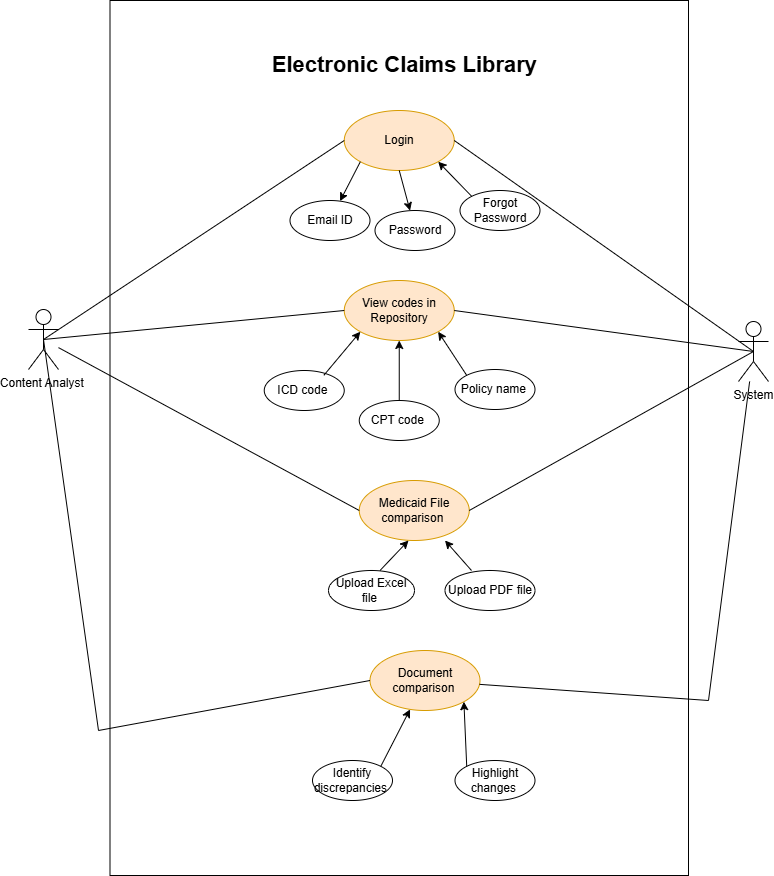
**Waterfall Project2 – Part -2/2**

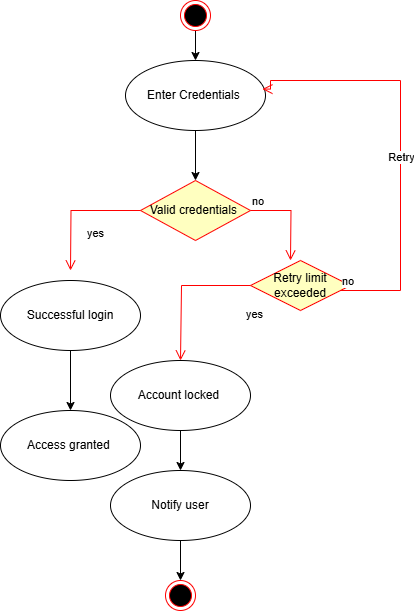
**Document 6- Please prepare a use case diagram, activity diagram and a use case specification document.**

