University System — Phone Validation, Account Creation Flow, and Student Code Generation

This document explains the **three main parts** of your university system backend:

1. **Egyptian Phone Number Validation**
2. **Account Creation Flow (Admin → Student/Doctor)**
3. **Student Code Generation (with Year + Department + Level)**

All explanations include logic + C# code ready for use.

1️⃣ Egyptian Phone Number Validation

📣 Requirements

An Egyptian phone number must:

- Be **11 digits** exactly.
- Start with one of the official prefixes:
  - `010` (Vodafone)
  - `011` (Etisalat)
  - `012` (Orange)
  - `015` (WE)
- After the prefix → must be **8 digits**.

📣 Recommended Regex

```
public bool IsValidEgyptianPhone(string phone)
{
    return Regex.IsMatch(phone, "^(010|011|012|015)[0-9]{8}$");
}
```

📣 Usage

```
if(!IsValidEgyptianPhone(student.Phone))
    return BadRequest("Invalid Egyptian phone number.");
```

---

2 Account Creation Flow (Admin → Student/Doctor)

📣 What the Admin Does

- Creates a user (student or doctor)
- Enters: name, department, level (for students), phone
- System generates:
  - **Username** (email or code)
  - **Temporary password**
  - **MustChangePassword = true**
  - **Fake inbox notification** with login info (instead of real email)

📣 Temporary Password Generation

```
public string GenerateTemporaryPassword()
{
    return Guid.NewGuid().ToString("N").Substring(0, 10);
}
```

📣 Account Creation Code (Admin)

```
public async Task<IActionResult> CreateStudent(CreateStudentDto dto)
{
    // Validate phone
    if(!IsValidEgyptianPhone(dto.Phone))
        return BadRequest("Invalid phone number.");

    // Generate student code
    string studentCode = await GenerateStudentCodeAsync(dto.FacultyCode,
dto.Level);

    // Generate password
    string tempPassword = GenerateTemporaryPassword();

    var student = new Student
    {
        Name = dto.Name,
        Phone = dto.Phone,
```

```
        FacultyCode = dto.FacultyCode,
        Level = dto.Level,
        Email = $"{studentCode}@demo-university.edu.eg",
        Username = studentCode,
        StudentCode = studentCode,
        PasswordHash = HashPassword(tempPassword),
        MustChangePassword = true,
        CreatedAt = DateTime.Now
    };

    _context.Students.Add(student);
    await _context.SaveChangesAsync();

    // Create fake notification
    var note = new Notification
    {
        UserId = student.Id,
        Title = "Your account has been created",
        Message = $"Welcome {student.Name}! Your username is
{student.Username} and your temporary password is {tempPassword}. Please
change it upon first login.",
        CreatedAt = DateTime.Now
    };

    _context.Notifications.Add(note);
    await _context.SaveChangesAsync();

    return Ok(new { message = "Student created successfully" });
}
```

📣 First Login Flow (Force Password Change)

```
public async Task<IActionResult> Login(LoginDto dto)
{
    var user = await _context.Users.SingleOrDefaultAsync(u => u.Username ==
dto.Username);
    if(user == null) return Unauthorized();

    if(!VerifyPassword(dto.Password, user.PasswordHash))
        return Unauthorized();

    if(user.MustChangePassword)
        return Ok(new { status = "force_change_password", userId = user.Id });
```

```
    return Ok(new { status = "success", token = GenerateJwt(user) });
}
```

📣 Change Password Endpoint

```
[HttpPost("change-password")]
public async Task<IActionResult> ChangePassword(ChangePasswordDto dto)
{
    var user = await _context.Users.FindAsync(dto.UserId);
    if(user == null) return NotFound();

    user.PasswordHash = HashPassword(dto.NewPassword);
    user.MustChangePassword = false;
    await _context.SaveChangesAsync();

    return Ok(new { message = "Password changed successfully" });
}
```

3️⃣ Student Code Generation (Year + Department + Level)

📣 Format

```
{Year}{FacultyCode}{Level}{Serial}
```

Example:
2501110001

- 25 → Year (2025)
- 01 → Faculty (CS)
- 1 → Level (Year 1)
- 0001 → Serial

📣 Code: Generate Student Code

```
public async Task<string> GenerateStudentCodeAsync(int facultyCode, int level)
{
    string year = DateTime.Now.Year.ToString().Substring(2); // "25"
    string faculty = facultyCode.ToString().PadLeft(2, '0');  // "01"
    string levelStr = level.ToString();                       // "1"

    string prefix = $"{year}{faculty}{levelStr}";             // "25011"

    var lastStudent = await _context.Students
        .Where(s => s.StudentCode.StartsWith(prefix))
        .OrderByDescending(s => s.StudentCode)
        .FirstOrDefaultAsync();

    int serial = 1;

    if(lastStudent != null)
    {
        string lastSerial = lastStudent.StudentCode.Substring(prefix.Length);
        serial = int.Parse(lastSerial) + 1;
    }

    string serialStr = serial.ToString().PadLeft(4, '0');

    return $"{prefix}{serialStr}";
}
```