Features Proposal for Project Option Mining Digital Work Artifacts

Team Number: 9
Team Members: Sami Jaffri 44165611, Kevin Zhang 10811057, Ryan Eveson 99775389, TianXing(Eric) Chen 47368527, Evan Pasenau 36403509, Jinxi Hu 48528608

1 Project Scope and Usage Scenario

Explain in one paragraph the basic usage scenario you intend to cover. This scenario may involve multiple user groups – be sure to clearly identify them (for example, an educational app may have *students*, *instructors*, and *administrators* as three different user groups.

The basic usage scenario involves graduating students and early-career professionals as the primary user group who regularly create digital work artifacts on their computers and need to systematically catalog and analyze their professional output for career advancement purposes. A secondary user group consists of career counselors and academic advisors who may guide students in using these insights to better articulate their professional growth and identify areas for skill development.

2 Proposed Solution

Explain in one to two paragraphs what your solution is. Highlight special features, or special technologies, that you are using. What is your value proposition? What do you think you will do better in comparison to other teams? State your main points here clearly. Be concise and catch your reader's attention immediately.

Our project provides value to students, employees, and their managers by quantifying the volume of work they have completed and providing insightful metrics to summarize their contributions concisely and discreetly. We will use Python Flask api's and connect to a Docker SQL database, which will allow us to collect a lot more data and find much deeper insights in the data than if we tried to process all the data locally. This will allow us to customize the graphs and metrics we show to each user based on what stands out the most for them, separating us from other teams that will likely show the same metrics to all users. Another way we will separate ourselves from other groups is by using a cloud-based deployment with our Flask api's will also allow us to gather files from multiple sources at once, making it easier for users to upload all their data and increasing the confidence and accuracy of our data analysis.

3 Use Cases

Based on the usage scenario, describe all the use cases in detail. Do this by providing a UML use case diagram. Then for each use case in your diagram, identify: the name of the use case, the primary actor, a general description, the precondition, the postcondition, the main scenario, and possible extensions to consider. Below is an example of a use case description.

All UML Diagrams are located at the end of the Document

Use Case 1: Scan & Ingest Artifacts

- **Primary actor:** Student/Early Professional
- **Description:** This use case describes the process where the user connects to one or more data sources like Github, Google Drive or local files, initiates a scan to collect and ingest digital work artifacts. The system fetches files and their associated metadata, standardizes them and stores them in a database for later searching, analytics, and portfolio generation.
- **Precondition:**
  1. The user must be logged into the system and authenticated via OAuth2 or other secure methods.
  2. At least one data source (e.g., GitHub account, local scan agent, or Google Drive) has been successfully connected.
  3. The user has granted proper permissions for the system to access these data sources.
- **Postcondition:**
  1. The scanned artifacts and metadata are stored in the database.

2. The scan job status is updated to *Completed* or *Failed*, and the user can view the scan summary on the dashboard.
3. An initial search index is generated, allowing the user to browse, search, and filter artifacts.

- **Main Scenario:**
  1. The user clicks the **"Start Scan"** button on the dashboard.
  2. The system prompts the user to select which data sources to scan (GitHub, Google Drive, or local sources).
  3. The user confirms their selection and submits the scan request.
  4. The system queues the scan task and returns a unique **Job ID** to the user.
  5. The backend scanning service retrieves files and metadata from the selected sources.
  6. The system processes and normalizes the scanned data while removing duplicates.
  7. Processed data and metadata are stored in the database, and an initial search index is created.
  8. The system updates the job status to **Completed** and displays a scan summary to the user.
  9. The user can view and interact with the imported artifacts on the dashboard.
- **Extensions:**
  1. Data Source Unavailable
     - In step 5, if a selected data source is unreachable (e.g., API rate limit or network issue), the system marks that source as Failed while continuing to scan other sources. A warning is displayed in the scan summary.
  2. User Cancels the Scan
     - Between steps 4 and 6, the user can cancel the scan job. The system safely terminates the process and marks the status as *Canceled*.
  3. Metadata Parsing Error
     - In step 6, if a file format is unsupported or parsing fails, the system logs the error, skips the problematic file, and flags it in the scan summary.
  4. Duplicate Detection
     - During step 6, if an artifact already exists in the database, the system updates its metadata instead of duplicating the record.

