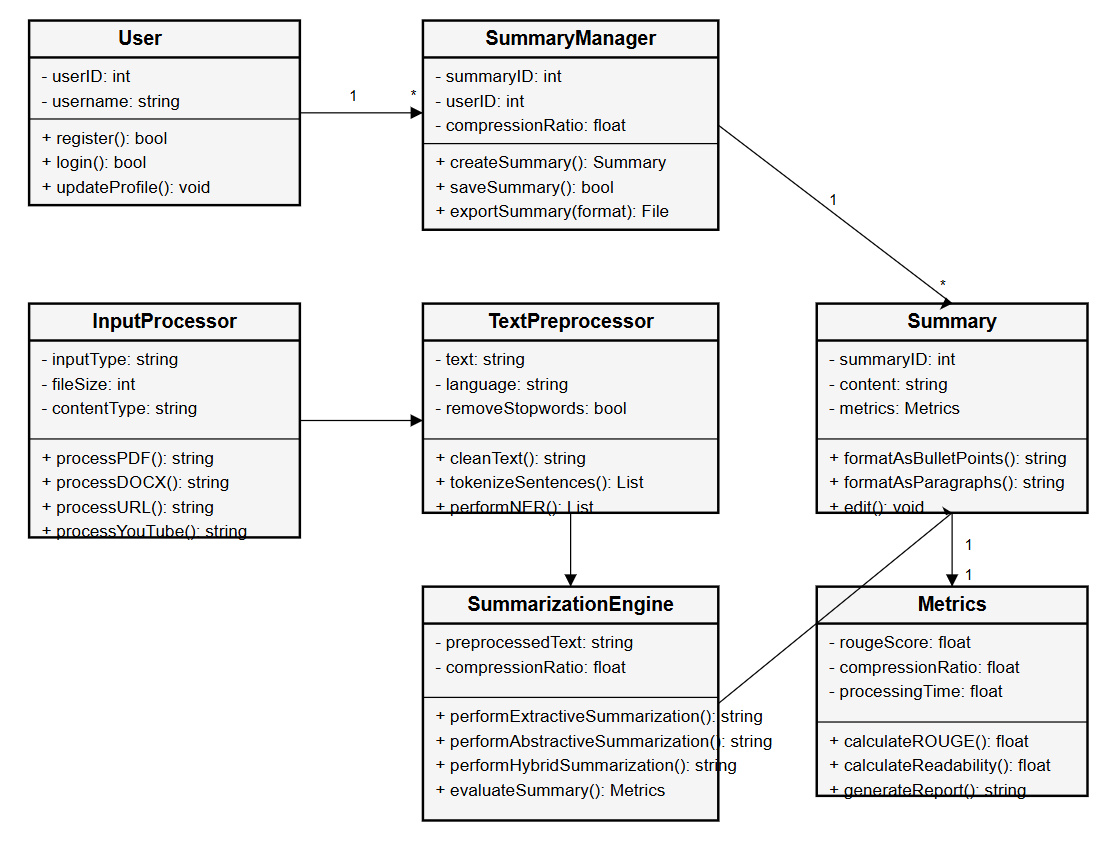
**1. UML Diagrams for SummarizeIQ**

Let's start with creating the key UML diagrams for your AI-powered multi-source text summarization platform.



**Explanation:**

This is a **UML class diagram** representing the architecture of an **AI-based Text Summarization System**. Let me walk you through the different components and their relationships:

**1. User**

Represents the person using the system.

* **Attributes:**
  + userID: Unique identifier.
  + username: Name of the user.
* **Methods:**
  + register(): For new user registration.
  + login(): For user login.
  + updateProfile(): Updates user details.

**2. SummaryManager**

Handles the logic for managing summaries.

* **Attributes:**
  + summaryID: ID for each summary.
  + userID: Links the summary to a user.
  + compressionRatio: Specifies how much to compress the text.
* **Methods:**
  + createSummary(): Triggers the summarization process.
  + saveSummary(): Saves the summary to storage.
  + exportSummary(format): Exports the summary (PDF, DOCX, etc.).

**Relationships:**

* A User can have many SummaryManager instances.
* A SummaryManager can generate multiple Summary entries.

