**ITEP 102 – WEB SYSTEM AND TECHNOLOGIES**

**ITEP 206 INTEGRATIVE PROGAMMING / ITEP 204 – ADVANCE DATABASE SYSTEM**

**FINAL PROJECT TEMPLATE**

|  |  |
| --- | --- |
| **GROUP MEMBERS:** | **Cristal, Princess Ashlie D.**  **Flores, Naomi Ginel**  **Ranas, Renz Tyron** |
| **Job Portal: CareerNest** | |
| ***WEB DESIGN:*** | |
| 1. Introduction | **CareerNest**  A job portal that makes hiring and job searching easier for everyone. It’s a one-stop platform for job seekers to discover, apply to, and keep track of opportunities. For employers, it simplifies posting jobs, reviewing candidates, and managing the hiring process—all in one place. Whether you’re looking for your next career move or trying to find the perfect hire, CareerNest helps connect the right people with the right roles, smoothly and efficiently. |
| 1. Planning | **a. Wireframe Design**  **Index**  A screenshot of a computer  AI-generated content may be incorrect.  **Jobs**  A screenshot of a computer  AI-generated content may be incorrect.  **Jobs Details**  **A screenshot of a computer  AI-generated content may be incorrect.**  **About Us**  **A screenshot of a computer  AI-generated content may be incorrect.**  **Contact**  **A screenshot of a computer  AI-generated content may be incorrect.**  **Login**  **A grey paper with text  AI-generated content may be incorrect.**  **Register**  **A grey paper with text  AI-generated content may be incorrect.**  **User Dashboard**  **A screenshot of a computer  AI-generated content may be incorrect.**  **User/Company Profile**  **A grey rectangular object with text  AI-generated content may be incorrect.**  **Company Dashboard**  **A screenshot of a computer  AI-generated content may be incorrect.**  **Post Job**  **A screenshot of a computer  AI-generated content may be incorrect.**  **View Applications**  **A screenshot of a computer  AI-generated content may be incorrect.**  **View Application Modal**    **Schedule Interview**  **A screenshot of a computer  AI-generated content may be incorrect.**  **Admin Dashboard**  **A screenshot of a computer  AI-generated content may be incorrect.**  **Tabs in Admin**  **A screenshot of a computer  AI-generated content may be incorrect.**  **b. Gantt Chart**    **c. Materials**  List the programming languages used   * HTML * CSS * JavaScript * PHP   Mention the IDE   * Figma (wireframes) * VS Code * XAMPP (database) |
| 1. Html structure | **HTML Structure**   * Overview of how the HTML is organized   The HTML of the job portal is organized in a modular structure using PHP includes for key components like the header, navigation bar, and footer. This modularization helps separate concerns, making the layout cleaner and easier to maintain. Each page loads these reusable components while placing its main content inside clearly defined sections for improved structure and consistency.   * Mention and describe:   Use of semantic tags (<header>, <nav>, <main>, <section>, <footer>)  **<header>** is indirectly present via header.php, which includes meta tags, title, stylesheets, and opening structure.  **<nav**> is correctly implemented with Bootstrap and is dynamically populated based on user role using PHP logic (isLoggedIn(), getUserRole()).  **<section>** is used extensively to organize key content areas (hero-section, Featured Jobs, How It Works), enhancing structure and readability.  **<footer>** is properly used and contains organized columns with links and contact info, following good semantic practices.   * + Inclusion of accessibility features (e.g., alt text, headings hierarchy) |
| 1. Css structure | * Describe your approach to CSS styling. * Our approach to CSS prioritizes clarity, maintainability, and scalability. We use intuitive class names (think .job-card, not generic-box) and structure styles logically—separating core foundations (typography, spacing, colors) from reusable components. Thoughtful micro-interactions, like smooth hovers and subtle animations, bring the UI to life without sacrificing performance. Every line of code is crafted to be fully responsive, ensuring a polished experience across all devices. Ultimately, we aim for clean, efficient CSS that’s as functional as it is visually appealing. * Explain the use of classes and IDs for styling. * We mainly use classes when we want the same style to appear on lots of different parts of our website, like making all our buttons blue with rounded corners, which keeps everything looking consistent without copying and pasting the same code over and over. IDs are our special labels for one-of-a-kind elements (like your main header or contact form) that need