Milestone 1: OptApply

Team Unemployed | 03/30

Github Repo: https://github.com/GeorgeZhang744/OptApply/tree/milestone-1-submission
Milestone: https://github.com/GeorgeZhang744/OptApply/milestone/1
Issues: https://github.com/GeorgeZhang744/OptApply/issues

Project Overview & Team Members

Project Overview :

- OptApply is a job application tracking tool designed to help job seekers organize their applications. It simplifies the job search process by providing a centralized hub to track applications, interviews, and deadlines.

Key Features :

- Automated job entry via job application links
- Categorized application status tracking
- Calendar integration for job deadline tracking
- Skill & notes storage for interview preparation

Team Members:

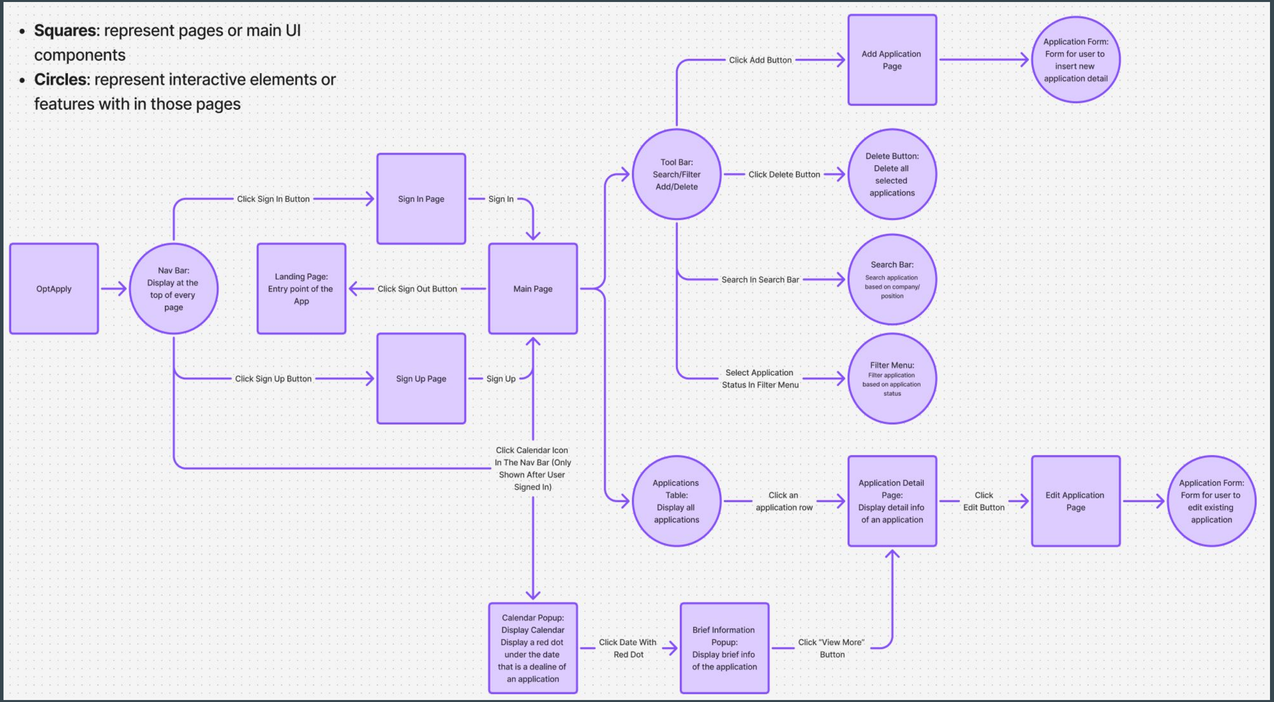
Adam Kaluzny:
- Routing between pages for smooth navigation
- Creation of the Navbar, Sign-up/Sign-in Pages, and Calendar component
- Implementation of signup / login logic with mock data

George Zhang:
- Github Repo management
- Creation of the Landing Page, Main Page, Detail Page, and Adding/Editing Pages
- Optimization of UI performance

Kevin Shang:
- Integration of mock job data into the front end
- Formatting and storing mock job postings to simulate real API functionality

Nick Wierzbowski:
- Styling and responsiveness across all pages
- Ensuring consistent themes

