Software Requirements Specification

for

Outsourcing System

Version 1.0 approved

Prepared by

Diaz Reyes Luis Ignacio
Gonzalez Leyva Cristopher Anahel
Estrada Zarate Diego Enrique
Soto Garcia Oscar Gael

Universidad Tecnológica de Tijuana

September 25, 2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This document explains the software requirements for a platform that manages outsourcing. The system will help companies with the whole process of hiring, from posting job openings to hiring people. The platform will cover key features like managing clients, posting job offers, managing applicants (prospects), and signing contracts. This document focuses on the main parts of the system and does not include extra features like task management or integration with other systems.

1.2 Document Conventions

This SRS uses simple text formatting to ensure easy readability. The following conventions are used:

- **Bold text** is used to highlight key terms or sections.
- *Italics* are used for examples or notes.
- All requirements are listed in normal text and are numbered for easy reference.
- Each requirement has its own priority, meaning the priority level is specified individually for every requirement.

No complex typographical conventions or special symbols are used in this document to keep it simple and clear.

1.3 Intended Audience and Reading Suggestions

This document is intended for several types of readers:

- **Developers:** To understand the functional and technical requirements needed to build the system.
- **Project Managers:** To track project scope and ensure all requirements are met.
- **Testers:** To use the requirements for creating test cases and validating the system.
- **Documentation Writers:** To prepare user manuals based on the system's features.

The document is organized into sections covering an overview of the system, followed by detailed functional and non-functional requirements. Readers should start with the overview sections, like the Purpose and Product Scope, before moving into the more specific requirements.

1.4 Product Scope

The software being developed is an outsourcing platform. Its main purpose is to help companies manage the hiring process by allowing them to post job vacancies, review applicants (prospects), and finalize contracts. The platform's goal is to automatize recruitment. This software aligns with business goals by automating key HR processes, saving time, and improving the quality of hires.

1.5 References

- *IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications*

2. Overall Description

2.1 Product Perspective

The outsourcing platform described in this SRS is a new product designed to automate the recruitment and hiring process for companies. It is not a replacement or part of a product family but an independent system planned to do job vacancies and contracts. The platform will serve as a tool for Human Resources departments, enabling efficient and automated management of recruitment.

2.2 Product Functions

The main functions of the platform include:

- **Client Management:** Track and manage company profiles and their recruitment needs.
- **Vacancy Posting:** Allow companies to post job vacancies and update them as needed.
- **Prospect Management:** Enable prospects (applicants) to apply for job vacancies and submit required documentation.
- **Contract Management:** Automate contract generation and finalization once a prospect is hired.
- **User Authentication:** Secure login for different user types (HR, Admin, Prospects).

2.3 User Classes and Characteristics

The platform will have 3 user classes, each with different needs and access levels:

- **HR Managers:**
    - **Frequency of Use:** Daily.
    - **Functions Used:** Vacancy posting, prospect management, certification tracking, contract generation, reporting.
    - **Technical Expertise:** Intermediate; basic knowledge of web platforms.
    - **Security Level:** High; access to sensitive applicant and company data.
    - **Importance:** Most important user class, as they drive the system's primary purpose.

- **Prospects (Applicants):**
  - **Frequency of Use:** Occasional, as they apply for jobs or check updates.
  - **Functions Used:** Profile creation, job application.
  - **Technical Expertise:** Basic; general familiarity with web forms.
  - **Security Level:** Medium; access to personal data and job applications.
  - **Importance:** Important to satisfy for smooth application processes.

- **System Administrators:**
  - **Frequency of Use:** Regular, for maintaining the system.
  - **Functions Used:** User management, system configurations, monitoring system performance.
  - **Technical Expertise:** Advanced; IT and database management knowledge.
  - **Security Level:** Very high; full access to system functions.
  - **Importance:** Essential for system maintenance and troubleshooting.

2.4 Operating Environment

The platform will run in a web-based environment. It will be compatible with modern hardware and software systems, specifically:

- **Hardware Platform:** Standard web server configurations, such as Intel-based servers with 8GB+ RAM and sufficient storage for user data.
- **Operating System:** Linux (Ubuntu) or Windows 8+
- **Database:** MySQL for storing data related to users, vacancies and prospects.
- **Web Browsers:** Google Chrome, Mozilla Firefox, Microsoft Edge (latest versions).

2.5 Design and Implementation Constraints

These constraints will impact the development of the outsourcing platform:
Minimum Requirements:

- **Operating System:** Windows Server 2016 or Linux (Ubuntu 18.04+)
- **Processor:** Dual-core Intel or AMD, 2.5 GHz or faster
- **Memory:** 4 GB RAM
- **Storage:** 10 GB available disk space
- **Browser Support:** Chrome (latest), Firefox (latest), Safari 13+

Recommended Requirements:

- **Operating System:** Windows Server 2019 or Linux (Ubuntu 20.04+)
- **Processor:** Quad-core Intel or AMD, 3.0 GHz or faster
- **Memory:** 8 GB RAM
- **Storage:** 20 GB available disk space
- **Browser Support:** Chrome (latest), Firefox (latest), Safari (latest), Microsoft Edge

(latest)

2.6 Assumptions and Dependencies

The successful development of the outsourcing platform relies on several critical factors that, if changed or unmet, could lead to system failures:

- **Operating Environment:** The system is expected to be deployed on Linux or Windows with the specified minimum configuration. Changes in the server environment, whether due to hardware limitations or software incompatibilities, could compromise performance or prevent the system from functioning.
- **Browser Compatibility:** The platform assumes that users will access it through supported modern browsers. Changes in browser standards or updates could lead to display or functionality issues.
- **Database Support:** The project depends on MySQL databases for data storage. If there are changes in database availability or version compatibility, this could necessitate adjustments to the application.
- **User Training:** It is assumed that users will have basic training on how to navigate and use the platform effectively. Insufficient user training may result in improper use, leading to errors or reduced efficiency.
- **Database Connection Errors:** Issues with access credentials (incorrect username or password) and problems with the database server being unavailable or down.

3. External Interface Requirements

3.1 User Interfaces

The outsourcing platform will have a user-friendly interface designed to help HR managers and job seekers interact with the system efficiently. Key aspects of the user interface include:

- **Navigation Menu:** A sidebar or top navigation bar with quick access to main modules (e.g., Vacancies, Prospects, etc.).
- **Standardized Buttons:** Frequently used buttons such as "Submit," "Cancel," "Save," and "Help" will be present on all screens.
- **Screen Layout:** Each screen will follow a consistent layout, with form fields on the left and action buttons on the right. A fixed header will provide easy access to the user’s profile and notifications.
- **Error Messages:** Clear, descriptive error messages will be displayed in red text below the relevant input field if validation fails.

3.2 Hardware Interfaces

The outsourcing platform will interact with various hardware components. Key characteristics include that the system will support desktop computers, laptops, and tablets. Mobile devices will be supported via responsive web design.

Also, the software will send and receive data such as user inputs and system responses through user interface elements. The interaction will include control commands, such as saving data and retrieving records from the database.

3.3 Software Interfaces

The outsourcing platform will connect with several software components:

- **Database:** MySQL version 8.0 for data storage and retrieval. Data items include user profiles, job vacancies, and application submissions.
- **Operating System:** The software will run on Linux (Ubuntu 20.04) or Windows Server (2019) environments.
- **Libraries and Tools:** The application will use PHP 8.0 and HTML5 for web development.
- **Data Sharing:** The system will share user data and application status between the database and the web interface. This will involve querying and updating records as users interact with the platform.

4. System Features

This section organizes the functional requirements for the outsourcing platform by its major features.

4.1 User Registration and Authentication

4.1.1 Description and Priority:

This feature allows users to create accounts and securely log in to the system. It is of High priority due to its critical role in user management and security.

- Benefit: 9
- Penalty: 7
- Cost: 5
- Risk: 3

4.1.2 Stimulus/Response Sequences:

1. User Action: User clicks on "Register" and fills in the registration form.
   System Response: The system validates the input and creates a new user account.
2. User Action: User submits the login form with credentials.
   System Response: The system verifies credentials and grants access if valid; otherwise, it displays an error message.

4.1.3 Functional Requirements:

- REQ-1: The system shall allow users to register by providing a password, and email address.
- REQ-2: The system shall validate that the email address is unique.
- REQ-3: The system shall send a confirmation upon successful registration.
- REQ-4: The system shall allow users to log in with their username and password.

4.2 Job Vacancy Management

4.2.1 Description and Priority:

This feature enables authorized users to create, edit, and delete job vacancies. It is of High priority as it directly impacts recruitment effectiveness.

- Benefit: 8
- Penalty: 5
- Cost: 4
- Risk: 2

4.2.2 Stimulus/Response Sequences:

1. User Action: An authorized user selects "Create Job Vacancy" and fills in the details.
   System Response: The system saves the vacancy and displays a confirmation message.
2. User Action: User clicks "Edit" on an existing vacancy.
   System Response: The system loads the current vacancy details for editing.

4.2.3 Functional Requirements:

- REQ-1: The system shall allow authorized users to create a job vacancy by filling in required fields (title, description, requirements, and salary).
- REQ-2: The system shall allow users to edit existing vacancies.
- REQ-3: The system shall allow users to delete vacancies and require confirmation before deletion.
- REQ-4: The system shall display a list of all current job vacancies to users.

4.3 Application Submission

4.3.1 Description and Priority:

This feature allows candidates to apply for job vacancies. It is of High priority as it directly supports the recruitment process.

- Benefit: 9
- Penalty: 6
- Cost: 3
- Risk: 4

4.3.2 Stimulus/Response Sequences:

1. User Action: A candidate selects a job vacancy and clicks "Apply."

   System Response: The system prompts the user to upload a resume and fill out application details.

2. User Action: Candidate submits the application.

   System Response: The system confirms receipt of the application and sends a notification email.

4.3.3 Functional Requirements:

- REQ-1: The system allows candidates to view job details before applying.
- REQ-2: The system enables candidates to submit their applications online.
- REQ-3: The system sends confirmation to candidates upon successful application submission.
- REQ-4: The system shall track the status of each application and allow candidates to view it.

4.4 Candidate Profile Management

4.4.1 Description and Priority:

This feature allows candidates to create and manage their profiles, including their qualifications, experience, and skills. It is of High priority for enhancing the user experience and streamlining applications.

- Benefit: 8
- Penalty: 4
- Cost: 3
- Risk: 3

4.4.2 Stimulus/Response Sequences:

1. User Action: Candidate navigates to their profile page and selects "Edit Profile."

   System Response: The system loads the profile form for editing.

2. User Action: Candidate updates their information and saves changes.

   System Response: The system confirms the updates and displays the updated profile.

4.4.3 Functional Requirements:

- REQ-1: The system shall allow candidates to create a profile with personal information, education, work experience, and skills.
- REQ-2: The system enables candidates to edit their profile information at any time.

4.5 Employer Dashboard

4.5.1 Description and Priority:

This feature provides employers with a dashboard to manage job vacancies, view applicants, and track hiring progress. It is of Medium priority as it enhances employer interaction with the system.

- Benefit: 7
- Penalty: 4
- Cost: 5
- Risk: 3

4.5.2 Stimulus/Response Sequences:

1. User Action: Employer logs into their account and accesses the dashboard.
   System Response: The system displays the dashboard with key metrics and recent activities.
2. User Action: Employer selects "View Applications" for a specific vacancy.
   System Response: The system shows a list of applicants with their statuses.

4.5.3 Functional Requirements:

- REQ-1: The system provides employers with an overview of their active vacancies, applications received, and hiring status.
- REQ-2: The system provides analytics on application trends and employer performance.

4.6 Contract Management

4.8.1 Description and Priority:

This feature allows the management of contracts between the outsourcing company and prospects. Contracts can be either direct or indirect, specifying terms such as start date, end date, and payment terms.

- Benefit: 8
- Penalty: 4
- Cost: 3
- Risk: 3

4.8.2 Stimulus/Response Sequences:

1. User Action: A company requests to create a new contract.
   System Response: The system generates a new contract record and saves the necessary details.
2. User Action: The user views the details of an existing contract.
   System Response: The system retrieves and displays the contract details.

4.8.3 Functional Requirements:

- REQ-1: The system allows users to create, edit, and delete contracts.
- REQ-2: The system stores information on whether the contract is direct or indirect.
- REQ-3: The system allows users to view contract details including start date, end date, payment terms, and status (active or terminated).

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The performance requirements for the outsourcing platform are crucial to ensure a smooth user experience and system reliability. These requirements include:

- **Scalability:** The system should be able to handle an increase of 50% in the number of users and data without needing major changes to the existing setup.
- **Compatibility**: The system should be compatible with the latest versions of popular web browsers (Chrome, Firefox, Edge)
- **Usability**: The user interface should be intuitive and accessible, with a design that allows users to complete key tasks (e.g., submitting applications or posting job vacancies) in a few steps.

5.2 Safety Requirements

Safety requirements are essential to protect both users and the system from potential risks associated with its use. The safety requirements include:

- **User Authentication:** User authentication is the process of verifying the identity of a user trying to access the system.
- **Error Handling:** The system must gracefully handle errors and provide user-friendly messages to prevent user confusion and frustration.

5.3 Security Requirements

This section outlines the security requirements necessary to protect user data and ensure the integrity of the system.

- **Access Control:** Implement roles to restrict user permissions based on their roles within the system, ensuring that users can only access data necessary for their functions.
- **User Identity Authentication:** Implement user authentication using secure login methods.

5.4 Software Quality Attributes

This section details the quality attributes that define the overall performance and user experience of the product.

- Usability: User interfaces must be intuitive and easy to navigate, aiming for a user satisfaction in usability testing.
- Maintainability: The system should be designed for ease of maintenance, allowing updates and bug fixes to be implemented within 24 hours of identification.
- Performance: The system should support at least 100 concurrent users with response times not exceeding 2 seconds for most actions.
- Security: The product must pass regular security assessments with no critical vulnerabilities identified.

6. Other Requirements

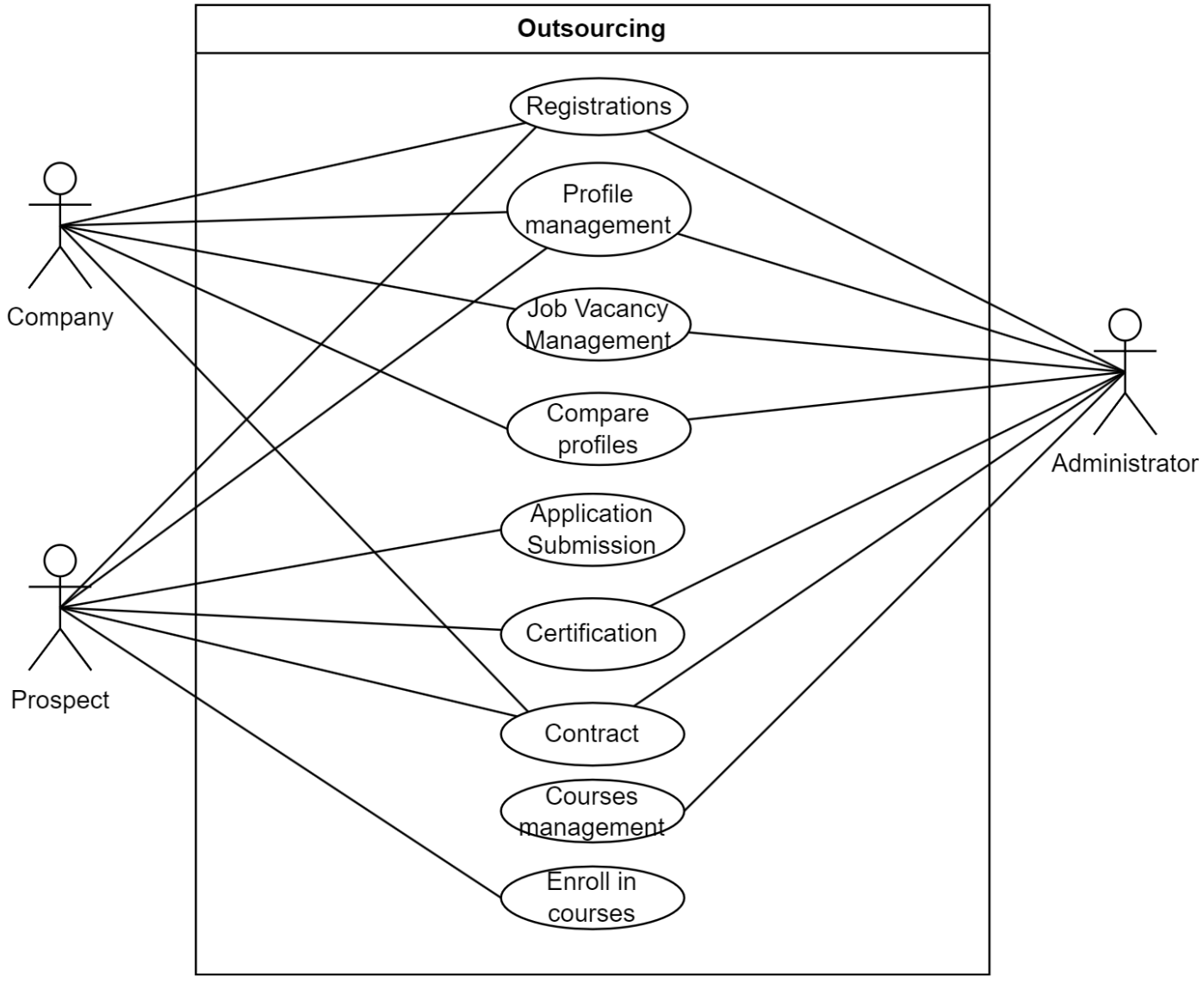
Documentation Requirements: Comprehensive documentation must be provided for both users and developers, including user manuals, and system architecture diagrams.

Appendix A: Behavioral UML diagrams

Behavioral diagrams in software design focus on the dynamic aspects of the system—how the system behaves, its interactions, and the flow of control or data. They describe the functionality and interactions of system components.

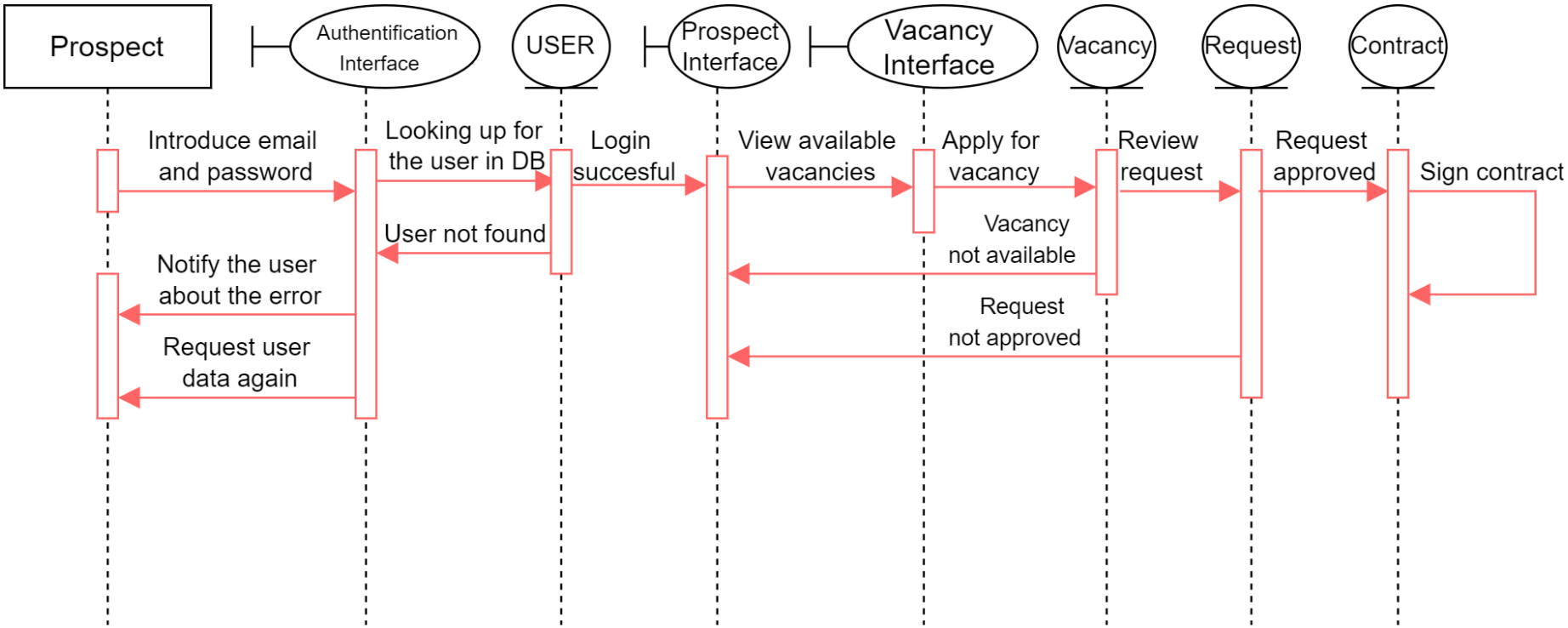
Use Case Diagram

Shows interactions between users (actors) and the system.



