iFork Website

XR GROUP – Challenge

Yll Berisha

INTRODUCTION

The iFork website is a project I developed as part of a challenge from XR Group. The goal of this project was to showcase my skills in web development, including front-end design and back-end functionality.

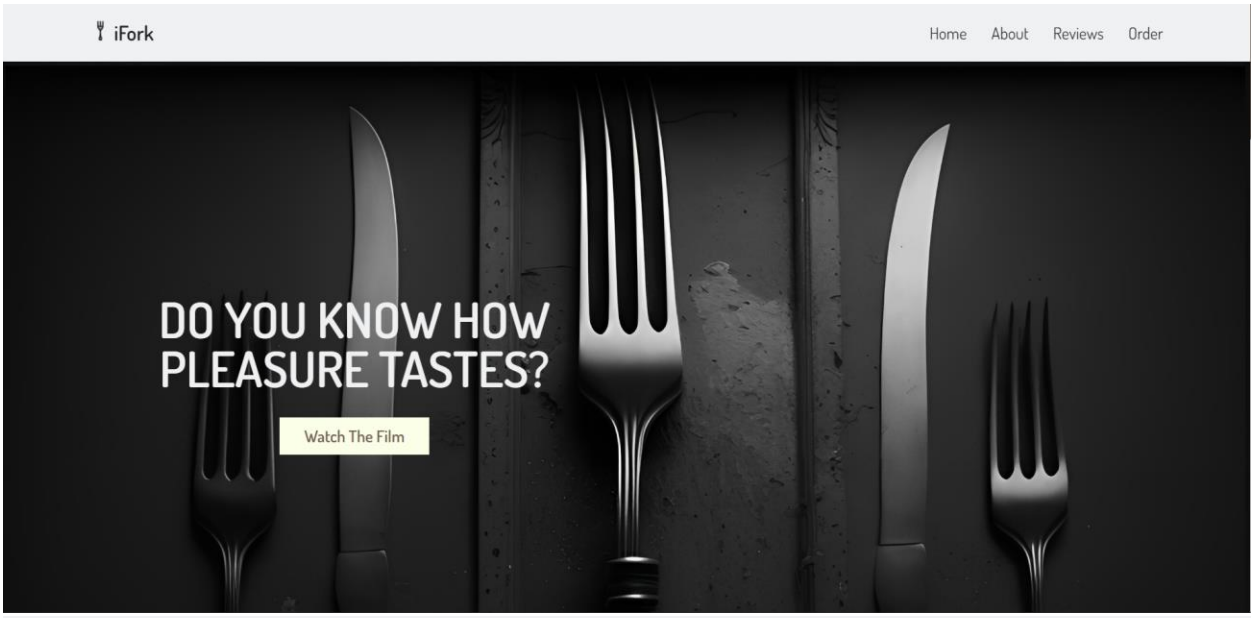
The website is designed to sell a fork, with a clean and modern design that focuses on the product itself. The homepage features a large image of the fork, along with a description and price. Users can easily navigate to the product page, where they can view more detailed information about the fork, including its features and specifications.

