FINAL SEMESTER ASSESSMENT (FSA)
B.TECH. (CSE)
VI SEMESTER

UE18CS355 – OBJECT ORIENTED ANALYSIS AND DESIGN WITH SOFTWARE ENGINEERING LABORATORY

PROJECT REPORT

ON

Knowledge Repository

SUBMITTED BY

NAME	SRN
1) Sneha Hegde	PES1201801157
2) Ojashvi Saxena	PES1201801254
3) Abhilash V.	PES1201800238

Team B7
(Exchanged titles with Team B15)

JANUARY – MAY 2021

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RR CAMPUS,

BENGALURU – 560100, KARNATAKA, INDIA

TABLE OF CONTENTS

ABSTRACT

Assimilate, in simple words, is an open-source knowledge repository containing abundant information pertaining to a variety of domains. Though information on the internet is available in a number of blog posts and articles, these usually require user authentication or even some form of subscription. Unlike these, Assimilate is aimed to make information ubiquitous.

The information will be gathered and pieced together by domain experts, whose expertise will be validated by a group of stakeholders. Whereas domain experts and stakeholders require accounts to publish articles, these articles are readily available to general users, who don't require an account to dive in and browse through the vast amounts of knowledge Assimilate has to offer, in a secure and reliable way.

Assimilate has been built using the Django framework, which internally uses a combination of Python, HTML, CSS, JavaScript, and sqlite3.

Use cases implemented-

- Managing Articles (Add, Edit, Delete)
- Search for articles based on the keyword
- Maintain Edit logs

SOFTWARE REQUIREMENTS SPECIFICATION

1. Introduction

1.1 Purpose

This document provides a comprehensive overview and helps analyze and assess the requirements needed to create an online knowledge repository. Assimilate 0.0 is aimed at making information available to everyone from anywhere with just a few clicks. Along with the specific requirements needed for the application's functionality, a brief outlook of some of the features, environments and non-functional aspects such as performance and security is included.

1.2 Intended Audience

Being a necessary guide for developers working on Assimilate 0.0, this document can indeed be referred by any developers, testers and document writers working towards the creation and maintenance of an online knowledge repository. Since Assimilate is open-sourced, any user eager to learn more about what it offers or intending to contribute to it, is free to refer to this document. This section is followed by a brief Description of the product and then the External Interface Requirements. The following half of this document includes System features, Non functional Requirements and finally ends with any other requirements not listed before.

1.3 Product Scope

Assimilate is targeted at users of all age groups who strive to learn or are in need of information relevant to a wide array of domains. Being open-sourced, the primary objective of Assimilate is not monetization but to make information available to everyone, everywhere and at any point in time. The feasibility of such a project of such scale is possible only due to the provision of open source contribution. Users who would like to share knowledge they possess with the world are free to contribute to Assimilate making it by the people and for the people.

1.4 References

IEEE - IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, Computer Society, 1998.

2. Overall Description

2.1 Product Perspective

Assimilate, in simple words, is an open-source knowledge repository containing abundant information pertaining to a variety of domains. Though information on the internet is available in a number of blog posts and articles, these usually require user authentication or even some form of subscription. Unlike these, Assimilate is aimed to make information ubiquitous.