Software Architecture Overview

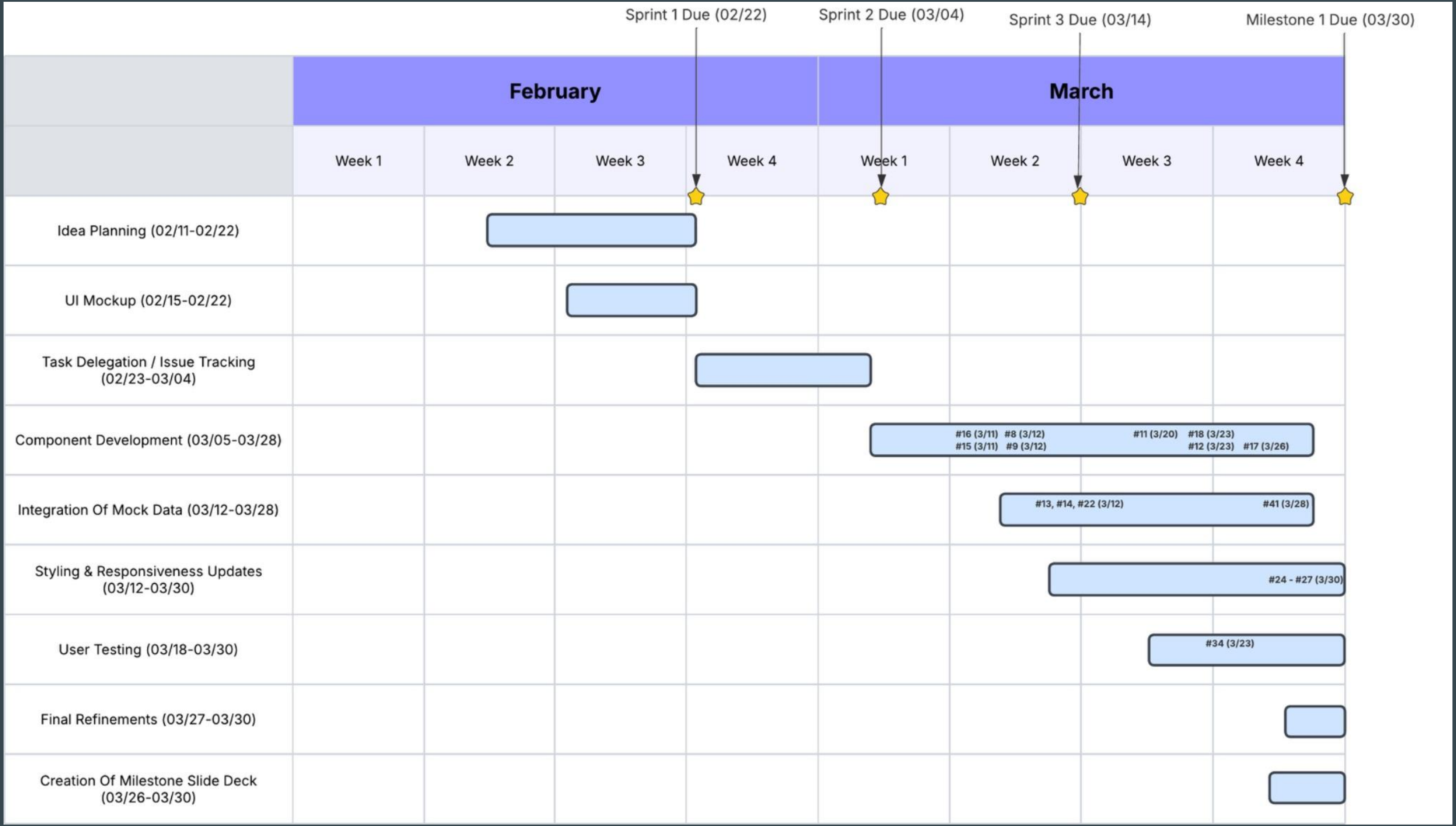


- **Squares**: represent pages or main UI components
- **Circles**: represent interactive elements or features with in those pages

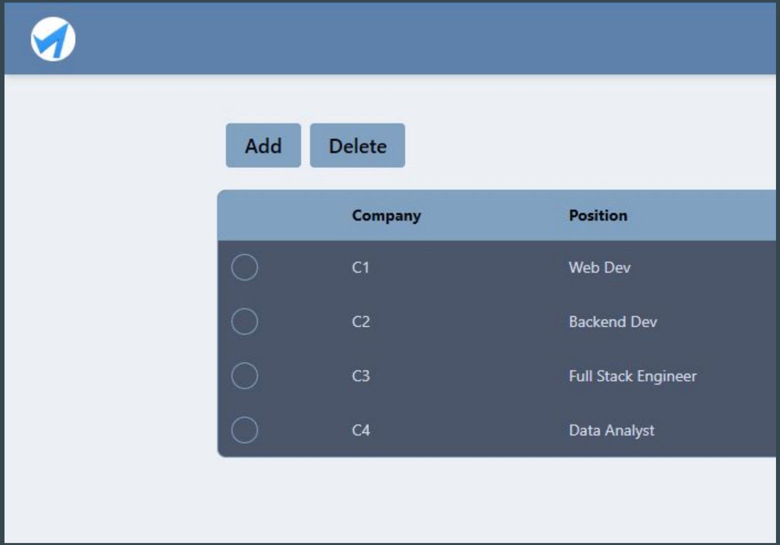
OptApply
Nav Bar: Display at the top of every page
Click Sign In Button
Sign In Page
Sign In
Landing Page: Entry point of the App
Click Sign Out Button
Main Page
Click Sign Up Button
Sign Up Page
Sign Up
Click Calendar Icon In The Nav Bar (Only Shown After User Signed In)
Calendar Popup: Display Calendar Display a red dot under the date that is a dealine of an application
Click Date With Red Dot
Brief Information Popup: Display brief info of the application
Click "View More" Button
Tool Bar: Search/Filter Add/Delete
Click Add Button
Add Application Page
Application Form: Form for user to insert new application detail
Click Delete Button
Delete Button: Delete all selected applications
Search In Search Bar
Search Bar: Search application based on company/ position
Select Application Status In Filter Menu
Filter Menu: Filter application based on application status
Applications Table: Display all applications
Click an application row
Application Detail Page: Display detail info of an application
Click Edit Button
Edit Application Page
Application Form: Form for user to edit existing application

- The flow starts from **Landing Page** (the app's entry point)
- **Nav Bar** connects different pages and is always present
- **Main Page** acts as the core interface, with functionalities including:
    - **Toolbar** (search, filter, add, delete applications)
    - **Applications Table** (displays applications)
    - **Calendar Popup** (shows deadlines)
- **Add Application Page** and **Edit Application Page** allow users to manage applications
- **Application Detail Page** displays the detail info of an application

Historical Development Timeline



Design & Styling Guidelines



	Company	Position
◯	C1	Web Dev
◯	C2	Backend Dev
◯	C3	Full Stack Engineer
◯	C4	Data Analyst

Color Scheme, Typography & Spacing

- Nord-inspired palette: Dark Blue, Light Blue, Red, Snow & Gray.
- Typography: Inter, Arial, sans-serif; scalable font weights & clear line heights.
- Consistent spacing: 4px, 8px, 16px, 32px, 64px.

Layout Principles & Responsiveness

- 12-column responsive grid with 16px gutters.
- Mobile-first design with defined breakpoints (sm, md, lg, xl).
- Clear content separation in cards, tables, and navigation.

Accessibility & WCAG Compliance

- Minimum contrast ratios of 4.5:1 for text.
- Clear focus states and keyboard navigability.
- ARIA labels & compliant interactions per WCAG 2.1.

