**E RATION PORTAL**

**A MINI PROJECT REPORT**

**Submitted by**

**GOPIKRISHNAN R 220701075**

**P HARIHARAVISWANATHAN 220701082**

# In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-6021

# BONAFIDE CERTIFICATE

Certified that this project report “**E RATION PORTAL**” is the bonafide work of

**“GOPIKRISHNAN R** **(220701075)”**

**“P HARIHARAVISWANATHAN (220701082)”**

who carried out the project work under my supervision.

**Submitted for the Practical Examination held on**

|  |  |
| --- | --- |
| **SIGNATURE** | **SIGNATURE** |
| **Dr.R.SABITHA** | **Dr.G.Dharani Devi** |
|  |  |

**Professor and II Year Academic Head Associate Professor ,**

|  |  |
| --- | --- |
| **Computer Science and Engineering,** | **Computer Science and Engineering,** |
| **Rajalakshmi Engineering College** | **Rajalakshmi Engineering College,** |
| **(Autonomous),** | **(Autonomous),** |
| **Thandalam, Chennai - 602 105** | **Thandalam, Chennai - 602 105** |
| **INTERNAL EXAMINER** | **EXTERNAL EXAMINER** |

# ABSTRACT

E RATION PORTAL is an online billing system for the government run ration shops where people can bill the items online and pay for it directly using digital payment methods and print the bill as a pdf. This eliminates the need to wait in long ques for billing items physically at the shops which also saves time for the busy population.

**TABLE OF CONTENTS**

1. **INTRODUCTION**
   1. INTRODUCTION
   2. OBJECTIVES
   3. MODULES
2. **SURVEY OF TECHNOLOGIES**
   1. SOFTWARE DESCRIPTION

## 2.2 LANGUAGES

2.2.1 SQL

### 2.2.2 PYTHON

1. **REQUIREMENTS AND ANALYSIS**

3.1 REQUIREMENT SPECIFICATION

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

` 3.3 ARCHITECTURE DIAGRAM

3.4 ER DIAGRAM

3.5 NORMALIZATION

1. **PROGRAM CODE**
2. **RESULTS AND DISCUSSION**

**6.CONCLUSION**

**7.REFERENCES**

**CHAPTER 1**

**INTRODUCTION**

# INTRODUCTION

**E RATION PORTAL: Revolutionizing Ration Shopping**

E RATION PORTAL is a project aimed at modernizing the process of purchasing ration items from government-run shops. By introducing an online billing system, this project seeks to provide convenience to the general public by eliminating the need to physically visit the shops and stand in long queues for billing. The portal allows users to bill their items online, pay for them using digital payment methods, and print the bill as a PDF, thereby saving time and streamlining the ration shopping process.

# 1.2 OBJECTIVES

* **Development of the online billing system:** The first milestone of the project involved the development of the E RATION PORTAL, which includes features such as item selection, billing, payment processing, and bill generation.
* **Integration of digital payment methods:** The project team worked on integrating various digital payment methods such as credit/debit cards, net banking, and mobile wallets to facilitate easy and secure payments.
* **User testing and feedback collection:** A key milestone in the project was conducting user testing to ensure the portal's usability and functionality. Feedback from users was collected to make necessary improvements.
* **Training for ration shop employees:** As part of the project, training sessions were conducted for ration shop employees to familiarize them with the online billing system and ensure smooth operations.
* **Launch of the E RATION PORTAL:** The final milestone of the project was the official launch of the portal, making it available for the public to use for their ration shopping needs.

**1.3 MODULES**

1. **Flask Web Application Module**: This module includes the Flask application setup, routes definition, and configuration.
2. **Database Module:** This module handles the connection to the PostgreSQL database and executes queries to retrieve product information.
3. **Authentication Module:** This module includes routes and functions related to user authentication, such as logging in and generating OTP (One-Time Password).
4. **Product Management Module:** This module handles rendering product pages, adding products to the cart, and retrieving product information from the database.
5. **Shopping Cart Module:** This module manages the user's shopping cart, including adding products, updating quantities, and clearing the cart.
6. **Payment Module:** This module handles the calculation of total prices and rendering the payment page

**CHAPTER 2**

**SURVEY OF TECHNOLOGIES**

**2.1 SOFTWARE DESCRIPTION**

**1.Architecture Overview:** The architecture of the Online Shopping Platform follows a model-view-controller (MVC) design pattern, facilitating separation of concerns and modularity. The application consists of the following components:

* **Model:** Represents the data model of the application, including user accounts, product information, and shopping cart contents. Data is stored and managed in a PostgreSQL database.
* **View:** Handles the presentation layer of the application, rendering HTML templates to display product catalogs, shopping carts, and payment pages. Templates are populated with data retrieved from the database and session variables.
* **Controller:** Implements the business logic and request handling of the application. Flask routes define URL endpoints for handling user requests such as authentication, product browsing, and payment processing.

**2. Component Description:**

* **Flask Application:** The core component of the platform, responsible for initializing the Flask application instance, configuring routes, and managing application-wide settings. The Flask application serves as the entry point for handling HTTP requests and dispatching them to the appropriate controllers.
* **Database Module:** Handles database connectivity and query execution using the psycopg2 library. This module establishes a connection to the PostgreSQL database and executes SQL queries to retrieve product information and update shopping cart contents.
* **Authentication Module:** Manages user authentication and session handling. This module includes routes for user login, OTP generation, and session management. User credentials are verified against stored records in the database, and session variables are used to maintain user authentication status and shopping cart contents.
* **Product Management Module:** Implements functionality for browsing product catalogs, adding products to the shopping cart, and retrieving product details. Routes within this module handle requests for rendering product pages, fetching product data from the database, and updating the shopping cart with selected items.
* **Shopping Cart Module:** Provides features for managing the user's shopping cart, including adding, removing, and updating product quantities. Session variables are used to store the contents of the shopping cart across multiple user interactions, ensuring persistence and continuity.
* **Payment Module:** Facilitates secure payment processing and order completion. This module calculates the total price of selected items, renders the payment page with appropriate payment options, and handles payment confirmation. Payment transactions are processed securely using encryption protocols and payment gateway integration.

**+2.2 LANGUAGES**

**PYTHON/SQL**

1. **ration-shop/app.py**:
   * Flask application entry point. Defines routes for login, OTP, product viewing, and payment processing.
2. **ration-shop/templates/products.html**:
   * HTML template for displaying product listings and details.
3. **ration-shop/templates/login.html**:
   * HTML template for user login page.
4. **ration-shop/templates/get\_otp.html**:
   * HTML template for OTP verification page.
5. **ration-shop/templates/payment.html**:
   * HTML template for payment processing page.
6. **ration-shop/static/tnlogo.png**:
   * Static image file for the platform's logo.
7. **admin/inserter.py**:
   * Python script for inserting data into the database.
8. **admin/creator.py**:
   * Python script for creating database tables.
9. **requirements.txt**:
   * List of Python dependencies for the application.
10. **README.md**:
    * Documentation file providing project overview and instructions.

**CHAPTER 3**

**REQUIREMENTS AND ANALYSIS**

**3.1 REQUIREMENT SPECIFICATION**

**USERSTORIES**

1. As a user, I want to be able to create an account on the E RATION PORTAL so that I can start purchasing ration items online.
2. As a user, I want to browse through the catalog of ration items available on the portal, including details like item name, quantity, and price.
3. As a user, I want to add desired items to my shopping cart and have the flexibility to adjust quantities or remove items before proceeding to checkout.
4. As a user, I want to be able to review the items in my shopping cart and see the total cost before initiating the billing process.
5. As a user, I want to choose from a variety of digital payment methods such as credit/debit cards, net banking, or mobile wallets to pay for my purchases.
6. As a user, I want the billing process to be seamless and secure, with clear indications of successful payment and generation of a PDF bill for my records.
7. As a user, I want the option to print the generated bill as a PDF document for future reference or reimbursement purposes.
8. As a user, I want the E RATION PORTAL to be accessible from various devices such as desktops, laptops, tablets, and smartphones, for convenient shopping anytime, anywhere.
9. As a user, I want the portal to provide timely updates on the availability of ration items and any changes in prices or stock.
10. As a user, I want the E RATION PORTAL to ensure the security of my personal information and payment details, with robust measures in place to prevent unauthorized access or data breaches.
11. As a user, I want the portal interface to be intuitive and user-friendly, with easy navigation and clear instructions, to enhance my overall shopping experience.
12. As a user, I want the E RATION PORTAL to comply with all relevant government regulations and guidelines regarding online transactions and data privacy, ensuring a trustworthy and legal platform for ration shopping.