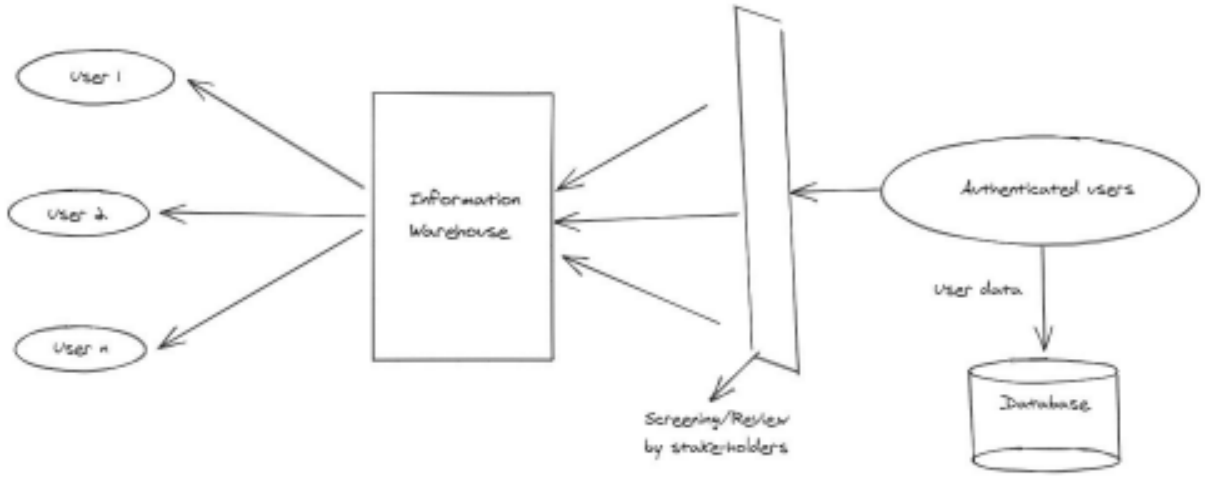


Fig 2.1. Context diagram

2.2 Product Functions

The application in general supports two broad functions:

- ***Provide relevant and meaningful information pertaining to the user intended topic.***

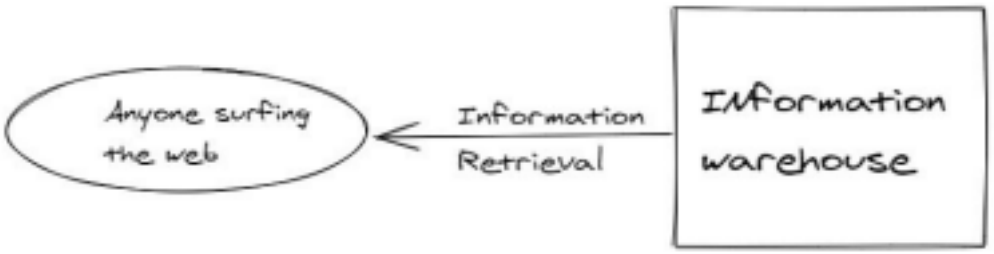


Fig 2.2a Use case diagram for information search

- ***Domain specific experts and authenticated users have a provision to modify/add information.***

Fig 2.2b Use case diagram for information addition

- ***Authenticate users who intend to add information and validate their expertise.***

Fig 2.2a Use case diagram for user validation

- ***Validate user contributions and make sure information available is always clean and reliable.***

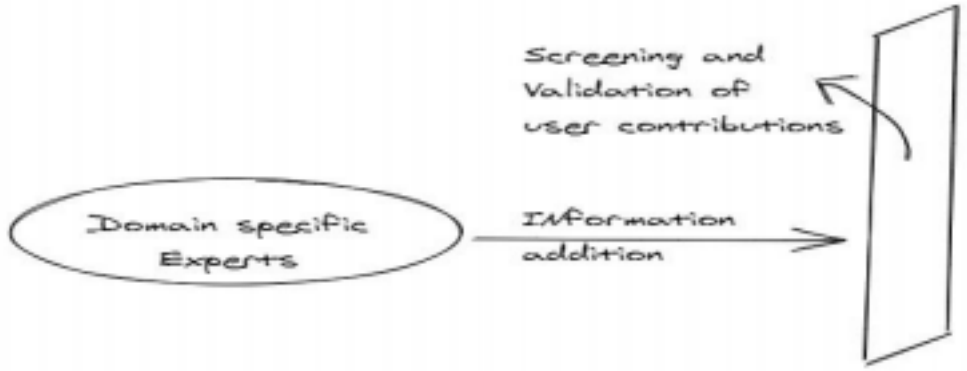


Fig 2.2a Use case diagram for information validation

2.3 User Classes and Characteristics

- ***General public***

  Any user who has access to the internet, is surfing the web and is in need of any information . Being open-sourced and motivated to make information available to everyone, every netizen belongs to this most important user class.

