Table of Contents

List of Figures

List of Tables

Acronyms/Definition of Terms

Content Uploader: Uploads historical content (text, images, etc.) for their institution.

Institution: Approved organization contributing content.

Institutional Admin: Manages their institution's account on the platform (staff, content).

Platform Admin: Oversees the entire platform.

Premium User: Paid subscription for access to more features or content.

Public User: Has free access to basic content.

Researcher: Conducts academic research and have access to restricted content.

Reviewer: Approves researcher access or uploaded content based on quality standards.

List of Abbreviations

E.C	Ethiopian calendar
EHPTS	Ethiopian History Preservation and Transmission System
FR	Functional Requirement
ID	Identification
IDE	Integrated Development Environment
NALA	National Archives and Library Authority
NFR	Non-Functional Requirement
OOP	Object-Oriented Programming
OOSAD	Object Oriented System Analysis and Design
SDLC	Software development lifecycle
UD	Use case diagram
UML	Unified modeling language

Executive Summary

The Ethiopian History Preservation and Transmission System (EHTPS) serves as a centralized platform for safeguarding and sharing Ethiopia's rich historical heritage. It facilitates the registration of institutions, allowing them to upload and contribute various historical content formats, including text documents, photographs, audio recordings, and videos. By providing tiered access levels, the EHTPS caters to a diverse user base, ranging from casual history enthusiasts to dedicated researchers. A robust user management system ensures a secure and personalized experience, while functionalities for content management empower institutions to contribute valuable historical materials. This centralized approach provides collaboration and knowledge exchange, making the EHTPS a promising tool for promoting historical awareness and fostering a deeper understanding of Ethiopia's past

Chapter 1

1. Introduction

1.1 Background

Ethiopia, located in the Horn of Africa, is one of Africa's oldest and most historically significant countries. Its history is a rich and complex tapestry of cultures, languages, and traditions that have evolved over centuries. Understanding how Ethiopians have transmitted their history is essential to appreciating the vibrant cultural landscape, diverse languages, and enduring traditions that define the nation [1].

This section will explore Ethiopian oldest history sharing methods, and how these methods of remembering and sharing history have changed over time.

1. **Ethiopia's Oral Tradition:** For thousands of years, people told stories about their kings, wars, and how things changed. These storytellers were called Guezal Tewolde and their stories weren't just fun, they taught people important things about their past. They even had different ways of telling stories, like long poems about kings, catchy songs about heroes sung in an old language called Ge'ez, and family stories passed down that told the history of a specific place [2].
2. **Written Records & the Role of the Church:** The arrival of Christianity 1600 years ago ushered in written records. Monks, the new custodians of knowledge, documented history in Ge'ez. These written forms complemented oral tradition, ensuring Ethiopia's rich past continues to resonate [3]. Also churches like Addis Ababa's St. George's Cathedral showcases treasures like coronation robes and religious artifacts, offering a glimpse into Ethiopia's religious and imperial past [4].
3. **Modern Approaches: Archives:** In recent years, Ethiopia has found new ways to keep its history safe. The government built special places called archives and libraries to collect and protect all the important documents and information about Ethiopia's past. Museums have popped up all over the country, showing off historical objects.

- **National Archives and Library of Ethiopia:** The National Archives and Library of Ethiopia, built in 1978 E.C , is a special place where they collect and keep safe all the important documents and information about Ethiopia's history. They also let researchers come and study this stuff [5].
- **Addis Ababa National Museums:** was established in 1986 E.C and serve as repositories for the nation's historical treasures. These museums house artifacts of daily life, archaeological finds, and captivating exhibits that illuminate Ethiopia's past [6].

1.2 Statement of the problem

This Section explores the critical issues surrounding Ethiopia's history transmission.

- **Scattered Resources:** Valuable historical documents, artifacts, and chronicles are dispersed across various locations – museums, libraries, churches, monasteries, and even private collections. This fragmentation creates significant obstacles for researchers and the public to locate and access these materials, hindering comprehensive research and creating a fragmented understanding of the historical narrative. And also without a unified system for managing these records, it becomes challenging to ensure their long-term preservation.
- **Limited Infrastructure:** Libraries, museums, and archives – the key locations for accessing historical knowledge – may not be readily available in all regions of Ethiopia, particularly rural areas. This creates an information gap, where certain communities are left disengaged from their own history.
- **Limited Formats:** Historical information might not be presented in engaging ways for a wider audience. Traditional resources like oral histories, while valuable, may not reach everyone.

1.3 The Project's Purpose:

The Ethiopian History Preservation and Transmission Platform is designed to bridge the gap in Ethiopian history transmission. Recognizing the limitations of current methods, the project aims to achieve the following:

1. **Create a Centralized Hub:** By digitally archiving historical documents, artifacts, and photographs, the system will establish a single, easily accessible online location for the public. This ensures these valuable materials are preserved for future generations and allows researchers and history enthusiasts to collaborate and share knowledge.
2. **Engage the Public:** The system will develop interactive and engaging resources for the general public. Utilizing digital tools and platforms, the platform will make historical information accessible beyond traditional academic institutions, allowing everyone to explore Ethiopia's rich past.
3. **Strengthen National Identity**: By making history accessible and engaging, the system aims to strengthen a sense of national identity among Ethiopians. This appreciation for their heritage empowers future generations to become guardians of their history and culture.

1.4 Objective

1.4.1 General objective

To develop a user-friendly and visually appealing web-based platform that serves as a central repository for preserving and promoting knowledge about Ethiopian history.

1.4.2 Specific objectives

- To conduct a thorough analysis of current system to identify limitations in accessibility, contextual information, and user engagement.
- To design a strong UML diagram to effectively store, manage, and retrieve diverse historical data collected for the platform. This model should facilitate future expansion of the platform.

- To design a clean, intuitive, and user-friendly interface that caters to a broad audience with varying technical skills. This includes ensuring easy navigation, clear information presentation, and accessibility features that comply with relevant standards.
- To enhance the user experience beyond usability by incorporating visually appealing design elements that capture user attention and encourage exploration of historical content.
- To develop and deploy a fully functional web-based platform that adheres to the design specifications and fulfills both functional and non-functional requirements as objectively measured through testing procedures.
- To document the entire project process objectively, including design decisions, challenges encountered, solutions implemented, and testing results. This documentation serves as a valuable resource for future iterations, system improvements, and knowledge sharing.

1.5 Scope and limitations of the project

1.5.1 Scope of the project

The Ethiopian History Preservation and Transmission System (EHPTS) aims to bridge the digital divide and make Ethiopia's rich history accessible to all Ethiopians. This web-based platform will create a centralized repository of digitized historical materials, including documents, artifacts, and oral histories.

Focus on Accessibility: The EHPTS prioritizes reaching a broad Ethiopian audience, with content available in English and Amharic. The user interface will be designed for ease of use, allowing users to search, browse, and learn about Ethiopian history.

User Roles: A tiered user system will manage access. Public users will have free access to basic information, while premium users will have access to a wider range of content for a fee. Researchers will have a separate tier with limited access to restricted materials, following strict copyright guidelines.

1.5.2 Limitations of the project

Our project strives to be inclusive and impactful, but navigating these limitations will be crucial:

- **Resource Constraints:** Limited resources may require prioritizing the digitization of critically important and at-risk materials initially.
- **Content Acquisition:** Gaining permission to digitize materials from private collections and religious institutions might require careful negotiation and collaboration strategies.
- **Digital Literacy:** Limited internet access and digital literacy in some regions.
- **Copy right issues**: Works protected by copyright might require obtaining permissions from copyright holders before digitizing and making them accessible online. Strategies will be needed to navigate copyright restrictions and ensure legal compliance.

1.6 Project Tools

This section dives into the various tools that brought our Ethiopian history website. Each tool played a specific role in the development process:

1.6.1 Planning and Organization

- **Microsoft Word:** likely used to create documents outlining the application's purpose, functionalities, development tasks, and a comprehensive project plan. This ensured a clear roadmap for development [7].
- **Google Forms:** is a free online tool to create surveys and questionnaires. You can use it to gather information, opinions, or feedback easily [8].

1.6.2 Designing the Look and Feel

- **Draw.io:** This program helped visualize the application's structure through user flow diagrams and potential layouts. Think of it as a digital whiteboard to brainstorm and plan the user experience [9].
- **Figma:** This popular design tool likely came in handy for crafting high-fidelity mockups. These mockups showcase the application's visual style on screen, including page layouts,

buttons, and other interactive elements. Imagine creating digital prototypes that look and feel like the final product [10].

1.6.3 Coding Environment (IDE)

- **Visual Studio Code:** The editor where the website code was written. It helps with readability, error checking, and collaboration [11].

1.6.4 Frontend Development

- **Next.js** is a React framework for building fast, server-rendered, and static web applications with features like routing, API support, and optimized performance. [12].
- **Tailwind CSS:** A utility-first CSS framework that allows for rapid styling of the website [13].
- **TypeScript:** is a superset of JavaScript that adds static typing for improved code quality and maintainability.

1.6.5 Backend Development

- **Next.js:** Built the server-side logic that handles data requests and interacts with the database [14].
- **PostgreSQL:** is a powerful, open-source relational database system known for its robustness, scalability, and support for advanced features like complex queries and JSON storage. [15].

1.6.6 Keeping Track of Changes

- **Git:** Tracks code changes, allows for collaboration, and lets you revert to previous versions if needed [16].

1.6.7 Hardware

Personal computer: Hp-DESKTOP-854KJI4. It has an Intel Core i7-10510U processor with 8GB of RAM. It runs a 64-bit operating system.

1.7 Methodology

1.7.1 System development life cycle

This part will explain about the methodology that will be used in our project. Every software project need to follow a framework call software development lifecycle (SDLC) to make sure a high quality software is created. This is because SDLC defines the task to be performed at each step in the software development Process which will make sure our work is planned, organized and following the schedule [17].

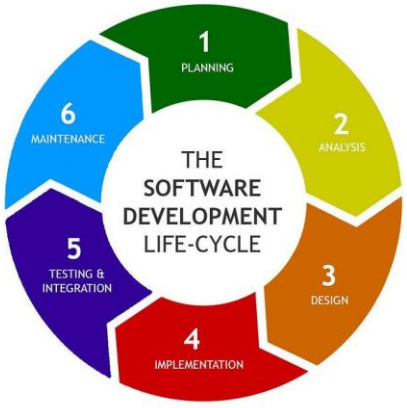


Figure 1.1 The software development lifecycle (SDLC) diagram [17]

Following a comprehensive assessment of various Software Development Life Cycle (SDLC) methodologies, we opted for an Iterative waterfall model best suited for our web-based Ethiopian history preserving and transmission system. This choice aligns with the project's structured nature and the curriculum's well-defined milestones.

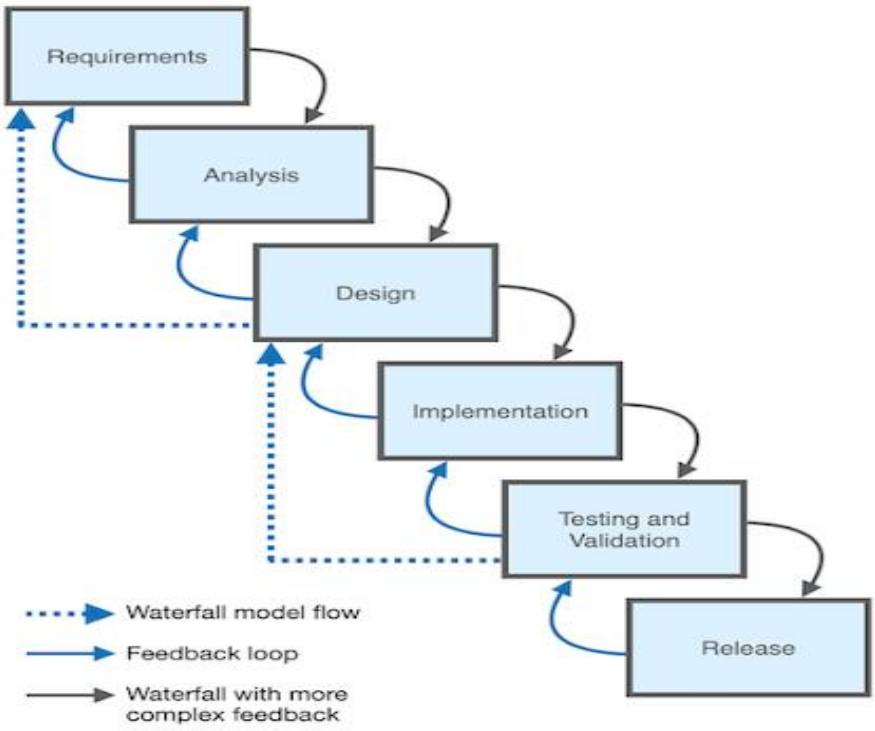


Figure 1.2 Iterative waterfall model diagram [18]

Key advantages of using Iterative waterfall model

- **Structured & Phased Approach:** The model enforces a clear, step-by-step process with defined deliverables at each stage. This minimizes errors and ensures efficient resource allocation throughout development [18].
- **Iterative Refinement:** While the overall model is linear, each phase allows for iterative improvement based on feedback. This ensures the final system aligns with user needs and project goals [18].
- **Adaptability to Change:** The model allows for some flexibility to accommodate new discoveries or changing requirements. This ensures the final system remains relevant as the understanding of Ethiopian history evolves [18].

1.7.2 Requirements Gathering Methodology

A successful system development lies in a comprehensive understanding of user needs and project constraints. To achieve this, a multifaceted requirements gathering methodology will be employed, encompassing the following techniques:

1. Document Analysis: A thorough analysis of relevant Ethiopian educational resources, including curriculum materials, historical documents, and existing digital archives, will be conducted. This analysis aims to identify existing practices, potential knowledge gaps, and the educational context surrounding Ethiopian history preservation and transmission.

2. Stakeholder Interviews: Semi-structured interviews will be conducted with a diverse range of stakeholders. This includes folks from museums, monasteries, libraries, and archives,

3. Observational Studies: Direct observation of existing methods of history preservation and transmission will be undertaken. This involve visiting museums, libraries, and cultural centers to understand current practices for storing, exhibiting, and disseminating historical information.

4. Questionnaires: A questionnaire will be distributed to a representative sample of potential users. This instrument will gather user feedback on their preferred formats for consuming historical content (text, audio, video) as well as their desired functionalities within the web-based system (search capabilities, interactive elements).

By employing this multi-pronged approach, we aim to gather a rich and comprehensive set of requirements that accurately reflect the needs of stakeholders and the project's overall objectives.

1.7.3 System Design Paradigm: Object-Oriented Programming (OOP)

The Object-Oriented Programming (OOP) paradigm has been meticulously chosen as the system design paradigm for this project due to its inherent strengths in managing complex projects:

- **Structured & Maintainable Code:** OOP encourages well-defined objects with clear data and functionality. This modular approach leads to organized and easy-to-update code [19].

- **Flexible for the Future:** New features can be easily integrated by creating new objects or expanding existing ones. This adaptability is key for accommodating future discoveries or user needs [19].
- **Real-World Modeling:** OOP excels at modeling real-world entities and their relationships. This allows us to effectively represent historical figures, events, artifacts, and their connections within Ethiopian history [19].

1.8 Time Schedule

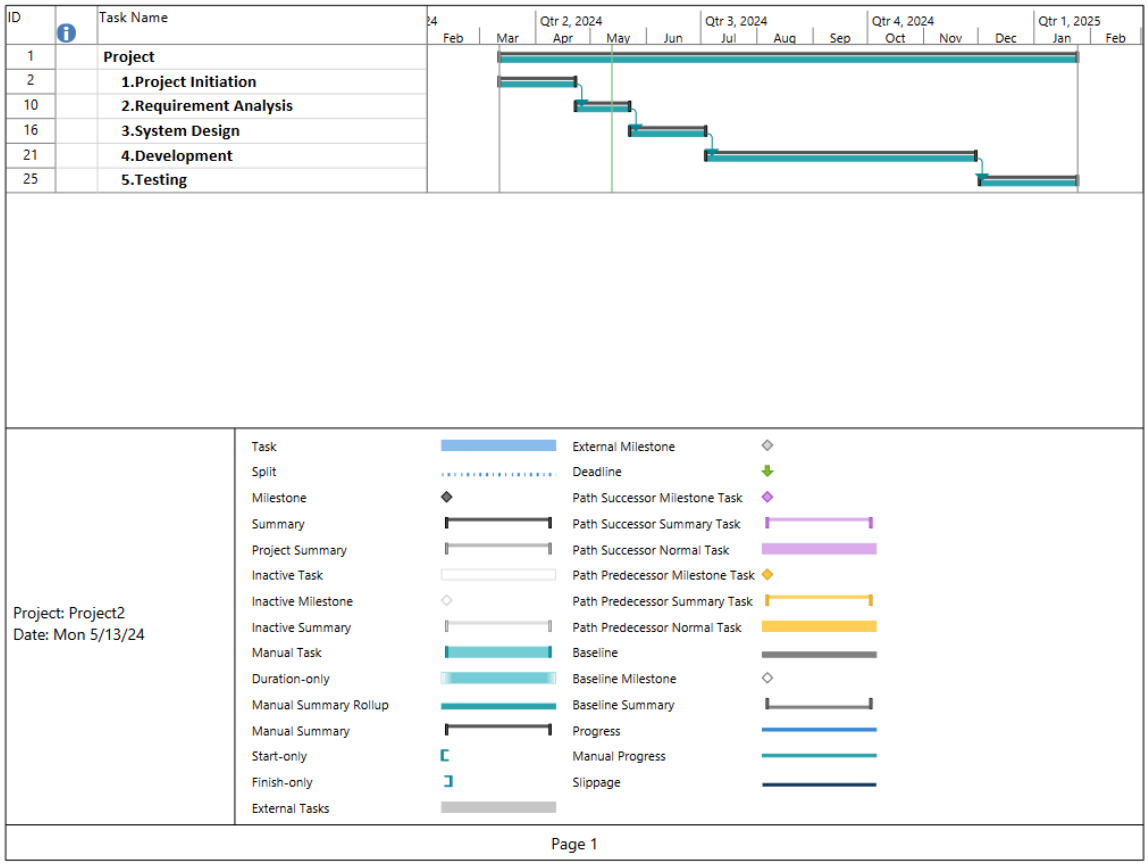


Figure 1.3 Time schedule to develop EHPTS

Chapter 2

2. Requirement Analysis

2.1 Introduction

The Ethiopian history platform we are creating will be a platform for preserving and sharing the rich story of Ethiopia's past. It will be a dynamic online space here users can dive deep into historical information, explore fascinating narratives, and connect with the wonders of Ethiopian history.

Why Requirements Analysis is Crucial

Before we dive headfirst into building this platform, it's essential to fully understand what users need and what we want the platform to achieve. This critical stage, called requirements analysis, acts as the blueprint for our project. By carefully examining these aspects, we'll be able to identify:

User Needs: What kind of information are users looking for? How do they prefer to access it? Understanding user needs allows us to design a platform that caters to their learning styles and preferences [10]

Project Goals: What are the key objectives we aim to achieve with this platform? Is it to increase accessibility to historical resources, raise awareness of Ethiopian history, or encourage people to engage with it more? Defining clear goals ensures the platform is developed with a specific purpose in mind [10].

By dedicating ourselves to this requirements analysis phase, we're laying the groundwork for a successful platform that fulfills the needs of its users and effectively achieves its intended goals.

2.2 Current System

In our investigation, we visited several key locations. First, we explored the National Archives and Library Authority (NALA), a vast repository overflowing with historical documents. Next,

we proceeded to Saint George's Cathedral, where a hidden gem of a museum houses a collection of religious artifacts that whisper stories of Ethiopia's past kingdoms. Finally, the Addis Ababa Museum offered a captivating journey through the city's history with its displays of royal garments and historical artifacts and art gallery. Here's a detailed look at the current situation:

- **National Archives and Library Authority (NALA):** It is a vast library, a true custodian of Ethiopian history. NALA houses an impressive collection of historical documents, whispering tales of the nation's past. However, its digital presence remains a mere library management system. While it helps navigate the physical collection, accessing these historical gems online remains a distant dream.
- **Saint George's Cathedral:** Standing tall, St. George's Cathedral embodies not only faith but also a hidden collection of history. Nestled within its grounds lies a museum, showcasing a captivating collection of religious artifacts. Crowns, crosses, and coronation garments whisper stories of past Ethiopian kingdoms, particularly those of Zewditu, Haile Selassie, and Menelik. Unfortunately, there's no dedicated website to unveil these treasures to the world.
- **Addis Ababa Museum:** Step inside the Addis Ababa Museum, Exhibits decorated with explanations showcase the city's rich tapestry. From royal garments to weapons used against the Italian colonial forces, each artifact speaks volumes. Yet, the museum's online presence is a mere shadow, focusing solely on announcements rather than mobilizing the power of the internet to share these treasures with a wider audience.

As we can see, Ethiopian history is not readily accessible through a single, centralized system. Instead, information is scattered across various resources, making it difficult for users to find comprehensive and reliable information.

2.2.1 Report generated by the current system

This section examines the reporting practices of key Ethiopian historical institutions, highlighting how they limit public access to valuable information.

National Archives and Library Authority (NALA): NALA's reports primarily focus on internal operations, utilizing tools like physical catalogs and internal inventory reports with

microfilm listings. These resources, while crucial for managing their vast collection, are not designed for public search or discovery.

Saint George's Cathedral Museum: Similar to NALA, the museum's reports likely address internal management needs, focusing on collection details, condition assessments. There is no indication of reports generated for public consumption, leaving the museum's collection shrouded in secrecy and inaccessible to the broader community.

Addis Ababa Museum: The museum's reports primarily revolve around temporary exhibitions, detailing attendance figures, visitor feedback, and logistical aspects in "Exhibition Reports." Annual reports might mention new acquisitions or ongoing projects, but they wouldn't offer a comprehensive picture of the entire collection for public access. This focus on temporary exhibitions, while valuable for gauging visitor engagement, neglects the vast historical resources housed within the museum, hindering public awareness and exploration of Ethiopian history.

2.2.2 Forms and Documents Used in Current System

This section analyzes the forms and documents utilized within the current systems at prominent Ethiopian historical institutions, highlighting the limitations for public access.

2.2.2.1 National Archives and Library Authority (NALA)

- **Internal Catalogs:** NALA maintains internal catalogs, potentially in physical binders and basic digital website to manage their vast collection. This comprehensive form likely captures all the details about each item, including descriptions, condition, and location. Additionally, it serves as a record of newly acquired artifacts or collections.

2.2.2.1.1 Catalog for books

The NALAS catalog form allows librarians to record details about library books. Fill in fields like title, publication date, and format. You can also track location information for easy browsing. NALAS offers a streamlined system for managing book collections.

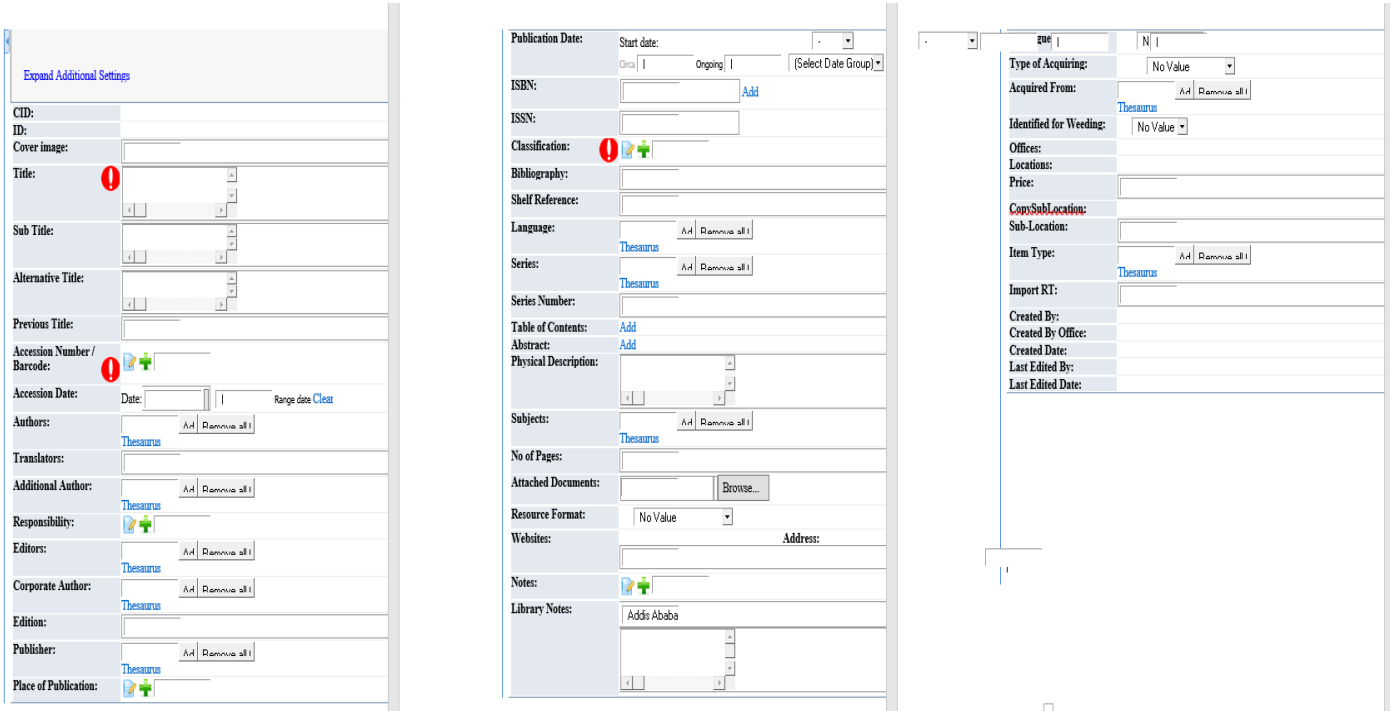


Figure 2.1 Catalog for books

2.2.2.1.2 Catalog for Music

The NALAS music cataloging form allows librarians to record details about music recordings. Fields include title, instruments, genre, and performer information. Physical media details and shelf location can also be entered. Although some data may require clarification, NALAS offers a comprehensive system for managing music collections in libraries.