Use Case 2: Summarize and Analyze Project Experience

- **Primary actor:** Student
- **Supporting actors:** Programming Staff (for setup/integration and access provisioning)
- **Description:**

  The system reads a user-specified local folder or GitHub repository, analyzes the project contents, and produces an analytical report. The report includes (at minimum) total project size, counts of files by type, total lines of code, and programming-language usage distribution. Results are saved as files and rendered in the UI.
- **Preconditions:**
  1. The user is logged in.
  2. The user has specified an accessible local folder or GitHub repository.
  3. The system has permission to read the folder/repository.
- **Postconditions:**
  1. A summary visualization (charts/tables) is generated and saved.
  2. The same visualization is displayed in the UI and can be downloaded.
  3. The analysis metadata (source, timestamp) is logged.

- **Main Scenario:**
    1. User navigates to **Project Analysis** and clicks **New Analysis**.
    2. User selects a data source: **Local Folder** or **GitHub Repository**.
    3. If GitHub is chosen, the system prompts for authorization; if Local Folder is chosen, the system requests a folder path.
    4. User confirms input and clicks **Start Analysis**.
    5. System validates the source and queues the job.
    6. System retrieves project contents and collects metadata.
    7. System classifies files, counts lines of code, and detects programming languages.
    8. System generates metrics and visualizations.
    9. System saves the results and displays them in the UI.
    10. User views the results and may download the report.
- **Extensions / Alternate Flows:**
    1. Invalid source provided → System shows error, user selects a valid one.
    2. Permission denied or authorization expired → System prompts re-authorization.
    3. No analyzable files found → System returns a minimal report.
    4. Analysis job fails or times out → System notifies user and suggests retry.
    5. User cancels the job → System stops processing and marks the job as canceled.

Use Case 3: Build and Edit Resume / Portfolio

- **Primary Actor:** Student / Early Professional

- **Description:**
  This use case covers the end-to-end process of creating, customizing, and editing a resume or portfolio using the artifacts and analytics stored in the system. The user can select projects, analytics charts, and highlights to include, export the result in various formats (PDF, Markdown, web page), and make further edits or updates after the initial version is generated.

- **Precondition:**
    - The user must be logged into the system.
    - The system must have at least one completed scan containing digital work artifacts and metadata.
    - The user has sufficient permissions to generate and export documents.

- **Postcondition:**
    - A completed and potentially edited resume/portfolio file is generated and saved in the system.
    - The file is available for download or online sharing.
    - The generated portfolio is linked to the user profile and accessible for viewing by authorized third parties (e.g., HR).

- **Main Scenario:**
    1. The user navigates to the Portfolio Builder page.
    2. The system displays available artifacts and analytics data.
    3. The user selects projects, highlights, and metrics to include.
    4. The user chooses a format for export (PDF, Markdown, or Web page).
    5. The system generates a preview of the resume/portfolio.
    6. The user reviews the preview, makes edits (adding/removing artifacts, reordering sections, or adjusting content), and confirms export.

7. The system finalizes the resume/portfolio, saves it to the database, and provides a download link or shareable URL.
8. The system updates the user profile to reflect the new or updated portfolio version.

- **Extensions:**

    - **Missing Artifacts:** If no artifacts are available for selection, the system displays a message and suggests running a scan (Use Case 1).
    - **Export Failure:** If the export process fails (e.g., server timeout), the system notifies the user and allows retry.
    - **Format Not Supported:** If the user selects a format not currently supported, the system grays out that option and displays available choices.
    - **Unsaved Changes:** If the user tries to exit while editing, the system prompts them to save or discard changes.
    - **Version Control:** The system may allow multiple portfolio versions to be saved instead of overwriting the existing one.