- ***Contributors***

  Users having domain specific knowledge and have something to add to Assimilate belong to this user group.

- ***Stakeholders***

  A contributor having proven his/her credibility can authenticate other contributors' credibility and also validate other contributions relevant to their domain expertise. These users form the stakeholders of Assimilate.

2.4 Operating Environment

Operating environment for Assimilate 0.0 is as listed below:
Operating system: *Windows , linux based or MacOS or any other operating system with web browsing capabilities*
Hardware requirements: *As this is a web application, the hardware requirements will be limited to the browser requirements.*
Software requirements: *As this web application will be hosted on a hosting platform(Heroku, PaaS), the user will not have to install anything more than a simple browser to access the application. On the backend, however, HTML, CSS, vanilla JavaScript, Python, Django, etc. will be required.*

2.5 Design and Implementation Constraints

Creating an enormous warehouse of information which is always available and ubiquitous demands scalable and affordable infrastructure. Migrating to cloud is a huge need for applications of this kind. Moreover, Assimilate 0.0 being decentralized and dependent on a peer to peer model for expansion will require a considerable amount of initial set up time. Information given by contributors can be validated only by other contributors making the information not completely reliable or relevant. Maintenance of the software is the responsibility of all the stakeholders which again involves any verified user on the internet.

2.6 Assumptions and Dependencies

Assimilate 0.0 has no assumptions or dependencies related to software or hardware platforms and tools it is built using. However, the only assumption which is key to the smooth functioning of Assimilate is that a majority of its stakeholders remain devoted to the idea of non-offensive and relevant information being available to the world.

3. External Interface Requirements

3.1 User Interfaces

The application GUI provides menus, toolbars, buttons, panes, containers, grids allowing for easy control by a keyboard and a mouse and also portable devices like smartphones, tablets etc.

3.2 Software Interfaces

3.2.1 Basic requirements

HTML, CSS, and vanilla JavaScript will be used for the frontend, to create the static pages, and dynamically create the pages when a user wants to add some information onto the website.

3.2.2 Python 3.x and Django 3.x

Django and Python will be used for creating the backend which will contain the functionalities like sign up, login, etc. Additional features that Django has to provide such as login with email will also be added. The user will not have to create a separate account, but instead they can login with Gmail, etc.

3.2.3 MySQL and Postgres

Django, by default, uses MySQL database and Heroku uses Postgres. Hence, MySQL will be used during the development phase, and the database will be switched to Postgres once the application is deployed on Heroku.

3.3 Communications Interfaces

Django does not require any special medium to communicate with the frontend. But HTTP protocol will be used to communicate with Heroku.
By default, Django prevents most common security mistakes:

- ***XSS (cross-site scripting) protection*** *— Django template system, by default, escapes variables, unless they are explicitly marked as safe.*
- ***CSRF (cross site request forgery) protection*** *— easy to turn on globally, guarantees that forms (POST requests) are sent from your own site.*
- ***SQL injection protection*** *— Django uses built-in ORM, thus there is no risk of SQL injection (raw queries are possible, but by no means something that a beginner would need to use).*

