**Q.1 Create html pages for website like login, registration and about us pages**

1. index.html - **Home Page**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Home - My Website</title>

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

</head>

<body>

<header>

<h1>Welcome to Our Website</h1>

<nav>

<a href="index.html">Home</a>

<a href="login.html">Login</a>

<a href="register.html">Register</a>

<a href="about.html">About Us</a>

</nav>

</header>

<section>

<img src="images/home.jpg" alt="Welcome Image" width="600">

<p>This is the home page of our website.</p>

</section>

</body>

</html>

2. login.html - **Login Page**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Login - My Website</title>

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

</head>

<body>

<header>

<h1>Login to Your Account</h1>

<nav>

<a href="index.html">Home</a>

<a href="login.html">Login</a>

<a href="register.html">Register</a>

<a href="about.html">About Us</a>

</nav>

</header>

<section>

<form action="login-action.php" method="post">

<label for="username">Username:</label>

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

<label for="password">Password:</label>

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

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

</form>

</section>

</body>

</html>

3. register.html **- Registration Page**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Register - My Website</title>

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

</head>

<body>

<header>

<h1>Create a New Account</h1>

<nav>

<a href="index.html">Home</a>

<a href="login.html">Login</a>

<a href="register.html">Register</a>

<a href="about.html">About Us</a>

</nav>

</header>

<section>

<form action="register-action.php" method="post">

<label for="name">Full Name:</label>

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

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<label for="password">Password:</label>

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

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

</form>

</section>

</body>

</html>

4. about.html **- About Us Page**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>About Us - My Website</title>

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

</head>

<body>

<header>

<h1>About Us</h1>

<nav>

<a href="index.html">Home</a>

<a href="login.html">Login</a>

<a href="register.html">Register</a>

<a href="about.html">About Us</a>

</nav>

</header>

<section>

<img src="images/about.jpg" alt="About Us Image" width="600">

<p>Learn more about our company and mission.</p>

</section>

</body>

</html>

5. styles.css **- Basic CSS for Styling (Optional)**

css

Copy code

body {

font-family: Arial, sans-serif;

text-align: center;

margin: 0;

}

header {

background-color: #4CAF50;

color: white;

padding: 10px;

}

nav a {

margin: 0 15px;

color: white;

text-decoration: none;

}

nav a:hover {

text-decoration: underline;

}

section {

padding: 20px;

}

form {

display: inline-block;

text-align: left;

margin-top: 20px;

}

label {

display: block;

margin: 10px 0 5px;

}

input[type="text"],

input[type="password"],

input[type="email"] {

width: 100%;

padding: 8px;

margin-bottom: 15px;

}

button {

width: 100%;

padding: 10px;

background-color: #4CAF50;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

**Image Setup**

Save the images as home.jpg and about.jpg in an images folder in your project directory.

**Q.2 design the website for restaurant using HTML and CSS**

### 1. index.html - Home Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Home - Restaurant</title>

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

</head>

<body>

<header>

<h1>Welcome to Our Restaurant</h1>

<nav>

<a href="index.html">Home</a>

<a href="menu.html">Menu</a>

<a href="reservation.html">Reservation</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<img src="images/restaurant.jpg" alt="Restaurant Interior" class="main-image">

<p>Experience the finest cuisine in a warm, welcoming atmosphere. Join us for an unforgettable dining experience.</p>

</section>

</body>

</html>

### 2. menu.html - Menu Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Menu - Restaurant</title>

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

</head>

<body>

<header>

<h1>Our Menu</h1>

<nav>

<a href="index.html">Home</a>

<a href="menu.html">Menu</a>

<a href="reservation.html">Reservation</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<img src="images/menu.jpg" alt="Restaurant Menu" class="main-image">

<h2>Popular Dishes</h2>

<ul>

<li>Grilled Salmon - $20</li>

<li>Spaghetti Bolognese - $15</li>

<li>Caesar Salad - $10</li>

<li>Margherita Pizza - $12</li>

</ul>

</section>

</body>

</html>

### 3. reservation.html - Reservation Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Reservation - Restaurant</title>

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

</head>

<body>

<header>

<h1>Make a Reservation</h1>

<nav>

<a href="index.html">Home</a>

<a href="menu.html">Menu</a>

<a href="reservation.html">Reservation</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<form action="reservation-confirmation.php" method="post">

<label for="name">Name:</label>

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

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<label for="date">Date:</label>

<input type="date" id="date" name="date" required>

<label for="time">Time:</label>

<input type="time" id="time" name="time" required>

<label for="guests">Guests:</label>

<input type="number" id="guests" name="guests" min="1" max="20" required>

<button type="submit">Reserve Now</button>

</form>

</section>

</body>

</html>

### 4. contact.html - Contact Us Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Contact Us - Restaurant</title>

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

</head>

<body>

<header>

<h1>Contact Us</h1>

<nav>

<a href="index.html">Home</a>

<a href="menu.html">Menu</a>

<a href="reservation.html">Reservation</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<p>For inquiries, please contact us:</p>

<p>Email: info@restaurant.com</p>

<p>Phone: (123) 456-7890</p>

<p>Address: 123 Gourmet St., Food City</p>

</section>

</body>

</html>

### 5. styles.css - Basic CSS Styling

css

Copy code

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

text-align: center;

}

header {

background-color: #8b0000;

color: white;

padding: 20px;

}

nav a {

color: white;

margin: 0 15px;

text-decoration: none;

font-size: 18px;

}

nav a:hover {

text-decoration: underline;

}

.main-image {

width: 80%;

margin: 20px 0;

border-radius: 8px;

}

section {

padding: 20px;

}

form {

margin: 0 auto;

text-align: left;

width: 300px;

}

label {

display: block;

margin-top: 10px;

}

input[type="text"],

input[type="email"],

input[type="date"],

input[type="time"],

input[type="number"] {

width: 100%;

padding: 8px;

margin-top: 5px;

}

button {

width: 100%;

padding: 10px;

margin-top: 15px;

background-color: #8b0000;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #a52a2a;

}

### Image Setup

Save your images as restaurant.jpg (for home page), menu.jpg (for menu page), etc., in an images folder.

**Q.3 design the website for hospital using HTML and CSS**

### 1. index.html - Home Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Home - Hospital</title>

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

</head>

<body>

<header>

<h1>Welcome to Our Hospital</h1>

<nav>

<a href="index.html">Home</a>

<a href="services.html">Services</a>

<a href="appointment.html">Appointment</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<img src="images/hospital.jpg" alt="Hospital Building" class="main-image">

<p>Providing compassionate care and the latest medical advancements to our community.</p>

</section>

</body>

</html>

### 2. services.html - Services Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Services - Hospital</title>

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

</head>

<body>

<header>

<h1>Our Services</h1>

<nav>

<a href="index.html">Home</a>

<a href="services.html">Services</a>

<a href="appointment.html">Appointment</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<img src="images/services.jpg" alt="Medical Services" class="main-image">

<h2>Specialties</h2>

<ul>

<li>Emergency Care</li>

<li>Cardiology</li>

<li>Orthopedics</li>

<li>Pediatrics</li>

<li>General Surgery</li>

</ul>

</section>

</body>

</html>

### 3. appointment.html - Appointment Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Appointment - Hospital</title>

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

</head>

<body>

<header>

<h1>Book an Appointment</h1>

<nav>

<a href="index.html">Home</a>

<a href="services.html">Services</a>

<a href="appointment.html">Appointment</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<form action="appointment-confirmation.php" method="post">

<label for="name">Full Name:</label>

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

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<label for="date">Date:</label>

<input type="date" id="date" name="date" required>

<label for="department">Department:</label>

<select id="department" name="department" required>

<option value="emergency">Emergency</option>

<option value="cardiology">Cardiology</option>

<option value="orthopedics">Orthopedics</option>

<option value="pediatrics">Pediatrics</option>

</select>

<button type="submit">Book Appointment</button>

</form>

</section>

</body>

</html>

### 4. contact.html - Contact Us Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Contact Us - Hospital</title>

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

</head>

<body>

<header>

<h1>Contact Us</h1>

<nav>

<a href="index.html">Home</a>

<a href="services.html">Services</a>

<a href="appointment.html">Appointment</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<p>For inquiries, please reach out to us:</p>

<p>Email: contact@hospital.com</p>

<p>Phone: (555) 123-4567</p>

<p>Address: 456 Health St., Wellness City</p>

</section>

</body>

</html>

### 5. styles.css - Basic CSS Styling

css

Copy code

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

text-align: center;

}

header {

background-color: #2a9d8f;

color: white;

padding: 20px;

}

nav a {

color: white;

margin: 0 15px;

text-decoration: none;

font-size: 18px;

}

nav a:hover {

text-decoration: underline;

}

.main-image {

width: 80%;

margin: 20px 0;

border-radius: 8px;

}

section {

padding: 20px;

}

form {

margin: 0 auto;

text-align: left;

width: 300px;

}

label {

display: block;

margin-top: 10px;

}

input[type="text"],

input[type="email"],

input[type="date"],

select {

width: 100%;

padding: 8px;

margin-top: 5px;

}

button {

width: 100%;

padding: 10px;

margin-top: 15px;

background-color: #2a9d8f;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #21867a;

}

### Image Setup

Save images such as hospital.jpg for the homepage, services.jpg for the services page, etc., in an images folder in your project directory.

**Q.4 design the website for library using HTML and CSS**

### 1. index.html - Home Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Library Home</title>

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

</head>

<body>

<header>

<h1>Welcome to Our Library</h1>

<nav>

<a href="index.html">Home</a>

<a href="catalog.html">Catalog</a>

<a href="membership.html">Membership</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<img src="images/library.jpg" alt="Library Interior" class="main-image">

<p>Your portal to endless knowledge, resources, and learning opportunities.</p>

</section>

</body>

</html>

### 2. catalog.html - Catalog Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Catalog - Library</title>

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

</head>

<body>

<header>

<h1>Library Catalog</h1>

<nav>

<a href="index.html">Home</a>

<a href="catalog.html">Catalog</a>

<a href="membership.html">Membership</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<img src="images/catalog.jpg" alt="Library Catalog" class="main-image">

<h2>Popular Collections</h2>

<ul>

<li>Fiction</li>

<li>Non-Fiction</li>

<li>Science & Technology</li>

<li>History</li>

<li>Children's Books</li>

</ul>

</section>

</body>

</html>

### 3. membership.html - Membership Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Membership - Library</title>

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

</head>

<body>

<header>

<h1>Become a Member</h1>

<nav>

<a href="index.html">Home</a>

<a href="catalog.html">Catalog</a>

<a href="membership.html">Membership</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<img src="images/membership.jpg" alt="Library Membership" class="main-image">

<h2>Membership Benefits</h2>

<p>As a member, you enjoy access to exclusive resources, online book reservations, and more.</p>

<form action="membership-confirmation.php" method="post">

<label for="name">Full Name:</label>

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

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<button type="submit">Join Now</button>

</form>

</section>

</body>

</html>

### 4. contact.html - Contact Us Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Contact Us - Library</title>

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

</head>

<body>

<header>

<h1>Contact Us</h1>

<nav>

<a href="index.html">Home</a>

<a href="catalog.html">Catalog</a>

<a href="membership.html">Membership</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<p>If you have any questions or need assistance, feel free to reach out to us:</p>

<p>Email: info@library.com</p>

<p>Phone: (123) 456-7890</p>

<p>Address: 789 Knowledge St., Book City</p>

</section>

</body>

</html>

### 5. styles.css - Basic CSS Styling

css

Copy code

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

text-align: center;

}

header {

background-color: #3b5998;

color: white;

padding: 20px;

}

nav a {

color: white;

margin: 0 15px;

text-decoration: none;

font-size: 18px;

}

nav a:hover {

text-decoration: underline;

}

.main-image {

width: 80%;

margin: 20px 0;

border-radius: 8px;

}

section {

padding: 20px;

}

form {

margin: 0 auto;

text-align: left;

width: 300px;

}

label {

display: block;

margin-top: 10px;

}

input[type="text"],

input[type="email"] {

width: 100%;

padding: 8px;

margin-top: 5px;

}

button {

width: 100%;

padding: 10px;

margin-top: 15px;

background-color: #3b5998;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #333;

}

### Image Setup

Store images such as library.jpg for the homepage, catalog.jpg for the catalog page, and membership.jpg for the membership page in an images folder within your project directory.