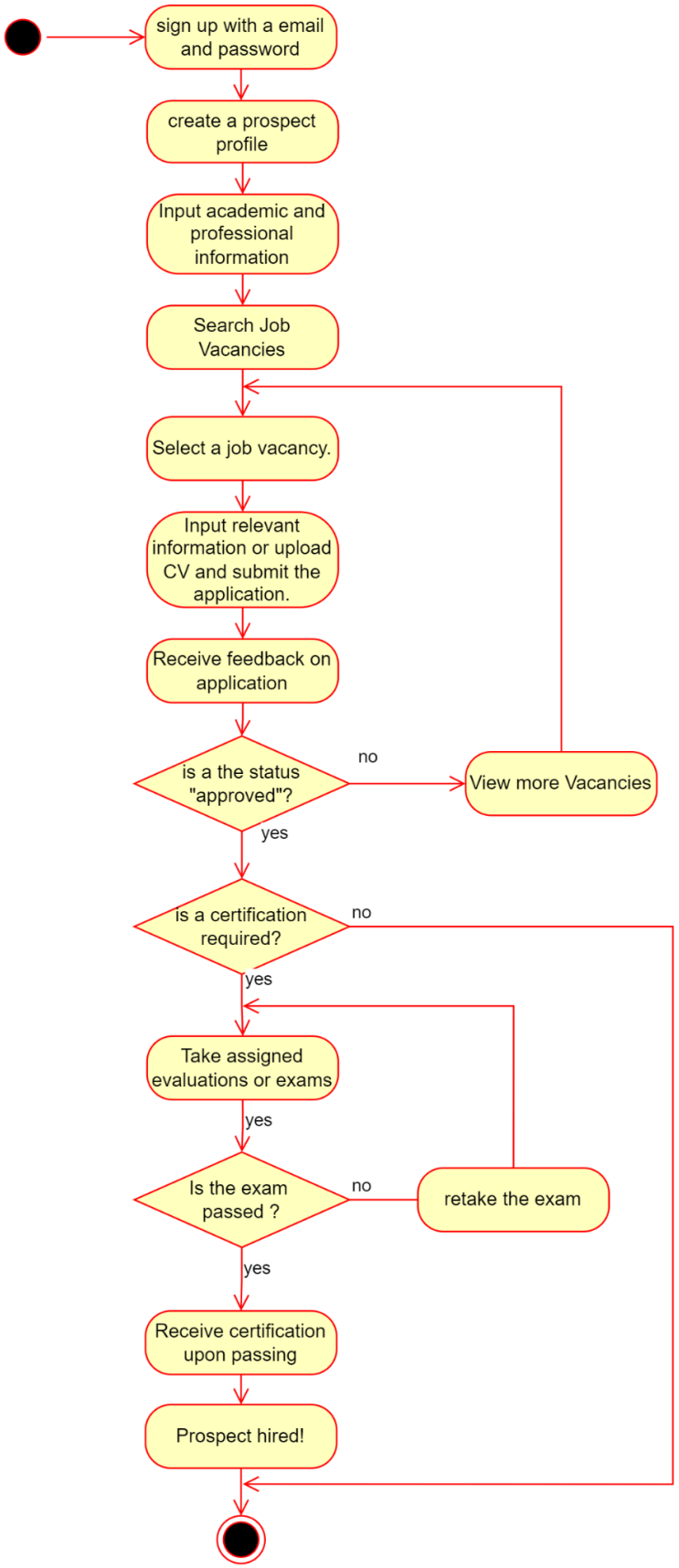
Sequence Diagram

Illustrates the sequence of messages exchanged between objects or components over time.

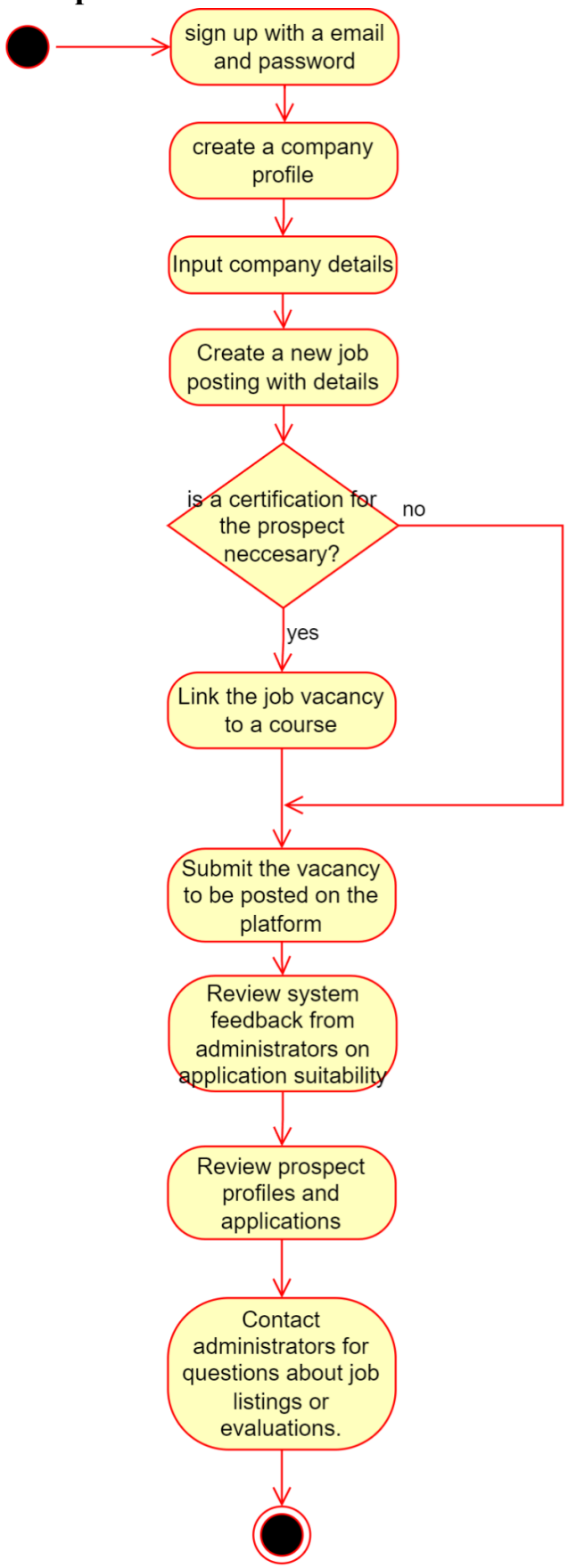


Activity Diagram for Prospects

Represents workflows or processes within the system, highlighting the sequence of actions.

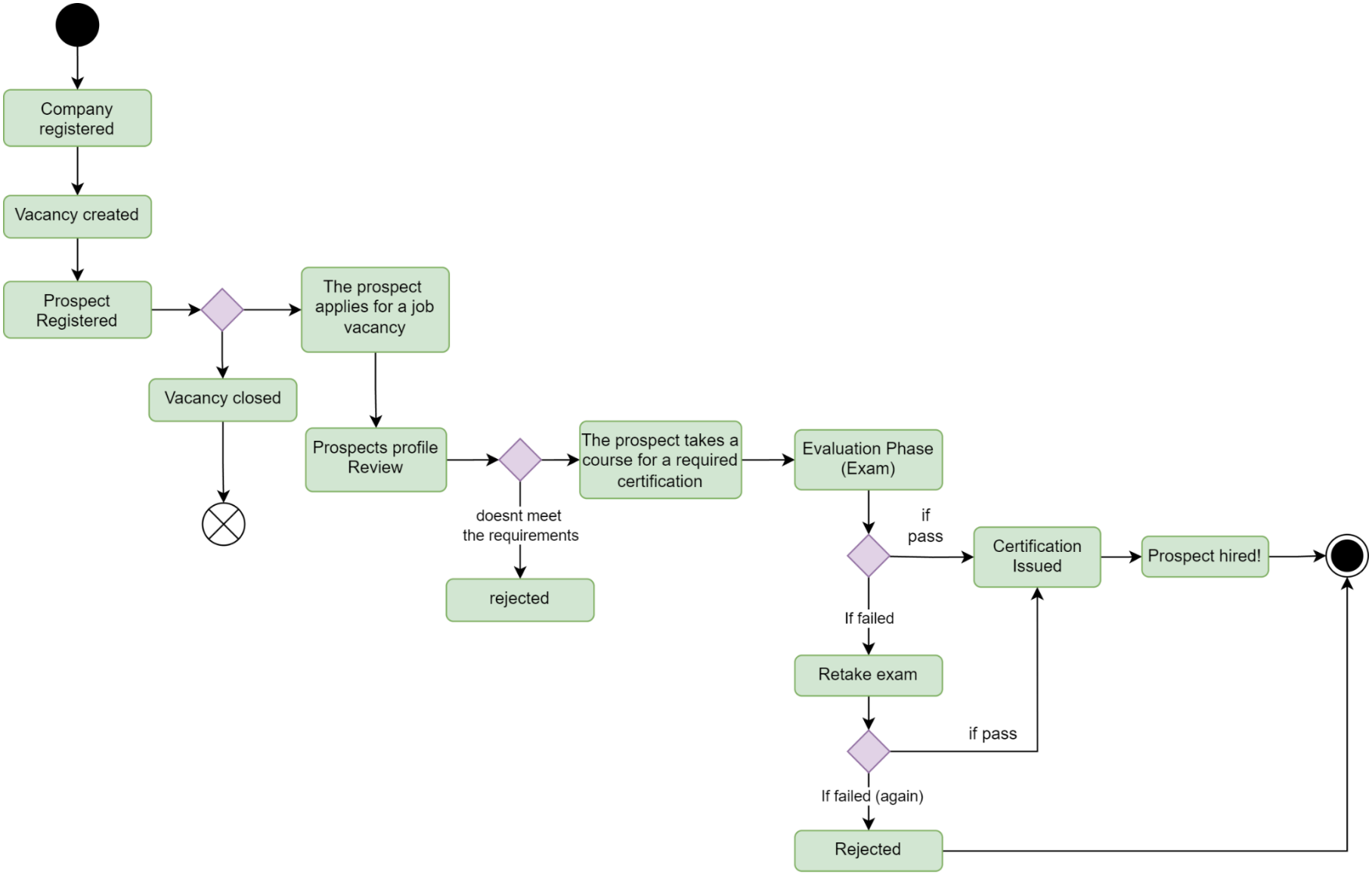


Activity Diagram for Companies



State Machine Diagram

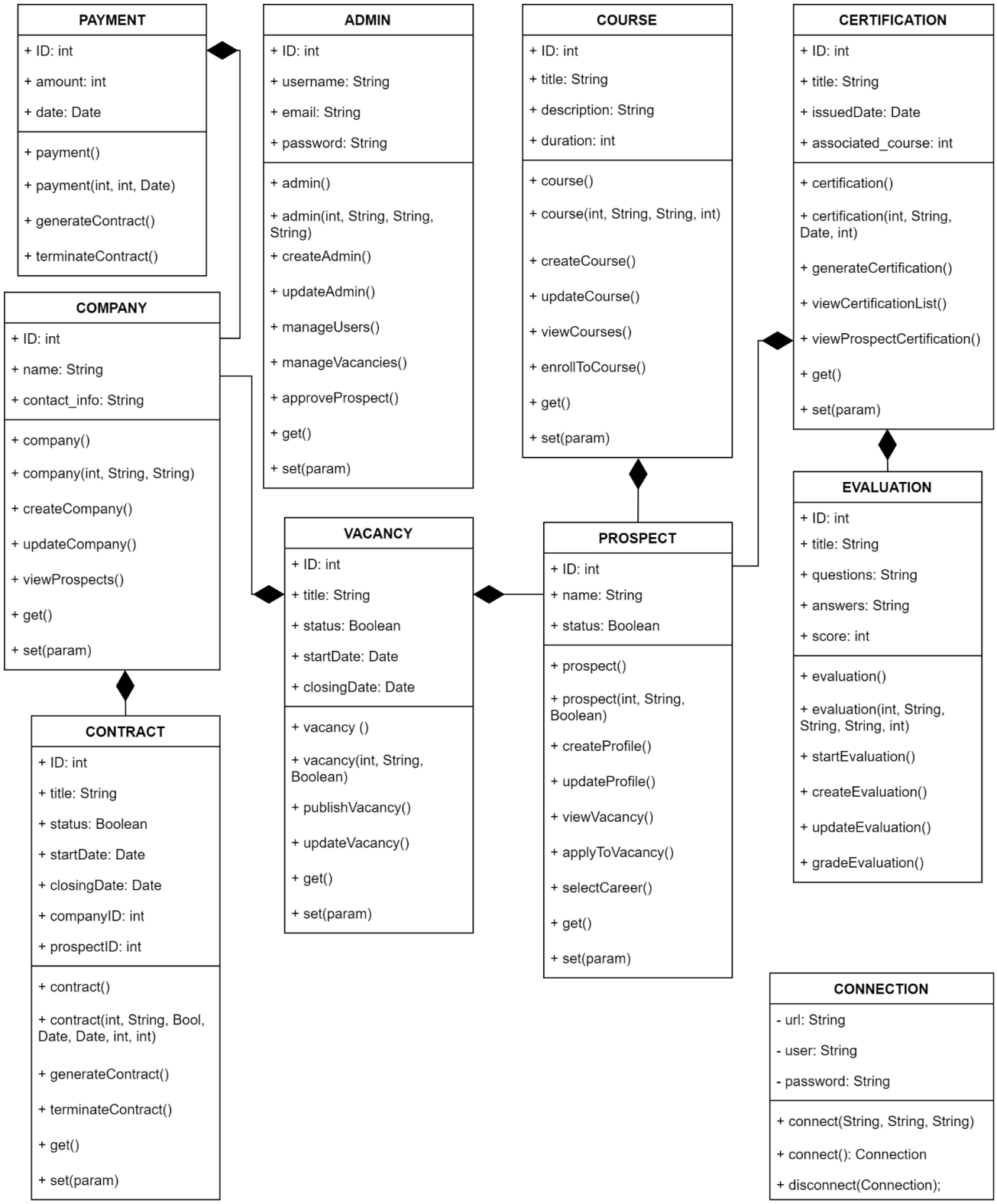
Depicts the states an object can be in and the transitions between those states based on events.



Appendix B: Structural UML diagrams

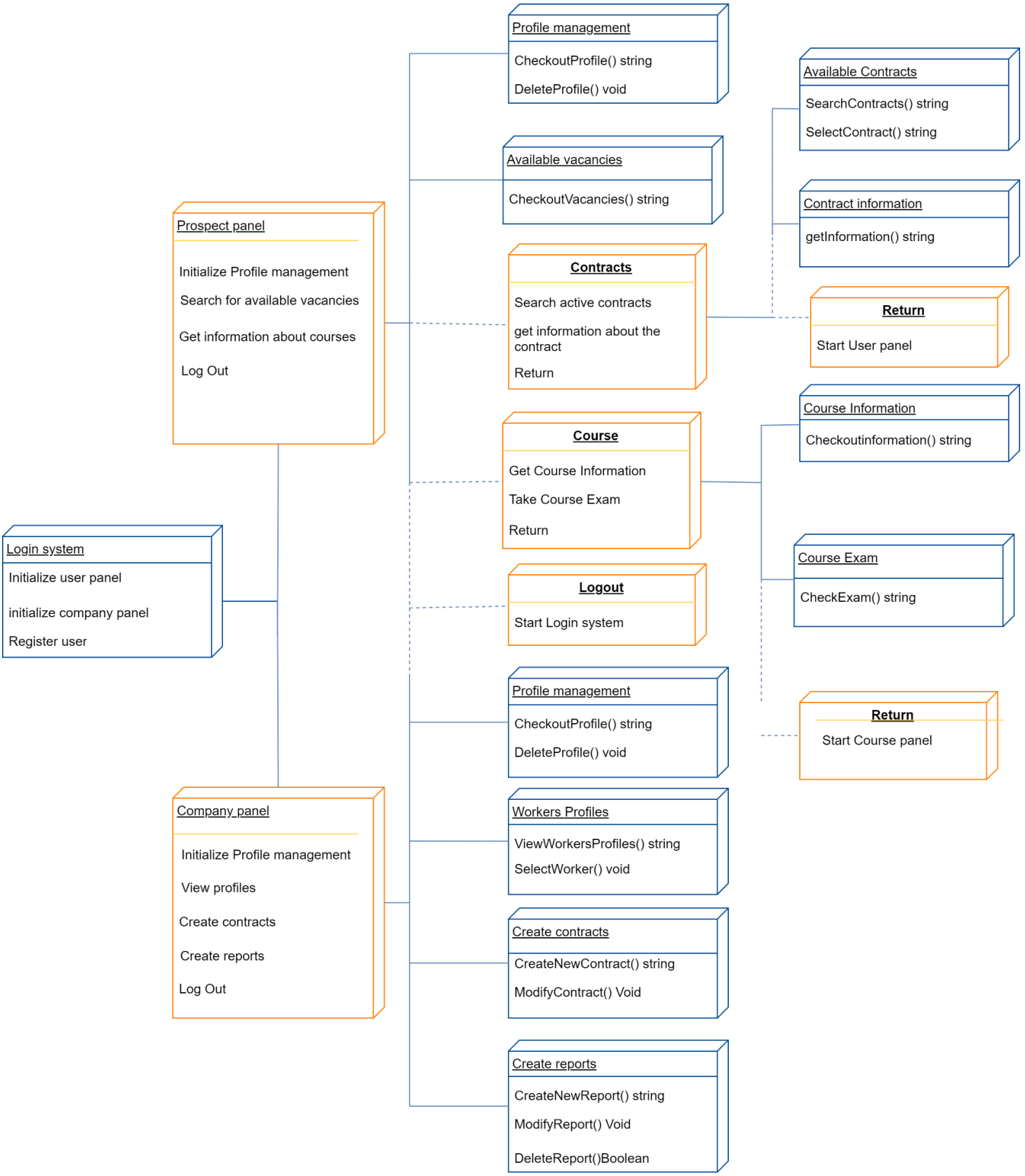
Class Diagram

Shows the classes, attributes, methods, and relationships in an object-oriented system.



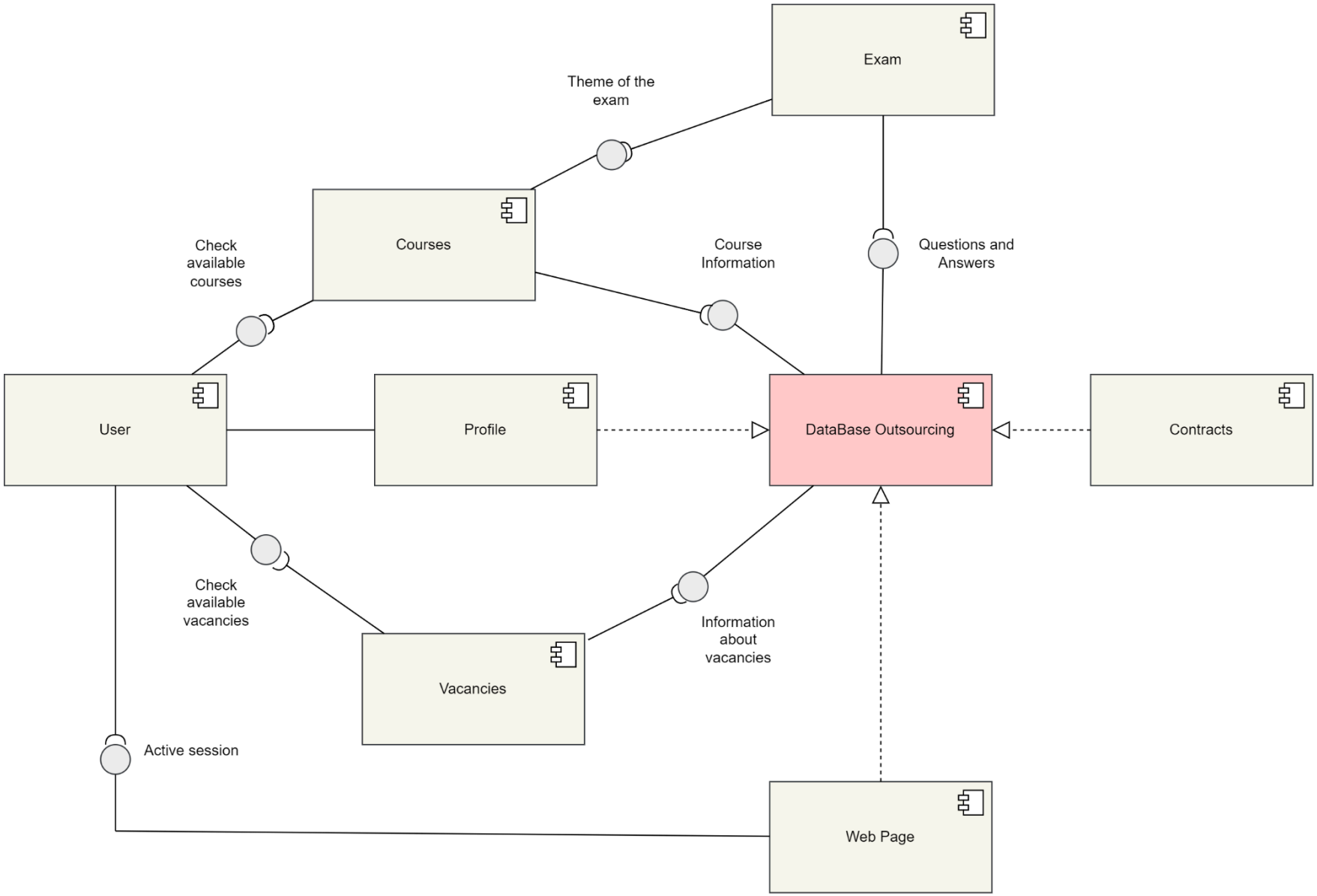
Deployment Diagram

Depicts the architecture of components and their relationships.



