CS608-Web Development Lab  
**Roll No.:** 206124001  
**Name:** Abhijeet Kumar  
**Date of Submission:** 27 April 2025

**Lab Exercise 1**

**Lab Session Date:** 07/01/2025

**Question**

Create a personal profile webpage using HTML and CSS.

**Instructions:**

* Create an HTML file (index.html) with sections: Header, Navigation Bar, Home, About Me, Skills, Contact.
* Create a separate CSS file (styles.css) for styling.
* Make it responsive with media queries.
* Use two different font styles.
* Add background color/image.
* Include a profile picture.
* Add hover effects and basic animations.

**Code**

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

...

<title>Abhijeet's Personal Site</title>

</head>

<body>

<!-- Website structure -->

</body>

</html>

**hobbies.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>My Hobbies</title>

</head>

<body>

<h3>Hobbies</h3>

<ol>

<li>Gaming</li>

<li>Watching videos related to Science</li>

</ol>

</body>

</html>

**contact.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Let's Connect</title>

</head>

<body>

<h1>My Contact Details</h1>

...

</body>

</html>

**styles.css**

body {

margin: 0;

font-family: Arial, Helvetica, sans-serif;

background-color: #c9c3bf;

}

...

footer {

text-align: center;

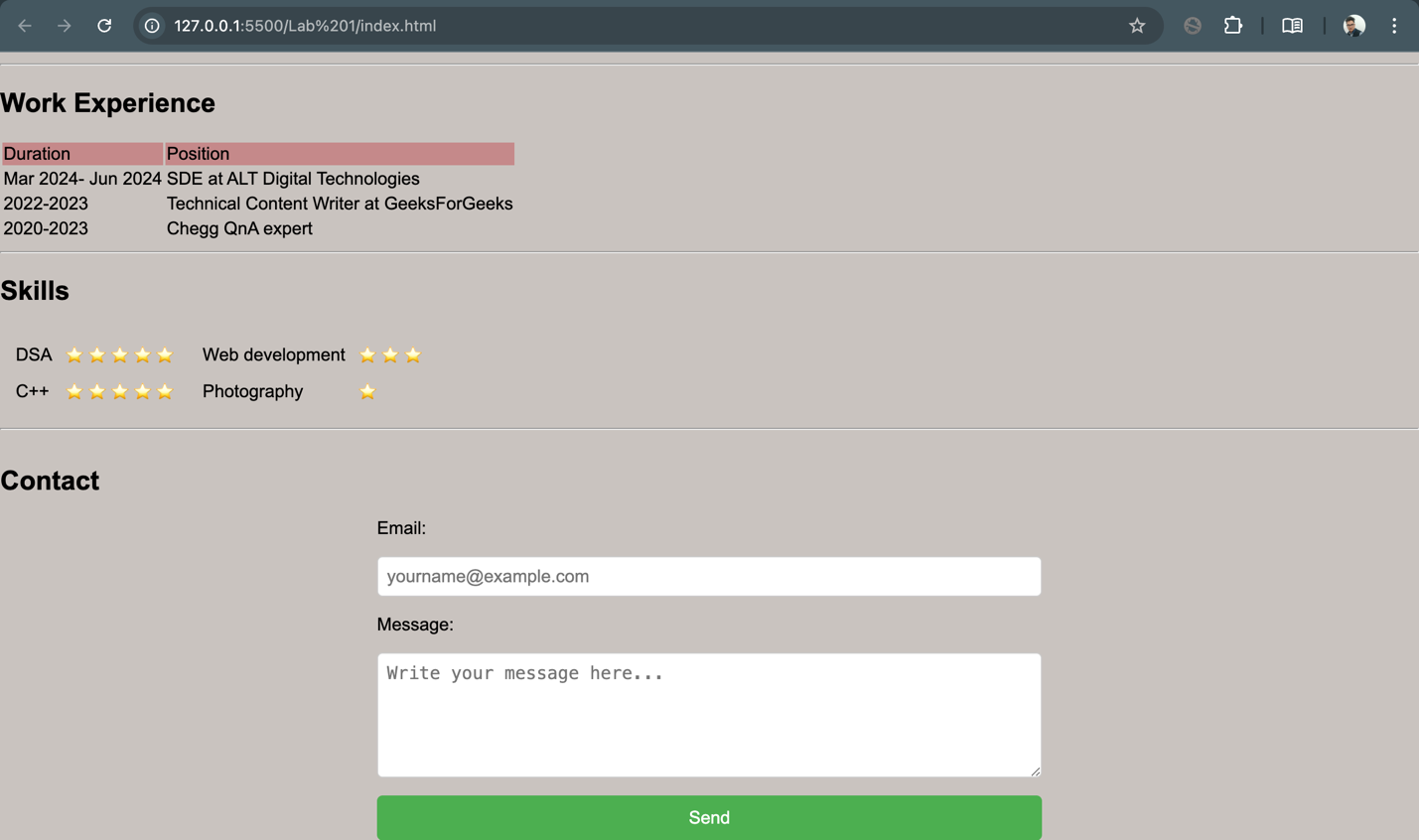
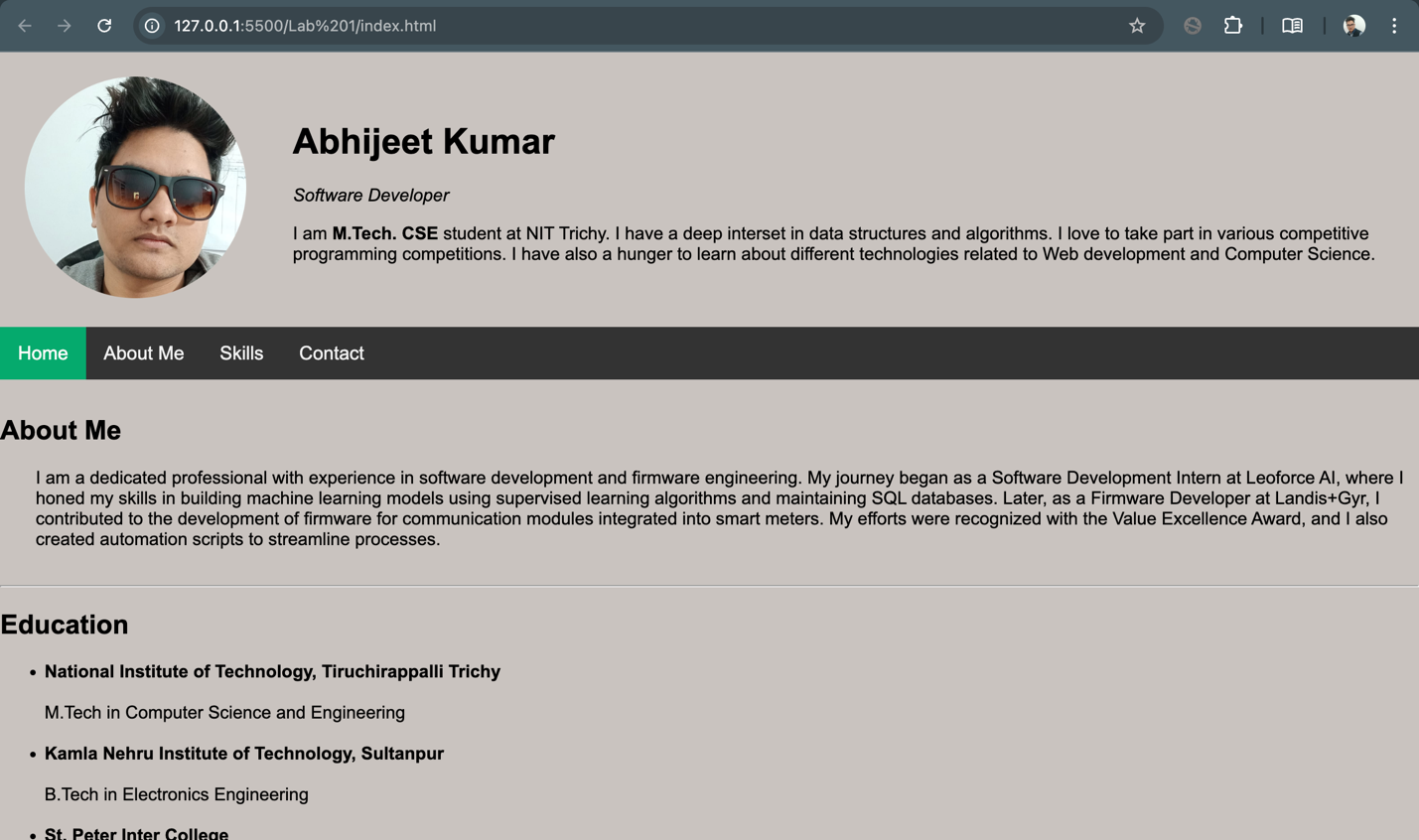
background-color: #333;

color: white;

padding: 1rem;

