Job description

Core Software Engineer

We are looking for a Core Software Engineer to join our engineering team to help build and enhance the UI / UX of the product. Our technology stack is based on NextJS, React, TypeScript and Chakra UI.

Responsibilities

- Build e2e features together with product and backend engineers
- Test features, integration testing, and unit testing
- Build reusable and testable UI components
- Maintainability and ongoing enhancements of the product's infrastructure
- Improve the client side infrastructure

Requirements

- Passionate about HTML/CSS semantics, standard and using modern evergreen web solutions.
- Proven experience (4+ years) working with React for building components for an internal system / component library / customer facing web app.
- Experience (4+ years) with vanilla / ES6+ JavaScript is a must.
- Working knowledge of Node and NPM for building frontend projects using: Webpack, babel, Parcel or Rollup.
- Have frontend design skills: thinking with components, design patterns, composition over inheritance - functional over OOP.
- Strong analytical skills and problem-solving aptitude.
- Attention to details.
- Good eye for design, UI / UX.
- Storybook experience is an advantage.
- Experience with module federation is an advantage.
- Chakra UI is an advantage.
- Product thinking is an advantage.
- Experience working in a monorepo environment is an advantage.
- High understanding of core web vitals an advantage.
- Working knowledge of Post processing CSS languages SASS/Less/PostCSS is a big plus.
- Experienced with SaaS applications is an advantage.

Home assignment

Desired application

- Develop a tasks management application that contains a list of tasks that one user can interact with.

Requirements Exploration

Time estimation

- We suggest you to invest a maximum of 8 hours for this task

What is a must?

- **Please use GitHub**
- **Project build should work on any laptop (Preferred Mac).**
- **Add documentation on how to build and run the application.**
- **Do not invest too much time on DevOps and Microservices architecture, a monolite and a monorepo that is easy to read is the preferred.**
- UI design is important, try to follow **Figma design** at your best (link below).
- Well defined JSON data models is a must
- Use TypeScript Typed models were you can
- Use GitHub issues / Readme to define all the things you would have supported if it was a production ready application (scale, infrastructure, etc.)

What is the techstack that is preferable to be used?

- VS Code
- React
- TypeScript
- NodeJS
- Working with Figma design
- Bonus:
    - Chakra UI (Material UI)
    - Express/NextJS/NestJS for RESTful API
    - Prisma ORM
    - Unit tests
    - Automation tests
    - State management
- **You may use any tech stack that can help you bring the best value in a short time and prove your experience (AngularJS / VueJS / etc.)**

What UI components should be designed and developed?

- Tasks list:
    - List of tasks cards ordered by creation date
        - Each task in the list contain only several fields from the task card
            - Task title
            - Task assignee name
            - Task assignee avatar
            - Task creation date
            - Task status
        - The list should be shown vertically
    - Ability to view more task details when clicking on a task (a task drawer / popup should be opened)
    - Ability to create a new task
- Task creation form:
    - Two steps form
        - First step
            - Should contain only important information
                - Title
                - Assignee name
        - Second step
            - Should contain below listed fields and the ability to correlate to another task
                - Status
                - Description
                - Related Tickets (vertical list of tickets)
                    - Ability to link multiple tasks from existing tasks
                    - Ability to show a list of related tasks
        - **Must: ability to go back to step 1**
        - **Must: ability to skip step 2**
- Task card:
    - Title
    - Assignee avatar
    - Assignee name
    - Status
    - Date created
    - Description
    - Related Tasks
        - Ability to link multiple tasks from existing tasks
        - Ability to show a list of related tasks
    - Watchers tab
        - List of watchers

What are the core features to be supported?

- List **view** of tasks, to enable view existing tasks and **create** new tasks
- Ability to view each task in a more **detailed** view
- **“Related Tickets”** feature in detailed view, should be the main capability and you should invest their your **best efforts**

What pagination UX should be used for the list?