**Q.5 design the website for AIML Department using HTML and CSS**

### 1. index.html - Home Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>AI & ML Department - Home</title>

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

</head>

<body>

<header>

<h1>AI & ML Department</h1>

<nav>

<a href="index.html">Home</a>

<a href="research.html">Research</a>

<a href="faculty.html">Faculty</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<img src="images/aiml\_home.jpg" alt="AI & ML Department" class="main-image">

<p>Welcome to the Department of Artificial Intelligence and Machine Learning. Discover the forefront of technology and innovation in AI & ML research and education.</p>

</section>

</body>

</html>

### 2. research.html - Research Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Research - AI & ML Department</title>

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

</head>

<body>

<header>

<h1>Our Research</h1>

<nav>

<a href="index.html">Home</a>

<a href="research.html">Research</a>

<a href="faculty.html">Faculty</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<img src="images/research.jpg" alt="AI & ML Research" class="main-image">

<h2>Research Areas</h2>

<ul>

<li>Natural Language Processing</li>

<li>Computer Vision</li>

<li>Robotics</li>

<li>Deep Learning</li>

<li>Reinforcement Learning</li>

</ul>

</section>

</body>

</html>

### 3. faculty.html - Faculty Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Faculty - AI & ML Department</title>

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

</head>

<body>

<header>

<h1>Our Faculty</h1>

<nav>

<a href="index.html">Home</a>

<a href="research.html">Research</a>

<a href="faculty.html">Faculty</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<img src="images/faculty.jpg" alt="AI & ML Faculty" class="main-image">

<h2>Meet Our Faculty</h2>

<ul>

<li>Dr. John Doe - Head of Department</li>

<li>Dr. Jane Smith - NLP Specialist</li>

<li>Dr. Alice Johnson - Robotics Researcher</li>

<li>Dr. Tom Brown - Deep Learning Expert</li>

</ul>

</section>

</body>

</html>

### 4. contact.html - Contact Us Page

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Contact Us - AI & ML Department</title>

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

</head>

<body>

<header>

<h1>Contact Us</h1>

<nav>

<a href="index.html">Home</a>

<a href="research.html">Research</a>

<a href="faculty.html">Faculty</a>

<a href="contact.html">Contact Us</a>

</nav>

</header>

<section>

<p>Get in touch with the AI & ML Department for any inquiries or support:</p>

<p>Email: aiml@university.com</p>

<p>Phone: (123) 456-7890</p>

<p>Address: 123 AI Street, Tech City</p>

</section>

</body>

</html>

### 5. styles.css - Basic CSS Styling

css

Copy code

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

text-align: center;

}

header {

background-color: #34495e;

color: white;

padding: 20px;

}

nav a {

color: white;

margin: 0 15px;

text-decoration: none;

font-size: 18px;

}

nav a:hover {

text-decoration: underline;

}

.main-image {

width: 80%;

margin: 20px 0;

border-radius: 8px;

}

section {

padding: 20px;

}

ul {

list-style-type: none;

padding: 0;

}

ul li {

font-size: 18px;

margin: 10px 0;

}

### Image Setup

Save images such as aiml\_home.jpg for the homepage, research.jpg for the research page, and faculty.jpg for the faculty page in an images folder within your project directory.

**Q. 6 Write a program demonstrating javascript functions and different validations**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Registration Form with Validation</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.form-container {

width: 300px;

padding: 20px;

border: 1px solid #ccc;

border-radius: 8px;

}

input[type="text"], input[type="email"], input[type="password"] {

width: 100%;

padding: 8px;

margin-top: 8px;

}

button {

width: 100%;

padding: 10px;

background-color: #28a745;

color: white;

border: none;

margin-top: 10px;

}

.error {

color: red;

font-size: 0.9em;

}

</style>

</head>

<body>

<div class="form-container">

<h2>Register</h2>

<form id="registrationForm">

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<span id="emailError" class="error"></span>

<label for="password">Password:</label>

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

<span id="passwordError" class="error"></span>

<label for="age">Age:</label>

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

<span id="ageError" class="error"></span>

<button type="button" onclick="validateForm()">Submit</button>

</form>

</div>

<script>

// Validate email format

function validateEmail(email) {

const emailPattern = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;

return emailPattern.test(email);

}

// Validate password length (at least 8 characters)

function validatePassword(password) {

return password.length >= 8;

}

// Validate age (must be a number and at least 18)

function validateAge(age) {

const ageNumber = parseInt(age, 10);

return !isNaN(ageNumber) && ageNumber >= 18;

}

// Function to validate form

function validateForm() {

// Get values from the form

const email = document.getElementById("email").value;

const password = document.getElementById("password").value;

const age = document.getElementById("age").value;

// Initialize validation flags

let isFormValid = true;

// Clear previous error messages

document.getElementById("emailError").textContent = "";

document.getElementById("passwordError").textContent = "";

document.getElementById("ageError").textContent = "";

// Email validation

if (!validateEmail(email)) {

document.getElementById("emailError").textContent = "Invalid email format";

isFormValid = false;

}

// Password validation

if (!validatePassword(password)) {

document.getElementById("passwordError").textContent = "Password must be at least 8 characters long";

isFormValid = false;

}

// Age validation

if (!validateAge(age)) {

document.getElementById("ageError").textContent = "Age must be a number and at least 18";

isFormValid = false;

}

// If all validations pass, display a success message

if (isFormValid) {

alert("Form submitted successfully!");

}

}

</script>

</body>

</html>

**Q. 7 Write a program demonstrating javascript functions and different validations**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Registration Form with Regex Validation</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.form-container {

width: 300px;

padding: 20px;

border: 1px solid #ccc;

border-radius: 8px;

}

input[type="text"], input[type="email"], input[type="password"] {

width: 100%;

padding: 8px;

margin-top: 8px;

}

button {

width: 100%;

padding: 10px;

background-color: #28a745;

color: white;

border: none;

margin-top: 10px;

}

.error {

color: red;

font-size: 0.9em;

}

</style>

</head>

<body>

<div class="form-container">

<h2>Register</h2>

<form id="registrationForm">

<label for="username">Username:</label>

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

<span id="usernameError" class="error"></span>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<span id="emailError" class="error"></span>

<label for="password">Password:</label>

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

<span id="passwordError" class="error"></span>

<label for="phone">Phone Number:</label>

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

<span id="phoneError" class="error"></span>

<button type="button" onclick="validateForm()">Submit</button>

</form>

</div>

<script>

// Validation functions using regex

// Validate username (only letters and numbers, min 3 chars)

function validateUsername(username) {

const usernamePattern = /^[a-zA-Z0-9]{3,}$/;

return usernamePattern.test(username);

}

// Validate email format

function validateEmail(email) {

const emailPattern = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;

return emailPattern.test(email);

}

// Validate password (min 8 chars, with at least 1 uppercase, 1 lowercase, 1 number, and 1 special character)

function validatePassword(password) {

const passwordPattern = /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/;

return passwordPattern.test(password);

}

// Validate phone number (exactly 10 digits)

function validatePhone(phone) {

const phonePattern = /^\d{10}$/;

return phonePattern.test(phone);

}

// Form validation function

function validateForm() {

const username = document.getElementById("username").value;

const email = document.getElementById("email").value;

const password = document.getElementById("password").value;

const phone = document.getElementById("phone").value;

// Initialize validation flag

let isFormValid = true;

// Clear previous error messages

document.getElementById("usernameError").textContent = "";

document.getElementById("emailError").textContent = "";

document.getElementById("passwordError").textContent = "";

document.getElementById("phoneError").textContent = "";

// Validate username

if (!validateUsername(username)) {

document.getElementById("usernameError").textContent = "Username must be at least 3 characters and only letters and numbers.";

isFormValid = false;

}

// Validate email

if (!validateEmail(email)) {

document.getElementById("emailError").textContent = "Invalid email format.";

isFormValid = false;

}

// Validate password

if (!validatePassword(password)) {

document.getElementById("passwordError").textContent = "Password must be at least 8 characters, include uppercase, lowercase, number, and special character.";

isFormValid = false;

}

// Validate phone number

if (!validatePhone(phone)) {

document.getElementById("phoneError").textContent = "Phone number must be exactly 10 digits.";

isFormValid = false;

}

// Show success message if form is valid

if (isFormValid) {

alert("Form submitted successfully!");

}

}

</script>

</body>

</html>

**Q. 8 Write a program demonstrating javascript functions and different validations**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Form Validation Example</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.form-container {

width: 300px;

padding: 20px;

border: 1px solid #ccc;

border-radius: 8px;

}

input[type="text"] {

width: 100%;

padding: 8px;

margin-top: 8px;

}

button {

width: 100%;

padding: 10px;

background-color: #28a745;

color: white;

border: none;

margin-top: 10px;

}

.error {

color: red;

font-size: 0.9em;

}

</style>

</head>

<body>

<div class="form-container">

<h2>Registration Form</h2>

<form id="registrationForm">

<label for="name">Name:</label>

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

<span id="nameError" class="error"></span>

<label for="phone">Phone Number:</label>

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

<span id="phoneError" class="error"></span>

<button type="button" onclick="validateForm()">Submit</button>

</form>

</div>

<script>

// Validate name (only letters and spaces, min 3 chars)

function validateName(name) {

const namePattern = /^[a-zA-Z\s]{3,}$/;

return namePattern.test(name);

}

// Validate phone number (10 digits)

function validatePhone(phone) {

const phonePattern = /^\d{10}$/;

return phonePattern.test(phone);

}

// Form validation function

function validateForm() {

const name = document.getElementById("name").value;

const phone = document.getElementById("phone").value;

// Initialize validation flag

let isFormValid = true;

// Clear previous error messages

document.getElementById("nameError").textContent = "";

document.getElementById("phoneError").textContent = "";

// Validate name

if (!validateName(name)) {

document.getElementById("nameError").textContent = "Name must contain only letters and spaces, with a minimum of 3 characters.";

isFormValid = false;

}

// Validate phone number

if (!validatePhone(phone)) {

document.getElementById("phoneError").textContent = "Phone number must be exactly 10 digits.";

isFormValid = false;

}

// Show success message if form is valid

if (isFormValid) {

alert("Form submitted successfully!");

}

}

</script>

</body>

</html>

**Q. 9 Develop a program to use of different layouts.**

### Step 1: Create a New Android Project

1. Open **Android Studio**.
2. Create a new project.
   * Choose "Empty Activity".
   * Name your project (e.g., LoginApp).
   * Select Kotlin as the programming language.
   * Choose the appropriate API level (API 21+ is a good option).

### Step 2: Define the Layout in XML

In the res/layout/activity\_main.xml file, define a layout that includes the login form with horizontal fields using a LinearLayout with a horizontal orientation for the username and password input.

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp"

android:gravity="center">

<!-- Title of the page -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Login"

android:textSize="24sp"

android:textStyle="bold"

android:layout\_marginBottom="24dp"

android:gravity="center" />

<!-- Username Field -->

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginBottom="16dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Username:"

android:layout\_gravity="center\_vertical"

android:paddingEnd="8dp" />

<EditText

android:id="@+id/username"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:hint="Enter username"

android:inputType="text" />

</LinearLayout>

<!-- Password Field -->

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginBottom="24dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Password:"

android:layout\_gravity="center\_vertical"

android:paddingEnd="8dp" />

<EditText

android:id="@+id/password"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:hint="Enter password"

android:inputType="textPassword" />

</LinearLayout>

<!-- Login Button -->

<Button

android:id="@+id/loginButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Login"

android:backgroundTint="#00796B"

android:textColor="#FFFFFF" />

<!-- Status Message -->

<TextView

android:id="@+id/statusMessage"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text=""

android:textSize="16sp"

android:layout\_marginTop="16dp"

android:textColor="#FF0000" />

</LinearLayout>

### Explanation of XML Layout

1. **Main Layout**: A LinearLayout with a vertical orientation is used to arrange elements vertically. It contains:
   * A TextView for the title ("Login").
   * A LinearLayout (horizontal) for the username and password fields, which includes a TextView (label) and an EditText (input field).
   * A Button for the login action.
   * A TextView for displaying status messages like login success or failure.
2. **Horizontal Layout for Input Fields**: The username and password fields are placed inside LinearLayout containers with horizontal orientation to align the label and input field side by side.

### Step 3: Handle the Login Logic in Kotlin

Now, open the MainActivity.kt file in the src/main/java/com/yourpackage/loginapp/ directory and write the Kotlin code for handling the login logic.

kotlin

Copy code

package com.example.loginapp

import android.os.Bundle

import android.widget.Button

import android.widget.EditText

import android.widget.TextView

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Initialize views

val usernameEditText = findViewById<EditText>(R.id.username)

val passwordEditText = findViewById<EditText>(R.id.password)

val loginButton = findViewById<Button>(R.id.loginButton)

val statusMessage = findViewById<TextView>(R.id.statusMessage)

// Set up login button click listener

loginButton.setOnClickListener {

val username = usernameEditText.text.toString().trim()

val password = passwordEditText.text.toString().trim()

// Check if username and password are correct

if (username == "admin" && password == "1234") {

statusMessage.text = "Login successful!"

statusMessage.setTextColor(getColor(R.color.teal\_700))

} else {

statusMessage.text = "Invalid username or password."

statusMessage.setTextColor(getColor(R.color.red))

}

}

}

}

### Explanation of Kotlin Code

* We retrieve the references to the EditText (username and password), Button (login button), and TextView (for status messages) using findViewById().
* When the login button is clicked, it checks whether the entered username is "admin" and the password is "1234".
* If the credentials match, a success message is shown. If not, an error message is displayed.