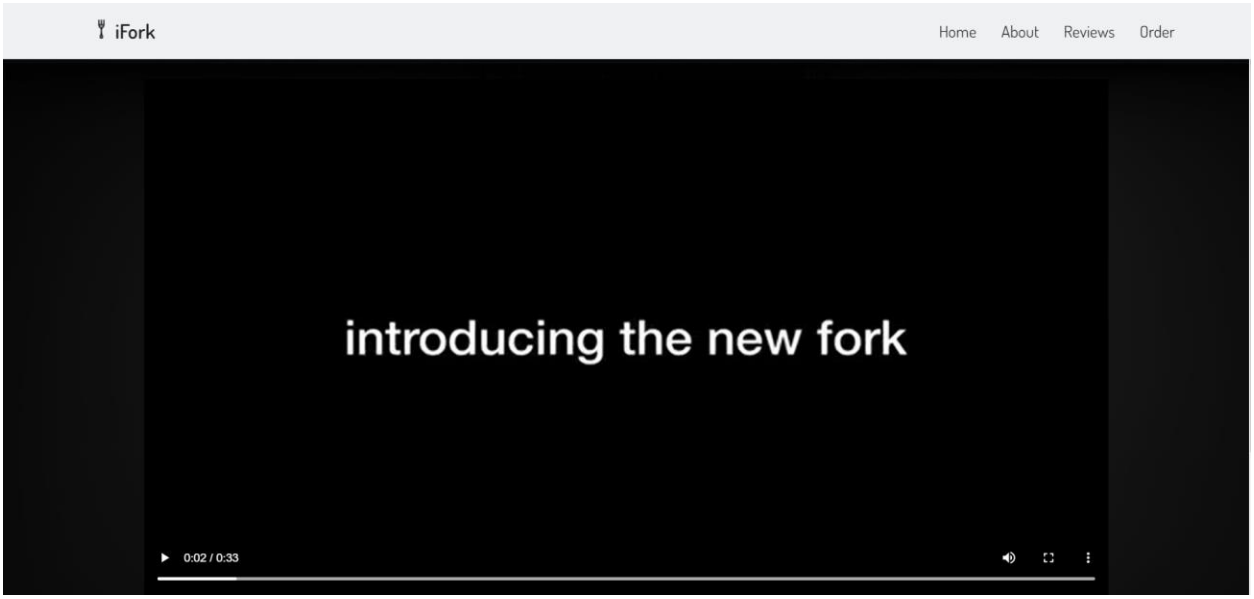
I want to mention that before working on this project, I had not worked with PHP before. However, it was a great opportunity for me to learn and improve my skills in web development. This project allowed me to challenge myself and step out of my comfort zone and learn something new. I am excited to continue learning and exploring new technologies in the future.

Overall, the iFork website is a great example of my skills in web development. It showcases my ability to create a modern and functional e-commerce website using a range of technologies, including HTML, CSS, JavaScript, PHP, and MySQL.

In order to give you a better idea of the website and its features, I have included some screenshots below. These screenshots will provide you with a visual representation of the website's design and functionality, and help you better understand how it works.

Home Page


iFork
Home
About
Reviews
Order
DO YOU KNOW HOW
PLEASURE TASTES?
Watch The Film


iFork
Home
About
Reviews
Order
introducing the new fork
0:02 / 0:33

iFork

Home About Reviews Order

Reviews

I'm Really Impressed With The Steel Fork I Bought From This Company. It's Made From High-Quality Stainless Steel, Which Makes It Durable And Easy To Clean. The Sleek Design Looks Great On My Table And The Comfortable Grip Is Perfect For Every Meal. Plus, It's Eco-Friendly And Reusable. Overall, A Great Purchase That I Highly Recommend!

Omer Kuleta

The Ifrok Is A Total Game-Changer, Folks. I Never Thought I Could Get So Excited About A Fork, But Here We Are. It's Like The James Bond Of Utensils - Sleek, Stylish, And Oh-So-Effective. I Feel Like I Could Take On Any Dish With This Thing, From A Fancy Steak To A Simple Bowl Of Spaghetti. And The Best Part? It's Basically Indestructible. I've Dropped It, Smacked It, And Even Used It As A Back Scratcher (Don't Judge Me). And It's Still Going Strong. So If You Want To Feel Like A Total Badass At Your Next Meal, Get The Ifrok. It'll Make Your Other Utensils Look Like A Bunch Of Wimps..

Nart Raci

The Ifrok Is The Ultimate Lazy Person's Dream Come True. I Mean, It Basically Washes Itself In The Washing Machine. It's Like Having A Magical Fairy Do Your Dishes For You. And Don't Even Get Me Started On The Way It Makes Me Feel - Like A Culinary Mastermind With A Futuristic Utensil. So If You Want To Impress Your Friends And Never Do Dishes Again, Get The Ifrok. It's Worth Every Penny.

