discussed, or referenced later. The potential needs of our client were greatly overlooked.

While splitting up and working on different tasks definitely helped us save, the returns from such a strategy diminished rapidly towards the end of the project. Our lack of communication and informing each other of the current happenings in the project led to confusion and frustration. We seem to be in agreeance that had we stuck to more paired programming , we not only would’ve had a more pleasant time working on the project, but also would probably have reached more of our goals.

Lastly, we realize that clearly distinguished roles and responsibilities would have been beneficial to our cooperation and productivity. While we did have roles that determined different aspects of the project itself, we didn’t have roles regarding communication among team members. It was also a reason why we took so long to settle on a set of tools. Those working on the back end would select tools beneficial to them without considering the impact they'd have on the front end and vice versa.

Moving forward we believe it would be best to treat our current iteration as a test spike. We believe that it would be in the best interest of the project to start over again with clear roles for the team members on the project. We also believe that enforcing paired programming practices, especially between front-end and back-end team members, would dramatically improve our communication, and by extension the quality of the application. The project thus far has been a great learning experience, and we believe we can take what we have learned and apply it moving forward.