### Step 4: Add Colors (Optional)

In the res/values/colors.xml, you can define the colors for success and error messages.

xml

Copy code

<resources>

<color name="teal\_700">#00796B</color>

<color name="red">#FF0000</color>

</resources>

### Step 5: Running the App

1. **Run the App**: Connect your Android device or launch an emulator.
2. **Click the "Run" Button** in Android Studio to deploy the app.
3. You should see the login form, where entering the correct username (admin) and password (1234) will display a success message. Entering incorrect credentials will show an error message.

**Q. 10 Develop a program to use of different layouts.**

**Step 1: Create a New Android Project**

1. **Open Android Studio**.
2. **Create a New Project**:
   * Choose **"Empty Activity"**.
   * Name your project (e.g., LoginApp).
   * Select **Kotlin** as the programming language.
   * Choose the appropriate **API level** (API 21+ is fine for most purposes).

**Step 2: Define the Layout in XML**

In the res/layout/activity\_main.xml file, create a layout with a **LinearLayout** having **vertical orientation** for stacking the login fields vertically.

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="32dp"

android:gravity="center">

<!-- Title of the page -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Login"

android:textSize="24sp"

android:textStyle="bold"

android:layout\_marginBottom="24dp"

android:gravity="center" />

<!-- Username Field -->

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginBottom="16dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Username:"

android:layout\_gravity="center\_vertical"

android:paddingEnd="8dp" />

<EditText

android:id="@+id/username"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter username"

android:inputType="text" />

</LinearLayout>

<!-- Password Field -->

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginBottom="24dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Password:"

android:layout\_gravity="center\_vertical"

android:paddingEnd="8dp" />

<EditText

android:id="@+id/password"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter password"

android:inputType="textPassword" />

</LinearLayout>

<!-- Login Button -->

<Button

android:id="@+id/loginButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Login"

android:backgroundTint="#00796B"

android:textColor="#FFFFFF" />

<!-- Status Message -->

<TextView

android:id="@+id/statusMessage"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text=""

android:textSize="16sp"

android:layout\_marginTop="16dp"

android:textColor="#FF0000" />

</LinearLayout>

**Explanation of Layout**

* The outer LinearLayout has a **vertical orientation** to arrange the UI elements vertically.
* Inside the layout:
  + The **title** (TextView) is centered at the top.
  + Each of the input fields (username and password) is inside a **LinearLayout** with **horizontal orientation**, so the label (TextView) and the input field (EditText) are arranged side by side.
  + A **Login button** at the bottom.
  + A **TextView** to display a status message, either showing login success or failure.

**Step 3: Handle the Login Logic in Kotlin**

Now, go to the MainActivity.kt file and add the Kotlin code to handle the login logic.

kotlin

Copy code

package com.example.loginapp

import android.os.Bundle

import android.widget.Button

import android.widget.EditText

import android.widget.TextView

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Initialize views

val usernameEditText = findViewById<EditText>(R.id.username)

val passwordEditText = findViewById<EditText>(R.id.password)

val loginButton = findViewById<Button>(R.id.loginButton)

val statusMessageTextView = findViewById<TextView>(R.id.statusMessage)

// Set up login button click listener

loginButton.setOnClickListener {

val username = usernameEditText.text.toString().trim()

val password = passwordEditText.text.toString().trim()

// Simple login logic (hardcoded username and password for demo purposes)

if (username == "admin" && password == "1234") {

statusMessageTextView.text = "Login successful!"

statusMessageTextView.setTextColor(getColor(R.color.teal\_700))

} else {

statusMessageTextView.text = "Invalid username or password."

statusMessageTextView.setTextColor(getColor(R.color.red))

}

}

}

}

**Explanation of Kotlin Code**

* We retrieve references to the **EditText** fields (username and password), the **Button** (loginButton), and the **TextView** (statusMessage).
* When the login button is clicked, the entered username and password are validated against hardcoded values (admin and 1234).
* If the login is successful, a success message is displayed. If the credentials are incorrect, an error message is shown.

**Step 4: Add Colors (Optional)**

To customize the color of the success and error messages, add colors in the res/values/colors.xml file:

xml

Copy code

<resources>

<color name="teal\_700">#00796B</color>

<color name="red">#FF0000</color>

</resources>

**Step 5: Running the App**

1. **Connect your device or start an emulator**.
2. **Click the "Run" button** in Android Studio.
3. The app will launch on the emulator or device, and you should see the login page where you can enter admin as the username and 1234 as the password to test the login functionality.

**Q. 11 Develop a program to use of different layouts.**

**Step 1: Create a New Android Project**

1. **Open Android Studio**.
2. **Create a New Project**:
   * Choose **"Empty Activity"**.
   * Name your project (e.g., LoginApp).
   * Select **Kotlin** as the programming language.
   * Choose the appropriate **API level** (API 21+ is fine for most purposes).

**Step 2: Define the Layout Using FrameLayout**

In the res/layout/activity\_main.xml file, we define the layout using a FrameLayout. While FrameLayout is typically used for stacking views on top of each other, we will use it to hold our login form components.

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:padding="32dp">

<!-- Title of the page -->

<TextView

android:id="@+id/title"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Login"

android:textSize="24sp"

android:textStyle="bold"

android:layout\_gravity="top|center\_horizontal"

android:layout\_marginTop="48dp"/>

<!-- Username Input -->

<EditText

android:id="@+id/username"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter username"

android:layout\_gravity="top|center\_horizontal"

android:layout\_marginTop="120dp"

android:padding="16dp"

android:inputType="text"/>

<!-- Password Input -->

<EditText

android:id="@+id/password"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter password"

android:layout\_gravity="top|center\_horizontal"

android:layout\_marginTop="200dp"

android:padding="16dp"

android:inputType="textPassword"/>

<!-- Login Button -->

<Button

android:id="@+id/loginButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Login"

android:layout\_gravity="top|center\_horizontal"

android:layout\_marginTop="280dp"

android:backgroundTint="#00796B"

android:textColor="#FFFFFF"/>

<!-- Status Message -->

<TextView

android:id="@+id/statusMessage"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text=""

android:textSize="16sp"

android:textColor="#FF0000"

android:layout\_gravity="top|center\_horizontal"

android:layout\_marginTop="350dp"/>

</FrameLayout>

**Explanation of XML Layout**

1. **FrameLayout**:
   * FrameLayout is used as the root container to hold the views. This layout allows us to layer components on top of each other, but for this case, we'll position them in different areas of the screen using layout\_gravity.
2. **TextView (Title)**:
   * The title "Login" is displayed at the top of the screen using layout\_gravity="top|center\_horizontal".
3. **Username and Password Input**:
   * Two EditText components are used for the username and password. These fields are positioned below the title and stacked one after the other with margins to ensure spacing.
4. **Login Button**:
   * A Button component is placed below the password field. It's centered horizontally at the bottom of the login fields.
5. **Status Message**:
   * A TextView is used to show status messages (whether the login is successful or not). It appears below the login button.

**Step 3: Handle the Login Logic in Kotlin**

Next, we'll add the Kotlin code for handling the login process in the MainActivity.kt file.

kotlin

Copy code

package com.example.loginapp

import android.os.Bundle

import android.widget.Button

import android.widget.EditText

import android.widget.TextView

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Initialize views

val usernameEditText = findViewById<EditText>(R.id.username)

val passwordEditText = findViewById<EditText>(R.id.password)

val loginButton = findViewById<Button>(R.id.loginButton)

val statusMessageTextView = findViewById<TextView>(R.id.statusMessage)

// Set up login button click listener

loginButton.setOnClickListener {

val username = usernameEditText.text.toString().trim()

val password = passwordEditText.text.toString().trim()

// Simple login logic (hardcoded username and password for demo purposes)

if (username == "admin" && password == "1234") {

statusMessageTextView.text = "Login successful!"

statusMessageTextView.setTextColor(getColor(R.color.teal\_700))

} else {

statusMessageTextView.text = "Invalid username or password."

statusMessageTextView.setTextColor(getColor(R.color.red))

}

}

}

}

**Explanation of Kotlin Code**

1. **Retrieve UI elements**:
   * We use findViewById() to get references to the EditText fields (for username and password), the Button (login button), and the TextView (for displaying status messages).
2. **Login Logic**:
   * When the login button is clicked, the app checks whether the username is "admin" and the password is "1234". If the login is successful, a success message is displayed; otherwise, an error message is shown.
3. **Display Status Message**:
   * Based on whether the login is successful or not, the TextView updates with a success or error message and changes its text color accordingly.

**Step 4: Add Colors (Optional)**

For better visual feedback, you can add colors in the res/values/colors.xml file:

xml

Copy code

<resources>

<color name="teal\_700">#00796B</color>

<color name="red">#FF0000</color>

</resources>

**Step 5: Running the App**

1. **Connect a device or start the emulator**.
2. **Click the "Run" button** in Android Studio.
3. You should now see a login screen. Enter the username admin and password 1234 to test the login functionality.

**Q. 12 Develop a program to use of different layouts**

**Step 1: Create a New Android Project**

1. **Open Android Studio**.
2. **Create a New Project**:
   * Choose **"Empty Activity"**.
   * Name your project (e.g., LoginApp).
   * Select **Kotlin** as the programming language.
   * Choose the appropriate **API level** (API 21+ is fine for most purposes).

**Step 2: Define the Layout Using AbsoluteLayout**

In the res/layout/activity\_main.xml file, we define the layout using AbsoluteLayout. We set the positions of the UI components explicitly using layout\_x and layout\_y attributes.

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:padding="32dp">

<!-- Title of the page -->

<TextView

android:id="@+id/title"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Login"

android:textSize="24sp"

android:textStyle="bold"

android:layout\_x="120dp"

android:layout\_y="50dp"/>

<!-- Username Input -->

<EditText

android:id="@+id/username"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter username"

android:layout\_x="40dp"

android:layout\_y="150dp"

android:padding="16dp"

android:inputType="text"/>

<!-- Password Input -->

<EditText

android:id="@+id/password"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter password"

android:layout\_x="40dp"

android:layout\_y="220dp"

android:padding="16dp"

android:inputType="textPassword"/>

<!-- Login Button -->

<Button

android:id="@+id/loginButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Login"

android:layout\_x="40dp"

android:layout\_y="290dp"

android:backgroundTint="#00796B"

android:textColor="#FFFFFF"/>

<!-- Status Message -->

<TextView

android:id="@+id/statusMessage"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text=""

android:textSize="16sp"

android:textColor="#FF0000"

android:layout\_x="120dp"

android:layout\_y="360dp"/>

</AbsoluteLayout>

**Explanation of Layout**

1. **AbsoluteLayout**: This layout allows you to place UI components at specific positions using layout\_x and layout\_y attributes. It is not recommended for modern development due to poor flexibility and scalability.
2. **TextView (Title)**:
   * The title ("Login") is placed at coordinates (120, 50) on the screen using layout\_x="120dp" and layout\_y="50dp".
3. **EditText (Username and Password)**:
   * The username field is positioned at (40, 150) and the password field at (40, 220).
4. **Button (Login)**:
   * The login button is positioned at (40, 290).
5. **TextView (Status Message)**:
   * The status message is displayed at (120, 360), and this will show whether the login is successful or not.

**Step 3: Handle the Login Logic in Kotlin**

Next, go to the MainActivity.kt file and add the Kotlin code for handling the login process.

kotlin

Copy code

package com.example.loginapp

import android.os.Bundle

import android.widget.Button

import android.widget.EditText

import android.widget.TextView

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Initialize views

val usernameEditText = findViewById<EditText>(R.id.username)

val passwordEditText = findViewById<EditText>(R.id.password)

val loginButton = findViewById<Button>(R.id.loginButton)

val statusMessageTextView = findViewById<TextView>(R.id.statusMessage)

// Set up login button click listener

loginButton.setOnClickListener {

val username = usernameEditText.text.toString().trim()

val password = passwordEditText.text.toString().trim()

// Simple login logic (hardcoded username and password for demo purposes)

if (username == "admin" && password == "1234") {

statusMessageTextView.text = "Login successful!"

statusMessageTextView.setTextColor(getColor(R.color.teal\_700))

} else {

statusMessageTextView.text = "Invalid username or password."

statusMessageTextView.setTextColor(getColor(R.color.red))

}

}

}

}

**Explanation of Kotlin Code**

1. **Views Initialization**:
   * We use findViewById() to access the **username** and **password** EditText fields, the **login button**, and the **status message** TextView.
2. **Login Logic**:
   * When the login button is clicked, the app checks whether the username is "admin" and the password is "1234". If the login is successful, a success message is shown; otherwise, an error message is displayed.
3. **Status Message Update**:
   * The statusMessageTextView is updated with either a success or error message, and its color changes accordingly.