Use Case 4: Search & Filter Artifacts

- **Primary actor:** Student/Early Professional
- **Description:** This use case allows the user to search and filter through scanned and stored digital work artifacts. The system provides keyword search and multiple filters such as file type, project tags, date range, and source (GitHub, Google Drive, Local). This feature helps the user locate specific files or groups of files efficiently.
- **Precondition:**
    1. The user must be logged into the system.
    2. The system must already have scanned and ingested artifacts stored in the database (Use Case 2).
- **Postcondition:**
    1. The system displays a list of artifacts matching the search criteria.
    2. Selected search results can be used for further actions like analysis or portfolio building.
- **Main Scenario:**
    1. The user navigates to the **Search Dashboard**.
    2. The user enters keywords or applies filters such as file type, date range, or tags.
    3. The system queries the indexed database.
    4. Matching artifacts are displayed in a results list with metadata such as file name, source, and last updated date.
    5. The user can sort results by relevance, date, or file type.
    6. The user selects specific files for viewing, analysis, or adding to a portfolio.
- **Extensions:**

    1. No Results Found
        - If no artifacts match the search, the system displays a message suggesting adjustments to filters or keywords.
    2. Network Delay
        - If search takes longer than expected, the system displays a loading animation and partial results when available.
    3. Invalid Filter Combination
        - The system warns the user if conflicting filters are applied (e.g., incompatible date ranges).

Use Case 5: View Portfolio

- **Primary actor:** HR Representative
- **Supporting Actor:** Student / Early Professional (portfolio owner)

- **Description: This use case describes how an HR representative or recruiter views a student or early professional's portfolio through a secure link or login. The portfolio includes highlighted projects, analytics, and resume data generated in Use Case 4.**
- **Precondition:**
    1. The student has generated at least one resume or portfolio.
    2. HR representative has either been given a shareable link or login credentials.
    3. Access permissions are properly configured to prevent unauthorized access.
- **Postcondition:**
    1. HR representative successfully views the portfolio content.
    2. System logs access details for auditing (time, IP, viewer ID).
- **Main Scenario:**
    1. HR representative opens the provided link or logs into the system.
    2. The system verifies access credentials and permissions.
    3. The portfolio is loaded and displayed, showing projects, metrics, and analytics.
    4. HR can navigate through different sections such as project history, technical skills, and achievements.
    5. HR can download a PDF version of the resume if enabled by the student.
- **Extensions:**
    1. Expired Link
        - If the link has expired, the system displays an error message and instructs the HR representative to request a new link from the user.
    2. Permission Denied
        - If access permissions are invalid, the system prevents viewing and logs the failed attempt.
    3. Download Disabled
        - If PDF download is disabled, the system hides the download option.

4 Requirements, Testing, Requirement Verification

Start by mentioning your technology stack and test framework. Then use the example table below to identify the requirements, test cases, and people assigned to work on those requirements.

- Based on your use cases above, flush out the necessary requirements. These use cases will only identify functional requirements. You will want to think about the usage scenario and the client's needs to identify non-functional requirements also.
- For each requirement, develop test cases - positive and negative cases - that can be written in the code base and automated. While non-functional requirements such as "usability" will require manual testing (not by your team), most of the requirements should have automatable test cases.
- Discuss with your teams the difficulty of each requirement and who wants to be assigned to them. Choose one of: hard, medium, or easy and put it in the column for "H/D/E".