Temp Link: https://docs.google.com/document/d/115wmBmK20qLrCS8_cTwytofm7p_IqMYXVrlOpCq45S8/edit?tab=t.0

Performance Considerations

Rendering Optimizations:

- **useMemo** : avoid recomputing applications in MainPage.tsx, memoizing selected applications and creating stable initial form in application detail files
- **useCallback** : avoid unnecessary function recreation (toggleApplicationSelection, handleDelete, etc)

React Router:

- Code is written to prepare for lazy-loading routes (to be implemented later on)

Async Mock Data Fetching:

- **setTimeout** : mimics real-world latency, prepares application for backend without changing logic, forces components to support loading states

TailwindCSS:

- Reducing CSS files, avoiding unused styles in production

Assigned Work Summary - Adam Kaluzny

Login / Signup Page Development:

- **PR:** #30 (Initial login/signup pages / navbar component) , #37 (Login / Signup Logic with mock user accounts)
- **Commits:** 0a2f317, 6f8b68e, 79d7b8f, 3e97814, af38d90, 4efa597, 3c3db93
- **Contributions:**
  - Built login / signup pages
  - Created mock user accounts
  - Developed basic login / signup logic and integrated with mock users
  - Established navigation between all these components and others they connect to
- **Issues Resolved:** #15, #16, #20, #41

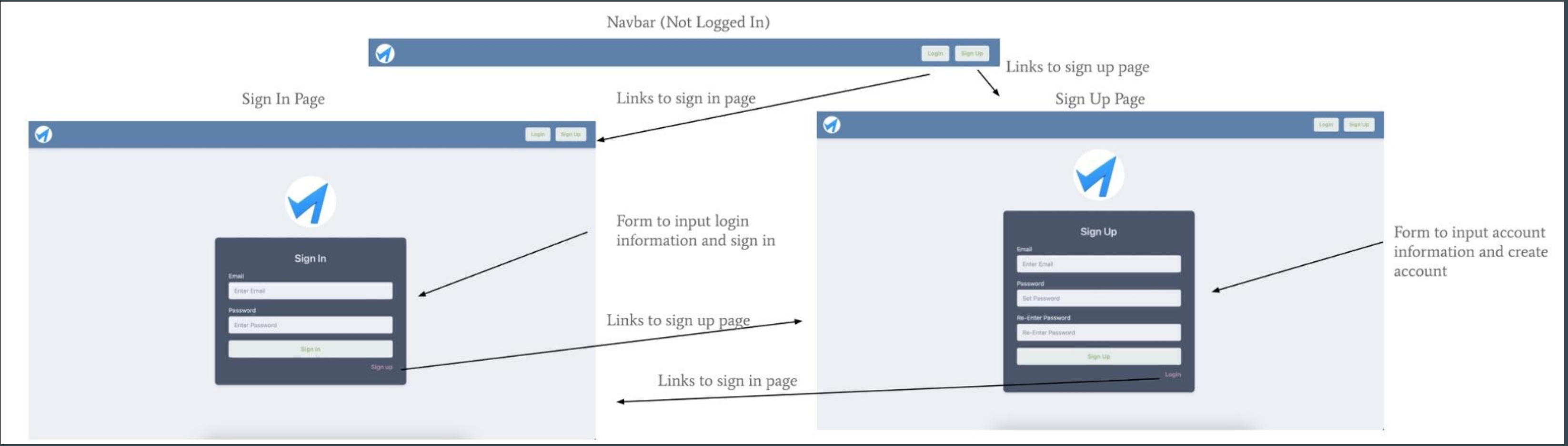
Navbar Development

- **PR:** #30 (Initial login/signup pages / navbar component)
- **Commits:** 0a2f317, d115961, d8ee2db
- **Contributions:**
  - Built navbar component with dynamic display depending on area of site user is on
  - Established navigation between navbar and all other components it connects to
- **Issues Resolved:** #18, #20, #21, #34

Calendar Component Development:

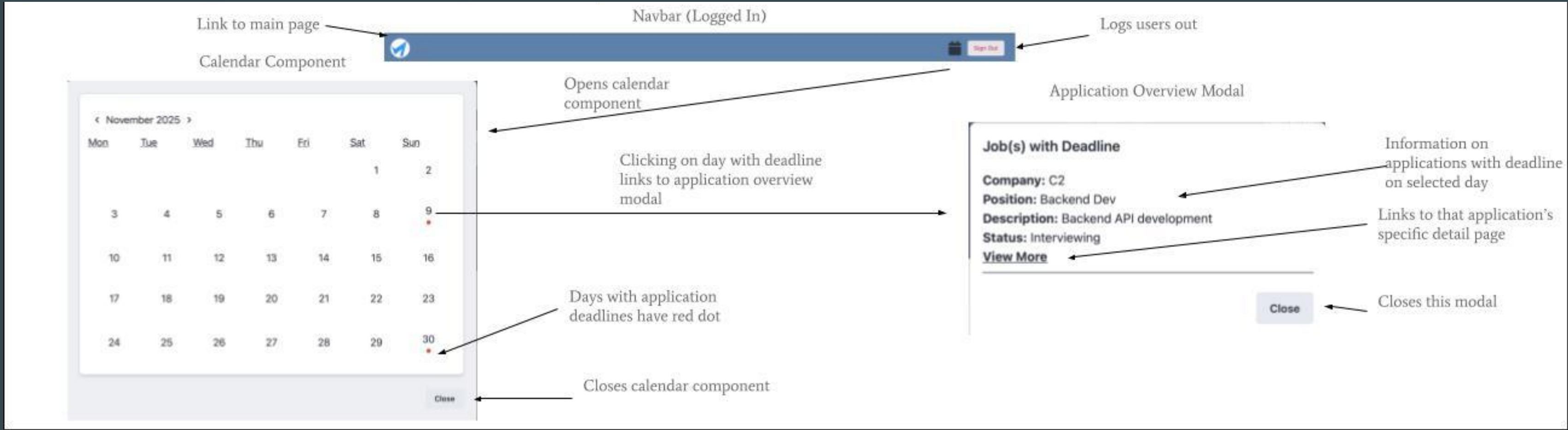
- **PR:** #36 (Calendar component development) , #39 (Calendar UI update)
- **Commits:** 85fc460, bf7416b, 0e274ba, 3d59dc4, e14ece2, 2a6a32a, e5e5031, df6ef1a
- **Contributions:**
  - Built calendar component (includes application overview component seen when clicking on date with application deadline)
  - Integrated calendar component with mock application data to allow user to see months / days with deadlines and view basic information on those applications
  - Established navigation between calendar component all all others it connects to
- **Issues Resolved:** #17