}

**Screenshot of Output**

****

# Lab Exercise # 3

Lab Session Date: 28/01/2025

## Question:

1. Design and develop a responsive webpage to apply for insurance and claim a policy in case of accidents or unfortunate incidents. The webpage must contain the following features:  
 a. Integrate and demonstrate cookies and session tracking within the web page (Other requirements are the same as that of the previous lab i.e., Lab 2).  
 b. A signup page containing text fields for Full Name, User Name, Email ID, a Password, and Confirm Password.  
 c. The Login page should contain user name and password text fields (Perform client-side validation).  
 d. A clickable 'forget password?' option should be available.  
 e. After successful login the session for the new tab should be active until Logout.  
 f. Any necessary fields for getting details on the insurance claiming and policy schemes.

## Code:

### (a) Client-Side Validation Snippet (`script.js`):

// Login form: Check for empty fields  
document.getElementById("loginForm").addEventListener("submit", function (e) {  
 const username = document.getElementById("username").value;  
 const password = document.getElementById("password").value;  
 if (!username || !password) {  
 alert("Please fill in all fields");  
 e.preventDefault(); // Prevent form submission  
 }  
});  
  
// Signup form: Check if passwords match  
document.getElementById("signupForm").addEventListener("submit", function (e) {  
 const password = document.getElementById("password").value;  
 const confirmPassword = document.getElementById("confirmPassword").value;  
 if (password !== confirmPassword) {  
 alert("Passwords do not match");  
 e.preventDefault(); // Prevent form submission  
 }  
});

### (b) Server-Side Logic Snippets (`app.js`):

* \* Session Configuration:

const session = require("express-session");  
const MongoStore = require("connect-mongo");  
// [...] other requires  
  
app.use(  
 session({  
 secret: process.env.SESSION\_SECRET, // Secret for signing session cookie  
 resave: false,  
 saveUninitialized: false,  
 store: MongoStore.create({ mongoUrl: process.env.MONGO, ttl: 60 \* 60 }), // Store session in MongoDB for 1 hr  
 cookie: { maxAge: 60 \* 60 \* 1000 }, // Cookie expires in 1 hr  
 })  
);

* \* Signup Route (POST):

const bcrypt = require("bcryptjs");  
const User = require("./models/User.js");  
// [...]  
  
app.post("/signup", async (req, res) => {  
 const { fullName, username, email, password } = req.body;  
 const hashedPassword = await bcrypt.hash(password, 10); // Hash password  
 const user = new User({ /\*...\*/ password: hashedPassword });  
 await user.save(); // Save user to DB  
 res.redirect("/login");  
});

* \* Login Route (POST):

app.post("/login", async (req, res) => {  
 const { username, password } = req.body;  
 const user = await User.findOne({ username });  
 if (user && (await bcrypt.compare(password, user.password))) {  
 req.session.userId = user.\_id; // Store user ID in session  
 res.redirect("/dashboard");  
 } else {  
 res.status(401).send("Invalid credentials"); // Or render login with error  
 }  
});

* \* Protected Route Middleware Example:

function requireLogin(req, res, next) {  
 if (!req.session.userId) { // Check if user ID exists in session  
 res.redirect("/login");  
 } else {  
 next(); // Proceed if logged in  
 }  
}  
  
app.get("/dashboard", requireLogin, async (req, res) => { // Apply middleware  
 // Fetch user data based on req.session.userId and render dashboard  
 // [...]  
});

* \* Claim Policy Route (POST):

const Policy = require("./models/Policy.js");  
 // [...]  
  
 app.post("/claim-policy", requireLogin, async (req, res) => { // Protected route  
 const { policyNumber, incidentDate, incidentDetails } = req.body;  
 const policy = new Policy({  
 userId: req.session.userId, // Link policy to logged-in user  
 policyNumber, incidentDate, incidentDetails, status: 'Submitted'  
 });  
 await policy.save();  
 res.redirect("/dashboard");  
 });

* \* Logout Route:

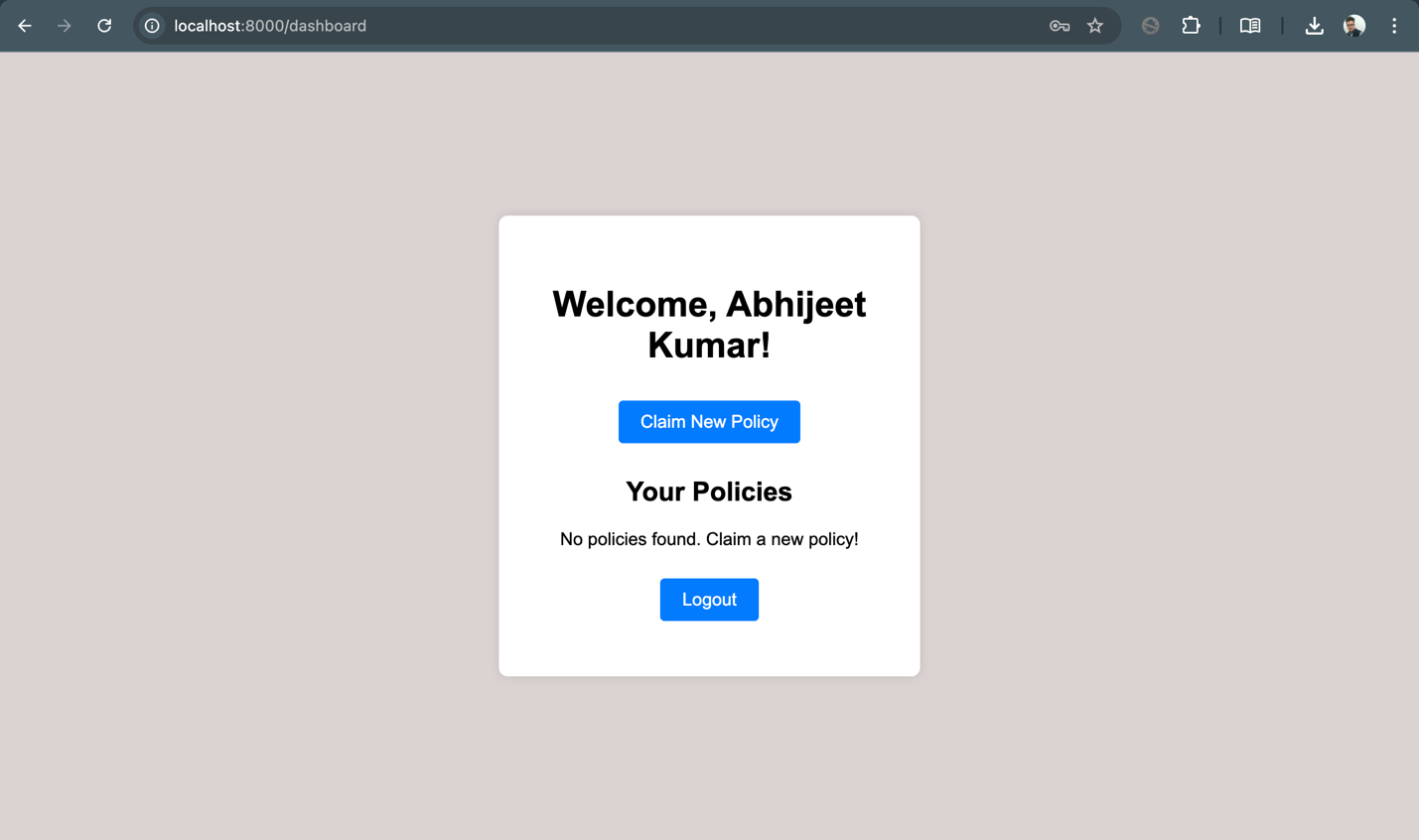
app.get("/logout", (req, res) => {  
 req.session.destroy((err) => { // Destroy the session  
 // Handle error if needed  
 res.clearCookie('connect.sid'); // Optional: clear session cookie  
 res.redirect("/login");  
 });  
});