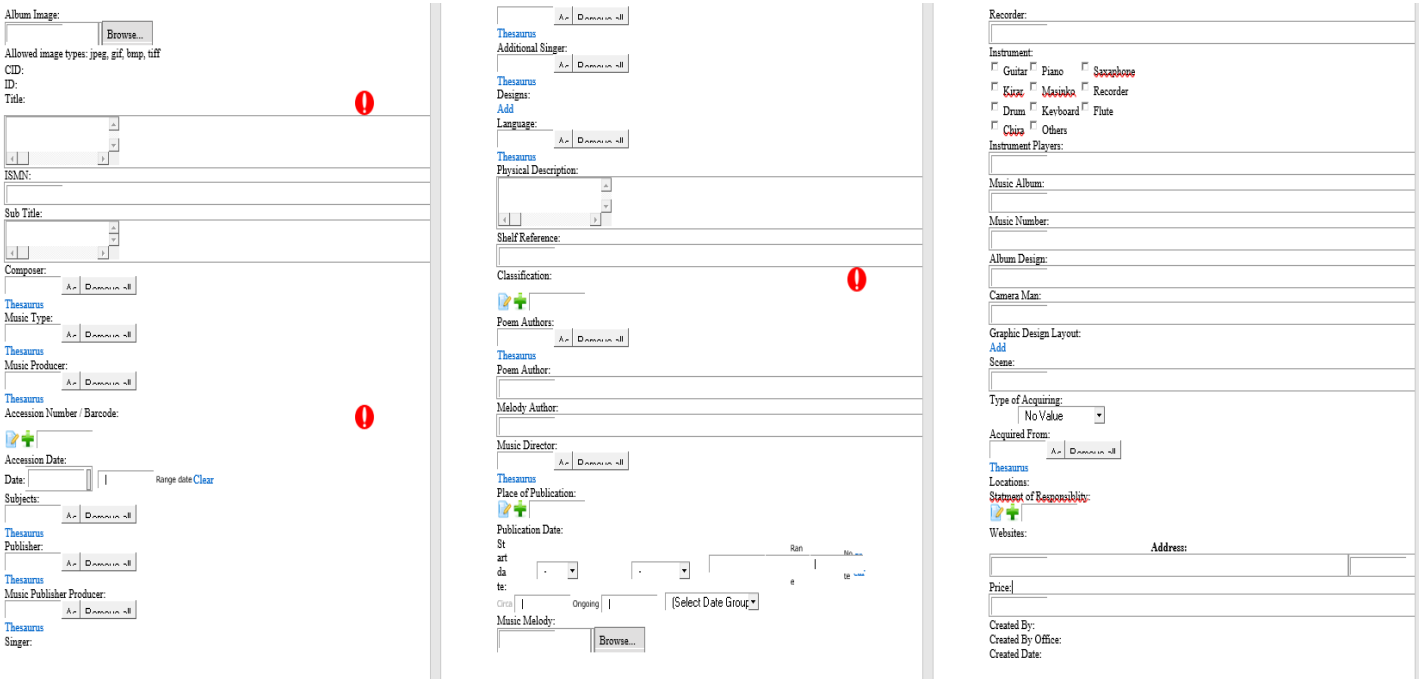


Figure 2.2 Catalog for Music

2.2.2.1.3 Catalog for Microfilm

The NALAS microfilm catalog form helps librarians track microfilm holdings. Key details include the microfilm's title, a unique ID number, and its physical condition. Information about the original microfilmed materials (can also be captured.

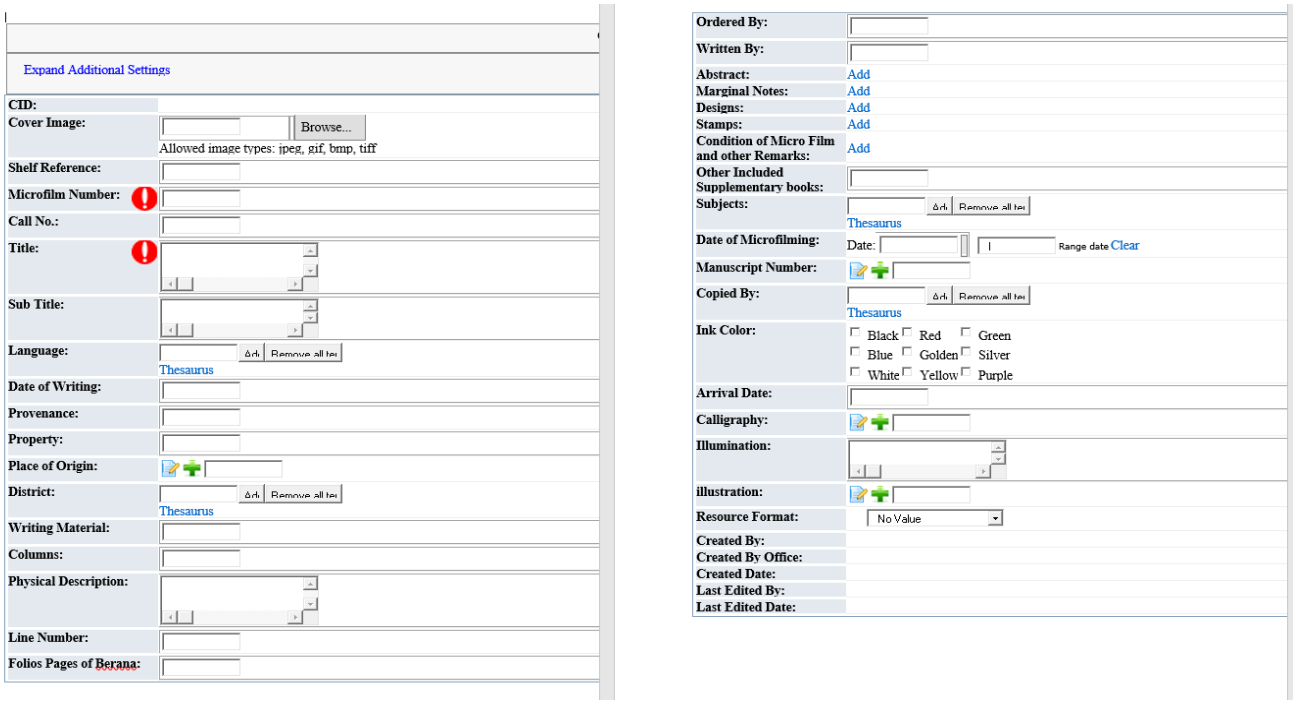


Figure 2.3 ***Catalog for Microfilm***

2.2.2.2 Saint George's Cathedral Museum

- **Collection Management Reports:** The museum utilizes internal reports detailing information about artifacts, including condition assessments, and other relevant data.
- **Acquisition Records:** Internal documents record newly acquired artifacts or collections entering the museum's possession.

2.2.2.3 Addis Ababa Museum

- **Exhibition Proposal Forms:** The museum employs forms for processing proposals from those seeking to host temporary exhibitions within their space.

- **Visitor Feedback Forms:** The museum utilize paper forms for visitors to provide feedback on their experience.
- **Acquisition Records:** Internal documents record newly acquired artifacts or collections entering the museum's collection.

We can't include Saint George's Cathedral Museum and Addis Ababa Museum internal forms in picture due to privacy concerns.

2.2.3 Players in the Existing System

- **Museums:** These are places that display historical objects, artwork, and writings. Some museums also have archives with historical documents and photos.
- **Libraries & Archives:** These are places that hold collections of historical documents, books, and magazines for people to learn from.
- **Monasteries & Temples:** These religious places often have ancient writings, scrolls, and objects that tell us about the past firsthand.
- **Private Collectors:** Some people collect historical items like old photos, letters, diaries, or even things found during digs. These collections can be very valuable, but access might be limited.
- **Visitors:** People who visit the institutions to learn about Ethiopian history through the artifacts and documents on display.

2.2.4 Problems of the Current System

- **Scattered Resources**: Valuable historical items like documents, objects from the past, and written accounts are spread out across many different places. These places include museums, libraries, churches, monasteries, and even people's private collections. Because everything is spread out, it's difficult for the public to find what they're looking for and learn about Ethiopian history.
- **Limited Access:** Not everyone has easy access to the places where historical knowledge is kept. These places, like museums with artifacts and archives with important documents, are not be close by, especially for people in rural areas. This creates a gap in information, where some

communities don't have the chance to learn about their own history. There's also a problem if the historical information isn't available online.

- **Limited Formats:** The way history is presented now might not be interesting or easy to understand for everyone. While stories passed down by word of mouth are valuable, they may not reach a wide audience. Also, if the information isn't presented in digital formats or online resources, it's harder for people today to learn about it.
- **Preservation Concerns:** Physical objects from the past can be damaged over time, especially if they aren't stored properly or if the environment isn't right for them. If these objects aren't also saved in digital formats, there's a chance they could be lost forever. Because there isn't one central place to take care of everything, it's harder to make sure Ethiopia's history is preserved for future generations.

The limitations identified in the current system create significant challenges in fulfilling user needs. Scattered resources, limited access, and outdated formats all hinder users' ability to explore and learn about Ethiopian history.

2.2.5 Practices to be preserved from the current system

While the current system presents challenges, there are valuable aspects we can incorporate into our platform to enhance the user experience and effectively deliver Ethiopian history:

- **Intuitive Organization:** Existing library catalogs, like the well-regarded system employed by the National Archives of Ethiopia, provide valuable methods for organizing historical information. By leveraging these methods, we can design a user-friendly platform that facilitates efficient discovery and exploration of Ethiopian history.
- **Tiered Access for Researchers:** Adapt existing practices for managing sensitive documents. Implement a user system with varying access levels. Public users see general information, while verified researchers access a designated section for sensitive materials. This balances transparency with information protection.
- **Documentation:** Maintaining proper documentation of physical artifacts is crucial even when the focus shifts to digital formats. This ensures a clear record for future reference and research, and could include detailed descriptions, photographs, and historical context for each artifact.

- **Collaboration:** Partnering with existing institutions like museums, libraries, and universities can provide the EHPTS with access to their expertise, historical resources, and potentially unique content to share on the platform.

2.3 Proposed system

2.3.1 Overview

The EHPTS represents a significant advancement in making Ethiopia's rich history accessible and engaging for a broad audience. This user-friendly platform caters to Ethiopian people, and anyone with a curiosity about Ethiopia's history.

The EHPTS empowers various participants within the system. Super Admins, responsible for the overall well-being of the platform, will establish a transparent revenue-sharing model. This innovative model ensures fair compensation for institutions that contribute valuable historical content. The primary source of funding will come from subscription fees paid by premium users. This approach not only supports content providers but also guarantees the long-term sustainability of the platform.

Institution play a crucial role by uploading valuable historical materials. They will categorize the content for easy searching, manage access for researchers, and moderate comments left by users. The system caters to all user needs through a comprehensive search function. Public users will have access to a vast collection of free resources, while premium subscribers will enjoy an expanded content library and advanced features. Researchers, following an approval process, can access highly sensitive documents for in-depth study.

The EHPTS encourages user engagement through features such as personalized profiles, feedback mechanisms, and dedicated comment sections for uploaded content. These functionalities encourage interaction and community building around Ethiopia's rich historical tapestry.

In essence, the EHPTS aims to revolutionize access to Ethiopian historical resources. It offers a comprehensive and user-friendly platform for exploration, learning, and encouraging a sense of community around Ethiopia's past. Furthermore, the unique revenue-sharing model ensures the platform's long-term viability while supporting the preservation efforts of institutions contributing historical content.

2.3.1 Functional Requirement

All Users:

- **FR-101: User Registration:** The system allows users to register for accounts.
- **FR-102: User Login and Profile Management:** Registered users can log in and manage their profiles.
- **FR-103: Free Content Access:** users can access free content available on the platform.
- **FR-104: Premium Content Access:** Premium users can access exclusive content unavailable to public users.
- **FR-105: Researcher Access:** Researchers can access highly sensitive documents with restricted previews, subject to approval and research proposal acceptance.
- **FR-106: Content Search:** All users can search for content by Title, Historical Figures and media type.
- **FR-107: Premium Content Information and Subscription:** The system displays information about premium content and offers options for users to subscribe for full access.
- **FR-108: Content Access Indicators:** The system displays clear visual cues or labels to indicate different access levels for content (public, premium, and researcher).
- **FR-109: Researcher Document Access Request:** Researchers can submit applications for access to sensitive documents.
- **FR-110: User Feedback:** Users can provide feedback on platform functionalities to improve user experience.
- **FR-111: User Commenting:** Users can comment on publicly available content, Encouraging interaction and discussion.

Institutional Admin and Users:

- **FR-201: Institutional Account Request:** The system allows Institutional Admins to represent their institution and submit a request to join the EHPTS platform through a dedicated online form.
- **FR-202: Staff Account Creation:** Institutional Admins can create accounts for staff users involved in content management activities within their institution.
- **FR-203: User Role Assignment:** Institutional Admins can assign specific roles (Uploader and Reviewer) to each new user account based on their expertise and responsibilities.
- **FR-204: Content Upload and Editing:** Uploaders can upload various historical content formats (and perform basic editing to ensure clarity, proper formatting, and Following to platform style guidelines.
- **FR-205: Institutional Profile Management:** Institutional Admins can edit and update the profile of the institution ensuring accurate information.
- **FR-206: Researcher Access Review:** Reviewers can review researcher applications and grant access to sensitive documents based on defined criteria.
- **FR-207: Comment Moderation:** Institutional Admins can moderate inappropriate comments left on content uploaded by their institution, maintaining a respectful and informative platform.
- **FR-208: Content Deactivation:** Institutional Admins can Deactivation content uploaded by their institution, following to any platform-specific content retention policies.

Platform Admin:

- **FR-302: User Account Management:** Platform Admins can manage user accounts for all user types.
- **FR-303: Subscription Management:** Platform Admins can manage subscription plans for premium content, defining access levels and pricing structures.
- **FR-304: User Feedback Response:** Platform Admins can respond to user feedback, demonstrating responsiveness to user needs and fostering a positive user experience.

2.3.1 Non-Functional Requirements

- **NFR-401: Security & Privacy:** User logins, data, and content are protected using robust security measures.
- **NFR-402: Performance & Scalability:** The EHPTS delivers quick search results, adapts to user growth, and aims for 24/7 uptime for uninterrupted access.
- **NFR-403: Maintainability:** The platform is built for future updates and long-term use, ensuring its continued relevance and functionality.
- **NFR-404: Multi-device Compatibility:** The EHPTS is accessible from a variety of devices, including phones, computers, and tablets.

2.4 Use case modeling

2.4.1 Usecase diagram

A use case describes how a user uses a system to accomplish a particular goal. A use case diagram consists of the system, the related use cases and actors and relates these to each other to visualize: what is being described? (**System**), who is using the system? (**Actors**) and what do the actors want to achieve? (**Use cases**), thus, use cases help ensure that the correct system is developed by capturing the requirements from the user's point of view [21].

Dividing use cases in package [22]

Complex software systems often lead to messy use case models. Use case packaging tackles this by grouping related functionalities, promoting:

- **Clarity:** Easier to understand and navigate broken-down models.
- **Efficiency:** Enables parallel development and simplifies maintenance.
- **Communication:** Tailored presentations for stakeholders based on roles.

So we divide our use cases diagram in to 5 use case packages as follow:

2.4.1.1 User Management Package

This use case package outlines the functionalities for users interacting with the Ethiopian History Preservation and Transmission System (EHTPS) platform. It details the interaction between users and the system regarding registration, login, profile management, and also covers the platform administrator's role in managing user accounts.

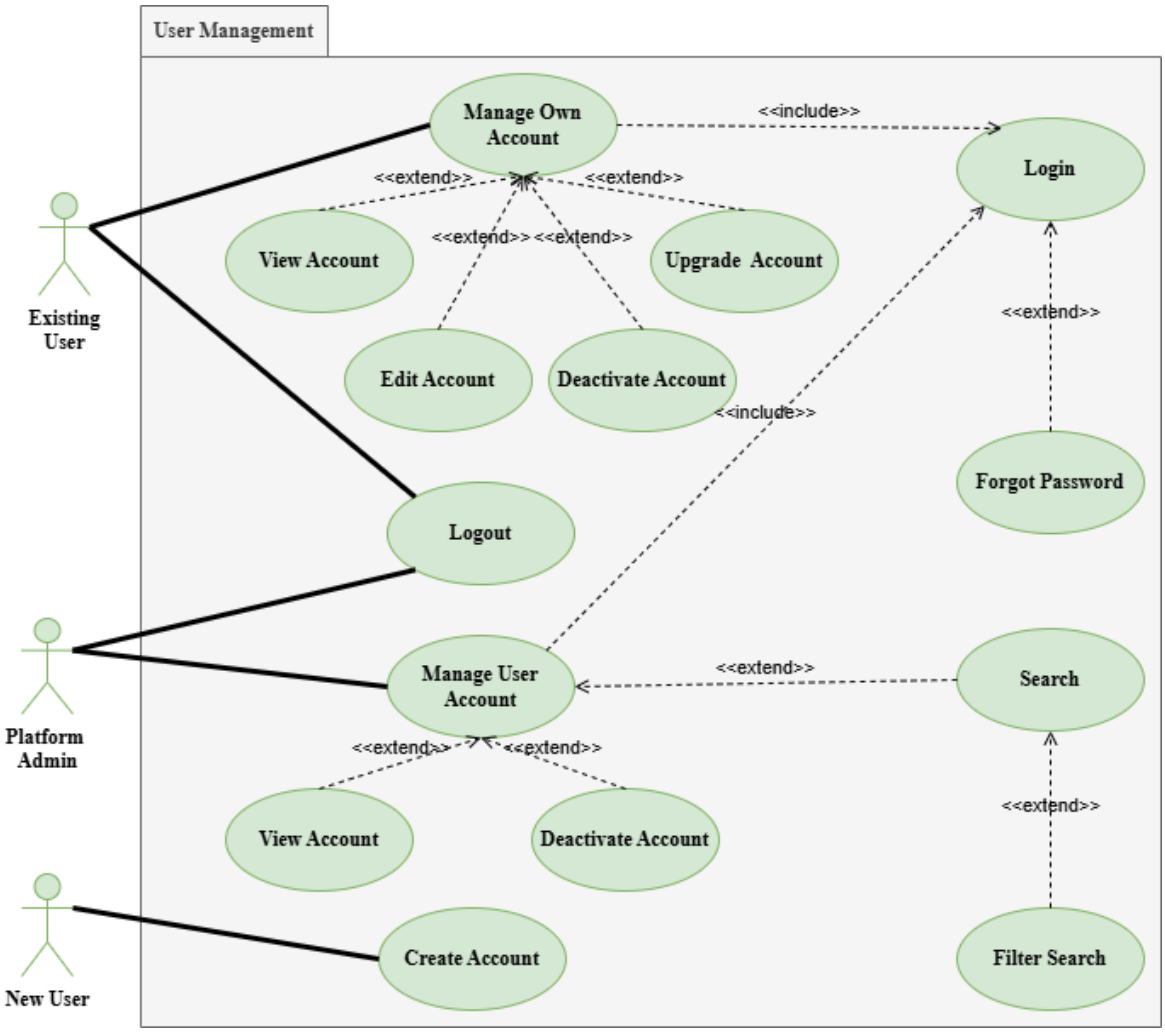


Figure 2.4 User management Package.

2.4.1.2 Institutional Account Management

This use case package depicts the interaction between Institutional Admins and the EHTPS platform for managing users within their institutions. It illustrates the functionalities involved in registering institutions, creating and managing staff accounts, and assigning user roles

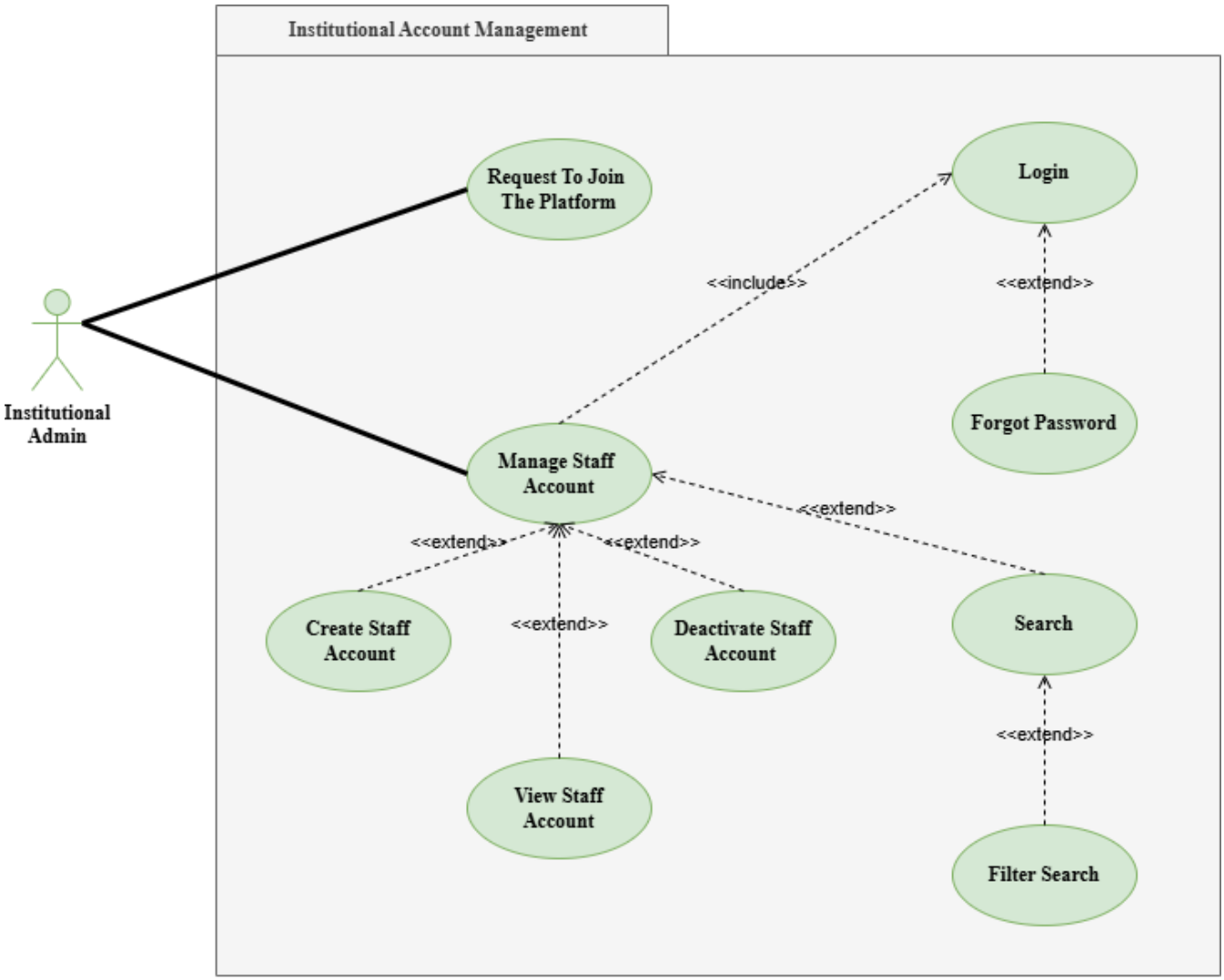


Figure 2.5 Institutional Account Management

2.4.1.3 Content Management

This use case diagram outlines the functionalities for managing historical content on the EHTPS platform. It focuses on the interaction between authorized users (institutional administrators or designated uploaders) and the system regarding uploading, editing, categorizing, and deleting historical content.