unique styling or when JavaScript needs to find that exact element on the page. This approach keeps our code organized and makes updating the website's look much easier, since changing one class definition instantly updates all elements using that class across the entire site. * Mention your organization strategy (external stylesheets, comments for clarity). * We organize our CSS by using external stylesheets to keep the HTML clean and improve loading speed since browsers can cache these files for returning users. This method also simplifies maintenance because we can update styles globally without digging through HTML. Inside each stylesheet, we add clear comments to mark different sections—like “Global Variables” or “Button Styles”—and explain any special effects such as animations or hover states. These comments serve as helpful guides for our team, making it easier to understand the styles and get up to speed quickly. * Discuss styling techniques like Flexbox, Grid, and Responsive Design. * We use Flexbox to handle one-dimensional layouts like navbars, while Grid helps us manage complex grids such as dashboards. Media queries allow us to adjust spacing and layout at breakpoints to ensure responsiveness. These tools keep our design adaptable across different devices. * Describe how CSS enhances the website’s aesthetics and usability. * CSS transforms plain HTML into visually appealing websites by adding colors, fonts, and layout that match your brand's personality, while making buttons and menus respond naturally to your clicks and taps. It makes your website work beautifully on phones and computers alike, with smooth animations that feel satisfying without being distracting. These style enhancements turn a basic webpage into an enjoyable experience that feels professional and keeps visitors engaged with your content. |
| ***System Integration:*** | |
| 1. Define Requirements: | (Clearly outline the objectives of the integration, including what data needs to be shared between the systems and how often.) |
| 2. Web Systems: | Select two web systems that support database integration and are compatible with each other. Consider factors like ease of integration, scalability, and security**.**(Admin and Client) |
| 4. Integration Points: | (Determine where the integration will occur within each web system. This could include APIs, database connections, or custom code.) |
| 5. Data Mapping: | (Create a mapping document that defines how data will be mapped between the two databases. Specify which fields will be synchronized and how they will be transformed, if necessary.) |
| ***Database:*** | |
| 1. Project Overview | The job portal database connects job seekers and employers, allowing users to register, search jobs, and apply, while companies post vacancies and manage applications. Admins oversee users, verify employers, and ensure smooth operations. The system simplifies hiring and job hunting through an organized, secure platform. |
| 2. Requirements Specification | *2.1 Functional Requirements*  List core functionalities that the database must support.   * Users can register by providing (name, email, and password). * Users can log in by using their email and password. * Companies can post job listings (title, description, requirements, location). * Companies can delete or edit the job posted. * Companies can approve (for the schedule) and cancel your application. * Admins can view the total number of registered users, companies, jobs, and also applicants. * Admins can manage user/company accounts.   *2.2 Non-Functional Requirements*  Performance, scalability, security, backup, etc.   * **Performance**: The system performs well for basic CRUD operations, with acceptable response times under normal loads. * **Scalability**: The system is designed to be scalable, with the potential to handle increased load and user data as needed. * **Security**: User and company passwords are securely hashed. * **Backup**: Regular backups are feasible, and schema and codebase copies are maintained to ensure recoverability. |
| 3. Conceptual Design | *3.1 Entity-Relationship (ER) Diagram*    *3.2 Entities and Descriptions*   * **User (Job Seeker) -** Stores information about registered job seekers, including personal details, contact information, skills, and resumes**.** * **Admin –** Stores administrator details, including access privileges and system control permissions. * **Company—**This section contains details of employers/companies posting job listings**,** including company information, contact details, and industry. * **Job Listing -** Stores all job postings with details like title, description, requirements, salary range, and location. |