**Step 4: Add Colors (Optional)**

For better visual feedback, you can define colors in the res/values/colors.xml file:

xml

Copy code

<resources>

<color name="teal\_700">#00796B</color>

<color name="red">#FF0000</color>

</resources>

**Step 5: Running the App**

1. **Connect a device or start an emulator**.
2. **Click the "Run" button** in Android Studio.
3. You should now see a login page. Enter the username admin and password 1234 to test the login functionality.

**Q. 13 Develop a app for demonstration of Registration form using various UI components such as Edittext, checkboxes, radiobuttons, Togglebutton ,and display users response in Textview**

**Step 1: Create a New Android Project**

1. **Open Android Studio**.
2. **Create a New Project**:
   * Choose **"Empty Activity"**.
   * Name your project (e.g., RegistrationApp).
   * Select **Kotlin** as the programming language.
   * Choose the appropriate **API level** (API 21+ is fine for most purposes).

**Step 2: Define the Layout with Various UI Components**

In the res/layout/activity\_main.xml file, define the layout for the registration form. Here, we'll include EditText, RadioButton, CheckBox, ToggleButton, and Button.

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp">

<!-- Name Input -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Name:"

android:textSize="18sp" />

<EditText

android:id="@+id/editTextName"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your name"

android:inputType="textPersonName"/>

<!-- Email Input -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Email:"

android:textSize="18sp" />

<EditText

android:id="@+id/editTextEmail"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your email"

android:inputType="textEmailAddress"/>

<!-- Gender Radio Buttons -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Gender:"

android:textSize="18sp" />

<RadioGroup

android:id="@+id/radioGroupGender"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<RadioButton

android:id="@+id/radioMale"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Male" />

<RadioButton

android:id="@+id/radioFemale"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Female" />

</RadioGroup>

<!-- Hobbies Checkboxes -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hobbies:"

android:textSize="18sp" />

<CheckBox

android:id="@+id/checkBoxReading"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Reading" />

<CheckBox

android:id="@+id/checkBoxTraveling"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Traveling" />

<CheckBox

android:id="@+id/checkBoxSports"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Sports" />

<!-- Subscribe to Newsletter Toggle Button -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Subscribe to Newsletter:"

android:textSize="18sp" />

<ToggleButton

android:id="@+id/toggleButtonNewsletter"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textOff="No"

android:textOn="Yes" />

<!-- Submit Button -->

<Button

android:id="@+id/submitButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Submit" />

<!-- Display User Response -->

<TextView

android:id="@+id/textViewResponse"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text=""

android:textSize="18sp"

android:layout\_marginTop="16dp" />

</LinearLayout>

**Explanation of UI Components**

1. **EditText**:
   * For capturing the user's **name** and **email**.
   * android:inputType="textPersonName" and android:inputType="textEmailAddress" help set the appropriate keyboard types.
2. **RadioGroup and RadioButton**:
   * A RadioGroup is used to group the gender RadioButton elements (Male and Female), allowing only one gender to be selected.
3. **CheckBox**:
   * Multiple CheckBox elements for selecting hobbies: Reading, Traveling, and Sports.
4. **ToggleButton**:
   * A ToggleButton for subscribing to the newsletter. It displays "Yes" or "No" based on the user's choice.
5. **Button**:
   * A submit button that triggers the display of the user's responses when clicked.
6. **TextView**:
   * A TextView to display the user's responses after they click the submit button.

**Step 3: Handle User Input in Kotlin**

Now, in the MainActivity.kt file, write the Kotlin code to handle the logic of gathering the user's input and displaying the response when the user clicks the **Submit** button.

kotlin

Copy code

package com.example.registrationapp

import android.os.Bundle

import android.widget.\*

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Initialize views

val nameEditText = findViewById<EditText>(R.id.editTextName)

val emailEditText = findViewById<EditText>(R.id.editTextEmail)

val genderRadioGroup = findViewById<RadioGroup>(R.id.radioGroupGender)

val readingCheckBox = findViewById<CheckBox>(R.id.checkBoxReading)

val travelingCheckBox = findViewById<CheckBox>(R.id.checkBoxTraveling)

val sportsCheckBox = findViewById<CheckBox>(R.id.checkBoxSports)

val newsletterToggleButton = findViewById<ToggleButton>(R.id.toggleButtonNewsletter)

val submitButton = findViewById<Button>(R.id.submitButton)

val responseTextView = findViewById<TextView>(R.id.textViewResponse)

// Set onClickListener for submit button

submitButton.setOnClickListener {

val name = nameEditText.text.toString().trim()

val email = emailEditText.text.toString().trim()

// Get selected gender

val selectedGenderId = genderRadioGroup.checkedRadioButtonId

val selectedGender = findViewById<RadioButton>(selectedGenderId)?.text.toString()

// Get selected hobbies

val hobbies = mutableListOf<String>()

if (readingCheckBox.isChecked) hobbies.add("Reading")

if (travelingCheckBox.isChecked) hobbies.add("Traveling")

if (sportsCheckBox.isChecked) hobbies.add("Sports")

// Get newsletter subscription status

val isSubscribed = if (newsletterToggleButton.isChecked) "Yes" else "No"

// Display user response

val response = """

Name: $name

Email: $email

Gender: $selectedGender

Hobbies: ${hobbies.joinToString(", ")}

Subscribe to Newsletter: $isSubscribed

"""

responseTextView.text = response

}

}

}

**Explanation of Kotlin Code**

1. **Views Initialization**:
   * We initialize the views by using findViewById() to access the EditText, RadioGroup, CheckBox, ToggleButton, and TextView.
2. **Handling Button Click**:
   * When the user clicks the **Submit** button, we retrieve the values entered in the form fields.
   * For RadioButton, we check the selected option in the RadioGroup to get the gender.
   * For CheckBox, we check which hobbies are selected.
   * The ToggleButton tells us whether the user subscribed to the newsletter (either "Yes" or "No").
3. **Displaying the Response**:
   * We use a TextView to display the collected information in a formatted manner.

**Step 4: Run the App**

1. **Connect a device or start an emulator**.
2. **Click the "Run" button** in Android Studio.
3. Once the app starts, fill in the registration form, select gender, hobbies, and toggle the newsletter option.
4. **Click the "Submit" button**, and the user's responses will be displayed in the TextView.

**Q. 14 Develop a app for demonstration of Registration form using various UI components such as Edittext, checkboxes, radiobuttons, Togglebutton ,and display users response using simple Toast**

**Step 1: Create a New Android Project**

1. **Open Android Studio**.
2. **Create a New Project**:
   * Choose **"Empty Activity"**.
   * Name your project (e.g., RegistrationApp).
   * Select **Kotlin** as the programming language.
   * Choose the appropriate **API level** (API 21+ is fine for most purposes).

**Step 2: Define the Layout with Various UI Components**

In the res/layout/activity\_main.xml file, define the layout for the registration form. Here, we'll include EditText, RadioButton, CheckBox, ToggleButton, and Button.

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp">

<!-- Name Input -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Name:"

android:textSize="18sp" />

<EditText

android:id="@+id/editTextName"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your name"

android:inputType="textPersonName"/>

<!-- Email Input -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Email:"

android:textSize="18sp" />

<EditText

android:id="@+id/editTextEmail"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your email"

android:inputType="textEmailAddress"/>

<!-- Gender Radio Buttons -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Gender:"

android:textSize="18sp" />

<RadioGroup

android:id="@+id/radioGroupGender"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<RadioButton

android:id="@+id/radioMale"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Male" />

<RadioButton

android:id="@+id/radioFemale"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Female" />

</RadioGroup>

<!-- Hobbies Checkboxes -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hobbies:"

android:textSize="18sp" />

<CheckBox

android:id="@+id/checkBoxReading"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Reading" />

<CheckBox

android:id="@+id/checkBoxTraveling"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Traveling" />

<CheckBox

android:id="@+id/checkBoxSports"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Sports" />

<!-- Subscribe to Newsletter Toggle Button -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Subscribe to Newsletter:"

android:textSize="18sp" />

<ToggleButton

android:id="@+id/toggleButtonNewsletter"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textOff="No"

android:textOn="Yes" />

<!-- Submit Button -->

<Button

android:id="@+id/submitButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Submit" />

</LinearLayout>

**Explanation of UI Components**

1. **EditText**:
   * Used for capturing the **name** and **email** from the user.
   * android:inputType="textPersonName" and android:inputType="textEmailAddress" ensure that the right keyboards appear.
2. **RadioGroup and RadioButton**:
   * The RadioGroup contains two RadioButton components (Male and Female) to allow gender selection.
3. **CheckBox**:
   * Multiple CheckBox components are used for hobbies (Reading, Traveling, and Sports).
4. **ToggleButton**:
   * A ToggleButton to allow the user to subscribe to the newsletter (Yes/No).
5. **Button**:
   * A Button to submit the form.

**Step 3: Handle User Input in Kotlin**

Now, let's implement the logic to handle the user input in the MainActivity.kt file. When the user clicks the **Submit** button, the app will display the user's inputs in a **Toast**.

kotlin

Copy code

package com.example.registrationapp

import android.os.Bundle

import android.widget.\*

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Initialize views

val nameEditText = findViewById<EditText>(R.id.editTextName)

val emailEditText = findViewById<EditText>(R.id.editTextEmail)

val genderRadioGroup = findViewById<RadioGroup>(R.id.radioGroupGender)

val readingCheckBox = findViewById<CheckBox>(R.id.checkBoxReading)

val travelingCheckBox = findViewById<CheckBox>(R.id.checkBoxTraveling)

val sportsCheckBox = findViewById<CheckBox>(R.id.checkBoxSports)

val newsletterToggleButton = findViewById<ToggleButton>(R.id.toggleButtonNewsletter)

val submitButton = findViewById<Button>(R.id.submitButton)

// Set onClickListener for submit button

submitButton.setOnClickListener {

val name = nameEditText.text.toString().trim()

val email = emailEditText.text.toString().trim()

// Get selected gender

val selectedGenderId = genderRadioGroup.checkedRadioButtonId

val selectedGender = findViewById<RadioButton>(selectedGenderId)?.text.toString()

// Get selected hobbies

val hobbies = mutableListOf<String>()

if (readingCheckBox.isChecked) hobbies.add("Reading")

if (travelingCheckBox.isChecked) hobbies.add("Traveling")

if (sportsCheckBox.isChecked) hobbies.add("Sports")

// Get newsletter subscription status

val isSubscribed = if (newsletterToggleButton.isChecked) "Yes" else "No"

// Display user response in a Toast

val response = """

Name: $name

Email: $email

Gender: $selectedGender

Hobbies: ${hobbies.joinToString(", ")}

Subscribe to Newsletter: $isSubscribed

"""

Toast.makeText(this, response, Toast.LENGTH\_LONG).show()

}

}

}

**Explanation of Kotlin Code**

1. **Views Initialization**:
   * We initialize the UI components using findViewById(). This includes EditText for name and email, RadioGroup for gender, CheckBox for hobbies, and ToggleButton for newsletter subscription.
2. **Handling Button Click**:
   * When the user clicks the **Submit** button, the app retrieves the values entered by the user in the input fields.
   * We check which RadioButton is selected for gender, which CheckBox options are selected for hobbies, and whether the ToggleButton is checked for the newsletter.
3. **Displaying User Response in a Toast**:
   * We use Toast.makeText() to display the response in a simple Toast message.
   * The Toast message includes the user's name, email, gender, hobbies, and newsletter subscription status.

**Step 4: Run the App**

1. **Connect a device or start an emulator**.
2. **Click the "Run" button** in Android Studio.
3. Once the app starts, fill in the registration form, select gender, hobbies, and toggle the newsletter option.
4. **Click the "Submit" button**, and the user's responses will be displayed in a **Toast**.

**Q. 15 Develop a app for demonstration of SMS and Telephony. For sending messages**

### Step 1: Create a New Android Project

1. **Open Android Studio**.
2. **Create a New Project**:
   * Choose **"Empty Activity"**.
   * Name your project (e.g., SMSSenderApp).
   * Select **Kotlin** as the programming language.
   * Choose the appropriate **API level** (API 21+ is fine for most purposes).