**3. InputProcessor**

Processes various input types before summarization.

* **Attributes:**
  + inputType, fileSize, contentType: Metadata for the input.
* **Methods:**
  + processPDF(), processDOCX(), processURL(), processYouTube(): Extracts raw text from various sources.

**4. TextPreprocessor**

Cleans and prepares text for summarization.

* **Attributes:**
  + text: The raw input text.
  + language: Language of the text.
  + removeStopwords: Option to remove common words.
* **Methods:**
  + cleanText(): Removes noise.
  + tokenizeSentences(): Splits into sentences.
  + performNER(): Named Entity Recognition.

**5. SummarizationEngine**

Core engine to perform summarization.

* **Attributes:**
  + preprocessedText: Cleaned text input.
  + compressionRatio: Desired summary length.
* **Methods:**
  + performExtractiveSummarization(): Picks key sentences.
  + performAbstractiveSummarization(): Rewrites content.
  + performHybridSummarization(): Mix of both.
  + evaluateSummary(): Returns metrics (ROUGE, etc.).

**6. Summary**

Represents the generated summary.

* **Attributes:**
  + summaryID: Unique ID.
  + content: Summarized text.
  + metrics: Evaluation results.
* **Methods:**
  + formatAsBulletPoints(): Converts summary into bullet form.
  + formatAsParagraphs(): Converts summary into paragraph format.
  + edit(): Allows manual editing.

**Relationships:**

* Each Summary is associated with one Metrics object.

**7. Metrics**

Contains evaluation details for the summary.

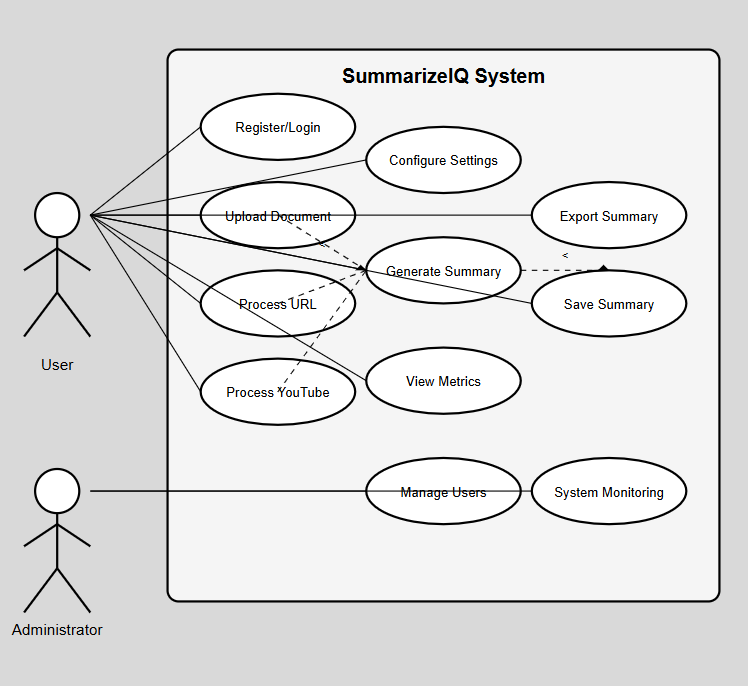
* **Attributes:**
  + rougeScore: Accuracy vs original text.
  + compressionRatio: Degree of summarization.
  + processingTime: Time taken to summarize.
* **Methods:**
  + calculateROUGE(): Returns ROUGE score.
  + calculateReadability(): Evaluates how easy the text is to read.
  + generateReport(): Generates a metrics report.

**Relationships Summary:**

* **User ↔ SummaryManager:** One user can have multiple summaries.
* **SummaryManager ↔ Summary:** One manager manages multiple summaries.
* **Summary ↔ Metrics:** Each summary has one set of evaluation metrics.
* **InputProcessor → TextPreprocessor → SummarizationEngine → Summary**

**2. Use Case Diagram**

The use case diagram will show the main interactions between users and the SummarizeIQ system.



**Use Case Diagram: SummarizeIQ System**

This diagram illustrates the interaction between two types of users (User and Administrator) and the functionalities provided by the SummarizeIQ system.