**3.2 Hardware and Software Requirements:**

**\*Hardware Requirements:**

1. **Server Infrastructure:**

- The portal will require a robust server infrastructure capable of handling concurrent user sessions and managing database operations efficiently.

- Recommended specifications:

- Processor: Multi-core processor with sufficient processing power to handle simultaneous requests.

- RAM: Adequate RAM to support the operating system, web server, database server, and concurrent user sessions.

- Storage: Sufficient storage space to store application files, user data, and transaction records.

- Network Interface: High-speed internet connection to ensure fast data transfer between clients and the server.

2. **Client Devices:**

- Users will access the portal from various client devices such as desktop computers, laptops, tablets, and smartphones.

- The portal should be optimized for compatibility with different screen sizes and resolutions.

- Minimum requirements for client devices:

- Desktop/Laptop: Standard configuration with a modern web browser installed (e.g., Google Chrome, Mozilla Firefox, Safari).

- Mobile Devices: Compatibility with iOS and Android platforms, supporting popular mobile browsers.

**Software Requirements:**

1. **Operating System:**

- Server: Linux-based operating system (e.g., Ubuntu Server, CentOS) for hosting the web server and database server.

- Client Devices: Compatibility with major operating systems including Windows, macOS, iOS, and Android.

2. **Web Server:**

- Apache HTTP Server or Nginx: To host the web application and serve web pages to clients.

- The web server should support HTTPS encryption to ensure secure communication between the server and clients.

3. **Database Management System:**

- MySQL or PostgreSQL: Relational database management system (RDBMS) for storing user data, product information, and transaction records.

- The database system should be scalable to accommodate increasing data volumes over time.

4. **Programming Languages and Frameworks:**

- Backend: Node.js with Express.js or Python with Django framework for server-side application logic and API development.

- Frontend: HTML, CSS, and JavaScript frameworks/libraries such as React.js or AngularJS for building the user interface.

- JavaScript runtime environment such as Node.js for executing server-side JavaScript code.

5. **Payment Gateway Integration:**

- Integration with payment gateway APIs for processing digital payments securely.

- Compatibility with popular payment gateways such as PayPal, Stripe, or Razorpay, based on user preferences and availability in the target region.

6. **PDF Generation Library:**

- A library or tool for generating PDF documents dynamically, based on the billing information and transaction details.

- Options include libraries like PDFKit (Node.js), jsPDF (JavaScript), or server-side tools like wkhtmltopdf.

7. **Security Measures:**

- Implementation of security protocols and libraries to safeguard user data and prevent security breaches.

- Integration with authentication and authorization mechanisms for user access control.

- Implementation of secure coding practices to mitigate common security vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

8. **Regulatory Compliance:**

- Compliance with relevant government regulations and standards governing online transactions, data privacy, and security.

- Regular updates and maintenance to ensure continued compliance with evolving regulatory requirements.

9. **Monitoring and Analytics:**

- Tools for monitoring server performance, tracking user activities, and analyzing usage patterns.

- Integration with analytics platforms for gathering insights into user behavior, preferences, and trends, to inform decision-making and future enhancements.

**3.3 ARCHITUCTURE DIGRAM:**

**+---------------------+**

**| User |**

**+---------------------+**

**|**

**| Accesses**

**|**

**+---------------------+**

**| E RATION PORTAL |**

**| Web Application |**

**+---------------------+**

**|**

**| Communicates with**

**|**

**+------------------|------------------+**

**| | |**

**+---------+---------+ +-+----------------+--+**

**| Frontend/UI | | Backend/API |**

**| (HTML/CSS/JS) | |(Node.js/Express.js/|**

**+------------------+ | Django/Python) |**

**| +----------------------+**

**| |**

**| | Queries**

**| |**

**+---------+---------+ |**

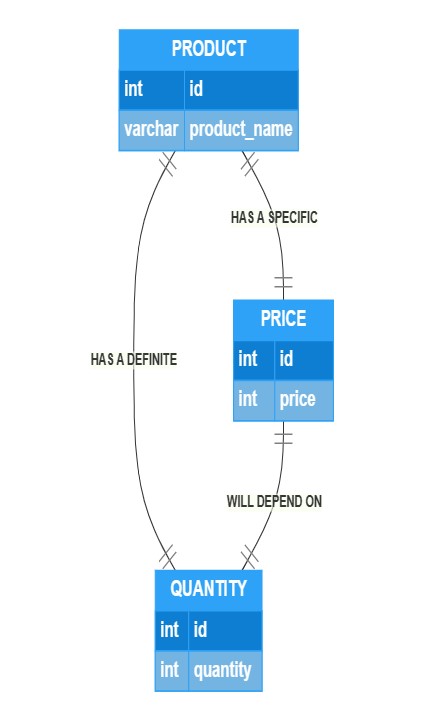
**| Payment | +------------------+**

**| Gateway APIs | | Database |**

**+------------------+ | (MySQL/PostgreSQL|**

**+------------------+**

**3.4 ER DIAGRAM**



**3.5 NORMALISATION**

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **product\_name** | **quantity** | **price** |
| 1 | Rice | 100 | 10 |
| 2 | Lentils | 50 | 20 |
| 3 | Sugar | 75 | 15 |
| 4 | Kerosine | 30 | 25 |
| 5 | Wheat | 50 | 10 |
| 6 | Tumeric | 75 | 15 |
| 7 | Goldwinner | 50 | 40 |
| 8 | Salt | 75 | 10 |
| 9 | Wheat | 50 | 10 |
| 10 | Turmeic | 75 | 15 |

**1NF: FIRST NORMAL FORM**

* All the attributes in table are atomic
* They all have unique values
* Hence table is in 1NF

|  |  |
| --- | --- |
| **id** | **product\_name** |
| 1 | Rice |
| 2 | Lentils |
| 3 | Sugar |
| 4 | Kerosine |
| 5 | Wheat |
| 6 | Tumeric |
| 7 | Goldwinner |
| 8 | Salt |
| 9 | Wheat |
| 10 | Turmeric |

|  |  |  |
| --- | --- | --- |
| **id** | **price** | **quantity** |
| 1 | 100 | 10 |
| 2 | 50 | 20 |
| 3 | 75 | 15 |
| 4 | 30 | 25 |
| 5 | 50 | 10 |
| 6 | 75 | 15 |
| 7 | 50 | 40 |
| 8 | 75 | 10 |
| 9 | 50 | 10 |
| 10 | 75 | 15 |

**2NF: SECOND NORMAL FORM**

* Table is in 1NF
* But they have partial dependency
* After removing partial dependency the above table is in 2NF

|  |  |
| --- | --- |
| **id** | **product\_name** |
| 1 | Rice |
| 2 | Lentils |
| 3 | Sugar |
| 4 | Kerosine |
| 5 | Wheat |
| 6 | Tumeric |
| 7 | Goldwinner |
| 8 | Salt |
| 9 | Wheat |
| 10 | Turmeric |

|  |  |
| --- | --- |
| **id** | **price** |
| 1 | 100 |
| 2 | 50 |
| 3 | 75 |
| 4 | 30 |
| 5 | 50 |
| 6 | 75 |
| 7 | 50 |
| 8 | 75 |
| 9 | 50 |
| 10 | 75 |

|  |
| --- |
| **quantity** |
| 10 |
| 20 |
| 15 |
| 25 |
| 10 |
| 15 |
| 40 |
| 10 |
| 10 |
| 15 |

|  |  |
| --- | --- |
| **id** |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |

**3NF:THIRD NORMAL FORM**

* Table is in 2NF
* But they have transitive dependency

**BCNF: BOYCE C0DD NORMAL FORM**

* Table is in 3NF
* All attributes are associated with a primary key only

**CHAPTER 4**

**PROGRAM CODE**

**FILE STRUCTURE**

RATION/

**│**

**├──** ration-shop/

**│ ├──** app.py

**│ ├──** templates/

**│ │ ├──** products.html

**│ │ ├──** login.html

**│ │ ├──** get\_otp.html

**│ │ └──** payment.html

**│ │**

**│ └──** static/

**│ └──** tnlogo.png

**├──** admin/

**│ ├──** inserter.py

**│ ├──** doc

**│ └──** creator.py

**│**

**├──** requirements.txt

**└──** README.md

**app.py**

---from flask import Flask, render\_template, request, session, redirect, url\_for

import psycopg2

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'