Requirement	Description	Test Cases	Who	H/M/E
File discovery	Automatically scan and identify work artifacts across file system	Verify tool finds all test files in nested directories within 30 seconds	Student	Med
Metadata Extraction	Extract creation dates, modification history, file types, and sizes	Validate metadata accuracy against known file properties for 100+ test files	Students	Easy
Code Analysis	Analyze programming languages, complexity metrics, and repository structure	Confirm correct language detection and LOC counting for sample repositories	Students	Hard
Portfolio Generation	Create exportable summaries and project descriptions	Generate portfolio content and verify formatting matches user requirements	Professionals	Med
User Authentication	Secure access to personal work analysis	Test login/logout functionality with various password scenarios	All users	Easy

—----------

Technology Stack & Test Frameworks

- **Primary actor:** Student/Early Professional
- **Backend**: Python Flask for REST APIs
- **Frontend**: React + TypeScript
- **Database**: Dockerized PostgreSQL for storing artifacts and metadata
- **CI/CD Deployment**: GitHub Actions + Cloud-based deployment
- **Test Frameworks**:
    - **Backend Tests:** Pytest (unit & integration)
    - **Frontend Tests:** Jest + React Testing Library
    - **API Testing:** Postman / Newman
    - **Automation:** GitHub Actions for continuous integration

1. **Requirements**

   Based on the use cases, the following functional requirements (FR) and non-functional requirements (NFR) have been identified.

ID	Type	Description	Linked Use Cases
FR-01	Functional	The system shall allow users to connect to at least one external data source (GitHub, Google Drive, or Local Folder) via OAuth2 or secure authentication methods.	UC2: Scan & Ingest Artifacts
FR-02	Functional	The system shall allow users to upload files from multiple local folders or devices for scanning.	UC2: Scan & Ingest Artifacts
FR-03	Functional	The system shall scan connected data sources and collect digital work artifacts and associated metadata.	UC2: Scan & Ingest Artifacts
FR-04	Functional	The system shall standardize, categorize, and index collected files by type (e.g., code, documents, images).	UC2, UC3
FR-05	Functional	The system shall provide search and filter functions for imported artifacts.	UC3
FR-06	Functional	The system shall generate analytics visualizations (e.g., project size, file counts, language distribution).	UC3
FR-07	Functional	The system shall allow the user to generate a portfolio export (PDF/Markdown/Web page) from selected artifacts.	UC1
NFR-01	Non-Functional	The system shall process up to 500 files in less than 10 seconds per scan.	UC2
NFR-02	Non-Functional	All data in transit must be encrypted using TLS 1.2+ to ensure security.	UC2, UC3
NFR-03	Non-Functional	The system shall maintain 99% uptime during scanning and analytics operations.	UC2, UC3
NFR-04	Non-Functional	The UI shall be responsive and load in under 3 seconds on standard broadband.	UC1, UC2, UC3

2. **Testing Plan**

   For each functional and non-functional requirement, positive and negative test cases are provided to ensure comprehensive coverage.

ID	Type	Description	Expected Result
FR-01	Positive	Connect GitHub via OAuth2 with a valid token.	GitHub account successfully linked and token saved securely.
FR-01	Negative	Connect GitHub with expired token.	The system rejects the token and prompts for re-authentication.
FR-02	Positive	Upload files from two different folders	All files appear on the dashboard and are ready for scanning.
FR-02	Negative	Attempt upload with no files selected.	The system displays error messages and blocks scans.
FR-03	Positive	Start scan with GitHub and Google Drive connected.	The system fetches files and metadata from both sources.
FR-03	Negative	Start scan with unreachable data source (e.g., network issue).	The system marks that source as failed and continues scanning others.
FR-04	Positive	Upload mixed file types (PDF, code, images).	Files are categorized correctly by

			type.
FR-05	Positive	Search for artifacts by keyword.	Correct artifacts are returned in search results.
FR-05	Negative	Search with invalid query or empty keyword.	The system returns no results and displays a proper message.
FR-06	Positive	Generate analytics from a sample GitHub repository.	Accurate project size, file counts, and language breakdown displayed.
FR-07	Positive	Export selected projects as PDF portfolio.	Exported PDF contains correct selected artifacts
NFR-01	Performance	Upload and scan 500 files.	Scan completes in ≤ 10 seconds.
NFR-02	Security	Inspect network requests via developer tools.	All requests are encrypted using HTTPS (TLS 1.2+).
NFR-04	Usability	Open dashboard on mobile device.	Dashboard loads fully in ≤ 3 seconds and is fully responsive.

