

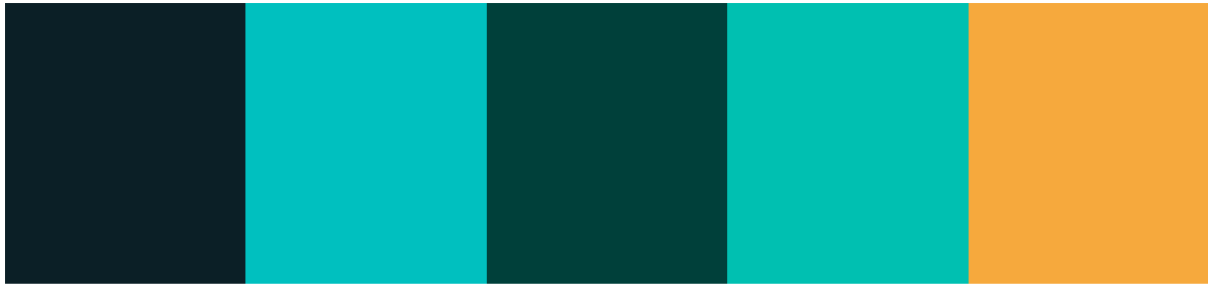
Reference Document for “blueprint”

Styling Rules

Logo Font:

- “Gogh”
- <https://www.dafont.com/gogh.font>

Color References



#0D1F26 

#24BFBF 

#03403A 

#24BFB0 

#F2AB27 

#0D1F26

#24BFBF

#03403A

#24BFB0

#F2AB27

Github Workflow

Branch Naming Convention:

- feature/example-branch-name
- bug/example-branch-name
- styling/example-branch-name
- experimental/example-branch-name
- setup/example-branch-name
- refactor/example-branch-name

Branch deleting:

- Deletion only after Week 11 (during refactoring and cleanup in Week 12 once project is considered complete)

Git Workflow:

- Pull latest version of main to your local machine
- Merge main *into* your own local branch
- Fix merge issues on the local branch on your computer
- Once those merged issue are fixed, push branch to github
- Create a pull request for your branch
 - Get approved by at least one other team member
 - Merge into main

Golden rules of Merging:

- Never push to main from your computer
 - Only merge using pull requests
- Never merge your branch *into* main on your computer
 - Do the opposite, merge main *into* your branch in your computer
- Merges always happen via pull requests

Merge Approval Process

- Must be reviewed/approved by one other team member before merging

Coding Styles

Creating Components Convention

```
export default function TestComponent() {}
```

Coding Rules

- Use “e” for variables referencing events
- Use function declaration for all functions
 - Exception:
 - Functions for map/filter/forEach
 - OnClick or OnChange functions
- Create constants for any type of default values that we may want to easily access to change

```
const NEW_PROJECT_NAME = "";
```

- Error messages for Axios calls

```
function displayServerError(error){  
  console.log("Server Error:", error);  
}
```

```
.catch((error) => displayServerError(error));
```

- Import of chakra at the bottom of all the imports
-

Common Pattern for Updating State:

- <https://docs.google.com/document/d/1LDGI-3MjZud4hVP7wR758k1of7BksljwAzNPW4qjgYl/edit>

Planning

Steps for building App:

- Detailed user stories (review with mentor or lecturer)
- Entity Relationship Diagram (review with mentor or lecturer)
- Tables
 - Create table schemas
 - Create seeds
- Setup project folder for backend (Node and Express) and frontend (React)
 - Pair programming so all members are well versed in architecture
 - Test database setup/reset works
- Detailed wireframes

Project Ideas:

Pablo:

- COVID Dashboard
- Work management platform
- An app that has a collection of games (like hangman, pictionary, brainteasers, challenges, etc)
- Travel site app that can use flight apis/hotel apis to get all that info for destination + maybe destination hotspots? Can send receipt, pay, save flight to google calendar, people can leave comments/reviews for destinations, etc