# Connect to PostgreSQL database

conn\_string = 'postgres://wnoqsymb:d3lVJfHNP5xs7A8-nyKHmMzehIqSDK3b@cornelius.db.elephantsql.com/wnoqsymb'

conn = psycopg2.connect(conn\_string)

# Route for the login page

@app.route("/", methods=["GET", "POST"])

def login():

if request.method == "POST":

session['username'] = request.form['username']

return redirect(url\_for('get\_otp'))

return render\_template("login.html")

# Route for the OTP page

@app.route("/get\_otp", methods=["GET", "POST"])

def get\_otp():

if request.method == "POST":

# Dummy check for OTP, should be replaced with actual OTP validation

if request.form['otp'] == "123456":

return redirect(url\_for('render\_products'))

return render\_template("get\_otp.html")

# Route for rendering the products page

@app.route("/render\_products")

def render\_products():

cur = conn.cursor()

cur.execute("SELECT \* FROM stock")

rows = cur.fetchall()

product\_data = [{'id': row[0], 'product\_name': row[1], 'price': row[3]} for row in rows]

cur.close()

return render\_template("products.html", products=product\_data)

# Route for adding product to cart and displaying payment page

@app.route("/add\_to\_cart/<int:product\_id>", methods=["GET", "POST"])

def add\_to\_cart(product\_id):

if request.method == "POST":

quantity = int(request.form.get('quantity', 1)) # Get quantity from form, default to 1

# Retrieve product information from the database

cur = conn.cursor()

cur.execute("SELECT \* FROM stock WHERE id = %s", (product\_id,))

row = cur.fetchone()

cur.close()

if row:

product\_name = row[1]

price = row[3]

# Construct the product dictionary

product = {'id': product\_id, 'name': product\_name, 'price': price, 'quantity': quantity}

# Retrieve the cart from the session or create a new one if it doesn't exist

cart = session.get('cart', [])

# Check if the product is already in the cart

for item in cart:

if item['id'] == product\_id:

item['quantity'] += quantity

break

else:

# If the product is not in the cart, add it

cart.append(product)

# Update the cart in the session

session['cart'] = cart

# Redirect to the payment page

return redirect(url\_for('payment'))

# If the request method is GET or if there's an error, redirect back to the products page

return redirect(url\_for('render\_products'))

# Route for displaying payment page

@app.route("/payment")

def payment():

cart = session.get('cart', [])

valid\_items = [item for item in cart if item['price'] is not None]

total\_price = sum(float(item['price']) \* item['quantity'] for item in valid\_items)

return render\_template("payment.html", cart=cart, total\_price=total\_price)

# Route for clearing the cart

@app.route("/clear\_cart", methods=["POST"])

def clear\_cart():

session.pop('cart', None) # Remove the 'cart' from session

return '', 204 # Return empty response with status code 204 (No Content)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**login.html**

<!DOCTYPE html>

<html lang="en">

<head>

<link rel="icon" href="https://cdn.swisscows.com/image?url=https%3A%2F%2Fclipground.com%2Fimages%2Ftamil-nadu-government-logo-png-6.png" type="image/x-icon">

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login</title>

<!-- Bootstrap CSS -->

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

<style>

body {

font-family: Arial, sans-serif;

background-color: #f5f5f5;

background-image: url('data:image/svg+xml,%3Csvg width="10" height="10" viewBox="0 0 10 10" xmlns="http://www.w3.org/2000/svg"%3E%3Ccircle cx="5" cy="5" r="3" fill="%23b4e1b1" fill-opacity="0.3" /%3E%3C/svg%3E');

}

.header {

background-color: #ffffff;

color: #fff;

padding: 10px;

display: flex;

align-items: center;

justify-content: space-between;

}

.logo {

max-width: 100px;

height: auto;

margin-right: 10px;

}

.container {

background-color: #fff;

padding: 20px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.info {

margin-bottom: 20px;

}

.info-text {

padding: 20px;

overflow-y: auto;

max-height: 200px; /\* Set a fixed height for the container \*/

}

.info-text h3 {

margin-top: 0;

}

.login-form {

max-width: 60%;

width: 100%;

margin: 0 auto; /\* Center the form horizontally \*/

}

h2 {

text-align: center;

color: #333;

}

form {

margin-top: 20px;

}

label {

display: block;

margin-bottom: 5px;

color: #333;

}

input[type="text"],

input[type="password"] {

width: 100%;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 5px;

box-sizing: border-box; /\* Ensure that padding and border are included in the width \*/

}

input[type="submit"] {

width: 100%;

padding: 10px;

background-color: #007bff;

color: #fff;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

input[type="submit"]:hover {

background-color: #0056b3;

}

.error {

color: red;

text-align: center;

margin-top: 10px;

}

.toggle-btn {

margin-top: 20px;

padding: 10px 20px;

background-color: transparent;

color: #7f7f7f;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

.toggle-btn:hover {

background-color: #0056b3;

}

footer {

position: fixed;

bottom: 0;

width: 100%;

background-color: #035703;

color: #fff;

padding: 10px;

text-align: center;

}

/\* New CSS for call button \*/

.call-btn {

display: inline-block;

background-color: transparent;

color: #7f7f7f;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

margin-right: 20px;

font-size:20px;

}

.dropdown-content {

display: none;

position: absolute;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);

z-index: 1;

}

.dropdown-content a {

color: black;

padding: 12px 16px;

text-decoration: none;

display: block;

}

.dropdown-content a:hover {

background-color: #ddd;

}

.dropdown:hover .dropdown-content {

display: block;

}

</style>

</head>

<body>

<div class="header">

<img src="static/tnlogo.png" alt="Tamil Nadu Logo" class="logo">

<h2>Tamil Nadu Government</h2>

<div class="dropdown">

<button class="call-btn">📞Call</button>

<div class="dropdown-content">

<a href="#">A State Level Tollfree PDS Helpline is functional under short code 1967 and 1800-425-5901.</a>

</div>

</div>

</div>

<div class="container">

<div class="row">

<div class="col-md-6 offset-md-3">

<div class="info">

<div class="login-form">

{% if error %}

<p class="error">{{ error }}</p>

{% endif %}

<h2>Login</h2>

<form action="{{ url\_for('login') }}" method="post">

<label for="username">Ration Card Number:</label>

<input type="text" id="username" name="username" required>

<label for="password">Mobile Number:</label>

<input type="text" id="password" name="password" required>

<input type="submit" value="Get OTP">

</form>

</div>

</div>

<div class="info-text" id="info-text">

<h3>ரேஷன் அட்டை</h3>

<p>Ration Card serves as an important document for availing various government schemes and benefits. It can be used to apply for scholarships, subsidized housing, and other social welfare programs. It is also a valid proof of identity and residence for various administrative and legal purposes.</p>

<p style="display: none;">அரசின் பல்வேறு திட்டங்கள் மற்றும் பலன்களைப் பெறுவதற்கு ரேஷன் கார்டு ஒரு முக்கிய ஆவணமாகச் செயல்படுகிறது. உதவித்தொகை, மானிய விலையில் வீடுகள் மற்றும் பிற சமூக நலத் திட்டங்களுக்கு விண்ணப்பிக்க இதைப் பயன்படுத்தலாம். இது பல்வேறு நிர்வாக மற்றும் சட்ட நோக்கங்களுக்காக அடையாளம் மற்றும் வசிப்பிடத்திற்கான சரியான ஆதாரமாகும்.</p>

</div>

</div>

</div>

<button class="toggle-btn" onclick="toggleText()">தமிழ்</button>

</div>

<footer>

Copyright Ⓒ Government of Tamil Nadu. All Rights Reserved

</footer>

<!-- Bootstrap JS and jQuery -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<script>

function toggleText() {

var infoText = document.getElementById('info-text');

var paragraphs = infoText.getElementsByTagName('p');

var index = 0;

if (paragraphs[0].style.display === 'none') {

paragraphs[0].style.display = 'block';

paragraphs[1].style.display = 'none';

document.querySelector('.toggle-btn').textContent = 'தமிழ்';

} else {

paragraphs[0].style.display = 'none';

paragraphs[1].style.display = 'block';

document.querySelector('.toggle-btn').textContent = 'English';

}

}

</script>

</body>

</html>

**products.html**

User

<!-- products.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<link rel="icon" href="https://cdn.swisscows.com/image?url=https%3A%2F%2Fclipground.com%2Fimages%2Ftamil-nadu-government-logo-png-6.png" type="image/x-icon">