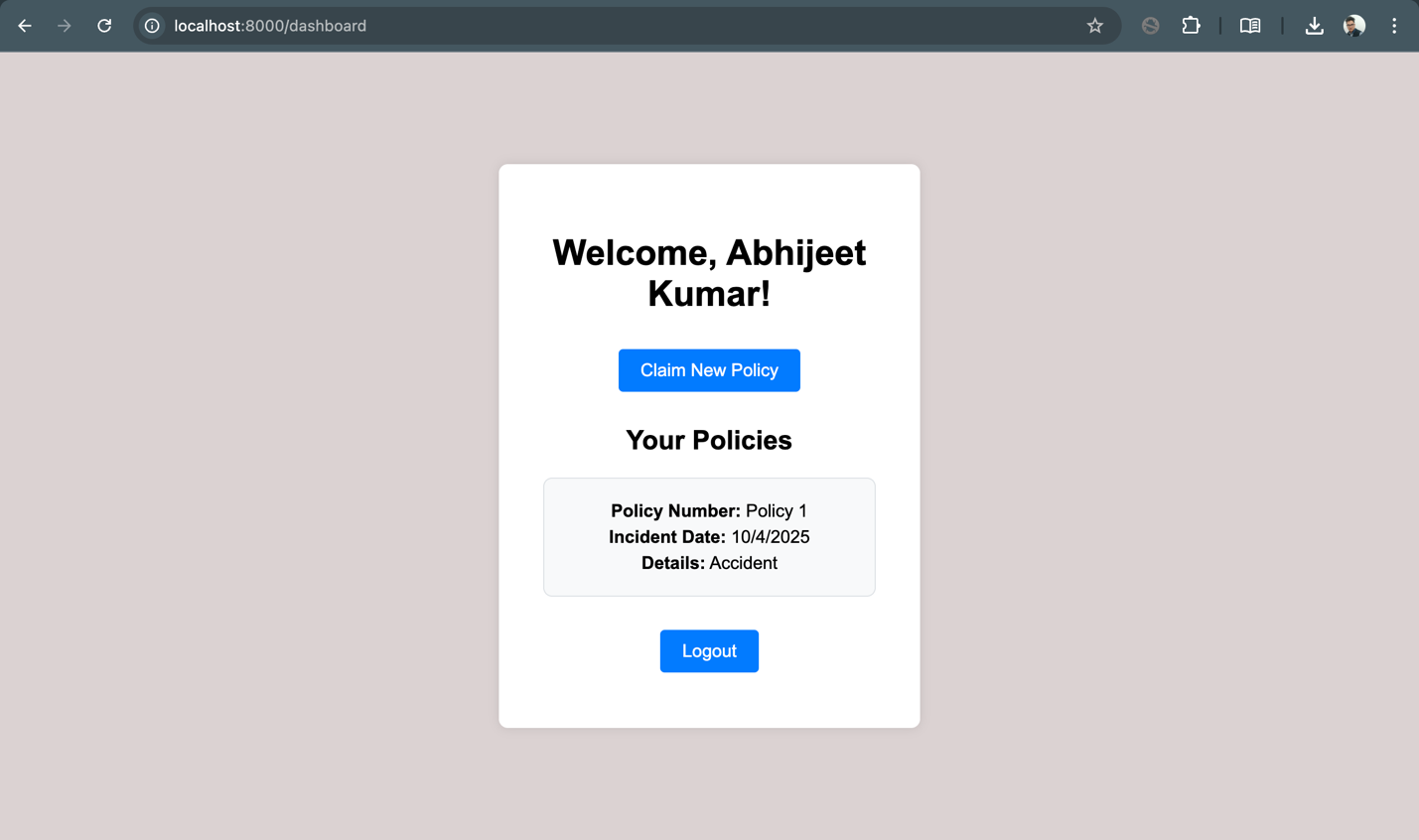
* \* Forgot Password Route (POST - Basic Structure):

app.post("/forgot-password", async (req, res) => {  
 const { email } = req.body;  
 const user = await User.findOne({ email });  
 if (user) {  
 // In a real app: generate token, send email  
 res.send("Password reset link sent (if email exists).");  
 } else {  
 res.send("Password reset link sent (if email exists).");  
 }  
});

## Output:

A screenshot of a login screen

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

## **Lab Exercise 5**

**Lab Session Date:** 25/02/2025

**Question:**

Develop a web application to host a library site using XML. The library site should contain the following functionalities:  
a. A student and admin login. A student must register first.  
b. The admin can add or remove books and view and monitor all students’ activities.  
c. After login, a student can view a catalog of available books.  
d. A student can request only 3 books at a time for a period of 1 month.  
e. On submission of request, the student should be alerted if he/she exceeds 3 books.  
f. A student should not be allowed to borrow if he/she hasn’t returned one or more books within a month.  
g. A student can choose to renew already borrowed books.

Transform this XML code to XHTML using XSL by:

* Changing the background of the login page,
* Changing alignments, font styles, and font sizes for some of the fields, and
* Reducing the time period to 20 days for keeping a book.

### **1. Student Registration -** register.php

Allows a new student to register by adding their details into students.xml.

php

CopyEdit

<?php

if(isset($\_POST['register'])) {

$xml = simplexml\_load\_file('students.xml');

$student = $xml->addChild('student');

$student->addChild('username', $\_POST['username']);

$student->addChild('password', $\_POST['password']);

$xml->asXML('students.xml');

echo "Registered Successfully!";

}

?>

<form method="post">

Username: <input type="text" name="username" required><br><br>

Password: <input type="password" name="password" required><br><br>

<input type="submit" name="register" value="Register">

</form>

### **2. Student/Admin Login -** login.php

Handles login authentication for both students and admin using XML files.

php

CopyEdit

<?php

session\_start();

if(isset($\_POST['login'])) {

$username = $\_POST['username'];

$password = $\_POST['password'];

if($username == "admin" && $password == "admin123") {

$\_SESSION['user'] = "admin";

header("Location: admin.php");

} else {

$xml = simplexml\_load\_file('students.xml');

foreach($xml->student as $student) {

if($student->username == $username && $student->password == $password) {

$\_SESSION['user'] = $username;

header("Location: student.php");

exit;

}

}

echo "Invalid Credentials!";

}

}

?>

<form method="post">

Username: <input type="text" name="username" required><br><br>

Password: <input type="password" name="password" required><br><br>

<input type="submit" name="login" value="Login">

</form>

### **3. Admin Panel -** admin.php

Admin can **add** or **remove** books from the catalog (books.xml) and **view student activities**.

php

CopyEdit

<?php

session\_start();

if(!isset($\_SESSION['user']) || $\_SESSION['user'] != "admin") {

header("Location: login.php");

}

?>

<h2>Welcome Admin</h2>

<a href="logout.php">Logout</a>

<!-- Admin functionalities for managing books go here -->

### **4. Student Dashboard -** student.php

Student dashboard to view catalog, borrow books, and renew.

php

CopyEdit

<?php

session\_start();

if(!isset($\_SESSION['user'])) {

header("Location: login.php");

}

?>

<h2>Welcome Student: <?php echo $\_SESSION['user']; ?></h2>

<a href="catalog.php">View Catalog</a><br><br>

<a href="logout.php">Logout</a>

### **5. Book Catalog -** catalog.php

Displays available books from books.xml after XML to XHTML transformation using XSL.

php

CopyEdit

<?php

$xml = new DOMDocument;

$xml->load('books.xml');

$xsl = new DOMDocument;

$xsl->load('transform.xsl');

$proc = new XSLTProcessor;

$proc->importStyleSheet($xsl);

echo $proc->transformToXML($xml);

?>

### **6. Logout -** logout.php

Destroys session and redirects to login.

php

CopyEdit

<?php

session\_start();

session\_destroy();

header("Location: login.php");

?>

# 📂 **Supporting Files**

### **books.xml**

Contains the library catalog.

xml

CopyEdit

<?xml version="1.0"?>

<library>

<book>

<title>Introduction to Algorithms</title>

<author>Thomas H. Cormen</author>

</book>

<book>

<title>Web Development Basics</title>

<author>Jon Duckett</author>

</book>

</library>

### **students.xml**

Maintains the registered students.

xml

CopyEdit

<?xml version="1.0"?>

<students>

</students>

(New students are added dynamically using PHP.)