Screenshots & Demonstration - Adam Kaluzny



- Navbar (Not Logged In)
- Sign In Page
- Sign Up Page

- Navbar (Logged In)
- Calendar Component
- Application Overview Modal


Link to main page
Navbar (Logged In)
Sign Out
Logs users out
Calendar Component
Opens calendar component
Application Overview Modal
November 2025
Mon Tue Wed Thu Fri Sat Sun
Clicking on day with deadline links to application overview modal
Information on applications with deadline on selected day
Links to that application's specific detail page
Days with application deadlines have red dot
Close
Closes this modal
Close
Closes calendar component

Job(s) with Deadline

Company: C2
Position: Backend Dev
Description: Backend API development
Status: Interviewing
View More

Code & UI Explanation - Adam Kaluzny

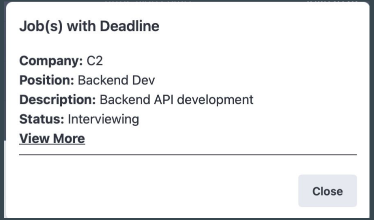
Key Components In Code
- Loads calendar modal when user clicks on calendar icon on navbar
- Renders red dots below months / days with application deadline
- Opens application overview modal when user clicks on date containing an application deadline

Integration With Overall UI Architecture
- Calendar component can be opened via navbar from any page in the application (when user is logged in)
- Application overview modal links to application detail component (when user clicks on view more)

Challenges / Solutions
- Needed to find way to let user know if they had a deadline on any given day
    - Utilized the ‘hasDeadline’ function which takes advantage of the JS array.some function, conditionally rendering a red circle below dates / months with a deadline
- Navigation between months was difficult due to the presence of double arrows right next to the single arrows (around the month name)
    - Looked into react-calendar’s documentation and identified a strategy to remove the double arrows (seen in code on the lines: next2Label={null} & prev2Label={null}

Styling Guidelines
- Stuck to correct color scheme, proper font, proper spacing / sizing, and other related styling techniques, adhering to overall styling guidelines

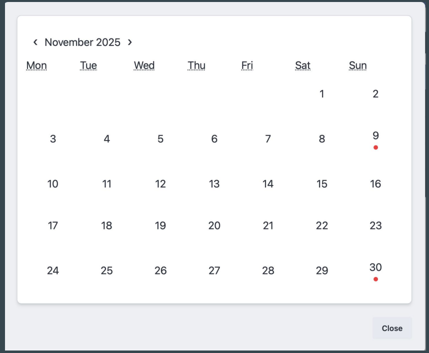
Application Overview Modal

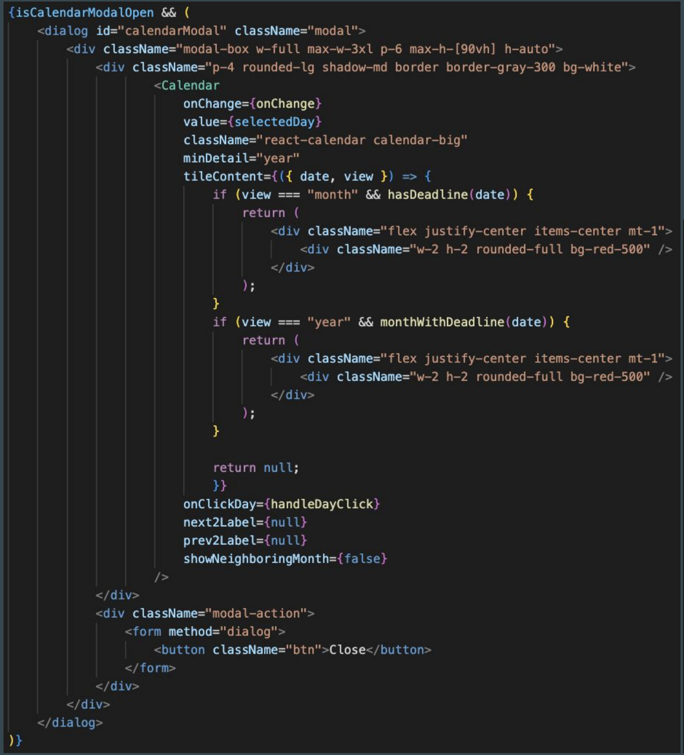


Job(s) with Deadline

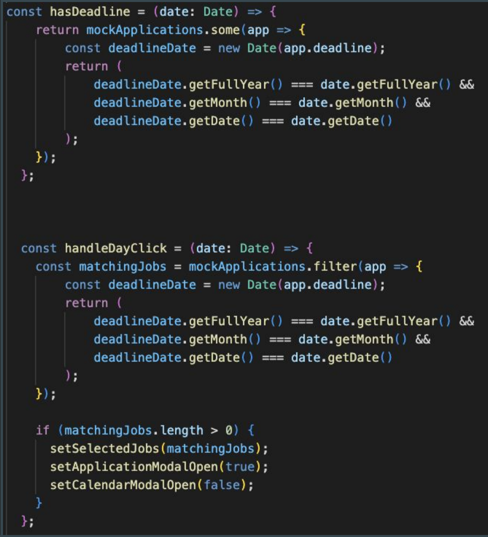
Company: C2
Position: Backend Dev
Description: Backend API development
Status: Interviewing
View More