4. Analysis Models

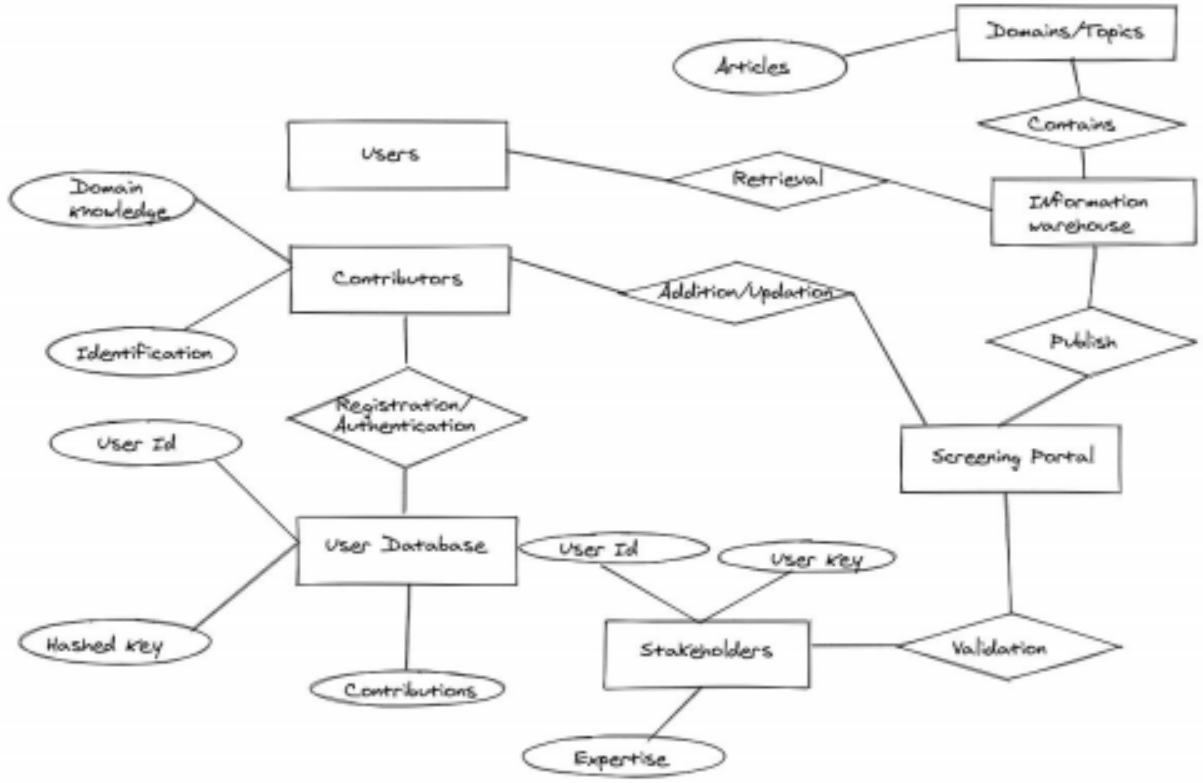


Fig 4. Entity Relationship diagram showing the entire structure of Assimilate

5. System Features

5.1 Information retrieval

5.1.1 Description and Priority

Assimilate being a knowledge repository indicates that capability to read information is a bare minimum functionality. This just involves displaying the already available information when asked for, involving low cost and risk but high priority.

5.1.2 Stimulus/Response Sequences

The user is exposed to a wide number of options to choose from. On selecting the required section, the user can view the information available about the chosen domain with just a click.

5.1.3 Functional Requirements

REQ-1*: *Any user in need of information and has access to a web browser can retrieve information.

5.2 Information addition

5.2.1 Description and Priority

This can be considered the most important feature as only the information added to the repository is available to the users for viewing. The cost and risk associated with this section is almost none as these aspects are highly dependent on user authentication and information validation which are functions which precede Information addition.

5.2.2 Stimulus/Response Sequences

A user needs to first verify and prove his credibility in order to contribute to the knowledge repository. However, having contributed to Assimilate, the information added/modified goes through a screening test , this means the user is not assured that his/her updates are visible to everyone.