**Actors:**

1. **User**  
   Represents a general user of the system who can use summarization features.
2. **Administrator**  
   A privileged user responsible for overseeing and managing the system and its users.

**Use Cases for User:**

* **Register/Login**: Allows the user to create an account and access the system.
* **Upload Document**: Enables the user to upload documents (e.g., PDF, DOCX) for summarization.
* **Process URL**: Allows the user to input a web URL whose content needs to be summarized.
* **Process YouTube**: Allows the user to provide a YouTube link, from which transcripts are fetched and summarized.
* **Generate Summary**: Core functionality that creates summaries based on uploaded content.
* **Configure Settings**: Lets the user set options like compression ratio or summary format.
* **Save Summary**: Saves the generated summary to the user’s profile.
* **Export Summary**: Allows the user to download the summary in a chosen format (PDF, DOCX, etc.).
* **View Metrics**: Displays evaluation metrics such as ROUGE score, compression ratio, or readability.

Dependencies:

* Upload Document, Process URL, and Process YouTube all lead into the Generate Summary use case.
* Generate Summary has optional extensions to Save Summary and Export Summary.

Use Cases for Administrator:

Manage Users: Add, delete, or update user information.

System Monitoring: Monitor system usage, performance, and logs.

**3. Design Phase**

The design phase for the SummarizeIQ system consists of architectural decisions, component design, and interface specifications. Here's a detailed breakdown of the design phase:

**3.1 Architectural Design**

SummarizeIQ follows a three-tier architecture:

1. **Presentation Layer:**
   * Responsive web interface built with Flask, HTML5, CSS3, and JavaScript
   * User dashboard for managing summaries and account settings
   * Intuitive input forms for document processing
   * Summary viewing and export interface
2. **Application Layer:**
   * Authentication and user management module
   * Document processing module for multiple formats
   * Text preprocessing pipeline
   * Summarization algorithms (extractive, abstractive, hybrid)
   * Performance metrics calculation module
   * Export functionality for different formats
3. **Data Layer:**
   * Database for user accounts and preferences
   * Storage for summary history and processed documents
   * Logging and performance tracking

**3.2 Component Design**

**3.2.1 User Authentication Module**

* Registration, login, and password reset functionality
* Session management and security features
* User profile settings and preferences

**3.2.2 Input Processing Module**

* Format-specific parsers for PDF, DOCX, TXT files
* Web scraping component for URL processing
* YouTube transcript extraction
* Input validation and error handling

**3.2.3 Text Preprocessing Module**

* Content cleaning and normalization
* Tokenization and segmentation
* Linguistic processing (POS tagging, NER)
* Domain-specific preprocessing options

**3.2.4 Summarization Engine**

* TextRank implementation for extractive summarization
* Transformer-based models for abstractive summarization
* Hybrid approach combining both methods
* Customization options for compression ratio and domain

**3.2.5 Output and Metrics Module**

* Summary formatting options (paragraphs, bullet points)
* ROUGE score calculation
* Readability metrics
* Export functionality (TXT, DOCX, PDF)

**3.2.6 Admin Module**

* User management interface
* System monitoring and logs
* Performance analytics dashboard

**3.3 Interface Design**

**3.3.1 Web Interface**

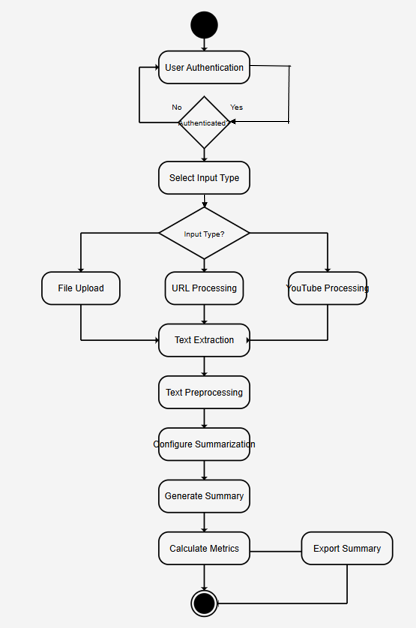
* Clean, intuitive UI with responsive design
* Dashboard with summary history
* Input selection and parameter controls
* Summary display with formatting options
* Metrics visualization
* Export controls

**3.3.2 API Interface**

* RESTful API endpoints for programmatic access
* Authentication and rate limiting
* Input processing endpoints
* Summarization control endpoints
* Results retrieval endpoints

**4. Activity Diagram**

The activity diagram illustrates the flow of activities in the SummarizeIQ system, particularly focusing on the summarization process.