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Products</title>

<!-- Bootstrap CSS -->

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

<style>

body {

font-family: Arial, sans-serif;

background-color: #f5f5f5;

background-image: url('data:image/svg+xml,%3Csvg width="10" height="10" viewBox="0 0 10 10" xmlns="http://www.w3.org/2000/svg"%3E%3Ccircle cx="5" cy="5" r="3" fill="%23b4e1b1" fill-opacity="0.3" /%3E%3C/svg%3E');

}

.header {

background-color: #ffffff;

color: #fff;

padding: 10px;

display: flex;

align-items: center;

justify-content: space-between;

}

.logo {

max-width: 100px;

height: auto;

margin-right: 10px;

}

.container {

background-color: #fff;

padding: 20px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.info {

margin-bottom: 20px;

}

.info-text {

padding: 20px;

overflow-y: auto;

max-height: 200px; /\* Set a fixed height for the container \*/

}

.info-text h3 {

margin-top: 0;

}

.login-form {

max-width: 60%;

width: 100%;

margin: 0 auto; /\* Center the form horizontally \*/

}

h2 {

text-align: center;

color: #333;

}

form {

margin-top: 20px;

}

label {

display: block;

margin-bottom: 5px;

color: #333;

}

input[type="text"],

input[type="password"] {

width: 100%;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 5px;

box-sizing: border-box; /\* Ensure that padding and border are included in the width \*/

}

input[type="submit"] {

width: 100%;

padding: 10px;

background-color: #007bff;

color: #fff;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

input[type="submit"]:hover {

background-color: #0056b3;

}

.error {

color: red;

text-align: center;

margin-top: 10px;

}

.toggle-btn {

margin-top: 20px;

padding: 10px 20px;

background-color: transparent;

color: #7f7f7f;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

.toggle-btn:hover {

background-color: #0056b3;

}

footer {

position: fixed;

bottom: 0;

width: 100%;

background-color: #035703;

color: #fff;

padding: 10px;

text-align: center;

}

/\* New CSS for call button \*/

.call-btn {

display: inline-block;

background-color: transparent;

color: #7f7f7f;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

margin-right: 20px;

font-size:20px;

}

.dropdown-content {

display: none;

position: absolute;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);

z-index: 1;

}

.dropdown-content a {

color: black;

padding: 12px 16px;

text-decoration: none;

display: block;

}

.dropdown-content a:hover {

background-color: #ddd;

}

.dropdown:hover .dropdown-content {

display: block;

}

.card {

margin-bottom: 20px;

border: none;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

transition: box-shadow 0.3s ease;

}

.card:hover {

box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);

}

.card-body {

padding: 20px;

}

.product-name {

font-size: 18px;

margin-bottom: 10px;

color: #333;

}

.price {

font-size: 16px;

color: #007bff;

}

.quantity-input {

width: 60px;

padding: 5px;

text-align: center;

border: 1px solid #ccc;

border-radius: 5px;

margin-right: 10px;

}

.add-to-cart-btn {

background-color: #ff9900;

color: #fff;

border: none;

border-radius: 5px;

padding: 10px 20px;

cursor: pointer;

transition: background-color 0.3s;

}

.add-to-cart-btn:hover {

background-color: #ff8000;

}

</style>

</head>

<body>

<div class="header">

<img src="static/tnlogo.png" alt="Tamil Nadu Logo" class="logo">

<h2>Tamil Nadu Government</h2>

<div class="dropdown">

<button class="call-btn">📞Call</button>

<div class="dropdown-content">

<a href="#">A State Level Tollfree PDS Helpline is functional under short code 1967 and 1800-425-5901.</a>

</div>

</div>

</div>

<h1>Products</h1>

{% for product in products %}

<div class="card">

<div class="card-body">

<h3 class="product-name">{{ product['product\_name'] }}</h3>

<p class="price">Price: ₹{{ product['price'] }}</p>

<form action="{{ url\_for('add\_to\_cart', product\_id=product['id']) }}" method="post" class="d-flex align-items-center">

<input type="number" name="quantity" value="1" min="1" class="quantity-input">

<input type="hidden" name="product\_id" value="{{ product['id'] }}">

<button type="submit" class="add-to-cart-btn">Add to Cart</button>

</form>

</div>

</div>

{% endfor %}

</body>

</html>

**payments.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Payment</title>

<style>

body {

font-family: Arial, sans-serif;

}

table {

width: 100%;

border-collapse: collapse;

margin-bottom: 20px;

}

th, td {

border: 1px solid #ddd;

padding: 8px;

text-align: left;

}

th {

background-color: #f2f2f2;

}

tfoot {

font-weight: bold;

}

.container {

max-width: 800px;

margin: 0 auto;

padding: 20px;

}

.header {

text-align: center;

margin-bottom: 20px;

}

.qr-code {

text-align: center;

margin-bottom: 20px;

}

.qr-code img {

width: 200px; /\* Adjust the size of the QR code as needed \*/

}

</style>

</head>

<body>

<div class="container">

<div class="header">

<h1>Payment Bill</h1>

</div>

<div class="qr-code">

<img src="https://cdn.swisscows.com/image?url=https%3A%2F%2Ftse2.mm.bing.net%2Fth%3Fid%3DOIP.MF7dS6zQ0D6bnRltUz\_oawHaHa%26pid%3DApi" alt="QR Code">

</div>

<table>

<thead>

<tr>

<th>Product Name</th>

<th>Quantity</th>

<th>Price</th>

</tr>

</thead>

<tbody>

{% for item in cart %}

<tr>

<td>{{ item.name }}</td>

<td>{{ item.quantity }}</td>

<td>₹{{ item.price }}</td>

</tr>

{% endfor %}

</tbody>

<tfoot>

<tr>

<td colspan="2">Total Price:</td>

<td>₹{{ total\_price }}</td>

</tr>

</tfoot>

</table>

<p>Thank you for your payment!</p>

<form id="clear-cart-form" method="post" action="/clear\_cart">

<button type="button" class="btn btn-danger" onclick="clearCart()">Clear Cart</button>

</form>

<script>

function clearCart() {

var form = document.getElementById('clear-cart-form');

var xhr = new XMLHttpRequest();

xhr.open('POST', form.action, true);

xhr.onload = function () {

if (xhr.status === 200) {

// Reload the page after successful cart clear

window.location.reload();

}

};

xhr.send(new FormData(form));

}

</script>

</body>

</html>

**get\_otp.html**

**<**!DOCTYPE html>

<html lang="en">

<head>

<link rel="icon" href="https://cdn.swisscows.com/image?url=https%3A%2F%2Fclipground.com%2Fimages%2Ftamil-nadu-government-logo-png-6.png" type="image/x-icon">

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>OTP</title>

<!-- Bootstrap CSS -->

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

<style>

body {

font-family: Arial, sans-serif;

background-color: #f5f5f5;

background-image: url('data:image/svg+xml,%3Csvg width="10" height="10" viewBox="0 0 10 10" xmlns="http://www.w3.org/2000/svg"%3E%3Ccircle cx="5" cy="5" r="3" fill="%23b4e1b1" fill-opacity="0.3" /%3E%3C/svg%3E');

}

.header {

background-color: #ffffff;

color: #fff;

padding: 10px;

display: flex;

align-items: center;

justify-content: space-between;

}

.logo {

max-width: 100px;

height: auto;

margin-right: 10px;

}

.container {

background-color: #fff;

padding: 20px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.info {

margin-bottom: 20px;

}

.info-text {

padding: 20px;

overflow-y: auto;

max-height: 200px; /\* Set a fixed height for the container \*/

}

.info-text h3 {

margin-top: 0;

}

.login-form {

max-width: 60%;

width: 100%;

margin: 0 auto; /\* Center the form horizontally \*/

}

h2 {

text-align: center;

color: #333;

}

form {

margin-top: 20px;

}

label {

display: block;

margin-bottom: 5px;

color: #333;

}

input[type="text"],

input[type="password"] {

width: 100%;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 5px;

box-sizing: border-box; /\* Ensure that padding and border are included in the width \*/

}

input[type="submit"] {

width: 100%;

padding: 10px;

background-color: #007bff;

color: #fff;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

input[type="submit"]:hover {

background-color: #0056b3;

}

.error {

color: red;

text-align: center;

margin-top: 10px;

}

.toggle-btn {

margin-top: 20px;

padding: 10px 20px;

background-color: transparent;

color: #7f7f7f;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

.toggle-btn:hover {

background-color: #0056b3;

}

footer {

position: fixed;

bottom: 0;

width: 100%;

background-color: #035703;

color: #fff;

padding: 10px;

text-align: center;

