**Roadmap to Build and Deploy the Placement Preparation Portal**

**1. Develop a Roadmap**

**Step 1: Set up the Development Environment**

* Install IDEs: Choose an IDE (e.g., VS Code for frontend, IntelliJ IDEA for Java backend).
* Install Java, configure Maven/Gradle for dependency management.
* Install MySQL or other databases for storing user, company, and test data.
* Set up OAuth with Google API for user authentication.

**Step 2: Plan and Design UI/UX**

* Use Figma or any UI/UX tool to wireframe the landing page, dashboard, mock test page, company page, and login screens.
* Ensure responsive design using CSS media queries and Bootstrap (if needed).

**Step 3: Build the Frontend**

* Create static HTML pages first to design the layout.
* Use JavaScript to handle button clicks, redirects, and fetch data from APIs (using Fetch or Axios).
* Use CSS to style the dashboard, mock test page, and company details.

**Step 4: Backend Development (Java)**

* Use Java with Spring Boot to handle authentication (OAuth), mock test API, and company details API.
* Use JWT tokens for session management after successful OAuth login.
* Write services for:
  + **OAuth Authentication**: Integrate Google OAuth login.
  + **Mock Test API**: Create API endpoints for fetching test data and results.
  + **Company Details API**: Fetch data for company details and previous papers.
* Connect to MySQL or any database to store user details, mock tests, and company details.

**Step 5: Implement Mock Test Functionality**

* Frontend: Create a mock test page with a timer (using JavaScript setInterval).
* Backend: Write APIs for fetching test questions and storing answers/results.

**Step 6: Connect Frontend with Backend**

* Use JavaScript to fetch mock test data and company details via API.
* Handle state transitions between pages (e.g., after logging in, redirect to the dashboard).

**Step 7: Test, Debug, and Optimize**

* Test OAuth flow, test APIs for companies, and ensure mock test timers work.
* Ensure responsive design and optimize performance.

**Step 8: Deploy**

* Deploy the frontend using Netlify or GitHub Pages.
* Deploy the backend using services like Heroku, AWS, or Google Cloud Platform.
* Set up a database (e.g., MySQL) on a cloud service or locally for deployment.

**2. Files and Directories Structure**

bash

Copy code

/PlacementPrepPortal

│

├── /frontend

│ ├── /assets

│ │ ├── /css

│ │ │ └── styles.css

│ │ ├── /images

│ │ │ └── logo.png

│ ├── /js

│ │ └── app.js

│ ├── index.html (Landing Page)

│ ├── login.html

│ ├── dashboard.html

│ ├── mocktest.html

│ ├── companies.html

│ └── company-details.html

│

├── /backend

│ ├── /src

│ │ ├── /main

│ │ │ ├── /java

│ │ │ │ └── /controllers

│ │ │ │ └── MockTestController.java

│ │ │ │ └── CompanyController.java

│ │ │ │ └── AuthController.java

│ │ │ ├── /resources

│ │ │ │ └── application.properties

│ └── /tests

│ └── MockTestServiceTest.java

│

└── README.md

* **Frontend Files**
  + **index.html**: Landing page post-login, includes dashboard sections for mock tests and companies.
  + **login.html**: OAuth login integration.
  + **dashboard.html**: Displays dashboard with links to mock test and company sections.
  + **mocktest.html**: Displays available mock tests, handles timers.
  + **companies.html**: Lists companies.
  + **company-details.html**: Shows selected company's details and previous year papers.
  + **app.js**: Handles JavaScript for page transitions, API calls, and mock test functionality.
  + **styles.css**: Central CSS file for styling.
* **Backend Files**
  + **AuthController.java**: Handles OAuth sign-in.
  + **MockTestController.java**: API to provide mock test questions and accept responses.
  + **CompanyController.java**: API to serve company details and past placement papers.

**3. Flow of the Website**

1. **Login Flow (Google OAuth)**
   * User lands on the login page (login.html).
   * After sign-in with Google, the OAuth system redirects the user to index.html (Dashboard page).
   * A guest login option also redirects users to the dashboard.
2. **Dashboard**
   * Dashboard has two sections: "Mock Tests" and "Companies."
   * User clicks on the "Mock Tests" section and is redirected to mocktest.html.
   * User clicks on the "Companies" section and is redirected to companies.html.
3. **Mock Test Section**
   * On the mocktest.html page, the user selects a test.
   * After selecting, the user starts the test with a timer displayed on the page.
   * After submitting the test, results are stored in the backend, and the user can view their score.
4. **Company Section**
   * On the companies.html page, 13 companies are listed.
   * Clicking on any company redirects to company-details.html, displaying that company's details, interview experiences, and past placement questions.

**4. Page Interaction Flow**

* **Login Page → Dashboard**: After successful OAuth login, user is redirected to the dashboard (index.html).
* **Dashboard → Mock Test Page**: User selects the mock test section on the dashboard and is redirected to mocktest.html.
* **Mock Test Page → Results API**: User answers questions, submits the test, and the backend API records the results.
* **Dashboard → Company Section**: User selects the company section, which redirects to companies.html.
* **Company Section → Company Details Page**: User clicks a company, and it redirects to company-details.html with data fetched from the backend.