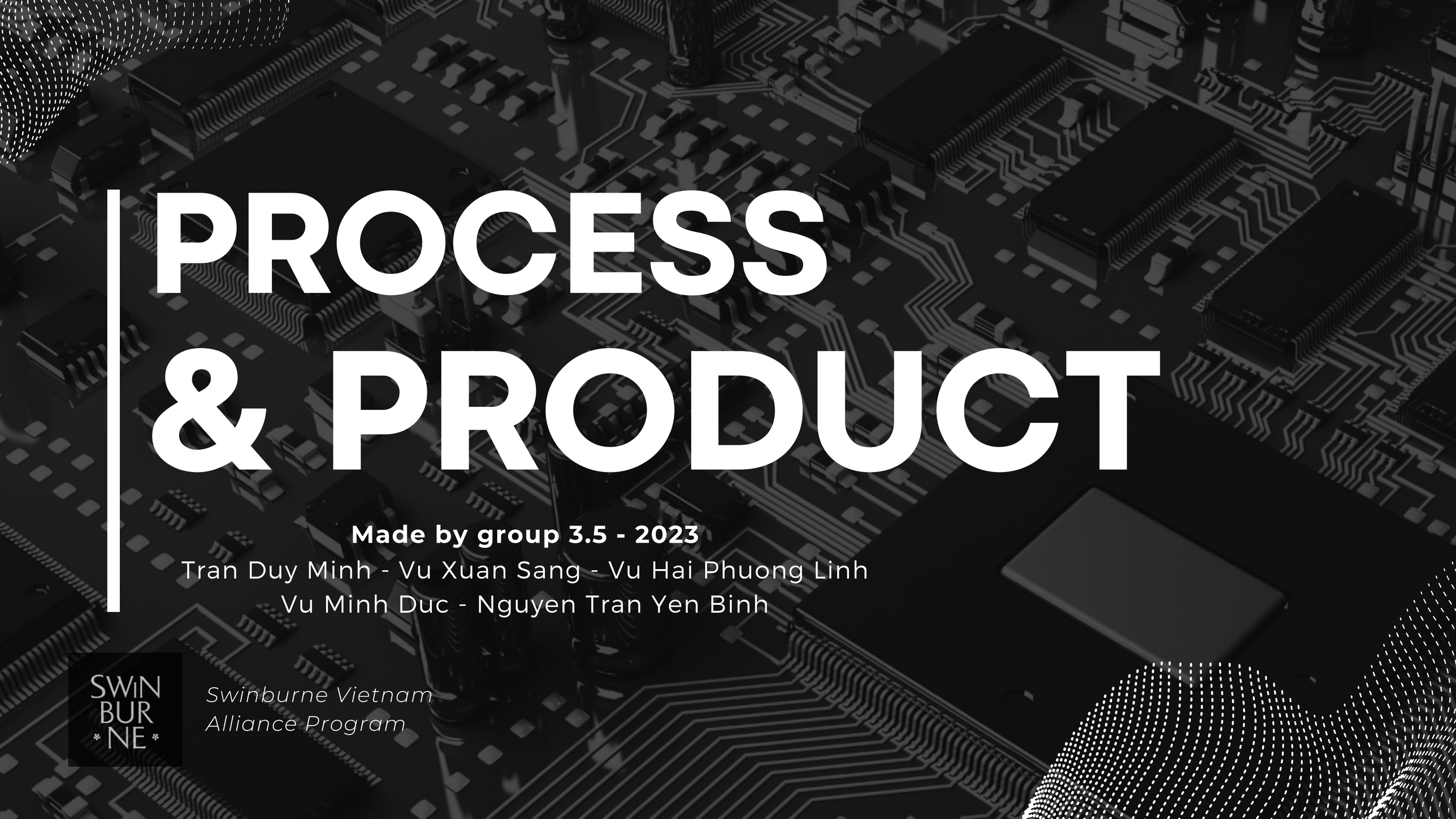


PROCESS & PRODUCT

Made by group 3.5 - 2023

Tran Duy Minh - Vu Xuan Sang - Vu Hai Phuong Linh

Vu Minh Duc - Nguyen Tran Yen Binh

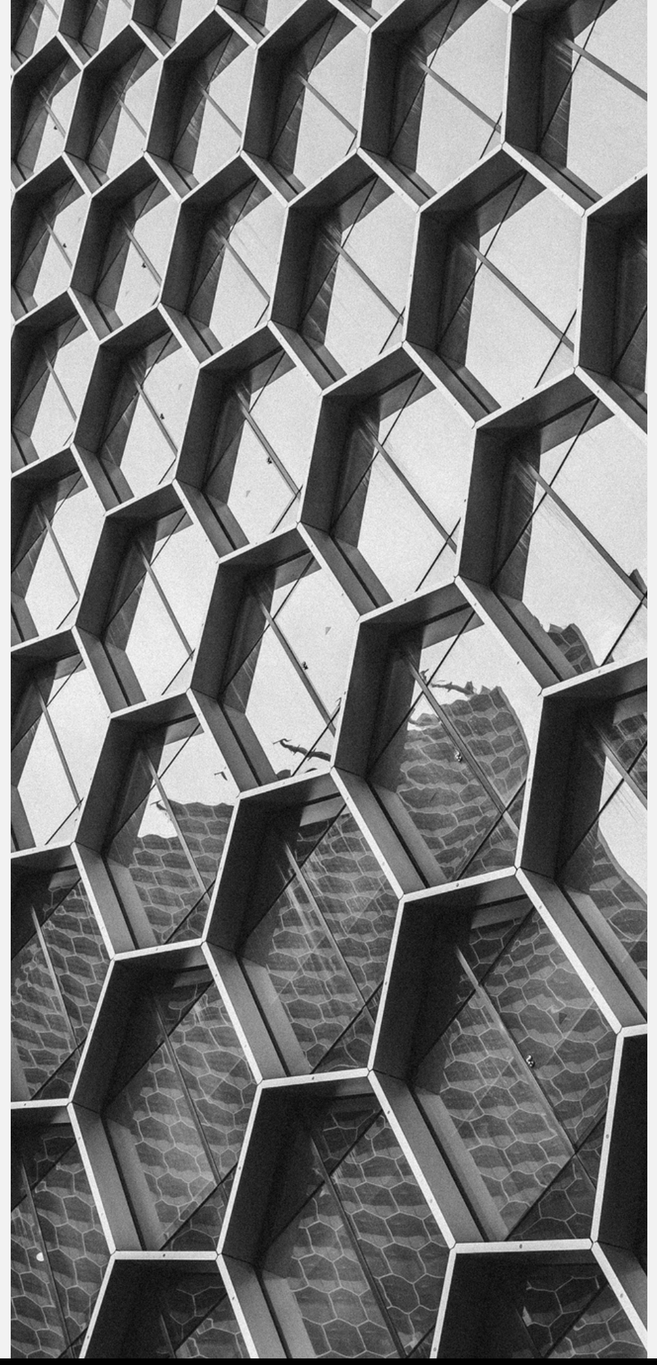
SWINBURNE

Swinburne Vietnam
Alliance Program

CONTENT

INTRODUCTION

- The competitive job landscape demands a streamlined and efficient platform for connecting job seekers and employers.
- The conventional approaches to hiring and job hunting are becoming outdated, which is calling for a contemporary answer.
- The appearance of “Job Connect” is the solution to replacing traditional methods of finding and hunting jobs.
- Project “Job Connect” aims to streamline the finding and hiring jobs process for job hunters and job seekers.



PROJECT BACKGROUND

- Finding a job is part of the job market, although it may be difficult and include a variety of techniques and kinds of propaganda. Employers will use the media to disseminate employment information, while job searchers will use it to establish connections with employers.
- The exchange process between the two sides will take place quite long, from basic communication to discussing work, and then there will be work-related questions (sometimes there are silly questions), and then comes the recruitment process.
- Two primary stakeholders of the project are employers and job seekers.
- The goal of this project is to provide an easily navigable database and website for users and revolutionize the way individuals and businesses navigate the job market.

SUMMARY OF REQUIREMENTS

ER diagrams

Data

Jira Log

Performance

Design and Use Cases

Major-specific contributions

MISSION STATEMENT AND OBJECTIVES

MISSION STATEMENT: Building a web-based Recruitment System

Providing a user-friendly platform for searching, applying, and communicating with potential employers, the website connects job searchers with employment prospects.