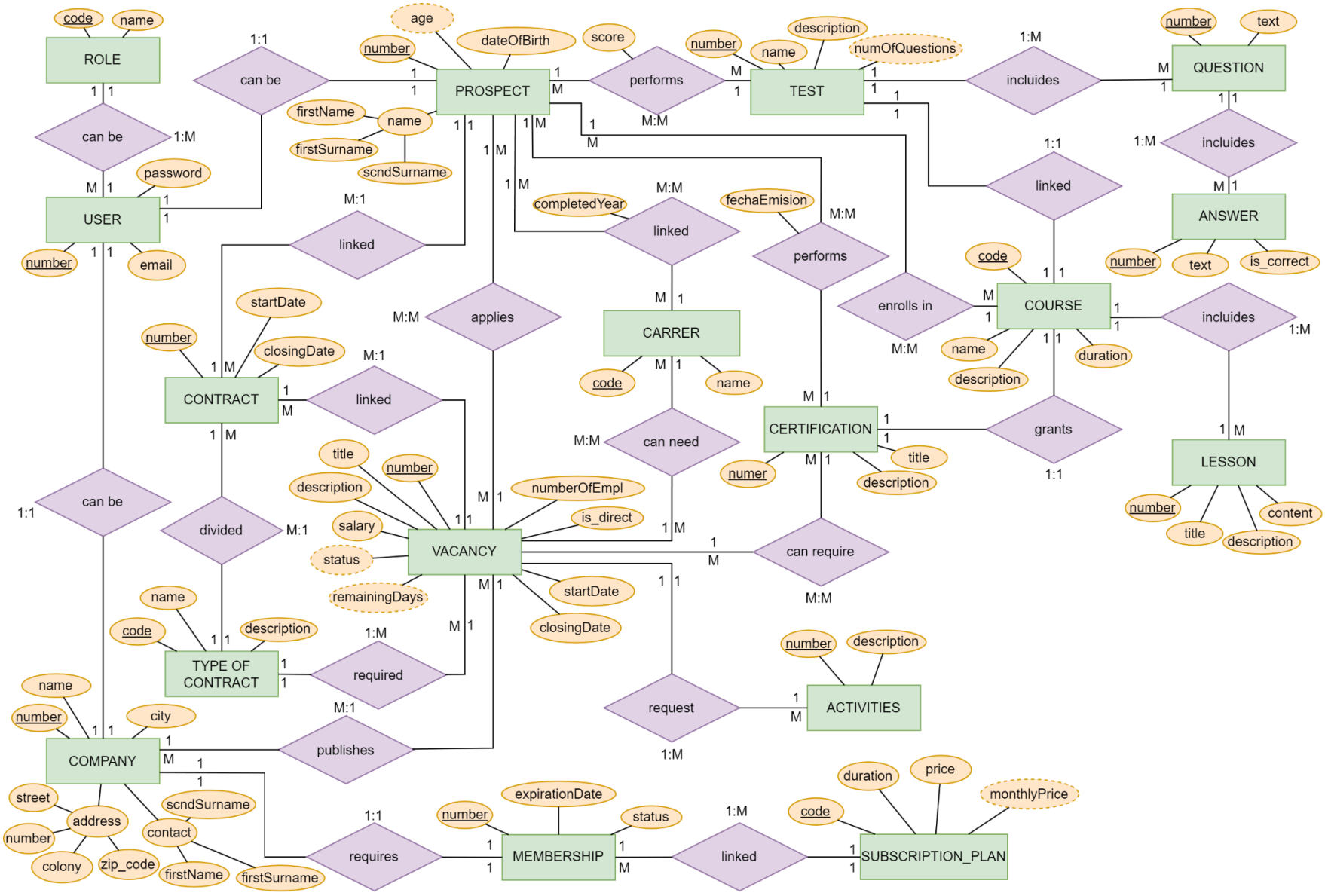
Component Diagram

Represents the different components of a system are organized and interact with each other.

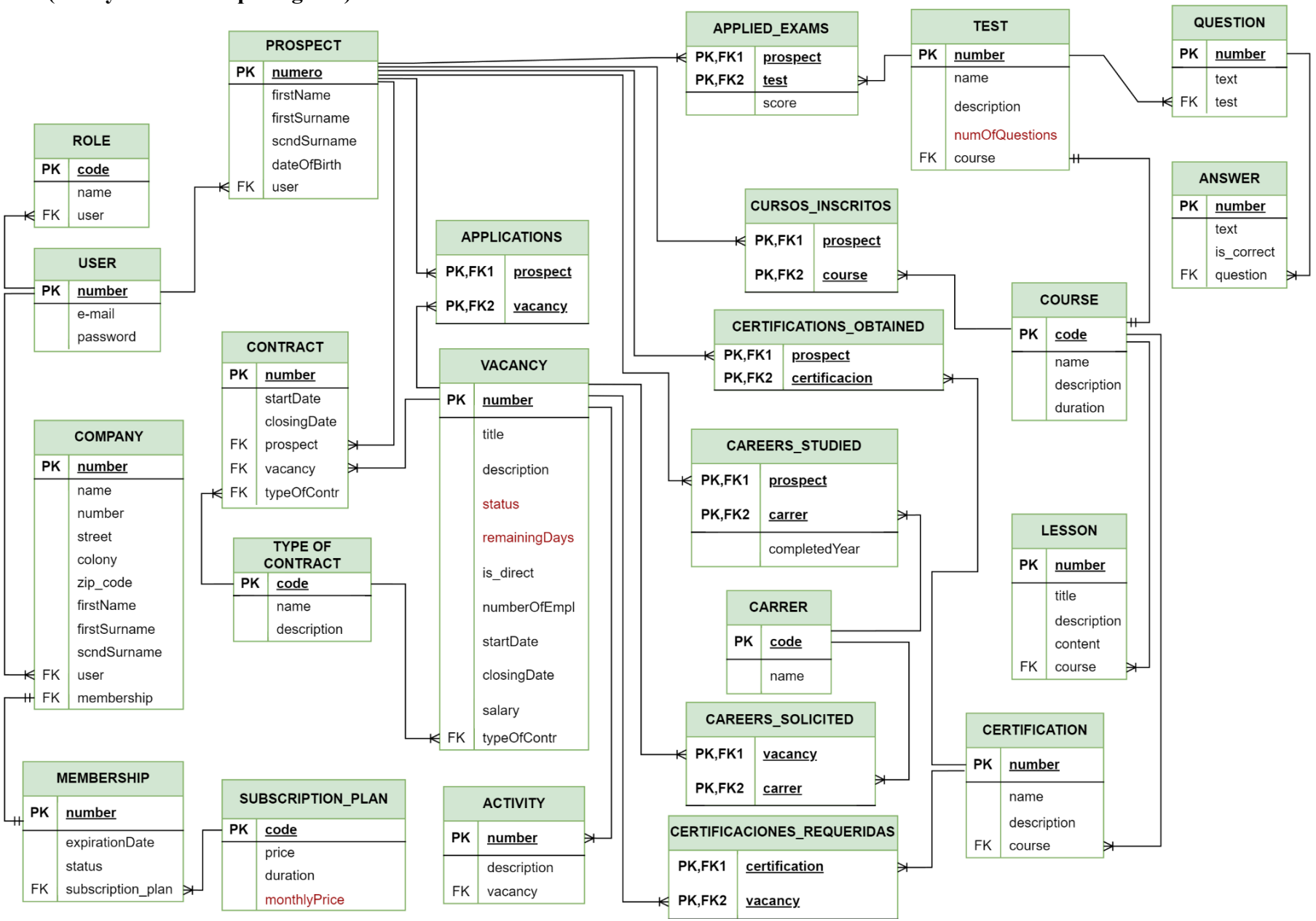


Appendix C: Database Structure Diagrams

DR (Data Flow Diagram - DFD or Data Model Diagram):



ERD (Entity-Relationship Diagram)



Appendix D: Stages for Creating the Web Application

Creating a website or web application involves a series of structured steps to ensure successful development, design, and deployment. This process allows the project team to organize tasks, plan content, and maintain quality standards while meeting client needs. The following seven stages outline the essential phases for bringing a website or web application to life, from initial planning to final launch.

1) Preprocess

This phase determines who will be responsible for each part of the project. The client is considered part of the team as they share their vision, and the team brings it to life.

1.1) Role assignment.

1. **Content Creator**: Cristopher
2. **Architect**: Oscar y Diego
3. **Tester**: Oscar y Diego
4. **Graphic Designer**:Ignacio
5. **UI/UX Designer**:Ignacio
6. **Frontend Developer**: Ignacio y Cristopher
7. **Backend Developer**: Diego, Oscar y Cristopher
8. **Testing Supervisor**: Ignacio
9. **Product Owner:** UTT Career Teachers

2) Content Modeling for the Outsourcing System

The content modeling phase identifies and organizes the key types of information that will be featured in the outsourcing system. It is crucial to ensure that the content supports user tasks and business goals effectively.

2.1) Content Modeling

For this project, the system will primarily cater to three types of users: **Companies**, **Prospects**, and **System Administrators**. Below is a breakdown of the content structure for each user type.

2.1.1) Companies

Companies represent the clients who will use the system to find potential employees. Key content for companies will include:

- **Company Profiles**: Overview of the company, contact details, industry type, and location.
- **Job Vacancies**: Listings of available positions, including job descriptions, required skills, and job locations.
- **Hiring Requests**: Requests for employees, specifying the number of vacancies, required roles, and other criteria.

2.1.2. Prospects

Prospects are individuals looking for job opportunities through the system. The system will need to provide the following content:

- **Prospect Profiles**: Personal details, resumes/CVs, skills, educational background, and work experience.
- **Application History**: List of positions the prospect has applied for, including the status of each application (e.g., pending, reviewed, hired).

2.1.3) System Administrators

System administrators are responsible for overseeing the system's operations. They will need access to the following content:

- **User Management**: Tools to manage companies and prospects, including creating, editing, or deleting profiles.
- **Vacancy Management**: Review and approve job postings and monitor hiring activity.
- **System Reports**: Access to system usage statistics, hiring trends, and overall performance.

2.2) Content Hierarchy

The hierarchy will define the importance and relationships between pages and sections of the system. The structure will be:

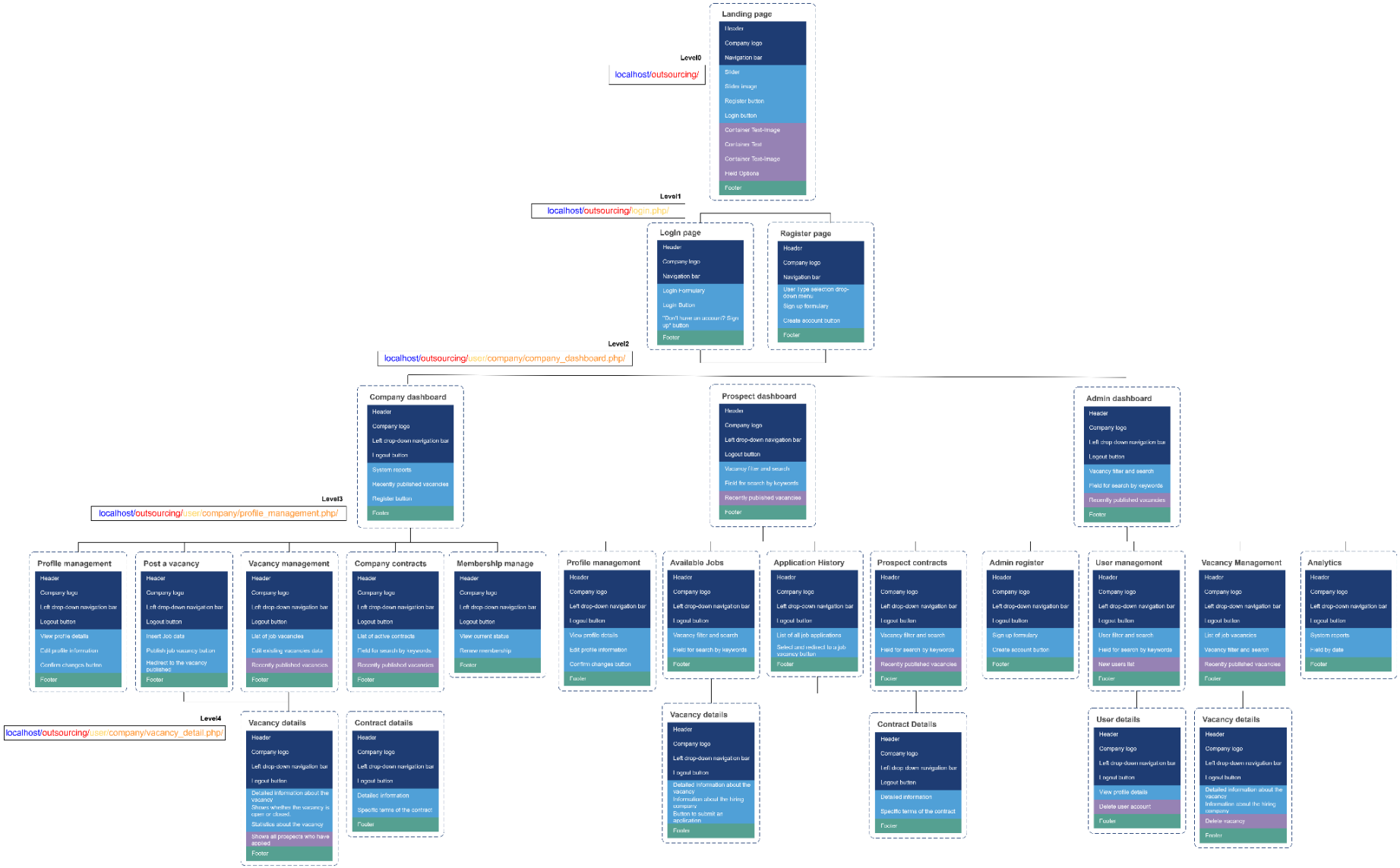
- **Home**
  - ➢ Quick access to main features based on user type.
- **Dashboard**
  - ➢ Separate dashboards for Companies, Prospects, and Administrators, each displaying relevant content.
- **Job Vacancies**
  - ➢ Listing of open positions for prospects, filterable by company, location, and job type.
- **Application Status**
  - ➢ For prospects to view their application history.
- **User Profiles**
  - ➢ Manage profiles for companies and prospects, showing their details and information.

2.3) Information Architecture

The navigation between content areas will follow a logical flow to ensure a smooth user experience. Here's a suggested navigation structure:

- **Home Page**: Provides links to log in for companies, prospects, and system administrators.
- **Company Dashboard**:
  - View Job Listings
  - Post a Job
  - View Applications
  - Manage Company Profile
- **Prospect Dashboard**:
  - View Available Jobs
  - Apply for a Job
  - View Application Status
  - Manage Profile
- **Admin Dashboard**:
  - Manage Users (Companies and Prospects)
  - Review Job Listings
  - System Reports

2.4) Information Architecture (Diagram)

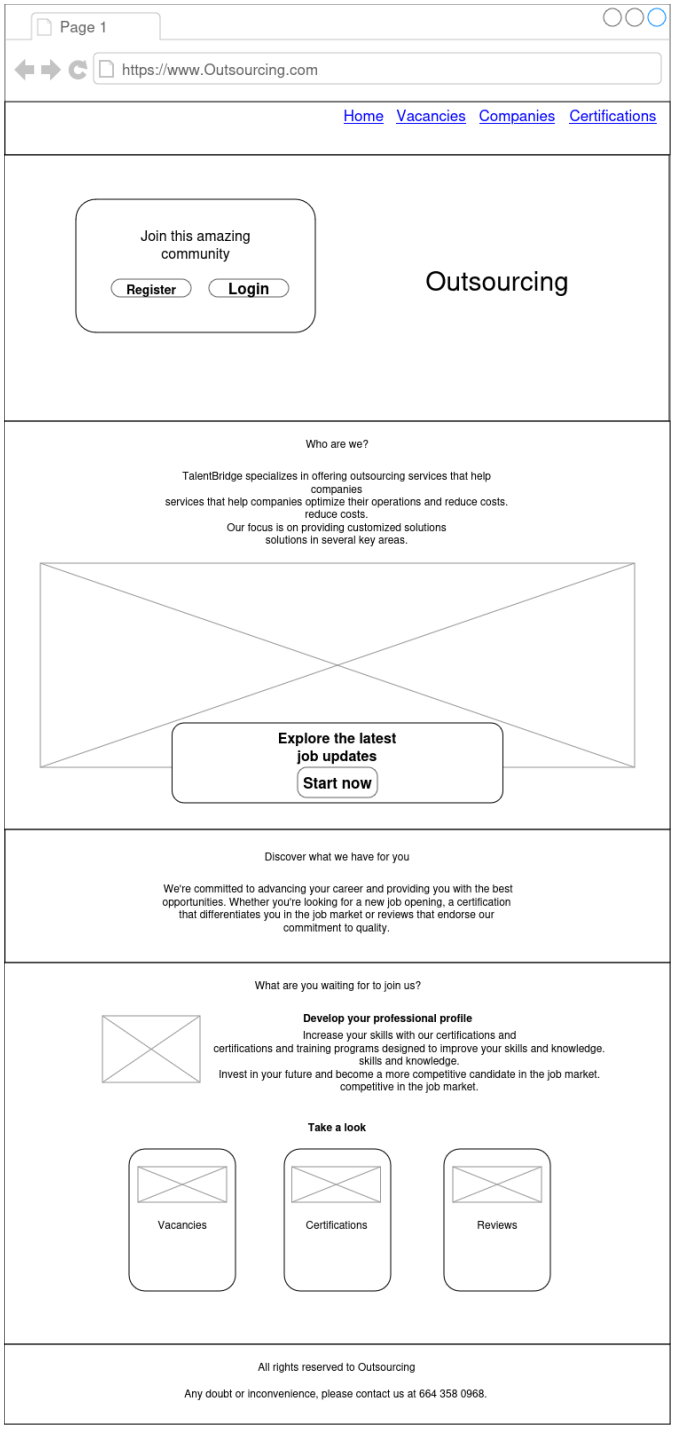


3) Preconstruction

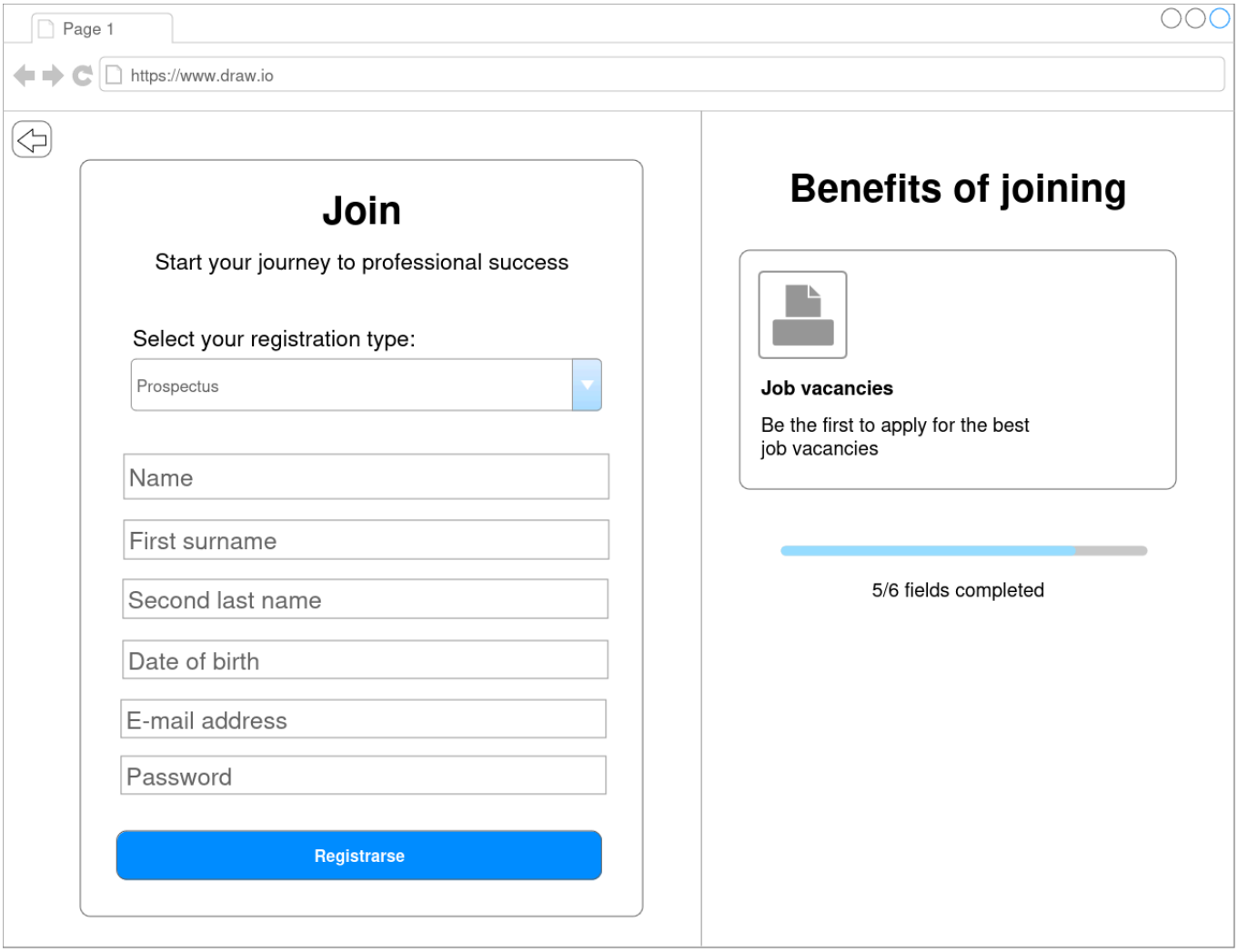
Create a series of wireframes that define the flow from section to section. With these wireframes, identify the necessary resources (images, videos, logos, sections, etc.).

