Technical Presentation: Job Dashboard (10 Minutes)

Slide 1: Title Slide

- **Project Title:** Job Dashboard
- **Your Name(s)/Team Name**
- **Date**

Slide 2: Intended User (1 min)

- **User Profile:**
  - Job Seeker (individual looking for employment opportunities).
  - Assumed Role: End-user interacting directly with the web application.
- **Thought Process:**
  - [Explain *your* thought process for choosing/defining the Job Seeker profile - e.g., course requirements, personal interest, perceived need.]
- **Context:**
  - Assumed Knowledge: User is familiar with general web browsing, job searching concepts (titles, locations, companies), and potentially tracking applications. Needs an efficient way to find relevant jobs and manage their search process.

Slide 3: User's Decision-Making Needs (1 min)

- **Key Responsibilities/Goals:**
  - a. Find relevant job openings based on criteria like location, company, skills, and potentially salary.
  - b. Explore companies posting jobs.
  - c. Keep track of jobs they are interested in or have applied to.

- **Application's Role:**
  - The dashboard helps find jobs via filtering (`JobList` component) and company exploration (`Explore` page).
  - It helps track applications using the client-side state management (`ApplicationsContext`) and the 'My Applications' page.
- **Identifying Needs:**
  - [Explain how *you* determined these specific needs - e.g., brainstorming, user stories, course prompts, common job seeker tasks.]

Slide 4: Data Validation & Preparation (2 mins) - Part 1

- **Data Sources:**
  - Primary Data Structure: `Job` interface (defined in `lib/data.ts`).
  - Key Fields Used: `Title` , `Description` , `Primary_Description` , `Detail_URL` , `Location` , `Skill` (implicitly via extraction), `Company_Name` , `Company_Logo` , `Created_At` .
  - Data likely originates from an external source (database/API, fetched via `getJobsData` in `lib/data.server.ts`), although the exact source isn't specified in the viewed code.
- **Identification Process:**
  - Fields were identified based on typical job board information and features required (filtering, display, application tracking).
- **Validation:**
  - Implicit validation: `getValidLogoUrl` in `lib/data.ts` checks logo URL validity and provides a fallback.
  - Data Loading: Basic validation in `ApplicationsProvider` ensures `localStorage` data is a valid array.

Slide 5: Data Validation & Preparation (2 mins) - Part 2

- **Data Transformation/Computation (from `lib/data.ts`):**
  - Date Formatting: `formatDate` converts date strings to relative time (e.g., "Today", "2 weeks ago").
  - Work Type Extraction: `extractWorkType` determines Remote/Hybrid/On-site from description text.
  - Description Truncation: `truncateDescription` shortens descriptions.

- Salary Extraction: `extractSalary` uses regex to find potential salary information in descriptions.
- Skill Extraction: `extractSkills` uses regex to identify common technical skills in descriptions.
- Statistics: `getJobStatistics` calculates aggregate counts (total, remote, companies, locations, etc.).

Slide 6: User Interface Design (3 mins) - Page 1 (Explore Companies Page)

- **Screenshot/Demo of `app/explore/page.tsx`**
- **Information Presented:** Grid of company cards, each showing Company Logo (`getValidLogoUrl`), Company Name, and number of job listings associated with that company.
- **User Interactions:** Viewing the list of companies. (Intended interaction: Clicking a card to see company details - currently commented out).
- **Meeting User Needs:** Addresses the need to discover companies that are actively hiring.

Slide 7: User Interface Design (3 mins) - Page 2 (My Applications Page)

- **Screenshot/Demo of `app/my-applications/page.tsx`**
- **Information Presented:** Uses the `JobList` component to display jobs previously saved by the user. Shows a message if no jobs are saved.
- **User Interactions:** Viewing the list of saved applications.
- **Meeting User Needs:** Addresses the need to track jobs of interest or applications submitted.

Slide 8: User Interface Design (3 mins) - Core Feature (Job List & Filtering)

- **Component: `components/kokonutui/job-list.tsx` (Used in main dashboard view and potentially other pages)**

- **Information Presented:** List of filterable job cards. Each card shows: Company Logo, Job Title, Company Name, Location (derived), Work Type (`extractWorkType`), Date Posted (`formatDate`). Skills and Salary are used for filtering but not explicitly shown on the card preview.
- **User Interactions:**
  - Filtering jobs by Location, Company, Skill, and Minimum Salary using dropdowns/inputs.
  - Adding a job to the 'My Applications' list using the "Add" button (uses `addJob` and `isJobApplied` from `ApplicationsContext`).
  - Viewing job details (implicitly by clicking the card/link - requires inspecting main page).
- **Meeting User Needs:** Directly addresses the core need to find relevant jobs efficiently and save them for later.

Slide 9: Validation & Verification (3 mins)

- **Verification Rationale:**
  - We verified that the application meets user needs through a combination of methods:
    - **Manual Testing:** Systematically tested core features like job display, filtering (Location, Company, Skill), adding jobs to 'My Applications', and link functionality.
    - **Automated Script:** Utilized a Python script to verify the underlying dataset integrity and consistency.
    - **Cross-Referencing:** Checked displayed data (titles, companies, locations, etc.) against the source data for accuracy.
    - **Functionality Checks:** Ensured dynamic elements like the job count updated correctly after filtering/search actions.
- **Traceability Matrix:**
  - We created a traceability matrix (referencing Requirement IDs A-F) to explicitly link user requirements to the application's implementation and verification.
  - **Example Linkages:**
    - Requirement A (Display Listings) -> UI (Job cards in `JobList`) -> Data (`getJobsData`) -> Validation (Manual check of displayed count vs source).
    - Requirement C (Keyword Search) -> UI (Search input/button) -> Logic (Filtering based on Title/Description) -> Validation (Manual search tests, check filtered results/count).
    - Requirement D (Filtering) -> UI (`Select` dropdowns in `JobList`) -> Data (`Location` , `Company_Name` , `extractSkills`) -> Validation (Manual tests of individual/combined filters, check results/count).
    - Requirement F (Link to Original Post) -> UI ("View Job" link/button) -> Data (`Detail_URL`) -> Validation (Manual click test, verify URL and new tab).

    - This matrix ensures all specified requirements have corresponding features and have been adequately tested.

Slide 10: Q&A / Thank You

- **Questions?**
- **Thank you.**