- Nice to have: pagination using “next” button
- Bonus: infinite scroll pagination

Will the application be used on mobile devices?

- Not a priority, but a good mobile experience would be nice.

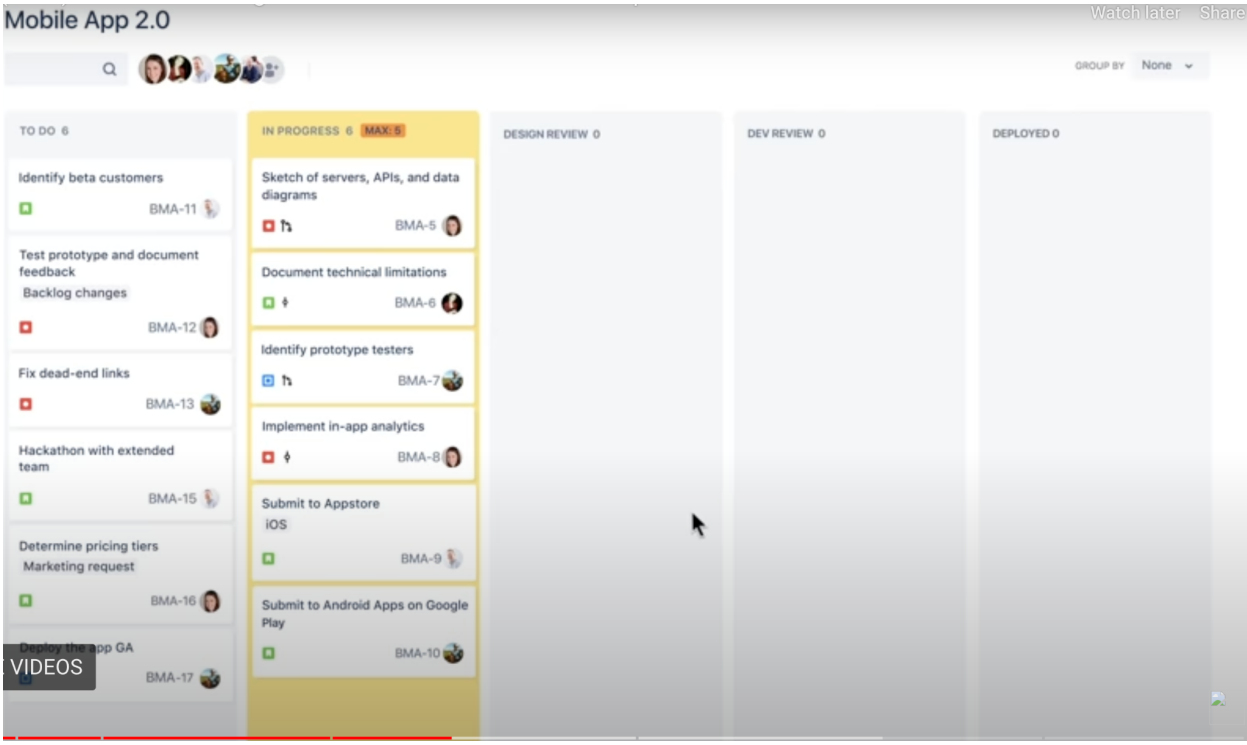
- _____

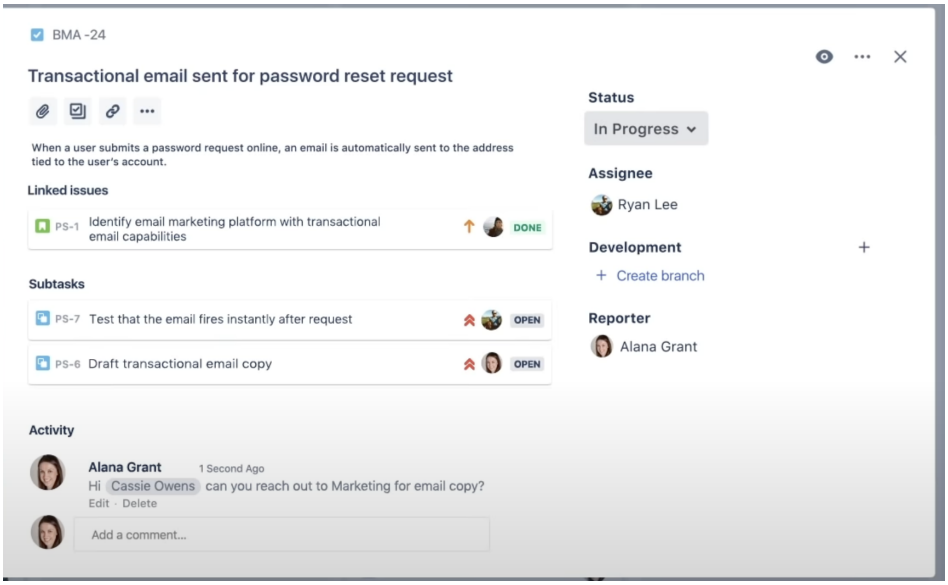
-

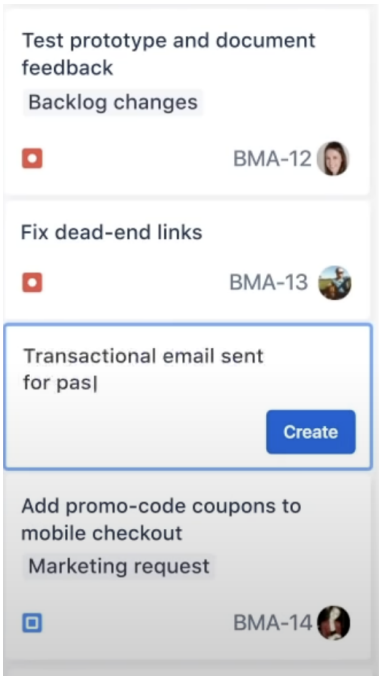
Bonus:

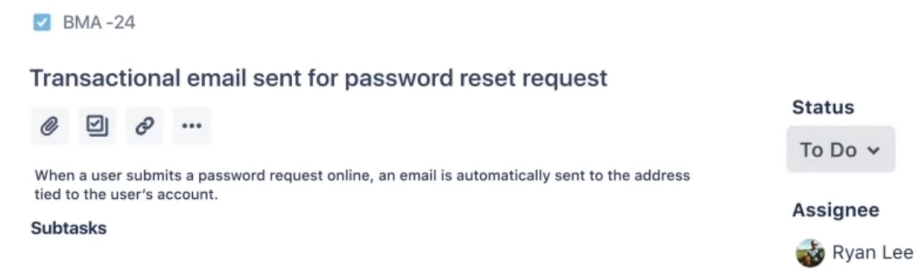
- Backend:
    - Use Prisma or any ORM to store the entities created
    - Connect the app with Slack integration, and choose what user action should send a notification to the slack #channel
        - Example: when a user creates a new task the details around the task are sent to the Slack channel
        - Bonus: Enable the channel to use a command to close a ticket by specifying the task id
- Features documentation (Release notes).
- Deploy the code to Vercel / AWS any Cloud Service so it is easy for us to run
- Filtering tasks by fields
- Order tasks by fields
- support infinite scroll (using virtualization)
- Responsive design
- view details when swiping right
- use keyboard shortcut Command + D + #number to open details view of a task located at index #number
- use keyboard shortcut Command + Shift + T to create a new task
- Ability to duplicate a task
    - Bonus: use keyboard shortcut Command + Shift + D + #number to duplicate a task located at index #number
- Ability to delete a task
    - Bonus: use keyboard shortcut Command + Shift + R + #number to delete a task located at index #number
- Ability to edit a task
    - use keyboard shortcut Command + Shift + E + #number to edit a task located at index #number
- Ability to add more watchers
- Comments Tab (ability to add comments)

