

**LA GRANDEE INTERNATIONAL COLLEGE**

**Simalchaur, Pokhara, Nepal**

Assignment I

**Submitted to**

Mr. Ashwin Poudel

La Grandee International College

Bachelor of Computer Application (BCA) Program

In partial fulfillment of the requirements for the degree of Program Name under

Pokhara University

**Submitted by**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name:** | **Course** | **Semester** | **P.U. Registration Number** |
| Subash Gurung | BCA | 6th | 2022-1-53-0147 |

**Date: 4th May 2025**

## Acknowledgment

I would like to express my sincere gratitude to **Er. Ashwin Poudel**, our respected instructor, for his invaluable guidance, encouragement, and continuous support throughout the completion of this lab work. His clear explanations and comprehensive demonstrations greatly facilitated my understanding of even the most complex concepts.

I am particularly thankful for the time and effort he devoted to assisting us in setting up the XAMPP environment. Despite various technical challenges and delays, he consistently provided patient, individualized support to ensure that each student could proceed confidently.

The opportunity to perform practical tasks on my personal laptop significantly enhanced my comprehension of PHP concepts and their real-world applications in web development. Writing and testing code independently enabled me to gain hands-on experience in server-side scripting, form handling, and database interactions. This practical exposure not only deepened my technical knowledge but also strengthened my confidence in developing dynamic web applications.

I would also like to acknowledge my classmates for their collaboration, mutual support, and willingness to share knowledge and resources throughout this period. Their cooperation made the learning process more engaging, effective, and enjoyable.

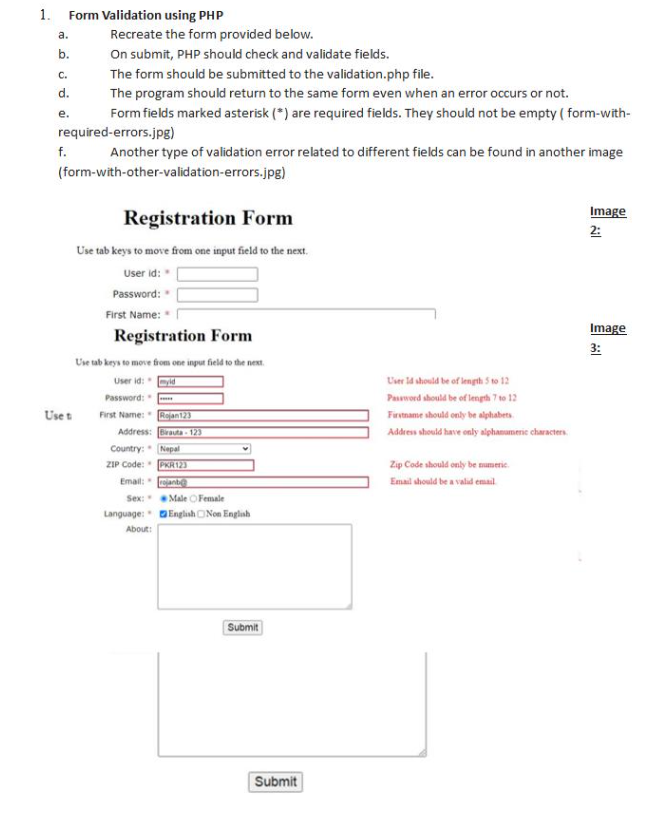
Finally, I am grateful for the opportunity to bridge theoretical learning with practical implementation, which has been instrumental in solidifying my understanding of the subject matter.

Name: Subash Gurung

Roll no: 28

Date: June 3, 2025

**Question:**



1. **Introduction**

This is a PHP backend description of a registration form with field validation using regular expressions, password hashing, and safe insertion into a MySQL database using prepared statements.

**2.form.php**

<form method="post" action="">

    User id: \* <input type="text" name="userid" value="<?= $userId ?>"><span class="error"> <?= $userIdErr ?></span><br><br>

    Password: \* <input type="password" name="password"><span class="error"> <?= $passwordErr ?></span><br><br>

    First Name: \* <input type="text" name="firstname" value="<?= $firstname ?>"><span class="error"> <?= $firstnameErr ?></span><br><br>

    Address: \* <input type="text" name="address" value="<?= $address ?>"><span class="error"> <?= $addressErr ?></span><br><br>

    Country: \*

    <select name="country">

        <option value="Nepal">Nepal</option>

        <option value="India">India</option>

        <option value="USA">USA</option>

    </select><br><br>

    ZIP Code: \* <input type="text" name="zip" value="<?= $zip ?>"><span class="error"> <?= $zipErr ?></span><br><br>