3. Requirement Verification

A Requirements Traceability Matrix (RTM) ensures each requirement is verifiable and linked to its respective test case(s).

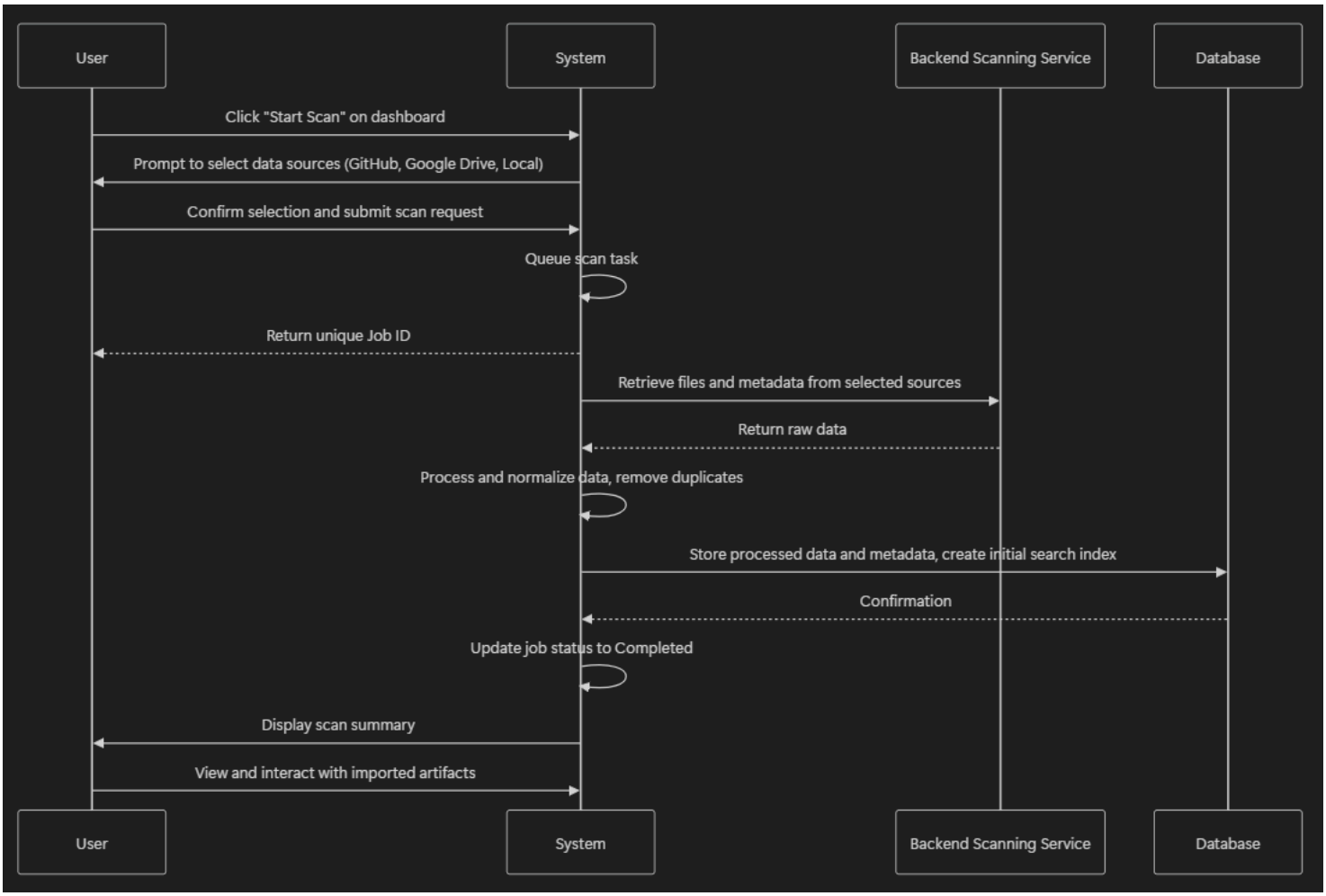
ID	Linked Use Case	Verification Method	Automation
FR-01	UC2	API Test: OAuth2 login workflow.	Automated
FR-02	UC2	Functional Test: Upload multiple folders and devices.	Automated
FR-03	UC2	Integration Test: Scanning multiple data sources.	Automated
FR-04	UC2, UC3	Functional Test: File categorization and indexing.	Automated
FR-05	UC3	Functional Test: Search and filter queries.	Automated
FR-06	UC3	Integration Test: Analytics accuracy.	Automated
FR-07	UC1	End-to-End Test: Portfolio generation workflow.	Automated
NFR-01	UC2	Performance Test with large dataset.	Semi-automated
NFR-02	UC2, UC3	Manual security review and automated API tests.	Hybrid
NFR-03	UC1, UC2, UC3	Manual usability testing on different devices.	Manual