}

/\* New CSS for call button \*/

.call-btn {

display: inline-block;

background-color: transparent;

color: #7f7f7f;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

margin-right: 20px;

font-size:20px;

}

.dropdown-content {

display: none;

position: absolute;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);

z-index: 1;

}

.dropdown-content a {

color: black;

padding: 12px 16px;

text-decoration: none;

display: block;

}

.dropdown-content a:hover {

background-color: #ddd;

}

.dropdown:hover .dropdown-content {

display: block;

}

</style>

</head>

<body>

<div class="header">

<img src="static/tnlogo.png" alt="Tamil Nadu Logo" class="logo">

<h2>Tamil Nadu Government</h2>

<div class="dropdown">

<button class="call-btn">📞Call</button>

<div class="dropdown-content">

<a href="#">A State Level Tollfree PDS Helpline is functional under short code 1967 and 1800-425-5901.</a>

</div>

</div>

</div>

<div class="container">

<div class="row">

<div class="col-md-6 offset-md-3">

<div class="info">

<div class="login-form">

{% if error %}

<p class="error">{{ error }}</p>

{% endif %}

<h2>Login</h2>

<form action="/get\_otp" method="post">

<div class="inputfield">

<input type="text" name="otp" maxlength="6" class="input" required>

<input type="submit" value="SUBMIT">

</div>

</form>

</div>

</div>

</div>

</div>

<button class="toggle-btn" onclick="toggleText()">தமிழ்</button>

</div>

<footer>

Copyright Ⓒ Government of Tamil Nadu. All Rights Reserved

</footer>

<!-- Bootstrap JS and jQuery -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<script>

function toggleText() {

var infoText = document.getElementById('info-text');

var paragraphs = infoText.getElementsByTagName('p');

var index = 0;

if (paragraphs[0].style.display === 'none') {

paragraphs[0].style.display = 'block';

paragraphs[1].style.display = 'none';

document.querySelector('.toggle-btn').textContent = 'தமிழ்';

} else {

paragraphs[0].style.display = 'none';

paragraphs[1].style.display = 'block';

document.querySelector('.toggle-btn').textContent = 'English';