Hana Hoxha

iFork

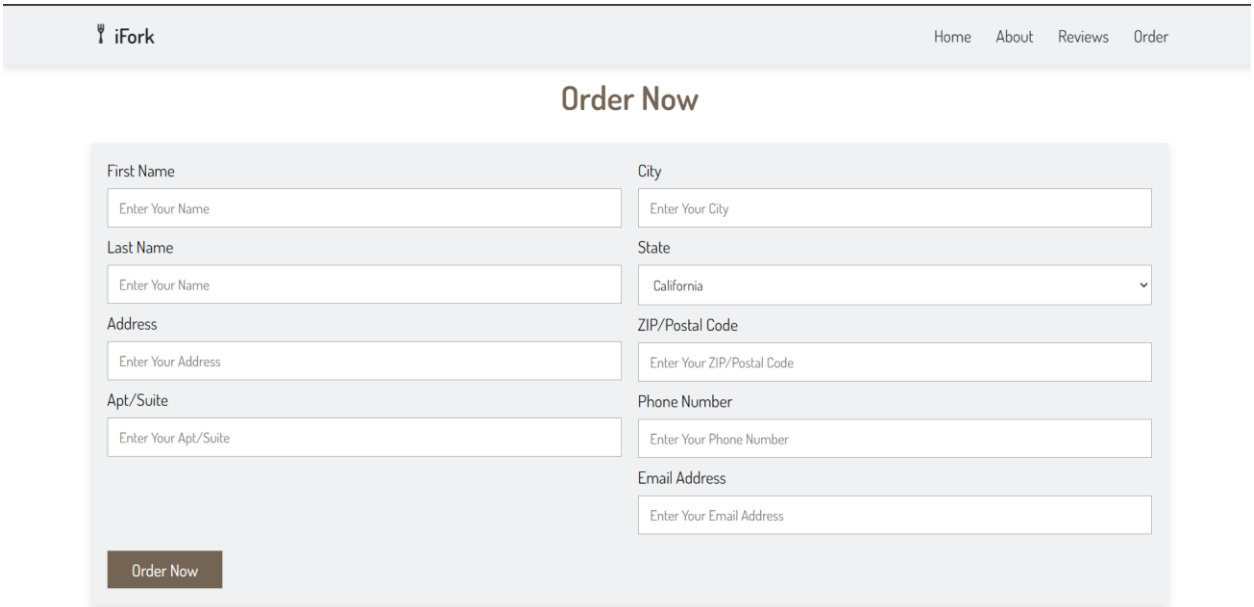
Home About Reviews Order

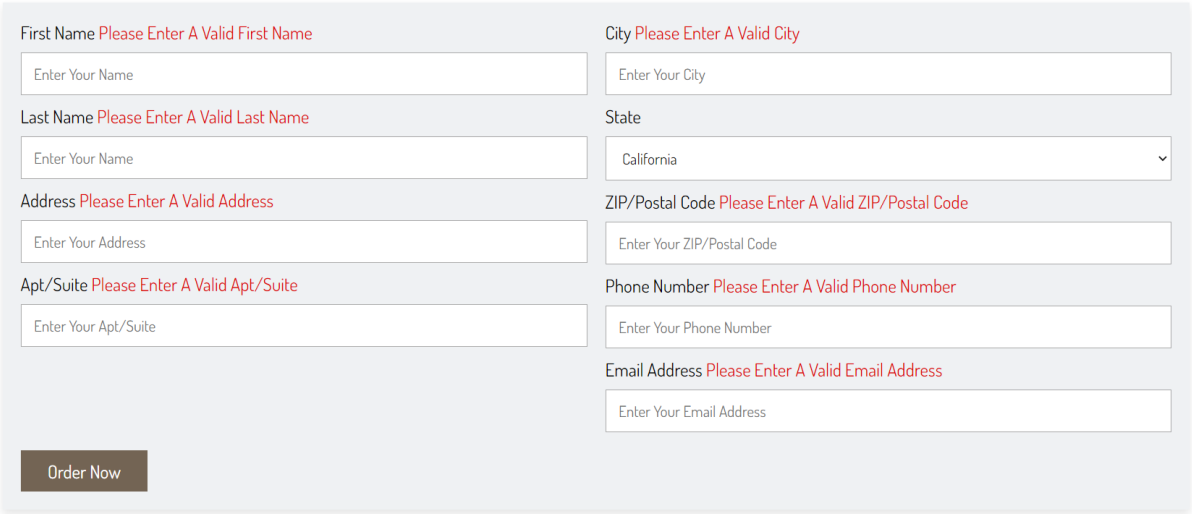
About Us

Introducing The Ultimate Fork: Perfect For Every Meal

Crafted With The Finest Stainless Steel, The IFork Will Bring Joy To Every Meal. With Its Sleek Design And Comfortable Grip, It Will Make Your Dining Experience A Wonder To Behold. But The IFork Is More Than Just An Eating Tool. It Represents A Sense Of Unity And Balance In Our Daily Lives. It Reminds Us That Even The Smallest Things Can Bring Us Together And Create Moments Of Happiness.

So Don't Just Invest In A Fork, Invest In A Mindset. Let The IFork Guide You On Your Culinary Adventures, And Watch As Your Meals Become Not Just A Source Of Sustenance, But A Source Of Wonder And Inspiration. Join Us In Our Quest For A World Filled With Delicious Food And Meaningful Connections, One IForkful At A Time!