**Use Case Diagram**

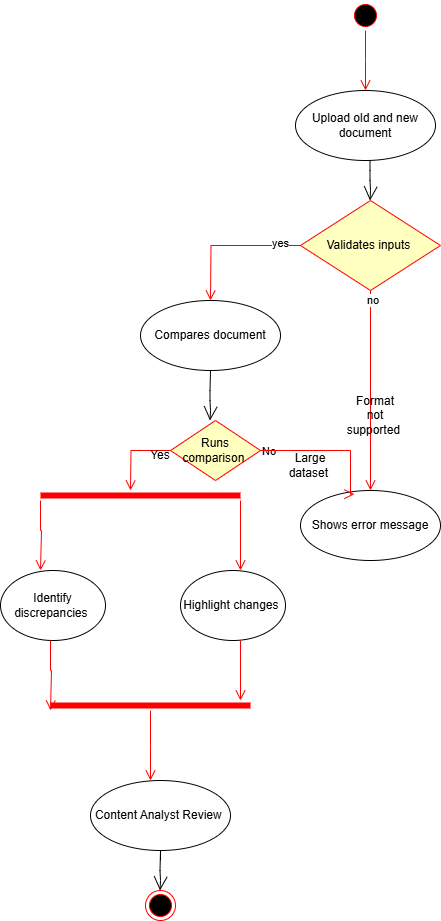
****

**Activity Diagram**

**Activity diagram for “Login” use case.**

****

**Activity diagram for “Medicaid file comparison” use case.**

****

**Use case Specifications**

| **Field** | **Details** |
| --- | --- |
| **Use Case ID** | UC001 |
| **Use Case Name** | Login to Electronic Claims Library |
| **Created By** | Kavya |
| **Last Updated** | 03-03-2024 |
| **Actor(s)** | Content Analyst, System |
| **Description** | This use case allows a registered user (Content Analyst) to log in to the Electronic Claims Library (ECL) to access their account and system features. |
| **Trigger** | The user wants to log in to access their account. |
| **Precondition** | The user must already have an account registered in the system. |
| **Postcondition** | The user is successfully logged into the system and redirected to the dashboard or home page. |
| **Normal Flow** | 1. System displays the 'Login' option.  2. User clicks on the 'Login' button.  3. System displays a login form requesting Email ID and Password.  4. User enters valid Email ID and Password.  5. User clicks on 'Submit' or 'Login'.  6. System validates the credentials.  7. On successful validation, the system logs the user into their account.  8. User is redirected to their account dashboard or home page. |
| **Alternative Flow** | **Step 6A**: If incorrect email or password is entered, the system shows an 'Invalid credentials' error message and allows the user to retry. |
| **Exception Flow** | **E1**: If a technical issue occurs during login, the system shows a 'System error. Please try again later' message. |
| **Priority** | High |
| **Frequency of Use** | Daily or frequent use |
| **Assumptions** | 1. The user has valid login credentials.  2. The user has internet connectivity. |
| **Constraints** | The system should enforce strong password policies and multi-factor authentication (if applicable). |
| **Dependencies** | 1. Requires an existing user account in the system.  2. Database connectivity must be available to validate credentials.  3. Integration with authentication service (e.g., LDAP, OAuth). |
| **Inputs and Outputs** | **Input**: User provides Email ID and Password.  **Output**: User is authenticated and redirected to the dashboard or an error message is displayed. |
| **Business Rules** | 1. Login attempts should be limited to prevent brute-force attacks.  2. System should enforce session timeout for security.  3. Password reset should be available in case of forgotten credentials. |
| **Miscellaneous Information** | 1. The system should log all login attempts for audit purposes.  2. Login failures should trigger an alert after multiple unsuccessful attempts.  3. The login page should be accessible only over a secure HTTPS connection. |

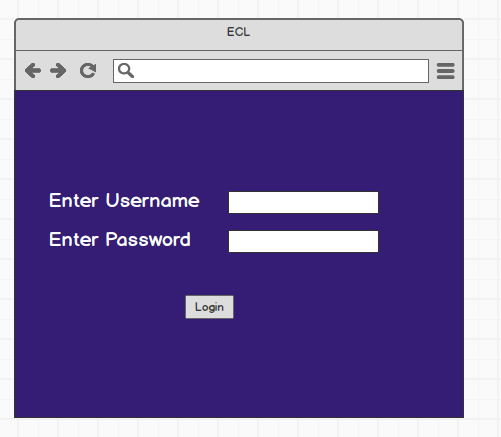
### -----------------------------------------------------------------------------------------