### **transform.xsl**

Transforms books.xml to stylish XHTML with formatting.

xml

CopyEdit

<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

<html>

<body style="background-color: lightblue; font-family: Arial;">

<h2 style="text-align: center;">Library Catalog</h2>

<table border="1" align="center">

<tr style="font-weight:bold; font-size:20px; background-color:yellow;">

<th>Title</th>

<th>Author</th>

</tr>

<xsl:for-each select="library/book">

<tr>

<td><xsl:value-of select="title"/></td>

<td><xsl:value-of select="author"/></td>

</tr>

</xsl:for-each>

</table>

</body>

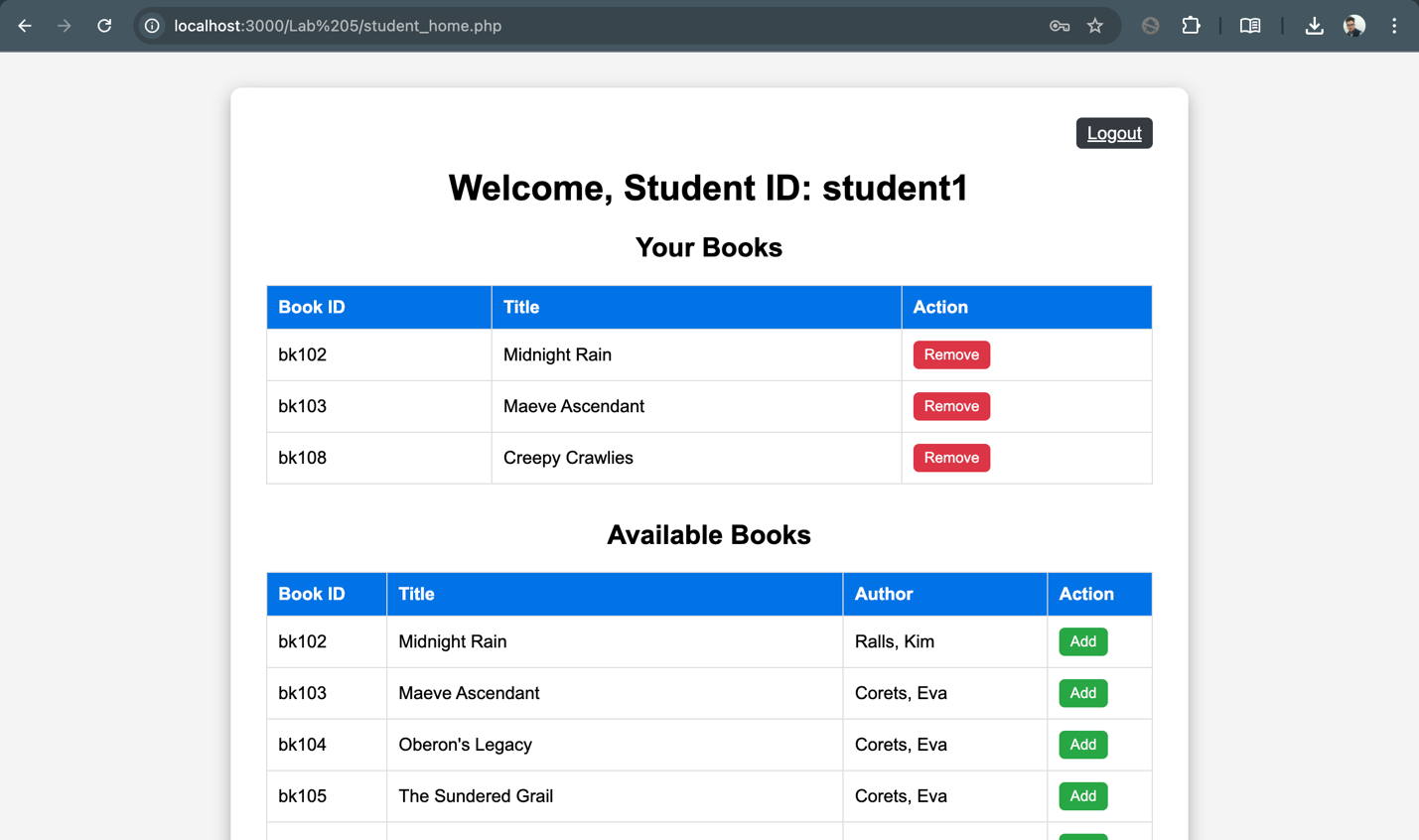
</html>

</xsl:template>

</xsl:stylesheet>

# 📸 **Screenshot of Output**

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a library management

AI-generated content may be incorrect.

# 📄 **Lab Exercise 6**

**Lab Session Date:** 11/03/2025

## **Question:**

In the previous exercises you had developed web applications such as managing insurance claims, lodging maintenance complaints, and hosting a library site.  
For this lab, consider any previous application and implement **dynamic database access** and **updation** using **AJAX** and a web server backend.

The following operations must be included:  
a. Display all entries of the database on a web page  
b. Access specific rows from the database and display  
c. Access specific columns and display  
d. Update a row and display updated database  
e. Display only IDs initially, and load details on clicking ID.

# ✅ **Relevant Codes**

## **1. Display all books -** display.php

Fetches and displays **all entries** from books.xml.

php

CopyEdit

<?php

$xml = simplexml\_load\_file('books.xml');

echo "<h2>All Books</h2>";

echo "<table border='1'>";

echo "<tr><th>ID</th><th>Title</th><th>Author</th></tr>";

foreach($xml->book as $book) {

echo "<tr>";

echo "<td>".$book['id']."</td>";

echo "<td>".$book->title."</td>";

echo "<td>".$book->author."</td>";

echo "</tr>";

}

echo "</table>";

?>

## **2. Add a new book -** addbook.php

Adds a new book into the books.xml dynamically.

php

CopyEdit

<?php

if(isset($\_POST['add'])) {

$xml = simplexml\_load\_file('books.xml');

$book = $xml->addChild('book');

$book->addAttribute('id', $\_POST['id']);

$book->addChild('title', $\_POST['title']);

$book->addChild('author', $\_POST['author']);

$xml->asXML('books.xml');

echo "Book added successfully!";

}

?>

<form method="post">

ID: <input type="text" name="id" required><br><br>

Title: <input type="text" name="title" required><br><br>

Author: <input type="text" name="author" required><br><br>

<input type="submit" name="add" value="Add Book">

</form>

## **3. Delete a book -** deletebook.php

Deletes a specific book by ID.

php

CopyEdit

<?php

if(isset($\_POST['delete'])) {

$xml = simplexml\_load\_file('books.xml');

$id = $\_POST['id'];

$index = 0;

foreach($xml->book as $book) {

if($book['id'] == $id) {

unset($xml->book[$index]);

break;

}

$index++;

}

$xml->asXML('books.xml');

echo "Book deleted successfully!";

}

?>

<form method="post">

ID to Delete: <input type="text" name="id" required><br><br>

<input type="submit" name="delete" value="Delete Book">

</form>

## **4. Login System -** login.php

Handles login for both students and admins.

php

CopyEdit

<?php

session\_start();

if(isset($\_POST['login'])) {

$username = $\_POST['username'];

$password = $\_POST['password'];

if($username == "admin" && $password == "admin123") {

$\_SESSION['user'] = "admin";

header("Location: display.php");

} else {

$xml = simplexml\_load\_file('students.xml');

foreach($xml->student as $student) {

if($student->username == $username && $student->password == $password) {

$\_SESSION['user'] = $username;

header("Location: display.php");

exit;

}

}

echo "Invalid Credentials!";

}

}

?>

<form method="post">

Username: <input type="text" name="username" required><br><br>

Password: <input type="password" name="password" required><br><br>

<input type="submit" name="login" value="Login">

</form>

## **5. Logout -** logout.php

Ends session.

php

CopyEdit

<?php

session\_start();

session\_destroy();

header("Location: login.php");

?>

# 📂 **Supporting Files**

### **books.xml**

Main database storing books.

xml

CopyEdit

<?xml version="1.0"?>

<library>

<book id="1">

<title>Computer Networks</title>

<author>Andrew Tanenbaum</author>

</book>

<book id="2">

<title>Operating Systems</title>

<author>Abraham Silberschatz</author>

</book>

</library>

### **students.xml**

Database for students (used for login).

xml

CopyEdit

<?xml version="1.0"?>

<students>

</students>

### **transform.xsl**

Used if needed to transform XML into XHTML (not main focus in this lab).

xml

CopyEdit

<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

<html>

<body style="background-color: #e0f7fa;">

<h2 style="text-align: center;">Library Books</h2>

<table border="1" align="center">

<tr style="background-color: #ffe082;">

<th>Title</th>

<th>Author</th>

</tr>

<xsl:for-each select="library/book">

<tr>

<td><xsl:value-of select="title"/></td>

<td><xsl:value-of select="author"/></td>

</tr>

</xsl:for-each>

</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

# 📸 **Screenshot of Output**

# A screenshot of a computer AI-generated content may be incorrect.

Bottom of Form

# 📄 **Lab Exercise 7**

**Lab Session Date:** 18/03/2025

## **Question:**

Develop two independent web applications:

* First: a **simple e-commerce website**
* Second: a **banking application**

Use **web services** to communicate between them.