The input fields on the website are validated both in the front-end and the back-end to ensure that only valid data is accepted. In the front-end, the validation is done using JavaScript, while in the back-end it is done using PHP. Both front-end and back-end validation use regular expressions (regex) to ensure that the data is in the correct format. This helps to prevent errors and ensure that the data entered by users is accurate.

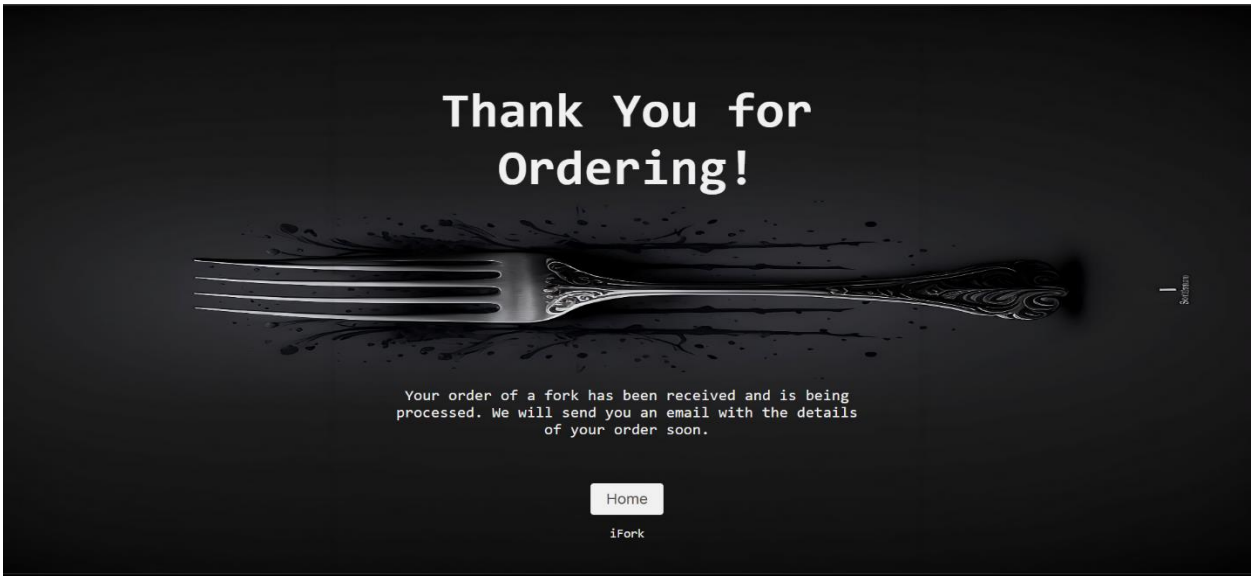
Check Out Page

The checkout page contains a form for users to enter their card information, including the card number, expiration date, and security code. Similar to the registration and login forms, the fields in the checkout form are validated both at the front-end using JavaScript and at the back-end using PHP to ensure that the information entered is correct and in the correct format.

