Introduction

This document is the first thing you complete as a part of your Lambda Labs project. It will help you understand your project and plan out a technical design to implement it successfully.

As a team, complete this document and have it approved by the Labs Director or Instructor before beginning work on the code base.

Part 1: Understanding Your Project

Idea

As a developer, you are not directly responsible for the design of your project, but it is very important that you understand it. The best developers are those that can help the team bridge the gap between a skeleton outlining an idea, and a real, written, and usable application.

Write three to five sentences explaining each of the following questions:

What problem does this application solve?

Cadence is an enterprise SaaS application that allows supervisors to schedule their employees' shifts at work. It enables managers to see their employees' availability and paid time off at a glance. It also allows employees to see their own shifts and PTO, as well as to make PTO requests.

Who are your competitors and how do they solve this problem? List at least three.

1. Sling: Sling is a design-led shift scheduling SaaS product priced at $2 / user / month. Its core functionality includes shift scheduling, clock-in and -out, and labor costing. Notably, it includes social features such as chat, newsfeed, and to-do lists.

2. TSheets: TSheets is built for small to medium business to track scheduling and time on the clock. 2-99 users is $4.99 per month with a $16 base fee. 100+ users has a base fee of $80 per month. It's owned by intuit so it integrates with quickbooks.

3. Humanity: Provides easy-to-access employee scheduling app with timeclock, shift trading, availability communication, automation, and more. Rated easiest to use by Business.com. $3/user/month

4. When I Work: When I Work uses an excel formatted sheet that can add new users as employees, schedule their shifts and track over time. It can also forecast your labor costs to keep you within your budget. Free up 75 employees, $1.50 (+1.30 per user/per month) + 10 day scheduling, schedule multiple teams, remote job sites. $2.25 (+$1.60 additional employees) + Manage Team Tasks Assign Tasks to Employees Monitor Task Progress.

5. Shiftboard: Self-description: “Shiftboard transforms complex, hourly workforce scheduling processes to help lower labor costs, improve employee retention, & ensure workplace compliance.” UI is not featured prominently. Also does timesheet and on-boarding. Basic $3/user, Professional $6/user. Seems to be positioning for higher level, broader solutions, also scheduling as a strategic decision process.

Users

Your users are the reason you have a job. By serving their needs, you get paid. As a developer, you can help your designers create “little moments of wonder” as they interact with your product. These touches, such as pictures moving when you swipe through an iPhone’s photo library, will cement your users to you.

To do this, you must understand who they are and how they interact with your application. This understanding will also have a direct impact on many elements of your technical solution, particularly your user accounts, database type and structure, and APIs or services required to meet your users’ needs.

How many types of user accounts will you need for this project? For <u>each user account type</u>, answer the following:

Account Type: Owner / Superuser
Description: Superuser account(s) are used by the business owner(s) in order to perform HR tasks, such as hiring and firing employees.
Needs: Add/remove managers and employees; view, create, edit, save and delete all data on the site.

Account Type: Supervisor
Description: Can create new employees and set the schedule for them.
Needs: View, create, edit, and delete employees information (name, phone etc) and create edit and delete the schedule.

Account Type: Employee

Description: Employees need to be able to communicate their availability in advance, to ensure they can get scheduled without unnecessary conflicts. They can also request time off in advance. The interface allows them to see whether requests and schedules are pending, or have been accepted.
Needs: Can access schedules to which they are assigned, request time off, provide availability. Can change some personal information.

Features

Features are how your app solves your users needs. A Minimally Viable Product (MVP) is the smallest set of features that can deliver a useful experience to your users. It is important that you understand the features in your design including what they are and how your users will interact with them.

List <u>all</u> the features found in your spec and answer the following:

Name: Landing Page
User type: All
Description: Marketing site for existing and prospective users, contains login and registration buttons to access software
Use Case: A prospective user is seeking shift scheduling software so they come to our landing page to learn more about our specific value proposition and potentially register for our product. An existing user wishes to login.

Name: Login Page
User type: All
Description: Performs login authentication, redirects based on result
Use Case: A user wants to login using proper credentials; malicious actors and others using bad credentials will be stopped.

Name: Registration Page
User type: All
Description: Allows for creation of new users, organizations, possibly need roles assigned by owners. Need to figure out who creates accounts?
Use Case: A new employee needs an account, that needs to get created. New owner sets up own account and organization. Supervisors get registered

Name: Schedule Now Button
User type: Mixed
Description: Takes owner/supervisor to schedule page, takes employee to employee page. Only shows up if logged in.
Use Case: A user is logged in and wants to use the scheduling functionality of the app

Name: Hours of Operation Modal
User type: Supervisor / Owner
Description: Hours of Operation Modal - ask Brian what this is.
Use Case: Shows hours if if operation

Name: Left Bar
User type: All (with limitations based on role)
Description: Slide out drawer sidebar that includes user menu/settings options
Use Case: User will have a list of options to choose and navigate through the site.

