**Algorithm-1 Classify Legal Document Type**

**Input:** Legal document

**Output:** Classified document type, Success message or Error message

```
1.  Ensure the user is logged in.
2.  Verify the document is uploaded successfully.
3.  Extract text from the uploaded document:
    3.1. Process different types of documents.
    3.2. If the format is unsupported
        3.2.1.  Display an error message: “Unsupported file format.”
4.  Pass the extracted text to the Document Tagger Module.
    4.1. Use the LLM to classify the document into a predefined type.
    4.2. If classification fails:
        4.2.1.  Display an error message: “Unable to classify document.”
5.  Store the classified type in the database.
6.  Display the classified type to the user.
```

**Algorithm-2 User Login**

**Input:** Email, Password

**Output:** Success message or Error message

```
1.  Initialize variables for email, password, and loginStatus.
2.  Accept input for email.
3.  Send a request to the database to validate the credentials:
    3.1.  Query the database to check if the entered email exists.
    3.2.  If email does not exist:
        3.2.1.  Display an error message: “Invalid Email”.
        3.2.2.  Halt the process.
4.  If email exists:
    4.1. Accept input for password.
    4.2. Query the database to validate the entered password for the given email.
5.  If the password is valid:
    5.1. Set loginStatus = Success.
    5.2. Display a success message: “Login Successful”.
    5.3. Proceed to the next process.
6.  If the password is invalid:
    6.1. Display an error message: “Invalid Password”.
    6.2. Halt the process.
```

**Algorithm-3 RAG-Based Query Processing**

**Input:** User query, Knowledge base

**Output:** Relevant response or Error message

```
1. Validate user query input.
2. Process the query through LangChain:
   2.1. Tokenize and vectorize the query
   2.2. If query processing fails:
        2.2.1. Display error message: "Invalid query format."
3. Search through knowledge base:
   3.1. Retrieve relevant documents using vector similarity.
   3.2. If no relevant documents found:
        3.2.1. Display message: "No relevant information found."
4. Generate response using LLM:
   4.1. Combine retrieved documents with query context.
   4.2. If response generation fails:
        4.2.1. Display error message: "Unable to generate response."
5. Format and return the response to user.
```

**Algorithm-3 Contract Generation**

**Input:** User requirements, Template selection

**Output:** Generated contract document or Error message

```
1. Validate template selection:
   1.1. Check if template exists.
   1.2. If template not found:
        1.2.1. Display error message: "Template not found."
2. Process user inputs:
   2.1. Validate required fields.
   2.2. If validation fails:
        2.2.1. Display error message: "Missing required information."
3. Generate contract:
   3.1. Populate template with user data.
   3.2. Apply formatting rules.
   3.3. If generation fails:
        3.3.1. Display error message: "Contract generation failed."
4. Save generated contract:
   4.1. Store in database.
   4.2. Create audit log.
5. Return generated contract to user.
```

**Algorithm-5 Law Search and Filter**

**Input:** Search query, Filter parameters

**Output:** Filtered law results or Error message

```
1. Validate search input:
   1.1. Check query format.
   1.2. If invalid:
        1.2.1. Display error message: "Invalid search query."
2. Process search parameters:
   2.1. Apply jurisdiction filter.
   2.2. Apply date range filter.
   2.3. Apply relevance range criteria.
3. Execute search:
   3.1. Query legal database.
   3.2. If search fails:
        3.2.1. Display error message: "Search operation failed."
4. Process results:
   4.1. Sort by relevance.
   4.2. Apply user filters.
   4.3. If no results:
        4.3.1. Display message: "No matching results found."
5. Return filtered results to user.
```

**Algorithm-6 Create a Case and Upload Multiple Documents**

**Input:** Case details, Documents to be uploaded

**Output:** Success message or Error message

```
1. Validate case details:
   1.1. Check case name and associated user ID.
   1.2. If invalid:
        1.2.1. Display error message: "Invalid case details."
2. Allow the user to upload multiple documents:
   2.1. Verify document file formats
   2.2. If unsupported format:
        2.2.1. Display error message: "Unsupported file format."
3. Store case details and uploaded documents in the database:
   3.1. Assign a unique ID to the case.
   3.2. Save document metadata and files in storage.
4. If case creation succeeds:
   4.1. Display success message: "Case created successfully."
```

```
5.  If any error occurs during storage:
    5.1. Display error message: "Error creating case."
```

**Algorithm-7 Modify Existing Cases**

**Input:** Case ID, Modified case details

**Output:** Success or Error message

```
1.  Retrieve existing case details:
    1.1. Query the database using the Case ID.
    1.2. If the case does not exist:
         1.2.1.  Display error message: "Case not found."
2.  Allow the user to edit case details:
    2.1. Validate the modified details.
    2.2. If invalid:
         2.2.1.  Display error message: "Invalid case details."
3.  Update the case in the database:
    3.1. Replace old details with modified details.
    3.2. Update any associated documents if modified.
4.  If the update succeeds:
    4.1. Display success message: "Case updated successfully."
5.  If any error occurs during update:
    5.1. Display error message: "Error modifying case."
```

**Algorithm-8 Document Summarization**

**Input:** Legal document

**Output:** Structured summary or Error message

```
1. Validate document input:
   1.1. Check document format.
   1.2. If invalid:
        1.2.1. Display error message: "Invalid document format."
2. Process document:
   2.1. Extract text content.
   2.2. Identify document type.
   2.3. If processing fails:
        2.3.1. Display error message: "Document processing failed."
3. Generate summary using LLM:
   3.1. Apply document-type specific rules.
   3.2. Extract key points.
   3.3. If generation fails:
        3.3.1. Display error message: "Summary generation failed."
4. Format summary:
   4.1. Structure based on document type.
   4.2. Highlight key terms.
5. Return formatted summary to user.
```