    Email: \* <input type="text" name="email" value="<?= $email ?>"><span class="error"> <?= $emailErr ?></span><br><br>

    Sex: \*

    <input type="radio" name="sex" value="Male">Male

    <input type="radio" name="sex" value="Female">Female<br><br>

    Language: \*

    <input type="checkbox" name="language[]" value="English">English

    <input type="checkbox" name="language[]" value="Non English">Non English<br><br>

    About: <br><textarea name="about" rows="5" cols="40"></textarea><br><br>

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

</form>

**3.Validation**

<?php

$userId = $password = $firstname = $address = $zip = $email = "";

$userIdErr = $passwordErr = $firstnameErr = $addressErr = $zipErr = $emailErr = "";

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

    // User ID

    if (empty($\_POST["userid"]) || strlen($\_POST["userid"]) < 5 || strlen($\_POST["userid"]) > 12) {

        $userIdErr = "User Id should be of length 5 to 12";

    } else {

        $userId = $\_POST["userid"];

    }

    // Password

    if (empty($\_POST["password"]) || strlen($\_POST["password"]) < 7 || strlen($\_POST["password"]) > 12) {

        $passwordErr = "Password should be of length 7 to 12";

    } else {

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

    }

    // First Name

    if (!preg\_match("/^[a-zA-Z]+$/", $\_POST["firstname"])) {

        $firstnameErr = "Firstname should only be alphabets.";

    } else {

        $firstname = $\_POST["firstname"];

    }

    // Address

    if (!preg\_match("/^[a-zA-Z0-9\s\-]+$/", $\_POST["address"])) {

        $addressErr = "Address should have only alphanumeric characters.";

    } else {

        $address = $\_POST["address"];

    }

    // Zip Code

    if (!preg\_match("/^[0-9]+$/", $\_POST["zip"])) {

        $zipErr = "Zip Code should only be numeric.";

    } else {

        $zip = $\_POST["zip"];

    }

    // Email

    if (!filter\_var($\_POST["email"], FILTER\_VALIDATE\_EMAIL)) {

        $emailErr = "Email should be a valid email.";

    } else {

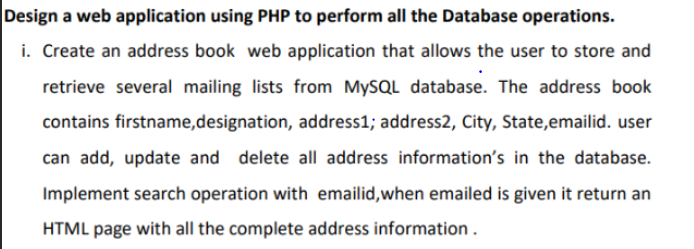
        $email = $\_POST["email"];

    }

}

?>

**Question:**



1. **Introduction**

This project is an Address Book web application developed using PHP and MySQL. It allows users to add, update, delete, and search address records with fields like name, designation, address, city, state, and email. The system includes input validation and stores data securely in the database, providing a simple and efficient way to manage mailing lists.

1. **db.php file code**
2. <?php
3. $host = "localhost";
4. $user = "root";
5. $pass = "";
6. $db = "assignment2";
7. $conn = new mysqli($host, $user, $pass, $db);
8. if ($conn->connect\_error) {
9. die("Connection failed: " . $conn->connect\_error);
10. }

**4.delete.php**

<?php

include 'db.php';

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

$conn->query("DELETE FROM contacts WHERE id=$id");

echo "Deleted! <a href='index.php'>Back</a>";

?>

**5.index.php**

<?php include 'db.php'; ?>

<!DOCTYPE html>

<html>

<head><title>Address Book</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h2>Add New Contact</h2>

<div class="container">

    <form action="insert.php" method="POST">

    First Name: <input name="firstname" required><br>

    Designation: <input name="designation"><br>

    Address1: <input name="address1"><br>

    Address2: <input name="address2"><br>

    City: <input name="city"><br>

    State: <input name="state"><br>

    Email: <input name="emailid" type="email" required><br>

    <button type="submit">Add</button>

</form>

</div>

<h2>Search Contact by Email</h2>

<div class="container">

<form action="search.php" method="GET">

    <input type="email" name="emailid" placeholder="Enter email">

    <button type="submit">Search</button>

</form>

</div>

<h2>All Contacts</h2>

<?php

$result = $conn->query("SELECT \* FROM contacts");

while($row = $result->fetch\_assoc()) {

    echo "<p>{$row['firstname']} | {$row['emailid']}

    <a href='update.php?id={$row['id']}'>Edit</a> |

    <a href='delete.php?id={$row['id']}'>Delete</a></p>";