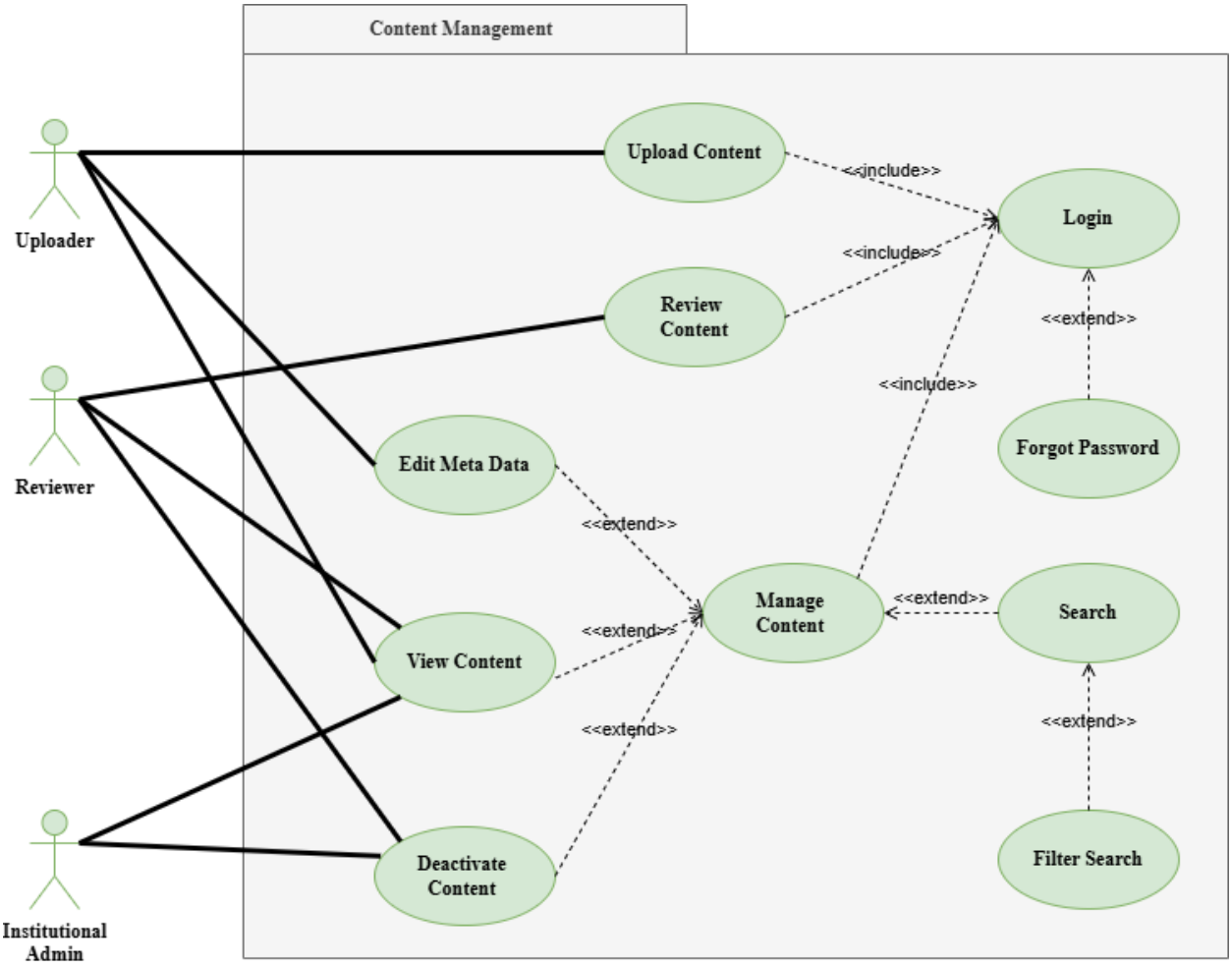


Figure 2.6 Content Management

2.4.1.4 User Access and Permission

This use case package outlines how different user types interact with the EHTPS platform to access historical content. It details the functionalities and access levels for public users, premium users, and researchers.

Figure 2.7 User Access and Permission

2.4.1.5 Feedback and Administration

This use case package outlines how users interact with the EHTPS platform to provide feedback, comments, and reports on content, while also enabling moderators to manage and address these interactions.

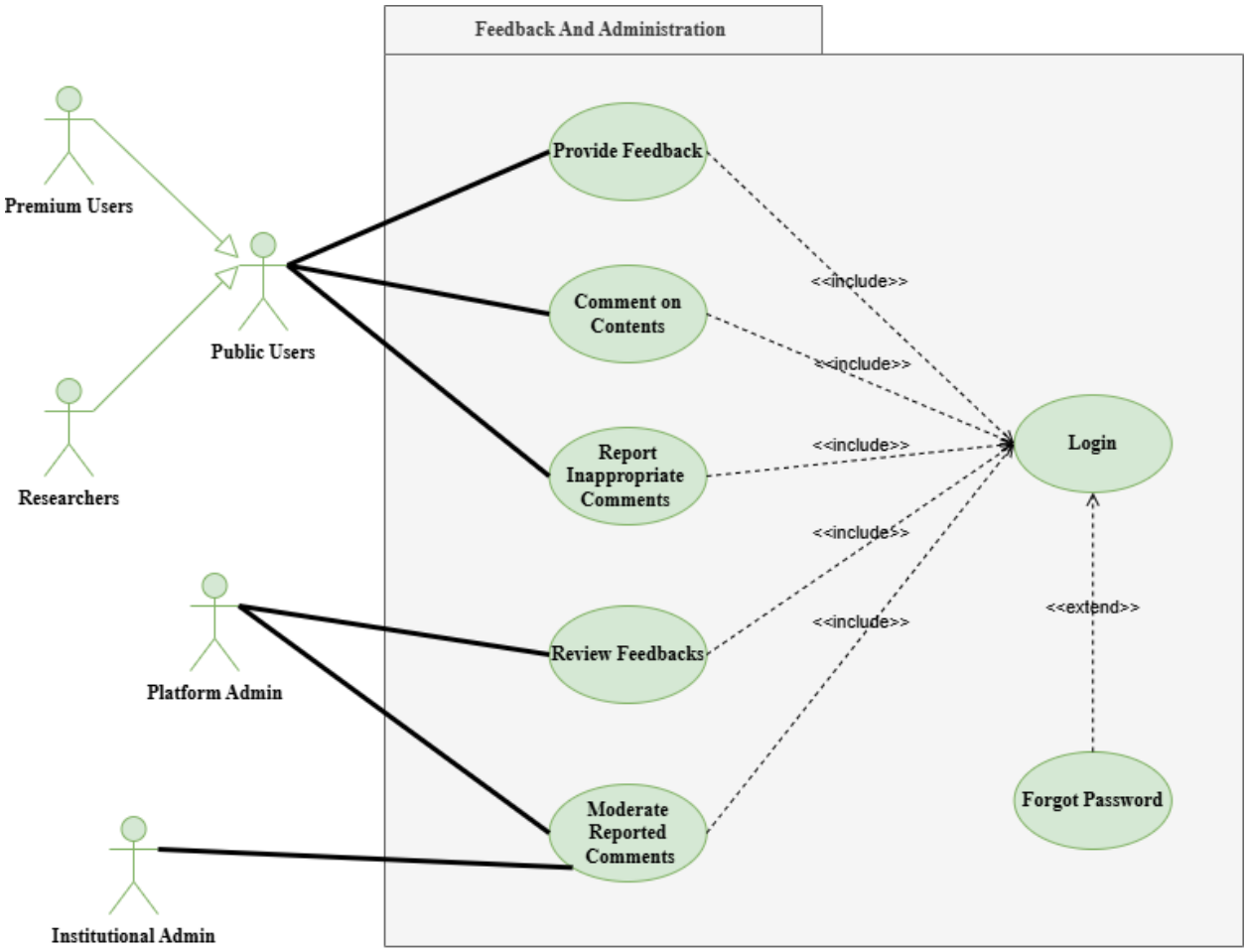


Figure 2.8 Role and Permission

2.4.2 Use case Description Table

Table 2:1 Create Account Use Case

Use Case Name	Create Account
Use Case Id	UC-1
Actor	New user
Description	This use case allows a new user to register and create a new account on the platform.
Precondition	• The user does not have an existing account on the platform. • The user wants to register for the first time.

Basic course of action	1. The user navigates to the designated "Sign Up" page.
	2. The system presents a registration form requiring the user to enter necessary information:
	  • First Name
	  • Last Name
	  • Username
	  • Email
	  • Date of Birth (DOB)
	  • Gender
	  • Password
	  • Confirm Password
	  • Agreement to terms and conditions.
	3. The user fills out the registration form and submits it.
	4. The system validates the submitted information:
	  • Checks for required fields.
	  • Validates the format of the email and date of birth.

	• Checks the username and password length. • Validates the selected gender. • Ensures the user agrees to the terms and conditions. 5. If all information is valid, the system displays a congratulatory message informing the user that their free account is created. The system also offers the option to select Preference 6.1. **Extension point:** Select Preference 6.2. **Condition:** User selected "Select Preference". 6. The use case concludes.
Alternative course of action	A3. If the user enters invalid username and/or password: A4. The system displays a clear error message indicating the attempt was unsuccessful. A5. The use case resumes at step 4, allowing the user to correct their input.
Post Condition	The user has a new account on the platform and can log in to access its features.

Table 2:2 Login Use Case

Use Case Name	Login
Use Case Id	UC-2
Actors	Public users, Premium users, Researchers, platform admin, Institutional admin, uploader, Reviewer.
Description	This use case describes the process of users verifying their identity and gaining access to the platform. The system supports functionality to handle login submissions, including error handling for incorrect username or password entries.

Precondition	• The user has a valid account on the platform. • The user has access to the login page.
Basic course of action	1. The user opens the platform and navigates to the login page. 2. The user enters their username and password in the designated fields. 2.1. **Extension point:** Forgot Password 2.2. **Condition:** User selected “forgot password”. 3. The user clicks the "Sign In" button. 4. The system validates the entered credentials against the user database. 5. The user is redirected to the appropriate dashboard per their role. 6. The use case concludes.
Alternate course of action	A4. The user enters an incorrect username and/or password. A5. The system: • Displays an error message indicating the login attempt was unsuccessful. • The use case resumes at step 2, allowing the user to re-enter their credentials. A6. The user re-enters their credentials and clicks the "Sign In" button again.
Post Condition	A valid user session is established, enabling the user to interact with the platform's features according to their assigned role and permissions.

Table 2:3 Logout Use Case

Use Case Name	Logout
Use Case Id	UC-3
Actors	Public users, Premium users, Researchers, platform admin, Institutional admin, uploader, Reviewer.
Description	This use case describes the process for a user to log out of the platform, terminating their current session.
Precondition	The user is currently logged in to the platform.
Basic course of action	1. The user initiates the logout process by Clicking a dedicated "Logout" button typically located in the user account menu or navbar. 2. The system prompts the user with a confirmation message (e.g., "Are you sure you want to log out?"). 3. If the user confirms the logout by selecting "Confirm": The system invalidates and terminates the user's current session. 4. The use case concludes.
Alternative course of action	A3.The user choose “Cancel” after the confirmation prompt, allowing them to continue their session without interruption.
Post Condition	• If the user confirmed the logout, the user's session is successfully terminated, preventing further access to the platform using that session. • If the user canceled the logout, they remain logged into the platform.

Table 2:4 Forgot Password Use Case

Use Case Name	Forgot Password
Use Case Id	UC-4
Actors	Public users, Premium users, Researchers, platform admin, Institutional admin, Content uploader, Reviewer.
Description	This use case details the process by which a registered user retrieves their forgotten password and regains access to their account. The user initiates a password reset request by providing their registered email address.
Precondition	• The user has forgotten their password and cannot log in. • The user has access to the "Forgot Password" link on the login page.
Extends	UC2: Login
Basic course of action	1. The user clicks on the "Forgot Password" link on the login page. 2. The system presents a form prompting the user to enter their registered email address. 3. The user enters their registered email address. 4. The system validates the email format. 5. The system generates a unique password reset token. 6. The system sends an email to the user's registered email address containing a password reset link. 7. The user receives the email and clicks on the password reset link. 8. The user enters a new password twice for confirmation. 9. The system validates the new password to ensure it meets complexity requirements. 10. The system securely stores the new password and updates the user's account. 11. The system displays a success message informing the user that their

	password has been reset and they can now log in with the new password. 12. The use case concludes.
Alternative course of action	A4. The user inputs an email that is not registered in the system. A5. The system informs the user that the email is not found, allowing them to try again. A6. The user is prompted to enter their email again (back to step 3).
Post Condition	The user has reset their password and can log in to their account with the new password.

Table 2:5 Manage Own Account Use Case

Use case Name	Manage Own Account
Use Case Id	UC-5
Actors	Public users, Premium users, Researchers, platform admin, Institutional admin, uploader, Reviewer.
Description	This use case details the functionalities available to registered users for managing their accounts on the platform. These functionalities include editing account information, deleting accounts, and upgrading to premium plans.
Precondition	The user has a registered account on the platform.
Includes	UC-1: Login (This ensures the user has a valid account before managing)
Basic course of action	1. The User logs in to their EHPTS account via UC-1: Login. 2. The user navigates to the "Account Management" section of the platform.

	3. The system displays the user's account based on their user type. 4. The user selects the desired management action from available options: **4.1 View Account:** 4.1.1. **Extension point:** View Account. 4.1.2. **Condition:** The user selects the "View Account" option. **4.2 Edit Account:** 4.1.3. **Extension point:** Edit Account. 4.1.4. **Condition:** The user selects the "Edit Account" option. **4.3 Deactivate Account:** 4.2.1. **Extension point:** Delete Account. 4.2.2. **Condition:** The user selects the “Deactivate Account” option. **4.4 Upgrade Account** 4.3.1. **Extension point:** Upgrade Account. 4.3.2. **Condition:** The user selects the “Upgrade Account” option. 5. The use case concludes.
Post Condition	The user's account information is updated (upon a successful editing in UC-5.1) Deactivation (upon confirmation in UC-5.2), or upgraded (upon successful payment in UC-5.3) depending on the chosen management action.

Table 2:6 View Account Use Case

Use Case Name	View Account
Use Case Id	UC-5.1
Actors	Public users, Premium users, Researchers.
Description	This use case details the process for a registered user to view their account information on the platform. Users can see a read-only version of their profile details.
Precondition	The user has navigated to the "Account Management" section of the platform.
Includes	UC-1: Login (This ensures the user has a valid account before editing)
Extends	UC-5: Manage User Account
Basic course of action	1. The User logs in to their EHPTS account via UC-1: Login. 2. The user selects the "View Account" option from the menu. 3. The system displays the user's current account information, 4. The use case concludes.
Post Condition	• The user can view all relevant account information.

Table 2:7 Edit Account Use Case

Use Case Name	Edit Account
Use Case Id	UC-5.1
Actors	Public users, Premium users, Researchers.
Description	This use case details the process for a registered user to edit their account information on the platform.
Precondition	The user has navigated to the "Account Management" section of the platform.
Includes	UC-1: Login (This ensures the user has a valid account before editing)
Extends	UC-5: Manage User Account
Basic course of action	5. The User logs in to their EHPTS account via UC-1: Login. 6. The user clicked the "Edit" Button. 7. The system presents editable fields for the user's information. 8. The user modifies the desired information within the designated fields. 9. The user confirms the changes by clicking a "Save" button. 10. The system validates the updated information. 11. The system updates the user's account information in the database. 12. The use case concludes.
Alternative course of action	A5. User decides to cancel the edits by clicking “Cancel” button. A6.The system closes the Edit Account view and discards any unsaved changes. B6. The validation is unsuccessful. B7. The system displays an error message indicating the issue. B8. The user can correct the information and retry saving the changes (user returns to step 4).
Post Condition	The user's account information is updated successfully

Table 2:8 Deactivate Account Use Case

Use Case Name	Deactivate Account
Use Case Id	UC-5.2
Actors	Public users, Premium users, Researchers.
Description	This use case outlines the process for a registered user to deactivate their account on the platform.
Precondition	The user has navigated to the "Account Management" section of the platform.
Includes	UC-2: Login (This ensures the user has a valid account before deleting)
Extends	UC-5: Manage User Account
Basic course of action	1. The user logs in to their EHPTS account via UC-1: Login. 2. The user selects the "Deactivate Account" option. 3. The system displays a confirmation message highlighting the consequences of account deactivation. 4. The user clicks a "Confirm Deactivation" button. 5. The system deactivates the user's account while retaining their data for future reactivation as per platform policies. 6. The use case concludes.
Alternative course of action	**A3**. The user chooses to cancel the deactivate process after initiating it. **A4**.The system returns the user to the "Account Management" section. **A5**.The user's account remains active and unchanged.
Post Condition	• The user's account is deactivated and potentially deleted (upon confirmation of deletion in step 3). • The user's account remains active and unchanged (upon cancellation in step 3 or A3).

Table 2:9 Upgrade Account Use Case

Use Case Name	Upgrade Account
Use Case Id	UC-5.3
Actor	Public User
Description	Users upgrade their account to premium for additional features. The upgrade can be initiated from the content page when users attempt to access premium content or during account management.
Precondition	• The user has an existing account on the platform. • The user desires to upgrade their account to a premium level.
Includes	UC-2: Login (This ensures the user has a valid account before editing)
Extends	UC-4: View Content UC-5: Manage User Account

Basic course of action	1. The User logs in to their EHPTS account via UC-1: Login.
	2. The user initiates the upgrade process:
	  • **From the Content Page:** When the user clicks on premium content, the system prompts the option to upgrade.(UC-)
	  • **From the Account Management Section:** The user clicks an "Upgrade" button within the account management area.
	3. The system presents available premium plans along with their associated features, pricing, and payment methods:
	4. The user selects a premium plan by clicking on the corresponding "Upgrade" button as depicted in for their chosen plan.
	5. The system generates a secure checkout URL for the chosen premium plan using Chapa and presents it to the user.
	6. The user is redirected to Chapa's secure payment gateway through the checkout URL.

	7. The user enters their payment information directly on Chapa's secure platform. 8. The user completes the payment process following Chapa's instructions. 9. Upon successful payment, Chapa securely transmits the payment confirmation back to the EHPTS platform. 10. The system displays a confirmation message indicating successful payment and account upgrade. 11. The user's account is automatically upgraded to the chosen premium level, with immediate access to its associated features and benefits. 12. The use case concludes.
Alternative course of action	A6. The user chooses to cancel the upgrade process. A7. The system exits the upgrade process, and the user remains at their current account level.
Post Condition	• The user's account is upgraded to the chosen premium level, granting them access to additional features and benefits. • The user's account remains at its current level if they canceled the upgrade process.

Table 2:10 Search Use Case

Use Case Name	Search
Use Case Id	UC-6
Actors	Public users, Premium users, Researchers, platform admin, Institutional admin, Content uploader, Reviewer.
Description	This use case details the process for registered users to search for historical content within the System.
Precondition	The user has access to the platform.
Basic course of action	1. The user navigates to the designated search bar or search interface within the platform. 2. The system displays a search field where the user can enter their search query. 3. The user enters their search query in the search field. 4.1. **Extension Point:** Filter Search 4.2. **Condition**: The user selects the "Filter Search" option. 4. The user initiates the search by clicking a designated button. 5. The system performs the search based on the user's query. 6. The system displays the search results on a dedicated results page. 7. The user can browse through the search results and select individual items to view them in detail. 8. The use case concludes.
Alternative course Action	A4. The User enters query that is not on the platform or miss typed text. A5. The system displays a massage that shows not on the platform. A6. The user resume their attempt from step 4.
Post Condition	The user has searched for content on the platform and is presented with a list of relevant search results based on their query and filters.

Table 2:11 ***Filter Search Use Case***

Use Case Name	Filter Search
Use Case Id	UC-7
Actor	Public users, Premium users, Researchers, platform admin, Institutional admin, Content uploader, Reviewer.
Description	This use case details the process for registered users to refine their search for historical content within the EHPTS platform using advanced search filters.
Precondition	The user has access to the platform.
Extends	UC-6: Search Content.
Basic course of action	1. The user identifies the "Filter Search" option within the search interface. 2. Clicking on "Filter Search" reveals additional search filter fields. 3. The user selects desired filters from the available options. 4. The system retrieves and displays search results based on the applied filters. 5. The user can browse through the refined search results and select individual items for further exploration. 6. The use case concludes.
Post Condition	The user has successfully utilized advanced search filters to refine their initial search and is presented with a list of relevant historical content based on their specific criteria.

Table 2:12 View Free Content Use Case

Use Case Name	View Free content
Use Case Id	UC-8
Actor	Public users, Premium users, Researchers
Description	This use case details the process for users to access and view free historical content available within the EHPTS platform, including various media formats like photos, audio, video, and PDFs.
Precondition	• The content must be marked as "Free" and accessible to all users.
Includes	UC-2: Login (This ensures the user is logged in to their account)
Basic course of action	1. The User logs in to their EHPTS account via UC-1: Login. 2. The user navigates to the EHPTS website. 3. The platform interface displays a designated section or category for "Free Content". 4. The user can browse the available free content using filtering options or a search bar. **5.** The user selects a specific piece of free content they are interested in viewing. 6. The system displays the detailed view of the selected content, rendering the media based on the type (photo, audio, video, or PDF). 7. The use case concludes.
Alternative course of Action	A3. The user encounters content marked as "Premium" or "Researcher". A4. The system will provide a message that says "This content is exclusive for Premium Users or Researchers." A5. The usecase resumes at step 3.

Post Condition	The user has successfully accessed and viewed free historical content available within the EHPTS platform.

Table 2:13 View Premium Content Use Case

Use Case Name	View Premium Content
Use Case Id	UC-9
Actor	Premium users
Description	This use case details the process for premium users to access and view exclusive historical content available within the EHPTS platform.
Preconditions	The user has a registered account and holds a valid premium subscription for the EHPTS platform.
Includes	**UC-2**: Login (This ensures the user is logged in to their account).
Basic Course of Action	1. The premium user logs in to their EHPTS account via **UC-1: Login**. 2. The platform interface displays a designated section or category for "Premium" Content. 3. The user can browse through the premium content offerings. 4. The user selects a specific piece of premium content they are interested in viewing. 5. The system verifies the user's premium subscription status. 6. Upon successful verification, the system displays the selected premium content in full detail. 7. The use case concludes.
Alternative Course of Action	**A4**: The user's premium subscription has expired. **A5**: The system will prompt them to renew their subscription.
Post Condition	The premium user has successfully viewed and accessed a piece of premium content on the platform.

Table 2:14 View Researchers Preview Use Case

Use Case Name	View Researchers Preview
Use Case Id	UC-10
Actor	Researchers
Description	This use case details the process for approved researchers to access and view sensitive content designated as "Researcher Preview" within the EHPTS platform.
Precondition	• The user has a registered account and is logged in. • The user's researcher profile has been approved for access to sensitive content through “UC- 13: Review researchers’ application” conducted by the EHPTS platform.
Includes	UC-2: Login (This ensures the user is logged in to their account) UC-12: Apply for researcher preview
Basic course of action	**1.** The Researcher logs in to their EHPTS account via **UC-1: Login.** **2.** The Researcher apply for researcher review via **UC-12: Apply for researcher preview** 3. The approved researcher navigates to a content item previously identified as "Researcher” within the platform. 4. The system automatically recognizes the user's approved researcher status based on their profile information and access permissions granted during the review process. 5. Upon successful verification, the system grants the user full access to the "Researcher Preview" content. 6. The researcher can now view the complete "Researcher Preview" content. 7. The use case concludes.
Post Condition	The approved researcher has successfully viewed the researcher preview on the platform.

Table 2:15 Apply For Researcher Preview Use Case

Use Case Name	Apply For Researcher Preview
Use Case Id	UC-12
Actor	Researcher
Description	This use case allows researchers to apply for access to view sensitive researcher content on the EHPTS platform, which requires approval before full access is granted. The application form ensures that all necessary information is collected and validated before submission.
Precondition	• The user has identified content requiring researcher preview access.
Includes	UC-2: Login (This ensures the user has a valid account before searching)
Basic course of action	**1.** The user will login in to their EHPTS account **Via UC-1.** 2. The researcher navigates to a content item that requires researcher preview access. 3. The system displays a clear indication that the content is restricted and requires approval for access. 4. The user is presented with an "Apply for Access" option. 5. The user clicks on the "Apply for Access" option. 6. The system displays a dedicated application form for requesting access to researcher preview content. 7. The user completes the application form, providing all necessary information and justification for access, including: • First Name • Last Name • Email Address • Phone Number (Optional) • Gender

	• Research Topic. • Purpose of Research • Specific Historical Content Requested • Intended Use of the Content • Researcher Type (Institutional or Independent) • Proof of Affiliation • Supporting Documents 8. The user submits their access request application. 9. The system validates the application form for completeness and correctness. 10. If valid, the system displays a confirmation message acknowledging the submitted request. 11. The use case concludes.
Alternative course of action	A9. The user submits an incomplete application form. A10. The system will display an error message indicating which fields are required and prevent submission until complete. A11. The use case resumes at step 7.
Post Condition	The user's access request for viewing sensitive researcher content is submitted successfully, pending approval from the appropriate authority.

Table 2:16 Upload Content Use Case

Use Case Name	Upload Content
Use Case Id	UC-13
Actor	Uploader
Description	This use case details the process for Institution to submit content items to the EHPTS platform. It includes functionalities for specifying content visibility,

	categorization, and potentially associating the upload with an institution.
Precondition	• The user has a registered Uploader account on the platform. • The user has prepared the content they wish to upload in a supported format.
Includes	UC-2: Login (This ensures the user is logged in to their account)
Basic course of action	**1.** The Uploader logs in to their EHPTS account via **UC-1: Login.** 2. The Uploader navigates to the designated content upload section of the platform. 3. The Uploader selects the type of content they are uploading from a predefined menu. 4. The system displays an upload interface. 5. The Uploader selects the content file from their device using the file selection option. 6. The Uploader fills out a form with relevant metadata about the content. 7. The Uploader selects the desired visibility level for the content from predefined options (public, Premium, researcher). 8. The Uploader confirms and initiates the upload process. 9. The system validates the uploaded content and metadata. 10. Upon successful upload, the system displays a confirmation message. 11. The use case concludes.
Alternative course of action	A9. The user tries to upload content without filling out required metadata fields. A10.The system displays an error message indicating missing information and highlights the required fields. A11. The use case resumes at step 6.

Post Condition	The user has successfully uploaded a content item to the EHPTS platform with the specified metadata, visibility, and category.

Table 2:17 Edit Content Metadata Use Case

Use Case Name	Edit Content Metadata
Use Case Id	UC-14.1
Actor	Institutional Admin
Description	This use case details the functionalities for Institution to modify the metadata associated with their uploaded content items within the EHPTS platform.
Precondition	• The Institutional Admin navigates to the "Content Management" section of the platform. • The Institutional Admin has access to the content item they wish to edit metadata for.
Includes	UC-2:Login
Extends	UC-14:Manage Content
Basic course of action	1. The Institutional admin logs in to their EHPTS account via **UC-1: Login.** 2. The Institutional Admin identifies the specific content item they want to edit. 2.1. **Extension point:** Search 2.2. **Condition:** The user selects “search” option 3. The Institution selects the "Edit" option. 4. The system presents editable fields for the Content information. 5. The system displays a form pre-populated with the existing metadata for the content item.