3.1) Overall Wireframes

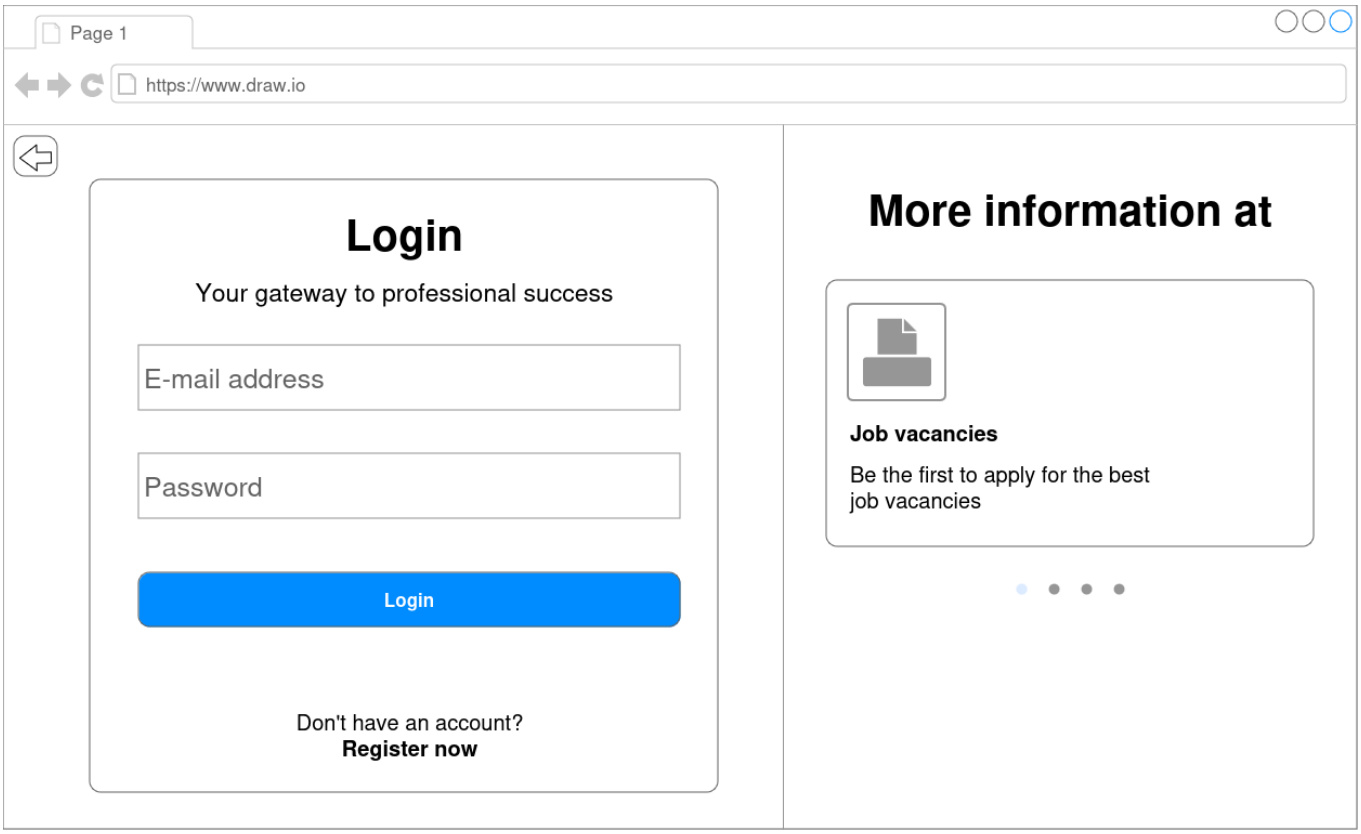
3.1.1) Index homepage



3.1.2) Register

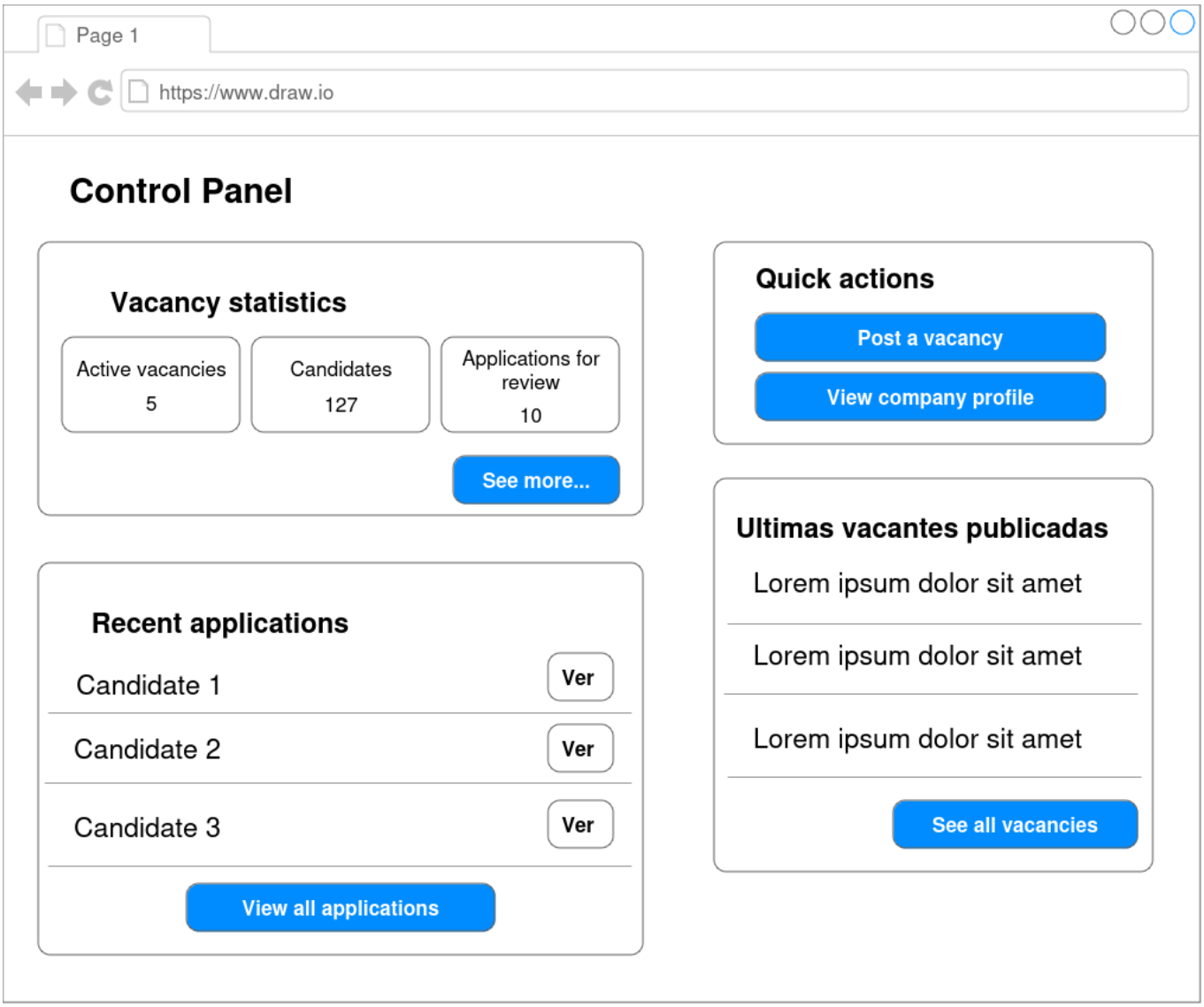


3.1.3) Login

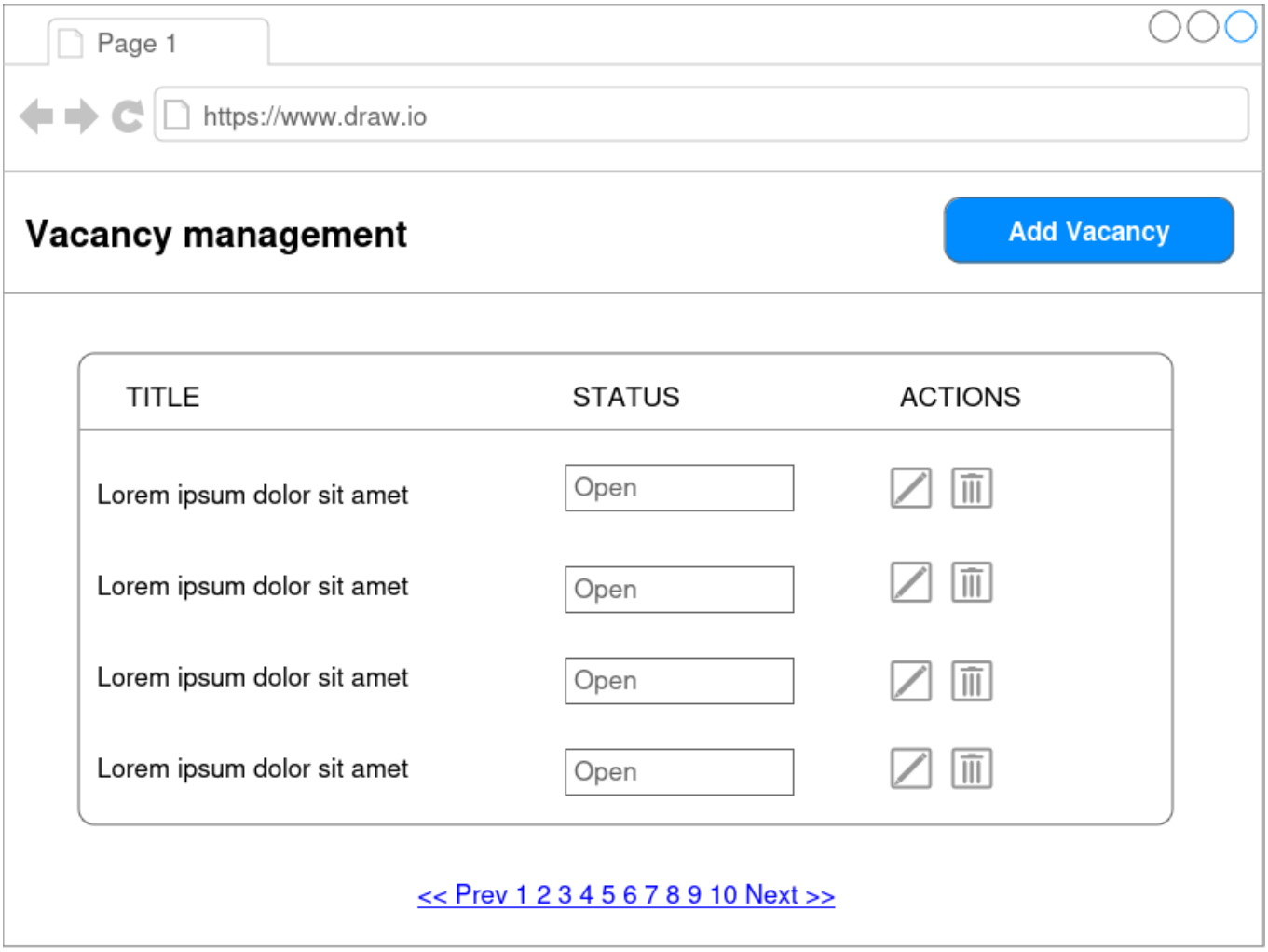

Page 1
https://www.draw.io
Login
Your gateway to professional success
E-mail address
Password
Login
Don't have an account?
Register now
More information at
Job vacancies
Be the first to apply for the best job vacancies

3.2) Company Wireframes

3.2.1) Company Dashboard

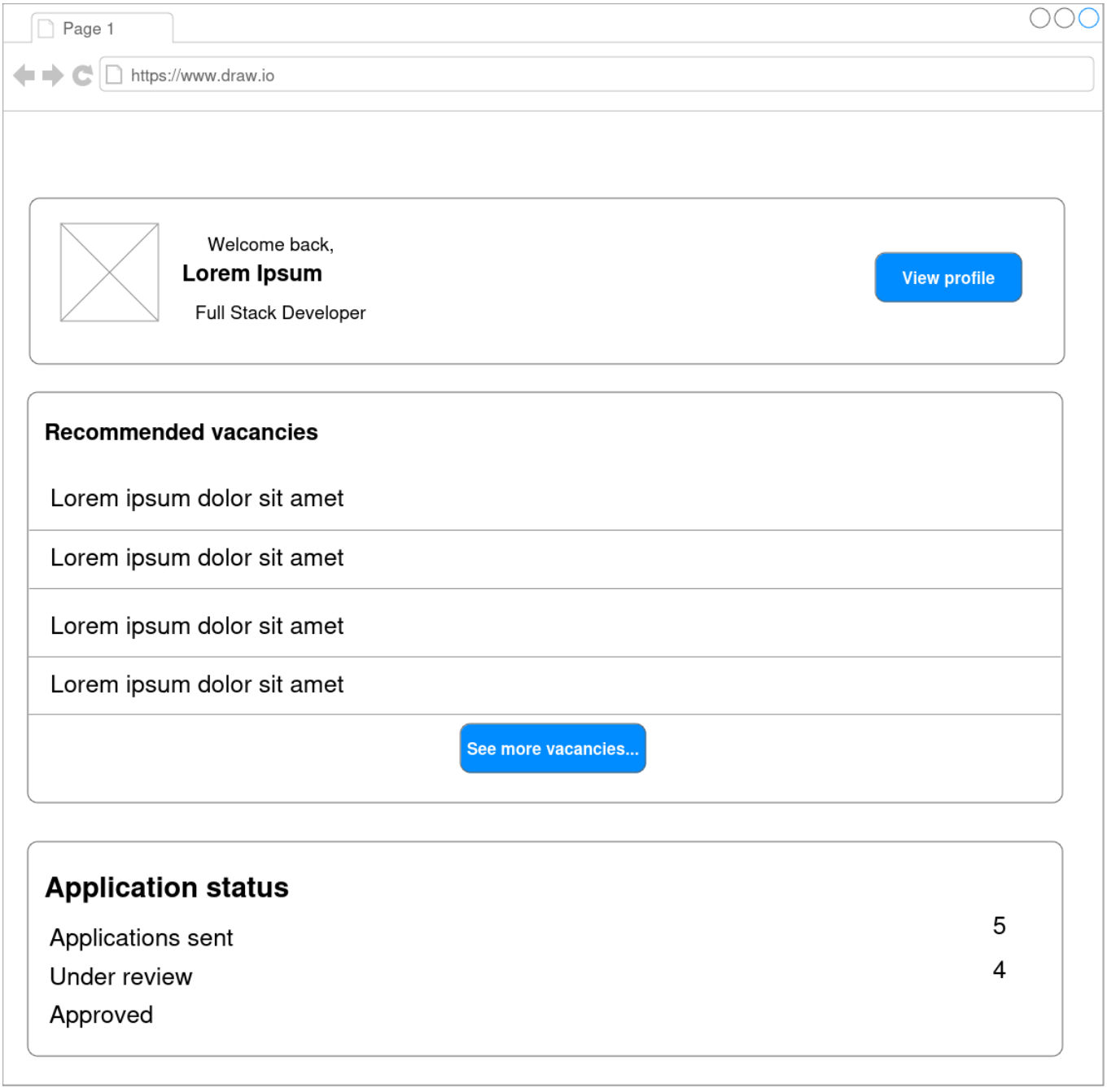


3.2.2) Vacancy management



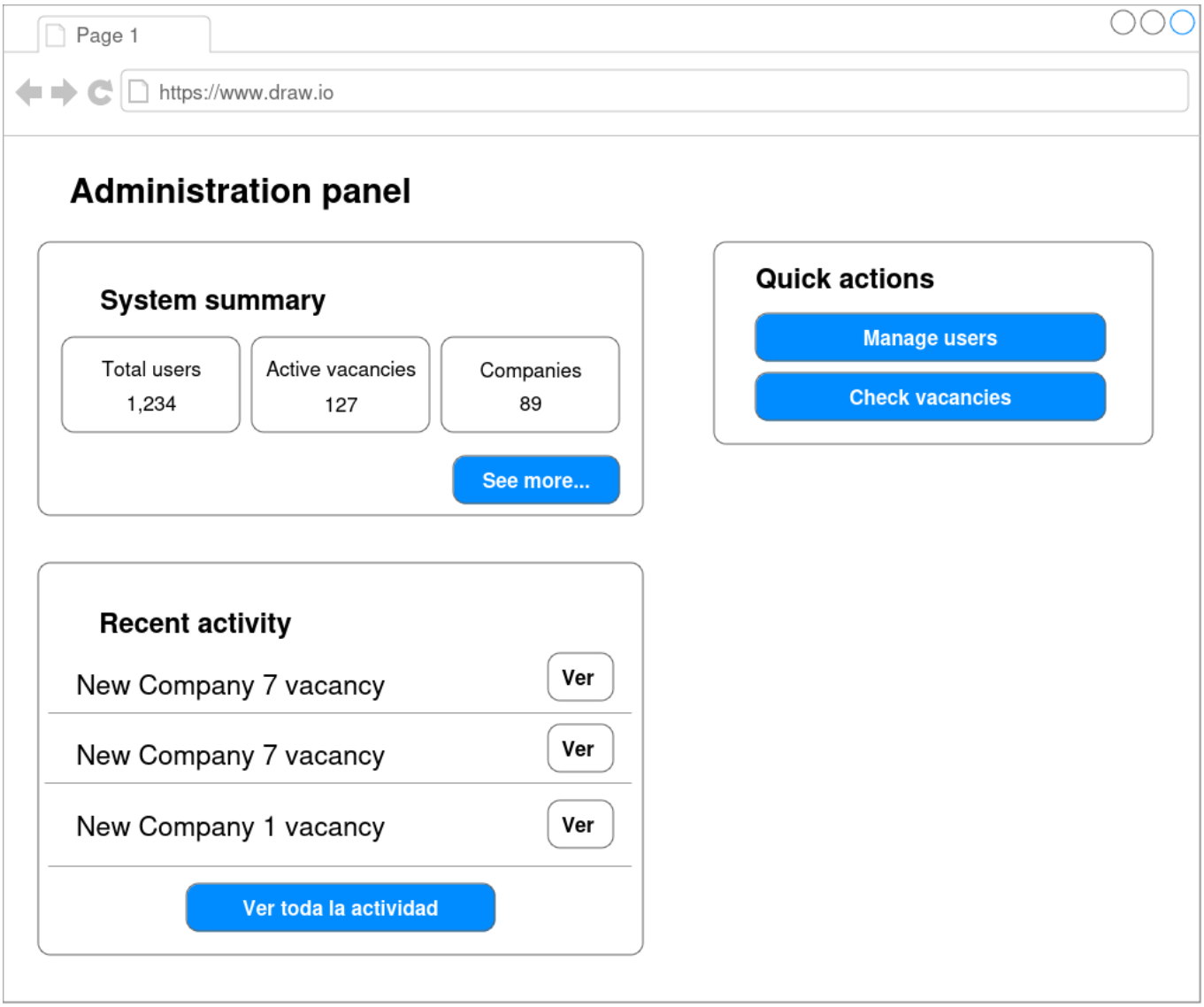
3.3) Prospect Wireframes

3.3.1) Prospect dashboard



3.4) Administrator Wireframes

3.4.1) Administrator dashboard



4) Construction

Begin building the website or application, including both design and functionality. All stages should progress simultaneously and conclude at the same time. Work should be continuous throughout the construction of the website.

SIDEBAR

```
<body>
    <section class="sidebarfool">
        <div class="container">
            <div class="sidebar active">
                <div class="menu-btn">
                    <i class="ph-bold ph-caret-left"></i>
                </div>
                <div class="head">
                    <div class="user-img">
                        <img src="img/user.jpg" alt="No photo">
                    </div>
                    <div class="user-details">
                        <p class="title">Desarrollador de software</p>
                        <p class="name">Alejandro</p>
                    </div>
                </div>
```

- **Sidebar Section:** Main sidebar container with the sidebarfool class. Inside this section, .container holds the sidebar which will be styled and made interactive.