4. Workload Distribution (Assigned Member need to change later!!!)

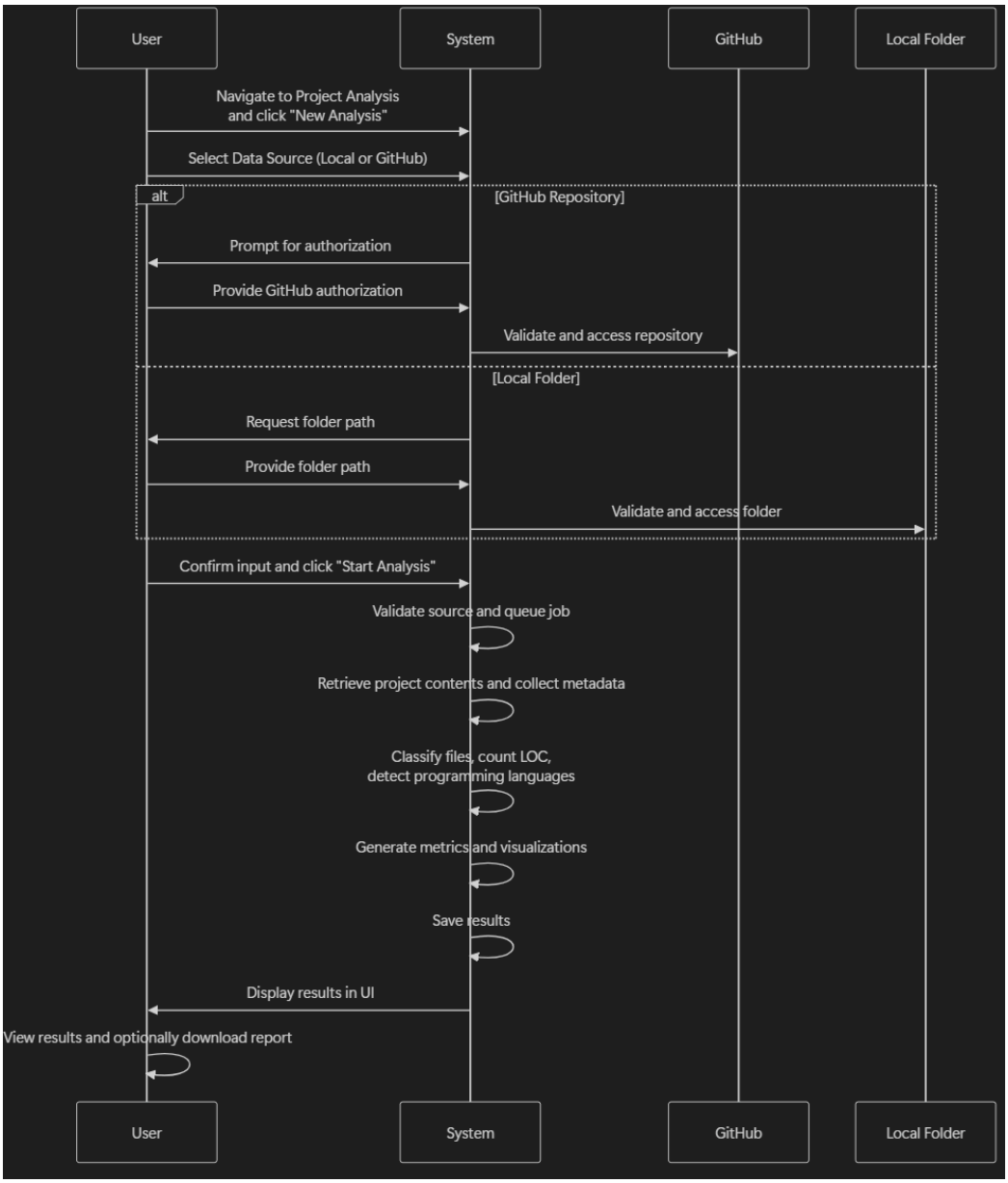
ID	Assigned Member(Full Name)	Difficulty(HARD/MEDIUM/EASY)
FR-01, FR-02	Sami	Medium
FR-03	Kevin	Hard
FR-04	Ryan	Hard
FR-05, FR-06	Evan	Medium
FR-07	Eric	Medium
NFR-01	Jinxi	Hard
NFR-02	Kevin	Medium
NFR-03	Eric	Easy