Obj.1: Efficient Hiring

saves time and effort by automating tasks like interview scheduling and resume screening

Obj.2: Enhancing Quality Hires

Recruit and place competent individuals who align with the organization's culture and job requirements

Obj.3: Optimizing Collaboration

Enhance cooperation and communication among recruiters, hiring managers, and stakeholders

Obj.4: Building a Strong Employer Brand

providing a positive applicant experience and maintaining professionalism throughout the recruiting process

Obj.5: Data-Driven Hiring

Use valuable data and analysis to evaluate hiring success, identify obstacles, and make informed improvements

TEAM PROCESS

What the team has done throughout the semester

Main records are doccumented on Jira and Confluence

Minh

- Collaboration: Team leader, assign work, slides, disscuss project requirements and meet up with team, Jira
- Roles: Create and implement database, Confluence content
- Contribution: ER diagram, Product requirements

Linh

- Collaboration: slides, disscuss project requirements and meet up with team, Team meeting notes
- Roles: Data creation, Confluence Content
- Contribution: Mockaroo database Persona, Risk Assessment

Sang

- Collaboration: slides, disscuss project requirements and meet up with team, set up team meeting
- Roles: Program website, Confluence Content
- Contribution: Website, website introduction

Binh

- Collaboration: slides, disscuss project requirements and meet up with team
- Roles: Confluence content
- Contribution: Team homepage, Project plan, Team Health Monitor, Member Underperformance

Duc

- Collaboration: Decide slides layout, slides, disscuss project requirements and meet up with team, Jira
- Roles: Team video, Confluence content
- Contribution: Team homepage, Risk management, Team Reflection

01 02 03 04 05

MAJOR-SOFTDEV

This script includes functions to add, validate and insert data from customers to the database. Here is overview of what each function does:

•Form Submission Handling: The PHP script checks if the form is submitted using $_SERVER["REQUEST_METHOD"] == "POST".

•Data Retrieval: It retrieves the form data using $_POST.

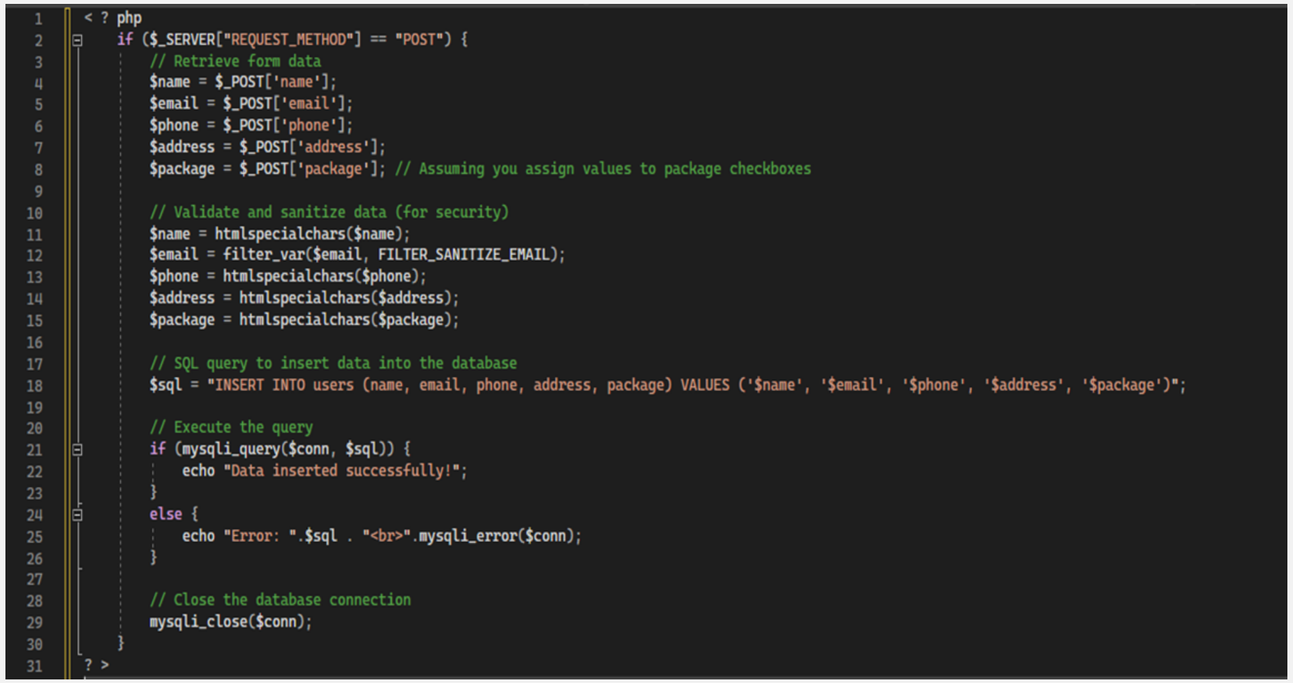
•Data Validation: For security purposes, it sanitizes and validates the data received from the form using functions like htmlspecialchars and filter_var.