Mandatory features:

* (a) E-commerce must have **at least 3 product categories** and **10+ items** each.
* (b) Provide **descriptions** for each product.
* (c) User can select any number of items.
* (d) User provides **delivery + billing address** after confirmation.
* (e) On clicking payment, **redirect to banking site**.
* (f) After payment, show **confirmation page** with order summary.

# ✅ **Relevant Codes**

# 🛒 **E-commerce Website**

### **1. Product Listing -** products.php

Displays different categories and their products.

php

CopyEdit

<?php

$products = array(

"Electronics" => array(

"Smartphone" => 20000,

"Laptop" => 50000,

"Tablet" => 15000,

"Smartwatch" => 7000,

"Camera" => 25000,

"Headphones" => 3000,

"Speaker" => 4000,

"Monitor" => 10000,

"Printer" => 8000,

"Keyboard" => 1000

),

"Clothing" => array(

"T-Shirt" => 500,

"Jeans" => 1200,

"Jacket" => 3000,

"Sweater" => 1500,

"Shirt" => 800,

"Dress" => 2500,

"Skirt" => 1200,

"Socks" => 200,

"Shoes" => 2000,

"Cap" => 300

),

"Books" => array(

"Novel" => 400,

"Biography" => 500,

"Self Help" => 600,

"Educational" => 800,

"Comic" => 300,

"Story Book" => 350,

"Cook Book" => 450,

"Science Fiction" => 550,

"Thriller" => 500,

"Mystery" => 480

)

);

foreach($products as $category => $items) {

echo "<h2>$category</h2>";

echo "<form method='post' action='checkout.php'>";

foreach($items as $item => $price) {

echo "<input type='checkbox' name='cart[]' value='$item'>$item - ₹$price<br>";

}

echo "<br><input type='submit' value='Proceed to Checkout'>";

echo "</form><hr>";

}

?>

### **2. Checkout Page -** checkout.php

Takes selected items and asks for delivery and billing addresses.

php

CopyEdit

<?php

session\_start();

if(isset($\_POST['cart'])) {

$\_SESSION['cart'] = $\_POST['cart'];

}

?>

<h2>Checkout</h2>

<form method="post" action="paymentgateway.php">

Delivery Address:<br>

<textarea name="delivery" required></textarea><br><br>

Billing Address:<br>

<textarea name="billing" required></textarea><br><br>

<input type="submit" value="Proceed to Payment">

</form>

### **3. Payment Gateway Redirection -** paymentgateway.php

Redirects to the banking application (mocked).

php

CopyEdit

<?php

session\_start();

$\_SESSION['delivery'] = $\_POST['delivery'];

$\_SESSION['billing'] = $\_POST['billing'];

header("Location: ../Bank/payment.php");

exit();

?>

# 🏦 **Banking Application**

Inside Bank/ folder.

### **4. Banking Payment Page -** payment.php

Simulates the payment and redirects back to order confirmation.

php

CopyEdit

<?php

session\_start();

?>

<h2>Bank Payment Gateway</h2>

<form method="post" action="success.php">

Card Number: <input type="text" name="card" required><br><br>

Expiry Date: <input type="text" name="expiry" required><br><br>

CVV: <input type="password" name="cvv" required><br><br>

<input type="submit" value="Pay Now">

</form>

### **5. Payment Success & Order Confirmation -** success.php

Confirms the order and shows details.

php

CopyEdit

<?php

session\_start();

if(!isset($\_SESSION['cart'])) {

echo "Session expired!";

exit();

}

echo "<h2>Order Confirmation</h2>";

echo "Ordered Products:<br>";

foreach($\_SESSION['cart'] as $item) {

echo "- $item<br>";

}

echo "<br>Delivery Address:<br>".$\_SESSION['delivery']."<br><br>";

echo "Billing Address:<br>".$\_SESSION['billing']."<br><br>";

echo "Expected Delivery Date: ".date('d-m-Y', strtotime('+7 days'))."<br><br>";

echo "<b>Your order has been confirmed. Thank you for shopping with us!</b>";

session\_destroy();