### Step 2: Update Android Manifest to Include SMS Permissions

To send an SMS from an app, you need to request the necessary permissions in your AndroidManifest.xml file. You will need the SEND\_SMS permission.

#### AndroidManifest.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.smssenderapp">

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="SMS Sender App"

android:theme="@style/Theme.SMSSenderApp">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

<!-- SMS Permission -->

<uses-permission android:name="android.permission.SEND\_SMS" />

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

</manifest>

### Step 3: Define the Layout

In the res/layout/activity\_main.xml, define the layout for sending an SMS. This will include:

* EditText for entering the phone number.
* EditText for entering the message text.
* Button to send the message.

#### activity\_main.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp">

<!-- Phone Number Input -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Phone Number:"

android:textSize="18sp" />

<EditText

android:id="@+id/editTextPhoneNumber"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter phone number"

android:inputType="phone"/>

<!-- Message Input -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Message:"

android:textSize="18sp" />

<EditText

android:id="@+id/editTextMessage"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your message"

android:inputType="textMultiLine"/>

<!-- Send SMS Button -->

<Button

android:id="@+id/buttonSendSMS"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Send SMS" />

</LinearLayout>

### Step 4: Handle User Input and Send SMS

In the MainActivity.kt file, implement the logic to send an SMS using the SmsManager class.

#### MainActivity.kt

kotlin

Copy code

package com.example.smssenderapp

import android.os.Bundle

import android.telephony.SmsManager

import android.widget.Button

import android.widget.EditText

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Get references to UI components

val phoneNumberEditText = findViewById<EditText>(R.id.editTextPhoneNumber)

val messageEditText = findViewById<EditText>(R.id.editTextMessage)

val sendSMSButton = findViewById<Button>(R.id.buttonSendSMS)

// Set the onClickListener for the send button

sendSMSButton.setOnClickListener {

val phoneNumber = phoneNumberEditText.text.toString().trim()

val message = messageEditText.text.toString().trim()

// Validate inputs

if (phoneNumber.isEmpty() || message.isEmpty()) {

Toast.makeText(this, "Please enter both phone number and message", Toast.LENGTH\_SHORT).show()

} else {

sendSMS(phoneNumber, message)

}

}

}

// Function to send SMS

private fun sendSMS(phoneNumber: String, message: String) {

try {

// Get the default SmsManager

val smsManager = SmsManager.getDefault()

smsManager.sendTextMessage(phoneNumber, null, message, null, null)

Toast.makeText(this, "Message sent!", Toast.LENGTH\_SHORT).show()

} catch (e: Exception) {

Toast.makeText(this, "Failed to send message: ${e.message}", Toast.LENGTH\_LONG).show()

}

}

}

### Step 5: Handle Runtime Permissions (Android 6.0 and Above)

From **Android 6.0 (API 23)**, you need to request permissions at runtime for actions like sending SMS. Add this code in MainActivity.kt to handle permissions:

#### Modify MainActivity.kt to request runtime permissions

kotlin

Copy code

import android.Manifest

import android.content.pm.PackageManager

import android.os.Build

import android.telephony.SmsManager

import android.widget.Button

import android.widget.EditText

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

import androidx.core.app.ActivityCompat

class MainActivity : AppCompatActivity() {

private val SEND\_SMS\_PERMISSION\_CODE = 101

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

val phoneNumberEditText = findViewById<EditText>(R.id.editTextPhoneNumber)

val messageEditText = findViewById<EditText>(R.id.editTextMessage)

val sendSMSButton = findViewById<Button>(R.id.buttonSendSMS)

// Check permission for SMS

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {

if (ActivityCompat.checkSelfPermission(

this,

Manifest.permission.SEND\_SMS

) != PackageManager.PERMISSION\_GRANTED

) {

ActivityCompat.requestPermissions(

this,

arrayOf(Manifest.permission.SEND\_SMS),

SEND\_SMS\_PERMISSION\_CODE

)

}

}

// Set the onClickListener for the send button

sendSMSButton.setOnClickListener {

val phoneNumber = phoneNumberEditText.text.toString().trim()

val message = messageEditText.text.toString().trim()

if (phoneNumber.isEmpty() || message.isEmpty()) {

Toast.makeText(this, "Please enter both phone number and message", Toast.LENGTH\_SHORT).show()

} else {

sendSMS(phoneNumber, message)

}

}

}

// Function to send SMS

private fun sendSMS(phoneNumber: String, message: String) {

try {

val smsManager = SmsManager.getDefault()

smsManager.sendTextMessage(phoneNumber, null, message, null, null)

Toast.makeText(this, "Message sent!", Toast.LENGTH\_SHORT).show()

} catch (e: Exception) {

Toast.makeText(this, "Failed to send message: ${e.message}", Toast.LENGTH\_LONG).show()

}

}

// Handle permission result

override fun onRequestPermissionsResult(

requestCode: Int,

permissions: Array<out String>,

grantResults: IntArray

) {

super.onRequestPermissionsResult(requestCode, permissions, grantResults)

if (requestCode == SEND\_SMS\_PERMISSION\_CODE) {

if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {

Toast.makeText(this, "Permission granted", Toast.LENGTH\_SHORT).show()

} else {

Toast.makeText(this, "Permission denied", Toast.LENGTH\_SHORT).show()

}

}

}

}

### Explanation

1. **Permissions**:
   * The app requires the SEND\_SMS permission to send SMS messages.
   * From Android 6.0 (API 23) and above, you need to request permissions at runtime.
2. **Sending SMS**:
   * The SmsManager class is used to send the SMS.
   * The phone number and message are retrieved from the input fields and sent to the sendSMS function.
3. **Runtime Permissions**:
   * Before sending the SMS, the app checks if the required permission (SEND\_SMS) is granted.
   * If not, it requests the permission from the user at runtime.

### Step 6: Run the App

1. **Connect a device or start an emulator**.
2. **Click the "Run" button** in Android Studio.
3. When the app launches, enter a phone number and a message, then press **Send SMS**.

**Q. 16 Program to demonstrate Buttons, Text Fields, Checkboxes, Radio Buttons, and Toggle Buttons with their events handler.**

### Step 1: Create a New Android Project

1. **Open Android Studio**.
2. **Create a New Project** with an **Empty Activity**.
3. Name your project (e.g., UIComponentsDemo).
4. Select **Kotlin** as the language.

### Step 2: Define the Layout (activity\_main.xml)

In this layout, we will use:

* **EditText** for entering text.
* **Button** to trigger actions.
* **Checkbox** for selecting options.
* **RadioButtons** for selecting one option from a set.
* **ToggleButton** for switching between two states.

#### activity\_main.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="20dp">

<!-- Text Field (EditText) -->

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Enter some text:"

android:textSize="18sp"/>

<EditText

android:id="@+id/editTextText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Type something here"

android:inputType="text"/>

<!-- Button -->

<Button

android:id="@+id/buttonShowText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Show Text"

android:textSize="18sp"/>

<!-- Checkboxes -->

<TextView

android:id="@+id/checkboxLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Select options:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<CheckBox

android:id="@+id/checkBox1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Option 1" />

<CheckBox

android:id="@+id/checkBox2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Option 2" />

<!-- Radio Buttons -->

<TextView

android:id="@+id/radioButtonLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Choose an option:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<RadioGroup

android:id="@+id/radioGroup"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<RadioButton

android:id="@+id/radioButton1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Option A" />

<RadioButton

android:id="@+id/radioButton2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Option B" />

</RadioGroup>

<!-- Toggle Button -->

<TextView

android:id="@+id/toggleButtonLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Toggle the switch:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<ToggleButton

android:id="@+id/toggleButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textOn="ON"

android:textOff="OFF"/>

<!-- Output Text -->

<TextView

android:id="@+id/outputText"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Your selected options will appear here."

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

</LinearLayout>

### Step 3: Implement Logic in MainActivity (MainActivity.kt)

In the MainActivity.kt, we'll handle the events of the UI components: buttons, checkboxes, radio buttons, and toggle buttons. When the user interacts with these components, the app will display their responses.

#### MainActivity.kt

kotlin

Copy code

package com.example.uicomponentsdemo

import android.os.Bundle

import android.widget.\*

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// References to UI components

val editText = findViewById<EditText>(R.id.editTextText)

val buttonShowText = findViewById<Button>(R.id.buttonShowText)

val checkBox1 = findViewById<CheckBox>(R.id.checkBox1)

val checkBox2 = findViewById<CheckBox>(R.id.checkBox2)

val radioGroup = findViewById<RadioGroup>(R.id.radioGroup)

val toggleButton = findViewById<ToggleButton>(R.id.toggleButton)

val outputText = findViewById<TextView>(R.id.outputText)

// Show entered text when button is clicked

buttonShowText.setOnClickListener {

val enteredText = editText.text.toString()

outputText.text = "Entered Text: $enteredText"

}

// Handle checkbox events

checkBox1.setOnCheckedChangeListener { \_, isChecked ->

val selectedOptions = StringBuilder()

if (checkBox1.isChecked) selectedOptions.append("Option 1 is selected\n")

if (checkBox2.isChecked) selectedOptions.append("Option 2 is selected\n")

outputText.text = selectedOptions.toString()

}

// Handle radio button selection

radioGroup.setOnCheckedChangeListener { \_, checkedId ->

val selectedRadioButton = findViewById<RadioButton>(checkedId)

outputText.text = "Selected Radio Button: ${selectedRadioButton.text}"

}

// Handle toggle button state change

toggleButton.setOnCheckedChangeListener { \_, isChecked ->

if (isChecked) {

outputText.text = "Toggle Button is ON"

} else {

outputText.text = "Toggle Button is OFF"

}

}

}

}

### Explanation:

1. **EditText (Text Field)**:
   * The EditText allows users to type a message.
   * When the button (buttonShowText) is clicked, the text entered is retrieved and displayed in the outputText TextView.
2. **Checkboxes**:
   * checkBox1 and checkBox2 let users select options.
   * An OnCheckedChangeListener is used to detect when a checkbox is checked or unchecked.
   * The selected options are shown in the outputText.
3. **Radio Buttons**:
   * radioGroup contains RadioButton options.
   * An OnCheckedChangeListener detects which radio button is selected, and the selection is displayed in the outputText.
4. **Toggle Button**:
   * The toggleButton allows users to switch between two states (ON/OFF).
   * The state change is detected using setOnCheckedChangeListener and updates the outputText.

### Step 4: Run the App

1. **Connect your device or start an emulator**.
2. **Click "Run"** in Android Studio to launch the app.
3. The app will display the text fields, buttons, checkboxes, radio buttons, and toggle buttons. Interacting with these UI components will trigger the event handlers and update the output text.

**Q. 17 Develop a app for demonstration of Login form using various UI components such as Edittext, checkboxes, radiobuttons, Togglebutton ,and display users response using simple Toast**

### Step 1: Create a New Android Project

1. **Open Android Studio**.
2. **Create a New Project** with an **Empty Activity**.
3. Choose **Kotlin** as the language and name the project (e.g., LoginFormDemo).

### Step 2: Define the Layout (activity\_main.xml)

Here, we will include:

* **EditText** for the user to input their username and password.
* **Checkbox** to remember the username (optional).
* **RadioButton** for selecting user type (Admin or User).
* **ToggleButton** to enable or disable the login button.
* **Button** for submitting the form.

#### activity\_main.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="20dp"

android:gravity="center">

<!-- Username Field -->

<TextView

android:id="@+id/usernameLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Username:"

android:textSize="18sp"/>

<EditText

android:id="@+id/usernameEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your username"

android:inputType="text"/>

<!-- Password Field -->

<TextView

android:id="@+id/passwordLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Password:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<EditText

android:id="@+id/passwordEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your password"

android:inputType="textPassword"/>

<!-- Remember Me Checkbox -->

<CheckBox

android:id="@+id/rememberMeCheckbox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Remember me"

android:layout\_marginTop="20dp"/>

<!-- User Type Radio Buttons -->

<TextView

android:id="@+id/userTypeLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Select User Type:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<RadioGroup

android:id="@+id/userTypeGroup"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginTop="10dp">

<RadioButton

android:id="@+id/radioAdmin"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Admin"/>

<RadioButton

android:id="@+id/radioUser"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="User"/>

</RadioGroup>

<!-- Toggle Button to Enable/Disable Login Button -->

<TextView

android:id="@+id/toggleButtonLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Enable Login:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<ToggleButton

android:id="@+id/toggleLoginButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textOn="Enabled"

android:textOff="Disabled"/>

<!-- Login Button -->

<Button

android:id="@+id/loginButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Login"

android:layout\_marginTop="20dp"/>

</LinearLayout>

### Step 3: Implement Logic in MainActivity.kt

In this Kotlin file, we will handle the user's input and interactions with the UI components, such as:

* **Username and Password** (using EditText).
* **Checkbox** (Remember me option).
* **RadioButton** (User Type selection).
* **ToggleButton** (Enable or disable the login button).
* Display the response via a **Toast** message.