5.2.3 Functional Requirements

REQ-1: *Only a user possessing domain knowledge about the topic to which he/she wishes to contribute can add information to Assimilate.*

5.3 User Authentication

5.3.1 Description and Priority

A user who wishes to contribute needs to verify his/her details. This is a measure taken in order to prevent unwanted updates by anonymous users. This function implementation comes with the cost and risk of maintaining the privacy of user data and handling catastrophic failures.

5.3.2 Stimulus/Response Sequences

A user needs to first verify and prove his/her credibility in order to contribute to the knowledge repository. However, having contributed to Assimilate, the information added/modified goes through a screening test , this means the user is not assured that his/her updates are visible to everyone.

5.3.3 Functional Requirements

REQ-1: *Only valid users are allowed to contribute. These users also prove their credibility and knowledge relevant to the domain they choose to contribute to.*

REQ-2: *If a user does not remember his/her credentials, he*

will be assisted with a password recovery mechanism.

REQ-3: *Fault tolerance and quick data recovery to prevent leak of user data or downtime is a key necessity.*

5.4 Information validation

5.4.1 Description and Priority

Having contributed, a user's updates go through a screening test in order to evaluate the credibility of the information. At this stage any offensive or irrelevant information is filtered. This function involves the role of stakeholders hence including their time and energy.

5.4.2 Stimulus/Response Sequences

The user of this function is a stakeholder of Assimilate. He/she does a thorough review of uploaded information and takes necessary actions.

5.4.3 Functional Requirements

REQ-1: *An interface where stakeholders can view contributions, make changes and publish information for the general public.*

6. Other Nonfunctional Requirements

6.1 Performance Requirements

Performance refers to the qualitative or quantitative measure of how well a system reacts in a user workflow. This can be measured in time from a user or system perspective, as well as the amount of resources (CPU, memory, etc.) that software must consume to complete a task. All these resources must be utilised at the server's end to make sure web pages are completely returned within seconds of request. A higher performance can be achieved by reducing the delay in response. Assimilate 0.0 is aimed at making information available to everyone from anywhere and this information has to be retrieved from the database within fraction of seconds. Hence information retrieval time acts as

one of the factors that affects the performance of Assimilate 0.0 . The requests and the responses are both real time entities and the ability to interpret the requests, check the spell and grammatical errors and return the closest possible result affects the performance of the software. The closer the returned result is to the expected result the greater is the performance. The amount of data in the repository determines the chance of the results being returned in real time every time a request is made. All these factors contribute to the performance requirements of Assimilate 0.0 .

6.2 Safety Requirements

The application will address section 508 of the Rehabilitation Act of 1973 where appropriate and reasonable.
If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.
Information exists in various kinds, that which can be accessed or read by anyone in general public and the one that is meant to be accessed by people over a certain age. Use of child safe filters are seen in web browsers, a safe-search filter in default enable mode must be used to ensure that the reader is well aware of the fact that the information that is being accessed is not meant people under a certain age or kids, for example, information

related to violence is rated 16 and above, therefore taking the consent of the user before providing him the knowledge is a must to ensure safety.

6.3 Security Requirements

Automated auditing and logging of events in order to provide an audit trail that can be used to understand the activity and/or to diagnose problems.
Assimilate 0.0 must audit each and every user action that results in database read.
The audit information must contain the following information:

- *User who performed the action*
- *IP address of the computer from which the action is performed*
- *Timestamp of action*
- *Object and data element (i.e. table name and column name)*

The audited and logged information must be accessible in a timely manner to system administrators. Logging must be implemented in all the architectural layers. Assimilate 0.0 being open sourced and providing readable only

knowledge doesn't require any more authentication than specified above. The application will address Title 21 Code of Federal Regulations (21 CFR Part 11) Electronic Records where appropriate and reasonable. https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?fr=11.100 The application will address section 508 of the Rehabilitation Act of 1973 where appropriate and reasonable.

6.4 Software Quality Attributes

Quality is the ability to provide sufficient uptime and availability of software to guarantee a certain level of performance to access and data flow.