CSS (SIDEBAR)

```
* {
    margin: 0;
    padding: 0;
    font-family: "Inter", sans-serif;
    box-sizing: border-box;
}

body {
    background-color: #efeff6;
}

.container {
    width: 100%;
    grid-template-columns: 1fr;
}
```

- **(*):** Removes default styling like margins and paddings across all elements and sets a standard font.
- **Body**: Sets a background color for the overall page.

```css
.sidebar {
    position: fixed;
    left: 0;
    top: 0;
    width: 330px;
    height: 100vh;
    display: flex;
    flex-direction: column;
    gap: 10px;
    background-color: #000;
    padding: 24px;
    transition: all 0.3s;
    z-index: 1000;
}

.sidebar .head {
    display: flex;
    gap: 20px;
    padding-bottom: 20px;
    border-bottom: 1px solid #f6f6f6;
}
```

- .**sidebar**: This is the fixed sidebar on the left, styled to take up the full viewport height with a width of 330px.
- **Transition**: Controls animation speed for expanding and contracting.

```css
.sidebar .head {
    display: flex;
    gap: 20px;
    padding-bottom: 20px;
    border-bottom: 1px solid #f6f6f6;
}

.user-img {
    width: 44px;
    height: 44px;
    border-radius: 50%;
    overflow: hidden;
}

.user-img img {
    width: 100%;
    object-fit: cover;
}

.user-details .title,
.menu .title {
    font-size: 10px;
    font-weight: 500;
    color: white;
    text-transform: uppercase;
    margin-bottom: 10px;
}
```

- .**head**: Contains the user image and details.
- .**user-img**: Styles the image as a circle, and .user-img img ensures the image fits correctly.
- **.user-details**: Contains the user role (.title) and name (.name).

```css
.menu ul li {
    position: relative;
    list-style: none;
    margin-bottom: 5px;
}

.menu ul li a {
    display: flex;
    align-items: center;
    gap: 10px;
    font-size: 14px;
    font-weight: 500;
    color: white;
    text-decoration: none;
    padding: 12px 8px;
    border-radius: 8px;
    transition: all 0.3s;
}

.menu ul li>a:hover,
.menu ul li.active>a {
    color: #000;
    background-color: #f6f6f6;
}
```

- .**menu**: Organizes the main navigation with clickable list items (li).
- **Hover:** Adds background color on hover for active states.

JAVASCRIPT

```javascript
$(".menu-btn").click(function () {
    $(".sidebar").toggleClass("active");
})
```

Toggles the sidebar width by adding or removing the .active class when the menu-btn button is clicked.

```
$(".menu > ul > li").click(function (e) {
    // Remove active class from already active siblings
    $(this).siblings().removeClass("active");
    // Add active class to the clicked item
    $(this).toggleClass("active");
    // Open submenu
    $(this).find("ul").slideToggle();
    // Open menu and slide up
    $(this).siblings().find("ul").slideUp();
    // remove active class of submenu

});
```

Expanding Menus: Adds active class to show a submenu only for the clicked menu item and closes any other open submenus by removing their active state.

```
$(document).click(function (e) {
    // If the click is outside of the sidebar, then it closes
    if (!$(e.target).closest('.sidebar, .menu > ul > li').length) {
        // If it is outside, then it will desactive the eleements
        $(".menu > ul > li").removeClass("active");
        $(".menu > ul > li ul").slideUp(); // close any submenu

    }
});
```

Closes any open menus or submenus when the user clicks outside the sidebar area, ensuring a smooth and uncluttered user interface.

HOME PAGE

Document Declaration

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/index.css">
    <title>TalentBridge</title>
</head>
```

- Defines the document type and language.
- Sets character encoding and viewport for responsive design.
- Links to the external CSS stylesheet.

Header Section

```
<header class="header">
  <div class="main-nav-bar">
    <div class="marginbar">
      <svg xmlns="http://www.w3.org/2000/svg" class="icon icon-tabler icon-tabler-brand-cucumbe
      <path stroke="none" d="M0 0h24v24H0z" fill="none"/>
      <path d="M20 10.99c-.01 5.52 -4.48 10 -10 10.01v-2.26l-.01 -.01c-4.28 -1.11 -6.86 -5.47 -
      <path d="M10.5 8l-.5 -1" />
      <path d="M13.5 14l.5 1" />
      <path d="M9 12.5l-1 .5" />
      <path d="M11 14l-.5 1" />
      <path d="M13 8l.5 -1" />
      <path d="M16 12.5l-1 -.5" />
      <path d="M9 10l-1 -.5" />
    </svg>
    </div>
    <nav class="main-nav">
      <a href="#inicio">Home</a>
      <a href="#caracteristicas">Vacancies</a>
      <a href="#testimonios">Companies</a>
      <!-- Language Switch Links -->
      <a href="/Outsourcing/en/index.php">English</a>
      <a href="/Outsourcing/es/index.php">Español</a>
    </nav>
  </div>
```

- Contains the main navigation bar (main-nav-bar), an SVG icon, and navigation links.

Hero Section

```
<section>
<div>
<div class="hero-section slider-box">
  <ul>
    <li>
    <img src="img/slider1.png" alt="No image" class="hero-image">
    </li>
    <li>
    <img src="img/body1.png" alt="No image" class="hero-image">
    </li>
    <li>
    <img src="img/slider2.png" alt="No image" class="hero-image">
    </li>
    <li>
    <img src="img/slider3.png" alt="No image" class="hero-image">
    </li>
  </ul>
```

- Displays a slider (slider-box) with multiple images that slide automatically

```
<div class="hero-overlay">
  <div class="containerslide">
      <div class="card-container-slide">
          <div class="cardslide">
              <div class="card-content-slide cardbackground">
                  <h2 class="card-title-slide slide-title-left">Join this amazing community!</h2>
                  <p>Connect with the talent you need, whenever you need it.</p>
                  <a href="login.php" class="slide-button">Log In</a>
                  <a href="registro.php" class="slide-button">Register</a>
              </div>
          </div>
          <div class="cardslide">
              <div class="card-content">
                  <h2 class="slide-title-right">TalentBridge</h2>
              </div>
          </div>
      </div>
  </div>
```

- Contains a title, descriptive text, and buttons for "login" and "register".

About us section

```
<body>
<section>
<div class="container">
    <div class="design-section">
        <h1 class="textcolor">Who are we?</h1>
        <p class="textbody textcolor">
          TalentBridge specializes in providing outsourcing services that help companies optimize their operations and reduce costs.
          Our focus is on delivering personalized solutions in various key areas.
        </p>
    </div>
</div>
```

- Provides information about the company and its services.

```
<div class="image-container">
        <img src="img/trabajo1.png" alt="Background Image" class="background-image">
        <div class="overlay-container">
            <h1>Explore the latest job updates</h1>
            <p>Discover amazing exclusive offers for you.</p>
            <a href="registro.php" class="slide-button">Start Now</a>
        </div>
</div>
```

- Contains an image with a text overlay encouraging users to explore job updates.

Services section

```
<!-- SERVICES WE OFFER -->
 <div class="containerservices">
        <h1>Take a look</h1>
        <div class="card-container">
            <div class="card">
                <img src="img/servicio1.png" alt="Photo of vacancy x" class="card-image">
                <div class="card-content">
                    <h2 class="card-title">Vacancies</h2>
                    <p>Search for vacancies in various fields and levels. Find the ideal opportunity to boost your career.</p>
                    <a href="#" class="card-link">Learn more</a>
                </div>
            </div>
        </div>
  </div>
```

- Displays a card with an image, title, and description for available job vacancies.

Footer

```
<section class="footer-down">
  <p>All rights reserved to TalentBridge</p>
  <p>For any questions or issues, please contact 664 358 0968</p>
</section>
```

- Provides copyright information and a contact number for the TalentBridge team.

CSS HOME PAGE

```
/* General styles */
html, body{
    overflow-x: hidden;
    background-color:    #f7f7f9   ;
  }

  body {
      margin: 0;
      font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, 'Helvetica Neue', Arial, sans-serif;
    }
```

- General settings for font, background color, and overflow properties.

```
/* Main navigation bar styles */
.main-nav-bar {
  background-color:    black   ;
  color: white;
  padding: 0.5rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
  font-weight: bold;
}

.main-title {
  font-size: 1.5rem;
  font-weight: bold;
  margin-left: 1rem;
}

.main-nav {
  display: flex;
  align-items: center;
  gap: 1rem;
  margin-right: 1rem;
}

.main-nav a {
  color: white;
  text-decoration: none;
}

.main-nav a:hover {
  color: #e2e2e2;
}
```

- Styles for navigation, background color, text color, and hover effects.

```css
/* Animación del slider */
@keyframes slide {
  0% { margin-left: 0; }
  20% { margin-left: 0; }

  25% { margin-left: -100vw; }
  45% { margin-left: -100vw; }

  50% { margin-left: -200vw; }
  70% { margin-left: -200vw; }

  75% { margin-left: -300vw; }
  100% { margin-left: -300vw; }
}

/* Texto dento del slide */

.containerslide{
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
}
```

- Keyframe animation for the image slider in the hero section.

```css
/* FOOTER */

.footer-down {
  background-color:  black  ;
  color: white;
  padding: 0.5rem;
  text-align: center;
  font-weight: bold;
  margin-top: 5rem;
}
```

- Footer background color, text alignment, and padding.

LOGIN PAGE

PHP Logic and Session Management

```php
if (isset($_SESSION['user_id'])) {
    switch ($_SESSION['user_role']) {
        case 'PRO':
            header("Location: user/prospect/prospecto_dashboard.php");
            break;
        case 'EMP':
            header("Location: user/company/empresa_dashboard.php");
            break;
        case 'ADM':
            header("Location: user/admin/admin_dashboard.php");
            break;
    }
    exit();
}
```

- Starts a session and checks if the user is already logged in.
- Redirects the user to the appropriate dashboard based on their role if logged in.

```php
include_once($_SERVER['DOCUMENT_ROOT'] . '/Outsourcing/config.php');

$error = '';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $correo = mysqli_real_escape_string($conexion, $_POST['correo']);
    $contrasenia = mysqli_real_escape_string($conexion, $_POST['contrasenia']);

    $query = "SELECT numero, rol FROM Usuario WHERE correo = ? AND contrasenia = ?";
    $stmt = mysqli_prepare($conexion, $query);
    mysqli_stmt_bind_param($stmt, "ss", $correo, $contrasenia);
    mysqli_stmt_execute($stmt);
    $resultado = mysqli_stmt_get_result($stmt);

    if ($usuario = mysqli_fetch_assoc($resultado)) {
        $_SESSION['user_id'] = $usuario['numero'];
        $_SESSION['user_role'] = $usuario['rol'];


        session_set_cookie_params(1800);
        session_regenerate_id(true);

        switch ($usuario['rol']) {
            case 'PRO':
                header("Location: user/prospect/prospecto_dashboard.php");
                break;
            case 'EMP':
                header("Location: user/company/empresa_dashboard.php");
                break;
            case 'ADM':
                header("Location: user/admin/admin_dashboard.php");
                break;
        }
        exit();
    } else {
        $error = "Incorrect email or password";
    }
}
```

- Uses prepared statements to prevent SQL injection when validating user credentials.
- If valid, initializes session variables and redirects based on role.

HTML STRUCTURE (LOGIN FORM)

```
<div class="container">
    <div class="login-section">
        <button class="back-button" onclick="window.location.href='index.php'">← Back</button>
        <div class="login-form">
            <h2>Login</h2>
            <?php if ($error): ?>
                <p class="error"><?php echo $error; ?></p>
            <?php endif; ?>
            <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
                <div class="form-group">
                    <label for="correo">Email Address:</label>
                    <input type="email" id="correo" name="correo" required>
                </div>
                <div class="form-group">
                    <label for="contrasenia">Password:</label>
                    <input type="password" id="contrasenia" name="contrasenia" required>
                </div>
                <div class="form-group">
                    <input type="submit" value="Login">
                </div>
            </form>
            <div class="register-link">
                Don't have an account? <a href="registro.php">Sign up now</a>
            </div>
        </div>
    </div>
```

- Contains an error message section, email and password input fields, and a submit button.
- Includes a back button that navigates back to the homepage

```
<div class="features-section">
    <div class="feature active">
        <h3>Connect with Talent</h3>
        <p>Find the best professionals for your company with our advanced recruitment platform.</p>
    </div>
    <div class="feature">
        <h3>Vacancy Management</h3>
        <p>Post and manage your job listings efficiently and easily.</p>
    </div>
    <div class="feature">
        <h3>Candidate Tracking</h3>
        <p>Maintain a detailed record of candidates and their progress in the selection process.</p>
    </div>
    <div class="feature">
        <h3>Analytics and Reports</h3>
        <p>Gain valuable insights into your recruitment processes with our analysis tools.</p>
    </div>
    <div class="dots">
        <span class="dot active"></span>
        <span class="dot"></span>
        <span class="dot"></span>
        <span class="dot"></span>
    </div>
</div>
```

- A sliding section highlighting different platform features with an auto-advancing carousel.

JAVASCRIPT FOR CAROUSEL FUNCTIONALITY

```
<script>
    const features = document.querySelectorAll('.feature');
    const dots = document.querySelectorAll('.dot');
    let currentFeature = 0;

    function showFeature(index) {
        features[currentFeature].classList.remove('active');
        dots[currentFeature].classList.remove('active');
        features[index].classList.add('active');
        dots[index].classList.add('active');
        currentFeature = index;
    }

    function nextFeature() {
        let next = (currentFeature + 1) % features.length;
        showFeature(next);
    }

    setInterval(nextFeature, 5000);

    dots.forEach((dot, index) => {
        dot.addEventListener('click', () => {
            showFeature(index);
        });
    });
</script>
```

- Automatically advances to the next feature every 5 seconds.
- Allows users to manually select features by clicking dots.

CSS (LOGIN)

```
body, html {
    margin: 0;
    padding: 0;
    font-family: 'Roboto', sans-serif;
    height: 100%;
    background-color: #f5f5f5;
}
.container {
    display: flex;
    height: 100%;
}
```

- Sets the background color, font family, and centers the content

```
.login-section {
    flex: 1;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #ffffff;
}
.login-form {
    width: 80%;
    max-width: 400px;
}
.login-form h2 {
    text-align: center;
    margin-bottom: 30px;
    color: #000000;
}
.form-group {
    margin-bottom: 20px;
}
.form-group label {
    display: block;
    margin-bottom: 5px;
    color: #000000;
}
.form-group input {
    width: 100%;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
    font-size: 16px;
}
.form-group input[type="submit"] {
    background-color: #000000;
    color: #fff;
    cursor: pointer;
    transition: background-color 0.3s ease;
}
.form-group input[type="submit"]:hover {
    background-color: #555;
}
```

- Styles the login form, button hover effects, and error messages.

```
.features-section {
    flex: 1;
    background-color: #161616;
    color: #fff;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    position: relative;
    overflow: hidden;
}
.feature {
    text-align: center;
    padding: 20px;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 80%;
    opacity: 0;
    transition: opacity 0.5s ease;
}
```

- Adds a background color, centers text, and handles the carousel dot style.

REGISTER PAGE

```php
$correo = filter_var($_POST['correo'], FILTER_VALIDATE_EMAIL);
if (!$correo || !preg_match('/\.[a-z]{2,}$/i', $_POST['correo'])) {
    $error = "The email address is not valid.";
}

$contrasenia = $_POST['contrasenia'];
$confirmar_contrasenia = $_POST['confirmar_contrasenia'];
if (strlen($contrasenia) < 5) {
    $error = "The password must be at least 5 characters long.";
} elseif ($contrasenia !== $confirmar_contrasenia) {
    $error = "The passwords do not match.";
}

$numTel = preg_replace('/[^0-9]/', '', $_POST['numTel']);
if (strlen($numTel) != 10) {
    $error = "The phone number must have 10 digits.";
}
```

- **Common Validations**: Performed for both registration types:

```php
if (empty($error)) {
    if ($tipo_registro == 'prospecto') {
        // Specific validations for prospect
        $nombre = preg_replace('/[^a-zA-Z]/', '', $_POST['nombre']);
        $primerApellido = preg_replace('/[^a-zA-Z]/', '', $_POST['primerApellido']);
        $segundoApellido = preg_replace('/[^a-zA-Z]/', '', $_POST['segundoApellido']);

        if (strlen($nombre) < 2 || strlen($primerApellido) < 2) {
            $error = "The name must be at least 2 characters long, and the first surname must be at least 2 letters long.";
        }

        $fechaNacimiento = $_POST['fechaNacimiento'];
        $resumen = mysqli_real_escape_string($conexion, $_POST['resumen']);

        // Verify age
        $edad = date_diff(date_create($fechaNacimiento), date_create('today'))->y;
        if ($edad < 18) {
            $error = "You must be over 18 years old to register.";
        }

        if (empty($error)) {
            // Insert into User table
            $query_usuario = "INSERT INTO Usuario (correo, contrasenia, rol) VALUES (?, ?, 'PRO')";
            $stmt_usuario = mysqli_prepare($conexion, $query_usuario);
            mysqli_stmt_bind_param($stmt_usuario, "ss", $correo, $contrasenia);
```

- Prospect Specific Validation: Name, surname, and date of birth, verifies minimum character requirements and age.

```
} elseif ($tipo_registro == 'empresa') {
    // Specific validations for company
    $nombre = mysqli_real_escape_string($conexion, $_POST['nombre']);
    $ciudad = mysqli_real_escape_string($conexion, $_POST['ciudad']);
    $calle = mysqli_real_escape_string($conexion, $_POST['calle']);
    $numeroCalle = intval($_POST['numeroCalle']);
    $colonia = mysqli_real_escape_string($conexion, $_POST['colonia']);
    $codigoPostal = preg_replace('/[^0-9]/', '', $_POST['codigoPostal']);
    $nombreCont = trim($_POST['nombreCont']);
    $primerApellidoCont = preg_replace('/[^a-zA-Z]/', '', $_POST['primerApellidoCont']);
    $segundoApellidoCont = preg_replace('/[^a-zA-Z]/', '', $_POST['segundoApellidoCont']);

    if (strlen($codigoPostal) != 5) {
        $error = "The postal code must have 5 digits.";
    }

    if (strlen($nombreCont) < 2 || strlen($primerApellidoCont) < 2) {
        $error = "The contact's name must be at least 2 characters long, and the first surname must be at least 2 letters long.";
    }

    if (empty($error)) {
        // Insert into User table
        $query_usuario = "INSERT INTO Usuario (correo, contrasenia, rol) VALUES (?, ?, 'EMP')";
        $stmt_usuario = mysqli_prepare($conexion, $query_usuario);
        mysqli_stmt_bind_param($stmt_usuario, "ss", $correo, $contrasenia);
```

Company Specific Validation: address fields and contact information, checks format and length of fields such as postal code, city, and contact names.

```
<?php if ($error): ?>
    <p class="error"><?php echo $error; ?></p>
<?php endif; ?>

<?php if ($success): ?>
    <p class="success"><?php echo $success; ?></p>
<?php else: ?>
```

- Displays errors or success messages at the top of the form.

HTML AND FORM STRUCTURE

```html
<form id="tipoRegistroForm">
    <div class="form-group">
        <label for="tipo_registro">Select the type of registration:</label>
        <select id="tipo_registro" name="tipo_registro">
            <option value="">Select an option</option>
            <option value="prospecto">Prospect</option>
            <option value="empresa">Company</option>
        </select>
    </div>
</form>
```

```html
<form id="registroProspectoForm" style="display:none;" method="POST">
    <input type="hidden" name="tipo_registro" value="prospecto">
    <div class="form-group">
        <label for="nombre">Name:</label>
        <input type="text" id="nombre" name="nombre" required minlength="2">
    </div>

    <div class="form-group">
        <label for="primerApellido">First Surname:</label>
        <input type="text" id="primerApellido" name="primerApellido" required pattern="[A-Za-z]{2,}" title="Enter at least 2 letters, no spaces">
    </div>

    <div class="form-group">
        <label for="segundoApellido">Second Surname:</label>
        <input type="text" id="segundoApellido" name="segundoApellido" pattern="[A-Za-z]*" title="Only letters allowed, no spaces">
    </div>

    <div class="form-group">
        <label for="fechaNacimiento">Date of Birth:</label>
        <input type="date" id="fechaNacimiento" name="fechaNacimiento" required max="<?php echo date('Y-m-d', strtotime('-18 years')); ?>">
```

```html
<form id="registroEmpresaForm" style="display:none;" method="POST">
    <input type="hidden" name="tipo_registro" value="empresa">
    <div class="form-group">
        <label for="nombre">Company Name:</label>
        <input type="text" id="nombre" name="nombre" required>
    </div>

    <div class="form-group">
        <label for="ciudad">City:</label>
        <input type="text" id="ciudad" name="ciudad" required>
    </div>

    <div class="form-group">
        <label for="calle">Street:</label>
        <input type="text" id="calle" name="calle" required>
    </div>

    <div class="form-group">
        <label for="numeroCalle">Street Number:</label>
        <input type="number" id="numeroCalle" name="numeroCalle" required>
    </div>

    <div class="form-group">
        <label for="colonia">Neighborhood:</label>
        <input type="text" id="colonia" name="colonia" required>
    </div>

    <div class="form-group">
        <label for="codigoPostal">Postal Code:</label>
        <input type="text" id="codigoPostal" name="codigoPostal" required pattern="[0-9]{5}" title="The postal code must have 5 digits">
    </div>
```

- The HTML form provides fields for **Prospect** and **Company** registration, displaying the relevant form based on user selection.

JAVASCRIPT

```
document.getElementById('tipo_registro').addEventListener('change', function() {
    var tipoRegistro = this.value;
    document.getElementById('registroProspectoForm').style.display = 'none';
    document.getElementById('registroEmpresaForm').style.display = 'none';
    if (tipoRegistro === 'prospecto') {
        document.getElementById('registroProspectoForm').style.display = 'block';
        updateProgress('registroProspectoForm');
    } else if (tipoRegistro === 'empresa') {
        document.getElementById('registroEmpresaForm').style.display = 'block';
        updateProgress('registroEmpresaForm');
    }
});
```

- **Form Toggle:** Displays the relevant form based on user selection.