#### MainActivity.kt

kotlin

Copy code

package com.example.loginformdemo

import android.os.Bundle

import android.widget.\*

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Get references to UI components

val usernameEditText = findViewById<EditText>(R.id.usernameEditText)

val passwordEditText = findViewById<EditText>(R.id.passwordEditText)

val rememberMeCheckbox = findViewById<CheckBox>(R.id.rememberMeCheckbox)

val radioAdmin = findViewById<RadioButton>(R.id.radioAdmin)

val radioUser = findViewById<RadioButton>(R.id.radioUser)

val toggleLoginButton = findViewById<ToggleButton>(R.id.toggleLoginButton)

val loginButton = findViewById<Button>(R.id.loginButton)

// Handle ToggleButton to enable/disable Login Button

toggleLoginButton.setOnCheckedChangeListener { \_, isChecked ->

loginButton.isEnabled = isChecked

}

// Set the Login Button's OnClickListener

loginButton.setOnClickListener {

// Get user inputs

val username = usernameEditText.text.toString()

val password = passwordEditText.text.toString()

val rememberMe = rememberMeCheckbox.isChecked

val userType = if (radioAdmin.isChecked) "Admin" else "User"

// Prepare the message

val message = StringBuilder()

message.append("Username: $username\n")

message.append("Password: $password\n")

message.append("Remember Me: $rememberMe\n")

message.append("User Type: $userType")

// Show response in a Toast

Toast.makeText(this, message.toString(), Toast.LENGTH\_LONG).show()

}

}

}

### Explanation:

1. **Username and Password Fields (EditText)**:
   * We use EditText for both username and password fields. The user enters their credentials here.
2. **Remember Me Checkbox**:
   * The CheckBox allows the user to opt for "Remember Me". If checked, it indicates that the user prefers their login credentials to be saved.
3. **User Type (RadioButtons)**:
   * We use a RadioGroup containing two RadioButtons (Admin and User) for selecting the user type.
   * Based on the selection, the user's type is stored and displayed in the Toast message.
4. **Login Button (Button)**:
   * The Button is initially enabled or disabled based on the state of the ToggleButton. When clicked, it collects and displays the input data as a Toast message.
5. **Toggle Button**:
   * The ToggleButton allows enabling or disabling the Login button. When checked, the Login button becomes enabled, and when unchecked, the Login button is disabled.

### Step 4: Run the App

1. **Connect your device or start an emulator**.
2. **Click "Run"** in Android Studio to launch the app.
3. The app will display the login form, and when the user enters their details and presses the login button, a **Toast** will appear showing the entered information.

**Q.18 Develop a app for demonstration of Login form using various UI components such as Edittext, checkboxes, radiobuttons, Togglebutton ,and display users response in Textview**

### Step 1: Create a New Android Project

1. **Open Android Studio**.
2. **Create a New Project** with an **Empty Activity**.
3. Choose **Kotlin** as the language and name the project (e.g., LoginFormWithTextView).

### Step 2: Define the Layout (activity\_main.xml)

Here, we will include:

* **EditText** for the user to input their username and password.
* **Checkbox** to remember the username (optional).
* **RadioButton** for selecting user type (Admin or User).
* **ToggleButton** to enable or disable the login button.
* **Button** for submitting the form.
* **TextView** to display the user's response.

#### activity\_main.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="20dp"

android:gravity="center">

<!-- Username Field -->

<TextView

android:id="@+id/usernameLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Username:"

android:textSize="18sp"/>

<EditText

android:id="@+id/usernameEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your username"

android:inputType="text"/>

<!-- Password Field -->

<TextView

android:id="@+id/passwordLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Password:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<EditText

android:id="@+id/passwordEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your password"

android:inputType="textPassword"/>

<!-- Remember Me Checkbox -->

<CheckBox

android:id="@+id/rememberMeCheckbox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Remember me"

android:layout\_marginTop="20dp"/>

<!-- User Type Radio Buttons -->

<TextView

android:id="@+id/userTypeLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Select User Type:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<RadioGroup

android:id="@+id/userTypeGroup"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginTop="10dp">

<RadioButton

android:id="@+id/radioAdmin"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Admin"/>

<RadioButton

android:id="@+id/radioUser"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="User"/>

</RadioGroup>

<!-- Toggle Button to Enable/Disable Login Button -->

<TextView

android:id="@+id/toggleButtonLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Enable Login:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<ToggleButton

android:id="@+id/toggleLoginButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textOn="Enabled"

android:textOff="Disabled"/>

<!-- Login Button -->

<Button

android:id="@+id/loginButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Login"

android:layout\_marginTop="20dp"/>

<!-- TextView to Display User Response -->

<TextView

android:id="@+id/responseTextView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="User Response Will Appear Here"

android:textSize="18sp"

android:layout\_marginTop="20dp"

android:textColor="@android:color/black"/>

</LinearLayout>

### Step 3: Implement the Logic in MainActivity.kt

In this Kotlin file, we will handle the user's input and interactions with the UI components, such as:

* **Username and Password** (using EditText).
* **Checkbox** (Remember me option).
* **RadioButton** (User Type selection).
* **ToggleButton** (Enable or disable the login button).
* Display the response via **TextView**.

#### MainActivity.kt

kotlin

Copy code

package com.example.loginformwithtextview

import android.os.Bundle

import android.widget.\*

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Get references to UI components

val usernameEditText = findViewById<EditText>(R.id.usernameEditText)

val passwordEditText = findViewById<EditText>(R.id.passwordEditText)

val rememberMeCheckbox = findViewById<CheckBox>(R.id.rememberMeCheckbox)

val radioAdmin = findViewById<RadioButton>(R.id.radioAdmin)

val radioUser = findViewById<RadioButton>(R.id.radioUser)

val toggleLoginButton = findViewById<ToggleButton>(R.id.toggleLoginButton)

val loginButton = findViewById<Button>(R.id.loginButton)

val responseTextView = findViewById<TextView>(R.id.responseTextView)

// Handle ToggleButton to enable/disable Login Button

toggleLoginButton.setOnCheckedChangeListener { \_, isChecked ->

loginButton.isEnabled = isChecked

}

// Set the Login Button's OnClickListener

loginButton.setOnClickListener {

// Get user inputs

val username = usernameEditText.text.toString()

val password = passwordEditText.text.toString()

val rememberMe = rememberMeCheckbox.isChecked

val userType = if (radioAdmin.isChecked) "Admin" else "User"

// Prepare the message

val message = StringBuilder()

message.append("Username: $username\n")

message.append("Password: $password\n")

message.append("Remember Me: $rememberMe\n")

message.append("User Type: $userType")

// Display response in the TextView

responseTextView.text = message.toString()

}

}

}

### Explanation:

1. **Username and Password Fields (EditText)**:
   * EditText is used for both the username and password fields. The user enters their credentials here.
2. **Remember Me Checkbox**:
   * The CheckBox allows the user to opt for "Remember Me". If checked, it indicates that the user prefers their login credentials to be saved.
3. **User Type (RadioButtons)**:
   * We use a RadioGroup containing two RadioButtons (Admin and User) for selecting the user type.
   * Based on the selection, the user's type is stored and displayed in the TextView.
4. **Login Button (Button)**:
   * The Button is initially enabled or disabled based on the state of the ToggleButton. When clicked, it collects and displays the input data in the TextView.
5. **Toggle Button**:
   * The ToggleButton allows enabling or disabling the Login button. When checked, the Login button becomes enabled, and when unchecked, the Login button is disabled.
6. **TextView for Response**:
   * A TextView is used to display the user’s responses after clicking the login button, such as the username, password, whether "Remember Me" is checked, and the selected user type.

### Step 4: Run the App

1. **Connect your device or start an emulator**.
2. **Click "Run"** in Android Studio to launch the app.
3. The app will display the login form, and when the user enters their details and presses the login button, their responses will be displayed in the **TextView**.

**Q.19 Develop a app for demonstration of Contact us form using various UI components such as Edittext, checkboxes, radiobuttons, Togglebutton ,and display users response using simple Toast**

### Step 1: Create a New Android Project

1. **Open Android Studio**.
2. **Create a New Project** with an **Empty Activity**.
3. Choose **Kotlin** as the language and name the project (e.g., ContactUsFormApp).

### Step 2: Define the Layout (activity\_main.xml)

Here, we will include:

* **EditText** for the user to input their name, email, and message.
* **Checkbox** for the user to accept terms and conditions.
* **RadioButton** for selecting the contact method (Phone or Email).
* **ToggleButton** to enable or disable the submit button.
* **Button** to submit the form.
* **TextView** to provide a heading or description for the form.

#### activity\_main.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="20dp"

android:gravity="center">

<!-- Heading TextView -->

<TextView

android:id="@+id/contactUsLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Contact Us"

android:textSize="24sp"

android:layout\_marginBottom="20dp"

android:gravity="center"/>

<!-- Name Field -->

<TextView

android:id="@+id/nameLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Name:"

android:textSize="18sp"/>

<EditText

android:id="@+id/nameEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your name"

android:inputType="textPersonName"/>

<!-- Email Field -->

<TextView

android:id="@+id/emailLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Email:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<EditText

android:id="@+id/emailEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your email"

android:inputType="textEmailAddress"/>

<!-- Message Field -->

<TextView

android:id="@+id/messageLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Message:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<EditText

android:id="@+id/messageEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your message"

android:inputType="textMultiLine"/>

<!-- Accept Terms Checkbox -->

<CheckBox

android:id="@+id/acceptTermsCheckbox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="I accept the terms and conditions"

android:layout\_marginTop="20dp"/>

<!-- Contact Method Radio Buttons -->

<TextView

android:id="@+id/contactMethodLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Preferred Contact Method:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<RadioGroup

android:id="@+id/contactMethodGroup"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginTop="10dp">

<RadioButton

android:id="@+id/radioPhone"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Phone"/>

<RadioButton

android:id="@+id/radioEmail"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Email"/>

</RadioGroup>

<!-- Toggle Button to Enable/Disable Submit Button -->

<TextView

android:id="@+id/toggleSubmitButtonLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Enable Submit Button:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<ToggleButton

android:id="@+id/toggleSubmitButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textOn="Enabled"

android:textOff="Disabled"/>

<!-- Submit Button -->

<Button

android:id="@+id/submitButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Submit"

android:layout\_marginTop="20dp"/>

</LinearLayout>

### Step 3: Implement the Logic in MainActivity.kt

In this Kotlin file, we handle the user's input and interactions with the UI components, such as:

* **Name**, **Email**, and **Message** fields (using EditText).
* **Accept Terms** (using CheckBox).
* **Preferred Contact Method** (using RadioButton).
* **Enable/Disable Submit Button** (using ToggleButton).
* **Submit Button** to handle the form submission and display a Toast message.

#### MainActivity.kt

kotlin

Copy code

package com.example.contactusformapp

import android.os.Bundle

import android.widget.\*

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Get references to UI components

val nameEditText = findViewById<EditText>(R.id.nameEditText)

val emailEditText = findViewById<EditText>(R.id.emailEditText)

val messageEditText = findViewById<EditText>(R.id.messageEditText)

val acceptTermsCheckbox = findViewById<CheckBox>(R.id.acceptTermsCheckbox)

val radioPhone = findViewById<RadioButton>(R.id.radioPhone)

val radioEmail = findViewById<RadioButton>(R.id.radioEmail)

val toggleSubmitButton = findViewById<ToggleButton>(R.id.toggleSubmitButton)

val submitButton = findViewById<Button>(R.id.submitButton)

// Handle ToggleButton to enable/disable Submit Button

toggleSubmitButton.setOnCheckedChangeListener { \_, isChecked ->

submitButton.isEnabled = isChecked

}

// Set the Submit Button's OnClickListener

submitButton.setOnClickListener {

// Get user inputs

val name = nameEditText.text.toString()

val email = emailEditText.text.toString()

val message = messageEditText.text.toString()

val acceptTerms = acceptTermsCheckbox.isChecked

val contactMethod = if (radioPhone.isChecked) "Phone" else "Email"

// Check if the terms are accepted

if (!acceptTerms) {

Toast.makeText(this, "You must accept the terms and conditions.", Toast.LENGTH\_SHORT).show()

return@setOnClickListener

}

// Prepare the message to display in Toast

val userMessage = """

Name: $name

Email: $email

Message: $message

Preferred Contact Method: $contactMethod

""".trimIndent()

// Display the response in a Toast

Toast.makeText(this, userMessage, Toast.LENGTH\_LONG).show()

}

}

}

### Explanation:

1. **Name, Email, and Message Fields (EditText)**:
   * EditText is used for the user to input their name, email, and message.
2. **Accept Terms Checkbox**:
   * The CheckBox asks the user to accept the terms and conditions before submitting the form. If not checked, a Toast message alerts the user to accept the terms.
3. **Preferred Contact Method (RadioButtons)**:
   * The RadioGroup contains two RadioButton options, Phone and Email. Based on the user selection, we display the preferred contact method.
4. **Enable/Disable Submit Button (ToggleButton)**:
   * The ToggleButton enables or disables the Submit button. When the ToggleButton is on, the Submit button is enabled; when off, the Submit button is disabled.
5. **Submit Button (Button)**:
   * When the user clicks the **Submit** button, we collect all inputs and show a **Toast** message displaying the information.

