# SE2062 – Programming Paradigms

## Take-Home Assignment: Building a DSL and HTML Form Generator with Lex and Yacc

**Duration:** ~18 hours
**Submission Deadline:** 21-04-2025
**Total Marks:** 100

### 🎯 Objective

Create a Domain-Specific Language (DSL) named **FormLang++** for form specification. Use Lex and Yacc to parse the DSL and generate equivalent HTML forms with validation.

### 📚 Learning Outcomes

- Design grammars for structured DSLs
- Use Lex and Yacc for syntactic analysis
- Translate structured definitions into executable code (HTML)
- Understand the role of parsing in programming paradigms

### 📘 DSL Field Specifications

| Field Type | Description | Supported Attributes |
|---|---|---|
| text | Single-line input | required, pattern, default |
| textarea | Multi-line input | rows, cols, default |
| number | Numeric input | min, max, required |
| email | Email input | pattern, required |
| date | Date picker | min, max, required |
| checkbox | Boolean switch | default |
| dropdown | Option list | required, default, options |
| radio | Single option from group | required, options |
| password | Masked input | required, pattern |
| file | Upload file | accept, required |

### 📝 Example FormLang++ Code

```
form Registration {
  meta author = "SE2062 Team";

  section PersonalDetails {
    field fullName: text required;
    field email: email required;
    field age: number min=18 max=99;
  }

  section Preferences {
    field gender: radio ["Male", "Female", "Other"];
    field newsletter: checkbox default=true;
  }

  validate {
    if age < 18 {
      error "You must be at least 18.";
    }
  }
}
```

### 🌐 Corresponding HTML Output

```
<form name="Registration">
  <label>Full Name: <input type="text" name="fullName" required></label><br>
```

```
  <label>Email: <input type="email" name="email" required></label><br>
  <label>Age: <input type="number" name="age" min="18" max="99" required></label><br>

  <label>Gender:</label><br>
  <input type="radio" name="gender" value="Male"> Male<br>
  <input type="radio" name="gender" value="Female"> Female<br>
  <input type="radio" name="gender" value="Other"> Other<br>

  <label><input type="checkbox" name="newsletter" checked> Subscribe to newsletter</label><br>
</form>
```

## 📦 Submission Requirements

- **1. Grammar Specification (EBNF)** –
  `grammar.pdf`
  - Clearly defined grammar rules for:
  - ✔ Form structure (`form`, `section`, `field`)
  - ✔ Field types and their attribute formats
  - ✔ Conditional rules (e.g., `if age < 25`)
  - ✔ Validation blocks (`validate { if ... }`)
  - ✔ Metadata declarations (`meta key = value;`)
  - ✔ Proper use of terminals and non-terminals
- **2. Lex and Yacc Implementation**
  - `lexer.l` – Token definitions for keywords, field types, numbers, strings, brackets, operators, etc.
  - `parser.y` – Parsing rules based on the EBNF grammar with meaningful semantic actions
  - **Required Output:** Either:
    - ✔ "Valid form" message, or
    - ✔ Structured output (e.g., JSON or intermediate form model), or
    - ✔ Direct HTML generation using `fprintf()`
  - **Error Handling:** Your parser should produce meaningful errors for malformed `.form` input
- **3. Sample Inputs and Outputs**
  - `example.form` – A DSL script that covers most field types and conditional rules
  - `output.html` – The HTML file generated from `example.form`
- **4. Demo Video (3 minutes max)**
  - `demo.mp4` or YouTube link
  - ✔ Walkthrough of code structure
  - ✔ Running the lexer/parser on `example.form`
  - ✔ The resulting HTML output or validation results
  - ✔ At least one example of invalid input and the error shown
  - ✔ Clear voice-over or subtitles for explanation

## 📊 Detailed Marking Rubric

| Criteria | Marks | Poor (0–35%) | Below Avg (36–45%) | Average (46–65%) | Good (66–75%) | Very Good (76–100%) |
|---|---|---|---|---|---|---|
| Grammar Design | 25 | Fragmented, major elements missing | Some field types missing or unsupported | Basic structures captured | All required rules with explanation | Full coverage + reusable + extensible |
| Lex & Parser | 30 | Does not compile or parse anything | Parses tokens, fails on structure | Parses basic forms | Handles conditionals, metadata | Full working parser with errors & recovery |
| HTML Generator | 30 | No output or incorrect format | Outputs tags with syntax issues | Basic working HTML for fields | Structured & styled output | Semantic HTML, accurate attributes |
| Demo Video | 15 | No or unclear video | Incomplete or not showing results | Shows input/output process | Covers key components clearly | Professional, shows error cases too |

### ⚠️ AI Usage & Plagiarism Policy

**You are allowed to use AI tools (such as ChatGPT, Copilot, etc.) as *learning aids only*, not as code generators.**

- ✔ You may use AI to understand concepts, clarify syntax, and brainstorm ideas.
- ❌ You may **not** use AI to directly generate your Lex/Yacc code, grammar rules, or HTML output.
- ❌ Submitting AI-generated code as your own work is considered plagiarism.
- ✔ All code and design submitted must be written and debugged by **you**.

**Plagiarism and similarity checks** will be strictly enforced:

- ✔ All submissions will be checked using code similarity detection tools.
- ✔ If plagiarism or excessive AI-generated content is detected, marks will be deducted or submissions will be rejected entirely.
- ✔ Cases of academic dishonesty will be reported and may lead to disciplinary action.

**When in doubt, cite sources and ask questions!**