```
function updateProgress(formId) {
    const form = document.getElementById(formId);
    const inputs = form.querySelectorAll('input:not([type="submit"]), select, textarea');
    const progressBar = document.getElementById('registrationProgress');

    function calculateProgress() {
        let filledInputs = 0;
        inputs.forEach(inp => {
            if (inp.value.trim() !== '') {
                filledInputs++;
            }
        });
        const progress = (filledInputs / inputs.length) * 100;
        progressBar.style.width = `${progress}%`;
    }

    inputs.forEach(input => {
        input.addEventListener('input', calculateProgress);
    });

    calculateProgress(); // Calcula el progreso inicial
}
```

- **Progress Bar Update**: Calculates the percentage of completed fields and updates the progress bar dynamically.

```
const features = document.querySelectorAll('.feature');
const dots = document.querySelectorAll('.feature-nav-dot');
let currentFeature = 0;

function showFeature(index) {
    features[currentFeature].classList.remove('active');
    dots[currentFeature].classList.remove('active');
    features[index].classList.add('active');
    dots[index].classList.add('active');
    currentFeature = index;
}

function nextFeature() {
    let next = (currentFeature + 1) % features.length;
    showFeature(next);
}

setInterval(nextFeature, 5000);

dots.forEach((dot, index) => {
    dot.addEventListener('click', () => showFeature(index));
});
```

- **Feature Carousel**: Cycles through various features, updating every five seconds.

B. COMPANY PAGES

DASHBOARD (company_dashboard.php)

The file first verifies that the user is logged in and has a company role. If they aren't, they're redirected to the login page. This access control protects the dashboard from unauthorized access.

```
<?php
session_start();
include_once($_SERVER['DOCUMENT_ROOT'] . '/Outsourcing/config.php');
require_once 'check_membership.php';

// Verificar si el usuario está logueado y es una empresa
if (!isset($_SESSION['user_id']) || $_SESSION['user_role'] !== 'EMP') {
    header(header: "Location: login.php");
    exit();
}
```

The code retrieves basic information about the company, including profile data from the Empresa table, to personalize the dashboard. To give the company an overview, statistics are shown. A stored procedure, obtenerDatosEmpresa, retrieves key metrics like active vacancies, total candidates, and applications that need review.

```
// Obtener información de la empresa
$query_empresa = "SELECT * FROM Empresa WHERE usuario = ?";
$stmt_empresa = mysqli_prepare(mysql: $conexion, query: $query_empresa);
mysqli_stmt_bind_param(statement: $stmt_empresa, types: "i", var: &$_SESSION['user_id']);
mysqli_stmt_execute(statement: $stmt_empresa);
$resultado_empresa = mysqli_stmt_get_result(statement: $stmt_empresa);
$empresa = mysqli_fetch_assoc(result: $resultado_empresa);

// Obtener estadísticas de la empresa
$query_stats = "CALL obtenerDatosEmpresa(?)";
$stmt_stats = mysqli_prepare(mysql: $conexion, query: $query_stats);
mysqli_stmt_bind_param(statement: $stmt_stats, types: "i", var: &$empresa['numero']);
mysqli_stmt_execute(statement: $stmt_stats);
$resultado_stats = mysqli_stmt_get_result(statement: $stmt_stats);
$stats = mysqli_fetch_assoc(result: $resultado_stats);
```

The dashboard also shows the company's most recent job postings and recent prospect applications.

```
<div class="dashboard-card">
    <h2>Recent Applications</h2>
    <?php while ($solicitud = mysqli_fetch_assoc(result: $resultado_solicitudes)): ?>
        <div class="list-item">
            <p><strong><?php echo htmlspecialchars(string: $solicitud['nombre'] . ' ' .
            <p>Vacancy: <?php echo htmlspecialchars(string: $solicitud['titulo']); ?></
        </div>
    <?php endwhile; ?>
    <a href="revisar_solicitudes.php" class="btn view-all-btn">View More Applications</
</div>
<div class="dashboard-card">
    <h2>Recent Vacancies</h2>
    <?php while ($vacante = mysqli_fetch_assoc(result: $resultado_vacantes)): ?>
        <div class="list-item">
            <a href="detalle_vacante.php?id=<?php echo $vacante['numero']; ?>">
                <?php echo htmlspecialchars(string: $vacante['titulo']); ?>
            </a>
        </div>
    <?php endwhile; ?>
    <a href="gestionar_vacantes.php" class="btn view-all-btn">View All Vacancies</a>
```

Vacancy_management.php

It identifies the user's associated company by checking the Empresa table. If no company is found, the system returns an error message. This ensures that only companies can manage job vacancies.

```php
// Obtener el número de la empresa asociada al usuario
$query_empresa = "SELECT numero FROM Empresa WHERE usuario = $user_id";
$result_empresa = mysqli_query(mysql: $conexion, query: $query_empresa);

if (mysqli_num_rows(result: $result_empresa) == 0) {
    die("Error: A company associated with this user was not found.");
}

$empresa = mysqli_fetch_assoc(result: $result_empresa);
$empresa_id = $empresa['numero'];
```

The page lists each job vacancy associated with the company, displaying the job title, status (Active or Inactive), and a link to view details. The status depends on the closing date of each job, showing "Active" for vacancies that are still open and "Inactive" if the closing date has passed.

Also, If a search term is entered, it filters the job vacancies by matching the term in the job title. This helps the company quickly find specific vacancies.

```php
// Obtener el término de búsqueda si existe
$search = isset($_GET['search']) ? mysqli_real_escape_string(mysql: $conexion, string: $_GET['search']) : '';

// Consulta para obtener las vacantes de la empresa
$query = "SELECT numero, titulo, fechaCierre FROM Vacante WHERE empresa = $empresa_id";
if (!empty($search)) {
    $query .= " AND titulo LIKE '%$search%'";
}
$query .= " ORDER BY fechaCierre DESC";

$result = mysqli_query(mysql: $conexion, query: $query);
```

The search bar allows users to search for specific job vacancies by title. It's implemented as an HTML form, where users can type a keyword and click "Search" to filter job listings. This feature helps companies locate specific vacancies quickly.

```
<div class="search-container">
    <form action="" method="GET" style="display: flex; width: 100%;">
        <input type="text" name="search" placeholder="Search vacancies..."
            value="<?php echo htmlspecialchars(string: $search); ?>">
        <button type="submit">Search</button>
    </form>
</div>
```

Each vacancy appears in a structured list (vacantes-list), displaying key details like the job title, status (Active/Inactive), and a link to view details.

```
<div class="vacantes-list">
    <?php
    if (mysqli_num_rows(result: $result) > 0) {
        while ($row = mysqli_fetch_assoc(result: $result)) {
            $is_active = strtotime(datetime: $row['fechaCierre']) >= strtotime(datetime: date(format: 'Y-m-d'
            ?>
            <div class="vacante-item">
                <div class="vacante-details">
                    <div class="vacante-title"><?php echo htmlspecialchars(string: $row['titulo']); ?></div>
                    <span class="vacante-status <?php echo $is_active ? 'status-active' : 'status-inactive';
                        <?php echo $is_active ? 'Active' : 'Inactive'; ?>
                    </span>
                </div>
                <a href="detalle_vacante.php?id=<?php echo $row['numero']; ?>" class="vacante-link">View Deta
            </div>
            <?php
        }
    } else {
        echo "<p>No job vacancies found.</p>";
    }
    ?>
```

Vacancy_details.php

Retrieves job vacancy details, including contract type and company information, using the job ID from the URL. If the ID is missing or invalid, it redirects the user back to the list of vacancies.

```
// Obtener el ID de la vacante de la URL
$vacante_id = isset($_GET['id']) ? intval(value: $_GET['id']) : 0;

if ($vacante_id === 0) {
    header(header: "Location: gestionar_vacantes.php");
    exit();
}
```

Shows job details like title, description, salary (if available), hiring type (direct or indirect), number of employees needed, start and end dates, days remaining, status (active or inactive), contract type, and company name.

BackEnd

```php
// Obtener los detalles de la vacante
$query_vacante = "SELECT v.*, tc.nombre AS tipo_contrato_nombre, tc.descripcion AS tipo_contrato_descripcion,
                         e.nombre AS nombre_empresa
                  FROM Vacante v
                  JOIN Tipo_Contrato tc ON v.tipo_contrato = tc.codigo
                  JOIN Empresa e ON v.empresa = e.numero
                  WHERE v.numero = ?";
$stmt_vacante = mysqli_prepare(mysql: $conexion, query: $query_vacante);
mysqli_stmt_bind_param(statement: $stmt_vacante, types: "i", var: &$vacante_id);
mysqli_stmt_execute(statement: $stmt_vacante);
$resultado_vacante = mysqli_stmt_get_result(statement: $stmt_vacante);
$vacante = mysqli_fetch_assoc(result: $resultado_vacante);

if (!$vacante) {
    header(header: "Location: gestionar_vacantes.php");
    exit();
}
```

FrontEnd

```php
<main class="main-content">
    <div class="vacante-details">
        <div class="vacante-header">
            <h1 class="vacante-title"><?php echo htmlspecialchars(string: $vacante['titulo']); ?></h1>
            <button class="btn-edit" onclick="toggleEditForm()">Edit Job</button>
        </div>
        <div class="vacante-info">
            <div class="info-item">
                <div class="info-label">Description:</div>
                <div class="info-value"><?php echo nl2br(string: htmlspecialchars(string: $vacante['descrip
            </div>
            <div class="info-item">
                <div class="info-label">Salary:</div>
                <div class="info-value">
                    <?php echo $vacante['salario'] ? '$' . number_format(num: $vacante['salario'], decimals
                </div>
            </div>
            <div class="info-item">
                <div class="info-label">Hiring Type:</div>
                <div class="info-value"><?php echo $vacante['es_directo'] ? 'Direct' : 'Indirect'; ?></div>
```

There is an "Edit Job" button that reveals a form for editing the job details. The form updates fields like title, description, salary, hiring type, start/end dates, status, etc. When submitted, it saves the updates to the database and refreshes the page with the new details.

```php
// Procesar la actualización de la vacante si se envió el formulario
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $titulo = mysqli_real_escape_string(mysql: $conexion, string: $_POST['titulo']);
    $descripcion = mysqli_real_escape_string(mysql: $conexion, string: $_POST['descripcion']);
    $salario = is_numeric(value: $_POST['salario']) ? $_POST['salario'] : 0;
    $es_directo = isset($_POST['es_directo']) ? 1 : 0;
    $cantEmpleados = intval(value: $_POST['cantEmpleados']);
    $fechaInicio = mysqli_real_escape_string(mysql: $conexion, string: $_POST['fechaInicio']);
    $fechaCierre = mysqli_real_escape_string(mysql: $conexion, string: $_POST['fechaCierre']);
    $estado = isset($_POST['estado']) ? 1 : 0;
    $tipo_contrato = mysqli_real_escape_string(mysql: $conexion, string: $_POST['tipo_contrato']);

    $query_update = "UPDATE Vacante SET
                    titulo = ?, descripcion = ?, salario = ?, es_directo = ?,
                    cantEmpleados = ?, fechaInicio = ?, fechaCierre = ?,
                    estado = ?, tipo_contrato = ?
                    WHERE numero = ?";
    $stmt_update = mysqli_prepare(mysql: $conexion, query: $query_update);
    mysqli_stmt_bind_param(statement: $stmt_update, types: "ssdiissssi", var: &$titulo, vars: &$descripcion,
                           $cantEmpleados, $fechaInicio, $fechaCierre, $estado, $tipo_contrato, $vacante_id);
```

Get_applications.php

The page loads the list of vacancies for the company, so the user can select a specific vacancy to view its applications.

```php
// Obtener las vacantes de la empresa
$sql_vacantes = "SELECT numero, titulo FROM Vacante WHERE empresa = $empresa_id";
$result_vacantes = $conexion->query(query: $sql_vacantes);
```

It includes a helper function that gets the name of each application status (e.g., "Pending," "Approved") from the database.

```php
// Función para obtener el nombre del estatus
0 references
function obtener_nombre_estatus($codigo): mixed {
    global $conexion;
    $sql = "SELECT nombre FROM Estatus_Solicitud WHERE codigo = '$codigo'";
    $result = $conexion->query(query: $sql);
    if ($result->num_rows > 0) {
        return $result->fetch_assoc()['nombre'];
    }
    return $codigo;
}
```

For each application, it displays a button that shows the applicant's full name, phone number, and the status of their application. Depending on the current status, the script shows different buttons to approve, reject, or advance the application to a contract.

```php
$prospecto_id = $_GET['prospecto_id'];
$sql = "SELECT * FROM Prospecto WHERE numero = $prospecto_id";
$result = $conexion->query(query: $sql);
?>
```

```php
<div class="profile-header">
    <h2><?= $prospecto['nombre'] . ' ' . $prospecto['primerApellido'] . ' ' . $prospecto['segundoApellido'];
    <p class="dob">📅 Date of Birth: <?= $prospecto['fechaNacimiento']; ?></p>
    <p class="phone">📞 Phone: <?= $prospecto['numTel']; ?></p>
    <p class="summary">💼 <?= $prospecto['resumen']; ?></p>
</div>

<!-- Studied Degrees -->
<?php
$sql_carreras = "SELECT c.nombre as nombre_carrera, ce.anioConcluido
                 FROM Carreras_estudiadas ce
                 JOIN Carrera c ON ce.carrera = c.codigo
                 WHERE ce.prospecto = $prospecto_id";
$result_carreras = $conexion->query(query: $sql_carreras);
?>
<?php if ($result_carreras->num_rows > 0): ?>
    <div class="profile-section">
        <h3>🎓 Studied Degrees</h3>
        <ul class="career-list">
            <?php while($carrera = $result_carreras->fetch_assoc()): ?>
                <li><?= $carrera['nombre_carrera'] . " (Year of Completion: " . $carrera['anioConcluido'] .
            <?php endwhile; ?>
```

This code updates the status of a job application in the database

```php
$solicitud_prospecto = $_POST['solicitud_prospecto'];
$solicitud_vacante = $_POST['solicitud_vacante'];
$nuevo_estatus = $_POST['nuevo_estatus'];

$sql_update = "UPDATE Solicitud SET estatus = '$nuevo_estatus'
               WHERE prospecto = $solicitud_prospecto AND vacante = $solicitud_vacante";

if ($conexion->query($sql_update) === TRUE) {
    echo "Successfully updated status";
} else {
    echo "Error updating status: " . $conexion->error;
}
```

VacancyPost.php

The form includes fields for job title, description, salary, employment type, number of employees, dates, contract type, requested careers, and requirements

```php
<?php endif; ?>
<section class="vacante-form">
    <form action="" method="post">
        <div class="form-group">
            <label for="titulo">Job Title:</label>
            <input type="text" id="titulo" name="titulo" required maxlength="30">
        </div>
        <div class="form-group">
            <label for="descripcion">Description:</label>
            <textarea id="descripcion" name="descripcion" required maxlength="250"></textarea>
        </div>
        <div class="form-group">
            <label for="salario">Salary (optional):</label>
            <input type="number" id="salario" name="salario">
        </div>
        <div class="form-group">
            <label>Is it direct hiring?</label>
            <div class="radio-group">
                <input type="radio" id="es_directo_si" name="es_directo" value="1" required>
                <label for="es_directo_si">Yes</label>
                <input type="radio" id="es_directo_no" name="es_directo" value="0" required>
                <label for="es_directo_no">No</label>
            </div>
        </div>
        <div class="form-group">
```

If the form is submitted (POST request), it collects and sanitizes the data (like job title, description, salary, and dates).

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Recoger y sanitizar los datos del formulario
    $titulo = mysqli_real_escape_string(mysql: $conexion, string: $_POST['titulo']);
    $descripcion = mysqli_real_escape_string(mysql: $conexion, string: $_POST['descripcion']);
    $salario = isset($_POST['salario']) ? intval(value: $_POST['salario']) : null;
    $es_directo = $_POST['es_directo'] == '1' ? 1 : 0;
    $cantEmpleados = intval(value: $_POST['cantEmpleados']);
    $fechaInicio = mysqli_real_escape_string(mysql: $conexion, string: $_POST['fechaInicio']);
    $fechaCierre = mysqli_real_escape_string(mysql: $conexion, string: $_POST['fechaCierre']);
    $tipo_contrato = mysqli_real_escape_string(mysql: $conexion, string: $_POST['tipo_contrato']);
    $carreras_solicitadas = isset($_POST['carreras']) ? $_POST['carreras'] : [];
    $requerimientos = isset($_POST['requerimientos']) ? $_POST['requerimientos'] : [];
```

It validates the start and end dates to ensure they are in the correct order.

```php
if ($fecha_inicio < $fecha_actual) {
    $mensaje = "Error: Start date must be later than current date.";
} elseif ($fecha_cierre <= $fecha_inicio) {
    $mensaje = "Error: The closing date must be after the start date.";
} else {
```

If the validation passes, the system attempts to insert the job vacancy into the database.

```php
// Insertar la vacante
$query_insertar = "INSERT INTO Vacante (titulo, descripcion, salario, es_directo, cantEmpleados, fechaInicio
$stmt_insertar = mysqli_prepare(mysql: $conexion, query: $query_insertar);
mysqli_stmt_bind_param(statement: $stmt_insertar, types: "sssiissisi", var: &$titulo, vars: &$descripcion, $
mysqli_stmt_execute(statement: $stmt_insertar);

$vacante_id = mysqli_insert_id(mysql: $conexion);


// Insertar carreras solicitadas
if (!empty($carreras_solicitadas)) {
    $query_carrera = "INSERT INTO Carreras_solicitadas (vacante, carrera) VALUES (?, ?)";
    $stmt_carrera = mysqli_prepare(mysql: $conexion, query: $query_carrera);
    foreach ($carreras_solicitadas as $carrera) {
        mysqli_stmt_bind_param(statement: $stmt_carrera, types: "is", var: &$vacante_id, vars: &$carrera);
        mysqli_stmt_execute(statement: $stmt_carrera);
    }
}
```

Membership.php

The system retrieves the current membership details for the logged-in company. It checks if the membership has expired by comparing the expiration date with the current date.

```php
// Obtener información de la membresía actual
$query = "SELECT * FROM Membresia WHERE empresa = $user_id ORDER BY fechaVencimiento DESC LIMIT 1";
$result = mysqli_query(mysql: $conexion, query: $query);
$membresia = mysqli_fetch_assoc(result: $result);

// Verificar si la membresía ha expirado
$membresia_expirada = strtotime(datetime: $membresia['fechaVencimiento']) < time();
```

This section gets all available subscription plans from the database. The plans are sorted by price and stored to be displayed later on the page.

```php
// Obtener planes de suscripción
$query = "SELECT * FROM Plan_suscripcion ORDER BY precio ASC";
$result = mysqli_query(mysql: $conexion, query: $query);
$planes = mysqli_fetch_all(result: $result, mode: MYSQLI_ASSOC);
```

This section handles the interactive part of the payment process. It includes a modal that appears when a user selects a subscription plan. The modal contains a form for entering payment details.

```html
<div id="paymentModal" class="modal">
    <div class="modal-content">
        <span class="close">&times;</span>
        <h2>Confirm Payment</h2>
        <form id="paymentForm" method="POST">
            <input type="hidden" id="plan_id" name="plan_id" value="">
            <div>
                <label for="numero_tarjeta">Card Number:</label>
                <input type="text" id="numero_tarjeta" name="numero_tarjeta" required maxlength="16">
            </div>
            <div>
                <label for="fecha_vencimiento">Expiration Date:</label>
                <input type="month" id="fecha_vencimiento" name="fecha_vencimiento" required>
            </div>
            <div>
                <label for="cvv">CVV:</label>
                <input type="text" id="cvv" name="cvv" required maxlength="3">
            </div>
            <button type="submit" class="btn">Confirm Payment</button>
```