- *The system development life cycle must include validity checks for all data fields.*
- *The system must be adequately validated during the system development life cycle .*
- *The system must provide the functionality to generate and manage accurate data records during the development processes.*
- *An intuitive user friendly graphical user interface must be developed.*
- *Web page requests must resolve in a timely manner. Where not otherwise specified, this is on the order of seconds.*

The application will address section 508 of the Rehabilitation Act of 1973 where appropriate and reasonable.
Availability: Assimilate should be available for use anytime and anywhere as long as there is a device that supports web browsers and has an internet connection.

Correctness: Assimilate must provide correct results as response to requests. Adaptability: Assimilate should be capable of running on any screen resolution as long as the operating environment is supported by the device.
Reusability: Assimilate can be reused any number of times.
Reliability: the query results provided by Assimilate 0.0

must be reliable.

6.5 Business Rules

The business model for Assimilate 0.0 deals with Users and Domain Experts.

- *Information can be accessed or retrieved from the data warehouse by anyone surfing on the web without any need for authentication.*

- *Only Domain specific experts and authenticated users have a provision to modify/add information.*
- *Users who intend to add information and validate their expertise need to get themselves authenticated.*
- *User contributions must be validated and made sure that the information available is always clean and reliable.*

7. Other Requirements

Basic version for slower connections:
Assimilate 0.0 needs access to the internet whenever a user wants to query something. Accessing information from the data warehouse needs stable internet connection and not having one results in delayed response, partial response and no response at all. To handle this Assimilate 0.0 must be capable of displaying the returned results in one of two ways, a Standard View that demands stable internet connection and has a proper GUI framework, with images and links associated with the information, and a Basic View that displays information that can be retrieved with less stable internet connections. The Basic View skips graphical contents, for example images, that needs a strong and stable network connection. The Basic View provides textual results to the request that was queried by anyone surfing on the web.

Appendix A: Glossary

SRS: Software Requirements Specification
GUI: Graphical User Interface
IP: Internet Protocol
OS: Operating Systems

Appendix B: Field Layouts

An Excel sheet containing field layouts and properties/attributes and report requirements.

Sample sheet with information required to register the Domain Expert/ User who wants to contribute to the warehouse:

Field	*Length*	*Data Type*	*Description*	*Is Mandatory*
Name	*16*	*Alphabets*	*Name of the user or domain expert*	*Y*
Domain	*16*	*Alphabets*	*Domain the user wishes to register for*	*Y*
Username	*16*	*Alphanumeric*	*Username for secure login and authentication*	*Y*
Password	*16*	*Alphanumeric*	*Password to deny access to unauthorised users.*	*Y*
Confirm Password	*16*	*Alphanumeric*	*confirmation*	*Y*

Appendix C: Requirement Traceability Matrix

Sl. No	Requirement ID	Brief Description of Requirement	Architecture Reference	Design Reference	Code File Reference	Test Case ID	System Test Status (in integrated system)
1.	Assimilate1	A page where all the topics can be viewed and searched for.	Arch1, Arch2	A2, S2	urls.py, views.py, post_list.html	UT2-01 - UT2-09, UT1-08, UT1-09, UT1-13	Pass
2.	Assimilate2	User Login and Authentication.	Arch1,	State2	urls.py,	UT1-01 -	Pass

			Arch2		views.py, register.html, registered.html, login.html	UT1-05, UT1-14	
3.	Assimilate3	An interface for users to manage article information (add, edit, delete)	Arch1, Arch2	A1, S1, State1	urls.py, views.py, post_edit.html, base.html, post_detail.html	UT1-06, UT1-07, UT1-10, UT1-11, UT1-12, UT1-15	Pass
4.	Assimilate4	An interface to view each change made to each article (version history/edit logs)	N/A	A3, S3	urls.py, views.py, edit_logs.html	UT3-01 - UT3-05	Pass
5.	Assimilate5	A way to search for articles, based on some attribute, to make lookup faster	N/A	A2, S2	urls.py, views.py, post_list.html, search_results.html	UT2-01 - UT2-09	Pass

PROJECT PLAN

1: Identify the lifecycle to be followed for the execution of your project and justify why you have chosen the model.