### Step 4: Run the App

1. **Connect your device or start an emulator**.
2. **Click "Run"** in Android Studio to launch the app.
3. The app will display the **Contact Us** form. When the user fills in the form and submits, the app will display their input via a **Toast** message.

**Q20. Develop a app for demonstration of Contact us form using various UI components such as Edittext, checkboxes, radiobuttons, Togglebutton ,and display users response in Textview**

### Step 1: Create a New Android Project

1. **Open Android Studio**.
2. **Create a New Project** with an **Empty Activity**.
3. Choose **Kotlin** as the language and name the project (e.g., ContactUsFormApp).

### Step 2: Define the Layout (activity\_main.xml)

This layout includes:

* **EditText** for the user to input their name, email, and message.
* **Checkbox** for accepting terms and conditions.
* **RadioButton** for choosing the preferred contact method.
* **ToggleButton** to enable or disable the submit button.
* **TextView** to display the user's response after submitting the form.

#### activity\_main.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="20dp"

android:gravity="center">

<!-- Heading TextView -->

<TextView

android:id="@+id/contactUsLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Contact Us"

android:textSize="24sp"

android:layout\_marginBottom="20dp"

android:gravity="center"/>

<!-- Name Field -->

<TextView

android:id="@+id/nameLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Name:"

android:textSize="18sp"/>

<EditText

android:id="@+id/nameEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your name"

android:inputType="textPersonName"/>

<!-- Email Field -->

<TextView

android:id="@+id/emailLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Email:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<EditText

android:id="@+id/emailEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your email"

android:inputType="textEmailAddress"/>

<!-- Message Field -->

<TextView

android:id="@+id/messageLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Message:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<EditText

android:id="@+id/messageEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your message"

android:inputType="textMultiLine"/>

<!-- Accept Terms Checkbox -->

<CheckBox

android:id="@+id/acceptTermsCheckbox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="I accept the terms and conditions"

android:layout\_marginTop="20dp"/>

<!-- Contact Method Radio Buttons -->

<TextView

android:id="@+id/contactMethodLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Preferred Contact Method:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<RadioGroup

android:id="@+id/contactMethodGroup"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginTop="10dp">

<RadioButton

android:id="@+id/radioPhone"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Phone"/>

<RadioButton

android:id="@+id/radioEmail"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Email"/>

</RadioGroup>

<!-- Toggle Button to Enable/Disable Submit Button -->

<TextView

android:id="@+id/toggleSubmitButtonLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Enable Submit Button:"

android:textSize="18sp"

android:layout\_marginTop="20dp"/>

<ToggleButton

android:id="@+id/toggleSubmitButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textOn="Enabled"

android:textOff="Disabled"/>

<!-- Submit Button -->

<Button

android:id="@+id/submitButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Submit"

android:layout\_marginTop="20dp"/>

<!-- TextView to display user response -->

<TextView

android:id="@+id/responseTextView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Your response will appear here."

android:textSize="16sp"

android:layout\_marginTop="20dp"/>

</LinearLayout>

### Step 3: Implement Logic in MainActivity.kt

This Kotlin file will handle the form submission logic and display the user's response in the TextView.

#### MainActivity.kt

kotlin

Copy code

package com.example.contactusformapp

import android.os.Bundle

import android.widget.\*

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Get references to UI components

val nameEditText = findViewById<EditText>(R.id.nameEditText)

val emailEditText = findViewById<EditText>(R.id.emailEditText)

val messageEditText = findViewById<EditText>(R.id.messageEditText)

val acceptTermsCheckbox = findViewById<CheckBox>(R.id.acceptTermsCheckbox)

val radioPhone = findViewById<RadioButton>(R.id.radioPhone)

val radioEmail = findViewById<RadioButton>(R.id.radioEmail)

val toggleSubmitButton = findViewById<ToggleButton>(R.id.toggleSubmitButton)

val submitButton = findViewById<Button>(R.id.submitButton)

val responseTextView = findViewById<TextView>(R.id.responseTextView)

// Handle ToggleButton to enable/disable Submit Button

toggleSubmitButton.setOnCheckedChangeListener { \_, isChecked ->

submitButton.isEnabled = isChecked

}

// Set the Submit Button's OnClickListener

submitButton.setOnClickListener {

// Get user inputs

val name = nameEditText.text.toString()

val email = emailEditText.text.toString()

val message = messageEditText.text.toString()

val acceptTerms = acceptTermsCheckbox.isChecked

val contactMethod = if (radioPhone.isChecked) "Phone" else "Email"

// Check if the terms are accepted

if (!acceptTerms) {

Toast.makeText(this, "You must accept the terms and conditions.", Toast.LENGTH\_SHORT).show()

return@setOnClickListener

}

// Prepare the message to display in TextView

val userMessage = """

Name: $name

Email: $email

Message: $message

Preferred Contact Method: $contactMethod

""".trimIndent()

// Display the response in the TextView

responseTextView.text = userMessage

}

}

}

### Explanation:

1. **Name, Email, and Message Fields (EditText)**:
   * Users can input their name, email, and message using EditText.
2. **Accept Terms Checkbox (CheckBox)**:
   * Users must check the checkbox to accept terms and conditions before submitting the form.
3. **Preferred Contact Method (RadioButtons)**:
   * Users can choose their preferred contact method (Phone or Email) using RadioButton.
4. **Enable/Disable Submit Button (ToggleButton)**:
   * The ToggleButton enables or disables the **Submit** button. If the button is disabled, the user cannot submit the form.
5. **Submit Button (Button)**:
   * When the user clicks the **Submit** button, it collects all inputs and displays the result in a **TextView**.
6. **Response Display (TextView)**:
   * After the user submits the form, the data is displayed in the TextView.

### Step 4: Run the App

1. **Connect your device or start an emulator**.
2. **Click "Run"** in Android Studio to launch the app.
3. The app will display the **Contact Us** form. When the user fills in the form and submits, the app will display their input in a **TextView**.

**Q 21 Develop a app for demonstration of Implicit intent which shows the communication between two activities in same app**

### Step 1: Create a New Android Project

1. **Open Android Studio**.
2. **Create a New Project** with an **Empty Activity**.
3. Choose **Kotlin** as the language and name the project (e.g., ImplicitIntentDemo).

### Step 2: Define the Layout for MainActivity (activity\_main.xml)

Create a simple UI in MainActivity with an **EditText** to enter the name and a **Button** to send the name to SecondActivity.

#### activity\_main.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="20dp"

android:gravity="center">

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Enter your name:"

android:textSize="18sp" />

<EditText

android:id="@+id/nameEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Name"

android:inputType="textPersonName"

android:layout\_marginTop="10dp" />

<Button

android:id="@+id/sendButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Send to Second Activity"

android:layout\_marginTop="20dp" />

</LinearLayout>

### Step 3: Define the Layout for SecondActivity (activity\_second.xml)

In SecondActivity, display a **TextView** to show the name received from MainActivity.

#### activity\_second.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="20dp"

android:gravity="center">

<TextView

android:id="@+id/receivedTextView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello!"

android:textSize="24sp"

android:gravity="center"/>

</LinearLayout>

### Step 4: Create the MainActivity Class

In MainActivity.kt, set up an implicit intent to launch SecondActivity. We’ll also add code to pass the user’s input to the second activity.

#### MainActivity.kt

kotlin

Copy code

package com.example.implicitintentdemo

import android.content.Intent

import android.os.Bundle

import android.widget.Button

import android.widget.EditText

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Get references to UI components

val nameEditText = findViewById<EditText>(R.id.nameEditText)

val sendButton = findViewById<Button>(R.id.sendButton)

// Set click listener on the send button

sendButton.setOnClickListener {

val name = nameEditText.text.toString()

// Create an implicit intent to open SecondActivity

val intent = Intent("com.example.implicitintentdemo.SHOW\_NAME")

intent.putExtra("USER\_NAME", name) // Pass the entered name

// Start the activity

startActivity(intent)

}

}

}

### Step 5: Create the SecondActivity Class

In SecondActivity.kt, retrieve the name from the intent and display it in the **TextView**.

#### SecondActivity.kt

kotlin

Copy code

package com.example.implicitintentdemo

import android.os.Bundle

import android.widget.TextView

import androidx.appcompat.app.AppCompatActivity

class SecondActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_second)

// Get the name from the intent

val name = intent.getStringExtra("USER\_NAME")

// Get reference to TextView and set the received name

val receivedTextView = findViewById<TextView>(R.id.receivedTextView)

receivedTextView.text = "Hello, $name!"

}

}

### Step 6: Register the SecondActivity and Intent Filter in AndroidManifest.xml

To use an implicit intent, define an **Intent Filter** for SecondActivity in AndroidManifest.xml so that it listens for a custom action.

#### AndroidManifest.xml

xml

Copy code

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.implicitintentdemo">

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/Theme.ImplicitIntentDemo">

<activity android:name=".SecondActivity">

<intent-filter>

<action android:name="com.example.implicitintentdemo.SHOW\_NAME" />

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

</activity>

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

### Explanation

1. **MainActivity**:
   * Contains an **EditText** for the user to input their name.
   * When the **Send Button** is clicked, an **implicit intent** is created with a custom action (com.example.implicitintentdemo.SHOW\_NAME).
   * The entered name is passed to SecondActivity using putExtra().
2. **SecondActivity**:
   * Contains a **TextView** to display the received name.
   * Retrieves the name from the intent using getStringExtra() and sets it to the TextView.
3. **Intent Filter**:
   * The **Intent Filter** in AndroidManifest.xml allows SecondActivity to respond to the implicit intent with the custom action name com.example.implicitintentdemo.SHOW\_NAME.

### Step 7: Run the App

1. **Connect your device** or start an **Android emulator**.
2. **Click "Run"** in Android Studio.
3. The app will launch with the **MainActivity** screen.
4. Enter a name, click **Send to Second Activity**, and you’ll be taken to SecondActivity, where the entered name will be displayed.

**Q 22 Develop a app for demonstration of Explicit intent which shows the communication between to the apple website and your app**

### Step 1: Create a New Android Project

1. **Open Android Studio**.
2. **Create a New Project** with an **Empty Activity**.
3. Choose **Kotlin** or **Java** as the language and name the project (e.g., ExplicitIntentDemo).

### Step 2: Define the Layout for MainActivity (activity\_main.xml)

In activity\_main.xml, create a simple UI with a **Button**. When this button is clicked, it will open the Apple website in a browser.

#### activity\_main.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="20dp"

android:gravity="center">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Visit Apple Website"

android:textSize="18sp"

android:layout\_marginBottom="20dp" />

<Button

android:id="@+id/openWebsiteButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Open Apple Website" />

</LinearLayout>

### Step 3: Create the MainActivity Class

In MainActivity.kt or MainActivity.java, set up an **Explicit Intent** to open the Apple website in the browser when the button is clicked.

#### MainActivity.kt (Kotlin)

kotlin

Copy code

package com.example.explicitintentdemo

import android.content.Intent

import android.net.Uri

import android.os.Bundle

import android.widget.Button

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Get reference to the button

val openWebsiteButton = findViewById<Button>(R.id.openWebsiteButton)

// Set click listener for the button

openWebsiteButton.setOnClickListener {

// Create an explicit intent to open a URL in the browser

val appleWebsiteUri = Uri.parse("https://www.apple.com")

val intent = Intent(Intent.ACTION\_VIEW, appleWebsiteUri)

// Start the intent to open the URL in the browser

startActivity(intent)

}

}

}

#### MainActivity.java (Java)

java

Copy code

package com.example.explicitintentdemo;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Get reference to the button

Button openWebsiteButton = findViewById(R.id.openWebsiteButton);

// Set click listener for the button

openWebsiteButton.setOnClickListener(view -> {

// Create an explicit intent to open a URL in the browser

Uri appleWebsiteUri = Uri.parse("https://www.apple.com");

Intent intent = new Intent(Intent.ACTION\_VIEW, appleWebsiteUri);

// Start the intent to open the URL in the browser

startActivity(intent);

});

}

}

### Explanation

1. **MainActivity**:
   * **Button**: The layout contains a button labeled "Open Apple Website".
   * **Intent**: In MainActivity, when the button is clicked, an **Intent** is created with the action Intent.ACTION\_VIEW and the URI for the Apple website (https://www.apple.com).
   * **startActivity**: This intent is then used with startActivity(intent), which opens the Apple website in the device’s default browser.