}

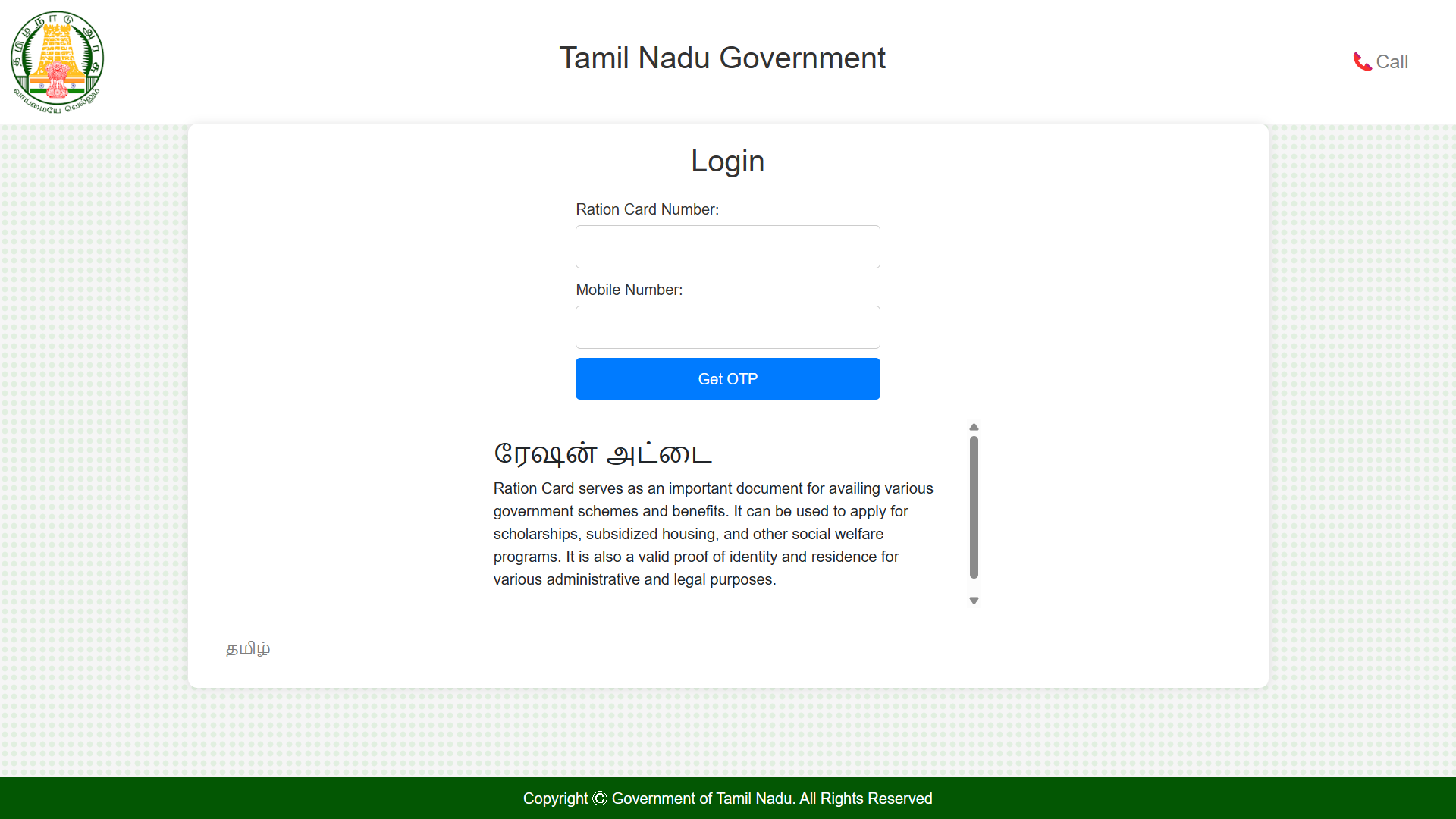
}

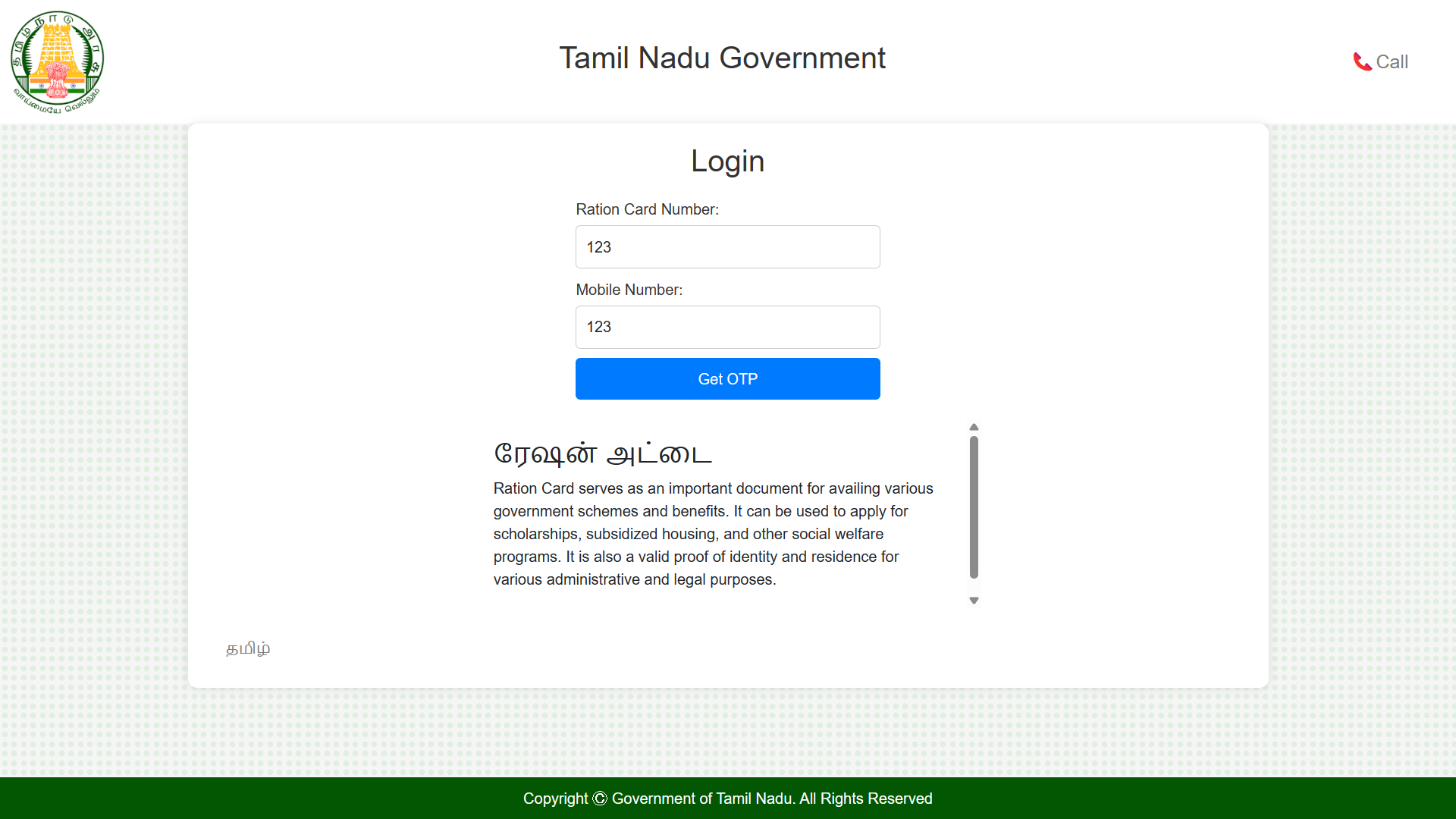
</script>

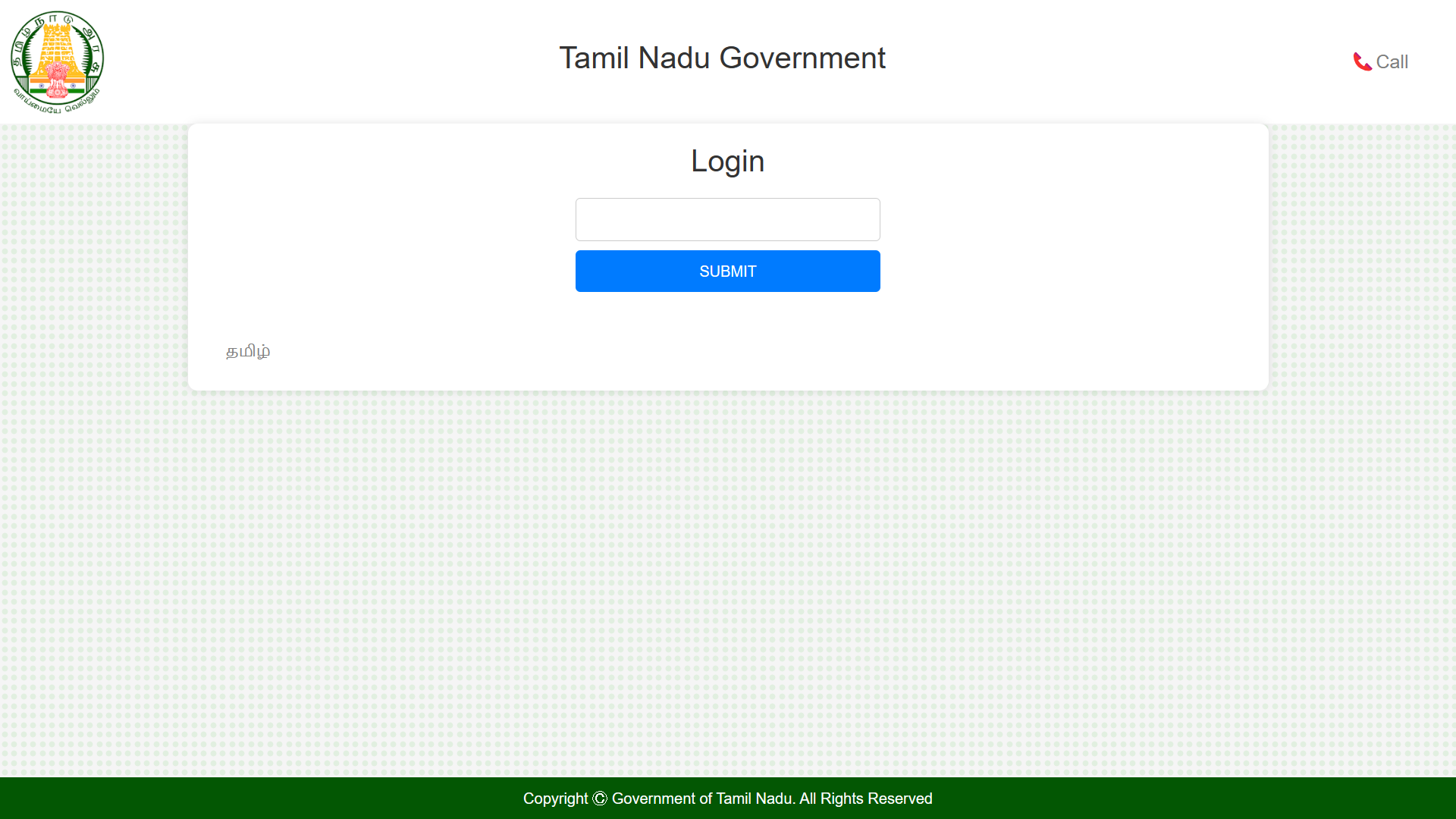
</body>