	6. The Institution makes the necessary changes to the metadata fields. 7. Once modifications are complete, the Institution confirms the changes by submitting the edited metadata form. 8. The system validates and saves the edited metadata for the content item. 9. The system updates the user's content metadata in the database. 10. The system displays a confirmation message informing the user that their content metadata have been successfully updated. 11. The use case concludes.
Post Condition	The content metadata is updated successfully

Table 2:18 Deactivate Content Use Case

Use Case Name	Deactivate Content
Use Case Id	UC-14.2
Actor	Institutional Admin
Description	This use case details the functionalities for Institutions to deactivate their uploaded content items on the EHPTS platform. Deactivating content can be necessary when it is outdated, incorrect, or no longer relevant to the platform's goals.
Precondition	• The Institutional Admin has access to the content item they wish to delete. • The Institutional Admin navigates to the "Content Management" section of the platform.
Includes	UC-2:Login
Extends	UC-14:Manage Content

Basic course of action	1. The Institutional admin logs in to their EHPTS account via **UC-1: Login.** 2. The Institutional Admin identifies the specific content item they want to delete. 2.1. **Extension point:** Search 2.2. **Condition:** The user selects “search” option. 3. The Institutional Admin selects an option to deactivate the content item. 4. The system displays a confirmation message highlighting the consequences of Content deactivate. 5. The Institutional Admin clicks a "Confirm" button. 6. The system potentially deactivate the content and associated data as per platform policies (consider data privacy regulations). 7. The system displays a confirmation message indicating that the content item has been successfully deactivate from the platform. 8. The use case concludes.
Alternative course of action	**A5**. The Institutional Admin chooses to cancel the deactivate process after initiating it. **A6**.The system returns the user to the "Content Management" section. **A6**.The Content remains unchanged.
Post Condition	• The Institution has successfully deactivate their uploaded content item from the EHPTS platform. The content is no longer accessible to users, and relevant records might be archived for internal purposes following platform policies. • The user's Content unchanged.

Table 2:19 Provide Feedback Use Case

Use Case Name	Provide Feedback
Use Case Id	UC-15
Actor	Public users, Premium users, Researchers
Description	This use case allows users to provide feedback on the platform's content, functionalities, or overall user experience, including the ability to rate their experience out of five stars.
Precondition	The user is a registered user and has logged into their EHPTS account via UC-1: Login.
Includes	UC-1: Login.
Basic course of action	1. The user logs in to their EHPTS account through UC-1: Login. 2. The user navigates to the designated section for providing feedback on the platform. 3. The system presents the user with options for submitting feedback, including a rating system and a text box for comments. 4. The user rates their experience by clicking on the star icons, with visual feedback showing the selected rating. For example: • 1 to 5 stars can be clicked to set the desired rating. 5. The user enters additional comments related to their feedback in a text box labeled "My feedback!!". 6. The user submits their feedback by clicking the "Submit" button. • Upon submission, the system processes the feedback data and may log it for analysis. 7. The user receives a confirmation message indicating their feedback has been successfully submitted, either through an on-page notification or a modal.

	8. The use case concludes.
Post Condition	The user's feedback is successfully submitted and recorded by the platform.

Table 2:20 Comment on Contents Use Case

Use Case Name	Comment On Contents
Use Case Id	UC-16
Actor	Public users, premium users, Researchers
Description	This use case allows registered users (public users, premium users, and researchers) to leave comments on various content types within the EHPTS platform, enhancing user engagement and discussion around the content.
Precondition	• The user is logged in to their account. • The user has accessed a piece of content that allows comments.
Includes	UC-1: Login.
Basic course of action	1. The user logs in to their EHPTS account via **UC-1: Login.** 2. The user navigates to a content item that displays a comments section. 3. The system displays existing comments left by other users on the content (if any). 4. The user composes their comment, providing their thoughts, questions, or insights related to the content. 4.1. **Extension point:** Report Inappropriate Comment. 4.2. **Condition :** The user selects " Report Inappropriate Comment" 5. The new comment is added to the comments section. 6. Use case concluded.

Post Condition	The user's comment is successfully posted and displayed in the comments section for the specific content item.

Table 2:21 Report Inappropriate Comment Use Case

Use Case Name	Report Inappropriate Comment
Use Case Id	UC-17
Actor	Public users, premium users, Researchers
Description	This use case describes the process for registered users to report comments that violate EHPTS platform guidelines or are considered offensive, irrelevant, or harmful.
Precondition	• The user is logged in to their account. • The user encounters a comment on a content item that they believe violates platform guidelines.
Includes	UC-1: Login.
Extends	UC-16: Comment On Contents
Basic course of action	1. The user logs in to their EHPTS account via **UC-1: Login.** 2. The user identifies a comment that they consider offensive, irrelevant, misleading, harassing, or otherwise in violation of platform guidelines. 3. The user locates the reporting mechanism associated with the comment. 4. The user selects a reason for reporting the comment from a predefined list. 5. The user submits the report. 6. The user receives confirmation that their report has been submitted

	successfully. 7. The reported comment remains visible until reviewed by a moderator. 8. Use case concluded.
Post Condition	• The user receives confirmation that their report has been submitted successfully. • The reported comment remains visible until reviewed by a moderator.

Table 2:22 Manage Reported Comments Use Case

Use Case Name	Manage Reported Comments
Use Case Id	UC-18
Actors	Institutional Admin
Description	This use case describes the actions an Institutional Admin can take when handling reported comments on the platform. The admin is able to evaluate reported remarks and respond accordingly by deleting comments, deactivating users, or dismissing reports without any action being taken.
Precondition	• The Institutional Admin is logged into the system. • A user has reported a comment for violating EHPTS platform guidelines (refer to UC-17: Report Inappropriate Comment).
Includes	Uc-1: Login
Basic course of action	1. The user logs in to their EHPTS account via **UC-1: Login.** 2. The Institutional Admin navigates to a dashboard that displays reported comments. 3. The system presents a list of reported comments, including details such as:

	• Comment content • User who reported the comment • Date of the report • Reason for reporting • User details (name, link to profile, and reporting history). 4. The Institutional Admin reviews the comment and the reporting details. 5. The Institutional Admin can choose to: • **Delete Comment**: Permanently remove the comment from the platform. • **Deactivate User:** Remove the comment and disable the user's account, preventing further interaction if deemed necessary. • **Dismiss Report:** Choose to ignore the report and retain both the comment and the user, indicating the report did not warrant action. 6. Upon selecting an action, a confirmation dialog box appears, detailing the action being taken and asking for confirmation (e.g., "Are you sure you want to delete this comment?"). 7. Upon confirmation, the system performs the selected action: • If "Delete Comment" is chosen, the comment is permanently removed. • If "Deactivate User" is selected, the user's account is disabled, and their comment is removed. • If "Dismiss Report" is chosen, the report is dismissed, and no further action is taken on the comment. 8. The system displays a message confirming the success of the action taken (e.g., "Comment successfully deleted," "User account deactivated," or "Report dismissed"). 9. Use case concluded.

Post Condition	• The selected comment may be deleted, a user may be deactivated, or the report may be dismissed. • The system updates to reflect the action taken.

Table 2:23 Request to Join Platform Use Case

Use Case Name	Request to Join Platform
Use Case Id	UC-19
Actor	Institutional Admin
Description	This use case describes the process for an Institutional Admin to submit a request for their institution to join the EHPTS platform, culminating in the creation of their institutional account.
Precondition	• The Institutional Admin does not currently have an account on the EHPTS platform. • The user is on the landing page of the platform.
Basic course of action	1. The user navigates to the landing page of the platform. 2. The user identifies the "Become a Contributor" button on the landing page. 3. Upon clicking the button, the system redirects the user to a page that explains the benefits of contributing to the platform. 4. The user sees a "Get Started" button at the bottom of the benefits explanation page. 5. Clicking the "Get Started" button leads the user to a contribution form. 6. The system presents the user with a form to fill out their contribution details, which may include: • Institution Name

- Brief Description of the Institution Address
- Website (optional)
- Collaboration Purpose
- Contact Phone
- Type (e.g., MUSEUM, CHURCH, LIBRARY, SCHOOL, OTHER)
- Establish Date
- Admin Email
- Admin First Name
- Admin Last Name
- Admin Gender
- Admin Username
- Admin Password
- Institutional Profile (Logo)
- Institutional Legal Document

7. Upon filling out the form, the Institutional Admin clicks the "Submit" button to submit the request.
8. The system captures the input data and triggers the review process by the Platform Admin.
9. The submitted request appears on the Platform Admin's dashboard for review.
10. The Platform Admin reviews the request and makes a decision:
    - If Approved: The Platform Admin sends an approval email to the Institutional Admin.
        - The Institutional Admin is notified that their request has been approved.
    - If Denied: The Platform Admin sends a denial email to the Institutional Admin, specifying the reasons for the denial.
11. Upon receiving an approval notification, the Institutional Admin can now access the platform to upload their institution's content and

	manage relevant information. 12. Use case concluded.
Alternative Course of action	A6.The Institutional Admin tries to submit the request with missing information. A7.The system displays an error message indicating required fields and prompts the user to complete them. A8.The use case resumes at step 6.
Post Condition	• The Institutional Admin has successfully submitted a request for their institution to join the EHPTS platform. • The Institutional Admin receives a notification of approval or denial via email. • Upon approval, the Institutional Admin can log in and upload content to the platform.

Table 2:24 Manage Staff Account Use Case

Use Case Name	Manage Staff Accounts
Use Case Id	UC-21
Actor	Institutional Admin
Description	This use case details the functionalities available to an Institutional Admin for managing staff accounts within their institution on the EHPTS platform. Staff accounts provide access to platform functionalities based on assigned roles and permissions.
Precondition	The Institutional Admin has an active account on the EHPTS platform with permission to manage staff accounts.

Includes	UC-2: Login.
Basic course of action	1. The Institutional Admin logs in to their EHPTS account via **UC-1: Login.** 2. The Institutional Admin navigates to the "Staff Account Management" section of the platform. 3. The Institutional Admin selects the desired management action from available options: 4. **View Staff Account:** 5.1 **Extension Point:** View Staff Account 5.2 **Condition:** the Institutional Admin selects the “View Staff account” option. 5. **Create Staff Account:** 5.1 **Extension point:** Edit Staff Account. 5.2 **Condition:** The Institutional Admin selects the "Edit Staff Account" option. 6. **Deactivate Staff Account:** 6.1. **Extension point:** Deactivate Staff Account. 6.2. **Condition:** The Institutional Admin selects the “Deactivate Staff Account” option. 7. The use case concludes.
Post Condition	• The staff account information is created. • The staff account is deactivated.

Table 2:25 View Staff Account Use Case

Use Case Name	View Staff Account
Use Case Id	UC-21.1
Actors	Institution Admin
Description	This use case details the process for an Institutional Admin to view information associated with a selected staff account on the EHPTS platform. This allows the admin to access and evaluate staff-specific details.
Precondition	• The Institutional Admin has logged into their EHPTS account via UC-1: Login. • The Institutional Admin has selected a specific staff account to view from the staff list.
Includes	UC-2: Login (This ensures the user has a valid account before editing)
Extends	UC-5: Manage Staff Account
Basic course of action	1. The Institutional Admin logs in to their EHPTS account via UC-1: Login. 2. The system displays a list of staff accounts associated with the Institutional Admin's institution. 3. The Institutional Admin selects a specific staff account from the list. 2.1. **Extension point:** Search 2.2. **Condition:** the Institution admin selects “Search” option. 4. The Institutional Admin navigates to the staff list and selects a specific staff account to view by clicking the "Detail" button associated with that account. 5. The system retrieves the details associated with the selected staff account, including: 6. ID

	7. Full Name (First and Last) 8. Email 9. Entering Date 10. Role/Permissions 11. Account Status (Active/Inactive) 12. The system displays the account information in a user-friendly format, potentially using a modal or a separate details page. 13. The Institutional Admin can review the displayed details about the staff account. 14. The Institutional Admin can choose to close the details view or return to the staff list. 15. The use case concludes.
Post Condition	The Institutional Admin has viewed the information associated with the selected staff account.

Table 2:26 Create Staff Account Use Case

Use Case Name	Create Staff Account
Use Case Id	UC-20
Actor	Institutional Admin
Description	This use case details the process for an Institutional Admin to create staff accounts for other users within their institution on the EHPTS platform. Staff accounts provide access to platform functionalities based on assigned roles and permissions.

Precondition	The Institutional Admin has an active account on the EHPTS platform with permission to create staff accounts. The Institutional Admin has gathered relevant information about the staff member.
Includes	UC-2: Login.
Basic course of action	1. The user logs in to their EHPTS account via **UC-1: Login.** 2. The Institutional Admin navigates to the designated section within the platform for managing staff accounts. 3. The Institutional Admin selects the option to create a new staff account. 4. The Institutional Admin fills out a form with the following information about the new staff member and submits the form: • First Name • Last Name • Email Address • Gender (as a dropdown: male, female) • Username • Password • Confirm Password • Desired Role/Permissions (as a dropdown: uploader, reviewer) 5. The system validates the input fields. 6. If all validations are passed, the system creates the staff account. 7. The system displays a confirmation message indicating that the staff account has been created successfully. 8. The Use Case Concludes.
	A4: If the Institutional Admin tries to submit the request with missing information:

	A5: The system displays an error message indicating required fields. A6: The use case resumes at step 4, allowing the admin to correct the input.
Post Condition	• The Institutional Admin has successfully created a staff account for a new user within their institution on the EHPTS platform.

Table 2:27 Deactivate Staff Account Use Case

Use Case Name	Deactivate Staff Account
Use Case Id	UC-21.3
Actors	Institution Admin
Description	This use case details the process for an Institutional Admin to deactivate a selected staff account on the EHPTS platform. This action will revoke the selected staff member's access to the platform.
Precondition	• The Institutional Admin has logged into their EHPTS account via **UC-1: Login.** • The Institutional Admin has selected a specific staff account to deactivate from the staff list.
Includes	UC-2: Login
Extends	UC-5: Manage Staff Account
Basic course of action	1. The Institutional Admin logs in to their EHPTS account via **UC-1: Login.** 2. The Institutional Admin navigates to the staff list and identifies the staff member to be deactivated. 3. The Institutional Admin clicks the "Deactivate" button associated with the selected staff account. This triggers a confirmation step. 4. The Institutional Admin confirms the deactivation action by clicking a designated confirmation button (this include a confirmation dialog).

	5. The system updates the selected staff account to set isActive to false, effectively deactivating the account and revoking the user's access to the platform. 6. The system may display a success message indicating that the staff account has been deactivated. 7. The use case concludes.
Post Condition	The selected staff account is deactivated, and their access to the platform is revoked.

Table 2:28 Review Researcher Application Use Case

Use Case Name	Review Researcher Application
Use Case Id	UC-22
Actors	Reviewer
Description	This Use Case outlines the process for a Reviewer to review and assess a Researcher application submitted to the EHPTS platform using the Researcher Management interface.
Precondition	• The Reviewer has a valid account on the EHPTS platform with reviewer permissions. • A Researcher application has been submitted to the platform and assigned to the Reviewer for evaluation.
Includes	UC-2: Login (This ensures the Reviewer has a valid account before reviewing)
Basic course of action	1. The Reviewer logs in to their EHPTS account via UC-1: Login. 2. The Reviewer navigates to a designated section within the platform to access assigned Researcher applications awaiting review. 3. The Reviewer selects the specific Researcher application assigned to

	them. The system displays the application details. 4. The Reviewer carefully reviews the application materials, evaluating the Researcher's qualifications against established criteria. • Based on the review, the Reviewer selects a recommendation from a predefined set of options: o **Approve**: The Researcher meets all criteria and should be granted access to the EHPTS platform. o **Request Clarification**: Additional information or clarification is needed from the Researcher before a final decision can be made. The Reviewer can specify the required information within the system. o **Reject:** The Researcher does not meet the eligibility requirements or qualifications for platform access. 5. The Reviewer submits their recommendation and any associated comments or justifications within the platform. 6. The Use Case Concludes.
Post Condition	The Reviewer's finalized decision is submitted.

Table 2:29 View Users Feedback

Use Case Name	View User Feedback
Use Case Id	UC-23
Actors	Platform Admin
Description	This use case details how users submit feedback and how the admin can view, filter, and analyze the feedback provided by users on the platform.
Precondition	• Users must have valid accounts to provide feedback.

	• Feedback data is stored in the system for the admin to access.
Includes	UC-2: Login (This ensures the Platform Admin has a valid account-)
Basic course of action	1. The Platform Admin logs in to their EHPTS account via UC-1: Login. 2. The system retrieves all feedback and calculates the average rating. 3. Feedback is displayed as cards containing user details, ratings, and comments. 4. The admin applies filters based on star ratings or searches by username. 5. The system dynamically updates the displayed feedback based on the selected filter or search query. 6. The admin can view complete comments by clicking "Read More" on truncated feedback. 7. The system toggles the display to show the full text or collapse the comment. 8. The Use Case Concludes.
Post Condition	• The admin successfully views and analyzes feedback. • Insights, such as average ratings and total feedback, are provided.

Table 2:30 Manage Users Account Use Case

Use Case Name	Manage Users Accounts
Use Case Id	UC-24
Actor	Platform Admin
Description	This Use Case details the functionalities available to a Platform Admin for managing

	all user accounts on the EHPTS platform.
Precondition	The Platform Admin has an active account on the EHPTS platform with permission to manage all user accounts.
Includes	UC-2: Login.
Basic course of action	8. The Platform Admin logs in to their EHPTS account via **UC-1: Login.** 9. The Platform Admin navigates to the designated section for managing all user accounts on the platform 10. The system displays a comprehensive list of all user accounts on the platform. 11. The Platform Admin selects a specific user account from the list. 2.1. **Extension point:** Search 2.2. **Condition:** the Institution admin selects "Search" option. 12. The Platform Admin selects the desired management action from available options: 13. **View User Account:** 5.3 **Extension Point:** View User Account 5.4 **Condition:** the Platform Admin selects the "View User account" option. 14. **Deactivate User Account:** 6.3. **Extension point:** Deactivate User Account. 6.4. **Condition:** The Platform Admin selects the "Deactivate the User Account" option. 15. The use case concludes.
Post Condition	• The user account is deactivated and their associated data removed.

Table 2:31 View User Account Use Case

Use Case Name	View User Account
Use Case Id	UC-24.1

Actors	Platform Admin
Description	This use case details the process for a Platform Admin to view information associated with a selected staff account on the EHPTS platform. The admin can access a list of users, filter them, and view detailed information upon selection.
Precondition	• The Platform Admin has logged into their EHPTS account via UC-1: Login. • The Platform Admin is on the user list page from which specific accounts can be selected.
Includes	UC-2: Login
Extends	UC-5: Manage User Account
Basic course of action	1. Login: The Platform Admin logs in to their EHPTS account through UC-1: Login. 2. The admin navigates to the user list via the sidebar menu, which includes options like "Institutional Admin", "Public Users", "Premium Users", "Researchers", "Uploaders", "Reviewers ","and Institutions". 3. The Platform Admin clicks the "Detail" button associated with a selected user from the filtered list. 4. The system retrieves and displays details associated with the selected staff account in a modal or detail view. 5. The Platform Admin reviews the user account details in the modal or detail view. 6. The Platform Admin can close the view/modal to return to the user list. 7. End Use Case: The use case concludes.
Post Condition	The Platform Admin has successfully viewed the selected user account.

Table 2:32 Deactivate User Account Use Case

Use Case Name	Deactivate User Account
Use Case Id	UC-24.3
Actors	Platform Admin
Description	This use case details the process for a Platform Admin to deactivate a selected user account on the EHPTS platform. The deactivation of an account will remove the user's access to the platform and possibly associated data.
Precondition	• The Platform Admin has logged into their EHPTS account via UC-1: Login. • The Platform Admin has selected a specific user account to deactivate from the user list.
Includes	UC-2: Login (This ensures the user has a valid account before editing)
Extends	UC-5: Manage user Account
Basic course of action	1. The Platform Admin logs in to their EHPTS account through UC-1: Login. 2. The Platform Admin navigates to the user list and selects the "Deactivate User Account" option next to the specific staff member they wish to deactivate. 3. The system displays a confirmation message clearly stating that deactivating the user account will remove their access to the platform and may result in data loss associated with that account. 4. The Platform Admin confirms the deactivation by clicking a designated button (e.g., "Confirm").The Platform Admin confirms the deletion by clicking a designated button. 5. The system deactivate the user account from the platform's user

	database, updating the relevant statuses. 6. The use case concludes.
Post Condition	• The selected user account is deactivated, and the user's access to the EHPTS platform is revoked. • The deactivation is recorded in the system, ensuring the account is no longer active.

2.5 Sequence diagram

UML sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system [23].