Here, the system handles the payment process. When the user submits the payment form, it validates the credit card information, such as the card number, expiration date, and CVV. If everything is valid, it simulates a successful payment, updates the membership expiration date, and logs the renewal in the database.

```php
// Procesar el pago si se envió el formulario
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['plan_id'])) {
    $plan_id = mysqli_real_escape_string(mysql: $conexion, string: $_POST['plan_id']);
    $numero_tarjeta = mysqli_real_escape_string(mysql: $conexion, string: $_POST['numero_tarjeta']);
    $fecha_vencimiento = mysqli_real_escape_string(mysql: $conexion, string: $_POST['fecha_vencimiento']);
    $cvv = mysqli_real_escape_string(mysql: $conexion, string: $_POST['cvv']);

    // Validar los datos de la tarjeta
    if (strlen(string: $numero_tarjeta) !== 16 || !ctype_digit(text: $numero_tarjeta)) {
        $error = "The card number must be 16 digits long.";
    } elseif (strtotime(datetime: $fecha_vencimiento) <= time()) {
        $error = "The card expiration date is not valid.";
    } elseif (strlen(string: $cvv) !== 3 || !ctype_digit(text: $cvv)) {
        $error = "The CVV must be 3 digits.";
    } else {
```

C. PROSPECT PAGES

generate_contract.php

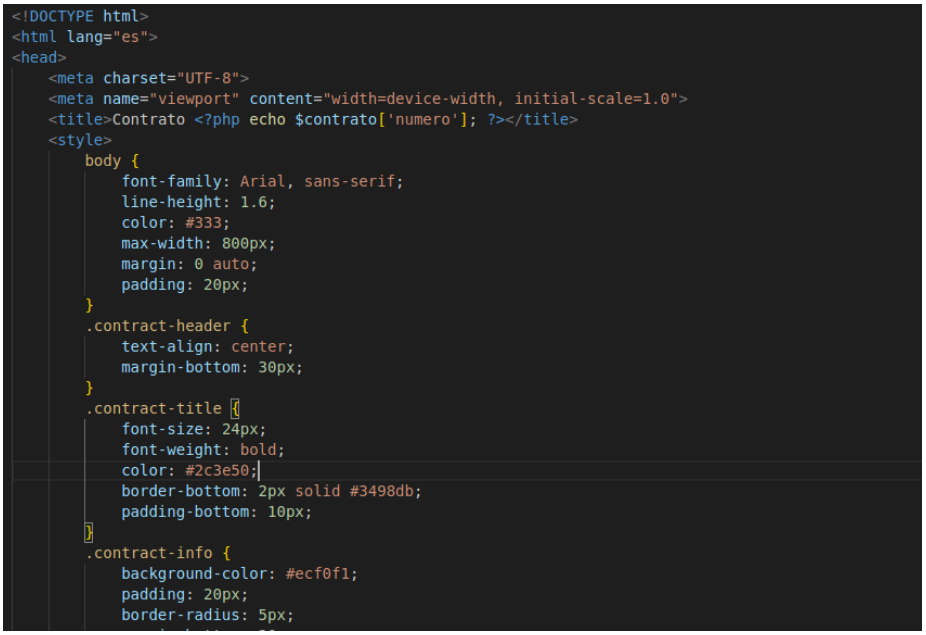
We assume that the contract number is passed as a GET parameter.

```php
<?php
include_once($_SERVER['DOCUMENT_ROOT'] . '/Outsourcing/config.php');


$numero_contrato = isset($_GET['numero']) ? intval($_GET['numero']) : 0;
$numero_contrato = 1;

if ($numero_contrato <= 0) {
    die("Número de contrato no válido");
}
```

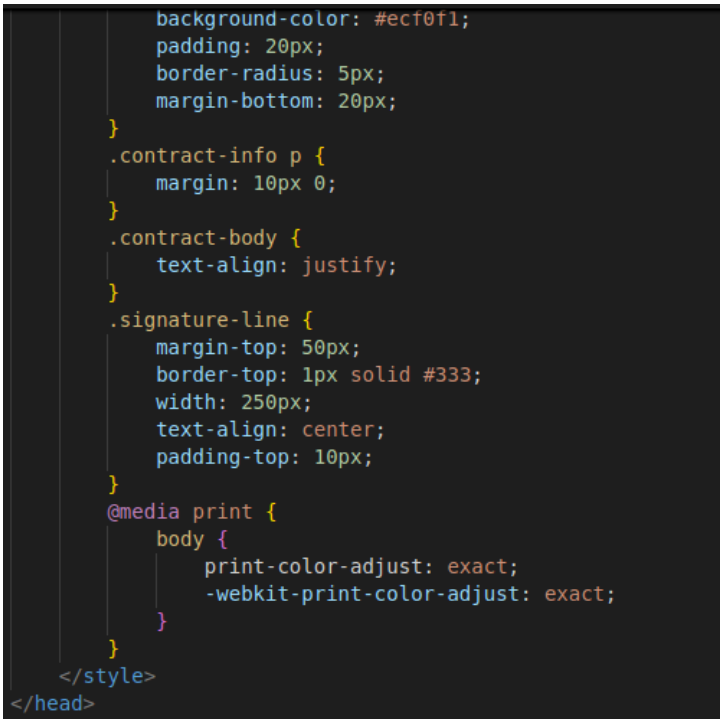
- Inquiry to obtain contract details

```php
$query = "SELECT c.numero, c.fechaInicio, c.fechaCierre,
                 p.nombre, p.primerApellido, p.segundoApellido,
                 tc.nombre AS tipo_contrato, tc.descripcion AS descripcion_contrato
          FROM Contrato c
          JOIN Prospecto p ON c.prospecto = p.numero
          JOIN Tipo_Contrato tc ON c.tipo_contrato = tc.codigo
          WHERE c.numero = ?";

$stmt = mysqli_prepare($conexion, $query);
mysqli_stmt_bind_param($stmt, "i", $numero_contrato);
mysqli_stmt_execute($stmt);
$resultado = mysqli_stmt_get_result($stmt);

if ($contrato = mysqli_fetch_assoc($resultado)) {
    $nombre_completo = $contrato['nombre'] . ' ' . $contrato['primerApellido'] .
                       ($contrato['segundoApellido'] ? ' ' . $contrato['segundoApellido'] : '');
} else {
    die("Contrato no encontrado");
}
?>
```

It allows you to consult the information of a specific contract and its type, along with the personal data of the associated person (prospect), and displays the full name if the contract is found or an error message if it is not found.



```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contrato <?php echo $contrato['numero']; ?></title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            color: #333;
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
        }
        .contract-header {
            text-align: center;
            margin-bottom: 30px;
        }
        .contract-title {
            font-size: 24px;
            font-weight: bold;
            color: #2c3e50;
            border-bottom: 2px solid #3498db;
            padding-bottom: 10px;
        }
        .contract-info {
            background-color: #ecf0f1;
            padding: 20px;
            border-radius: 5px;
```



```
            background-color: #ecf0f1;
            padding: 20px;
            border-radius: 5px;
            margin-bottom: 20px;
        }
        .contract-info p {
            margin: 10px 0;
        }
        .contract-body {
            text-align: justify;
        }
        .signature-line {
            margin-top: 50px;
            border-top: 1px solid #333;
            width: 250px;
            text-align: center;
            padding-top: 10px;
        }
        @media print {
            body {
                print-color-adjust: exact;
                -webkit-print-color-adjust: exact;
            }
        }
    </style>
</head>
```

It is generally used to display contracts in a clean and structured way, providing information such as the title, content and a signature area, as well as ensuring good presentation on screen and in print.

```
</head>
<body>
    <div class="contract-header">
        <h1 class="contract-title">Contrato de Trabajo</h1>
    </div>

    <div class="contract-info">
        <p><strong>Número de Contrato:</strong> <?php echo htmlspecialchars($contrato['numero']); ?></p>
        <p><strong>Fecha de Inicio:</strong> <?php echo htmlspecialchars($contrato['fechaInicio']); ?></p>
        <p><strong>Fecha de Cierre:</strong> <?php echo htmlspecialchars($contrato['fechaCierre']); ?></p>
        <p><strong>Nombre del Empleado:</strong> <?php echo htmlspecialchars($nombre_completo); ?></p>
        <p><strong>Tipo de Contrato:</strong> <?php echo htmlspecialchars($contrato['tipo_contrato']); ?></p>
    </div>

    <div class="contract-body">
        <h2>Descripción del Contrato</h2>
        <p><?php echo htmlspecialchars($contrato['descripcion_contrato']); ?></p>

        <h2>Términos y Condiciones</h2>
        <p>Este contrato se rige por las leyes laborales vigentes y establece los términos y condiciones de empleo entre la empresa y el empleado mencionado ante

        <p>El empleado se compromete a cumplir con sus responsabilidades y deberes según lo establecido por la empresa, y la empresa se compromete a proporcionar
    </div>

    <div class="signature-line">
        Firma del Empleado
    </div>

    <div class="signature-line">
        Firma del Representante de la Empresa
    </div>
</body>
</html>
```

This structured code allows the key data of a contract to be presented in an organized manner, making it legible and professional both on screen and in printed format.

prospect_dashboard.php

```php
<?php
session_start();
include_once($_SERVER['DOCUMENT_ROOT'] . '/Outsourcing/config.php');

// Verificar si el usuario está autenticado
if (!isset($_SESSION['user_id'])) {
    header("Location: login.php");
    exit();
}

$user_id = $_SESSION['user_id'];

// Obtener información del prospecto
$query_prospecto = "SELECT nombre, primerApellido, segundoApellido FROM Prospecto WHERE usuario = $user_id";
$result_prospecto = mysqli_query($conexion, $query_prospecto);
$prospecto = mysqli_fetch_assoc($result_prospecto);

// Obtener las últimas solicitudes del prospecto
$query_solicitudes = "
    SELECT s.*, v.titulo, e.nombre AS nombre_estatus
    FROM Solicitud s
    JOIN Vacante v ON s.vacante = v.numero
    JOIN Estatus_Solicitud e ON s.estatus = e.codigo
    WHERE s.prospecto = (SELECT numero FROM Prospecto WHERE usuario = $user_id)
    ORDER BY s.vacante DESC
    LIMIT 5
";
$result_solicitudes = mysqli_query($conexion, $query_solicitudes);
```

This code makes sure the user is authenticated, gets their information, and retrieves their latest job applications.

```php
$query_vacantes = "
    SELECT v.numero, v.titulo, e.ciudad, e.colonia
    FROM Vacante v
    JOIN Empresa e ON v.empresa = e.numero
    WHERE v.fechaCierre >= CURDATE()
    ORDER BY v.fechaInicio DESC
    LIMIT 5
";
$result_vacantes = mysqli_query($conexion, $query_vacantes);


$search_keyword = isset($_GET['search_keyword']) ? mysqli_real_escape_string($conexion, $_GET['search_keyword']) : '';
$search_city = isset($_GET['search_city']) ? mysqli_real_escape_string($conexion, $_GET['search_city']) : '';

if (!empty($search_keyword) || !empty($search_city)) {
    $query_vacantes = "
        SELECT v.numero, v.titulo, e.ciudad, e.colonia
        FROM Vacante v
        JOIN Empresa e ON v.empresa = e.numero
        WHERE v.fechaCierre >= CURDATE()
    ";

    if (!empty($search_keyword)) {
        $query_vacantes .= " AND (v.titulo LIKE '%$search_keyword%' OR v.descripcion LIKE '%$search_keyword%')";
    }

    if (!empty($search_city)) {
        $query_vacantes .= " AND e.ciudad LIKE '%$search_city%'";
    }

    $query_vacantes .= " ORDER BY v.fechaInicio DESC LIMIT 5";
    $result_vacantes = mysqli_query($conexion, $query_vacantes);
}
?>
```

Ensures that only current vacancies are displayed and allows vacancies to be dynamically filtered by keyword and city, providing a personalized and efficient search experience for the user.

```php
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dashboard - Prospecto</title>
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;700&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="css/css.css">
</head>
<body>
    <?php include 'incluides/sidebar.php'; ?>
    <!-- Main content -->
    <div class="container">
        <div class="welcome-message">
            <h1>Bienvenido, <?php echo $prospecto['nombre'] . ' ' . $prospecto['primerApellido'] . ' ' . $prospecto['segundoApellido']; ?></h1>
        </div>
```

Sets the structure and styles for a dashboard page where the authenticated user, called a "prospect," can see their name in a welcome greeting. Includes a sidebar and applies external CSS styles and Google typography for a better visual presentation.

```php
<div class="dashboard-grid">
    <div class="dashboard-section">
        <h2>Últimas novedades de tus solicitudes</h2>
        <?php if (mysqli_num_rows($result_solicitudes) > 0): ?>
            <?php while ($solicitud = mysqli_fetch_assoc($result_solicitudes)): ?>
                <div class="item">
                    <h3><?php echo htmlspecialchars($solicitud['titulo']); ?></h3>
                    <p>
                        <?php
                            switch ($solicitud['estatus']) {
                                case 'PEND': echo "Su solicitud está pendiente de revisión."; break;
                                case 'APRO': echo "¡Felicidades! Su solicitud ha sido aprobada."; break;
                                case 'RECH': echo "Lo sentimos, su solicitud ha sido rechazada."; break;
                                case 'PFRM': echo "Su contrato está pendiente de firma."; break;
                                case 'CERR': echo "Esta solicitud ha sido cerrada."; break;
                            }
                        ?>
                    </p>
                </div>
            <?php endwhile; ?>
        <?php else: ?>
            <p>No hay solicitudes recientes.</p>
        <?php endif; ?>
        <a href="ver_solicitudes.php" class="btn">Ver todas mis solicitudes</a>
    </div>
```

Presents a section in the dashboard where the prospect can see the news about their job applications. If there are requests, the title of each request is displayed along with a message indicating its current status. If there are no requests, the user is informed that there is no news. Additionally, a button is provided to view all requests on a separate page, thus improving navigation and user experience.

```php
        <div class="dashboard-section">
            <h2>Vacantes disponibles</h2>
            <form action="" method="GET" class="search-form">
                <input type="text" name="search_keyword" placeholder="Buscar vacantes..." value="<?php echo htmlspecialchars($search_keyword); ?>">
                <input type="text" name="search_city" placeholder="Ciudad" value="<?php echo htmlspecialchars($search_city); ?>">
                <button type="submit">Buscar</button>
            </form>
            <?php if (mysqli_num_rows($result_vacantes) > 0): ?>
                <?php while ($vacante = mysqli_fetch_assoc($result_vacantes)): ?>
                    <div class="item">
                        <h3><a href="detalles_vacante.php?id=<?php echo $vacante['numero']; ?>" style="color: #333; text-decoration: none;">
                            <?php echo htmlspecialchars($vacante['titulo']); ?></a></h3>
                        <p><?php echo htmlspecialchars($vacante['ciudad']) . ", " . htmlspecialchars($vacante['colonia']); ?></p>
                    </div>
                <?php endwhile; ?>
            <?php else: ?>
                <p>No hay vacantes disponibles que coincidan con tu búsqueda.</p>
            <?php endif; ?>
            <a href="buscar_vacantes.php" class="btn">Ver más vacantes</a>
        </div>
    </div>
</div>
</body>
</html>
```

Allows prospects to search for available positions on the dashboard. Provides a form to search by keyword and city, and displays the vacancies found in an organized format. Each vacancy includes a link to more details. If there are no vacancies available, the user is informed. A button is also provided to explore more vacancies, thus improving the user's search and browsing experience.

search_vacancies.php

```php
<?php
session_start();
include_once($_SERVER['DOCUMENT_ROOT'] . '/Outsourcing/config.php');


if (!isset($_SESSION['user_id'])) {
    header("Location: login.php");
    exit();
}

$user_id = $_SESSION['user_id'];

$search_keyword = isset($_GET['search_keyword']) ? mysqli_real_escape_string($conexion, $_GET['search_keyword']) : '';
$search_city = isset($_GET['search_city']) ? mysqli_real_escape_string($conexion, $_GET['search_city']) : '';

$query_vacantes = "
    SELECT v.numero, v.titulo, v.descripcion, v.salario, v.cantEmpleados, v.fechaInicio, v.fechaCierre,
           e.nombre AS empresa_nombre, e.ciudad, e.colonia, tc.nombre AS tipo_contrato
    FROM Vacante v
    JOIN Empresa e ON v.empresa = e.numero
    JOIN Tipo_Contrato tc ON v.tipo_contrato = tc.codigo
    WHERE v.fechaCierre >= CURDATE() AND v.fechaInicio <= CURDATE()
";

if (!empty($search_keyword)) {
    $query_vacantes .= " AND (v.titulo LIKE '%$search_keyword%' OR v.descripcion LIKE '%$search_keyword%')";
}

if (!empty($search_city)) {
    $query_vacantes .= " AND e.ciudad LIKE '%$search_city%'";
}
```

Starts a session and checks if a user is logged in using the user_id session variable. If this variable is not found, it redirects the user to the login page. It then gets the user's ID and searches for keywords and cities from the URL parameters, ensuring they are safe to use in a SQL query using

Next, build an SQL query to obtain information about job vacancies from a database, including details such as number, title, description, salary, number of employees, start and closing dates, company name, city, neighborhood and type of contract. The query also applies optional filters for keywords in the job title or description and for the city, if provided.

```
$query_vacantes .= " ORDER BY v.fechaInicio DESC";

$result_vacantes = mysqli_query($conexion, $query_vacantes);
?>

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Buscar Vacantes - Sistema de Outsourcing</title>
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;700&display=swap" rel="stylesheet">
```

In this snippet, the SQL query to obtain vacancies is completed by adding a sort by start date (startdate) in descending order. You then run the query using mysqli_query, which returns a result set that is stored in the $result_vacancy variable.

```
body {
    font-family: 'Roboto', sans-serif;
    line-height: 1.6;
    color: #333;
    background-color: #f4f4f4;
    margin: 0;
    margin-left: 100px;
    padding: 0;
}
.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
}
h1 {
    color: #000;
    text-align: center;
    margin-bottom: 30px;
}
.search-form {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    margin-bottom: 30px;
}
.search-form form {
    display: flex;
    gap: 10px;
}
```

This CSS code fragment establishes the visual style of a web page focused on searching for vacancies in an outsourcing system. Modern typography and a neutral color scheme are used to create a clean and professional design. The structure includes a centralized container that organizes content, with a prominent header and search form that features a white background, rounded edges, and a subtle shadow, improving usability and overall aesthetics.

```css
.search-form input[type="text"] {
    flex-grow: 1;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
}
.search-form button {
    padding: 10px 20px;
    background-color: #000;
    color: #fff;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}
.search-form button:hover {
    background-color: #333;
}
.vacantes-grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
    gap: 20px;
}
.vacante-card {
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    padding: 20px;
    transition: transform 0.3s ease;
}
.vacante-card:hover {
    transform: translateY(-5px);
}
.vacante-title {
    font-size: 18px;
    font-weight: bold;
    margin-bottom: 10px;
}
.vacante-company {
    font-weight: bold;
    color: #666;
```

The CSS snippet provides styles for a search form and a vacancy grid. The form's text fields have a 10px padding, a thin light gray border, and rounded edges, taking up the space available via the flex-grow property. The search button features a black background and white text, with rounded edges and a transition effect that changes to a dark gray on hover. The .vacancies-grid class uses a responsive grid layout that organizes vacancy cards into columns of at least 300px wide.

```css
.vacante-company {
    font-weight: bold;
    color: #666;
}
.vacante-details {
    margin-top: 10px;
    font-size: 14px;
}
.vacante-link {
    display: inline-block;
    margin-top: 10px;
    padding: 5px 10px;
    background-color: #000;
    color: #fff;
    text-decoration: none;
    border-radius: 4px;
    transition: background-color 0.3s ease;
}
.vacante-link:hover {
    background-color: #333;
}
.no-results {
    text-align: center;
    font-size: 18px;
    margin-top: 50px;
}
```

Defines styles for elements related to vacancies. The .vacante-company class applies a bold style and light gray color to company names, making them stand out visually. The vacancy details, represented by the .vacancy-details class, have a top margin of 10px and a font size of 14px, ensuring a clear and readable presentation of additional information. The .vacante-link class stylizes a link that acts as a button, featuring a black background with white text and rounded edges.

```php
<body>
    <?php include 'incluides/sidebar.php'; ?>
    <div class="container">
        <h1>Buscar Vacantes</h1>

        <div class="search-form">
            <form action="" method="GET">
                <input type="text" name="search_keyword" placeholder="Buscar por título o descripción" value="<?php echo htmlspecialchars($search_keyword); ?>">
                <input type="text" name="search_city" placeholder="Ciudad" value="<?php echo htmlspecialchars($search_city); ?>">
                <button type="submit">Buscar</button>
            </form>
        </div>
```

Este fragmento de código HTML se encarga de estructurar la sección principal de una página destinada a buscar vacantes.