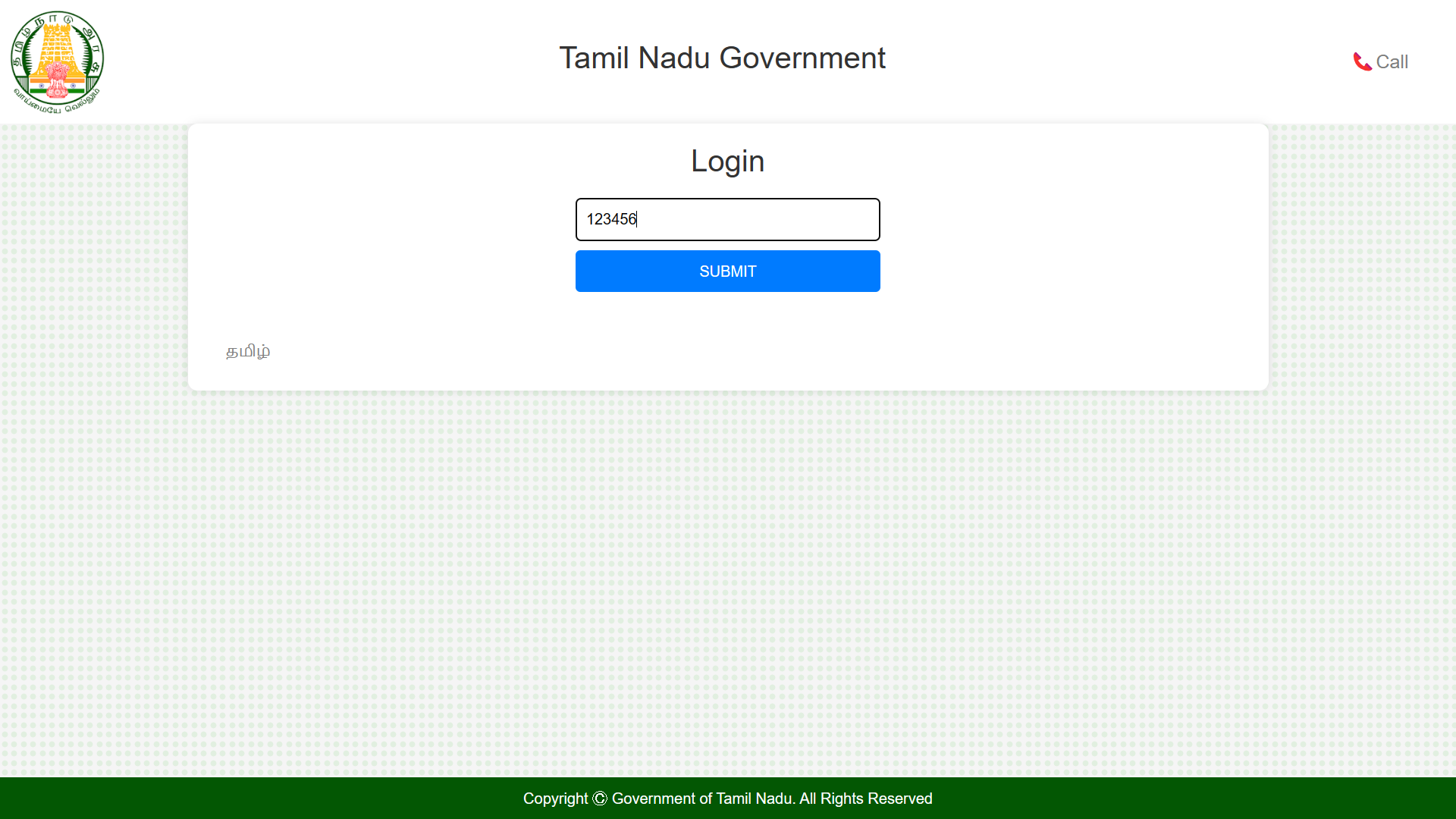
</html>

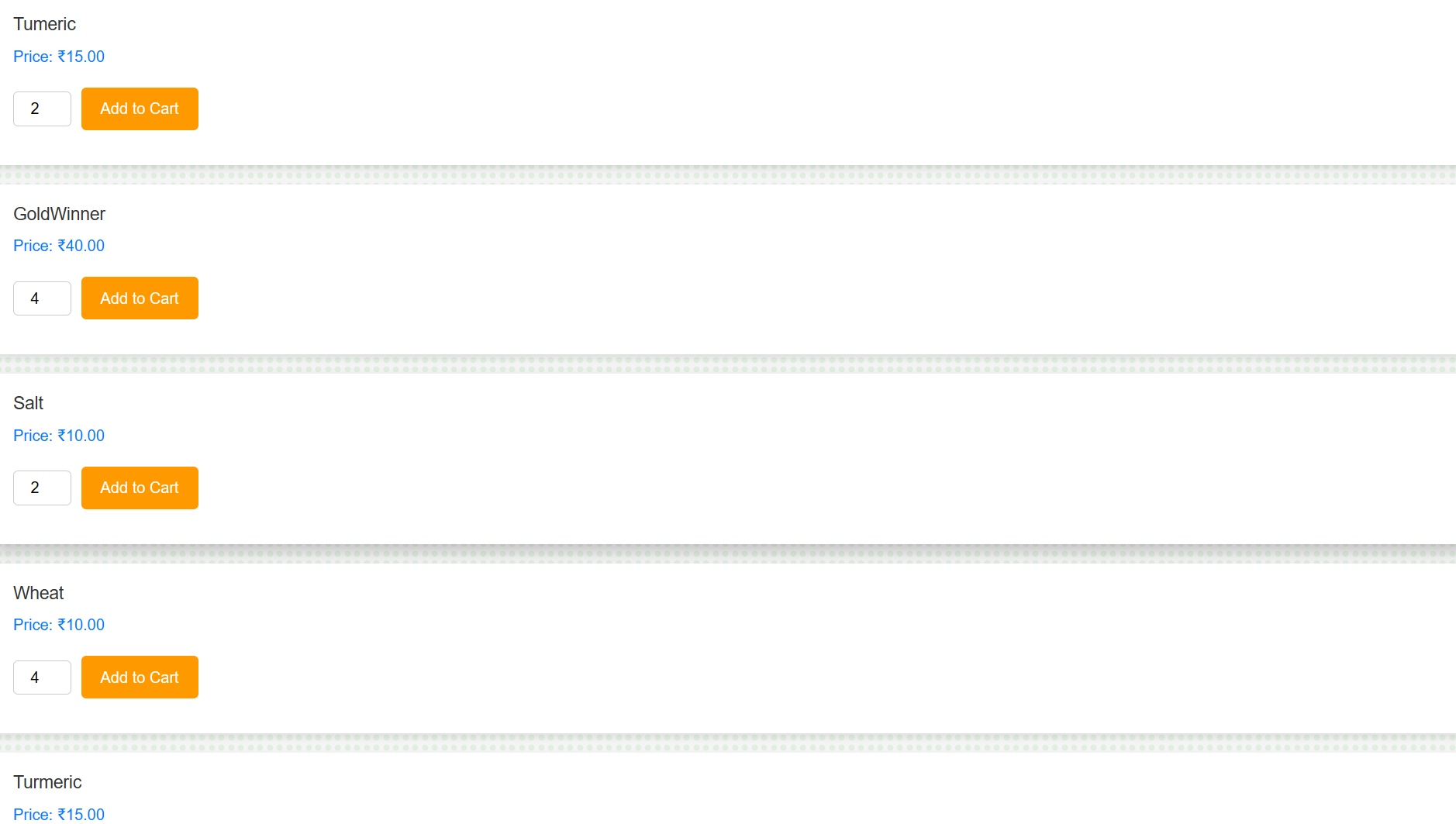
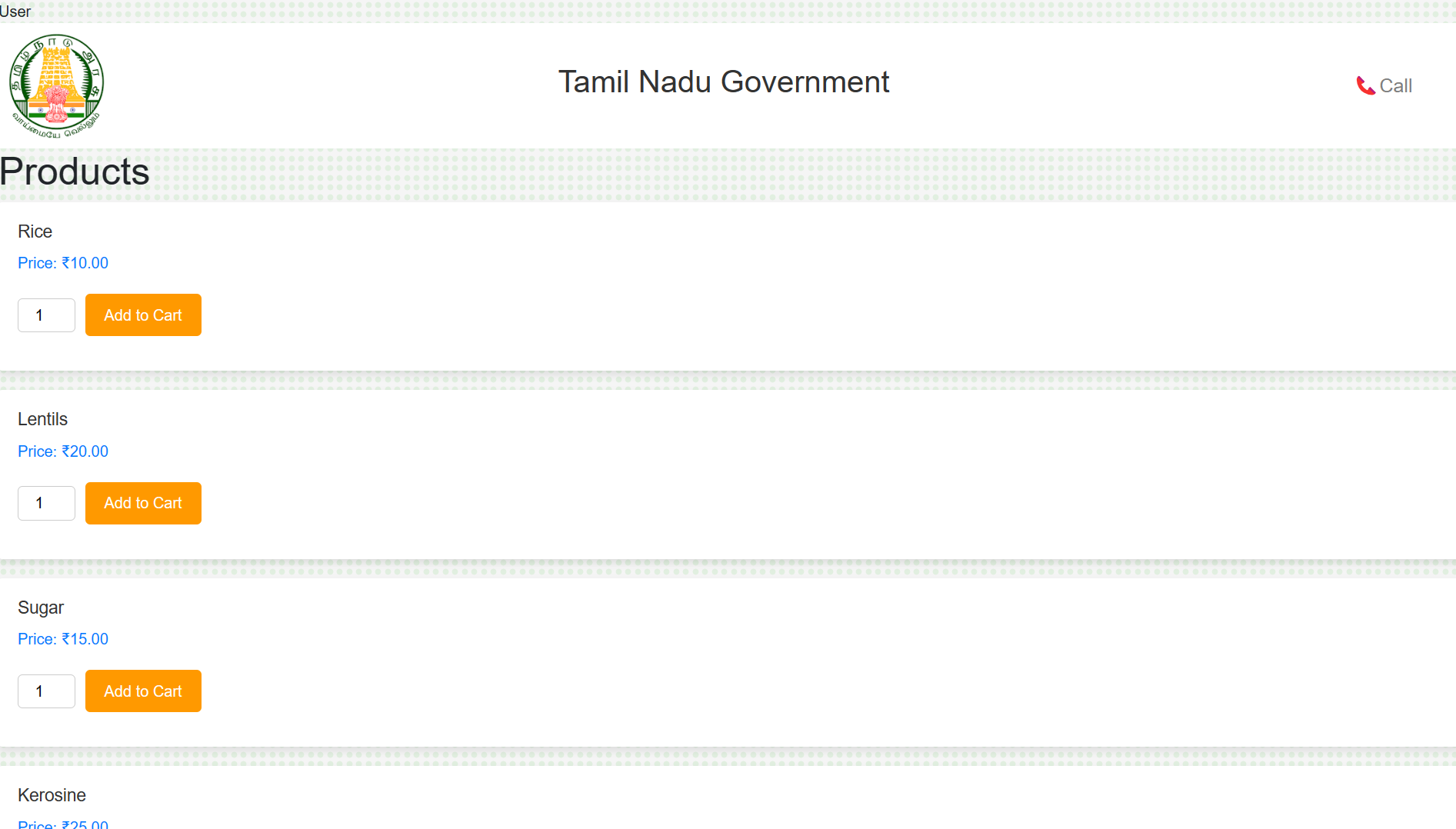
**RESULT**