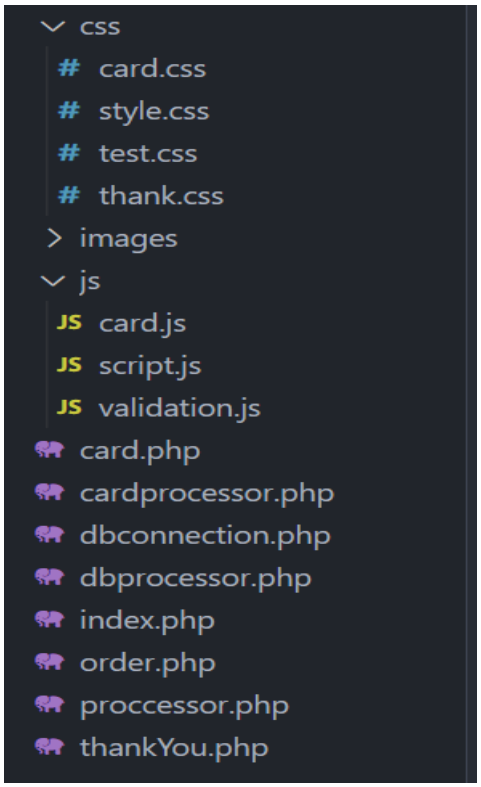
In addition, the website is designed to be responsive and optimized for different screen sizes. The checkout page includes a form for entering credit card information, and the website uses JavaScript to validate the fields and provide feedback to the user in real-time. Furthermore, the color of the card displayed on the checkout page changes dynamically based on the card type entered by the user, such as Visa, Mastercard, or American Express. This feature provides a visual cue to the user and enhances the user experience.

Final Page

After a successful order, the website will redirect the buyer to a page that thanks them for their purchase.



File Structure



- **index.php**: Main Home page
- **poccessor.php**: Validates the inputs and moves to the checkout page
- **card.php**: Checkout page where card information is taken
- **cardprocessor.php**: Validates the inputs from the checkout page
- **dbconnection.php**: Contains a class for database connection
- **order.php**: Represents the database table
- **dbprocessor.php**: Inserts data into the database
- **thankYou.php**: Confirmation page displayed after successful order

Database Connection

```php
<?php
class Database {
    private $host;
    private $user;
    private $password;
    private $database;
    private $conn;

    public function __construct($host, $user, $password, $database) {
        $this->host = $host;
        $this->user = $user;
        $this->password = $password;
        $this->database = $database;
    }

    public function connect() {
        $this->conn = new mysqli($this->host, $this->user, $this->password, $this->database);
        if ($this->conn->connect_error) {
            die("Connection failed: " . $this->conn->connect_error);
        }
        return $this->conn;
    }

    public function close() {
        $this->conn->close();
    }
}
// Initialize a new instance of the Database class
$db = new Database("localhost:4000", "root", "", "forkdb");
```

This is a PHP code for a Database class that has methods to connect and close a database connection. The class constructor takes four parameters which are the host name, database username, password, and database name. The class has a public method named **connect()** which connects to the database using the mysqli class and returns the connection object. If the connection fails, the **connect()** method will throw an error message. There is also a public method named **close()** which simply closes the database connection. At the end of the code, a new instance of the Database class is created and assigned to the variable **$db**, with the connection details passed as parameters to the constructor.

MODEL

```php
<?php
require_once 'dbconnection.php';

class Order {
    private $db;

    public function __construct(Database $db) {
        $this->db = $db;
    }

    public function addOrder($first_name, $last_name, $address, $apt_suite, $city, $state, $postal_code, $phone_number, $email, $card_number, $expiration_date, $cvv, $order_date) {
        $conn = $this->db->connect();

        $stmt = $conn->prepare("INSERT INTO orders (first_name, last_name, address, apt_suite, city, state, postal_code, phone_number, email, card_number, expiration_date, cvv, order_date) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
        $stmt->bind_param("sssssssssssss", $first_name, $last_name, $address, $apt_suite, $city, $state, $postal_code, $phone_number, $email, $card_number, $expiration_date, $cvv, $order_date);
        $result = $stmt->execute();

        $stmt->close();
        // $this->db->close();

        return $result;
    }
}
?>
```

This is a PHP class called "Order" that is used to add a new order to the "orders" table in the database. It has a constructor that takes a Database object as a parameter and a method called "addOrder" that inserts new data into the "orders" table using a prepared statement with parameters. The method returns a boolean value indicating whether the insertion was successful or not.