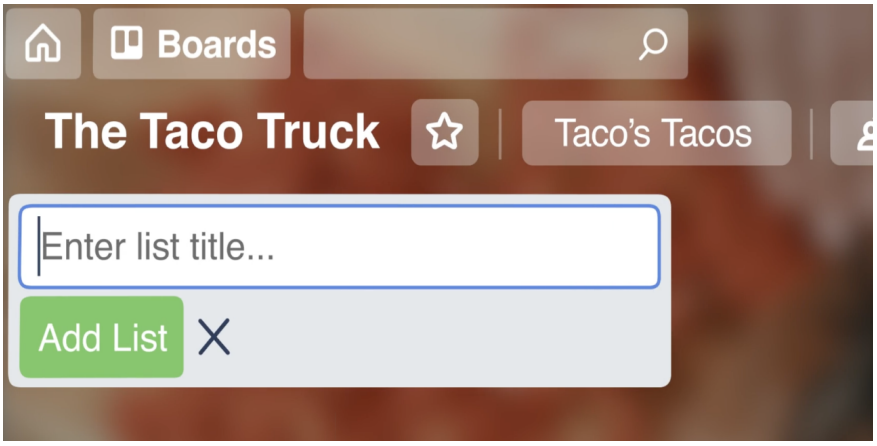
Real-life examples

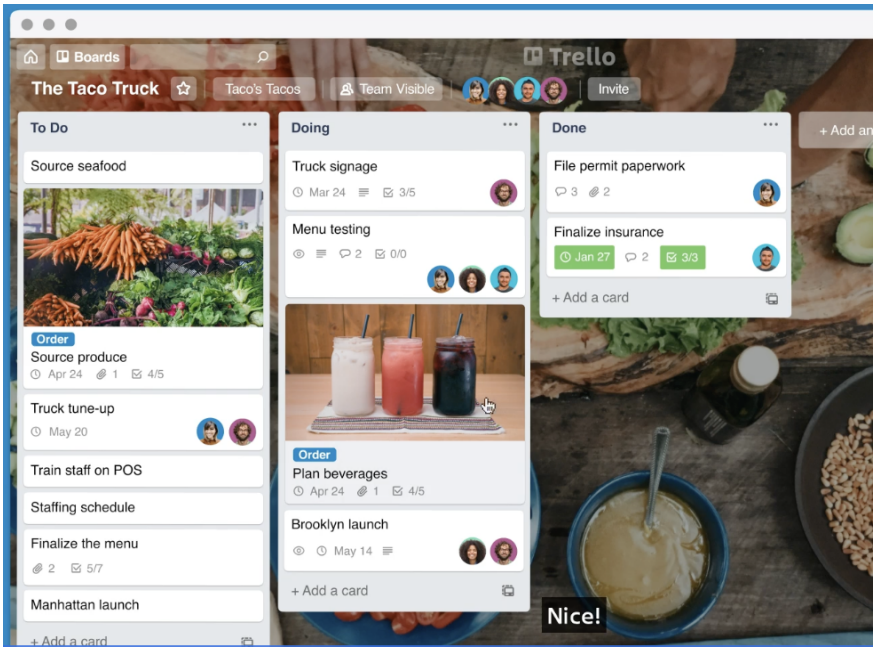
- Jira
  - https://www.atlassian.com/software/jira/guides
  - 