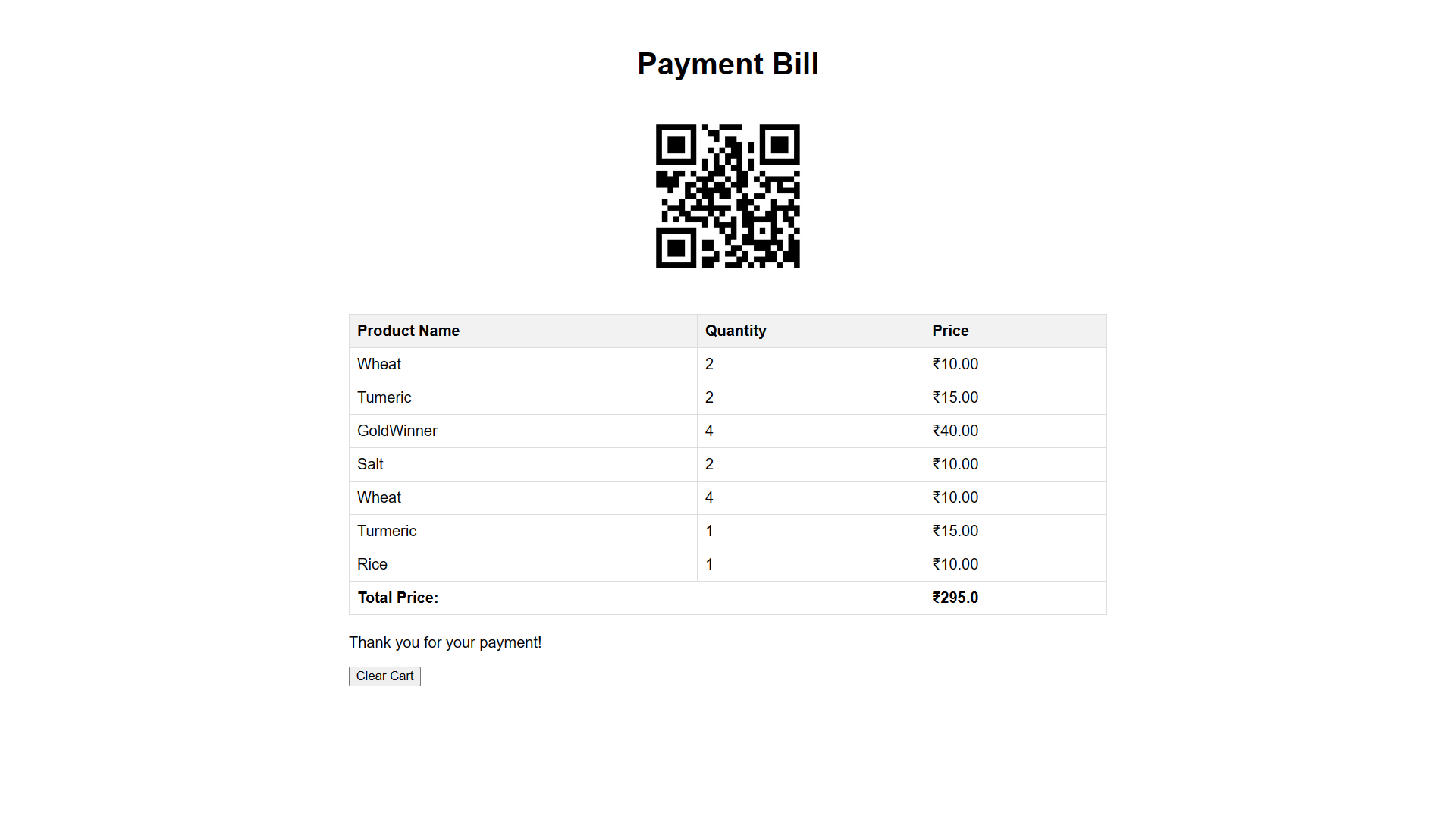
****

****

****

****



****

**CONCLUSION**

In conclusion, the E RATION PORTAL project aims to revolutionize the process of purchasing ration items by introducing an online billing system. The project encompasses various objectives, including the development of the portal, integration of digital payment methods, user testing, and training for ration shop employees. Key modules such as Flask Web Application, Database, Authentication, Product Management, Shopping Cart, and Payment have been identified to handle different aspects of the system. The architecture follows a model-view-controller (MVC) design pattern for modularity and separation of concerns. Requirements and analysis have been conducted, outlining user stories, hardware and software requirements, and technological considerations. Overall, the project seeks to provide a convenient, secure, and user-friendly platform for ration shopping, adhering to government regulations and ensuring compliance with data privacy standards.

Top of Form

**REFERENCES**

1. flask.pocoo.org, [Flask home](http://flask.pocoo.org/)
2. flask.pocoo.org, [Welcome to Flask (Docs):](http://flask.pocoo.org/docs/)
3. Full Stack Python, [Flask development](http://www.fullstackpython.com/flask.html)
4. Miguel Grinberg, [Flask mega-tutorial](http://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world)
5. Miguel Grinberg, [Flask book website](http://flaskbook.com/)
6. realpython.com, [Flask by example](https://realpython.com/blog/python/flask-by-example-part-1-project-setup/)
7. Michael Herman, Sean Vieira, [Python Web Applications With Flask. 1. Application setup](https://realpython.com/blog/python/python-web-applications-with-flask-part-i/)
8. Michael Herman, Sean Vieira, [- 2. Setup user accounts, Templates, Static files](https://realpython.com/blog/python/python-web-applications-with-flask-part-ii/)
9. Michael Herman, Sean Vieira, [- 3. Testing (unit and integration), Debugging, and Error handling](https://realpython.com/blog/python/python-web-applications-with-flask-part-iii/)
10. [Practical Flask Web Development Tutorials](https://www.youtube.com/playlist?list=PLQVvvaa0QuDc_owjTbIY4rbgXOFkUYOUB)
11. Randall Degges, [Build a Flask App in 30 Minutes](https://stormpath.com/blog/build-a-flask-app-in-30-minutes/)
12. Randall Degges, [Building websites in Python with Flask](http://maximebf.com/blog/2012/10/building-websites-in-python-with-flask/#.VdX1VHiNtBU)
13. [Instant Flask Web Development](https://www.packtpub.com/web-development/instant-flask-web-development-instant)
14. [Flask blog](https://github.com/dmaslov/flask-blog)
15. [How to make a Flask blog in one hour or less](http://charlesleifer.com/blog/how-to-make-a-flask-blog-in-one-hour-or-less/)
16. Miguel Grinberg, [RESTfull api with Python and Flask](http://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask)
17. [Developing RESTful Web APIs with Python, Flask and MongoDB](http://www.slideshare.net/nicolaiarocci/developing-restful-web-apis-with-python-flask-and-mongodb)
18. [REST web services with Python, MongoDB, and Spatial Data in the Cloud](https://blog.openshift.com/rest-web-services-with-python-mongodb-and-spatial-data-in-the-cloud-part-2/)
19. Adil Moujahid, [Interactive Data Visualization with D3.js, DC.js, Python, and MongoDB](http://adilmoujahid.com/posts/2015/01/interactive-data-visualization-d3-dc-python-mongodb/)
20. [Data visualization using D3.js and Flask](http://branetheory.org/2014/12/18/data-visualization-using-d3-js-and-flask/)
21. Ramiro Gómez, [Data Mining and Data Visualization Services](http://ramiro.org/)
22. [Flask, JSON and the Google Charts API](https://pythonspot.com/flask-json-and-the-google-charts-api/)
23. Randall Degges, [The Flask Authentication Problem](https://stormpath.com/blog/part-one-flask-authentication-problem/)
24. Miguel Grinberg, [RESTful Authentication with Flask](http://blog.miguelgrinberg.com/post/restful-authentication-with-flask)
25. pypi.python.org, [Flask-Login](https://pypi.python.org/pypi/Flask-Login)
26. Armin Ronacher, [Authentication snippets](http://flask.pocoo.org/snippets/category/authentication/)
27. [OAuth2 Python Example](https://github.com/reddit/reddit/wiki/OAuth2-Python-Example)
28. Flask-Bootstrap, [All Bootstrap, no boilerplate](http://pythonhosted.org/Flask-Bootstrap/)
29. [Write a Tumblelog Application with Flask and MongoEngine](http://docs.mongodb.org/ecosystem/tutorial/write-a-tumblelog-application-with-flask-mongoengine/)
30. Nicole White, [Building a Python Web Application Using Flask and Neo4j](http://neo4j.com/blog/building-python-web-application-using-flask-neo4j/)