Software development life cycle is indicative of a software's development from inception to finish. For obvious reasons, the selection of a particular model for SDLC, has a direct impact on shaping the success of the software.

Being open source and free, the scope for changes and modifications to a knowledge repository is enormous. With contributors and stakeholders increasing and changing dynamically, any static models like the Waterfall model or the V-shaped model, which demand upfront planning and decision making are not suitable here. We propose an Iterative and incremental model for the Project development life cycle simply because the initial costs and investments are bare minimum and product delivery is much faster. The choice of this model for the development life cycle makes sense for a lot of other reasons. Using this model, helps us prioritize our initial requirements and goals, this makes a working knowledge repository available to the users at the earliest. Being open to contributions, we can collect feedback from users, work on aspects which need improvement and roll out corrections dynamically. Since, the product will be developed through repeated cycles (Iterative) and smaller portions at a time (Incremental), we can not only add value to the product early in the life cycle, but also make use of the customer feedback loop and come up with a robust procedure for problem detection and solution. These advantages of using the Iterative and incremental model clearly help an open-source knowledge repository like Assimilate not only reach a large user base, but also incorporate user needs in the best way possible.

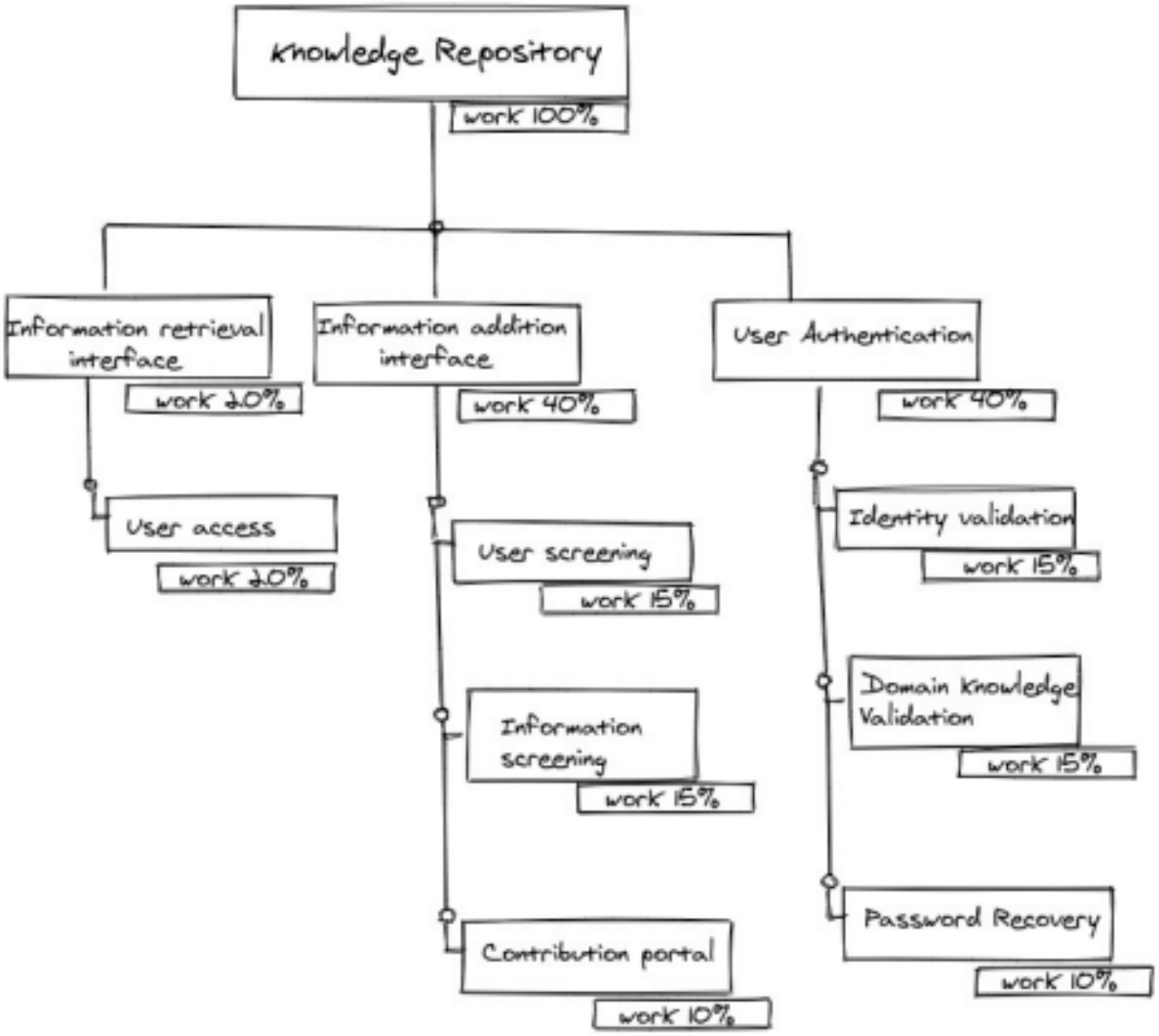
2: Identify the tools which you want to use at different phases of SDLC like planning tool, design tool, version control, development tool, bug tracking and testing tool.