Name: Top Bar
User type: All
Description: Minimalist top bar containing navigation help (e.g. home button) and sign out
Use Case: A user wishes to navigate back home or to sign out of our application.

Name: Calendar display
User type: All
Description: Employee schedule is shown in a calendar format
Use Case: Show off what is scheduled currently

Name: Calendar Data Entry -- add / delete shifts
User type: Supervisor, Owner
Description: Schedulers can add or delete shifts by dragging and dropping
Use Case: Employers can schedule employees through drag and drop.

Name: Calendar Data Entry -- move/edit shifts
User type: Supervisor, Owner
Description: Schedulers can change shifts by dragging extremities, or dragging entire block. Also a modal for text editing?
Use Case: Employers can schedule employees through drag and drop.

Name: Daily Summary
User type: Supervisor, Owner
Description: Shows the total employees and time scheduled for a day. Also some test for whether a day is fully scheduled / aka done ?
Use Case: Allows a scheduler to see at a glance the important stats for a day

Name: Employee List
User type: Admin
Description: A column listing employees. For each employee, there is a box showing availability and a box showing time off.

Use Case: In the shift scheduler page shows supervisor all of the employees with their availability and time off.

Name: Time off dialogues
User type: Manager, Owner
Description: View all employees and their availability
Use Case: Makes scheduling a lot easier

Name: Availability window
User type: Manager, Owner
Description: Supervisors see an aside menu with the availability of their employees
Use Case: Supervisors pick employees from this list to drop their names for work schedule.

Name: Employee Shift View
User type: Employee
Description: Shows employ shifts
Use Case: Employees will quickly be able to see scheduled shift days.

Name: Admin Employees
User type: Supervisor, Owner
Description: Shows approved time off
Use Case: Supervisor can easily see when employees have approved time off.

Name: Add Employee
User type: Supervisor, Owner
Description: Supervisor or owner can add an employee's details
Use Case: As new employees join, they needed to be added to the shift scheduling platform.

Name: Payment form
User type: Owner
Description: Billing info
Use Case: Pay the bill for using the app

Name: Checkbox for Recurring Payment
User type: Owner
Description: A checkbox in the payment dialog (Stripe integration)
Use Case: Allows owner to select a recurring payment option when checking out.

Name: Input Email / Phone
User type: employee
Description: On the settings page two boxes, one for email and one for phone, where the user can update the values.

Use Case: User can update their email and phone information so that the supervisor can contact them with schedule information.

Name: Select Emails / Text Preference
User type: employee
Description: Two checkmark boxes in the settings page, one for emails and one for text.
Use Case: Employees can select email and/or text, indicating to their supervisor how they would like to review updates on the schedule.

Name: Edit password
User type: Owner/supervisor/employee
Description: Two boxes that allow a user to enter their new desired password, a second box to confirm the password, and a button to submit the password.
Use Case: Allows users to to change their password.

Name: Time off request
User type: Employees
Description: Employee requested time off
Use Case: Employees can see times they have requested off

Name: Pricing confirmation
User type: Owner
Description: pricing confirmation
Use Case: Shows the user how much they are going to be charged

Name: Time Off Approval
User type: Employee
Description: Visual display of whether PTO request is approved, denied, or pending
Use Case: Employees should be able to see the status of their PTO requests, so they can follow up or not in an appropriate manner.

User Story

A user story is what it sounds like. A narrative, written from the perspective of a user that describes an interaction with multiple features so that they may complete a task with the application. A large design team may create a dozen user personas and write stories for them to help design the app. From the perspective of a developer, it is useful to complete this exercise for your core user(s) to better understand their needs.

Write a user story for your most common type of user interacting with the app for its central purpose. Like all good stories, makes sure you include who, what, where, when, why, and how.

Joel is the supervisor of a small, struggling, locally-owned, boutique cafe. He has 6 employees who need to know when to show up for work otherwise there will be no one there to man the store and it will close. That would be sad because he would get fired by the owner. So he needs to be able to manage his employees.

Luckily, Joel just discovered [name of app here] and behold! It has a schedule where he can add his employees. It has lots of cool features where the employees can tell him what time they want off and what their availability is. Joel is elated because now he can use this app to communicate with his employees about when they can work and when he needs them to work.

With this app, Joel is able to manage his employees shifts and keep the store running at all times so that it doesn't close and so that he isn't sad.

