Web Development Internship final project report

Project Title: GitHub Explorer: An Interactive Repository Discovery Platform

Submitted By: Harsh P Malaviya

Institution: Ganpat University

Submitted To: Elevate Labs

Date of Submission: 20th June, 2025

A Journey Through Building the GitHub Explorer App

1. Diving In: What We Set Out to Build

So, the idea was pretty simple: create a slick web app for exploring GitHub repositories. I wanted something that wasn't just functional but actually felt good to use – you know, filtering, checking out details, and easily bookmarking stuff. This report covers how we tackled it, piece by piece, from the ground up, highlighting some of the cool animations and the thinking behind them.

2. Phase 1: Laying the Groundwork – Getting Repos on Screen

Every big project starts with small steps. For GitHub Explorer, it was about getting something, *anything*, on the screen.

- **Initial React Setup:** Kicked things off with a standard React project. Nothing fancy, just the bare bones to get a development environment humming.
- **Fetching Our First Data:** Hooked up to the GitHub Search API. This involved some `useState` to handle the incoming data, loading indicators (so users aren't

left guessing), and useEffect to fire off those API calls when the component was ready. We just wanted to see some trending repos.

- **A Basic List View:** At this point, it was just a simple list. Each item showed the repo name, owner, and star count. Pretty rudimentary, but it was our first glimpse of real data.
- **Setting Up Navigation:** Threw in react-router-dom early on. We knew we'd need multiple pages, so setting up the main dashboard route (/) was a no-brainer.
- **Styling: The First Coat:** Laid down some global CSS in App.css and index.css. This was mostly about getting our dark theme colors right and setting up a basic responsive layout.

3. Phase 2: Bringing the Dashboard to Life – UI, UX, and the Magic of Animation

This is where the real fun began. We wanted the dashboard to feel alive and intuitive, not just a static page. This meant a lot of focus on how things looked and felt.

3.1. The Filter Panel: From Clunky to Clever

The filter panel was crucial for finding specific repos, but it needed to be smart, especially for folks on their phones.

- **First Pass:** Built out the FilterPanel.jsx component with inputs for search, language, min stars, and dropdowns for sorting. On desktops, it was always visible.
- **Making it Mobile-Friendly (and Smooth!):** The static panel wasn't cutting it for mobile. So, we added a "Show/Hide Filters" button. Tapping it would slide the panel in and out.
  - **How We Did It:** Used a React state (isFilterPanelOpen) to dynamically add an open class to the .filter-panel div.
  - **The CSS Trick:** For the smooth animation, we leaned on CSS max-height, opacity, and transform transitions. When open, max-height shot up (like 700px), opacity became 1, and it snapped into place. When closed, max-height went to 0, opacity to 0, and pointer-events: none; was added to ensure no accidental clicks.
- **"Apply Filters" Button Woes:** Ran into a pesky issue where the "Apply Filters" button sometimes got hidden behind the dropdowns on mobile.
  - **The Fix:** A simple but effective solution: bumping up the margin-top on the .apply-filters-button and, crucially, making sure the max-height of the open panel was generous enough to fit everything comfortably.
- **The 3D Hover Effect (on Filter Groups):** This was a subtle touch but added a lot. Each individual filter input group (.form-group) got a cool 3D lift.
  - **The Blueprint:**
    - The parent .filter-panel got transform-style: preserve-3d; and perspective: 1000px; – essential for creating that 3D space.

        - Each .form-group had a base transform: translate3d(0, 0, 0); and smooth transitions for transform, box-shadow, and background.
        - On hover (.form-group:hover) or when an input inside was focused (.form-group:focus-within), it transform: translate3d(0, 0, 3rem); – literally popping off the screen.
        - We even added cascading shadow effects on neighboring elements using clever CSS sibling selectors (&:hover + &, &:has(+ &:hover)). This made it feel like the hovered element was casting shadows on its buddies.
        - For a grand entrance, we made each form-group fade in and slightly pop using staggered @keyframes animations with animation-delay.
    - **Mobile Compromise:** Decided to ditch this specific 3D effect on mobile (@media (max-width: 768px)) to keep things snappy and less visually busy on smaller screens.