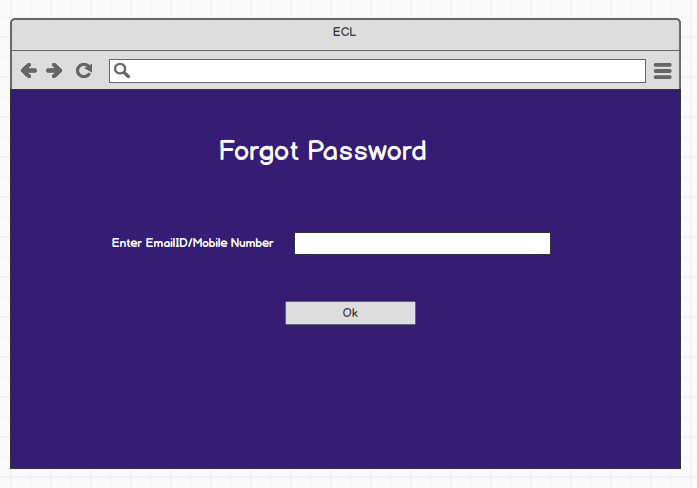
| **Use Case ID** | **UC002** |
| --- | --- |
| **Use Case Name** | View Codes in Repository |
| **Created By** | Kavya |
| **Last Updated** | 04-03-2024 |
| **Actor(s)** | Content Analyst, System |
| **Description** | This use case allows users to view and retrieve medical codes (ICD, CPT, Policy Name) from the repository. |
| **Trigger** | The user wants to access stored codes in the system. |
| **Precondition** | The user must be logged in and have appropriate access permissions. |
| **Postcondition** | The user successfully views the required codes from the repository. |
| **Normal Flow** | 1. System displays the "View Codes" option.  2. User selects "View Codes in Repository".  3. System retrieves and displays stored codes (ICD, CPT, Policy Name).  4. User searches for a specific code or browses the list.  5. System filters and displays relevant results.  6. User selects a code for detailed view.  7. System shows complete details of the selected code. |
| **Alternative Flow** | **Step 3A**: If no codes are found, the system displays "No codes available" message. |
| **Exception Flow** | **E1**: If there is a system error, the system shows a "Repository access failed. Try again later" message. |
| **Priority** | High |
| **Frequency of Use** | Daily or frequent use |
| **Assumptions** | 1. The repository contains medical codes.  2. The user has the necessary permissions to access the repository. |
| **Constraints** | The user must have role-based access control to view the repository. |
| **Dependencies** | Requires user authentication (Login use case must be completed successfully). |
| **Inputs and Outputs** | **Input**: User request to view stored codes. **Output**: List of ICD, CPT, and Policy Name codes displayed. |
| **Business Rules** | 1. Only authorized users can access the repository.  2. The repository should be updated regularly with the latest codes.  3. The system should support keyword-based search for quick code retrieval. |
| **Miscellaneous Information** | The system should log all user activities related to code access for audit purposes. |

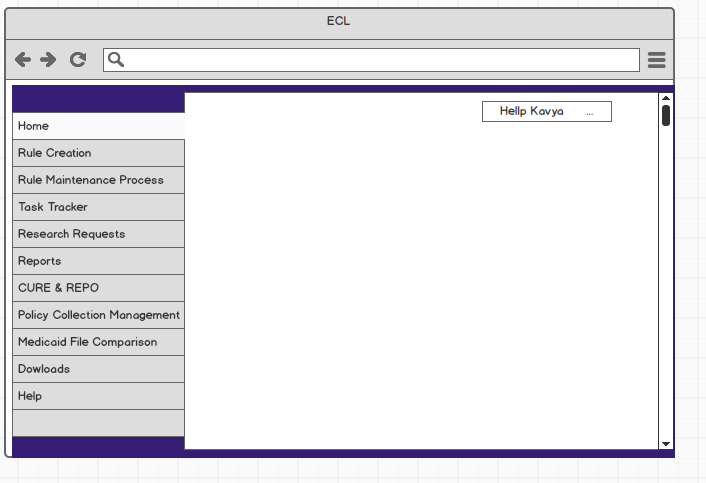
**----------------------------------------------------------------------------------------------------**