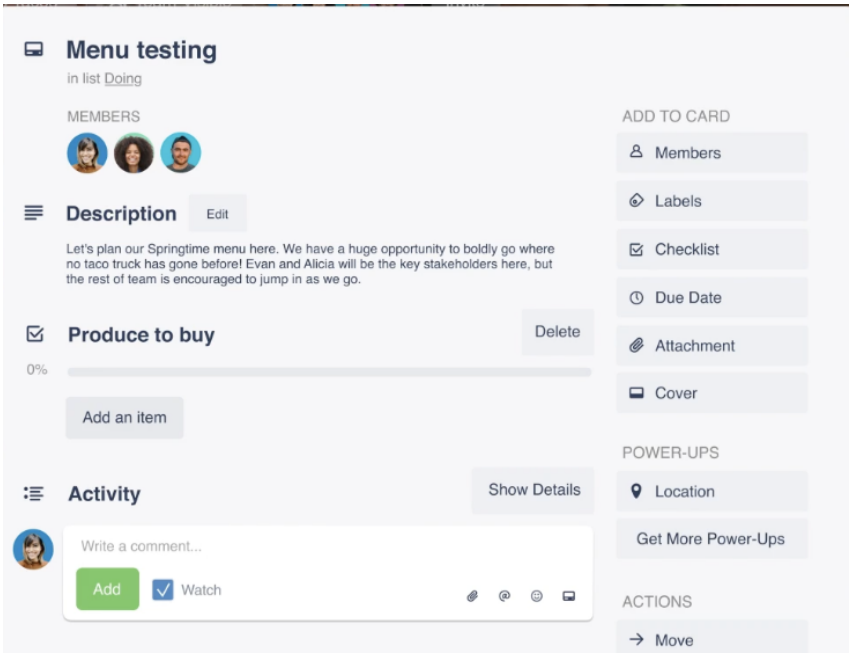
  - 

- 

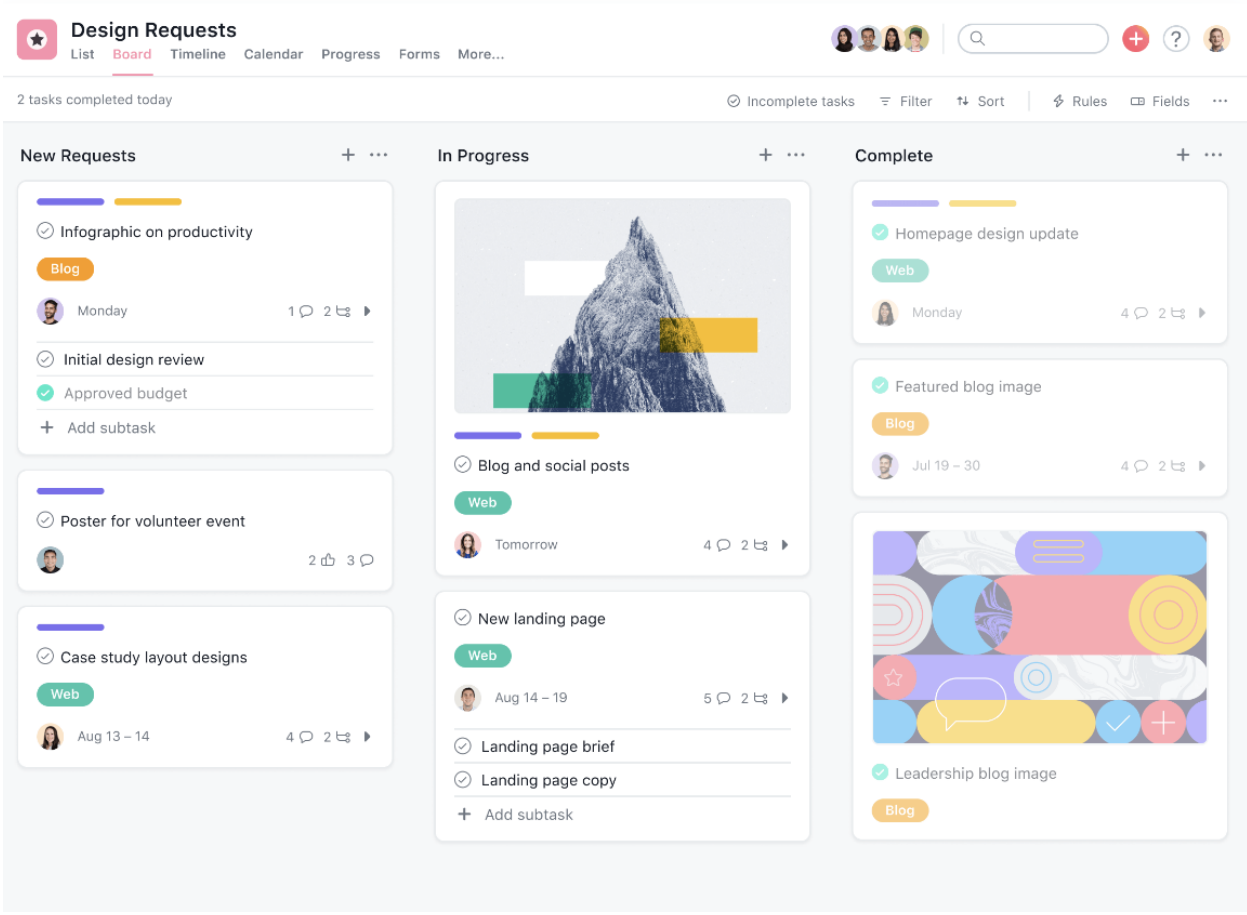
- 

- Trello
  - https://trello.com/tour
  - 