•Database Insertion: It constructs an SQL query (INSERT INTO) to insert the form data into the database.

•Executing Query: The query is executed using mysqli_query.

•Handling Errors: It checks for any errors during the database insertion and displays an error message if necessary.

•Database Connection Closure: Finally, it closes the database connection.



```php
< ? php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        // Retrieve form data
        $name = $_POST['name'];
        $email = $_POST['email'];
        $phone = $_POST['phone'];
        $address = $_POST['address'];
        $package = $_POST['package']; // Assuming you assign values to package checkboxes

        // Validate and sanitize data (for security)
        $name = htmlspecialchars($name);
        $email = filter_var($email, FILTER_SANITIZE_EMAIL);
        $phone = htmlspecialchars($phone);
        $address = htmlspecialchars($address);
        $package = htmlspecialchars($package);

        // SQL query to insert data into the database
        $sql = "INSERT INTO users (name, email, phone, address, package) VALUES ('$name', '$email', '$phone', '$address', '$package')";

        // Execute the query
        if (mysqli_query($conn, $sql)) {
            echo "Data inserted successfully!";
        }
        else {
            echo "Error: ".$sql . "<br>".mysqli_error($conn);
        }

        // Close the database connection
        mysqli_close($conn);
    }
? >
```

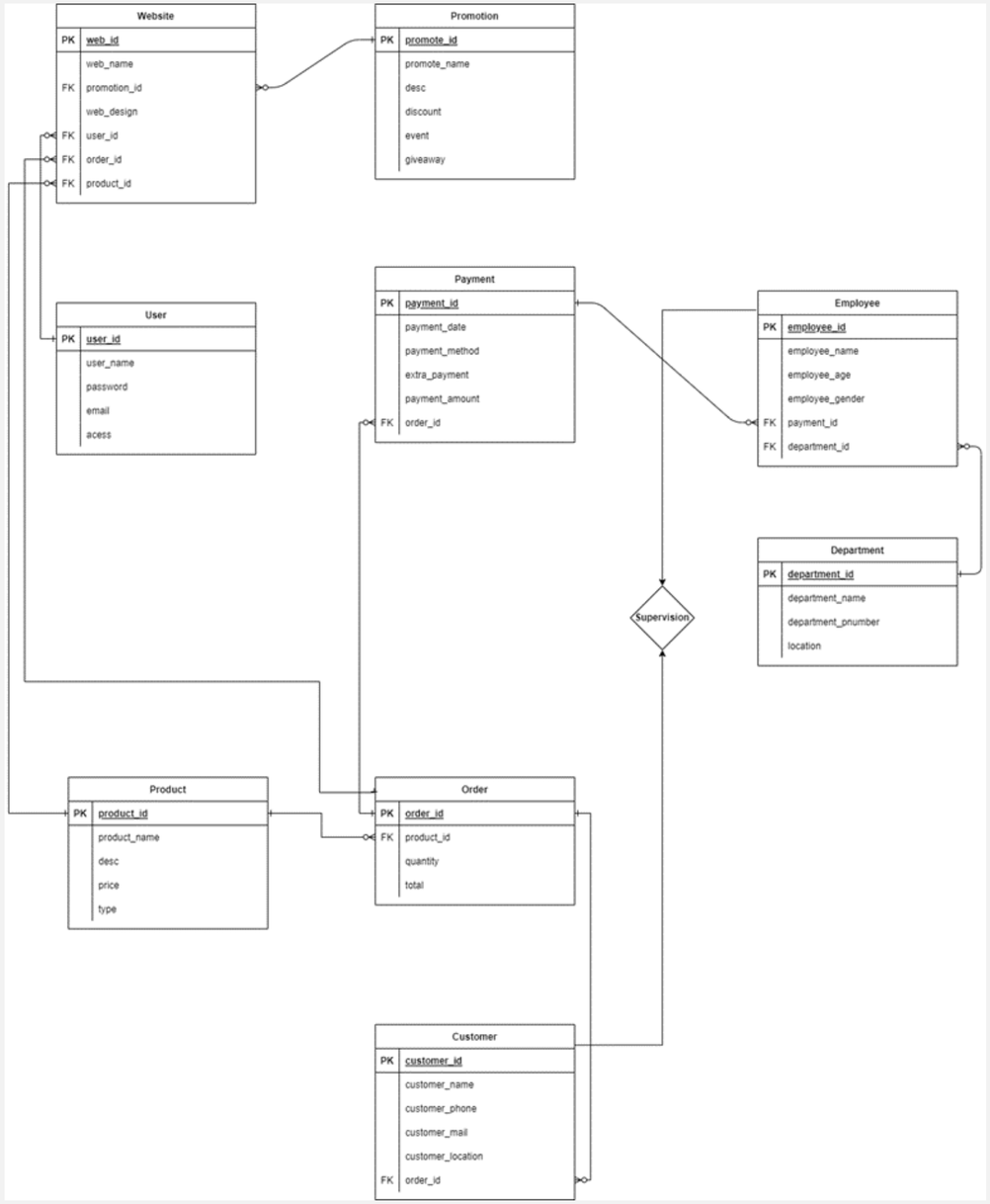
MAJOR-SOFTDEV

This script includes functions allows users to apply directly to the job postings through the website by uploading their resumes and cover letters:

•The showApplyForm() function displays the previously hidden apply form when called. This can be triggered, for example, when the user clicks on an "Apply Now" button associated with a job.