}

?>

</body>

</html>

**6.insert.php**

<?php

include 'db.php';

$stmt = $conn->prepare("INSERT INTO contacts (firstname, designation, address1, address2, city, state, emailid)

VALUES (?, ?, ?, ?, ?, ?, ?)");

$stmt->bind\_param("sssssss", $\_POST['firstname'], $\_POST['designation'], $\_POST['address1'], $\_POST['address2'], $\_POST['city'], $\_POST['state'], $\_POST['emailid']);

$stmt->execute();

echo "Contact added! <a href='index.php'>Back</a>";

?>

**7.search.php**

<?php

include 'db.php';

$email = $\_GET['emailid'];

$result = $conn->query("SELECT \* FROM contacts WHERE emailid='$email'");

if ($result->num\_rows > 0) {

    $row = $result->fetch\_assoc();

    echo "<h2>Details</h2>";

    echo "<p>Name: {$row['firstname']}</p>";

    echo "<p>Designation: {$row['designation']}</p>";

    echo "<p>Address1: {$row['address1']}</p>";

    echo "<p>Address2: {$row['address2']}</p>";

    echo "<p>City: {$row['city']}</p>";

    echo "<p>State: {$row['state']}</p>";

    echo "<p>Email: {$row['emailid']}</p>";

} else {

    echo "No contact found.";

}

?>

**8.style.css**

body {

    font-family: Arial, sans-serif;

    background: #f2f2f2;

    margin: 20px;

    padding: 20px;

    color: #333;

}

.container {

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

    min-height: 10vh;

}

h2 {

    color: #444;

    margin-top: 40px;

    border-bottom: 2px solid #ddd;

    padding-bottom: 5px;

    align-items: center;

    justify-content: center;

    display: flex;

}

form {

    background: white;

    padding: 20px;

    margin-bottom: 20px;

    border-radius: 8px;

    box-shadow: 0 2px 5px rgba(0,0,0,0.1);

    width: 400px;

}

input {

    width: 100%;

    padding: 8px;

    margin: 8px 0;

    border: 1px solid #ccc;

    border-radius: 4px;

}

button {

    background-color: #4CAF50;

    color: white;

    padding: 10px 15px;

    border: none;

    border-radius: 4px;

    cursor: pointer;

    margin-top: 10px;

}

button:hover {

    background-color: #45a049;

}

p {

    background: white;

    padding: 10px;

    border-radius: 5px;

    margin-bottom: 10px;

}

a {

    color: #007bff;

    margin-left: 10px;

    text-decoration: none;

}

a:hover {

    text-decoration: underline;

}

**9.update.php**

<?php include 'db.php';

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

$data = $conn->query("SELECT \* FROM contacts WHERE id=$id")->fetch\_assoc();

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    $stmt = $conn->prepare("UPDATE contacts SET firstname=?, designation=?, address1=?, address2=?, city=?, state=?, emailid=? WHERE id=?");

    $stmt->bind\_param("sssssssi", $\_POST['firstname'], $\_POST['designation'], $\_POST['address1'], $\_POST['address2'], $\_POST['city'], $\_POST['state'], $\_POST['emailid'], $id);

    $stmt->execute();

    echo "Updated! <a href='index.php'>Back</a>";