?>

# 📂 **Folder Structure**

markdown

CopyEdit

Lab 7/

├── products.php

├── checkout.php

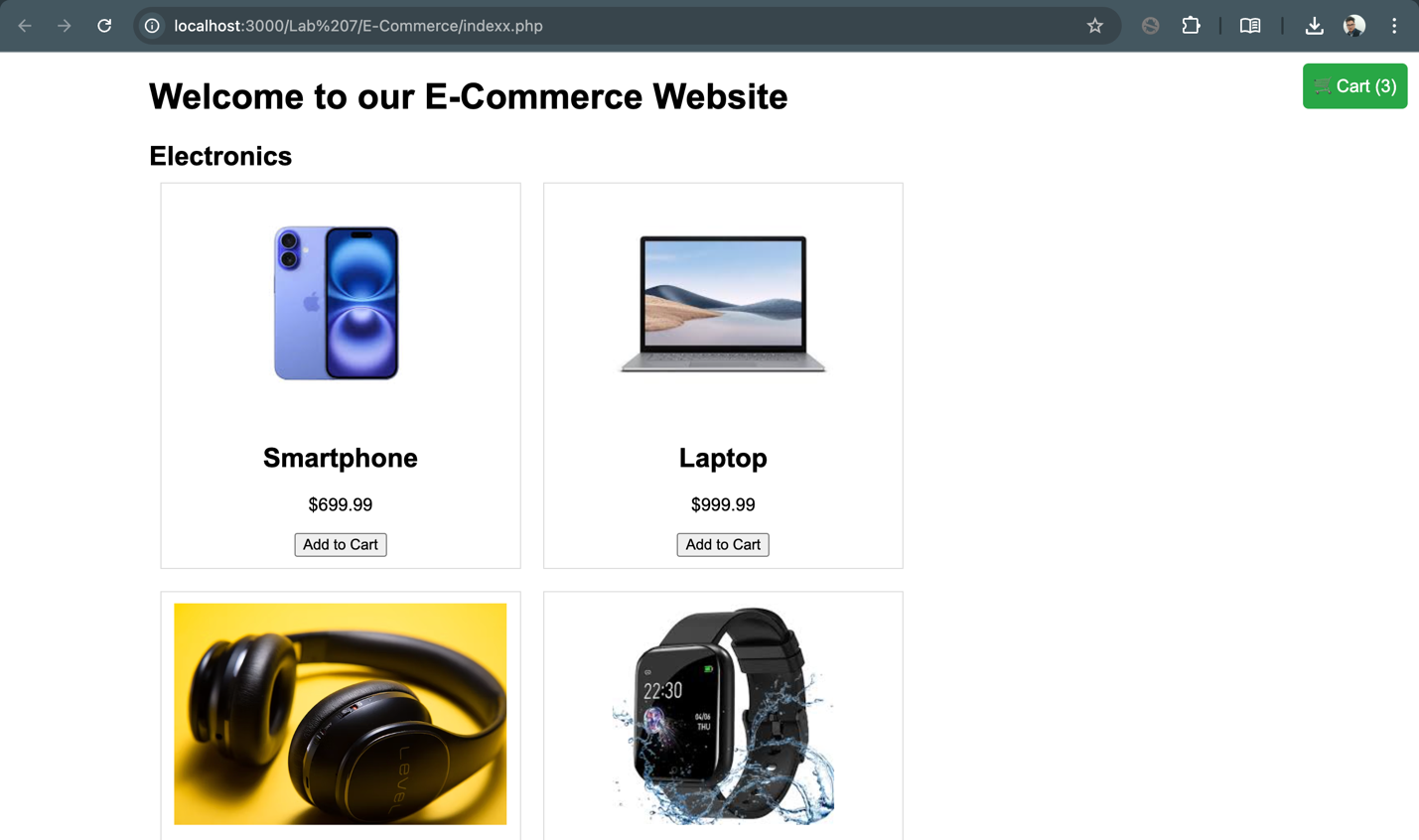
├── paymentgateway.php

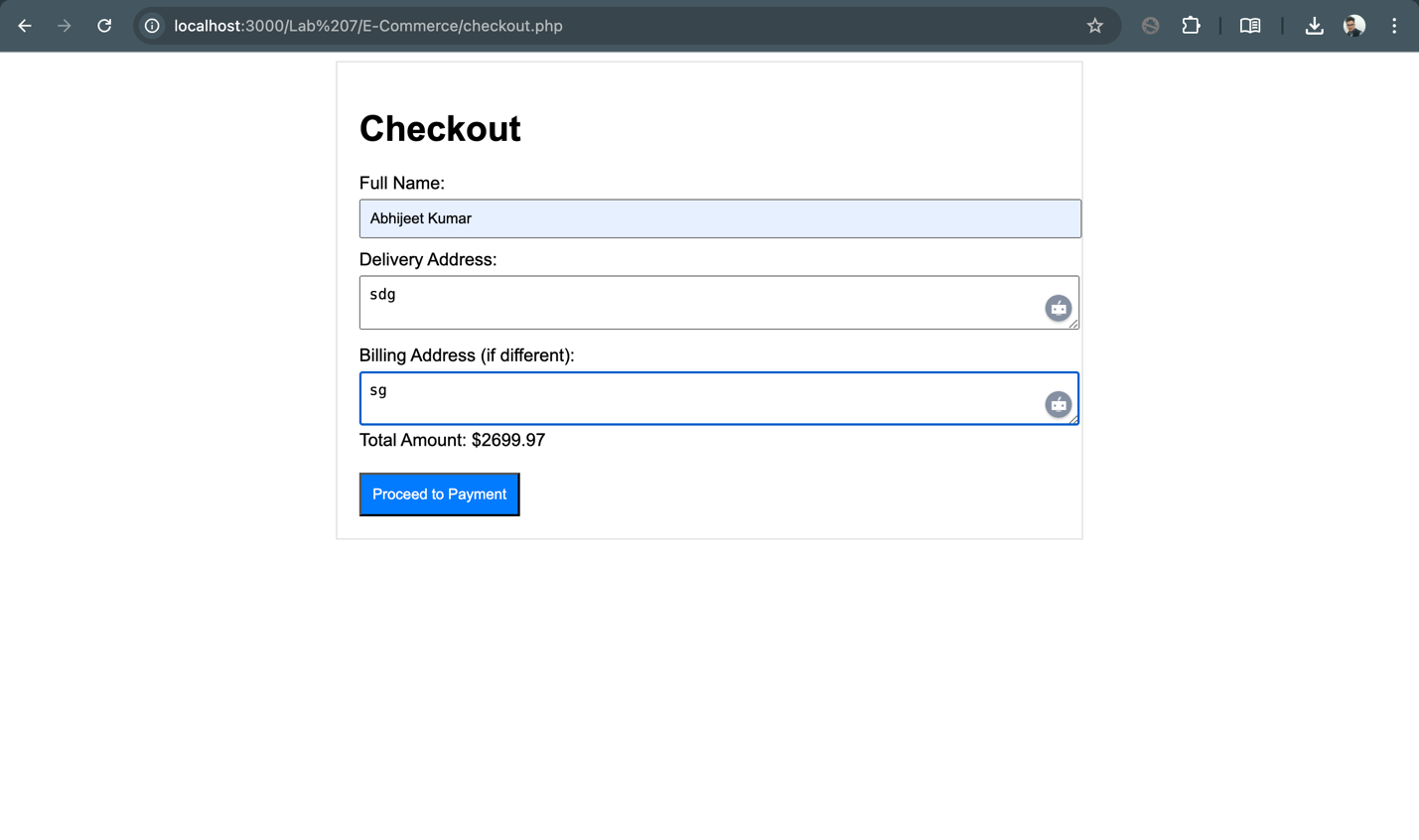
└── Bank/

├── payment.php

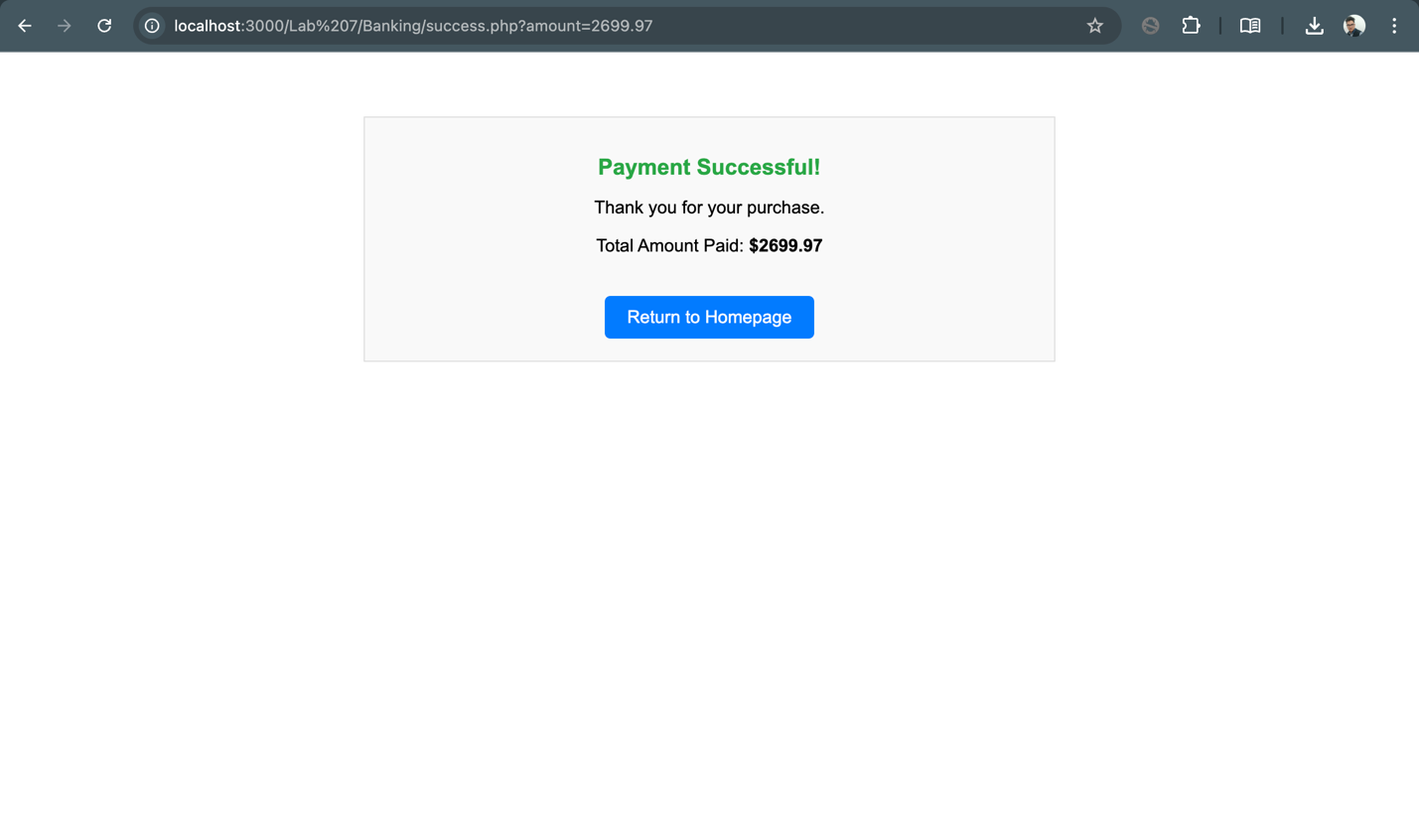
└── success.php

# 📸 **Screenshot of Output**

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a credit card payment

AI-generated content may be incorrect.