•The form submission is handled by listening for the submit event on the form. When the form is submitted, the function collects form data (name, email, resume file, cover letter), creates a FormData object, and sends it to a server endpoint using the fetch API. This is done asynchronously to prevent the page from refreshing.

```
function showApplyForm() {
    var applyForm = document.querySelector('.apply-form');
    applyForm.style.display = 'block';
}

// Function to handle form submission
document.getElementById('jobApplicationForm').addEventListener('submit', function(e) {
    e.preventDefault(); // Prevent default form submission

    // Fetch form values
    var fullName = document.getElementById('fullName').value;
    var email = document.getElementById('email').value;
    var resumeFile = document.getElementById('resume').files[0]; // Uploaded resume file
    var coverLetter = document.getElementById('coverLetter').value;

    // Handle the form submission
    var formData = new FormData();
    formData.append('fullName', fullName);
    formData.append('email', email);
    formData.append('resume', resumeFile, resumeFile.name);
    formData.append('coverLetter', coverLetter);

    // Example using fetch:
    fetch('https://jobfinder.com/posts', {
        method: 'POST',
        body : formData
        })
        .then(response = > {
        // Handle response from the server
        console.log('Application submitted:', response);

    })
        .catch (error = > {
        // Handle errors

    });

    // Clear form fields after submission (optional)
    document.getElementById('jobApplicationForm').reset();
});
```



ER DIAGRAM

- This JOB diagram is build for managing the Job finding website.
- It represent tables such as:
  - “User” for login and signup.
  - “Customer” and “Employee” to manage the order payment.
  - “Department” of employees for our service.
  - “Product”, “Order”, “Payment”.
- This is an efficient database which included all needed data and more.
- Suitable for managing and analyzing the data for potential update/improve.