UML Diagrams

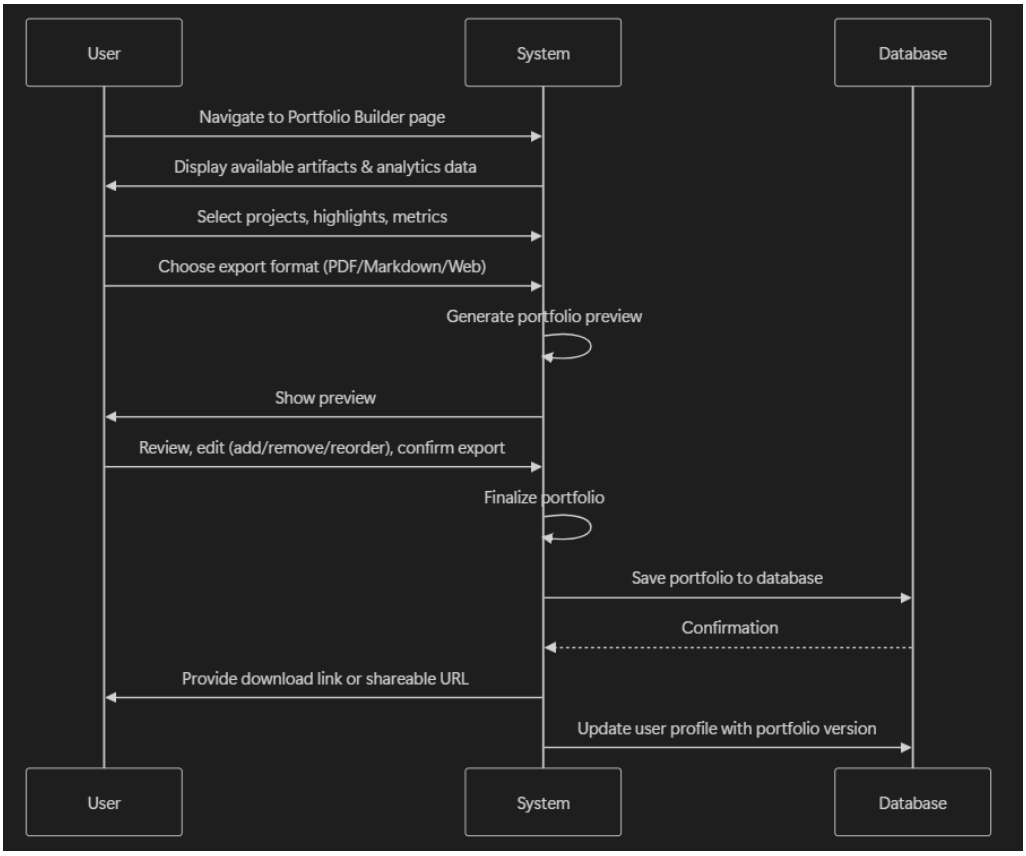
Use case 1