Vince

- An app to make continuation of programming learning as easy as possible. Every good programmer knows that it's important to stay relevant and keep learning. This would essentially be a "compass" remake but more catered for an individual person to structure their own gathered learning resources for themselves, rather than having a program already built for them by an institution.
- An app so that you can tune-in with multiple friends to watch videos. While a video is playing you can vote on which video to watch next from youtube. The "poll" would be generated from a list of youtube recommendations based on what you're currently all


























watching together. Call it “we-tube” or “popcorn time”? Maybe we also integrate a websocket chat? Or maybe some kind of a google hangouts link generator so people can also be on audio/webcam together while watching a video?

- “PartyWave: surfing the web together”. Similar idea as the one above. Can make it so that users can submit links to any video platform in order for it to be voted-on and watched together? Similar to idea above but looking to spin-off or expand the youtube video watching together idea.
- Drug tracking app for people who take lots of medications - Twilio for texting, etc

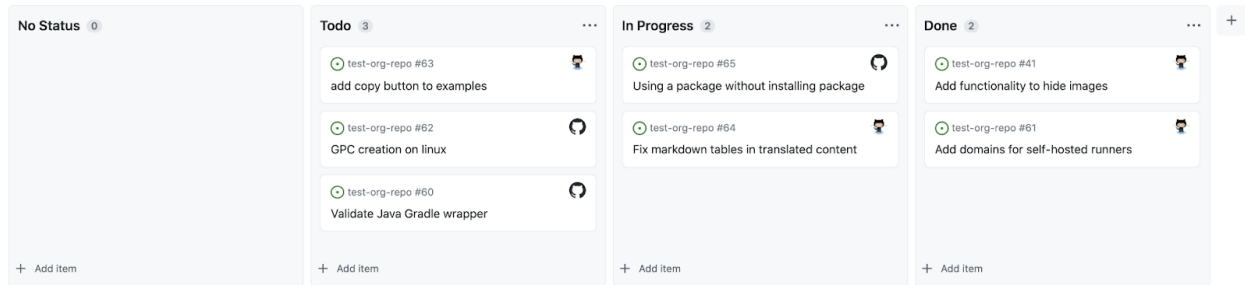
Dylan:

- Finding jobs
 - Matching skills to a job - job board?
- Matching ingredients in the fridge to match recipes to make food - can also keep track of expiry?

List View References:

Q	Title	Assignees	Status	Priority	≡	+
▼	 High 2					
1	 Fix markdown tables in translated content	 monalisa ▼	▼	 High ▼		
2	 Validate Java Gradle wrapper	 octocat ▼	▼	 High ▼		
+	Add item					
▼	 Medium 2					
3	 Using a package without installing package	 octocat ▼	▼	 Medium ▼		
4	 Add domains for self-hosted runners	 monalisa ▼	▼	 Medium ▼		
+	Add item					
▼	 Low 3					
5	 Add functionality to hide images	 monalisa ▼	▼	 Low ▼		
▼ 	 add copy button to examples	 monalisa ▼	▼	 Low ▼		
7	 GPC creation on linux	 octocat ▼	▼	 Low ▼		
+	Add item					
▼	No Priority 0					

Trello Board View References:



Wireframes:

- Dashboard
- View all Tasks (list view)
- View all Tasks (board view)
- View all projects (list view)
- View all projects (board view)
- View/Edit Project
 - (+) sign on dashboard
- View/Edit Task → After clicking on a task from anywhere

Login/Register:

- Make it so you just have 4 or so users that are set (avoid user management) - don't waste too much time

Projects:

- Show only 1-2 projects to show MVP (doesn't need like 5-6 projects to showcase)
- Focus on features people can see

API Privatizations (protecting API):

- Token
- Private API

Name ideas:

- | | |
|---------------------|------------|
| - <u>blueprint</u> | - draft |
| - <u>Bloom</u> | - Craft |
| - <u>Align</u> | - Overtime |
| - <u>Frame</u> | - Gather |
| - <u>foundation</u> | - Forge |
| - Crew | - Aim |
| - Simple | - Target |
| - clear | - ace |

Stretch Features:

- Dashboard
- Deploy
- Mobile responsive
- Websockets
- Gantt chart view - [react-timeline-gantt](#)
- Trello board view - [react-beautiful-dnd](#)/[react-DND](#)/[natural-drag-animation](#)
- Endpoints for APIs
- *Be careful of "feature creep"*
- Different views (calendar vs trello)
- Cypress testing
- Twilio (texting for tasks/projects close to due date)
- Report of some sort (graphs, reports, some kind of metrics)

Overall:

- Focus on polished features
- No obvious bugs
- Make sure all features work as intended
- Need a Unique Selling Point (USP)
 - Web Speech API - can be used to give navigate or add projects/tasks?
 - Need a listener for specific keywords
 - Not as difficult to implement
 - Artificial intelligence?
 - Animation library? Visually different?
 - Accessibility for visually impaired?
- Project needs to be interesting (whether it is technical or not)
- Make one technical aspect interesting (not all recruiters are just HR people)

App functionalities and research results

Tech Stack:

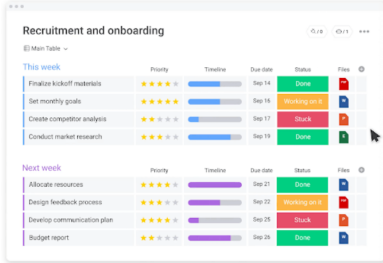
- Frontend - React
 - React development in general isn't done with Ruby/Rails
- Backend
 - Ruby makes it easy to start
 - Not as comfortable
 - Use as APIs
 - No Views/ERB
 - Queries are easier
 - Return as JSON

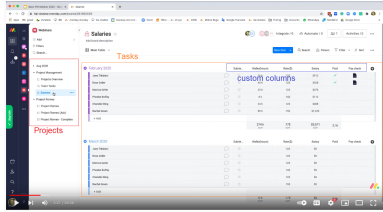
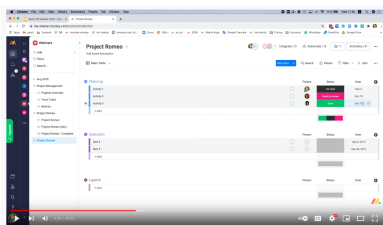
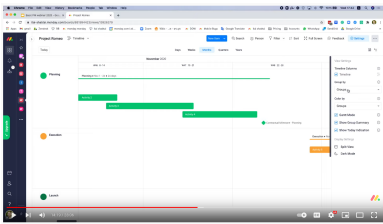
- Node/Express - no version problems
 - More common
 - Better stack for first job **
 - Solidify understanding as we haven't used it recently **
 - Queries in SQL
 - Mentors with node/express experience
- Database - schema does not determine type of database - might be most challenging
 - MongoDB doesn't gain us much but takes a while to learn
 - PSQL
 - AWS SQL DB * (stretch, once done in PSQL move to ElephantSQL/AWS/Oracle)
 - Projects Table
 - Users Table
 - Tasks Table
 - Many to many relationships
 - https://guides.rubyonrails.org/api_app.html
 - Use most updated version of Rails

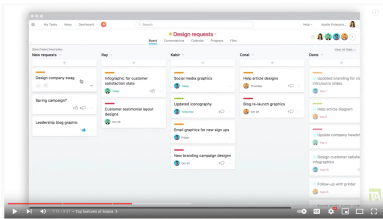
Commonalities of apps from research

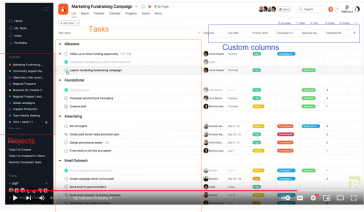
- View by projects
- Add tasks, set deadlines, assign to colleagues, status update options
- Sort/Filter by criteria (due dates, people who are responsible, etc)
- Multiple view options almost always available (calendar, list, trello board, etc)