Part 2: Technology

The development team is responsible for choosing a technical solution to the product envisioned by the design team. In large companies, this may be a senior engineer or someone with the word 'architect' in their title. In smaller organizations, the developers themselves make the choices. In either case, junior developers will be expected to contribution to these discussions. Each piece of technology must be selected because it has the best balance of pros and cons to solve the given problem.

Because the team knows it, it's easy, fast, convenient, etc. are not acceptable reasons to select a technology.

Front End

Solution: React with Redux & React Router

What problems does this solution solve for this specific project?:

- React abstracts away browser differences in the DOM, and gains performance benefits via the virtual DOM. It has well known compositional patterns and a strong community, while remaining relatively unopinionated
- Redux offers a single source of truth via a persistent store, sophisticated built-in debugging, and enforces prebaked patterns to impose good behavior among developers.
- React Router enables developers to create a single page application. Benefits of SPAs include faster routing between views and smoother transitions & animations. All of this serves to improve user experience.

What are the costs of using this solution?

- Use react pattern
- More expensive on the client side, slower initial load for SPAs in particular
- Dependent on third party libraries, and large npm dependency trees. Mitigated by strong institutional support for React ecosystem, and avoidance of small libraries
- Bad SEO performance
- Will break app if client disallows JS from running

What will you be using for styling and presentation?

Emotion is a high-performance, lightweight css-in-js library. It aims to minimize the runtime cost of css-in-js by parsing styles with Babel and PostCSS. It is 25x faster than styled components when using dynamic props. Kent C. Dodds, the author of glamorous, is considering deprecating it in favor of emotion.

As compared to CSS, emotion will enable us to be DRY. Compared to CSS preprocessing, emotion will enable us to write in a single language. Compared to React component libraries like Material UI, emotion gives us the flexibility to create our own styling without succumbing to the demands of opinionated 3rd party software.

Back End

Solution: Node & Express

What problems does this solution solve for this specific project?:

- Provides a clean way for us to implement authentication, authorization, and CRUD actions to our database.
- Node handles asynchronous as a first class citizen, which is well-suited for handling many requests in a performant way

What are the costs of using this solution?
- Single threading means we will need to explicitly handle scaling issues
- Modularity of npm and reliance on open source can expose vulnerabilities server side and requires diligent maintenance

Database

Solution: Postgres with Knex

What problems does this solution solve for this specific project?:

- Data persistence (especially with regard to hosting on Heroku, which does not persist data in SQLite)
- relational data modeling

- SQL-esque API

What are the costs of using this solution?

- Implementation and testing will take developer time.

What models do you need to represent your data fully in the database in a manner that is logical and consistent?

Pseudocode your models here:
[WORK IN PROGRESS]

Organization

- id
- name
- description

User

- id
- role (owner, supervisor, employee)
- First name
- Last name
- Password
- email
- phone

Availability

- user
- day
- start time
- end time

Time off requests

- date
- user
- reason
- confirmed/pending/denied

Events / Scheduled shifts (including time off)

- begin time
- end time
- date
- user

Deployment

Solution: Netlify & Heroku
What problems does this solution solve for this specific project?

- Enables us to put our project out into the world so that people can use it in an accessible, reliable way

What are the benefits of using this solution?

- Cost structure is friendly for initial scaling period, with a predictable growth pattern, and no lock in
- Continuous deployment built in with Github
- Integration testing possible on Heroku via Travis CI
- Heroku has a robust add-ons marketplace
- Heroku provides a dev-friendly layer on top of AWS

What are the costs of using this solution?

- Deployment relies on settings and problem solving specific to each platform, so there is not a fully contained model that can be painlessly transferred
- Implementation must be friendly to the underlying hardware and operating systems available from Heroku and Netlify

Features

For <u>each feature identified above</u>, provide the following information.

Name: Landing Page
What services, APIs, or platforms will you use to implement this feature?:
This will use local assets, Emotion styling, and React.

What are the costs and benefits of using this solution?
This is virtually free, as we have already taken on these dependencies. Costs are the need to implement, but given the customization needed this is equally a feature.

Feature Name: Registration Page / Login Page

What services, APIs, or platforms will you use to implement this feature?:
Firebase for Authentication

What are the costs and benefits of using this solution?
Relying on Google for security, and having a simplified JWT structure that plays nicely with the front and back end. Rely on openness of Firebase to allow users to authenticate via a number of options

Feature Name: Schedule Now Button
What services, APIs, or platforms will you use to implement this feature?:
React, a value in Redux as to whether the user is logged in, Firebase as described above

What are the costs and benefits of using this solution?
Requires no special functionality, aside from checking that the user is logged in.