PHYSICAL DATABASE

Create general columns

```sql
--

CREATE TABLE `customer` (
  `customer_id` int(50) NOT NULL ,
  `customer_name` varchar(255) NOT NULL,
  `customer_phone` int(50) NOT NULL,
  `customer_mail` varchar(255) NOT NULL,
  `customer_location` varchar(255) NOT NULL,
  `order_id` int(50) NOT NULL,
  `customer_age` int(50) NOT NULL
)

--
```

```sql
CREATE TABLE `employee` (
  `employee_id` int(50) NOT NULL,
  `employee_name` varchar(255) NOT NULL,
  `employee_age` int(50) NOT NULL,
  `employee_gender` varchar(255) NOT NULL,
  `payment_id` int(50) NOT NULL,
  `department_id` int(50) NOT NULL
) ;

-- --------------------------------------------------------


--
-- Cấu trúc bảng cho bảng `oder`
--

CREATE TABLE `oder` (
  `order_id` int(50) NOT NULL,
  `price` int(50) NOT NULL,
  `quantity` int(50) NOT NULL,
  `total` int(50) NOT NULL,
  `product_id` int(50) NOT NULL,
  `status` varchar(255) NOT NULL
) ;
```

```sql
ALTER TABLE `customer`
  MODIFY `customer_id` int(50) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=52;

--
-- AUTO_INCREMENT cho bảng `department`
--
ALTER TABLE `department`
  MODIFY `department_id` int(50) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=52;

--
-- AUTO_INCREMENT cho bảng `employee`
--
ALTER TABLE `employee`
  MODIFY `employee_id` int(50) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
```

Add auto increment for needed columns

Create general columns

PHYSICAL DATABASE

```
--
ALTER TABLE `customer`
  ADD PRIMARY KEY (`customer_id`),
  ADD KEY `order_of_customer` (`order_id`);

--
-- Chỉ mục cho bảng `department`
--
ALTER TABLE `department`
  ADD PRIMARY KEY (`department_id`);

--
-- Chỉ mục cho bảng `employee`
--
ALTER TABLE `employee`
  ADD PRIMARY KEY (`employee_id`),
  ADD KEY `employee_handle_payment` (`payment_id`),
  ADD KEY `employee_department` (`department_id`);
```

Add primary key and index keys

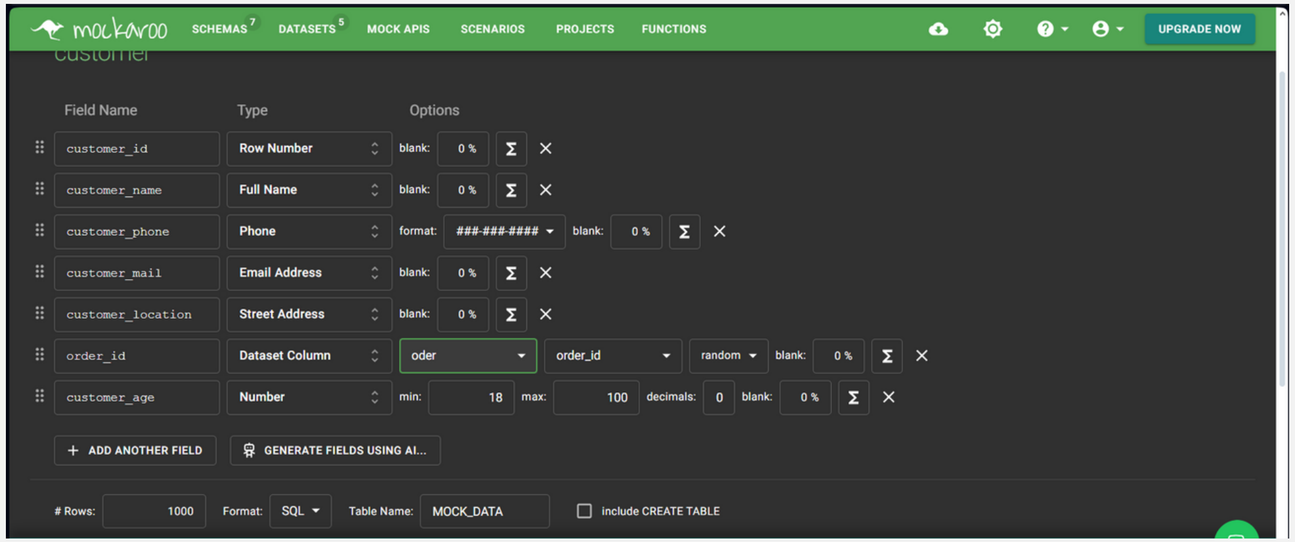
```
ALTER TABLE `customer`
  ADD CONSTRAINT `order_of_customer` FOREIGN KEY (`order_id`) REFERENCES `oder` (`order_id`);

--
-- Các ràng buộc cho bảng `employee`
--
ALTER TABLE `employee`
  ADD CONSTRAINT `employee_department` FOREIGN KEY (`department_id`) REFERENCES `department` (`department_id`),
  ADD CONSTRAINT `employee_handle_payment` FOREIGN KEY (`payment_id`) REFERENCES `payment` (`payment_id`);

--
-- Các ràng buộc cho bảng `oder`
--
ALTER TABLE `oder`
  ADD CONSTRAINT `order_od_prod` FOREIGN KEY (`product_id`) REFERENCES `product` (`product_id`);
```