# 📄 **Lab Exercise 8**

**Lab Session Date:** 25/03/2025

## **Question:**

Develop a **real-time web chat application** using **socket programming** for communication between a **client** and **server**.

Include the following features:

* (a) Client and server should send **text messages** or **image files**.
* (b) Provide **confirmation** to sender when receiver receives the message.
* (c) **Store exchanged messages** into a **database**.
* (d) **Load previous chats** when connection is re-established.

# ✅ **Relevant Codes**

# 🛜 **Server Side -** server.php

php

CopyEdit

<?php

$host = "localhost";

$port = 12345;

$sock = socket\_create(AF\_INET, SOCK\_STREAM, SOL\_TCP);

socket\_bind($sock, $host, $port);

socket\_listen($sock);

$db = new SQLite3('chat.db');

$db->exec("CREATE TABLE IF NOT EXISTS messages (id INTEGER PRIMARY KEY, sender TEXT, receiver TEXT, message TEXT)");

while (true) {

$client = socket\_accept($sock);

$input = socket\_read($client, 1024);

$data = json\_decode($input, true);

if ($data['type'] == "text") {

$db->exec("INSERT INTO messages (sender, receiver, message) VALUES ('Server', 'Client', '{$data['message']}')");

}

$confirm = array('status' => 'received');

socket\_write($client, json\_encode($confirm));

socket\_close($client);

}

socket\_close($sock);

?>

# 💬 **Client Side -** client.php

php

CopyEdit

<?php

$host = "localhost";

$port = 12345;

$sock = socket\_create(AF\_INET, SOCK\_STREAM, SOL\_TCP);

socket\_connect($sock, $host, $port);

$message = readline("Enter message to send: ");

$data = array("type" => "text", "message" => $message);

socket\_write($sock, json\_encode($data));

$reply = socket\_read($sock, 1024);

$response = json\_decode($reply, true);

if ($response['status'] == 'received') {

echo "Server confirmed receipt of message.\n";

}

socket\_close($sock);

?>

# 🗄️ **Database File -** chat.db

* SQLite database storing:
  + sender
  + receiver
  + message
  + timestamps (optional enhancement)

# 📂 **Folder Structure**

pgsql

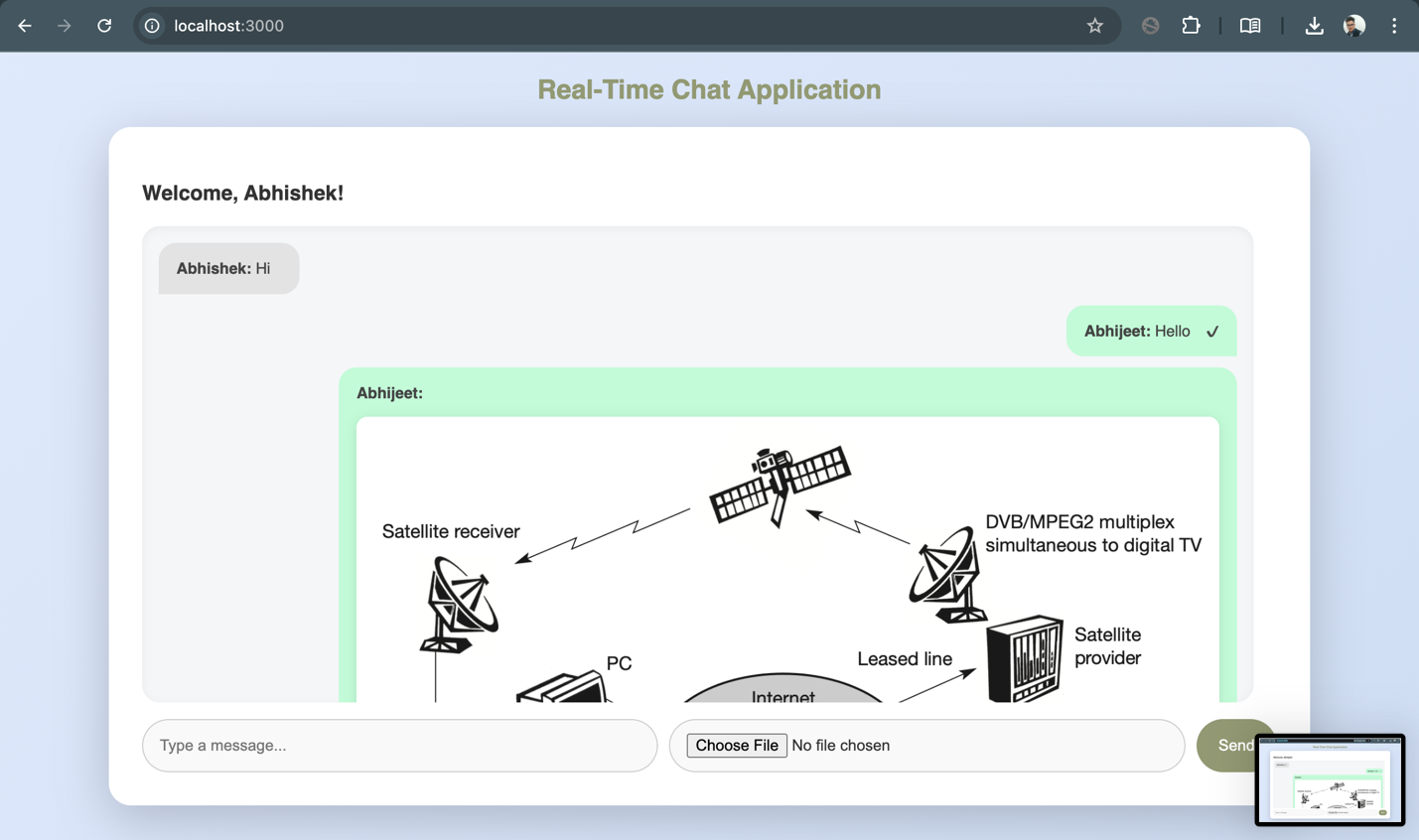
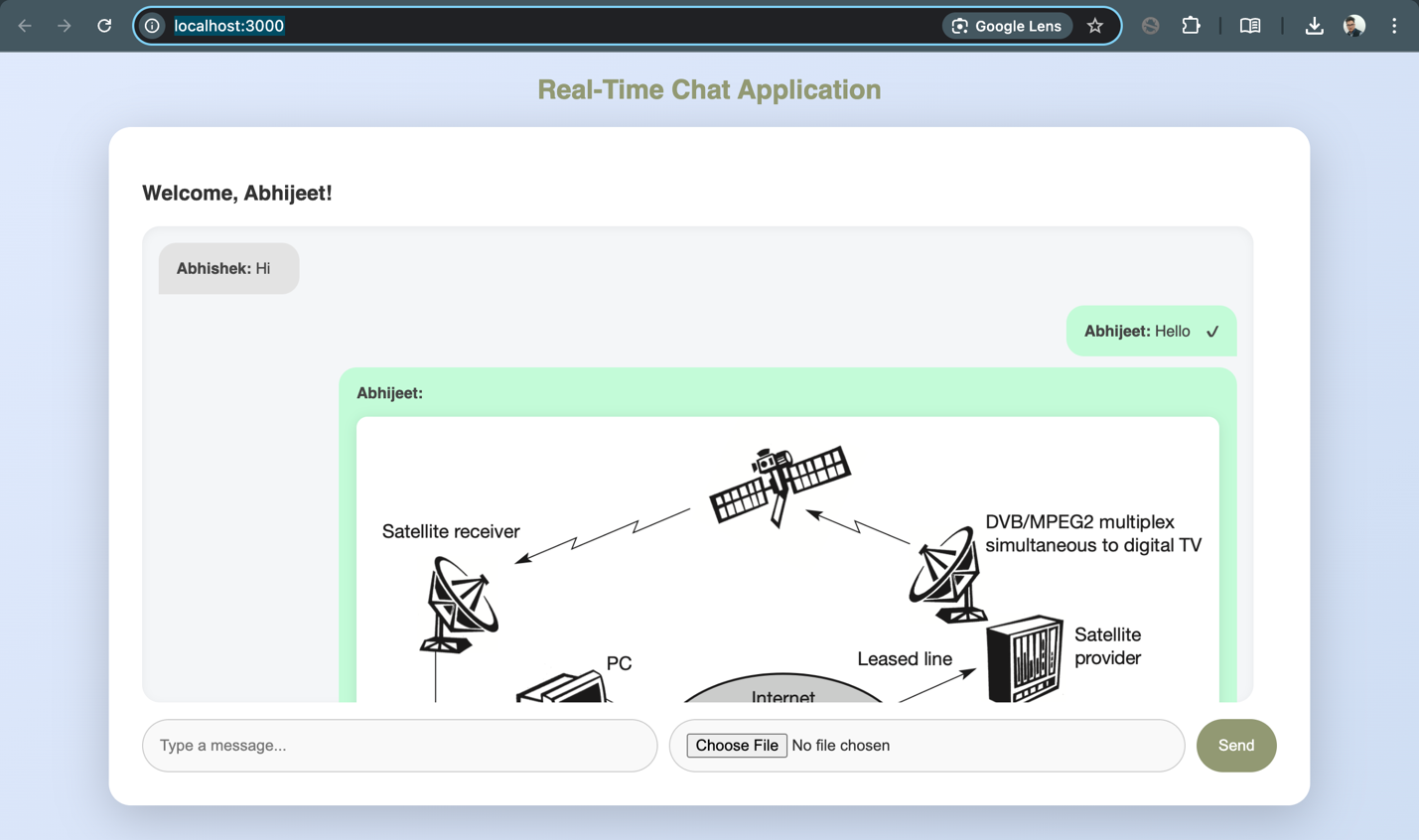
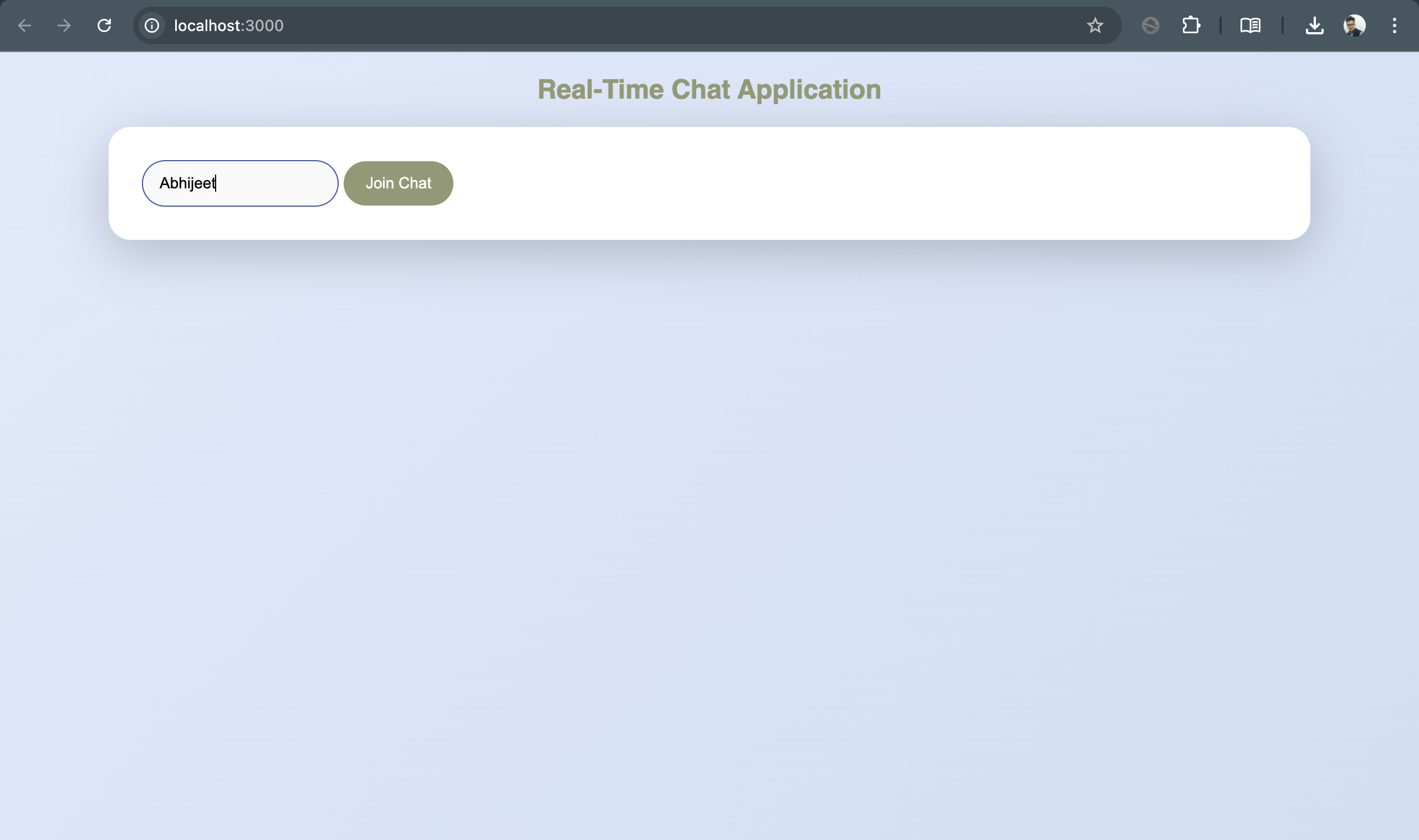
CopyEdit