Feature Name: Hours of Operation Modal
What services, APIs, or platforms will you use to implement this feature?:
TBD, need to talk to Brian

What are the costs and benefits of using this solution?
See above

Feature Name: Left Bar, Top Bar
What services, APIs, or platforms will you use to implement this feature?:
React Pose for animation, emotion for styling, flexbox to lay out elements

What are the costs and benefits of using this solution?
Animations for the left bar will improve UX for the users, and Pose is a developer-friendly animation library. Styling will proceed per best practices for navigational elements.

Name: Calendar Display Presentation and CRUD
What services, APIs, or platforms will you use to implement this feature?:

Feature Name: Time Off Dialogues
What services, APIs, or platforms will you use to implement this feature?:
Vanilla javascript + React. Possible but doubtful use of React Portal.

What are the costs and benefits of using this solution?
This is a straightforward prompt for the user to approve and decline time off requested. The only need is for modal and HTTP post requests, all of which we have with our basic framework and back end.

Feature Name: Employee Shift View
What services, APIs, or platforms will you use to implement this feature?:
Vanilla javascript + React.

What are the costs and benefits of using this solution?
This is the display of data in a structured way, and allowing for ajax requests as needed, which is directly in the React+Redux core competency. No further API/service is indicated.

Feature Name: Feature Name: Requested time off
What services, APIs, or platforms will you use to implement this feature?:
Reusable react component that maps over data from the backend.

What are the costs and benefits of using this solution?
Reusable and not hard coded.

Feature Name: Time Off Dialogues
What services, APIs, or platforms will you use to implement this feature?:
Vanilla javascript + React. Possible but doubtful use of React Portal.

What are the costs and benefits of using this solution?
This is a straightforward prompt for the user to approve and decline time off requested. The only need is for modal and HTTP post requests, all of which we have with our basic framework and back end.

Feature Name: Add Employee
What services, APIs, or platforms will you use to implement this feature?:
React/Redux, Axios, Express backend

What are the costs and benefits of using this solution?
This is essentially an input form that makes a POST request on submit. This is provided by the core choices in our stack and further services would likely result in equal to worse implementation while increasing package size and muddying structure.

Feature Name: Input Email / Phone
What services, APIs, or platforms will you use to implement this feature?:
React/Redux, Axios, Express backend

What are the costs and benefits of using this solution?
This is essentially an input form that makes a PUT request on submit, allowing the user to update their phone number and email. This is provided by the core choices in our stack and further services would likely result in equal to worse implementation while increasing package size and muddying structure.

Feature Name: Select Emails / Text Preference
What services, APIs, or platforms will you use to implement this feature?:
React / Redux, Axios, Express

What are the costs and benefits of using this solution?
This uses tools already provided by React / Redux and Express. Axios will be used to lower surface area of ajax calls and provide error handling support.

A potential stretch related to this feature would use Nodemailer for emailing functionality. This is not trivial to implement, so a well proven library would be desirable. However, this is not indicated in the spec and at this point is not considered a MVP feature.

Feature Name: Edit Password

What services, APIs, or platforms will you use to implement this feature?:
React / Redux, Axios, Express. Potentially Firebase

What are the costs and benefits of using this solution?
This uses tools already provided by React / Redux and Express. Axios will be used to lower surface area of ajax calls and provide error handling support.

Given that we are using Firebase, we will need to consider whether it is even possible to straightforwardly trigger password updates within our site.

Feature Name: Time off request
What services, APIs, or platforms will you use to implement this feature?:
React reusable component that uses Axios to make a post request to the DB.

What are the costs and benefits of using this solution?
Reusable and not hard coded

Feature Name: Time Off Approval
What services, APIs, or platforms will you use to implement this feature?:
This will GET a given user's PTO from the database and display it

What are the costs and benefits of using this solution?
This relies on the core components of our tech stack.

Feature Name: Pricing Confirmation / Payment Form / Checkbox for Recurring Payment
What services, APIs, or platforms will you use to implement this feature?:
Stripe API

What are the costs and benefits of using this solution?
Allows us to take payments from users with a widely used API

Feature Name: Availability window
What services, APIs, or platforms will you use to implement this feature?:

Reusable react component taking data from the backend

What are the costs and benefits of using this solution?
Reusable and not hard coded

Feature Name: Employee Shift View
What services, APIs, or platforms will you use to implement this feature?:
React component, Redux state management, querying data from Node API

What are the costs and benefits of using this solution?
Completely customizable, integrates well with rest of codebase, but needs to be developed and tested, which takes time.

Part 3: Summary

Use the information above to fill out the following table:

Front End	**B**	