} else {

?>

<form method="POST">

    First Name: <input name="firstname" value="<?= $data['firstname'] ?>"><br>

    Designation: <input name="designation" value="<?= $data['designation'] ?>"><br>

    Address1: <input name="address1" value="<?= $data['address1'] ?>"><br>

    Address2: <input name="address2" value="<?= $data['address2'] ?>"><br>

    City: <input name="city" value="<?= $data['city'] ?>"><br>

    State: <input name="state" value="<?= $data['state'] ?>"><br>

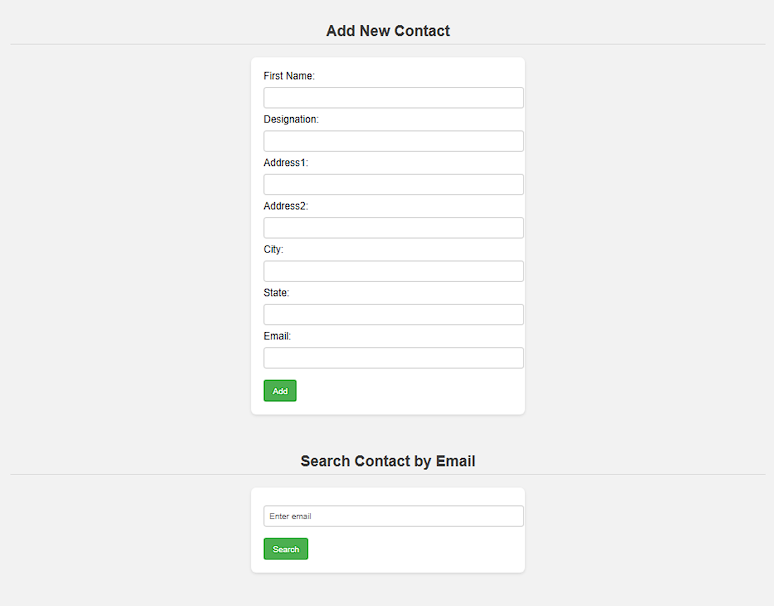
    Email: <input name="emailid" value="<?= $data['emailid'] ?>"><br>

    <button>Update</button>

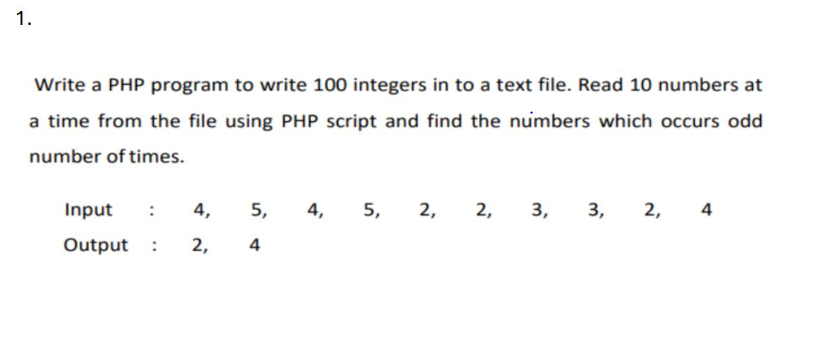
</form>

<?php } ?>

**10.Snapshots**



**Question 1:**



1. **Introduction**

This submission contains two PHP programs:

1. **Odd Numbers Counter** – A file handling program that writes 100 random numbers to a text file, reads them in chunks of 10, and displays numbers that occur an odd number of times.
2. **Resume & Photo Upload System** – A PHP-based file upload form that validates file types, size limits, and ensures secure storage of user-submitted resumes and photographs.

**2.index.php**

<?php

// *Step 1: Write 100 random numbers (1–10) to a file*

$file = fopen("nums.txt", "w");

for ($i = 0; $i < 100; $i++) {

    fwrite($file, rand(1, 10) . " ");

}

fclose($file);

// *Step 2: Read numbers from file*

$data = file\_get\_contents("nums.txt");

$all = explode(" ", trim($data));

$count = [];

// *Step 3: Read 10 at a time and count*

for ($i = 0; $i < count($all); $i += 10) {

    $chunk = array\_slice($all, $i, 10);

    foreach ($chunk as $n) {

        if ($n !== "") $count[$n] = ($count[$n] ?? 0) + 1;

    }

}

// *Step 4: Print numbers with odd occurrences*

echo "Odd occurring numbers: ";

foreach ($count as $num => $times) {

    if ($times % 2 != 0) echo "$num ";

}

?>

**3.resume\_upload.html**

<!DOCTYPE html>

<html>

<head>

  <title>Resume and Photo Upload</title>

</head>

<body>

  <h2>Submit Your Resume and Photograph</h2>

  <form action="upload.php" method="POST" enctype="multipart/form-data">

    <fieldset style="width: 300px;">

      <legend>Upload Details</legend>

      <label for="resume">Resume (PDF/DOC, max 500KB):</label><br>

      <input type="file" id="resume" name="resume" accept=".pdf,.doc" required><br><br>

      <label for="photo">Photograph (JPG/JPEG, max 1MB):</label><br>

      <input type="file" id="photo" name="photo" accept=".jpg,.jpeg" required><br><br>

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

    </fieldset>

  </form>

</body>

</html>

**4.upload.php**

<?php

$resumeDir = "resumes/";

$photoDir = "photos/";