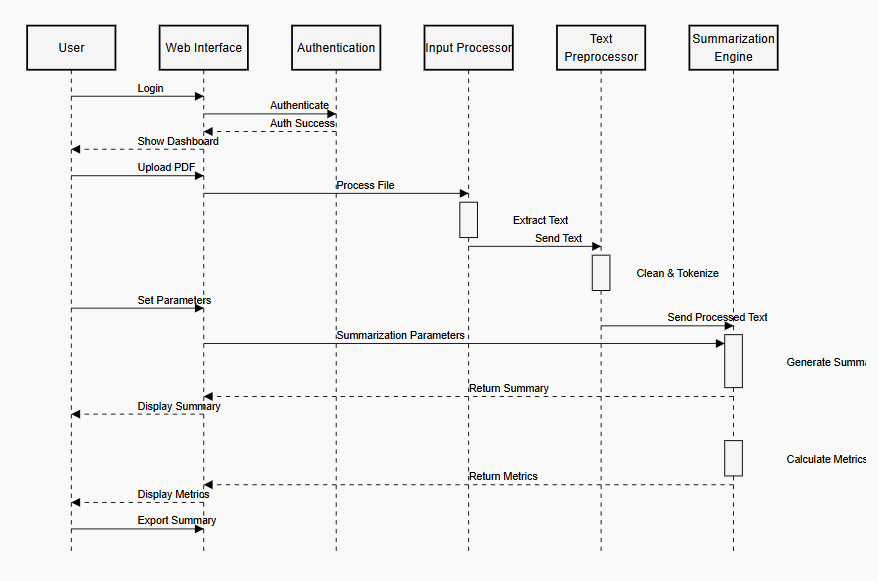


**Flow Steps:**

1. **User Authentication**  
   The process begins with the user attempting to authenticate.
   * If the user is not authenticated, the process loops back.
   * If authenticated, the process proceeds.
2. **Select Input Type**  
   The user selects the type of input they wish to summarize. Options include:
   * File Upload
   * URL Processing
   * YouTube Processing
3. **Input Handling**  
   Based on the selected input type, the system:
   * Uploads and reads file content
   * Fetches content from a URL
   * Extracts transcript from a YouTube video
4. **Text Extraction**  
   The raw text is extracted from the provided input.
5. **Text Preprocessing**  
   This step cleans the text (removal of stopwords, punctuation, tokenization, etc.) to prepare it for summarization.
6. **Configure Summarization**  
   The user sets parameters such as compression ratio and summary format.
7. **Generate Summary**  
   The core summary generation process is executed using the configured options.
8. **Calculate Metrics**  
   Evaluation metrics like ROUGE score, readability, and compression ratio are computed.
9. **Export Summary**  
   Optionally, the user can export the generated summary in a selected format.
10. **End**  
    The process concludes.

**5. Sequence Diagram**

The sequence diagram shows the interaction between different components of the SummarizeIQ system when processing a text summarization request.



**Explanation:**

**Actors and Components:**

* **User**: Initiates actions
* **Web Interface**: Frontend application
* **Authentication**: Verifies user identity
* **Input Processor**: Handles uploaded or input data
* **Text Preprocessor**: Prepares raw text for summarization
* **Summarization Engine**: Generates and evaluates summaries