Calendar Component





```
{isCalendarModalOpen && (
    <dialog id="calendarModal" className="modal">
        <div className="modal-box w-full max-w-3xl p-6 max-h-[90vh] h-auto">
            <div className="p-4 rounded-lg shadow-md border border-gray-300 bg-white">
                    <Calendar
                        onChange={onChange}
                        value={selectedDay}
                        className="react-calendar calendar-big"
                        minDetail="year"
                        tileContent={({ date, view }) => {
                            if (view === "month" && hasDeadline(date)) {
                                return (
                                    <div className="flex justify-center items-center mt-1">
                                        <div className="w-2 h-2 rounded-full bg-red-500" />
                                    </div>
                                );
                            }
                            if (view === "year" && monthWithDeadline(date)) {
                                return (
                                    <div className="flex justify-center items-center mt-1">
                                        <div className="w-2 h-2 rounded-full bg-red-500" />
                                    </div>
                                );
                            }

                            return null;
                            }}
                        onClickDay={handleDayClick}
                        next2Label={null}
                        prev2Label={null}
                        showNeighboringMonth={false}
                    />
            </div>
            <div className="modal-action">
                <form method="dialog">
                    <button className="btn">Close</button>
                </form>
            </div>
        </div>
    </dialog>
)}
```



```
const hasDeadline = (date: Date) => {
    return mockApplications.some(app => {
        const deadlineDate = new Date(app.deadline);
        return (
            deadlineDate.getFullYear() === date.getFullYear() &&
            deadlineDate.getMonth() === date.getMonth() &&
            deadlineDate.getDate() === date.getDate()
        );
    });
  };


  const handleDayClick = (date: Date) => {
    const matchingJobs = mockApplications.filter(app => {
        const deadlineDate = new Date(app.deadline);
        return (
            deadlineDate.getFullYear() === date.getFullYear() &&
            deadlineDate.getMonth() === date.getMonth() &&
            deadlineDate.getDate() === date.getDate()
        );
    });

    if (matchingJobs.length > 0) {
      setSelectedJobs(matchingJobs);
      setApplicationModalOpen(true);
      setCalendarModalOpen(false);
    }
  };
```

Component Interaction - Adam Kaluzny

Sign In Component

- Allows user to sign in to access to main areas / features of the site
- Will interact with backend to check for valid login credentials when user attempts to sign in
- Is linked to navbar component (login button links to this page) and signup page (user can access signup page via link on sign in page and vice versa)

Sign Up Component

- Allows user to make account which they can use to access main features of the application
- Will interact with backend to store new user account information and check if the inputted email is already in use
- Is linked to navbar component (login button links to this page), login page (user can access login page via link on sign up page and vice versa), and home page (user is sent to the home page upon making an account)

Navbar

- Is seen across the top of every page on the site
- Provides the user with access to other components (login, signup, or landing page when not logged in / calendar component, home page, or landing page when logged in)
- Will allow user sign out

Calendar Component

- Can be accessed from anywhere on the application via a button on the navbar (when user is logged in)
- Links to application overview modal when user clicks on a date containing an application deadline
- Application overview modal links to the application details component (if user clicks on the view more link)
- Will interact with backend to obtain user's application list so that red dots are rendered under months / dates with deadlines

Challenges & Insights - Adam Kaluzny

Obstacles Faced

- I had never used Tailwind, DaisyUI, or react-calendar prior to working on this milestone
  - I read the documentation of these frameworks / libraries
  - If I could not find my answer, I would use google to research solutions
  - I got more comfortable with consistent exposure to the frameworks / libraries
- I had never developed a front-end with scalability in mind
  - I utilized lessons learned from lecture, combined with research I did online to implement good practices
  - Worked to try and rid of unnecessary page re-renders

Key Takeaways & Lessons Learned

- Communication is essential for overall success
  - The team must consistently communicate about what each person is currently doing and plans to do each week
  - Incorrectly making assumptions about other team member's plans leads to unnecessary and avoidable confusion
- Documentation is an essential first place to look when searching for an answer to a question regarding a new framework or library
- It is essential to minimize dependencies between team members' concurrent tasks to maintain efficiency
- Always overestimate the amount of time tasks will take to complete
  - Underestimating how long a task takes to complete leads to avoidable and problematic time management issues

Future Improvements & Next Steps - Adam Kaluzny

Future Improvements & Features

- Development of and integration with backend / database
  - Will allow personalized list of applications to be displayed
- User authentication process put in place to prevent users from directly accessing certain areas of site via URL without logging in
- Password encryption established to protect users
- Establish method of automatically extracting job application information given user inputted URL

Technical Debt / Areas To Be Further Optimized

- Could optimize Calendar.tsx by breaking it down into multiple components as there is currently a lot going on in a singular component
- Further minimize unnecessary re-renders to improve performance on large scale
- Utilize performance and load testing on the frontend to identify performance bottlenecks
  - Once identified, improve performance in these areas

Assigned Work Summary - George Zhang

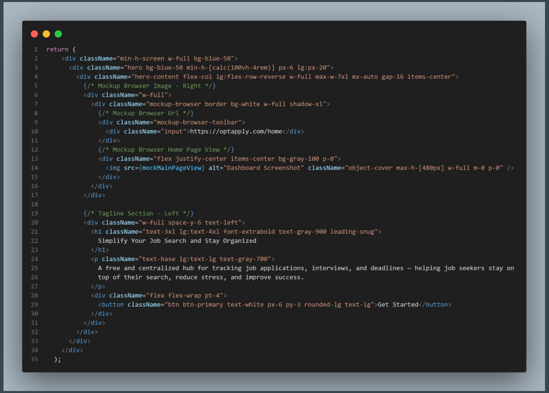
Main Page & Application Detail Page Development:
- **PR**: [#31 - Main Page / Application Detail Page] (Component Development)
- **Commits** : 8c2bebb, 8ca14f0, 83f2cdc, 130974e, 7adb797, bcaf6d7, 12002a0, d4ab78a, e1b3eee, b3d8b1b, df55fa0
- **Contributions** : Built the Main Page to display job applications and the Application Detail Page for viewing individual details. Implemented routing between them, ensuring correct navigation and access to application information.
- **Issues Resolved** : #8, #9, #12