// *Create directories if not exist*

if (!is\_dir($resumeDir)) mkdir($resumeDir);

if (!is\_dir($photoDir)) mkdir($photoDir);

// *Resume*

if (isset($\_FILES['resume']) && is\_uploaded\_file($\_FILES['resume']['tmp\_name'])) {

    $resume = $\_FILES['resume'];

    $name = basename($resume['name']);

    $path = $resumeDir . $name;

    $ext = strtolower(pathinfo($name, PATHINFO\_EXTENSION));

    $size = $resume['size'];

    if (file\_exists($path)) {

        echo "Resume already exists.<br>";

    } elseif (!in\_array($ext, ['pdf', 'doc'])) {

        echo "Resume must be PDF or DOC.<br>";

    } elseif ($size > 512000) {

        echo "Resume must be under 500KB.<br>";

    } else {

        move\_uploaded\_file($resume['tmp\_name'], $path);

        echo "Resume uploaded.<br>";

    }

}

// *Photo*

if (isset($\_FILES['photo']) && is\_uploaded\_file($\_FILES['photo']['tmp\_name'])) {

    $photo = $\_FILES['photo'];

    $name = basename($photo['name']);

    $path = $photoDir . $name;

    $ext = strtolower(pathinfo($name, PATHINFO\_EXTENSION));

    $size = $photo['size'];

    if (file\_exists($path)) {

        echo "Photo already exists.<br>";

    } elseif (!in\_array($ext, ['jpg', 'jpeg'])) {

        echo "Photo must be JPG or JPEG.<br>";

    } elseif ($size > 1048576) {

        echo "Photo must be under 1MB.<br>";

    } else {

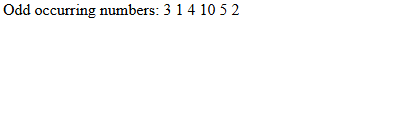
        move\_uploaded\_file($photo['tmp\_name'], $path);

        echo "Photo uploaded.<br>";

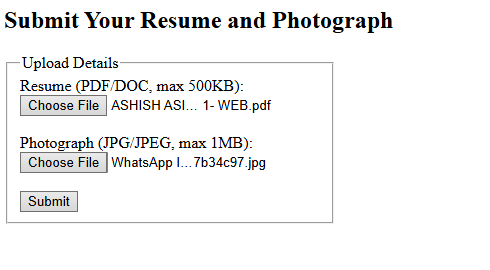
    }

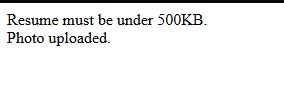
}

**5.Snapshots**

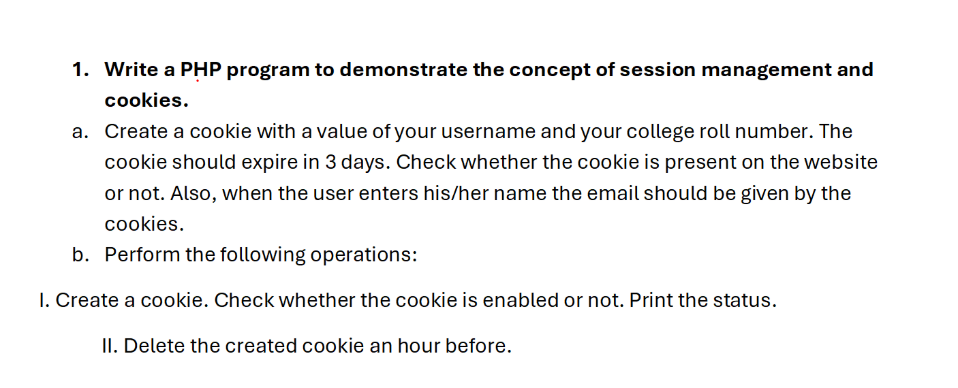


**ii.**





**Question 1:**



**Set\_cookie.php**

<?php

$username = "subash";

$roll = "Bca6th";

$email = "subash@mail.com";

// *Set cookie for 3 days*

setcookie("user\_data", "$username|$roll|$email", time() + (3 \* 24 \* 60 \* 60));

// *Check if cookie is set*

if(isset($\_COOKIE['user\_data'])) {

    echo "Cookie is enabled. Value: " . $\_COOKIE['user\_data'];

} else {

    echo "Cookie is not set (yet). Refresh to check again.";