Create constraint foreign keys

```
CREATE UNIQUE INDEX customer_id
ON customer;
```

Create unique index

DATA CREATION



Use Mockaroo to create dummy data for the database

Result

Add to the database

```
INSERT INTO `customer` (`customer_id`, `customer_name`, `customer_phone`, `customer_mail`, `customer_location`
(1, 'Cale Woolway', 493, 'cwoolway0@nifty.com', '27056 Sugar Trail', 1, 46),
(2, 'Chucho Bleasdale', 336, 'cbleasdale1@acquirethisname.com', '97 Lake View Point', 2, 100),
(3, 'Madonna Churcher', 491, 'mchurcher2@ft.com', '29 Holy Cross Road', 3, 24),
(4, 'Tadeas Purcer', 112, 'tpurcer3@sitemeter.com', '87575 Beilfuss Park', 2147483647, 44),
(9, 'Lucine Mathouse', 509, 'lmathouse8@princeton.edu', '09085 Roth Road', 2147483647, 73),
(10, 'Leicester Camis', 200, 'lcamis9@amazon.co.uk', '6 Hazelcrest Pass', 2147483647, 62),
(12, 'Raimundo Handscomb', 712, 'rhandscombb@gnu.org', '86 Dakota Park', 2147483647, 80),
(22, 'Gualterio Forbes', 265, 'gforbesl@sbwire.com', '371 Shasta Crossing', 2147483647, 56),
(23, 'Irita Cowperthwaite', 770, 'icowperthwaitem@aboutads.info', '12073 Dixon Junction', 2147483647, 91),
(25, 'Tiertza Godrich', 571, 'tgodricho@fema.gov', '147 Service Plaza', 2147483647, 100),
(26, 'Shermy Hargey', 240, 'shargeyp@de.vu', '0075 Raven Terrace', 2147483647, 23),
(27, 'Sella Gladwell', 460, 'sgladwellq@wikispaces.com', '89838 Bunting Avenue', 2147483647, 30),
(28, 'Karia Shower', 953, 'kshowerr@seattletimes.com', '3548 Roxbury Court', 2147483647, 100),
(29, 'Bradan MacAllen', 522, 'bmacallens@desdev.cn', '56 Forest Dale Point', 2147483647, 29),
(30, 'Augy Diment', 174, 'adimentt@oracle.com', '0692 Autumn Leaf Alley', 2147483647, 18),
(40, 'Raul Judgkins', 183, 'rjudgkins13@twitpic.com', '2583 Prairieview Alley', 2147483647, 63),
(41, 'Gwenore Roach', 366, 'groach14@archive.org', '610 Sheridan Alley', 2147483647, 61),
(46, 'Care Anthonies', 718, 'canthonies19@t-online.de', '9 School Court', 2147483647, 66),
(47, 'Cornie Gluyas', 487, 'cgluyas1a@ucsd.edu', '185 Golden Leaf Street', 2147483647, 89);
```

USE CASES

```sql
SELECT d.department_name, d.department_pnumber, d.location, COUNT(ep.payment_id) AS num_payments
FROM department d
LEFT JOIN employee e ON d.department_id = e.department_id
LEFT JOIN payment ep ON e.payment_id = ep.payment_id
GROUP BY d.department_name, d.department_pnumber, d.location
ORDER BY num_payments ASC
```