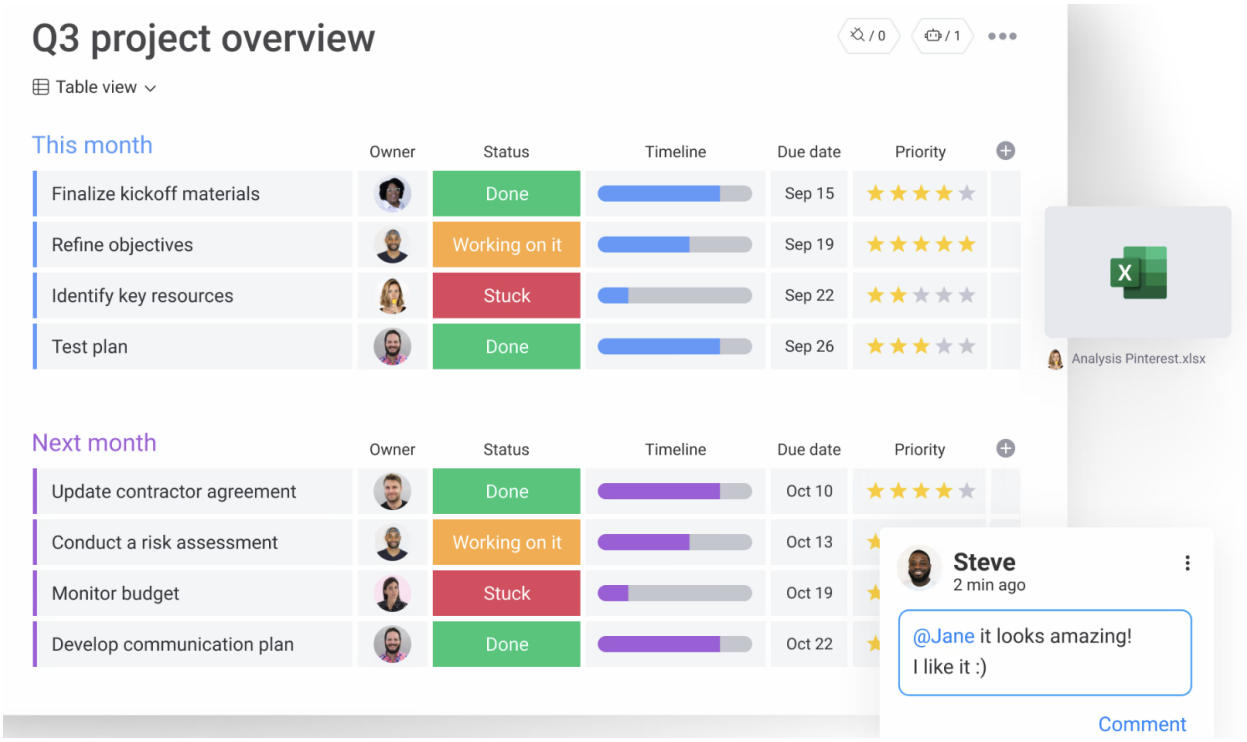
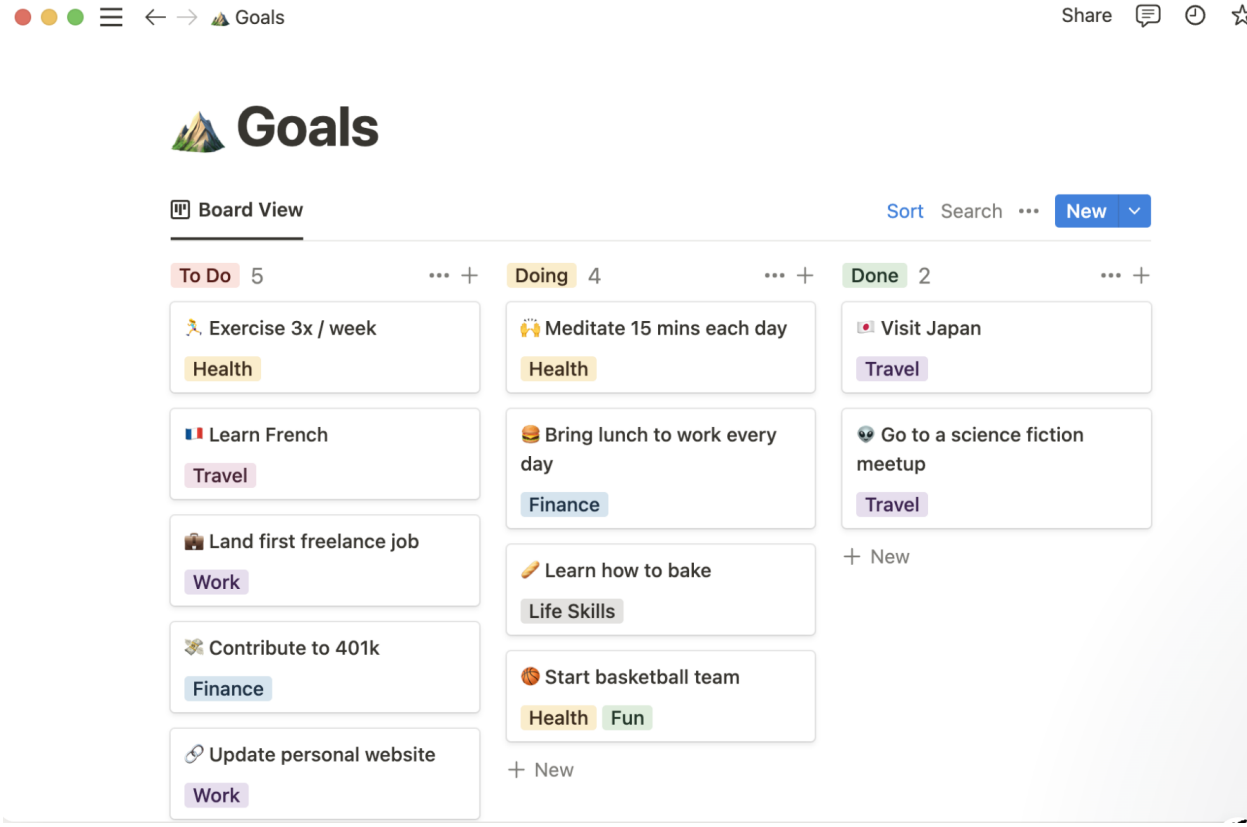
- 

- 

- Asana
  - https://asana.com/uses/kanban-boards
  - https://asana.com/product/timeline



- Monday
  - https://monday.com/
  - https://monday.com/dev



- Notion
  - https://www.notion.so/product