And a search form contained in a is also presented with the search-form class. This form uses the GET method and allows users to search for vacancies using two input fields: one to enter a search keyword, which can be the title or description of the vacancy, and another to specify the city.

```
        <div class="vacantes-grid">
            <?php
            if (mysqli_num_rows($result_vacantes) > 0) {
                while ($vacante = mysqli_fetch_assoc($result_vacantes)) {
                    ?>
                    <div class="vacante-card">
                        <div class="vacante-title"><?php echo htmlspecialchars($vacante['titulo']); ?></div>
                        <div class="vacante-company"><?php echo htmlspecialchars($vacante['empresa_nombre']); ?></div>
                        <div class="vacante-details">
                            <p><strong>Ubicación:</strong> <?php echo htmlspecialchars($vacante['ciudad']) . ', ' . htmlspecialchars($vacante['colonia']); ?></p>
                            <p><strong>Salario:</strong> $<?php echo number_format($vacante['salario'], 2); ?></p>
                            <p><strong>Tipo de contrato:</strong> <?php echo htmlspecialchars($vacante['tipo_contrato']); ?></p>
                            <p><strong>Vacantes:</strong> <?php echo $vacante['cantEmpleados']; ?></p>
                            <p><strong>Fecha de inicio:</strong> <?php echo date('d/m/Y', strtotime($vacante['fechaInicio'])); ?></p>
                            <p><strong>Fecha de cierre:</strong> <?php echo date('d/m/Y', strtotime($vacante['fechaCierre'])); ?></p>
                        </div>
                        <a href="detalles_vacante.php?id=<?php echo $vacante['numero']; ?>" class="vacante-link">Ver detalles</a>
                    </div>
                    <?php
                }
            } else {
                echo "<p class='no-results'>No se encontraron vacantes que coincidan con tu búsqueda.</p>";
            }
            ?>
        </div>
    </div>
</body>
</html>
```

This HTML and PHP code snippet dynamically generates a grid of cards to display available vacancies in the system. A with the vacancies-grid class is used as the main container for the vacancies cards.

Within a PHP block, the database query ($result_vacancies) is checked to see if any rows are returned using mysqli_num_rows(). If results are found, a while loop is started that loops through each vacancy obtained using mysqli_fetch_assoc(). For each vacancy, a is created with the vacancy-card class.

If no vacancies are found that match the search, an indicative message is displayed in the main container, informing the user that there are no results available.

D. ADMINISTRATOR PAGES

Vacancy Management

Document Declaration

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gestión de Vacantes</title>
    <link rel="stylesheet" href="estilos.css">
</head>
```

- Defines the HTML document and sets the language as Spanish
- Links to an external CSS file (estilos.css) to style the page.

Main Content

```
<div class="main-content">
    <h1>Gestión de Vacantes</h1>
    <button id="addVacancyBtn" class="btn-add">Agregar Vacante</button>
    <div id="addVacancyModal" class="modal">
        <div class="modal-content">
            <span class="close">&times;</span>
            <h2 id="modalTitle">Agregar Nueva Vacante</h2>

            <form id="vacancyForm">
                <div class="form-group">
                    <label for="vacancyTitle">Título de la Vacante:</label>
                    <input type="text" id="vacancyTitle" name="vacancyTitle" class="input-form" required>
                </div>
                <div class="form-group">
                    <label for="vacancyStatus">Estado:</label>
                    <select id="vacancyStatus" name="vacancyStatus" class="input-form">
                        <option value="Abierta">Abierta</option>
                        <option value="Cerrada">Cerrada</option>
                    </select>
                </div>
                <button type="submit" class="btn-add" id="saveButton">Guardar Vacante</button>
            </form>
        </div>
    </div>
    <table>
        <thead>
            <tr>
                <th>TITULO</th>
                <th>ESTADO</th>
                <th>ACCIONES</th>
            </tr>
        </thead>
        <tbody id="vacantesTableBody">

        </tbody>
    </table>

    <div class="pagination" id="paginationControls">
        <button class="pagination-btn" id="prevPageBtn"><< Prev</button>
        <button class="pagination-btn" id="nextPageBtn">Next >></button>
    </div>
</div>
```

- Contains the main content area with a title ("Gestión de Vacantes") and a button to add vacancies.
- Contains a modal (addVacancyModal) with a form that allows the user to add new job vacancies, with fields for title and status (open or closed).
- A table is used to list the vacancies with columns for title, status, and actions.
- Pagination controls (paginationControls) let users navigate through multiple pages of vacancies.

JavaScript Link

```
<script src="scripted.js"></script>
```

- Links to an external JavaScript file (scripted.js) which will make the elements like the modal and pagination have their own interactivity

Vacancy Management(JavaScript)

Variables for Editing and Pagination

```
let isEditing = false;
let currentRow = null;
let currentPage = 1;
const itemsPerPage = 10;
```

- isEditing: A flag to determine if the form is in edit mode.
- currentRow: Stores the row being edited.
- currentPage: Tracks the current page in the paginated list.
- itemsPerPage: Defines the maximum number of items displayed per page.

Loading Vacancies from LocalStorage on Page Load

```
document.addEventListener("DOMContentLoaded", () => {
    const modal = document.getElementById("addVacancyModal");
    const btn = document.getElementById("addVacancyBtn");
    const span = document.getElementsByClassName("close")[0];
    const form = document.getElementById("vacancyForm");
```

- Loads saved vacancies when the page loads.
- Initializes references to the modal, form, and button elements used throughout the script.
- Calls cargarVacantes() to populate vacancies on load.

Open Modal on "Add Vacancy" Button Click

```
btn.onclick = function() {
    console.log("Modal abierto");
    isEditing = false;
    currentRow = null;
    form.reset();
    document.getElementById("modalTitle").innerText = "Agregar Nueva Vacante";
    document.getElementById("saveButton").innerText = "Guardar Vacante";
    modal.style.display = "flex";
};
```

- Opens the modal and resets the form when "Add Vacancy" is clicked.
- Sets form mode to add (not edit).
- Updates modal title and button text.

Close Modal on "X"

```
    span.onclick = function() {
        modal.style.display = "none";
    };
    window.onclick = function(event) {
        if (event.target == modal) {
            modal.style.display = "none";
        }
    };
```

- Closes the modal when the "X" icon or area outside the modal is clicked.

Handling Form Submission

```
form.onsubmit = function(event) {
    event.preventDefault();
    agregarVacante();
};
```

- Prevents default form behavior and triggers agregarVacante() to add or update a vacancy.

Loading Vacancies from LocalStorage

```
function cargarVacantes() {
    const savedVacantes = JSON.parse(localStorage.getItem("vacantes")) || [];
    console.log("Vacantes guardadas:", savedVacantes);
    mostrarPagina(currentPage, savedVacantes);
    actualizarBotonesPrevNext(currentPage, savedVacantes.length);
}
```

- Loads vacancies from localStorage and displays the current page.
- Updates pagination controls based on the total vacancy count.

Adding a Vacancy Row to the Table

```
function agregarFilaVacante(title, status) {
    const newRow = document.createElement('tr');
    newRow.innerHTML =
        `<td>${title}</td>
        <td><span>${status}</span></td>
        <td>
            <div class="actions">
                <button onclick="abrirModalEditar(this)">
                    <img src="https://img.icons8.com/ios/452/edit--v1.png" alt="Edit">
                </button>
                <button onclick="borrarVacante(this)">
                    <img src="https://img.icons8.com/ios/452/delete--v1.png" alt="Delete">
                </button>
                <button class="redirect-button" onclick="redirigirPagina()">
                    <img src="https://img.icons8.com/ios/452/external-link.png" alt="Redirect"> <!-- Ícono de enlace -->
                </button>
            </div>
        </td>`;
    document.getElementById("vacantesTableBody").appendChild(newRow);
}
```

- Creates a new row for each vacancy and adds it to the table.

Redirect Function

```
function redirigirPagina() {
    window.location.href = "pagina_destino.html";
}
```

- Redirects to another page (destination URL).

Displaying the Current Page of Vacancies

```
function mostrarPagina(page, vacantes) {
    const start = (page - 1) * itemsPerPage;
    const end = start + itemsPerPage;
    document.getElementById("vacantesTableBody").innerHTML = "";
    vacantes.slice(start, end).forEach(v => agregarFilaVacante(v.title, v.status));

    actualizarControlesPaginacion(vacantes);
    actualizarBotonesPrevNext(page, vacantes.length);
}
```

- Displaying the Current Page of Vacancies

Pagination Controls Update

```
function actualizarControlesPaginacion(vacantes) {
    const totalPages = Math.ceil(vacantes.length / itemsPerPage);
    const pagination = document.getElementById("paginationControls");

    const buttons = pagination.querySelectorAll(".page-number");
    buttons.forEach(button => button.remove());

    for (let i = 1; i <= totalPages; i++) {
        const pageButton = document.createElement("button");
        pageButton.innerText = i;
        pageButton.classList.add("page-number");
        if (i === currentPage) {
            pageButton.classList.add("active");
        }
        pageButton.onclick = () => {
            currentPage = i;
            mostrarPagina(currentPage, vacantes);
        };
        pagination.insertBefore(pageButton, document.getElementById("nextPageBtn"));
    }
}
```

- Updates pagination buttons according to the total pages.

Previous/Next Button Management

```
function actualizarBotonesPrevNext(page, totalItems) {
    const totalPages = Math.ceil(totalItems / itemsPerPage);
    const prevButton = document.getElementById("prevPageBtn");
    const nextButton = document.getElementById("nextPageBtn");

    prevButton.disabled = page === 1;
    nextButton.disabled = page === totalPages;

    prevButton.onclick = () => {
        if (currentPage > 1) {
            currentPage--;
            mostrarPagina(currentPage, JSON.parse(localStorage.getItem("vacantes")));
        }
    };

    nextButton.onclick = () => {
        if (currentPage < totalPages) {
            currentPage++;
            mostrarPagina(currentPage, JSON.parse(localStorage.getItem("vacantes")));
        }
    };
}
```

- Enables or disables the previous and next buttons as per page limits.
- Loads the previous or next page when respective buttons are clicked.

Save Vacancies to LocalStorage

```
function guardarVacantes() {
    const rows = document.querySelectorAll("#vacantesTableBody tr");
    const vacantes = Array.from(rows).map(row => ({
        title: row.cells[0].innerText,
        status: row.cells[1].querySelector("span").innerText
    }));
    localStorage.setItem("vacantes", JSON.stringify(vacantes));
    actualizarControlesPaginacion(vacantes);
}
```

- Saves vacancy details in localStorage and updates pagination after saving.

Add or Update Vacancy

```
function agregarVacante() {
    const title = document.getElementById("vacancyTitle").value;
    const status = document.getElementById("vacancyStatus").value;

    console.log("Título:", title);
    console.log("Estado:", status);

    if (isEditing && currentRow) {

        currentRow.cells[0].innerText = title;
        currentRow.cells[1].querySelector("span").innerText = status;

        const vacantes = JSON.parse(localStorage.getItem("vacantes")) || [];
        const index = Array.from(document.querySelectorAll("#vacantesTableBody tr")).indexOf(currentRow);
        if (index > -1) {
            vacantes[index].title = title;
            vacantes[index].status = status;
            localStorage.setItem("vacantes", JSON.stringify(vacantes));
        }

        salirModoEdicion();
    } else {

        const vacantes = JSON.parse(localStorage.getItem("vacantes")) || [];
        vacantes.push({ title, status });
        localStorage.setItem("vacantes", JSON.stringify(vacantes));

        currentPage = Math.ceil(vacantes.length / itemsPerPage);
        mostrarPagina(currentPage, vacantes);
        actualizarBotonesPrevNext(currentPage, vacantes.length);
    }

    cerrarModal();
}
```

- Adds or updates vacancy details.
- If editing, updates the selected row; otherwise, adds a new vacancy.

Open Modal in Edit Mode

```
function abrirModalEditar(button) {
    const row = button.closest("tr");
    const title = row.cells[0].innerText;
    const status = row.cells[1].querySelector("span").innerText;

    document.getElementById("vacancyTitle").value = title;
    document.getElementById("vacancyStatus").value = status;
    document.getElementById("modalTitle").innerText = "Editar Vacante";
    document.getElementById("saveButton").innerText = "Editar Vacante";

    document.getElementById("addVacancyModal").style.display = "flex";
    isEditing = true;
    currentRow = row;
}
```

- Fills modal fields with selected row data and switches to edit mode.
- Sets isEditing flag and updates the currentRow.

Opening Modal on Button Click

```
btn.onclick = function() {
    console.log("Modal abierto");
    isEditing = false;
    currentRow = null;
    form.reset();
    document.getElementById("modalTitle").innerText = "Agregar Nueva Vacante";
    document.getElementById("saveButton").innerText = "Guardar Vacante";
    modal.style.display = "flex";
};
```

- This code defines the behavior when the "Add Vacancy" button (btn) is clicked, prints "Modal abierto" to the console, which helps track the modal opening for debugging purposes.
- Resets Editing Mode:
- Sets isEditing to false, indicating that we are not in edit mode.
- Sets currentRow to null, clearing any row selection from a previous edit.
- Reset Form: Calls form.reset() to clear all fields, ensuring the form starts blank for a new vacancy entry.
- Update Modal Text: Sets:
- The title of the modal is "Agregar Nueva Vacante," signaling to the user that a new vacancy is being created.
- The "Save" button text to "Guardar Vacante," making it clear that this action will save a new entry.
- Sets modal.style.display to "flex", which makes the modal visible by using the CSS Flexbox display style.

Delete Vacancy

```
function borrarVacante(button) {

    const row = button.closest("tr");


    const vacantes = JSON.parse(localStorage.getItem("vacantes")) || [];
    const index = Array.from(document.querySelectorAll("#vacantesTableBody tr")).indexOf(row);


    if (index > -1) {
        vacantes.splice(index, 1);
        localStorage.setItem("vacantes", JSON.stringify(vacantes));
    }


    row.remove();


    cargarVacantes();
}
```

- Deletes a vacancy from localStorage and refreshes the list.

Modal and Form Control Logic

```
const modal = document.getElementById("addVacancyModal");
const btn = document.getElementById("addVacancyBtn");
const span = document.getElementsByClassName("close")[0];
const form = document.getElementById("vacancyForm");
```

- **modal:** Refers to the modal dialog element for adding or editing vacancies.
- **btn:** The button to open the modal and add a new vacancy.
- **span:** The "close" icon element within the modal, used to close the modal.
- **form:** Refers to the form element inside the modal, where the vacancy information is entered.

Opening the Modal for Adding a Vacancy

```
btn.onclick = function() {
    console.log("Modal abierto");
    isEditing = false;
    currentRow = null;
    form.reset();
    document.getElementById("modalTitle").innerText = "Agregar Nueva Vacante";
    document.getElementById("saveButton").innerText = "Guardar Vacante";
    modal.style.display = "flex";
};
```

- Logging: Logs "Modal abierto" to the console to confirm the modal is opening.
- Reset State:
- isEditing is set to false to indicate that the modal is in "Add" mode rather than "Edit" mode.
- currentRow is set to null to clear any selected row from a previous edit.
- Form Reset: form.reset() clears all form fields to start fresh.
- Modal and Button Text Updates:
- modalTitle: Changes the modal title to "Agregar Nueva Vacante" to indicate that a new vacancy is being added.
- saveButton: Changes the button text to "Guardar Vacante" to clarify the action.
- Display Modal: Sets modal.style.display to "flex" to make the modal visible.

Closing the Modal with the "Close" Icon

```
span.onclick = function() {
    modal.style.display = "none";
}
```

Hides the modal by setting modal.style.display to "none" when the close icon is clicked.

Closing the Modal by Clicking Outside of It

```
window.onclick = function(event) {
    if (event.target == modal) {
        modal.style.display = "none";
    }
}
```

Closes the modal if the click event target is the modal itself (i.e., when the user clicks outside the modal content area).

Handling the Form Submission

```
form.onsubmit = function(event) {
    event.preventDefault();

    const title = document.getElementById("vacancyTitle").value;
    const status = document.getElementById("vacancyStatus").value;

    console.log("Título:", title);
    console.log("Estado:", status);

    agregarVacante();
}
```

prevents the form from reloading the page on submission, title and status capture the values from the form's input fields logging logs the captured title and status to the console for debugging and calls agregarVacante() to add the new vacancy entry with the captured data..

Vacancy Management(CSS)

Universal Selector

```
* {
    margin: 0;
    padding: 0;
    font-family: "Inter", sans-serif;
    box-sizing: border-box;
}
```

Removes default margin and padding from all elements, sets the font to "Inter," and applies box-sizing: border-box to include padding and borders within the element's width and height

Body

```
body {
    background-color: #dbdbdb;
    font-family: Arial, sans-serif;
    color: #333;
}
```

Sets the background color to a light gray, the font to Arial, and the text color to dark gray, providing a basic page styling.

Container Class

```
.container {
    width: 100%;
    grid-template-columns: 1fr;
    margin: 0 auto;
    padding: 20px;
}
```

Creates a full-width container with a single-column layout and 20px padding..

Main Content

```
.main-content {
    padding: 20px;
    background-color: #fff;
    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);
    border-radius: 8px;
    margin-left: 150px;
    transition: margin-left 0.3s;
}
```

Styles the main content area with padding, white background, soft shadow, rounded corners, and a transition effect for the margin, ensuring a smooth layout.

Right Section

```
.right {
    background-color: #fdfdfd;
    display: flex;
    justify-content: center;
    align-items: center;
    color: white;
    font-size: 24px;
}
```

Centers content within .right, applies a white background, white text color, and a font size of 24px.

Left Section

```
.left {
    background-color: white;
    display: flex;
    justify-content: center;
    width: 100%;
    align-items: center;
    color: white;
    font-size: 24px;
}
```

Centers content within .left, sets background color to white, text color to white, and font size to 24px.

Text Styles for Titles and Names

```css
.user-details .title,
.menu .title {
    font-size: 10px;
    font-weight: 500;
    color: white;
    text-transform: uppercase;
    margin-bottom: 10px;
}

.user-details .name {
    font-size: 14px;
    font-weight: 500;
    color: white;
}
```

Sets styles for .title and .name in .user-details and .menu, making text small, uppercase, and white for .title, and slightly larger and bold for .name.

Menu

```css
.menu ul li a {
    display: flex;
    align-items: center;
    gap: 10px;
    font-size: 14px;
    font-weight: 500;
    color: white;
    text-decoration: none;
    padding: 12px 8px;
    border-radius: 8px;
    transition: all 0.3s;
}

.menu ul li > a:hover,
.menu ul li.active > a {
    color: #000;
    background-color: #f6f6f6;
}
```

Styles each menu link as a flex container with padding, white text color, rounded corners, and hover effects for a better appearance.

Table

```css
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
    background-color: #fff;
    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.05);
    border-radius: 5px;
    overflow: hidden;
}

table, th, td {
    border: 1px solid #ccc;
}

th {
    background-color: #333;
    color: white;
    font-weight: bold;
    padding: 12px;
}
td {
    padding: 12px;
    color: #555;
}
```

Styles the table for uniform appearance with a white background, rounded corners, padding, and borders. The header row (th) has a dark background with white text.

Actions

```css
.actions {
    display: flex;
    justify-content: center;
    gap: 10px;
}

.actions button {
    border: none;
    background: none;
    cursor: pointer;
    transition: transform 0.2s;
}
.actions button:hover {
    transform: scale(1.1);
}
.actions button img {
    width: 20px;
    height: 20px;
}
```

Centers action buttons and applies a scale effect on hover for a dynamic look.

Add Button

```css
.btn-add {
    display: inline-block;
    padding: 10px 20px;
    background-color: #333;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-weight: bold;
    transition: background-color 0.3s;
}
.btn-add:hover {
    background-color: #555;
}
```

Styles the add button with a dark background, white text, bold font, and smooth hover transition.

Pagination

```css
.pagination {
    display: flex;
    justify-content: center;
    align-items: center;
    gap: 10px;
    margin-top: 20px;
}

.pagination-btn {
    background-color: #000000;
    color: white;
    border: none;
    padding: 8px 16px;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
}

.pagination-btn:hover {
    background-color: #585757;
}

.pagination-btn:disabled {
    background-color: #d3d3d3;
    cursor: not-allowed;
}

.page-number {
    background-color: white;
    border: 1px solid #222222;
    padding: 8px 16px;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
}

.page-number.active {
    background-color: #000000;
    color: white;
}

.page-number:hover {
    background-color: #585757;
    color: white;
}
```

- Provides styling and hover effects for pagination buttons, as well as styles for the active page number.

Modal

```
.modal {
    display: none;
    position: fixed;
    z-index: 1;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.5);
    justify-content: center;
    align-items: center;
}

.modal-content {
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    width: 400px;
    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.2);
}
```

Styles the modal to cover the entire screen with a semi-transparent background. Centers the .modal-content box with padding, a white background, and a shadow.

Close Button for Modal

```
.close {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
}

.close:hover,
.close:focus {
    color: black;
    text-decoration: none;
    cursor: pointer;
}
```

Gives a style for the close button in the modal, with hover effects for easier user interaction.

Form Elements in Modal

```css
.input-form {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 1rem;
    margin-top: 5px;
    background-color: #f7f7f7;
    color: #333;
}
```

Provides stable spacing and padding for form groups, and styles the input fields for readability and usability.

Media Query for Smaller Screens

```css
@media (max-width: 768px) {
    .sidebarfool {
        width: 100px;
    }
    .main-content {
        margin-left: 100px;
    }
}
```

Adjusts sidebar and main content widths for better layout on screens 768px wide or smaller.

Redirect Button

```css
.redirect-button img {
    width: 20px;
    height: 20px;
}
```

Gives a width and height to the image for the redirect button, allowing it to be more accommodated

5) Testing and Review

Perform rigorous tests and reviews of the website to identify and fix any issues before launch.

6) Optimization

Optimize the website for performance, accessibility, and SEO. This involves improving load times, ensuring mobile responsiveness, and enhancing overall user experience.

7) Launch

The final step involves deploying the website to a production environment and making it publicly available to users. Ensure that everything works as intended in the live environment.