3.2. Repository Cards: Polishing the Gems

The individual repository cards are the heart of the dashboard. We tweaked them to look great and feel intuitive.

- **Basic Card Structure:** Started with just displaying the repo's essential info: owner's avatar, name, description, and key stats like stars, forks, and language.
- **Buttons: Alignment & Identity:** Users needed to clearly see and interact with the "Bookmark" and "View on GitHub" buttons. We noticed they could sometimes look messy.
    - **The Solution:** Created a dedicated div (.card-actions) at the card's bottom, using display: flex; justify-content: space-between; and margin-top: auto;. This neatly pushed them to the bottom and spaced them out.
    - **Visual Flair:** The "Bookmark" button got a "filled" look, changing color when active. The "View on GitHub" button became a "ghost" button – transparent background with a border – to clearly signal it's an external link, then filling up on hover.
- **Clickable Areas – Getting it Right:** Initially, clicking anywhere on the card might take you to the detail page. We tightened this up: only the repo name/link and the explicit buttons are clickable, avoiding accidental navigations.
- **The 3D Hover Effect (on Repo Cards):** Just like the filter groups, each .repository-card got the full 3D treatment.
    - **The Same Magic:** We used the same CSS techniques – transform-style: preserve-3d; on the parent grid, and the transform: translate3d, box-shadow, transition, and sibling selectors on the cards themselves – to make them elegantly lift and cast shadows.
    - **First Impressions:** Staggered firstShow and show keyframe animations made the cards float into view as they loaded.

3.3. Dashboard Toggle Buttons: A Unique Interaction

The "All Repositories" and "Bookmarked Repositories" buttons at the top of the dashboard needed their own subtle flair.

- **The Scale and Blur:** Instead of the 3D lift, we went for a "focus" effect. Hovering over one button would subtly scale down and blur its sibling.
    - **The Secret Sauce:** This was all thanks to a clever CSS selector:
    - .toggle-button-group:has(.toggle-button:hover) .toggle-button:not(:hover),
    - .toggle-button-group:has(.toggle-button:focus) .toggle-button:not(:focus) {
    - scale: 0.8;
    - filter: blur(2px);
    - }

    This little gem says: "If the button group has a button being hovered (or focused), then *every other button* in that group that *isn't* hovered (or focused) should scale down by 20% and blur." Simple, but effective!

4. Phase 3: Deep Dive into Repos & Keeping Track of Favorites

The app really came into its own with a dedicated detail page and a way to save those cool repos.

4.1. The Repository Details Page (RepositoryDetail.jsx)

Once you click on a repo, you get to see everything about it.

- **Dynamic Loading:** Used useParams from react-router-dom to grab the owner and repoName right from the URL (like github.com/owner/repo). This made the page dynamic.
- **All the Data, All at Once:** Leveraged useEffect and Promise.all to fire off multiple API calls simultaneously – one for the main repo details, one for its contributors, and another for its issues.
- **A Wealth of Information:** The page displays a ton of info: the full description, owner's avatar, all the key stats, license info, creation/update dates, and clickable topics/tags.
- **Visualizing the Data:** Integrated a ChartDisplay component for some quick insights:
    - **"Top 10 Contributors":** A neat bar chart showing who's been busy.
    - **"Issue Status":** A doughnut chart to quickly see open vs. closed issues.
    - We used Chart.js with react-chartjs-2 to make these happen.
- **Bookmark It!:** The BookmarkButton (more on this below) was right there on the detail page, letting users bookmark or unbookmark directly.
- **Quick Links:** A clear "View on GitHub" button takes you straight to the repo's official page.