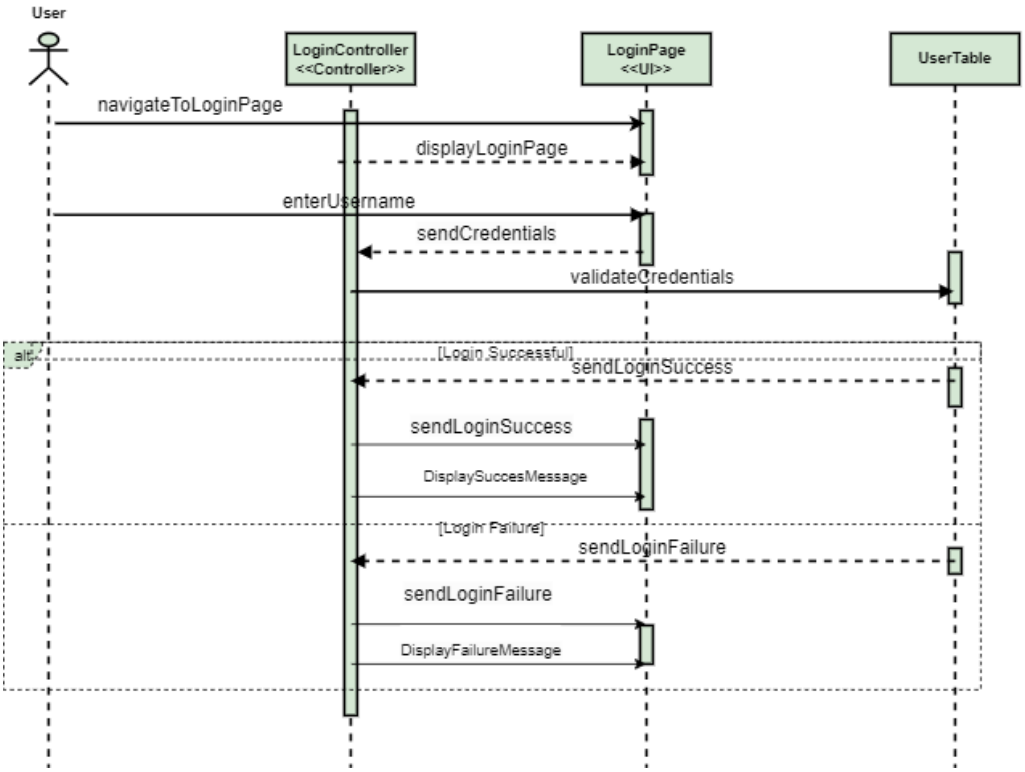


Figure 2.9 Login Sequence diagram

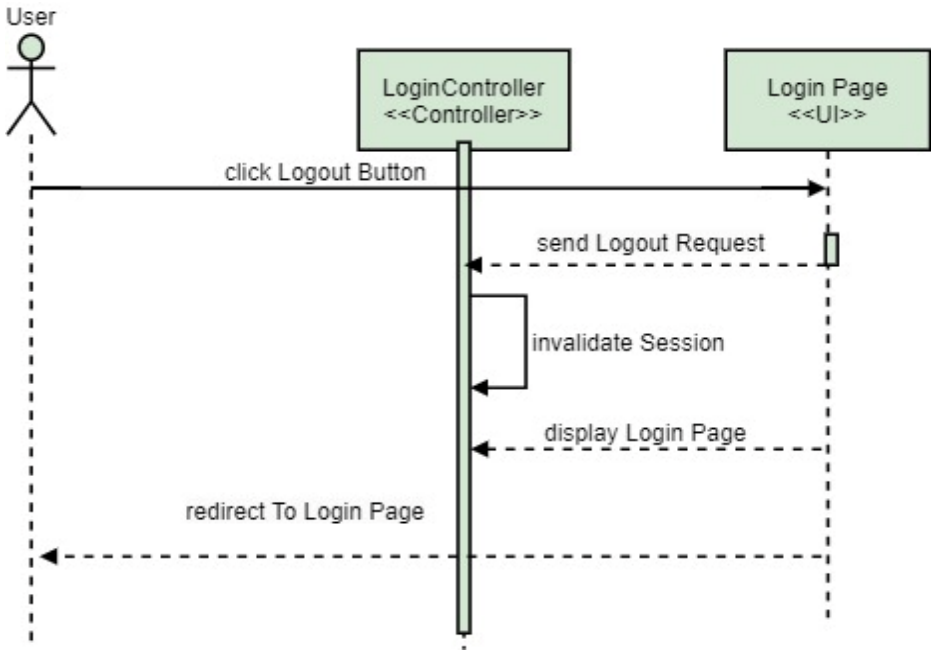


Figure 2.10 logout sequence diagram

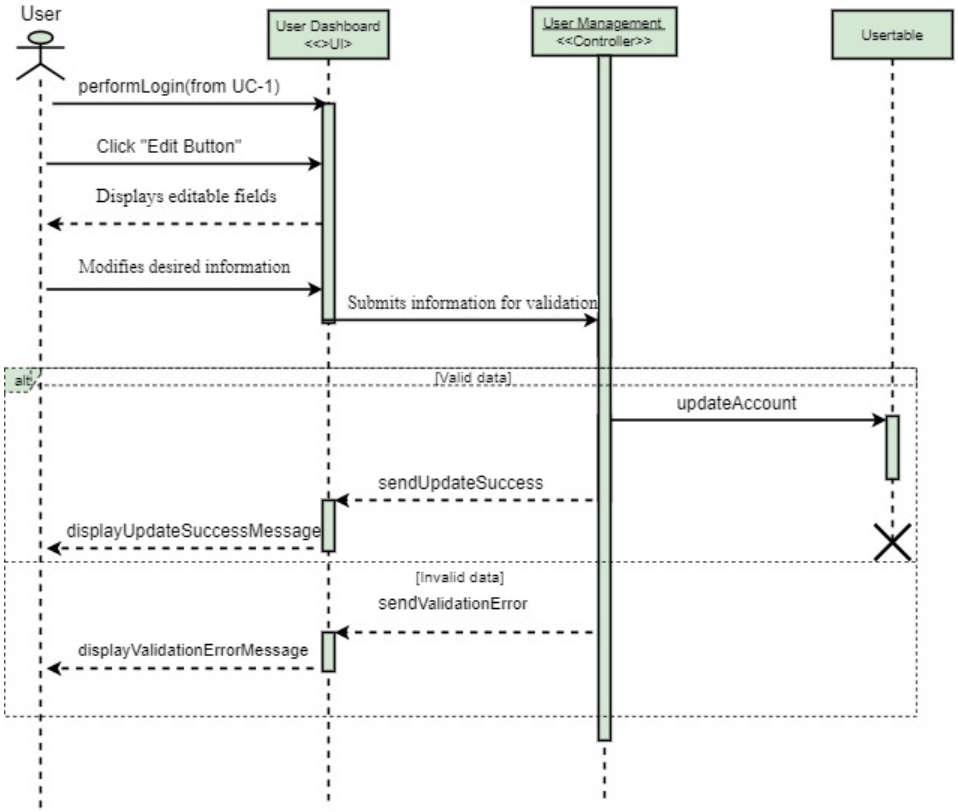


Figure 2.11 Edit Account sequence diagram

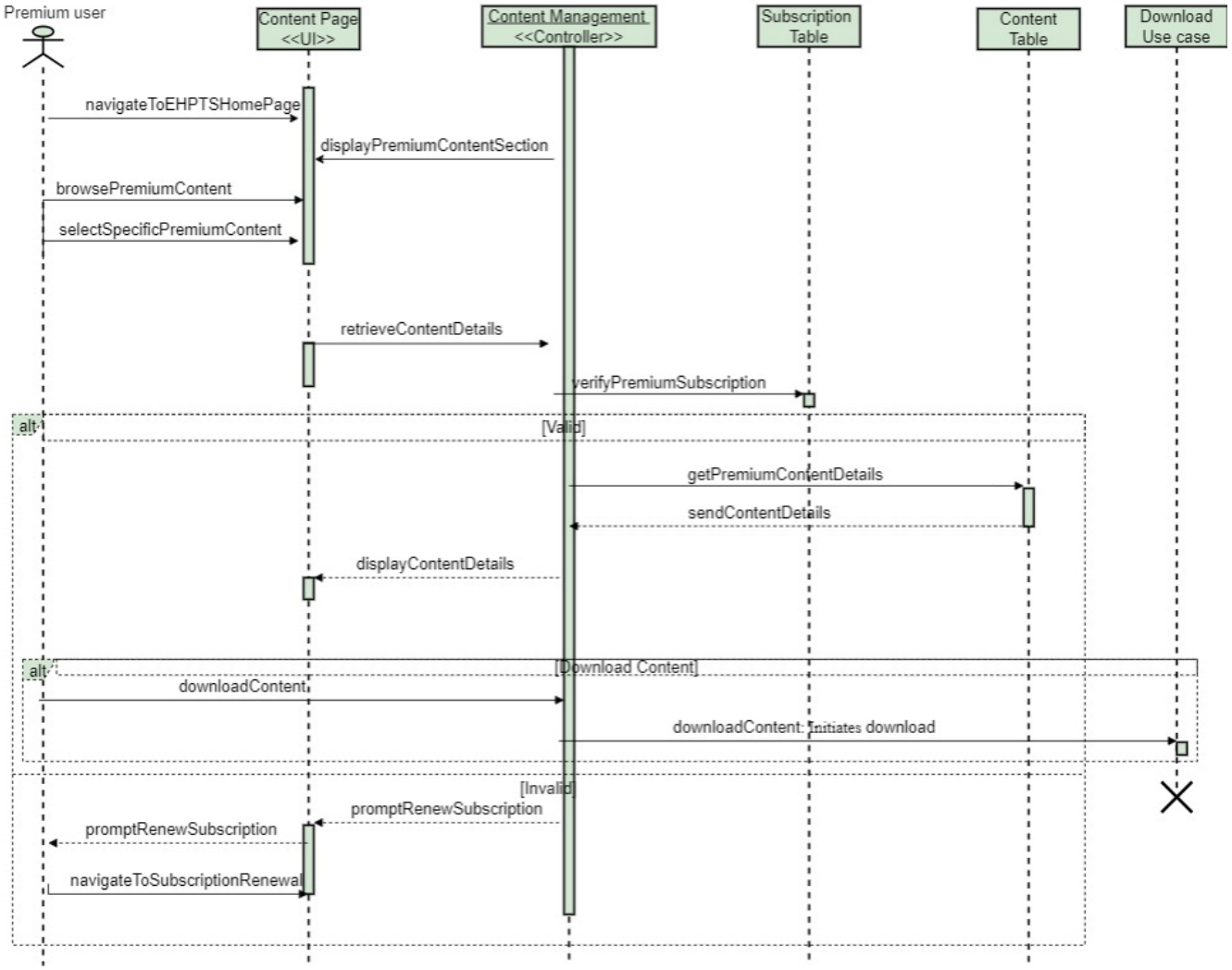


Figure 2.12 View Premium Content sequence diagram

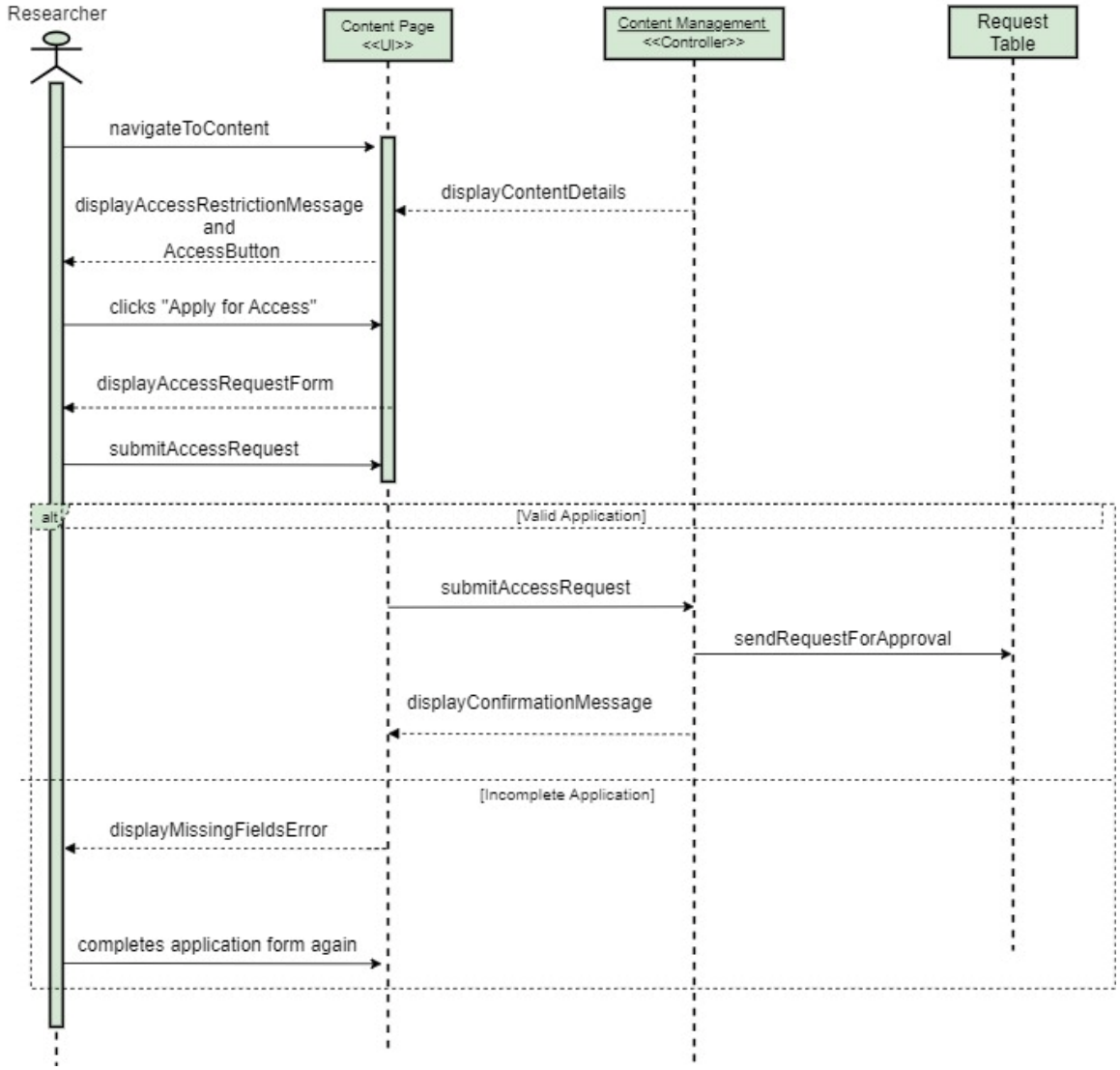


Figure 2.13 Apply Researcher Preview sequence diagram

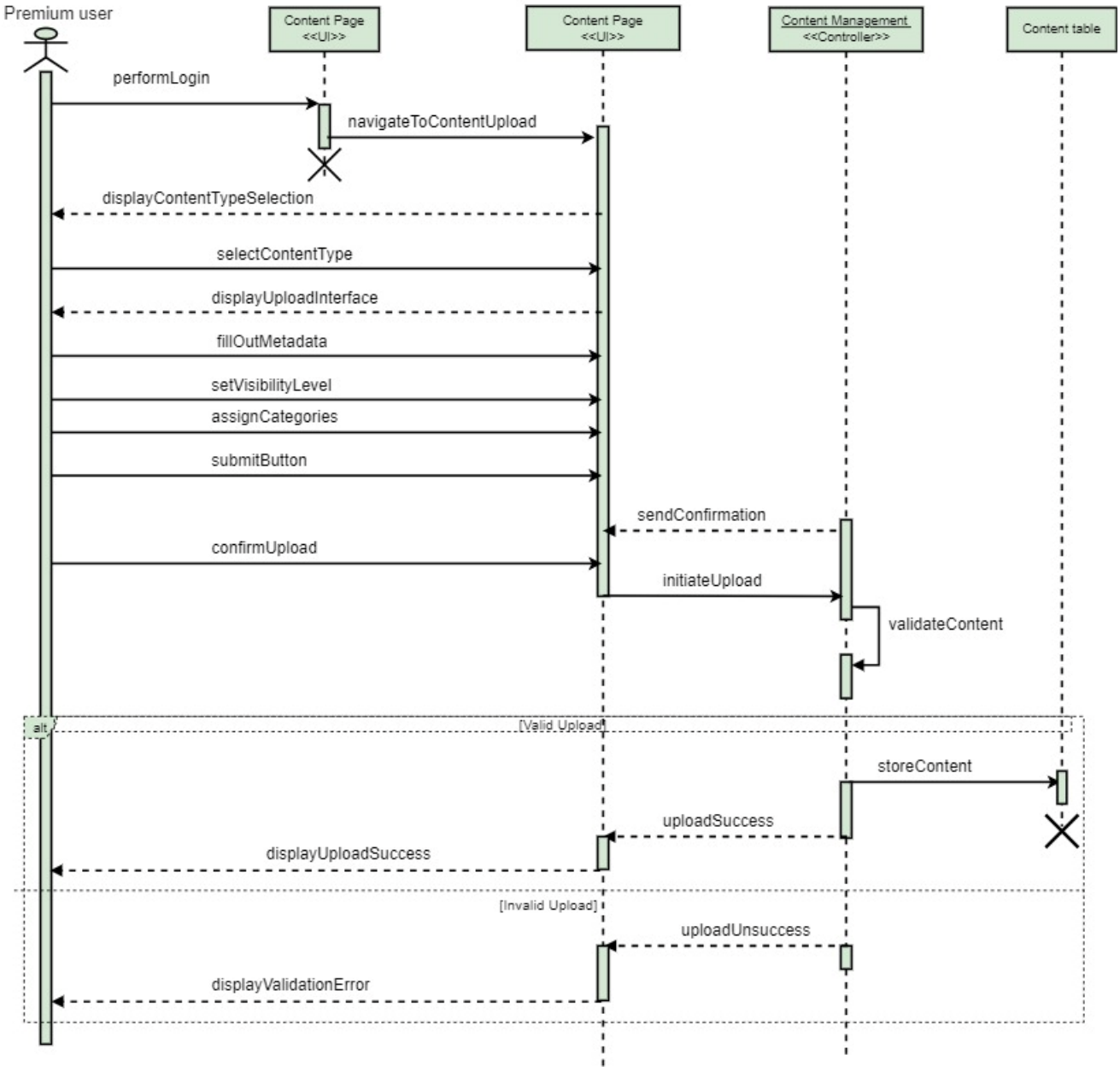


Figure 2.14 Upload Content sequence diagram

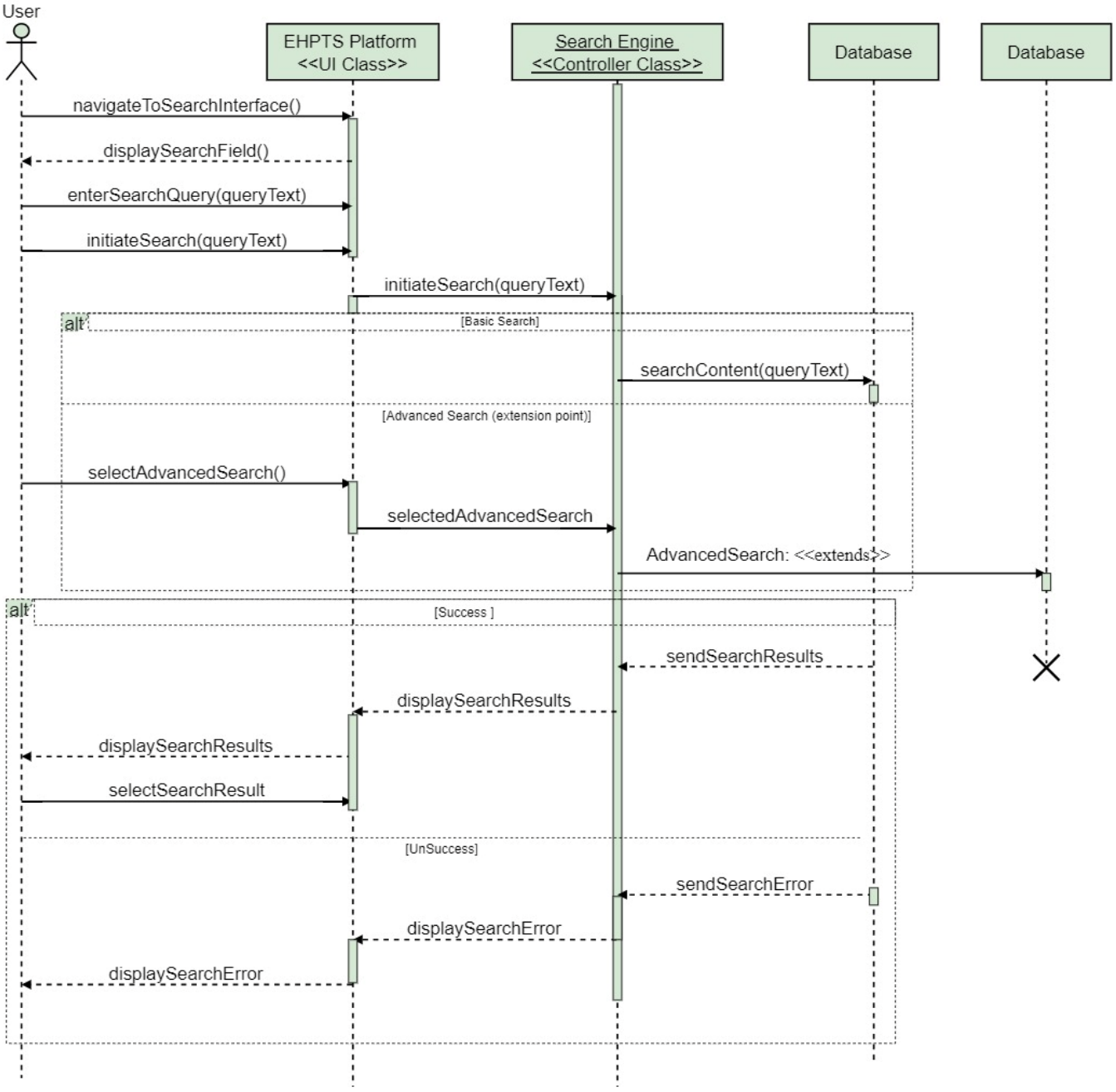


Figure 2.15 Search sequence diagram

Chapter 3

3. System Design

3.1 Overview

Following the identification of requirements and models in the previous chapter, this chapter decomposes the design of the Ethiopian History Preservation and Transmission System (EHPTS). System design focuses on translating those requirements into a technical blueprint that guides the development process. Here, we'll explore several key aspects:

- **Design Goals:** These define the qualities we strive to achieve with the EHPTS. They serve as guiding principles for our architecture and implementation choices.
- **Subsystem Decomposition:** We'll break down the system into smaller, manageable components with well-defined functionalities. This promotes modularity and simplifies development and maintenance.
- **Class Diagram:** This visual representation can illustrate the classes within the system, their attributes, and the relationships between them.
- **Persistent Model:** We'll explore how data will be stored and accessed within the system, ensuring its persistence and integrity.
- **Deployment Diagram:** This diagram will depict the physical or virtual distribution of the system's components across different servers or environments.

3.2 Design Goals

The design goals for the EHPTS are like the building blocks for its architecture. We're aiming to create a system that excels in these key areas:

- **Scalability:** As more people use the EHPTS and more historical content is added, the system should be able to handle it all. We can achieve this by independently scaling up specific parts of the system, like adding more storage for all the amazing historical content.

- **Maintainability:** By breaking the system down into smaller, self-contained pieces (micro services), it will be easier for our team to build, test, fix, and improve the EHPTS over time.
- **Performance:** Users should experience a smooth and responsive system, whether they're searching for content, accessing historical information, or uploading their own contributions.
- **Usability:** The user interface should be intuitive and user-friendly, catering to users with varying technical expertise.
- **Security:** Robust authentication, authorization, and data encryption mechanisms are crucial to protect user data and sensitive historical content.
- **Flexibility:** The architecture should allow for independent development and deployment of new features or functionalities, enabling the platform to evolve over time.

3.3 Proposed System Architecture

3.3.1 Overview

The Ethiopian History Preservation and Transmission System (EHPTS) website will use a **client-server** architecture to ensure scalability, maintainability, and a secure user experience. In this approach, the system is divided into two distinct parts:

- **Client-Side (Front-End):** This refers to the user interface that users interact with directly. Our case, it's the EHTPS website accessible through a web browser on any device. The client-side is responsible for displaying historical information, handling user interactions like searches or account creation, and sending requests to the server for data.
- **Server-Side (Back-End):** This acts as the brain of the system, handling data storage, processing, and communication with the client-side. The server will house the database containing digitized historical materials, user accounts, and system functionalities. It will process user requests received from the client-side, retrieve data from the database, and send responses back to the client to update the user interface.

Benefits of a Client-Server Architecture for the EHTPS Website:

- **Scalability:** A client-server architecture allows for independent scaling of the client and server sides. As the user base grows, the server's capacity can be increased to handle more requests without impacting the user interface. This ensures smooth operation even with a significant increase in traffic.
- **Maintainability:** Separating the user interface from the core functionality simplifies maintenance. Updates or bug fixes on the server-side won't require changes to the client-side application (the website), and vice versa. This reduces development time and effort.
- **Security:** Sensitive data like user accounts and historical resources can be securely stored and managed on the server-side. This minimizes the risk of unauthorized access on the client-side (the user's device) as the website itself doesn't hold the sensitive information directly

3.3.2 Subsystem Decomposition/Modularization

Decomposition is the process of dividing the system into smaller subsystem parts. By composing a system, we can present the overall system as a hierarchy and view the relationships between subordinate and higher-level subsystems.

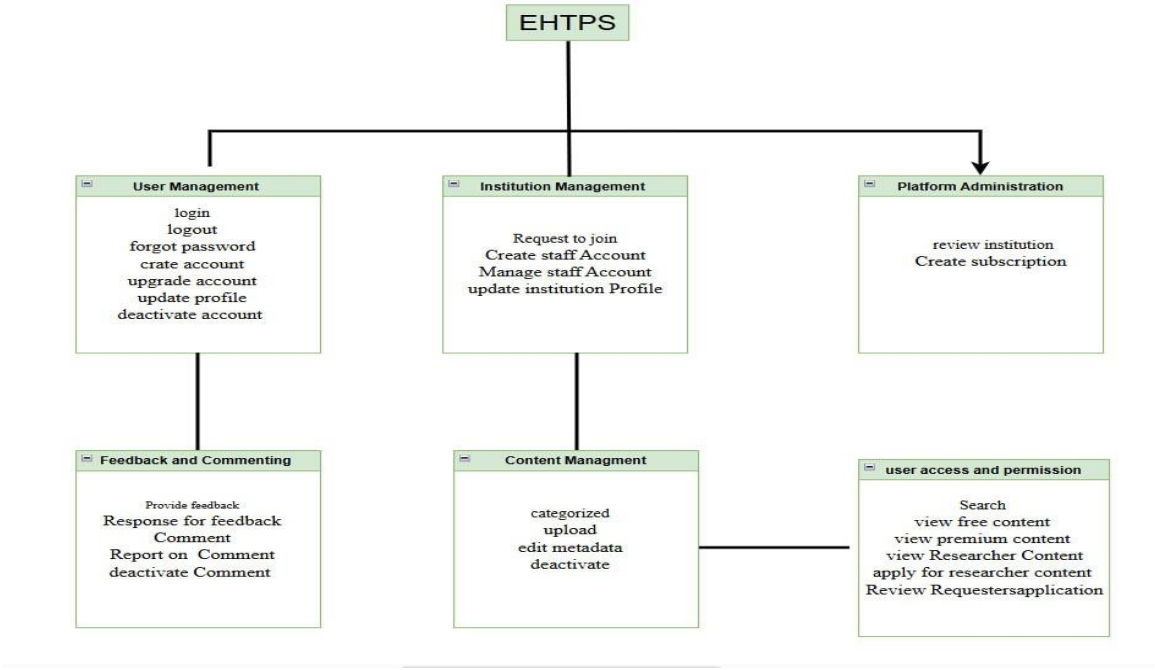


Figure 3.1 Subsystem Decomposition

3.4 Detailed Design

3.4.1 Class Diagram

UML class diagrams are the mainstay of object-oriented analysis and design. UML class diagrams show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations and attributes of the classes. Class diagrams are used for a wide variety of purposes, including both conceptual/domain modeling and detailed design modeling [23].

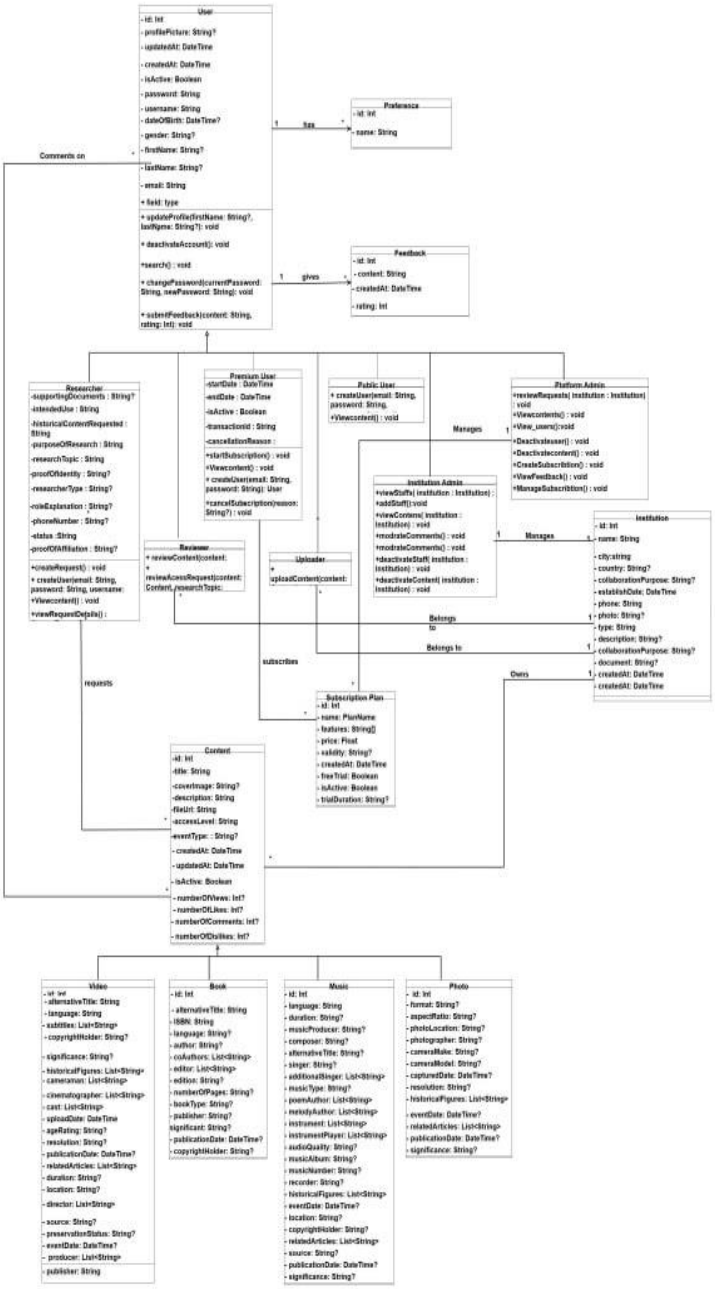


Figure 3.2 Class diagram

3.4.2 Persistence Diagram

A Persistence Model is a design framework used in software systems to define how data is stored, retrieved, and managed in a persistent storage medium, such as a database or file system. It ensures that data remains available and intact even after the application is closed or restarted. The model typically maps objects or entities in an application to corresponding tables or structures in the storage system, often using Object-Relational Mapping (ORM) tools or similar mechanisms. [24].