SORT DEPARTMENT WITH NUMBER OF PAYMENTS IT HAVE ASCENDING

```sql
SELECT d.department_name, d.department_pnumber, d.location, COUNT(ep.payment_id) AS num_payments FR
d.department_pnumber, d.location ORDER BY num_payments ASC;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

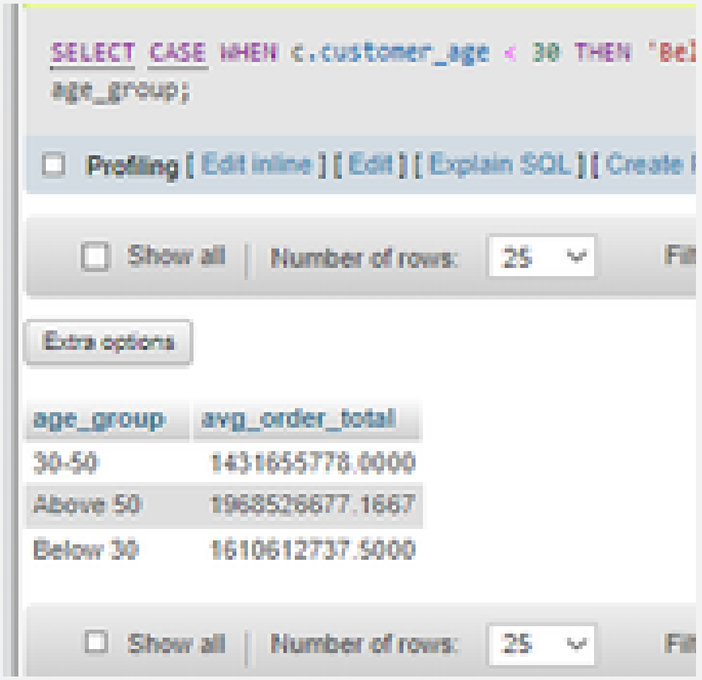
Extra options

department_name	department_pnumber	location	num_payments
GS CND	2353121	Canada	0
Lyons	216	Rongcheng	0
Maple Wood	867	Chengbei	0
GS FD	3314123	USA	0
Bayside	550	Dobříš	0
East	907	Ash Shafā	0
Killdeer	694	Hangchuan	0
Stone Corner	124	Gudang	0
7th	126	Gongjiahe	0
Almo	720	Hanyuan	0
Farragut	649	Cilebang Satu	0
Corben	456	Manjing	0
Merchant	650	Artemivs'k	0
Pond	953	Zhutang	0
Dovetail	652	Dalan	0
Independence	740	Anyar	0
Boyd	794	Gaborone	0
Donald	284	Hengshan	0
Crowley	795	Mao	0
Chive	687	Pedome	0
Knutson	218	Cap-Santé	0
Orin	318	Kodamachō-kodamaminami	0
Cardinal	294	Kuala Lumpur	0
Becker	290	Daoxu	1
Everett	158	Hikone	1

Show all | Number of rows: 25 | Filter rows: Search this table

USE CASES

```sql
SELECT CASE
    WHEN c.customer_age < 30 THEN 'Below 30'
    WHEN c.customer_age BETWEEN 30 AND 50 THEN '30-50'
    ELSE 'Above 50'
    END AS age_group,
    AVG(o.total) AS avg_order_total
FROM customer c
JOIN (
    SELECT order_id, total
    FROM oder
) o ON c.order_id = o.order_id
GROUP BY age_group;
```



AGE GROUP OF CUSTOMER THAT PURCHASE OUR BUNDLE

USE CASES

```
SELECT l.payment_id, l.payment_date, l.payment_method, o.order_id, o.quantity, o.total, p.product_id, p.product_name, p.price, l.payment_amount
FROM payment l
JOIN oder o ON l.order_id = o.order_id
JOIN product p ON o.product_id = p.product_id;
```

payment_id	payment_date	payment_method	order_id	quantity	total	product_id	product_name	price	payment_amount
1	2023-11-08	Visa Card	2	7	21	2	Pro	9	9
2	2023-11-09	Cash	3	6	54	3	Vip	20	9
3	2023-10-18	MasterCard	1	6	54	3	Vip	20	40

CHECKING ORDER AND PAYMENT STATUS

SUMMARY

- Our project offers job listing services, facilitating an extensive repository of employment opportunities.
- Our database incoporates systematic organization, storage, and retrieval of products and management data
- The adopted methods streamlines the recording process and improve productivity
- Users and administrators are granted optimized experience through personalized features and high security
- Our primary objective is to exert a positive impact within the job-searching space, inspiring others to implement technological solutions that effectively combat unemployment
- Collaborative teamwork fosters experiential learning, instilling the values of effective time management, structured scheduling, and iterative analysis through consistent contributions and critical assessments.