- **Personal Notes:** Added a NoteTaking component. The idea was to let users jot down personal notes for each repo (this component manages its own state and saves those notes).
- **Easy Navigation:** A "Back to Dashboard" button, powered by useNavigate, ensures a smooth return trip.
- **Handling the Unexpected:** Built in robust loading indicators and error messages, especially for those pesky GitHub API rate limits.
- **Dedicated Styling:** A RepositoryDetail.css file kept the styling consistent with our dark theme, using CSS Grid for the layout and ensuring it looked good on all screen sizes.

4.2. Global Bookmarking: Your Personal Collection

We needed a way for users to save their favorite repos, and have that status reflected everywhere.

- **BookmarkContext – The Hub:** Created a React Context (BookmarkContext) to hold all our bookmarked repos (bookmarkedRepos) and provide functions to toggleBookmark status and isBookmarked checks.
- **A Handy Hook:** Wrapped the context consumption in a useBookmark custom hook. It just makes things cleaner and easier to use in any component.
- **Saving to Your Browser (localStorage):**
    - When the app loads, we check localStorage first to see if there are any saved bookmarks, so your favorites are always there.
    - Every time the bookmarkedRepos list changes, we automatically save it back to localStorage. It's always up-to-date.
- **Friendly Notifications (Toasts):**
    - Integrated a ToastProvider at the very top of our app. This gave us a global addToast function.
    - Whenever you bookmark or unbookmark something, toggleBookmark fires off a clear message ("Added to bookmarks!", "Removed from bookmarks!") using addToast.
- **The Reusable BookmarkButton.jsx:** This was a hero component. It's totally self-contained, just needs the repo object, and uses useBookmark to figure out if it's bookmarked. It handles the icon change, text, and color automatically.
- **Consistent Button Styling:** A dedicated BookmarkButton.css ensured the button looked and behaved consistently across the dashboard and the detail page, even with its subtle hover effects.

5. Our Toolkit: The Technologies We Leaned On

- **React.js:** Our main workhorse for building the whole UI.
- **React Router DOM:** For navigating between pages effortlessly.
- **CSS3:** This was huge! We pushed modern CSS with Flexbox, Grid, and especially those transform-style: preserve-3d;, perspective, transform: translate3d(), box-shadow, transition, and the powerful :has() pseudo-class for our animations and layouts.
- **GitHub REST API:** The source of all our repository data.

- **Chart.js & React-Chartjs-2:** For those slick contributor and issue charts.
- **localStorage:** Simple but effective for keeping bookmarks persistent.
- **Context API:** Our go-to for managing global state like bookmarks and toasts.

6. The Journey: Challenges & Learning

Building this app was an iterative process, constantly refining and improving.

- **CSS Quirks:** Sometimes, getting specific CSS animations to play nice without breaking other styles was a puzzle. It meant a lot of testing and adjusting.
- **Mastering Complex Animations:** The 3D and the scale/blur effects, especially with :has(), required a deep dive into how CSS transforms and transitions work. Debugging visual glitches could be tricky!
- **Making it Responsive:** Ensuring everything looked good and performed well on tiny phone screens versus big monitors was a constant balancing act. Some animations had to be simplified or removed for mobile.
- **API Limits:** Dealing with GitHub's API rate limits during development was a real thing – we had to ensure our error handling for those 403 responses was solid.
- **State Sync:** Keeping the bookmark state perfectly synchronized across the dashboard, detail page, and localStorage required careful planning.

7. The End Result: A More Engaging GitHub Explorer

Starting with a basic idea, we've transformed GitHub Explorer into a dynamic, user-friendly application. By iteratively tackling UI/UX challenges, integrating rich data visualizations, and crafting engaging CSS animations, the app now offers a polished and intuitive way for users to discover, explore, and manage their favorite open-source projects. It's been a rewarding journey, and the app stands as a testament to what can be achieved with modern web technologies and a focus on user experience.