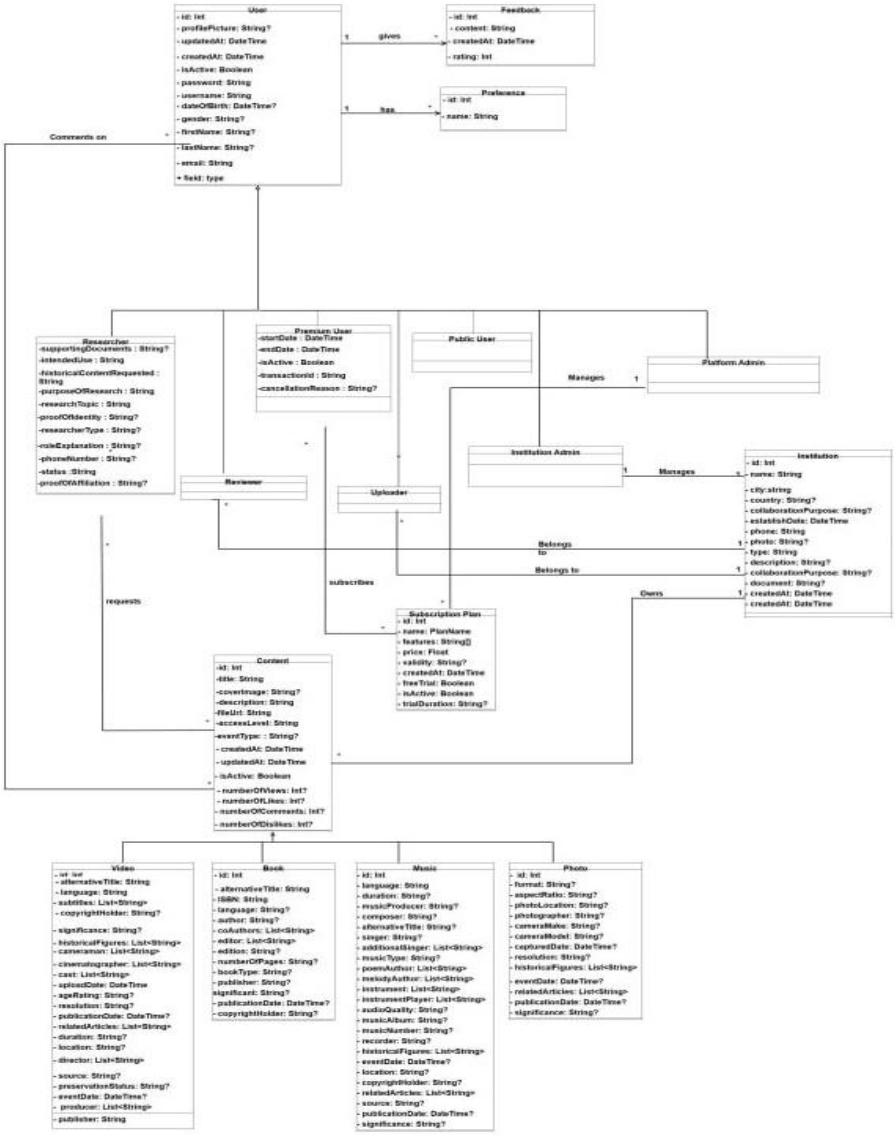


Figure 3.3 Persistence Diagram

3.4.3 Normalization

In relational databases, especially large ones, you need to arrange entries so that other maintainers and administrators can read them and work on them. This is why database normalization is important [25]. In simple words, database normalization entails organizing a database into several tables in order to reduce redundancy. You can design the database to follow any of the types of normalization such as 1NF, 2NF, and 3NF [25].

3.4.3.1 Unnormalized Data

1. **User** (user_id, profile_picture, email, first_name, last_name, gender, date_of_birth, username, password, user_type, phone_number, role , researcher_type , position_title, proof_of_affiliation,proof_of_identity,role_explanation,research_topic,purpose_of_research, historical_content_requested, intended_use, supporting_documents , startDate, EndDate, tranactionId, cancelreason ,User_id , Sub_Plan_Id , contend_id , institution_id)
2. **Feedback** (feedback_id, content, created_at, user_id, rating)
3. **Preference** (preference_id, name)
4. **Institution**(institution_id, name, city, country, website, email_domain, collaboration_purpose, establish_date, number_of_employ, document, photo, type, phone, registration_status)
5. **Plan**(plan_id, name, features, price, validity, created_at, free_trial, trial_duration, is_active)Content(content_id, title, cover_image, description, file_url, access_level, uploader_id, reviewer_id, institution_id, created_at, updated_at, is_active, last_edit_by, last_edit_date, user_id)
6. **Video**(video_id, alternativeTitle, language, significance, publisher, cameraman, eventDate, preservationStatus, source, ageRating, location, resolution, duration, publicationDate, content_id)
7. **Book**(book_id, alternativeTitle, ISBN, language, author, numberOfPages, bookType, publisher, significance, publicationDate, content_id)
8. **Music**(music_id, alternativeTitle, language, duration, audioQuality, musicAlbum, musicNumber, recorder, eventDate, location, significance, source, publicationDate, content_id)

9. **Photo**(photo_id, format, resolution, aspectRatio, photoLocation, capturedDate, photographer, cameraMake, cameraModel, eventDate, reviewerBy, significance, content_id)
10. **role** (platform_admin, premium_user, public_user, researcher_user, institution_admin, uploader, reviewer)

3.4.3.2 First Normal Form(1NF)

First Normal Form (1NF) Requirements:

1. **Atomicity**: Each field must contain atomic (indivisible) values. There should be no repeating groups or lists within a single field.
2. **Uniqueness**: Each record in the table must be unique, identified by a primary key.
3. **Order**: The order of rows or columns does not affect the data structure. Data should be logically organized into separate tables to maintain relational integrity.

The **User Table** violate **First Normal Form (1NF)** if it contains fields that are **multi-valued** or **non-atomic**, meaning they hold multiple values within a single field. This is a violation of the basic principles of relational databases, leading to data redundancy and inefficiency in data retrieval and updates.

Potential Violations in the User Table

The following fields in the User table could violate **1NF** by holding **multi-valued** or **non-atomic** data:

1. Research Topics:

- If a user can have **multiple research topics** (e.g., AI, Machine Learning, Data Science), and this is stored in a single field, it would violate 1NF. This is because that field would contain multiple values, which are not atomic.

2. Purpose of Research:

- If a user can specify **multiple purposes for their research** (e.g., Academic, Commercial, Personal), and this is stored in a single field, it would violate 1NF. Each field must contain only **one value**.

3. Historical Content Requested:

- If a user requests **multiple historical contents** and stores them in a single field (e.g., History of AI, History of Quantum Computing), it would violate 1NF because that field holds **multiple values**.

4. Supporting Documents:

- If multiple **document URLs or paths** are stored in a single field, like a comma-separated list or a JSON object, it would violate 1NF. Supporting documents should be stored atomically in separate rows or tables if necessary.

  To address with **1NF**, we must **separate multi-valued attributes** into distinct tables. Instead of storing multiple values in the research_topic, purpose_of_research, or historical_content_requested fields, we can create new tables to hold these values, with a **foreign key** referencing the User table.

To address the multi-valued fields associated with **user research requests**, we propose creating an **Access Request Table**. This table will store detailed information about the user's request for access, such as research topics, purposes of research, and supporting documents.

Access Request Table Design:-

AccessRequest (id, createdAt, status, userId, contentId, reviewerId, phoneNumber, researcherType, institutionName, positionTitle, proofOfAffiliation, proofOfIdentity, roleExplanation, researchTopic, purposeOfResearch, historicalContentRequested, intendedUse, supportingDocuments)

1. In **User table** stores the user's subscription details such as plan_id, start_date, and end_date. This design leads to the following issues:

1. **Data Duplication**: If a user has multiple subscriptions or changes their plan over time, the User table stores redundant information (e.g., the same user's first_name, last_name, etc.) for each subscription, leading to data duplication.
2. **Violation of 1NF (First Normal Form)**: Storing multiple subscription details in a single table row violates the **atomicity** requirement of 1NF. Multiple values for plan_id, start_date, and end_date are being stored in the same table, which results in non-atomic data.

Problem:

When multiple subscription details for a single user are stored in the **User table**, it results in:

- **Redundant Data**: Repeating the same user information across different rows for every subscription.
- **Non-Atomic Fields**: The subscription details (plan_id, start_date, end_date) are combined in a single field, which violates the principle of atomicity in database design.

To resolve the above issues and normalize the database to **First Normal Form (1NF)**, we propose the following solution:

1. **Create a separate Subscription table** to store the subscription details.
2. Link the Subscription table to the User table via a **foreign key relationship**.

This solution ensures that subscription details are stored in a **separate, atomic** table, removing redundancy and ensuring that each user's subscription data is treated as a distinct entity.

UserSubscription (id, startDate, endDate, isActive, userId, planId, transactionId, cancellationReason)

1NF

1. UserSubscription(id, startDate, endDate, isActive, userId, planId, transactionId, cancellationReason)
2. AccessRequest(id, createdAt, status, userId, contentId, reviewerId, phoneNumber, researcherType, institutionName, positionTitle, proofOfAffiliation, proofOfIdentity, roleExplanation, researchTopic, purposeOfResearch, historicalContentRequested, intendedUse, supportingDocuments)
3. User(user_id, profile_picture, email, first_name, last_name, gender, date_of_birth, username, password, user_type, phone_number, role, content_id)
4. Preference(preference_id, name)
5. Feedback(feedback_id, content, created_at, user_id, rating)Preference(preference_id, name)
6. Institution(institution_id, name, city, country, website, email_domain, collaboration_purpose, establish_date, number_of_employ, document, photo, type, phone, registration_status)
7. Plan(plan_id, name, features, price, validity, created_at, free_trial, trial_duration, is_active)
8. Content(content_id, title, cover_image, description, file_url, access_level, uploader_id, reviewer_id, institution_id, created_at, updated_at, is_active, last_edit_by, last_edit_date, user_id)
9. Video(video_id, alternativeTitle, language, significance, publisher, cameraman, eventDate, preservationStatus, source, ageRating, location, resolution, duration, publicationDate, content_id)
10. Book(book_id, alternativeTitle, ISBN, language, author, numberOfPages, bookType, publisher, significance, publicationDate, content_id)
11. Music(music_id, alternativeTitle, language, duration, audioQuality, musicAlbum, musicNumber, recorder, eventDate, location, significance, source, publicationDate, content_id)

12. Photo(photo_id, format, resolution, aspectRatio, photoLocation, capturedDate, photographer, cameraMake, cameraModel, eventDate, reviewerBy, significance, content_id)
13. **role** (platform_admin, premium_user, public_user, researcher_user, institution_admin, uploader, reviewer)

3.4.3.3 Second Normal Form (2NF)

All the tables meet the criteria for the Second Normal Form (2NF), each table has a primary key, and all non-key attributes are fully functionally dependent on the primary key

3.4.3.4 Third Normal Form (3NF)

All tables satisfy the criteria for Third Normal Form 3NF, each table is in 2NF and all non-key attributes are non-transitively dependent on the primary key.

Users can comment on content, and multiple users can comment on a single content item. This results in a **many-to-many relationship** between the **User** and **Content** entities. If we attempt to store all the comments directly within the **Content** table, we risk violating the **Third Normal Form (3NF)** of database normalization. This proposal outlines the solution to resolve this issue and properly normalize the database structure by creating a separate **Comment** table to handle the comments independently.

When users comment on content, it introduces a many-to-many relationship where:

- A **User** can comment multiple times on different content items.
- Multiple **Users** can comment on a single content item.

If we directly store comments in the **Content** table, it creates transitive dependencies, making it harder to maintain the database. Specifically, the **Content** table would store redundant and repetitive information about users and their comments.

Solution: Create a Comment **Table**

To resolve this issue, we propose creating a new **Comment** table to handle the many-to-many relationship between users and content. This approach will eliminate the direct dependency between the **User** and **Content** tables in regard to comments, making the database schema more efficient and normalized.

New Comment Table: Comment (comment_id , content_id, comment_text ,created_at)

Example:

- **User 1** comments on **Content A**.
- **User 2** comments on **Content A**

comment_id	content_id	comment_text	created_at
User 1	**A**	**"Great content!"**	**23/09/9**
User 2	**A**	**"Interesting!"**	**23/09/8**

3NF

1. UserSubscription(id, startDate, endDate, isActive, userId, planId, transactionId, cancellationReason)
2. AccessRequest(id, createdAt, status, userId, contentId, reviewerId, phoneNumber, researcherType, institutionName, positionTitle, proofOfAffiliation, proofOfIdentity, roleExplanation, researchTopic, purposeOfResearch, historicalContentRequested, intendedUse, supportingDocuments)
3. User(user_id, profile_picture, email, first_name, last_name, gender, date_of_birth, username, password, user_type, phone_number, role)
4. Preference(preference_id, name)

5. Feedback(feedback_id, content, created_at, user_id, rating)Preference(preference_id, name)
6. Institution(institution_id, name, city, country, website, email_domain, collaboration_purpose, establish_date, number_of_employ, document, photo, type, phone, registration_status)
7. Plan(plan_id, name, features, price, validity, created_at, free_trial, trial_duration, is_active)
8. Content(content_id, title, cover_image, description, file_url, access_level, uploader_id, reviewer_id, institution_id, created_at, updated_at, is_active, last_edit_by, last_edit_date)
9. Video(video_id, alternativeTitle, language, significance, publisher, cameraman, eventDate, preservationStatus, source, ageRating, location, resolution, duration, publicationDate, content_id)
10. Book(book_id, alternativeTitle, ISBN, language, author, numberOfPages, bookType, publisher, significance, publicationDate, content_id)
11. Music(music_id, alternativeTitle, language, duration, audioQuality, musicAlbum, musicNumber, recorder, eventDate, location, significance, source, publicationDate, content_id)
12. Photo(photo_id, format, resolution, aspectRatio, photoLocation, capturedDate, photographer, cameraMake, cameraModel, eventDate, reviewerBy, significance, content_id)
13. Comment (comment_id , content_id, comment_text ,created_at)
14. **role** (platform_admin, premium_user, public_user, researcher_user, institution_admin, uploader, reviewer)

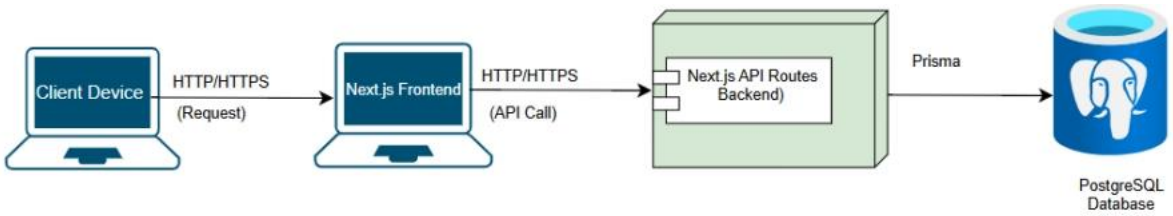
3.4.4 Deployment Diagram

A Deployment Diagram in software engineering is a type of Structural UML Diagram that shows the physical deployment of software components on hardware nodes. It illustrates the mapping of software components onto the physical resources of a system, such as servers, processors, storage devices, and network infrastructure [21].

In our project, the deployment diagram represents how the various components of our web application interact and are deployed across different nodes in the system. The application architecture includes a frontend built with Next.js, a backend also implemented with Next.js API routes, and the Prisma ORM interacting with a PostgreSQL database for data storage.

Our project's deployment diagram outlines the flow of information from the client device to the server and the database. Here's an overview of the components involved in our system's deployment:

1. **Client Device**:
    - Sends HTTP requests to the Next.js Frontend for web pages or API data.
2. **Next.js Frontend**:
    - Serves the UI and communicates with the Next.js Backend via HTTP API requests.
3. **Next.js Backend (API Routes)**:
    - Handles business logic, user authentication (JWT), and interacts with the Prisma ORM.
4. **Prisma ORM**:
    - Queries and updates the PostgreSQL database.
5. **PostgreSQL Database**:
    - Stores all application data.


Client Device
HTTP/HTTPS
(Request)
Next.js Frontend
HTTP/HTTPS
(API Call)
Next.js API Routes
Backend)
Prisma
PostgreSQL
Database

Chapter 4

4. Implementation and Testing

4.1 Introduction

The Implementation and Testing phase marks a critical transition where the system moves from conceptual design to a fully functional and operational platform. This chapter outlines the practical development of the system, highlighting the tools and technologies utilized, along with the thorough testing processes employed to ensure its reliability, scalability, and user-friendliness.

The implementation phase of the Ethiopian History Preservation and Transmission System (EHPTS) focused on transforming the system's design into a robust and user-centric platform. The goal was to centralize, preserve, and present Ethiopia's rich historical resources, leveraging modern web technologies to deliver both efficiency and scalability.

The process began with crafting the design in Figma, ensuring a visually cohesive and intuitive user interface. Next, the development environment was meticulously configured, integrating essential dependencies and libraries for both frontend and backend operations. The frontend was built using Next.js with TypeScript and Tailwind CSS, providing a seamless and responsive user experience. The backend leveraged PostgreSQL with Prisma ORM to enable reliable and scalable data management. Key functionalities such as user authentication and payment processing were implemented using NextAuth and Chapa, ensuring secure and streamlined interactions.

Why Testing Matters

Testing was critical to ensure the platform works as expected. Every feature, from user login to content searches, was tested to find and fix issues. This included checking that the platform performs well under heavy use, handles errors gracefully, and provides a smooth experience across different devices.

By carefully building and testing the system, the EHPTS is now a reliable and easy-to-use platform for preserving and sharing Ethiopia's history.

4.2 Algorithm Design

4.2.1 Authentication Workflow

Purpose: Ensures secure user login, token management, and role-based access control.

- **Frontend**: Validates input, sends login requests, and redirects users based on roles.
- **Backend**: Verifies credentials, generates JWTs, and handles account or role issues.

4.2.1.1 Pseudo code

```
Function handleLogin(username, password):
    IF username OR password is empty:
        RETURN "Error: All fields are required"

    RESPONSE = send POST request to "/api/auth/signin" with username and
password

    IF RESPONSE is successful:
        TOKEN = decode JWT from RESPONSE
        STORE TOKEN and user details in sessionStorage
        REDIRECT user based on role:
            IF role == "PUBLIC_USER":
                REDIRECT to "/content/content-page"
            ELSE IF role == "PLATFORM_ADMIN":
                REDIRECT to "/dashboard/overview"
            ELSE IF role == "UPLOADER":
                REDIRECT to "/uploader-dashboard/overview"
            ELSE:
                REDIRECT to "/auth/sign-in"
    ELSE:
        RETURN "Error: Invalid credentials or account inactive"
```

4.2.2 Upgrade Subscription Workflow

Purpose: Allows users to upgrade to premium plans via Chapa payment integration.

- **Frontend**: Fetches plans, displays them, and redirects to the payment page.
- **Backend**: Validates inputs, initializes payments, and updates subscriptions upon success.

4.2.2.1 Pseudocode

```
Function handleUpgrade(userId, email, planId):
    IF userId OR email is missing:
        RETURN "Error: Please log in"

    PLAN_DETAILS = fetch selected plan details
    IF PLAN_DETAILS is invalid:
        RETURN "Error: Invalid plan selected"

    RESPONSE = send POST request to "/api/chapa/payment" with:
        - amount
        - email
        - userId
        - planId

    IF RESPONSE is successful:
        REDIRECT to payment URL (checkoutUrl)
    ELSE:
        RETURN "Error: Failed to initiate payment"
```

4.2.3 Music Management Workflow

Purpose: Supports uploading music files with metadata, managing records, and toggling content statuses.

- **Frontend**: Uploads files, manages metadata, and displays paginated records with search/filter options.
- **Backend**: Validates uploads, stores files/metadata, and handles content visibility toggles.

2.2.3.1 Pseudocode

```
Function uploadMusic(musicFile, coverImage, metadata):
    IF musicFile OR coverImage is missing:
        RETURN "Error: Both music and cover image are required"

    IF metadata is incomplete:
        RETURN "Error: Missing required metadata fields"

    SAVE musicFile to "/uploads/HistoricalMusic"
    SAVE coverImage to "/uploads/CoverImage"

    FORMAT metadata:
        - Convert eventType to uppercase
        - Parse multi-value fields into arrays (e.g., instruments, additional
singers)

    INSERT data into the database:
        - File paths for music and cover image
        - Metadata fields (title, composer, language, etc.)

    IF database insertion is successful:
        RETURN "Success: Music uploaded successfully"
    ELSE:
        RETURN "Error: Failed to upload music
```

4.3 Screenshots and Sample Code

This section showcases the user interface and provides key snippets of the Ethiopian History Preservation and Transmission System (EHPTS) implementation. Screenshots highlight the system's interface, while annotated code snippets illustrate critical functionalities. These

elements collectively demonstrate the practical realization of the platform's design and objectives.

4.3.1 Screenshots

4.3.1.2 Homepage

The homepage introduces users to the Ethiopian History Preservation and Transmission System (EHPTS) with an engaging design and clear navigation.

Screenshot:

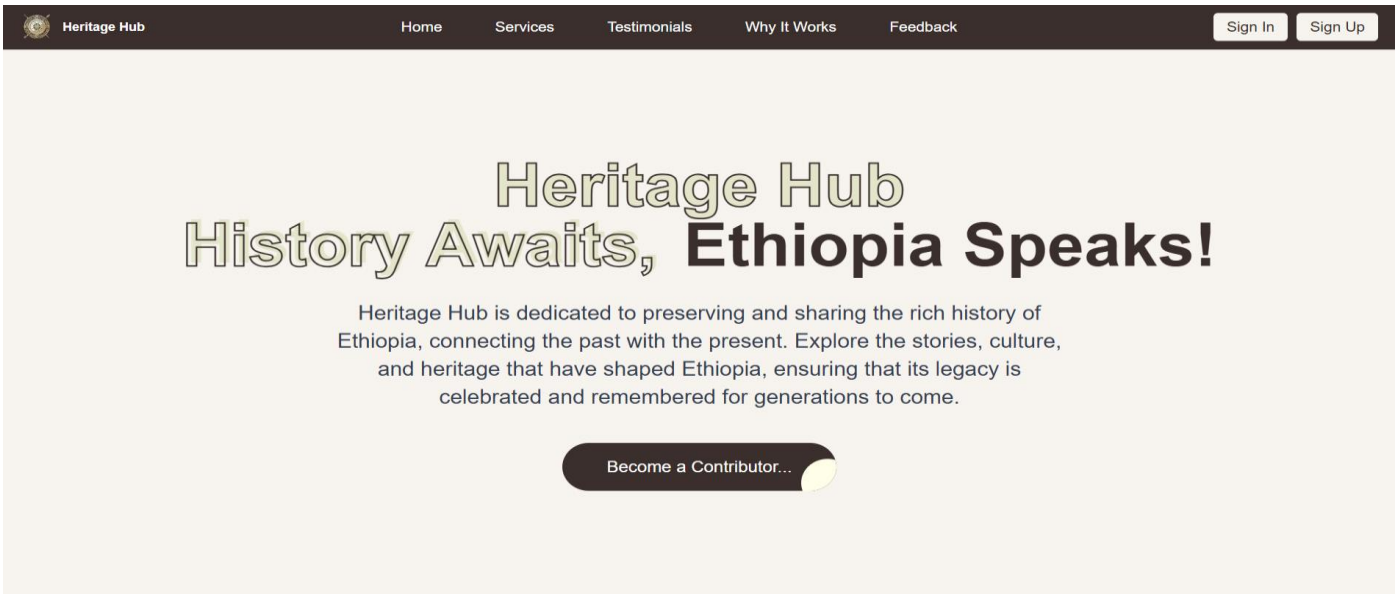


Figure 4.1 Homepage

Annotation:

- **Header Navigation**: Provides links to sections like "Home," "Services," and "Feedback" for seamless navigation.

- **Tagline**: "History Awaits, Ethiopia Speaks!" highlights the platform's mission to preserve Ethiopia's heritage.
- **Sign In/Sign Up**: Top-right buttons for secure user access and account creation.
- **Call-to-Action**: The "Become a Contributor" button encourages users to share historical content. **Purpose Section**: Briefly explains the platform's role in preserving and sharing Ethiopia's rich history.

4.3.1.2 SignUp

The sign-up page allows users to create an account and become a part of the Ethiopian History Preservation and Transmission System (EHPTS), enabling access to valuable historical resources and a collaborative community.