Performance Optimization & Asset Handling:
- **PR**: [#38 - Performance Optimizations and Asset Optimization]
- **Commits** : 6aa4410, 99c2137, e3f1171
- **Contributions** : Improved UI rendering performance using useCallback and useMemo, optimizing state management to reduce unnecessary re-renders. Converted PNG assets to WebP for better compression and faster page loading.
- **Issues Resolved** : #15, #16

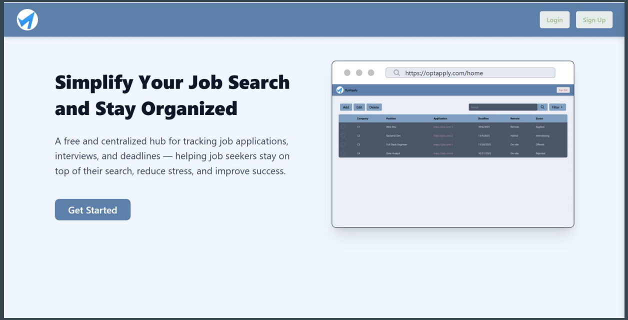
Application Management (Add & Edit Pages):
- **PR**: [#35 - Landing/add/edit pages] (Component Development)
- **Commits** : 2505427, c2522b7, 4da3712, 82e137a, 9ffd1ac, aecb3a3, 3c10713, 9c71e26
- **Contributions** : Developed the Add Application Page with a structured Application Form component and built the Edit Application Page for modifying existing applications. Implemented routing and navigation between these pages to enhance user workflow.
- **Issues Resolved** : #10, #11

Code & UI Explanation - George Zhang

Landing Page:



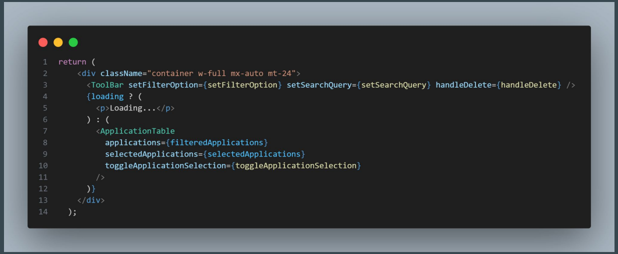
```
return (
    <div className="min-h-screen w-full bg-blue-50">
      <div className="hero bg-blue-50 min-h-[calc(100vh-4rem)] px-6 lg:px-20">
        <div className="hero-content flex-col lg:flex-row-reverse w-full max-w-7xl mx-auto gap-16 items-center">
          {/* Mockup Browser Image - Right */}
          <div className="w-full">
            <div className="mockup-browser border bg-white w-full shadow-xl">
              {/* Mockup Browser Url */}
              <div className="mockup-browser-toolbar">
                <div className="input">https://optapply.com/home</div>
              </div>
              {/* Mockup Browser Home Page View */}
              <div className="flex justify-center items-center bg-gray-100 p-0">
                <img src={mockMainPageView} alt="Dashboard Screenshot" className="object-cover max-h-[480px] w-full m-0 p-0" />
              </div>
            </div>
          </div>

          {/* Tagline Section - Left */}
          <div className="w-full space-y-6 text-left">
            <h1 className="text-3xl lg:text-4xl font-extrabold text-gray-900 leading-snug">
              Simplify Your Job Search and Stay Organized
            </h1>
            <p className="text-base lg:text-lg text-gray-700">
              A free and centralized hub for tracking job applications, interviews, and deadlines — helping job seekers stay on
              top of their search, reduce stress, and improve success.
            </p>
            <div className="flex flex-wrap pt-4">
              <button className="btn btn-primary text-white px-6 py-3 rounded-lg text-lg">Get Started</button>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
```



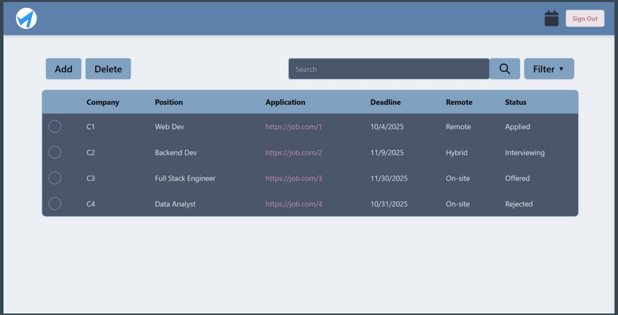
Simplify Your Job Search and Stay Organized

A free and centralized hub for tracking job applications, interviews, and deadlines — helping job seekers stay on top of their search, reduce stress, and improve success.

Main Page



```
return (
    <div className="container w-full mx-auto mt-24">
      <ToolBar setFilterOption={setFilterOption} setSearchQuery={setSearchQuery} handleDelete={handleDelete} />
      {loading ? (
        <p>Loading...</p>
      ) : (
        <ApplicationTable
          applications={filteredApplications}
          selectedApplications={selectedApplications}
          toggleApplicationSelection={toggleApplicationSelection}
        />
      )}
    </div>
  );
```



	Company	Position	Application	Deadline	Remote	Status
○	C1	Web Dev	https://job.com/1	10/4/2025	Remote	Applied
○	C2	Backend Dev	https://job.com/2	11/9/2025	Hybrid	Interviewing
○	C3	Full Stack Engineer	https://job.com/3	11/30/2025	On-site	Offered
○	C4	Data Analyst	https://job.com/4	10/31/2025	On-site	Rejected

Challenges & Future Steps - George Zhang

Challenges :

- Unable to properly implement a mock view of the Main Page in the Landing Page using DaisyUI's mock browser view due to significant styling issues
- Lack of experience with optimizing UI rendering

Solutions :

- Opted for a simpler approach by using an image instead of a coded mock view
- Watched online tutorials to understand how to utilize **useMemo** and **useCallback** hooks properly to reduce unnecessary UI re-render

Future Steps :

- Gain deeper knowledge of React's rendering behavior to proactively prevent performance issues before they arise
- Explore and practice using advanced performance monitoring tools to effectively identify and resolve performance issues.