Validation with JavaScript

```
  // Phone number
  var phone = document.getElementById("phoneNumf").value;
  var phoneRegex = /^(\()?\d{3}(\))?[- ]?\d{3}[- ]?\d{4}$/;
  // Email address
  var email = document.getElementById("emailf").value;
  var emailRegex = /^\S+@\S+\.\S+$/;

  if (!phone.match(phoneRegex)) {
    document.getElementById("phoneNumber").innerHTML ="Please enter a valid phone
number";
    isvalid = false;
  }
  else{
    document.getElementById("phoneNumber").innerHTML ="";
  }

  if (!email.match(emailRegex)) {
    document.getElementById("email").innerHTML ="Please enter a valid email
address";
    isvalid = false;
  }
  else{
    document.getElementById("email").innerHTML ="";
  }
```

This code snippet shows how the JavaScript validation for phone number and email address has been implemented. It first retrieves the values entered in the corresponding input fields using the **getElementById** method. Then, it checks whether they match the respective regular expressions (**phoneRegex** and **emailRegex**) using the **match** method. If they do not match, an error message is displayed using the **innerHTML** property of the corresponding error message element, and the variable **isvalid** is set to **false**. If they do match, the error message is cleared.

This logic is applied to other fields on the form as well, such as the first name, last name, address, city, state, zip code, credit card number, expiration date, and CVV code. Each field has its own regular expression that defines the format it should follow, and if the entered value does not match the format, an error message is displayed using the same logic as shown in this code snippet.

In this code snippet, a JavaScript event listener is added to the **nameInput** element to ensure that only valid characters are entered in real time. The listener listens for the **input** event and removes any non-letter and non-space characters using a regular expression. Then, it updates the input field with the cleaned value. This ensures that the **name** field only contains letters and spaces, and any other characters are removed immediately.

PHP Validation

```php
// Define regular expressions
$name_regex = "/^[a-zA-Z ]+$/";
/.. .. ..

$errors = array();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = filter_var($_POST['name'], FILTER_SANITIZE_STRING);
// .. ..
  if (!preg_match($name_regex, $name)) {
    $errors["name"] = "Name must contain only letters and spaces";
  }
//.. .. ..
//.. .. ..
//.. .. ..
  if (count($errors) == 0) {
    // Process form inputs if all fields are valid
    session_start();

    // Store form data in session variables
    $_SESSION['name'] = $name;
    $_SESSION['cardnumber'] = $cardnumber;
    $_SESSION['expirationdate'] = $expirationdate;
    $_SESSION['securitycode'] = $securitycode;

    // Redirect to dbprocessor page if form was successfully processed
    header("Location: dbprocessor.php");
    exit();
  }
}
// Store errors in session variable
session_start();
```

```php
$_SESSION["errorsCard"] = $errors;

// Redirect to index page with error messages
header("Location: card.php");
die();
?>
```

This PHP file contains server-side validation for the card information entered by the user. The regular expressions are defined for each input field to ensure that the user enters valid information. The code checks if the HTTP request method is POST and then retrieves the values from the form fields using the **filter_var()** function.

If any of the regular expressions fail to match, an error message is added to the **$errors** array. If there are no errors, the form data is stored in session variables and the user is redirected to the **dbprocessor.php** page to process the form data and insert it into the database.

If there are errors, the errors are stored in a session variable and the user is redirected back to the **card.php** page with error messages displayed next to the form fields.

Conclusion

In conclusion, this was a nice challenge that allowed me to apply my knowledge of web development and learn some new things along the way. Through this challenge, I was able to improve my understanding of PHP, JavaScript, and regular expressions. I hope my solutions demonstrate my skills and abilities as a web developer, and I look forward to the opportunity to work with you.