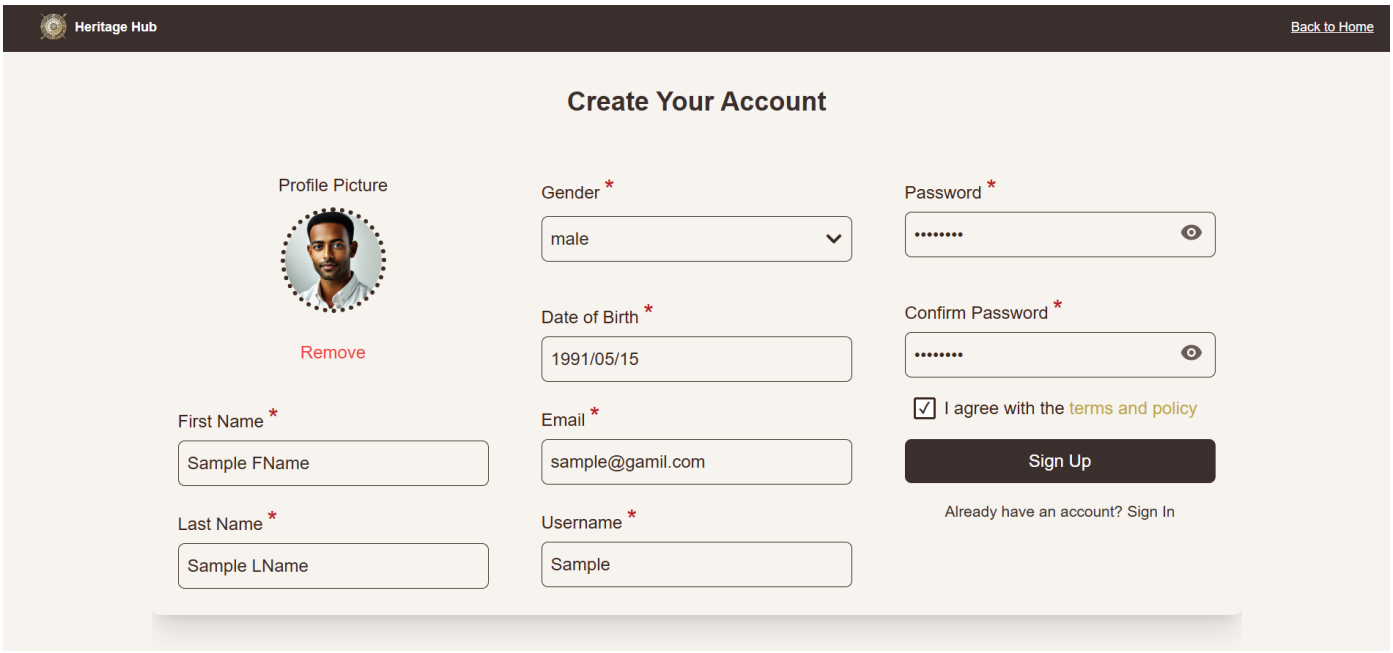


Figure 4.2 Signup

Annotation:

- **Header Navigation**: A "Back to Home" button for easy navigation to the landing page.

- **Form Fields**: Includes fields for uploading a profile picture, entering personal details (name, email, username, password), and agreeing to terms and conditions.
- **Submit Button**: Guides users to complete registration.
- **Purpose Section**: Highlights the benefits of joining the Ethiopian History Preservation and Transmission System (EHPTS) for accessing historical resources.

4.3.1.3 Sign In

The sign-in page provides registered users with access to the **Ethiopian History Preservation and Transmission System (EHPTS)**. By entering their credentials, users can securely log in to explore and contribute to the system's extensive historical resources and engage with the collaborative community.

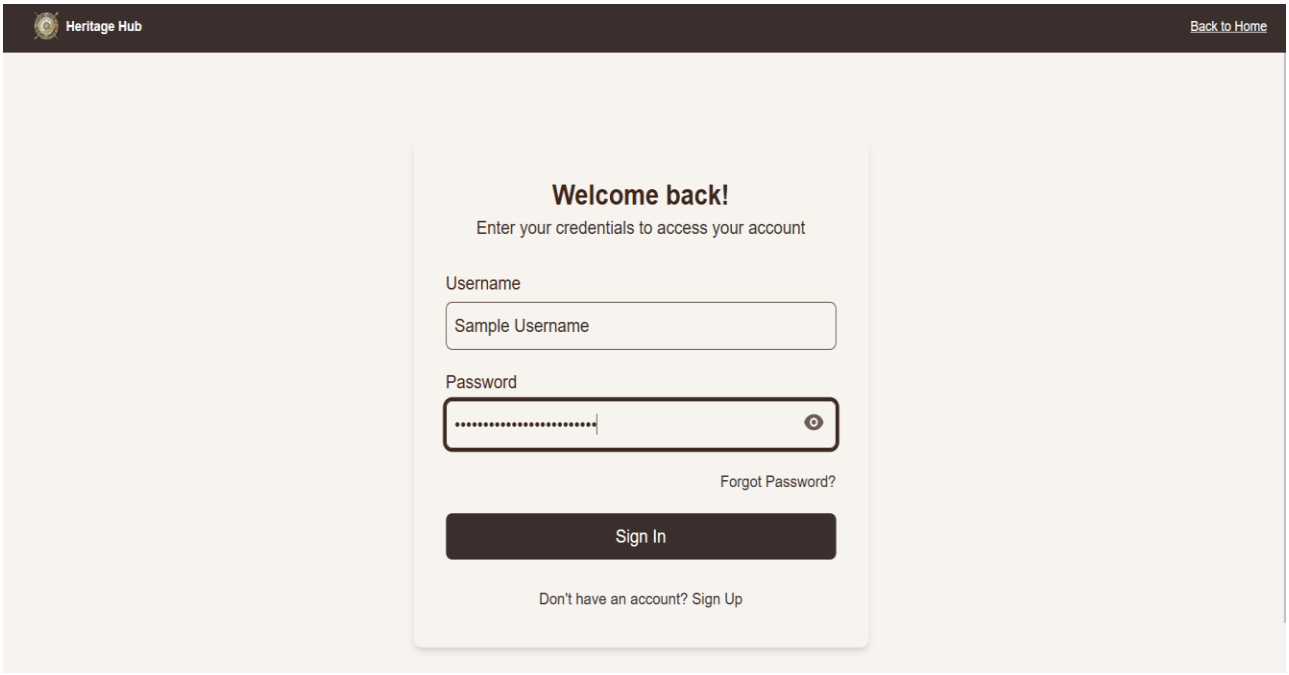


Figure 4.3 Sign in

Annotation:

- **Header**: A "Back to Home" button for easy navigation to the main page.
- **Welcome Message**: Encourages users to log in.
- **Login Form**: Fields for username and password with a "Show Password" option, and a "Forgot Password?" link for account recovery.
- **Sign-In Button**: Clearly labeled for user convenience.
- **Sign-Up Option**: Redirects new users to create an account.
- **Why Sign In?**: Highlights the benefits of accessing EHPTS resources and connecting with history enthusiasts.

4.3.1.4 Upgrade

The upgrade page allows users to enhance their experience by unlocking premium features within the Ethiopian History Preservation and Transmission System (EHPTS), providing access to exclusive content, advanced tools, and greater involvement in the platform's mission to preserve and share Ethiopia's rich heritage.

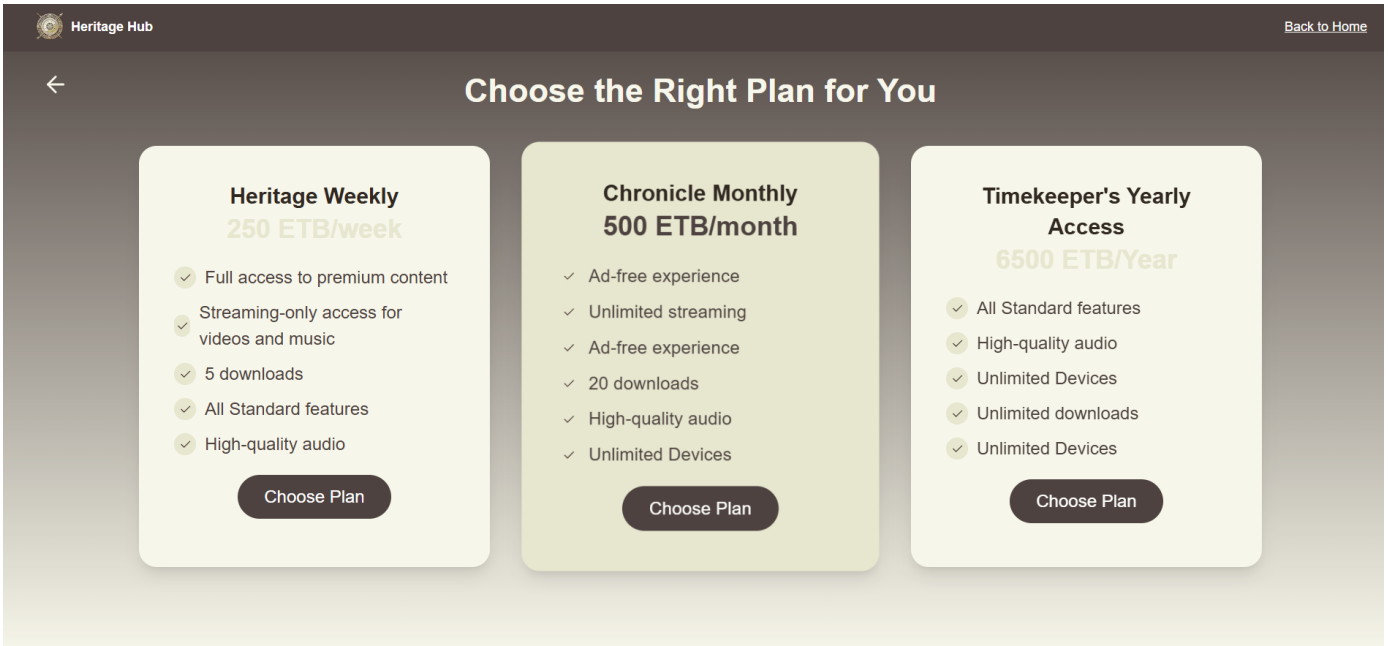


Figure 4.4 upgrade

Annotation:

- **Header Navigation**: Features a "Back to Home" button for easy navigation and an arrow button for returning to the previous page.
- **Plan Selection**: Displays cards with various upgrade plans, each detailing features and benefits.
- **Plan Buttons**: A "Choose Plan" button allows users to proceed to the payment process for the selected plan.
- **Purpose Section**: Highlights the advantages of subscribing to a premium plan, encouraging greater involvement in EHPTS and promoting Ethiopia's historical preservation.

4.3.1.5 Content View

Browse and interact with a curated selection of Ethiopia's historical and cultural content, displayed in easy-to-navigate cards. Access exclusive materials and deepen your understanding of Ethiopia's heritage,

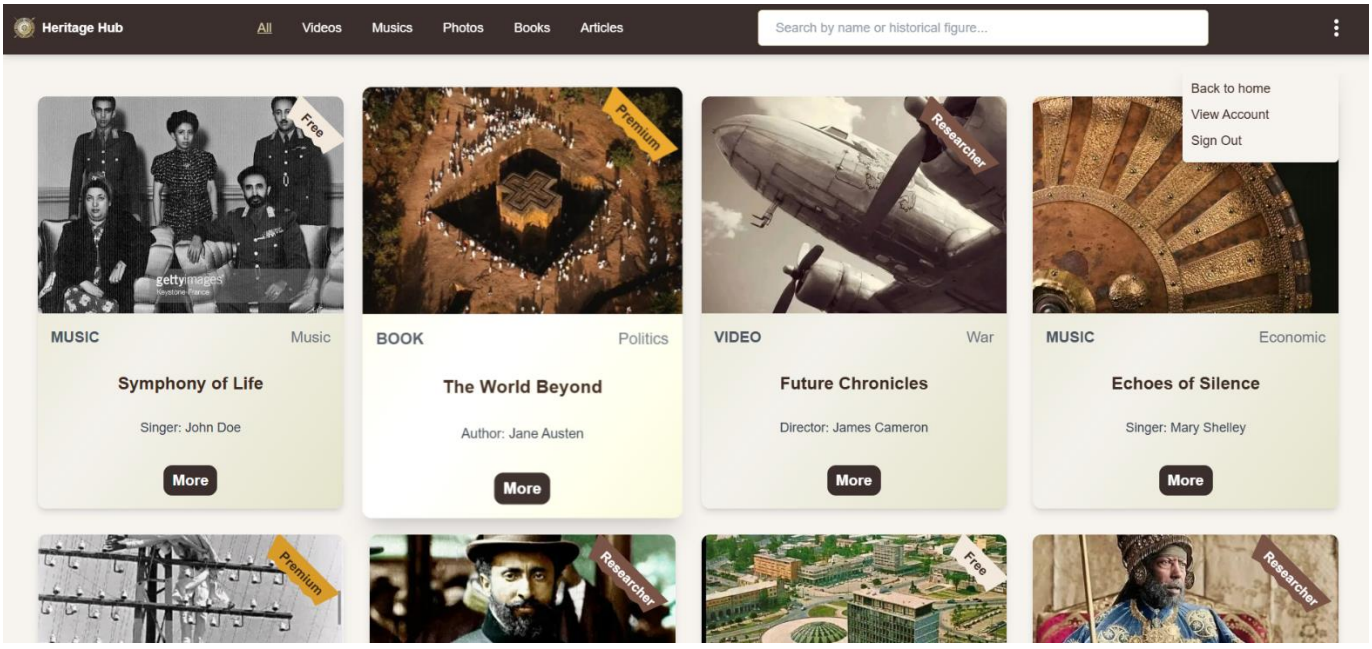


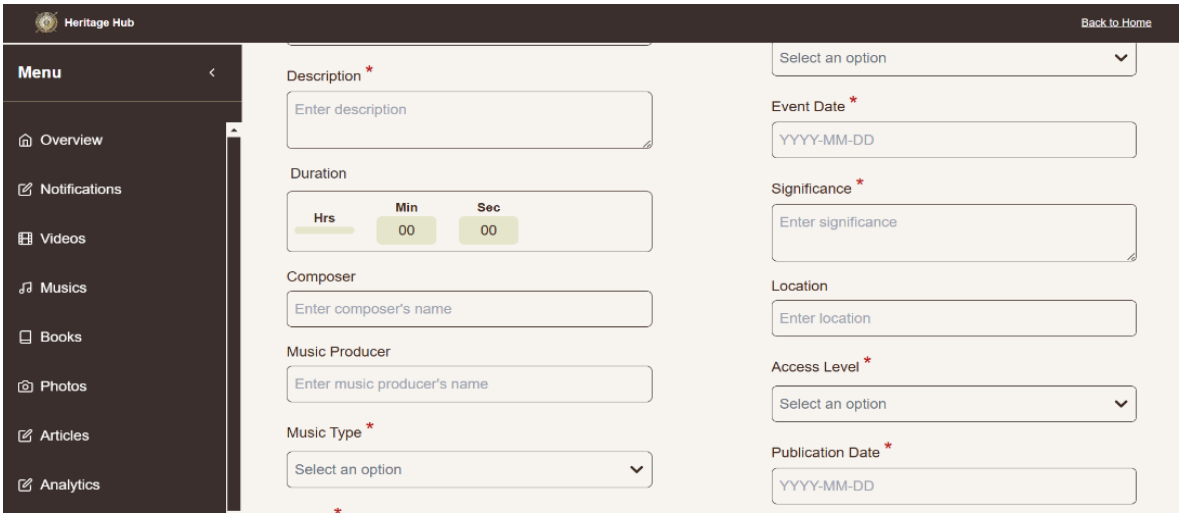
Figure 4.5 Content view

Annotation:

- **Navigation Links**: Provides links to key sections like Video, Book, Music, Photo, and Article for easy category-based browsing.
- **Search Input**: A search bar in the top-right corner for finding specific content across categories.
- **Three-Dot Menu**: Quick-access options for "View Account," "Back to Home," and "Sign Out" to manage navigation and account settings.
- **Content Highlights**: Displays content cards that briefly showcase Ethiopia's historical and cultural materials.
- **More Button**: Each card includes a "More" button to view detailed information, promoting in-depth exploration..

4.3.1.6 Music Upload

The Music Upload Page allows users to contribute to the Ethiopian History Preservation and Transmission System (EHPTS) by adding and organizing music content. This feature empowers users to preserve Ethiopia's rich musical heritage, ensuring its accessibility for future generations while enhancing the platform’s cultural archive.



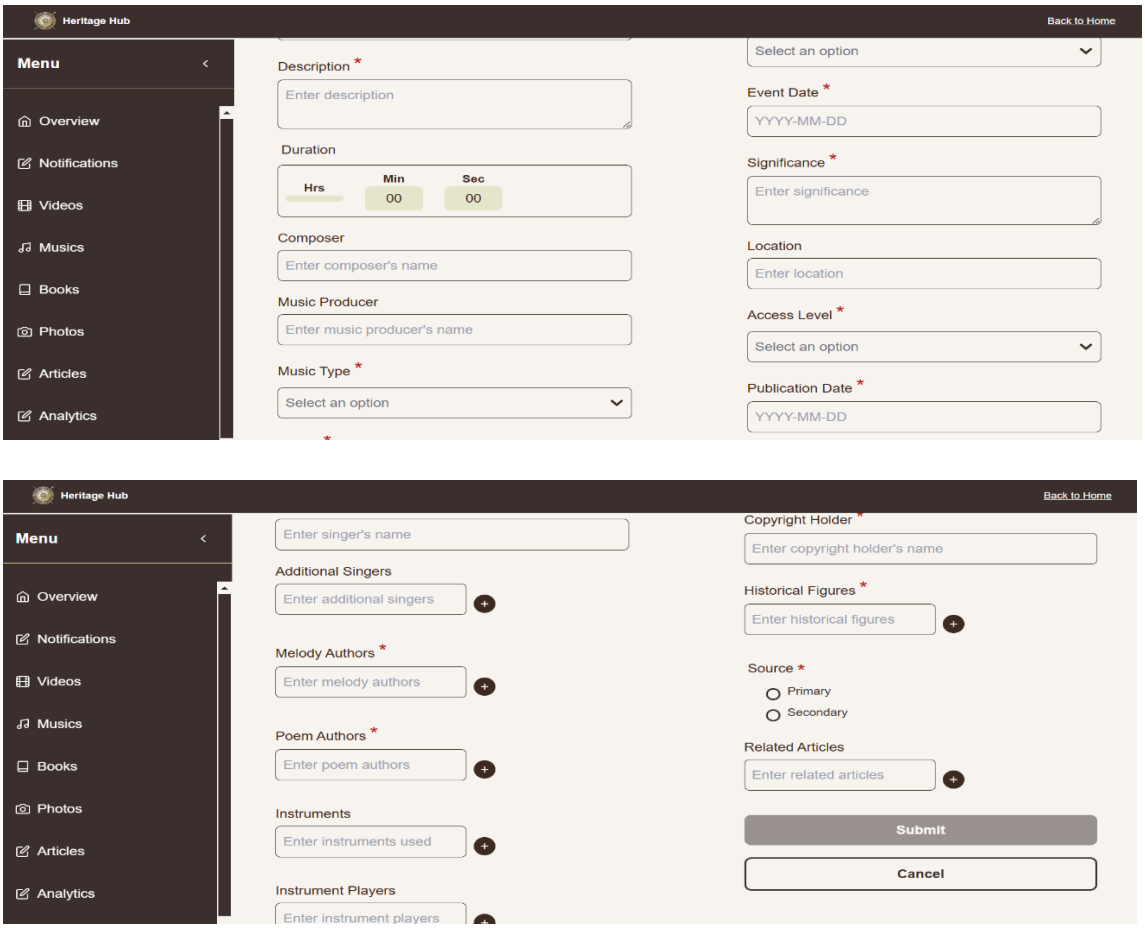


Figure 4.6 Music upload

Annotation:

- **Header Navigation**:
  - Includes a "Back to Music List" button for easy navigation to the music management section.
  - A "Back to Home" link is provided on the top-right corner for returning to the homepage.
- **Left Menu**:
  - Features a collapsible menu with links to sections like Overview, Notifications, Videos, Music, Books, Photos, Articles, and Analytics for effortless navigation between functionalities.
- **Music Upload Form**:

- Includes an "Upload Music" button for adding audio files.
- Provides fields to input metadata such as title, composer, language, and other relevant details.
- After completing the form, users can:
  - Click "Submit" to save the music data.
  - Click "Cancel" to discard changes and exit the form.

4.3.2 Sample Codes

```
import { NextResponse } from 'next/server';
import type { NextRequest } from 'next/server';

const verifyToken = (token: string) => {
  try {
    const decoded = JSON.parse(atob(token.split('.')[1]));
    return decoded;
  } catch (e) {
    throw new Error('Invalid token');
  }
};

export function middleware(req: NextRequest) {
  const token = req.cookies.get('token')?.value;
  console.log('Token:', token);


  if (!token) {
    return NextResponse.redirect(new URL('/auth/sign-in', req.url));
  }

  try {
    const decoded = verifyToken(token);
    console.log('Decoded:', decoded);

    const { pathname } = req.nextUrl;



    if (pathname.startsWith('/uploader-dashboard') && decoded.role !== 'UPLOADER') {

      return NextResponse.redirect(new URL('/auth/sign-in', req.url));
    }

    if (pathname.startsWith('/dashboard') && decoded.role !== 'PLATFORM_ADMIN') {
```

```
export function middleware(req: NextRequest) {
    }

    if (pathname.startsWith('/dashboard/overview') && decoded.role !== 'PLATFORM_ADMIN') {

      return NextResponse.redirect(new URL('/auth/sign-in', req.url));
    }

    if (pathname.startsWith('/reviewer-dashboard') && decoded.role !== 'REVIEWER') {

      return NextResponse.redirect(new URL('/auth/sign-in', req.url));
    }

    if (pathname.startsWith('/content') && decoded.role !== 'PUBLIC_USER' && decoded.role !== 'PREMIUM_USER' && decoded.role !== 'RESEARCHER_USER'

      return NextResponse.redirect(new URL('/auth/sign-in', req.url));
    }

    if (pathname.startsWith('/admin') && decoded.role !== 'PLATFORM_ADMIN') {

      return NextResponse.redirect(new URL('/auth/sign-in', req.url));
    }

  } catch (error) {
    console.error('JWT Error:', error);

    return NextResponse.redirect(new URL('/auth/sign-in', req.url));
  }

  return NextResponse.next();
}


export const config = {
  matcher: ['/uploader-dashboard/:path*', '/reviewer-dashboard/:path*', '/content/:path*', '/admin/:path*' , '/dashboard/:path*',
    '/dashboard/overview:path*'],
};
```

Figure 4.7Authentication code

This middleware handles **authentication** and **role-based access control**.

It:

1. Verifies user tokens to ensure they are valid.
2. Redirects unauthenticated users to the login page.
3. Checks user roles (e.g., UPLOADER, PLATFORM_ADMIN, etc.) and ensures they have permission to access specific routes.
4. Logs errors and provides a route-matching configuration to apply the middleware selectively.

```
const ethiopianDateSchema = Yup.string()
    .required("Date is required")
    .test("is-ethiopian-date", "Invalid date format. Use 'YYYY-MM-DD'.", (value) => {
        if (!value) return false;
        const match = value.match(/^(\d{4})-(\d{1,2})-(\d{1,2})$/);
        if (!match) return false;

        const [_, year, month, day] = match.map(Number);

        if (isNaN(year) || isNaN(month) || isNaN(day)) return false;
        if (month < 1 || month > 13) return false;
        if (day < 1 || day > 30) return false;
        if (month === 13 && (day < 1 || day > (isLeapYear(year) ? 6 : 5))) {
            return false;
        }
        return true;
    });

const musicSchema = Yup.object().shape({
    title: Yup.string().required("Title is required"),
    alternativeTitle: Yup.string().optional(),
    coverImage: Yup.mixed()
        .required("Album image is required")
        .test("fileType", "Unsupported file type", (value) => value && (value as File).type.startsWith("image/"))
        .test("fileSize", "File size must be less than 5MB", (value) => value && (value as File).size <= 5 * 1024 * 1024),
    music: Yup.mixed()
        .required("Music file is required")
        .test("fileType", "Unsupported audio file type", (value) => value && (value as File).type.startsWith("audio/"))
        .test("fileSize", "File size must be less than 10MB", (value) => value && (value as File).size <= 10 * 1024 * 1024),
    eventDate: ethiopianDateSchema,
```

```
        .test("fileSize", "File size must be less than 10MB", (value) => value && (value as File).size <= 10 * 1024 * 1024),
    eventDate: ethiopianDateSchema,
    publicationDate: ethiopianDateSchema,
    language: Yup.string().required(),
    description: Yup.string().required("Description is required"),
    duration: Yup.string().required("Duration is required"),
    composer: Yup.string().optional(),
    musicProducer: Yup.string().optional(),
    musicType: Yup.string().required("Music Type is required"),
    singer: Yup.string().required("Singer information is required"),
    additionalSinger: Yup.array().of(Yup.string()).optional(),
    melodyAuthor: Yup.array().of(Yup.string()).required("Melody Author(s) is/are required"),
    poemAuthor: Yup.array().of(Yup.string()).required("Poem Author(s) is/are required"),
    accessLevel: Yup.string().required("Access Level is required"),
    instrument: Yup.array().of(Yup.string()).optional(),
    instrumentPlayer: Yup.array().of(Yup.string()).optional(),
    audioQuality: Yup.string().required("Audio Quality is required"),
    musicAlbum: Yup.string().optional(),
    musicNumber: Yup.string().optional(),
    recorder: Yup.string().optional(),
    eventType: Yup.string().required("Event Type is required"),
    historicalFigures: Yup.array().of(Yup.string()).required("Historical Figure(s) is/are required"),
    location: Yup.string().optional(),
    significance: Yup.string().required("Significance is required"),
    source: Yup.string().required("Source is required"),
    copyrightHolder: Yup.string().required("Copyright Holder is required"),
    relatedArticles: Yup.array().of(Yup.string()).optional(),
});
```

Figure 4.8 Validation schema code

The provided code defines a **Yup validation schema** for validating input fields related to music data, including:

1. **Custom Ethiopian Date Validation**:
    - Ensures the date format is valid (YYYY-MM-DD) and within the Ethiopian calendar constraints (handles leap years, etc.).
2. **File Validations**:
    - Validates cover Image and music file inputs:
        - Ensures correct file types (image/ for cover images, audio/ for music).
        - Checks file size limits (5MB for images, 10MB for music).
3. **Music Details**:
    - Validates fields like title, description, duration, and other metadata (e.g., singer, music type, melody/poem authors).
    - Optional fields include composer, producer, additional singers, and instrument details.
4. **Access Control & Metadata**:
    - Checks for required roles, access levels, and related fields (e.g., significance, source, historical figures, copyright holder).