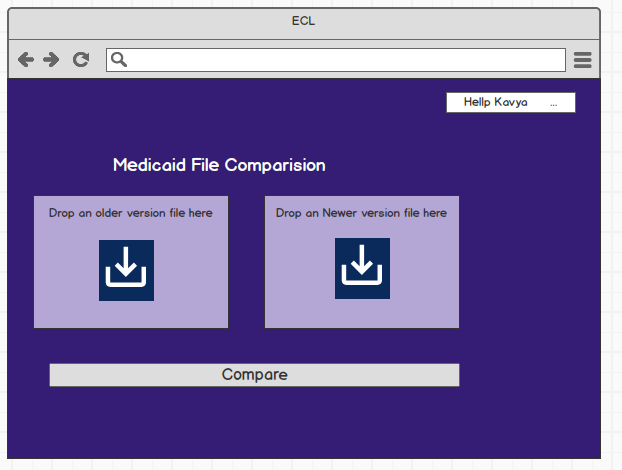
| **Field** | **Details** |
| --- | --- |
| **Use Case ID** | UC003 |
| **Use Case Name** | Medicaid File Comparison |
| **Created By** | Kavya |
| **Last Updated** | 20-03-2024 |
| **Actor(s)** | Content Analyst, System |
| **Description** | This use case allows the Content Analyst to upload Medicaid files (Excel or PDF) to compare and identify discrepancies in healthcare claim policies. |
| **Trigger** | The user wants to compare Medicaid files to ensure accuracy in claim policies. |
| **Precondition** | 1. The user must be logged into the system.2. The Medicaid files must be available for upload. |
| **Post condition** | The system successfully compares the uploaded Medicaid files and provides a report on discrepancies. |
| **Normal Flow** | 1. User navigates to the "Medicaid File Comparison" section.  2. System prompts the user to upload the required Medicaid files (Excel or PDF).  3. User selects and uploads the files.  4. System validates the file format and content.  5. System compares the uploaded files.  6. System identifies discrepancies, missing codes, or policy differences.  7. System generates a detailed report on the comparison results.  8. User reviews and downloads the report. |
| **Alternative Flow** | **Step 4A**: If the uploaded file format is incorrect, the system displays an error message and prompts the user to upload a valid file.**Step 6A**: If no discrepancies are found, the system notifies the user that the files match. |
| **Exception Flow** | **E1**: If the file upload fails due to system issues, the system shows a "File upload failed. Please try again later" message.**E2**: If the system cannot process the comparison, it displays a "Comparison error. Please contact support" message. |
| **Priority** | High |
| **Frequency of Use** | Weekly or as needed for Medicaid policy verification. |
| **Assumptions** | 1. The user has the necessary permissions to upload Medicaid files.  2. The system can handle large files efficiently. |
| **Constraints** | 1. File uploads should be limited to specific formats (Excel, PDF).  2. Maximum file size restrictions should be enforced.  3. Comparison processing time should be optimized for efficiency. |
| **Dependencies** | 1. Requires a functional file upload module.  2. Requires a document comparison algorithm integrated into the system.3. Database must store historical Medicaid data for accurate comparison. |
| **Inputs and Outputs** | **Input**: User uploads Medicaid files (Excel, PDF).  **Output**: System generates a report highlighting differences and discrepancies. |
| **Business Rules** | 1. System should support version control for Medicaid policy documents.  2. File comparison should be case-insensitive but should flag major formatting differences.  3. Users should be able to download the comparison results in PDF format. |
| **Miscellaneous Information** | 1. System should log all comparison activities for audit purposes.  2. If multiple users upload files simultaneously, the system should manage resources efficiently to prevent slowdowns.  3. A notification should be sent to the user once the comparison is complete. |

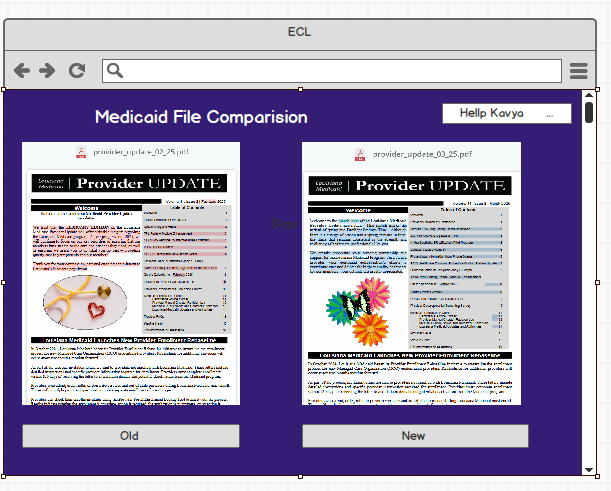
**Document 7- Screens and pages**











--------------------------------------------------------------------------------------------------------------------------

**Document 8- Tools-Visio and Azure**

**1. MS Visio**: It is a diagramming and vector graphics application used to create diagrams, flowcharts, and other visual representations of complex information.