Assigned Work Summary - Kevin Shang

Setting Up Mock Data in JSON Format:

PR: #13

Commits: f36ab51, 3404df0

Defined sample JSON data for different UI elements and stored mock data in a separate file (mockdata.ts). Fixed minor typos and inconsistencies in typings.

Displaying Mock Data in UI Components:

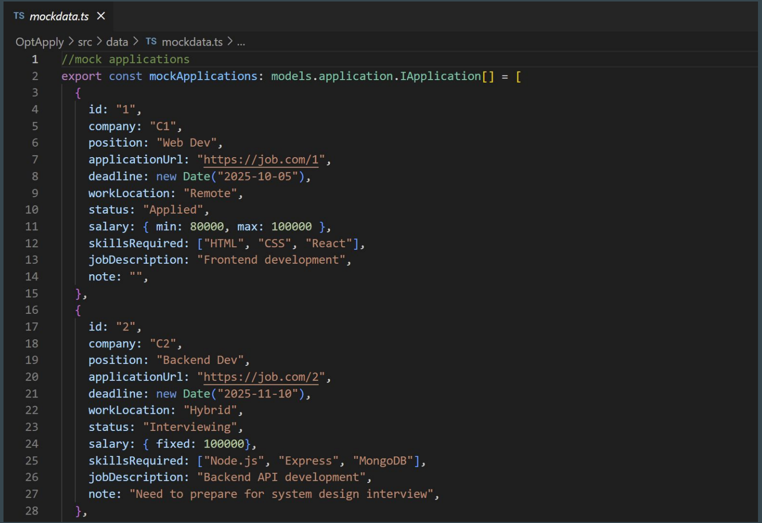
PR: #14

Commits: f36ab51, 3404df0

Updated imports in necessary files, as well as modifying code in existing files to incorporate mock data file instead of hard coded data.

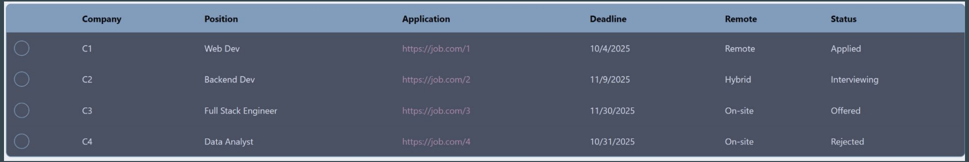
Main page: fetches and displays all applications in a table
Application Detail Page: Shows single application's details based on application's id
Job Card: Displays formatted info about an application

Code & UI Explanation - Kevin Shang

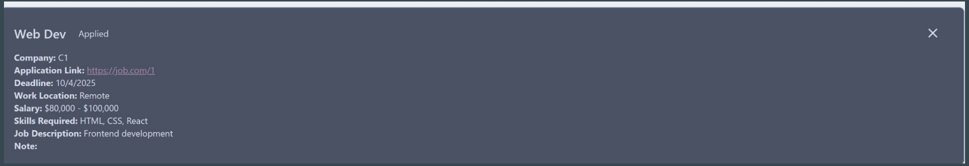


```
//mock applications
export const mockApplications: models.application.IApplication[] = [
  {
    id: "1",
    company: "C1",
    position: "Web Dev",
    applicationUrl: "https://job.com/1",
    deadline: new Date("2025-10-05"),
    workLocation: "Remote",
    status: "Applied",
    salary: { min: 80000, max: 100000 },
    skillsRequired: ["HTML", "CSS", "React"],
    jobDescription: "Frontend development",
    note: "",
  },
  {
    id: "2",
    company: "C2",
    position: "Backend Dev",
    applicationUrl: "https://job.com/2",
    deadline: new Date("2025-11-10"),
    workLocation: "Hybrid",
    status: "Interviewing",
    salary: { fixed: 100000},
    skillsRequired: ["Node.js", "Express", "MongoDB"],
    jobDescription: "Backend API development",
    note: "Need to prepare for system design interview",
  },
```

Mock JSON



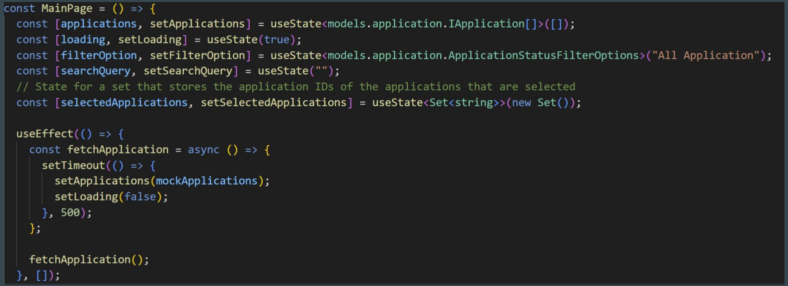
	Company	Position	Application	Deadline	Remote	Status
○	C1	Web Dev	https://job.com/1	10/4/2025	Remote	Applied
○	C2	Backend Dev	https://job.com/2	11/9/2025	Hybrid	Interviewing
○	C3	Full Stack Engineer	https://job.com/3	11/30/2025	On-site	Offered
○	C4	Data Analyst	https://job.com/4	10/31/2025	On-site	Rejected



Web Dev Applied ✕

Company: C1
Application Link: https://job.com/1
Deadline: 10/4/2025
Work Location: Remote
Salary: $80,000 - $100,000
Skills Required: HTML, CSS, React
Job Description: Frontend development
Note:

Job Data Listings



```
const MainPage = () => {
  const [applications, setApplications] = useState<models.application.IApplication[]>([]);
  const [loading, setLoading] = useState(true);
  const [filterOption, setFilterOption] = useState<models.application.ApplicationStatusFilterOptions>("All Application");
  const [searchQuery, setSearchQuery] = useState("");
  // State for a set that stores the application IDs of the applications that are selected
  const [selectedApplications, setSelectedApplications] = useState<Set<string>>(new Set());

  useEffect(() => {
    const fetchApplication = async () => {
      setTimeout(() => {
        setApplications(mockApplications);
        setLoading(false);
      }, 500);
    };

    fetchApplication();
  }, []);
```

Simulating latency and fetching

Challenges & Future Steps - Kevin Shang

- Begin implementing backend and replacing mock data with real fetch requests with API calls.
- Instead of using local storage, instead use a database (i.e, MongoDB) to store application and authentication data.
- Add user-specific data filtering for job applications once authentication is fully implemented, to only display applications for a specific user, and not the entire list.
- Prepare application for handling inconsistent data from API calls

Assigned Work Summary - Nick Wierzbowski

Worked on PR #33: Implemented daisyUI themes

This PR closed issue #2: Apply CSS/Tailwind/Bootstrap or another styling approach

- Implements daisyUI and applied provided css classes to components

Worked on PR #40: Responsiveness

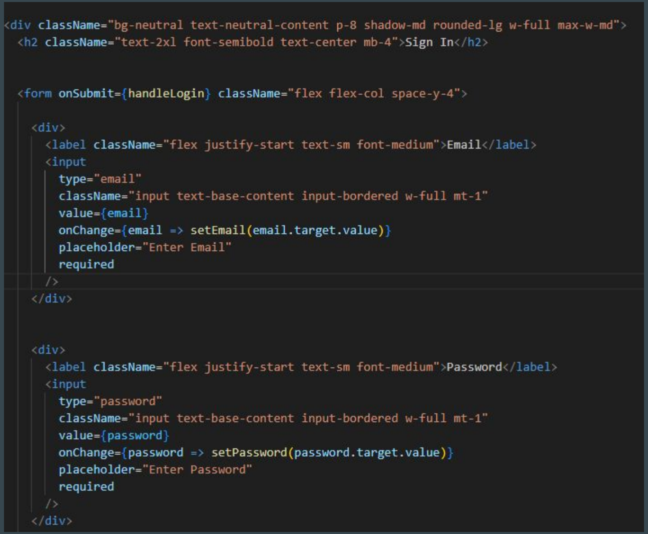
Closes issue #24: Ensure the UI is responsive across different screen sizes

- Uses Tailwind's responsive classes to make changes to the UI based on screen size cutoffs

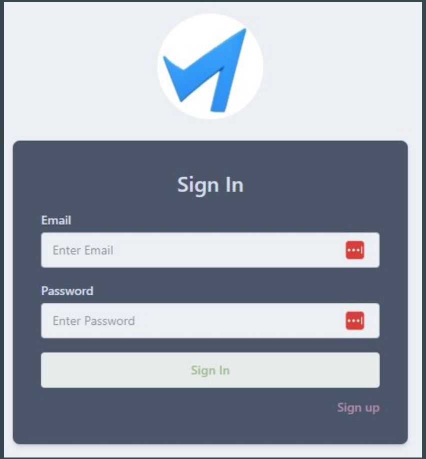
Code & UI Explanation - Nick Wierzbowski

Here is an example of where I applied DaisyUI/Tailwind to give our app consistent styling.

See in className where tags like ‘bg-neutral’ and ‘text-neutral-content’ are applied to the highest level div to apply the proper background and text colors depending on which DaisyUI theme is enabled.



```
<div className="bg-neutral text-neutral-content p-8 shadow-md rounded-lg w-full max-w-md">
  <h2 className="text-2xl font-semibold text-center mb-4">Sign In</h2>


  <form onSubmit={handleLogin} className="flex flex-col space-y-4">

    <div>
      <label className="flex justify-start text-sm font-medium">Email</label>
      <input
        type="email"
        className="input text-base-content input-bordered w-full mt-1"
        value={email}
        onChange={email => setEmail(email.target.value)}
        placeholder="Enter Email"
        required
      />
    </div>


    <div>
      <label className="flex justify-start text-sm font-medium">Password</label>
      <input
        type="password"
        className="input text-base-content input-bordered w-full mt-1"
        value={password}
        onChange={password => setPassword(password.target.value)}
        placeholder="Enter Password"
        required
      />
    </div>
```



Challenges & Future Steps - Nick Wierzbowski

Challenges

- Learning Tailwind and DaisyUI
- Choosing themes that look good and are consistent

Solutions

- Read through Tailwind and DaisyUI websites to introduce myself.
- Apply DaisyUI's variable CSS tags and test all default themes to find a dark and light that fit our requirements.

Future Steps

- Continue testing with our chosen themes and make tweaks to DaisyUI's themes where issues arise.
- Add better styling to application detail page and make Calendar feel more responsive to user input using styling

Future Improvements & Reflections

Future Improvements:

- **Backend Integration** : Implement a real database for storing user data, a authentication system for securing login/signup functionality, and an API gateway server for handling API requests
- **Job Data Automation** : Implement the feature that auto-fills job applications using user-submitted URLs
- **Security Enhancements** : Introduce password hashing, input validation, and restricted route access for better user protection
- **UI/UX Polishing** : Enhance responsiveness and visual feedback for complex components such as the calendar and application details

Reflections:

- **Clear Communication** : We learned that regularly syncing up on each team member's work helped avoid overlap of works and confusion
- **Scalability Mindset** : Building with future maintainability in mind helped reduce tech debt, even at this early stage
- **Starting Early** : We learned that starting early is essential, as it gave us the time to tackle unexpected challenges, improve our work through iteration, and collaborate more effectively as a team