| 4. Logical Design | *4.1 Relational Schema*  Define tables with attributes and data types.   * User   + id INT AUTO\_INCREMENT PRIMARY KEY,   + name VARCHAR(100) NOT NULL,   + email VARCHAR(100) NOT NULL UNIQUE,   + password VARCHAR(255) NOT NULL,   + role ENUM('user', 'company', 'admin') NOT NULL DEFAULT 'user',   + status BOOLEAN DEFAULT TRUE,   + created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP * Companies * id INT AUTO\_INCREMENT PRIMARY KEY, * company\_name VARCHAR(150) NOT NULL, * email VARCHAR(100) NOT NULL UNIQUE, * password VARCHAR(255) NOT NULL, * location VARCHAR(100), * description TEXT, * created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP   *4.2 Primary and Foreign Keys*  List keys and constraints for referential integrity.  **Primary Keys:**   * **users**: id (INT AUTO\_INCREMENT PRIMARY KEY) * **companies**: id (INT AUTO\_INCREMENT PRIMARY KEY) * **job\_listings**: id (INT PRIMARY KEY AUTO\_INCREMENT) * **job\_applications**: id (INT AUTO\_INCREMENT PRIMARY KEY) * **schedules**: id (INT AUTO\_INCREMENT PRIMARY KEY) * **notifications**: id (INT AUTO\_INCREMENT PRIMARY KEY)   **Foreign Keys:**   * **job\_listings**: company\_id references companies(id) * **job\_applications**: user\_id references users(id), job\_id references job\_listings(id) * **schedules**: application\_id references job\_applications(id) * **notifications**: user\_id references users(id)   *4.3 Normalization*  Indicate up to which normal form the design is normalized (1NF, 2NF, 3NF, etc.). |
| 5. Physical Design | *5.1 Database Management System (DBMS)*  MySQL is used in our system since it integrates seamlessly with PHP, allowing efficient development of our web applications. We chose it since it reliably manages complex, high-volume databases while maintaining optimal speed and performance.  *5.2 Indexing Strategy*  Tables/columns with indexes and justification.  CREATE INDEX idx\_job\_listings\_company ON job\_listings(company\_id);  CREATE INDEX idx\_job\_listings\_status ON job\_listings(status);  CREATE INDEX idx\_job\_applications\_user ON job\_applications(user\_id);  CREATE INDEX idx\_job\_applications\_job ON job\_applications(job\_id);  CREATE INDEX idx\_notifications\_user ON notifications(user\_id);  CREATE INDEX idx\_notifications\_read ON notifications(is\_read);  *5.3 Storage and Performance Considerations*  Storage requirements and performance enhancements.   * The system should have at least 256 GB of storage, with the ability to scale up if needed. * The system should efficiently handle basic Create, Read, Update, and Delete (CRUD) operations, delivering fast and reliable response times under typical workloads. |
| 6. Security and Access Control | *6.1 User Roles and Permissions*  Role Access Level  Admin Full access (login, register, view the user)  User Limited access (view jobs, apply for jobs)  Company Limited access (posting of jobs, viewing the applicant)  *6.2 Data Security Measures*  Encryption, authentication, backups, etc.   * Encryption is used to protect sensitive data such as passwords, which are stored in a hashed format to prevent easy access if breached. * Authentication mechanisms ensure that only authorized users can access the system using login credentials. This is possible because we have an admin side that manages user access. * Backups are implemented regularly to maintain copies of important files, database schemas, and other data. |
| 7. QUERIES | Provide examples of key SQL queries.  -- Create database  CREATE DATABASE careernest\_db;  USE careernest\_db;  -- Users table  CREATE TABLE users (      id INT AUTO\_INCREMENT PRIMARY KEY,      name VARCHAR(100) NOT NULL,      email VARCHAR(100) NOT NULL UNIQUE,      password VARCHAR(255) NOT NULL,      role ENUM('user', 'company', 'admin') NOT NULL DEFAULT 'user',      status BOOLEAN DEFAULT TRUE,      created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  );  -- Insert admin account  INSERT INTO users (name, email, password, role, status, created\_at) VALUES  ('Admin', 'admin@careernest.com', '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', 'admin', 1, NOW()); |
| 8. Data Dictionary | Table Column Data Type Description  Users id INT Unique ID for user  Users name VARCHAR(100) User’s name  Users email VARCHAR(100) User's email address  Users password VARCHAR (255) User’s password hashed  Users role ENUM User’s role (admin, company,job seeker)  Users status BOOLEAN Whether the user is active (true/false)  Users created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  Date and time the user created |