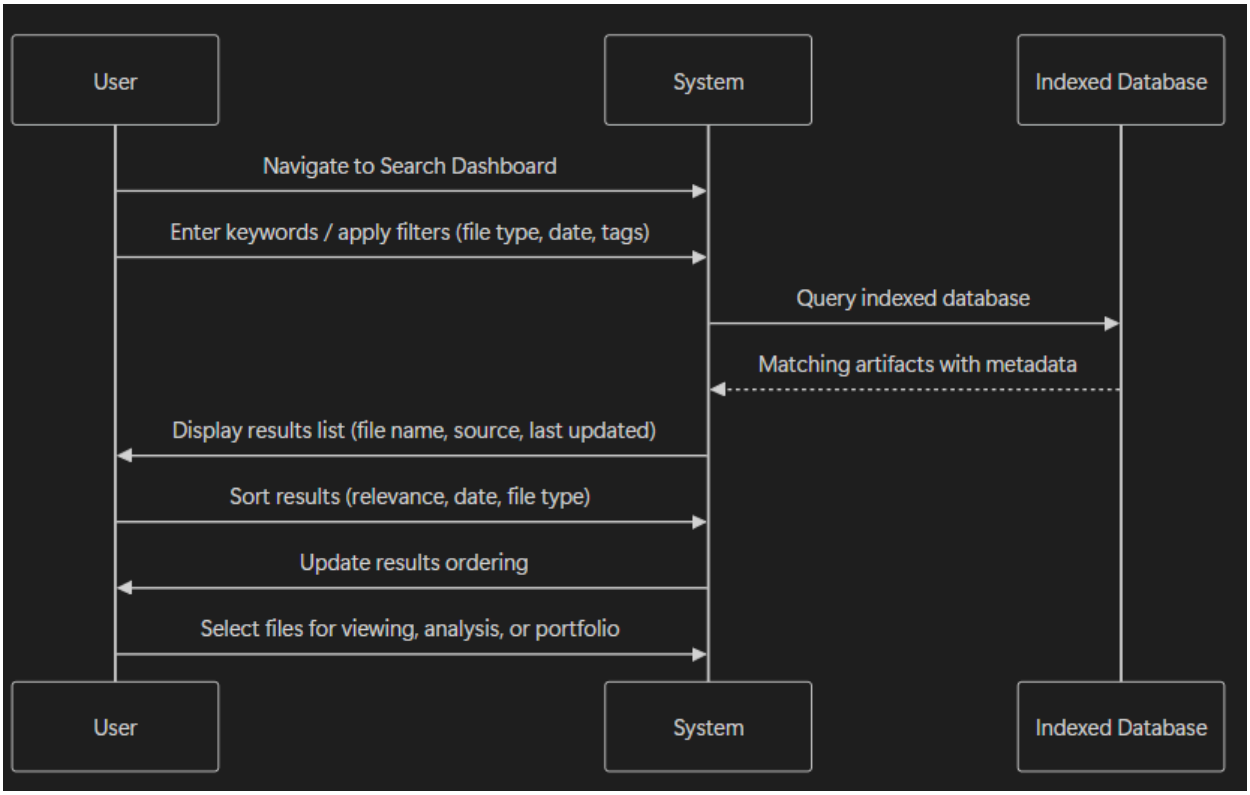
UML Diagram 2



UML Diagram 3



UML Diagram 4



UML Diagram 5