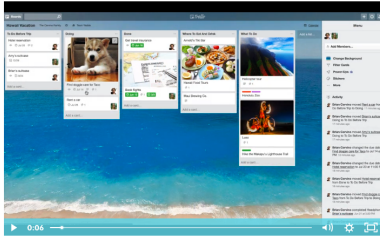
Pablo's Reviews of Project Management Apps:

	Pros	Cons
Monday.com	 <p>Simple layout (file attachment could be stretch feature) - visual timeline may not be the best at conveying information but looks nice</p>	Excess amount of features (widgets, automations, integrations of other apps)

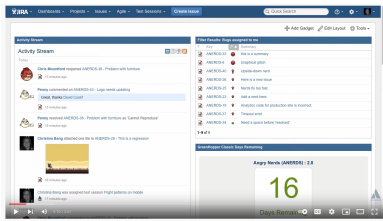
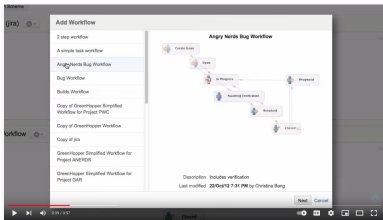
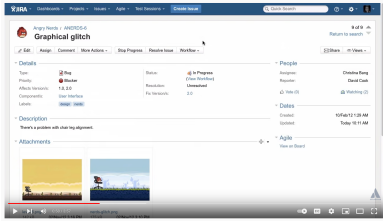
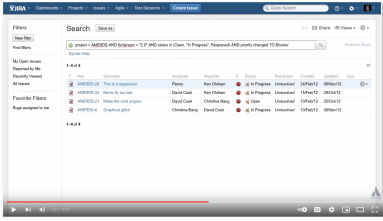
	 <p>HR Management tool</p>	App is apparently slow
	 <p>Project Management tool</p>	Bugs (duplication of tasks, a useful stretch we may consider, for example) causes things to break
	 <p>Display options (calendar/list, grouping options), can also show metrics</p>	Alert system isn't the best
	Need to think of authenticating users + seeing information/projects only available to them	Non mobile responsive

	Pros	Cons
Asana	 <p>Trello board view system</p>	A chat feature would be nice to be able to communicate and somehow integrate work management, but would be a lot of work
	Has form feature to make default task addition template (so everyone has	No time tracking

	the same fields when filling something out)	
	Approvals required for some tasks (a stretch feature) - Approve/Reject/Change Request as answer	Too many features
	Proofing feature that allows people to open up files (especially pictures) and leave comments on the picture	Assignment of tasks are not as easy (and limited options)
	 <p>General list view</p>	Long learning process

	Pros	Cons
Trello	 <p>Simple layout</p>	Has calendar view but otherwise limited in capabilities to show projects
	Design for trello is simple in that their “board” is known by everyone, user interface is mostly graphical	Filtering “boards” (projects) isn't as intuitive
	Mobile accessible	Layout isn't flexible

	Pros	Cons
--	------	------

JIRA	Exclusively for software development	Not general use
	 <p>Dashboard system to see updates (kinda like facebook/instagram statuses)</p>	Not many view options
	 <p>Can make different workflows (instead of Started/In Progress/Completed for example)</p>	No real metrics views
	 <p>Single project views provides lots of information</p>	Slow to load
	 <p>Project search abilities</p>	High learning curve

Commonalities our will require:

- Simple vs features (too many isn't very good)
- Make UI very smooth, nice, clean, user friendly

- Timeline for tickets (different views)
- Organization/team management