4.4 Test Report

Testing was a critical phase of the Ethiopian History Preservation and Transmission System (EHPTS) project, ensuring that the system operates reliably, securely, and efficiently. Using a **manual testing approach**, every component, feature, and workflow was rigorously tested during development. The testing process focused on identifying and resolving issues early, validating the integration of frontend and backend functionalities, and ensuring the system meets user expectations.

Testing Methodologies

1. **Manual Testing**:
    - All features were manually tested during development to ensure they functioned as expected.
    - Tested UI components built with **Tailwind CSS**, backend APIs developed with **Next.js**, and interactions in **TypeScript**.
    - Verified functionality, usability, and responsiveness across different devices and screen sizes.
2. **Component Testing**:
    - Each component (e.g., login form, upload buttons) was tested individually to ensure they performed their specific tasks correctly.
3. **Integration Testing**:
    - Verified interactions between the frontend and backend, such as user authentication, music uploads, and data retrieval.
4. **System Testing**:
    - Simulated end-to-end user workflows (e.g., signing in, uploading content, accessing dashboards) to confirm the system works as a cohesive unit.

4.4.1 Test Cases

Key test cases were designed to cover all major functionalities of the EHPTS system. Below is a table summarizing the test scenarios, inputs, expected outcomes, and actual results?

Table 4:1 Test case

Test Case ID	Scenario	Input	Expected Output	Actual Output	Status
Login					
TC001	Sign in with valid credentials	Username: user123,	Redirect to user-specific	Redirected to /dashboard/overvi	Pass

		Password: password123	dashboard	ew	
TC002	Sign in with invalid credentials	Username: user123, Password: wrongpass	Error: Invalid username or password	Error message displayed	Pass
TC003	User login attempt with deactivated account	User tries to log in when account is deactivated	Error : Your account is deactivated	"Your account is deactivated" message displayed	Pass
Music Upload					
TC004	Upload valid music file	Valid title, metadata, and audio file	Success: Music uploaded successfully	Music saved and listed in the dashboard	Pass
TC005	Upload without metadata	Missing required fields	Error: All fields are required	Error message displayed	Pass
TC006	Upload large audio file	Audio file > 50MB	Error: File size exceeds limit	Error message displayed	Pass
Upgrade Account					
TC007	Fetch subscription plans	Access the **Upgrade Page**	Plans are displayed with details	Plans fetched and displayed successfully	Pass
TC008	Select a valid plan	Click "Choose Plan" for a plan	Redirect to payment page	Redirected to payment page successfully	Pass
TC008	Select a free trial plan	Click "Choose Plan" for a free trial plan	User is notified of free trial	User notified of successful subscription	Pass

			activation without payment.	activation for free trial. No payment redirection occurred.	

4.4.2 Screenshots of Testing

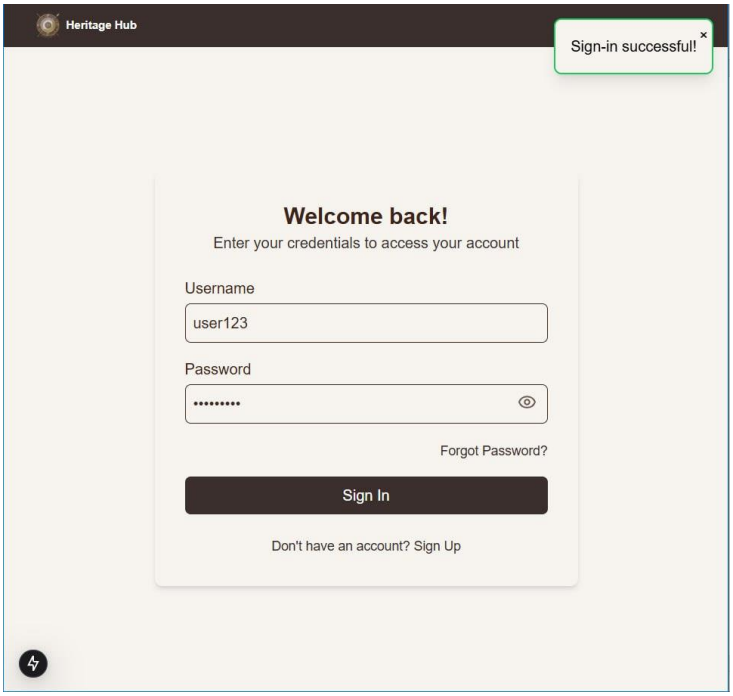


Figure 4.9 Test for Successful Sign in

The **"Sign-in successful!"** message in the top-right corner confirms the user's credentials were valid, and they have logged in successfully. It provides immediate feedback in a green notification box, ensuring clarity for the user.

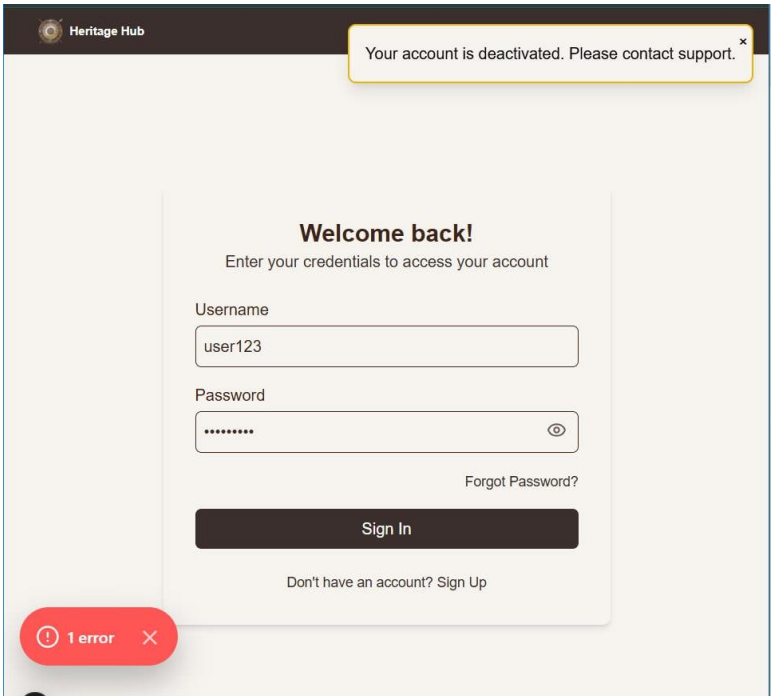


Figure 4.10 Test For Deactivate Account

The notification **"Your account is deactivated. Please contact support"** in the yellow box at the top-right indicates the user cannot log in because their account has been deactivated. The error provides clear instructions to contact support for resolution.

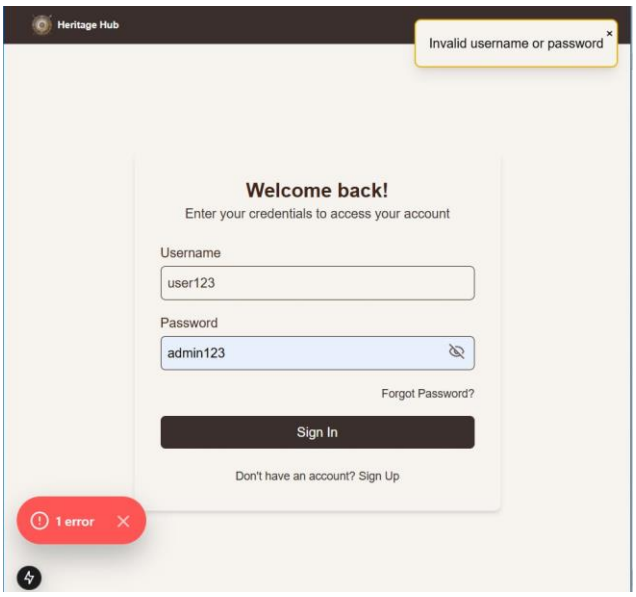


Figure 4.11 Test For Invalid credential

The notification **"Invalid username or password"** in the yellow box at the top-right indicates that the login attempt has failed because the entered username or password does not match the system's records. This issue could occur due to incorrect credentials being input by the user.

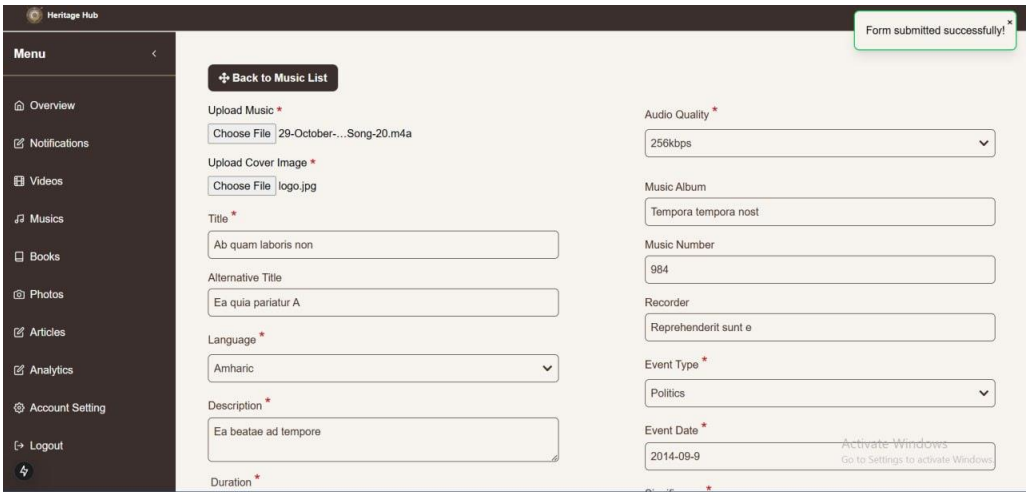


Figure 4.12 Test for successful music upload

The screenshot displays a **Music Upload Form** where a user has successfully filled out all required fields (indicated by the red asterisks) and submitted the form. The green notification at the top-right corner, **"Form submitted successfully!"** confirms that the music details have been uploaded without any issues.

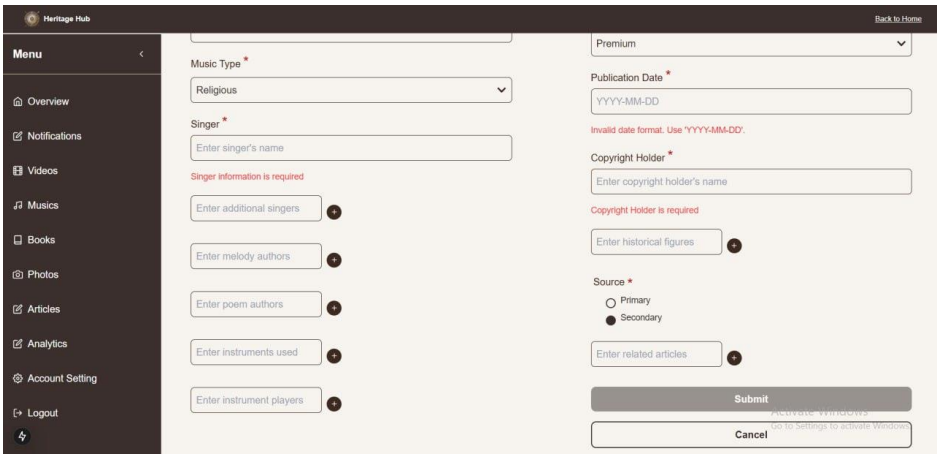


Figure 4.13 Test for missing required fields

The screenshot shows a **Music Upload Form** where some required fields are incomplete or have incorrect inputs, resulting in validation errors.

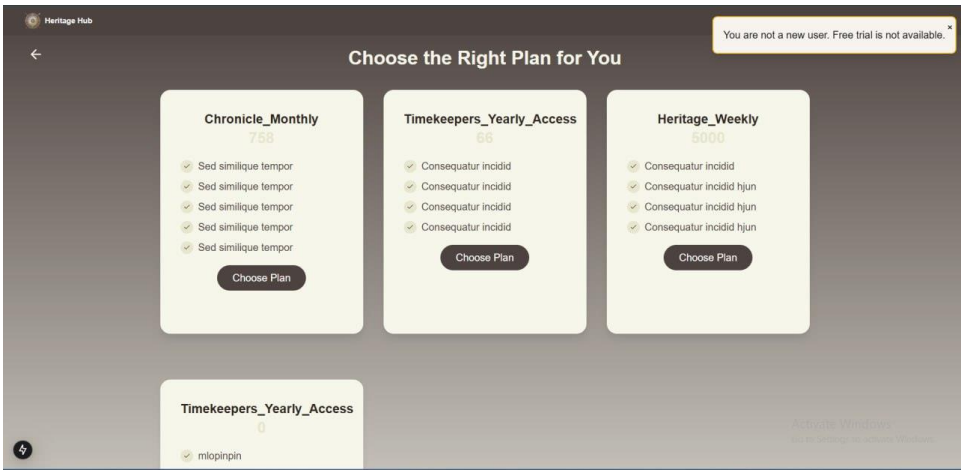


Figure 4.14 Test for double free trial access

The page displays subscription plans for "Heritage Hub." A notification states, "You are not a new user. Free trial is not available," indicating the free trial is only for new users.

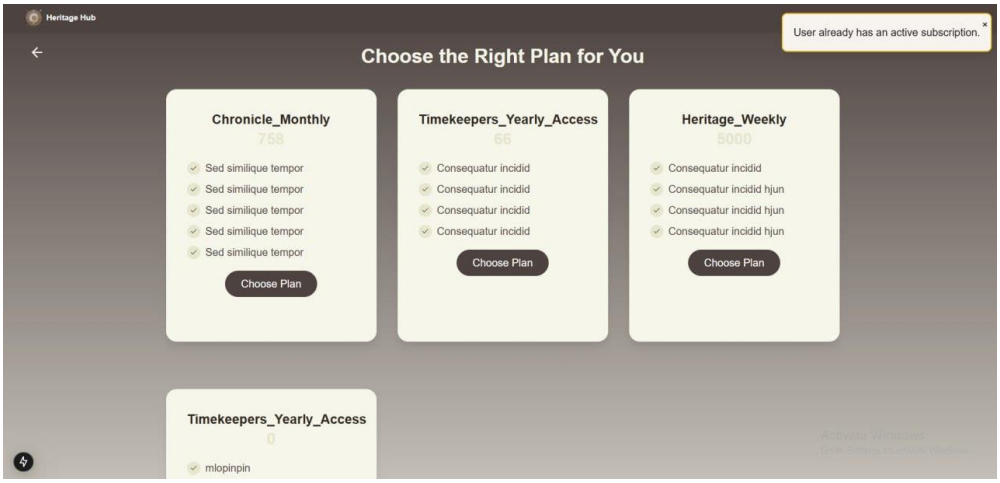


Figure 4.15 Test for double subscription

"User already has an active subscription" means the user cannot switch plans until their current subscription ends or is canceled.

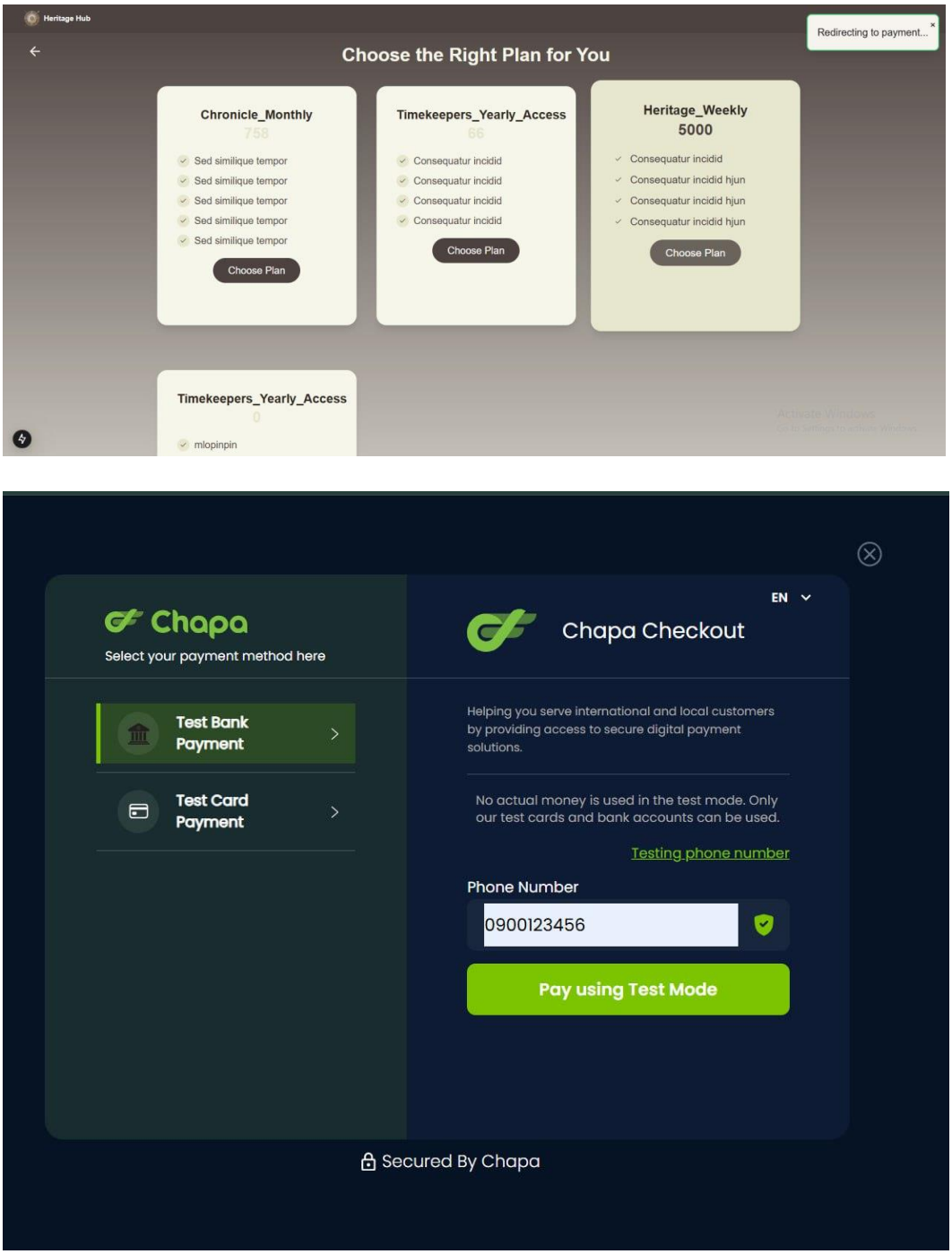


Figure 4.16 Test for successful subscription

The user selects a subscription plan, sees a "Redirecting to payment..." confirmation, and is taken to the Chapa Checkout page to complete the payment using test options.

Chapter 5

5. Conclusion and Recommendations

5.1 Conclusion

The Ethiopian History Preservation and Transmission System (EHPTS) is a significant step toward preserving and sharing Ethiopia's rich cultural heritage in a digital, user-centric format. The platform successfully integrates modern web technologies to provide a seamless and secure environment for users to explore, upload, and manage historical content. By supporting multiple content types, such as books, music, photos, videos, and articles, EHPTS has laid the groundwork for a comprehensive digital archive that empowers contributors and engages the public.

Key features, including token-based authentication, role-based access control, and dynamic content management, ensure the platform's usability and security. The platform's scalability, driven by robust backend infrastructure with PostgreSQL and Prisma ORM, allows for future growth and adaptability. Similarly, the frontend's intuitive design, built with Next.js and Tailwind CSS, ensures accessibility and a positive user experience.

The extensive testing process further validated the platform's reliability, ensuring that features like user login, music upload, and subscription upgrades operate as intended. While EHPTS has achieved its primary goals, there is potential for future enhancements to make the platform more impactful and user-friendly.

5.2 Recommendations and Future Work

To build upon the foundation established by EHPTS and address emerging user needs, the following recommendations are proposed:

5.2.1 Download Options with Uploader Control

- Introduce a download feature that enables users to access selected content offline.
- Provide uploaders with the ability to enable or disable downloads for their uploaded content. This ensures uploaders maintain control over their work and can safeguard sensitive materials.

5.2.2 Feedback Response Mechanism

- Enhance the existing feedback system by enabling two-way communication. Allow platform administrators or content uploaders to respond to user feedback, fostering transparency and engagement.
- Develop tools to categorize and prioritize feedback, enabling the team to address critical issues and implement user-driven improvements efficiently.

5.2.3 Platform Announcements and Administration

- Enable platform administrators to make announcements directly on the platform, such as updates, new features, or events, ensuring users remain informed and engaged.
- Provide administrators with a settings panel to manage and edit platform details, such as terms of service, privacy policies.

5.2.4 Content Type Expansion

- Broaden the scope of supported content types to include historical maps, interviews, artifacts, and multimedia content.
- Introduce enhanced metadata fields for these content types, ensuring they remain searchable and categorized appropriately.

5.2.5 Enhanced Security and User Management

- Regularly assess and enhance platform security to protect sensitive content and user data.
- Implement fine-grained user access controls for premium or sensitive materials to prevent unauthorized viewing or downloads.

5.2.6 Community Features

- Add collaborative features, such as forums or discussion boards, where users can connect and discuss topics related to Ethiopian history.
- Allow users to create groups or communities based on specific interests, fostering collaboration and shared learning.

Reference

[1] A. Mehretu and D. E. Crummey, “Ethiopia | History, Capital, Map, Population, & Facts,” Encyclopædia Britannica. Feb. 06, 2019. Available: https://www.britannica.com/place/Ethiopia.

[2] A. Meckelburg, Dege-MüllerS., and D. Bustorf, *Oral traditions in Ethiopian studies*. Wiesbaden: Harrassowitz Verlag, 2018.

[3] Unesco.org, 2024. https://unesdoc.unesco.org/ark:/48223/pf0000184034 (accessed Apr. 17, 2024).

[4] “ኢትዮጵያን እንወቅ ዳጋ እስጢፋኖስ ገዳም/DISCOVER ETHIOPIA SEASON 4 EP 7,” www.youtube.com. https://www.youtube.com/watch?v=wo4Ghg0A9HY (accessed Apr. 17, 2024).

[5] “Welcome to National Archive and Library Agency | National Archive and Library Agency,” *www.nala.gov.et*. https://www.nala.gov.et/ (accessed Jun. 18, 2024).

[6] “National Museum of Ethiopia,” *Wikipedia*, Oct. 21, 2020. https://en.wikipedia.org/wiki/National_Museum_of_Ethiopia

[7] Wikipedia Contributors, “Microsoft Word,” *Wikipedia*, May 31, 2019. https://en.wikipedia.org/wiki/Microsoft_Word

[8] “Google Forms: Online Form Builder for Business | Google Workspace,” *workspace.google.com*. https://workspace.google.com/products/forms/

[9] “Flowchart Maker & Online Diagram Software,” *Draw.io*, 2019. https://www.draw.io/

[10] Figma, “Figma: the Collaborative Interface Design tool.” *Figma*, 2016. https://www.figma.com/

[11] Microsoft, “Visual Studio Code,” *Visualstudio.com*, 2024. https://code.visualstudio.com/

[12] Vercel, “Next.js by Vercel - The React Framework,” *nextjs.org*, 2024. https://nextjs.org/

[13] tailwindcss, “Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.,” *tailwindcss.com*, 2023. https://tailwindcss.com/

[14] OpenJS Foundation, “Express - Node.js web application framework,” *Expressjs.com*, 2017. https://expressjs.com/

[15] Oracle, “MySQL,” *Mysql.com*, 2000. https://www.mysql.com/

[16] “Git,” *www.git-scm.com*. https://www.git-scm.com/

[17] "Insight Consultants | Lead with Insight," *Get Insight - Insight Consultants*. https://insightconsultants.co/ (accessed Jun. 18, 2024).

[18] DTU students, *Modified Waterfall Model*. Apppm, 2022.

[19] "Object Oriented System Analysis and Design (OOSAD) COMPILED BY: HABTAMU KENO DEPARTMENT OF INFORMATION SYSTEMS 3, June, 2O WOLISO, ETHIOPIA OOSAD Module." Available: http://ndl.ethernet.edu.et/bitstream/123456789/90366/3/Revised%20OOSAD%20Module2020.pdf

[20] "Requirements Analysis," *Jama Software*. https://www.jamasoftware.com/requirements-management-guide/requirements-gathering-and-management-processes/requirements-analysis#:~:text=The%20purpose%20of%20requirements%20analysis

[21] GeeksforGeeks, "Use Case Diagram," *GeeksforGeeks*, Nov. 24, 2023. https://www.geeksforgeeks.org/use-case-diagram/

[22] N. Daly, "What Is a Use Case & How To Write One | Wrike," *www.wrike.com*, Apr. 25, 2022. https://www.wrike.com/blog/what-is-a-use-case/

[23] "UML 2 Sequence Diagrams: An Agile Introduction – The Agile Modeling (AM) Method." https://agilemodeling.com/artifacts/sequencediagram.htm

[24] "When do you need a persistence model? · Vladimir Khorikov," *khorikov.org*. https://khorikov.org/posts/2020-04-20-when-do-you-need-persistence-model/ (accessed Jun. 18, 2024).

[25] K. Chris, "Database Normalization – Normal Forms 1nf 2nf 3nf Table Examples," *freeCodeCamp.org*, Dec. 21, 2022. https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/