}

?>

**Get\_email\_from\_cookie.php**

<?php

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

    $inputName = $\_POST['username'];

    if(isset($\_COOKIE['user\_data'])) {

        list($name, $roll, $email) = explode("|", $\_COOKIE['user\_data']);

        if(strtolower($name) === strtolower($inputName)) {

            echo "Welcome $name! Your email is: $email";

        } else {

            echo "Username doesn't match cookie.";

        }

    } else {

        echo "No cookie found.";

    }

}

?>

<form method="POST">

  Enter your name: <input type="text" name="username" required />

  <input type="submit" value="Check Email" />

</form>

**Delete\_cookie.php**

<?php

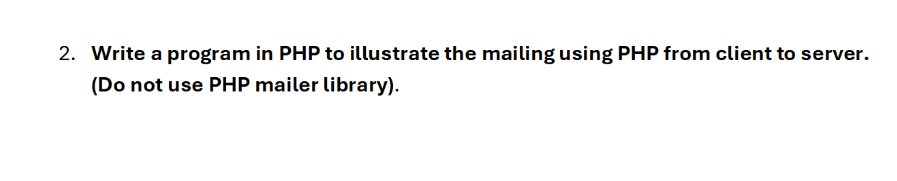
// *Set cookie with past time to delete it*

setcookie("user\_data", "", time() - 3600);

echo " Cookie deleted (expired 1 hour ago).";

?>

**Question 2:**



**Send\_mail.php**

<?php

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

    $to = $\_POST['to'];

    $subject = $\_POST['subject'];

    $message = $\_POST['message'];

    $headers = "From: subash@gmail.com";

    if(mail($to, $subject, $message, $headers)) {

        echo " Mail sent successfully.";

    } else {

        echo " Mail sending failed.";

    }

}

?>

<form method="POST">

  To: <input type="email" name="to" required /><br>

  Subject: <input type="text" name="subject" required /><br>

  Message:<br>

  <textarea name="message" rows="5" cols="30" required></textarea><br>

  <input type="submit" name="send" value="Send Mail" />

</form>

**ENCRYPTED COMMUNICATION APP**

An **encrypted communication app** ensures that messages sent between users are secure and cannot be read by anyone except the intended recipient, even if intercepted. It uses **end-to-end encryption (E2EE)** to make this possible.

Examples include:

* Signal
* WhatsApp (uses Signal Protocol)
* Telegram (secret chats only)
* ProtonMail (for emails)

### ⚙️ Core Features

1. **User Authentication**
   * Secure login/signup (email, phone number, or username)
   * Two-factor authentication (2FA)
2. **End-to-End Encrypted Messaging**
   * One-on-one chats
   * Group chats
   * Multimedia (images, audio, files)
3. **Voice & Video Calls**
   * Secure and encrypted (using WebRTC or similar)
4. **Message Security Features**
   * Self-destructing messages
   * Screenshot blocking (optional)
   * Message read receipts
   * Forwarding restrictions
5. **Key Management**
   * Generate and store keys securely
   * QR-code or key-fingerprint verification
6. **Offline Support**
   * Store encrypted messages until recipient comes online
7. **Cross-Platform**
   * Android, iOS, and Web apps

### 🔒 Encryption Approach

1. **End-to-End Encryption**
   * Use the **Signal Protocol** or **Double Ratchet Algorithm**.
   * Messages are encrypted on sender’s device and decrypted only on the receiver’s device.
2. **Asymmetric Cryptography (e.g., RSA or ECC)**
   * For exchanging secure session keys.
3. **Symmetric Cryptography (e.g., AES-256)**
   * For encrypting the message content.
4. **Forward Secrecy**
   * Session keys change frequently to ensure past messages remain secure even if one key is compromised.

### 💻 Tech Stack Suggestions

#### Backend:

* **Language:** Node.js, Python (Django/Flask), or Go
* **Database:** PostgreSQL / MongoDB
* **Authentication:** OAuth2.0, JWT
* **Encryption Libraries:** Libsodium, OpenSSL

#### Frontend:

* **Mobile:** Flutter / React Native / Kotlin / Swift
* **Web:** React.js, Vue.js

#### Realtime Communication:

* **WebSockets / Socket.IO** for chat
* **WebRTC** for video/audio calls

#### Deployment:

* **Cloud:** AWS / DigitalOcean / Firebase
* **CI/CD:** GitHub Actions / GitLab CI

### 🛡️ Additional Security Best Practices

* Do not store plain-text messages on the server.
* Enforce strong password policies.
* Secure key storage (e.g., Android Keystore, iOS Keychain).
* Open-source the app for transparency.
* Regular security audits.