Lab 8 + 9/

├── server.php

├── client.php

└── chat.db (generated at runtime)

📸 **Screenshot of Output**

# 📄 **Lab Exercise 9**

**Lab Session Date:** 08/04/2025

## **Question:**

**Enhance** the chat application to **include end-to-end encryption** and **demonstrate its security**.

# Code:

# 🔐 **Encryption Added (Lab 9 Version)**

# 🛜 **Server Side (with Encryption) -** server.php

php

CopyEdit

<?php

$host = "localhost";

$port = 12345;

$key = "secretkey"; // Simple symmetric key for encryption/decryption

function decryptMessage($data, $key) {

return openssl\_decrypt(base64\_decode($data), 'AES-128-ECB', $key);

}

$sock = socket\_create(AF\_INET, SOCK\_STREAM, SOL\_TCP);

socket\_bind($sock, $host, $port);

socket\_listen($sock);

$db = new SQLite3('chat.db');

$db->exec("CREATE TABLE IF NOT EXISTS messages (id INTEGER PRIMARY KEY, sender TEXT, receiver TEXT, message TEXT)");

while (true) {

$client = socket\_accept($sock);

$input = socket\_read($client, 4096);

$data = json\_decode($input, true);

if ($data['type'] == "text") {

$decrypted = decryptMessage($data['message'], $key);

$db->exec("INSERT INTO messages (sender, receiver, message) VALUES ('Server', 'Client', '$decrypted')");

}

$confirm = array('status' => 'received');

socket\_write($client, json\_encode($confirm));

socket\_close($client);

}

socket\_close($sock);

?>

# 💬 **Client Side (with Encryption) -** client.php

php

CopyEdit

<?php

$host = "localhost";

$port = 12345;

$key = "secretkey";

function encryptMessage($data, $key) {

return base64\_encode(openssl\_encrypt($data, 'AES-128-ECB', $key));

}

$sock = socket\_create(AF\_INET, SOCK\_STREAM, SOL\_TCP);

socket\_connect($sock, $host, $port);

$message = readline("Enter message to send: ");

$encrypted = encryptMessage($message, $key);

$data = array("type" => "text", "message" => $encrypted);

socket\_write($sock, json\_encode($data));

$reply = socket\_read($sock, 1024);

$response = json\_decode($reply, true);

if ($response['status'] == 'received') {

echo "Server confirmed receipt of encrypted message.\n";

}

socket\_close($sock);

?>

# 🔐 **Encryption Details**

* Symmetric encryption using **AES-128-ECB**.
* **Base64 encoding** used for safe transmission.
* The server decrypts received messages before saving to database.

# 📂 **Updated Folder Structure (Lab 9)**

pgsql

CopyEdit

Lab 8 + 9/

├── server.php (with encryption)

├── client.php (with encryption)

└── chat.db

📸 **Screenshot of Output**