It is used to create professional diagrams, flowcharts, organizational charts, network designs, floor plans, and much more. It's a versatile tool for visually representing processes, systems, and structures, making it popular among business analysts, project managers, and IT professionals.

Features: Pre-Built Templates and Shapes; Drag-and-Drop Interface; Integration with Other Microsoft Tools; Customization, Cloud Support

Uses: Swim lane diagrams; Network diagrams; System Diagrams; Charts; Gantt charts.

**2. Balsamiq**: It is a rapid wire framing tool used to create mock-ups and prototypes of interfaces. Balsamiq is a popular wire framing tool designed to create low-fidelity mock-ups for websites, mobile apps, or any other software application.

**3. Azure** : It is a more advanced prototyping tool used to create high-fidelity, interactive

wireframes, and prototypes for web and mobile applications. Azure provides a vast array of services, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), making it a versatile tool for various use cases.

Uses: App development and hosting; Data storage and backup; Disaster recovery and business continuity; Big data and analytics.

-----------------------------------------------------------------------------------------------------------------

**Document 9- My experience as BA**

### 1. Requirement Gathering:

* **Techniques Used:** MOSCOW and Prototyping
* **Challenges:** Client availability was limited, so I identified alternative points of contact to expedite information gathering.
* **Requirement Validation:** Used the FURPS technique to ensure completeness.
* **Refinement:** Removed duplicate and conflicting requirements.
* **Visualization:** Developed initial prototypes to clarify ambiguous requirements and enhance stakeholder engagement.

### 2. Requirement Analysis:

* **UML Diagrams:** Created Use Case Diagrams and Activity Diagrams for process flows.
* **Collaboration:** Presented diagrams to the team for feedback and made necessary modifications.
* **Documentation:** Prepared Business Requirements Specification (BRS) and Software Requirements Specification (SRS).

#### **Business Requirement Specification (BRS)**

* **Project Title:** Electronic Claims Library (ECL) Implementation
* **Objective:** To streamline claim documentation, storage, and retrieval while ensuring accuracy and compliance.
* **Business Goals & Benefits:**
  + Reduced manual documentation and paperwork.
  + Faster claim processing and reduced errors.
  + Enhanced fraud detection mechanisms.
  + Improved compliance with regulatory guidelines.
* **High-Level Requirements:**
  + Secure digital claim storage with version control.
  + Role-based access for different stakeholders.
  + Integration with existing claims processing platforms.
  + Compliance with healthcare data privacy laws.

#### **Software Requirement Specification (SRS)**

* **Functional Requirements:**
  + User authentication and role-based access.
  + Digital upload and tagging of claim documents.
  + Advanced search and filtering capabilities.
  + Claim approval workflow automation.
* **Non-Functional Requirements:**
  + High system availability and fault tolerance.
  + Data encryption and secure storage.
  + Scalability for future enhancements.
  + Compliance with HIPAA and other regulations.

### 3. Design:

* **Test Case Development:** Created test cases based on use cases.
* **Key Features Designed:**
  + User login and role management.
  + Search claimants by name and ID.
  + View, upload, and download claim files.
  + System authentication and security features.
  + Integration with external claims processing systems.
* **Requirement Traceability Matrix (RTM):** Maintained RTM to ensure all requirements were met during development.

### 4. Development:

* **Team Coordination:** Organized Joint Application Development (JAD) sessions.
* **Conflict Resolution:** Addressed disagreements among team members and encouraged collaborative decision-making.
* **Requirement Clarification:** Provided ongoing support to developers and testers.
* **Meeting Management:** Recorded sessions and documented discussions for absent team members.

### 5. Testing:

* **Test Case Execution:**
  + Verified login functionality, search features, and document uploads.
  + Tested system authentication and security.
  + Checked workflow automation for approvals and rejections.
* **User Acceptance Testing (UAT):** Conducted UAT sessions and gathered feedback.
* **RTM Update:** Ensured all requirements were validated against test cases.

### 6. Deployment:

* **Client Handover:** Provided end-user manuals and conducted training sessions.
* **Final Documentation:** Delivered RTM and project closure reports.
* **Post-Deployment Support:** Assisted with initial troubleshooting and system optimization.

END

------------------------------------------------------------------------------------------------------------