### Step 4: Run the App

1. **Connect your device** or start an **Android emulator**.
2. **Click "Run"** in Android Studio.
3. Once the app launches:
   * Click the **Open Apple Website** button, and it should open the Apple website in the device’s default browser.

**Q 23 Develop a app for demonstration of Simple toast implementation in app use atleast five UI elements**

### Step 1: Create a New Android Project

1. **Open Android Studio**.
2. **Create a New Project** with an **Empty Activity**.
3. Choose **Kotlin** or **Java** as the language and name the project (e.g., ToastDemoApp).

### Step 2: Define the Layout for MainActivity (activity\_main.xml)

In activity\_main.xml, add various UI components (EditText, Button, Checkbox, RadioButton, and ToggleButton).

#### activity\_main.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="20dp"

android:gravity="center">

<!-- EditText for inputting name -->

<EditText

android:id="@+id/nameEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your name"

android:layout\_marginBottom="10dp" />

<!-- Button to show name -->

<Button

android:id="@+id/showNameButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Show Name"

android:layout\_marginBottom="10dp" />

<!-- Checkbox for terms agreement -->

<CheckBox

android:id="@+id/agreeCheckBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="I agree to the terms and conditions"

android:layout\_marginBottom="10dp" />

<!-- RadioGroup with two RadioButtons -->

<RadioGroup

android:id="@+id/genderRadioGroup"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginBottom="10dp">

<RadioButton

android:id="@+id/maleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Male" />

<RadioButton

android:id="@+id/femaleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Female" />

</RadioGroup>

<!-- ToggleButton for notifications -->

<ToggleButton

android:id="@+id/notificationToggleButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textOn="Notifications ON"

android:textOff="Notifications OFF"

android:layout\_marginBottom="10dp" />

</LinearLayout>

### Step 3: Create the MainActivity Class

In MainActivity.kt or MainActivity.java, set up event listeners for each UI element and use Toast messages to display feedback when the user interacts with them.

#### MainActivity.kt (Kotlin)

kotlin

Copy code

package com.example.toastdemoapp

import android.os.Bundle

import android.widget.\*

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Get references to UI components

val nameEditText = findViewById<EditText>(R.id.nameEditText)

val showNameButton = findViewById<Button>(R.id.showNameButton)

val agreeCheckBox = findViewById<CheckBox>(R.id.agreeCheckBox)

val genderRadioGroup = findViewById<RadioGroup>(R.id.genderRadioGroup)

val notificationToggleButton = findViewById<ToggleButton>(R.id.notificationToggleButton)

// Show Name Button Click

showNameButton.setOnClickListener {

val name = nameEditText.text.toString()

Toast.makeText(this, "Your name is: $name", Toast.LENGTH\_SHORT).show()

}

// Checkbox Checked Change

agreeCheckBox.setOnCheckedChangeListener { \_, isChecked ->

if (isChecked) {

Toast.makeText(this, "You agreed to the terms and conditions", Toast.LENGTH\_SHORT).show()

} else {

Toast.makeText(this, "You did not agree to the terms", Toast.LENGTH\_SHORT).show()

}

}

// RadioGroup Checked Change

genderRadioGroup.setOnCheckedChangeListener { \_, checkedId ->

val gender = when (checkedId) {

R.id.maleRadioButton -> "Male"

R.id.femaleRadioButton -> "Female"

else -> "Unknown"

}

Toast.makeText(this, "Selected Gender: $gender", Toast.LENGTH\_SHORT).show()

}

// ToggleButton State Change

notificationToggleButton.setOnCheckedChangeListener { \_, isChecked ->

val status = if (isChecked) "Notifications are ON" else "Notifications are OFF"

Toast.makeText(this, status, Toast.LENGTH\_SHORT).show()

}

}

}

#### MainActivity.java (Java)

java

Copy code

package com.example.toastdemoapp;

import android.os.Bundle;

import android.widget.Button;

import android.widget.CheckBox;

import android.widget.EditText;

import android.widget.RadioGroup;

import android.widget.Toast;

import android.widget.ToggleButton;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Get references to UI components

EditText nameEditText = findViewById(R.id.nameEditText);

Button showNameButton = findViewById(R.id.showNameButton);

CheckBox agreeCheckBox = findViewById(R.id.agreeCheckBox);

RadioGroup genderRadioGroup = findViewById(R.id.genderRadioGroup);

ToggleButton notificationToggleButton = findViewById(R.id.notificationToggleButton);

// Show Name Button Click

showNameButton.setOnClickListener(view -> {

String name = nameEditText.getText().toString();

Toast.makeText(MainActivity.this, "Your name is: " + name, Toast.LENGTH\_SHORT).show();

});

// Checkbox Checked Change

agreeCheckBox.setOnCheckedChangeListener((buttonView, isChecked) -> {

if (isChecked) {

Toast.makeText(MainActivity.this, "You agreed to the terms and conditions", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(MainActivity.this, "You did not agree to the terms", Toast.LENGTH\_SHORT).show();

}

});

// RadioGroup Checked Change

genderRadioGroup.setOnCheckedChangeListener((group, checkedId) -> {

String gender;

if (checkedId == R.id.maleRadioButton) {

gender = "Male";

} else if (checkedId == R.id.femaleRadioButton) {

gender = "Female";

} else {

gender = "Unknown";

}

Toast.makeText(MainActivity.this, "Selected Gender: " + gender, Toast.LENGTH\_SHORT).show();

});

// ToggleButton State Change

notificationToggleButton.setOnCheckedChangeListener((buttonView, isChecked) -> {

String status = isChecked ? "Notifications are ON" : "Notifications are OFF";

Toast.makeText(MainActivity.this, status, Toast.LENGTH\_SHORT).show();

});

}

}

### Explanation

1. **EditText and Button**: Displays the entered name in a Toast when the **Show Name** button is clicked.
2. **Checkbox**: Shows a Toast message based on whether the user agrees to the terms.
3. **RadioGroup with RadioButtons**: Displays a Toast with the selected gender when a RadioButton is selected.
4. **ToggleButton**: Shows a Toast indicating whether notifications are on or off when toggled.

### Step 4: Run the App

1. **Connect your device** or start an **Android emulator**.
2. **Click "Run"** in Android Studio.
3. Interact with each UI component to see the corresponding Toast messages.

**Q 24 Develop a app for demonstration of Explicit intent which shows the communication between to the geeksforgeeks website and your app**

### Step 1: Create a New Android Project

1. **Open Android Studio**.
2. **Create a New Project** with an **Empty Activity**.
3. Choose **Kotlin** or **Java** as the language and name the project (e.g., GeeksforGeeksIntentApp).

### Step 2: Define the Layout for MainActivity (activity\_main.xml)

In activity\_main.xml, create a simple UI with a **Button**. When clicked, this button will open the GeeksforGeeks website in a browser.

#### activity\_main.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="20dp"

android:gravity="center">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Visit GeeksforGeeks Website"

android:textSize="18sp"

android:layout\_marginBottom="20dp" />

<Button

android:id="@+id/openWebsiteButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Open GeeksforGeeks Website" />

</LinearLayout>

### Step 3: Create the MainActivity Class

In MainActivity.kt or MainActivity.java, set up an **Explicit Intent** to open the GeeksforGeeks website in the browser when the button is clicked.

#### MainActivity.kt (Kotlin)

kotlin

Copy code

package com.example.geeksforgeeksintentapp

import android.content.Intent

import android.net.Uri

import android.os.Bundle

import android.widget.Button

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Get reference to the button

val openWebsiteButton = findViewById<Button>(R.id.openWebsiteButton)

// Set click listener for the button

openWebsiteButton.setOnClickListener {

// Create an explicit intent to open a URL in the browser

val gfgWebsiteUri = Uri.parse("https://www.geeksforgeeks.org")

val intent = Intent(Intent.ACTION\_VIEW, gfgWebsiteUri)

// Start the intent to open the URL in the browser

startActivity(intent)

}

}

}

#### MainActivity.java (Java)

java

Copy code

package com.example.geeksforgeeksintentapp;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Get reference to the button

Button openWebsiteButton = findViewById(R.id.openWebsiteButton);

// Set click listener for the button

openWebsiteButton.setOnClickListener(view -> {

// Create an explicit intent to open a URL in the browser

Uri gfgWebsiteUri = Uri.parse("https://www.geeksforgeeks.org");

Intent intent = new Intent(Intent.ACTION\_VIEW, gfgWebsiteUri);

// Start the intent to open the URL in the browser

startActivity(intent);

});

}

}

### Explanation

1. **MainActivity**:
   * **Button**: The layout contains a button labeled "Open GeeksforGeeks Website".
   * **Intent**: In MainActivity, when the button is clicked, an **Intent** is created with the action Intent.ACTION\_VIEW and the URI for the GeeksforGeeks website (https://www.geeksforgeeks.org).
   * **startActivity**: This intent is then used with startActivity(intent), which opens the GeeksforGeeks website in the device’s default browser.

### Step 4: Run the App

1. **Connect your device** or start an **Android emulator**.
2. **Click "Run"** in Android Studio.
3. Once the app launches:
   * Click the **Open GeeksforGeeks Website** button, and it should open the GeeksforGeeks website in the device’s default browser.

**Q 25 Develop a app for demonstration of Explicit intent which shows the communication between to the tutorialspoint website and your app**

### Step 1: Create a New Android Project

1. **Open Android Studio**.
2. **Create a New Project** with an **Empty Activity**.
3. Name the project (e.g., TutorialsPointIntentApp) and select **Java** or **Kotlin** as the language.

### Step 2: Define the Layout for MainActivity (activity\_main.xml)

In activity\_main.xml, create a simple UI with a **Button**. When clicked, this button will open the TutorialsPoint website in a browser.

#### activity\_main.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="20dp"

android:gravity="center">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Visit TutorialsPoint Website"

android:textSize="18sp"

android:layout\_marginBottom="20dp" />

<Button

android:id="@+id/openWebsiteButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Open TutorialsPoint Website" />

</LinearLayout>

### Step 3: Create the MainActivity Class

In MainActivity.kt or MainActivity.java, set up an **Explicit Intent** to open the TutorialsPoint website in the browser when the button is clicked.

#### MainActivity.kt (Kotlin)

kotlin

Copy code

package com.example.tutorialspointintentapp

import android.content.Intent

import android.net.Uri

import android.os.Bundle

import android.widget.Button

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Get reference to the button

val openWebsiteButton = findViewById<Button>(R.id.openWebsiteButton)

// Set click listener for the button

openWebsiteButton.setOnClickListener {

// Create an explicit intent to open a URL in the browser

val tutorialspointWebsiteUri = Uri.parse("https://www.tutorialspoint.com")

val intent = Intent(Intent.ACTION\_VIEW, tutorialspointWebsiteUri)

// Start the intent to open the URL in the browser

startActivity(intent)

}

}

}

#### MainActivity.java (Java)

java

Copy code

package com.example.tutorialspointintentapp;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Get reference to the button

Button openWebsiteButton = findViewById(R.id.openWebsiteButton);

// Set click listener for the button

openWebsiteButton.setOnClickListener(view -> {

// Create an explicit intent to open a URL in the browser

Uri tutorialspointWebsiteUri = Uri.parse("https://www.tutorialspoint.com");

Intent intent = new Intent(Intent.ACTION\_VIEW, tutorialspointWebsiteUri);

// Start the intent to open the URL in the browser

startActivity(intent);

});

}

}

### Explanation

1. **Button Setup**: The layout contains a button labeled "Open TutorialsPoint Website".
2. **Intent Creation**: In MainActivity, when the button is clicked, an **Intent** is created with the action Intent.ACTION\_VIEW and the URI for the TutorialsPoint website (https://www.tutorialspoint.com).
3. **Launching the Intent**: The startActivity(intent) call launches the intent, which opens the TutorialsPoint website in the device’s default browser.

### Step 4: Run the App

1. **Connect your device** or start an **Android emulator**.
2. **Click "Run"** in Android Studio.
3. Once the app launches:
   * Click the **Open TutorialsPoint Website** button, and it should open the TutorialsPoint website in the device’s default browser.