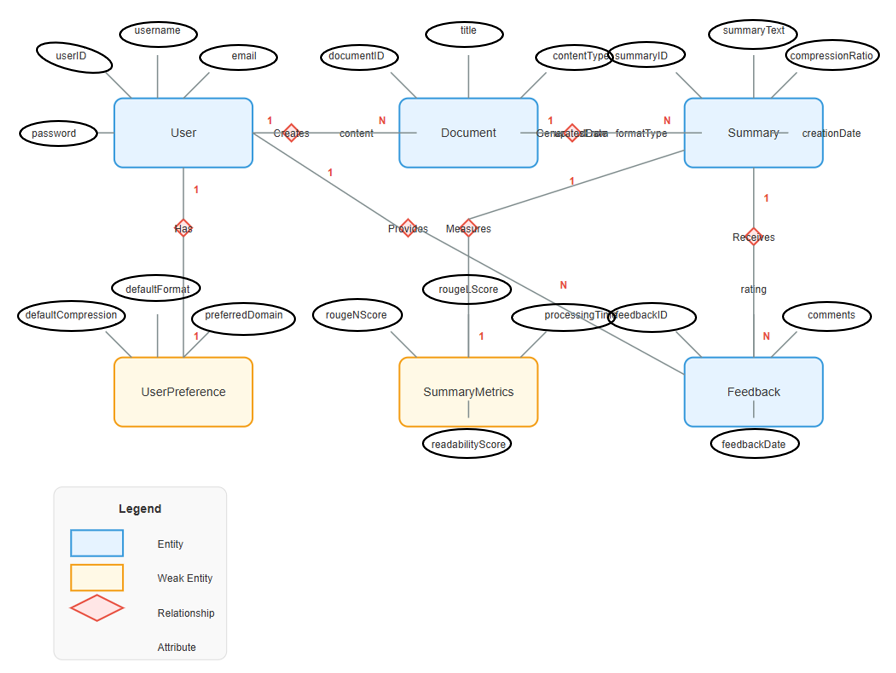
**Interaction Flow:**

1. **Login**  
   The user initiates login through the web interface.
2. **Authentication**
   * The web interface sends the login request to the authentication service.
   * If successful, authentication responds back.
   * The web interface then loads the dashboard for the user.
3. **Upload PDF**
   * The user uploads a document (e.g., a PDF).
   * The web interface sends it to the input processor.
4. **Process File**
   * The input processor extracts text from the document.
   * The extracted text is passed to the text preprocessor.
5. **Text Preprocessing**
   * The text is cleaned and tokenized.
   * The processed text is sent to the summarization engine.
6. **Set Parameters**
   * The user sets parameters like summary length or format.
   * These are forwarded by the web interface to the summarization engine.
7. **Generate Summary**
   * The summarization engine generates the summary based on the parameters.
   * The summary is returned to the web interface.
8. **Display Summary**
   * The summary is displayed to the user.
9. **Calculate Metrics**
   * The summarization engine calculates evaluation metrics.
   * The results are returned to the web interface.
10. **Display Metrics and Export Summary**

* Metrics are shown to the user.
* The user can export the summary if desired.

**6. ER Diagram**

The Entity-Relationship diagram shows the database structure for the SummarizeIQ system.



**Entities (Boxes):**

* Blue boxes represent strong entities: User, Document, Summary, and Feedback
* Yellow boxes represent weak entities: UserPreference and SummaryMetrics

**Relationships (Diamonds):**

* Creates: Users create Documents (1

relationship)

* Processes: Documents are processed into Summaries (1:1 relationship)
* Measures: Documents have SummaryMetrics (1:1 relationship)
* Receives: Summaries receive Feedback (1

relationship)

* Has: User has UserPreference (1:1 relationship)

**Attributes (Ovals):**

* User: userID, username, email, password
* Document: documentID, title, contentType, summaryID, creationDate, formatType
* Summary: summaryText, compressionRatio, creationDate
* Feedback: feedbackID, processingTime, comments, feedbackDate, rating
* UserPreference: defaultFormat, defaultCompression, preferredDomain
* SummaryMetrics: rougeScore, rougeNScore, readabilityScore

**Cardinality:**

* The numbers (1, N) on the relationship lines indicate how many instances of one entity can be associated with instances of the other entity.
* For example, a User (1) can create many Documents (N), but each Document is created by only one User.

This appears to be a data model for a system where users create documents, which get processed into summaries with various metrics. Users can configure their preferences, and feedback can be provided on the summaries. The model captures all the data elements and relationships needed for this document summarization system.