Software development lifecycle tools -

From project planning and source code management to CI/CD and monitoring, GitLab is a complete DevOps platform, delivered as a single application.

- Plan - GitLab enables portfolio planning and management through epics, groups (programs) and milestones to organize and track progress. Regardless of your methodology from Waterfall to DevOps, GitLab's simple and flexible approach to planning meets the needs of small teams to large enterprises.
- Design Management - Design Management allows you to upload design assets (wireframes, mockups, etc.) to GitLab issues and keep them stored in one single place, accessed by the Design Management's page within an issue, giving product designers, product managers, and engineers a way to collaborate on designs over one single source of truth.
- Gitlab Version Control - GitLab makes modern Source Code Management easy. Project management, issue tracking, branches, code reviews and continuous integration in one single place. Merge Requests are your launchpad to high frequency integration and good code quality practices.
- Using Gitlab for development - A lot of features such as review, comment, improve, reuse and inner source code makes this one of the best options for development of any kind. Also, file locking to prevent conflicts and a robust Web IDE accelerates development across all platforms.
- Issue tracker - Handling bugs and issues is quite simple with Gitlab. We can create an issue from a template, set a due date, bulk edit issues, import issues, export issues or even configure an external issue tracker such as Bugzilla or Jira.
- End to end testing - This is a strategy used to check whether your application works as expected across the entire software stack and architecture, including integration of all micro-services and components that are supposed to work together. The GitLab QA orchestrator tool, is a black-box testing framework for the API and the UI.

3: Create a Work Breakdown Structure for the entire functionalities of your project in detail.



4: Determine all the deliverables and categorize them as reuse/build components and justify the same.

Components which have to built from scratch -

- User Interface for viewing and retrieving information
- Login page for Identity validation
- Domain knowledge validation portal
- Password recovery page

Components which can be reused -

- User screening portal: We can reuse the domain knowledge validation portal to maintain a uniform user experience.

● Contribution portal: We can again reuse the domain knowledge validation portal of the User screening portal as this also involves addition of information from the user end. ● Information screening portal: The User Interface portal for information retrieval can be reused here as this is where the stakeholders can review contributions.

5: Do a rough estimate of effort required to accomplish each task in terms of person months.

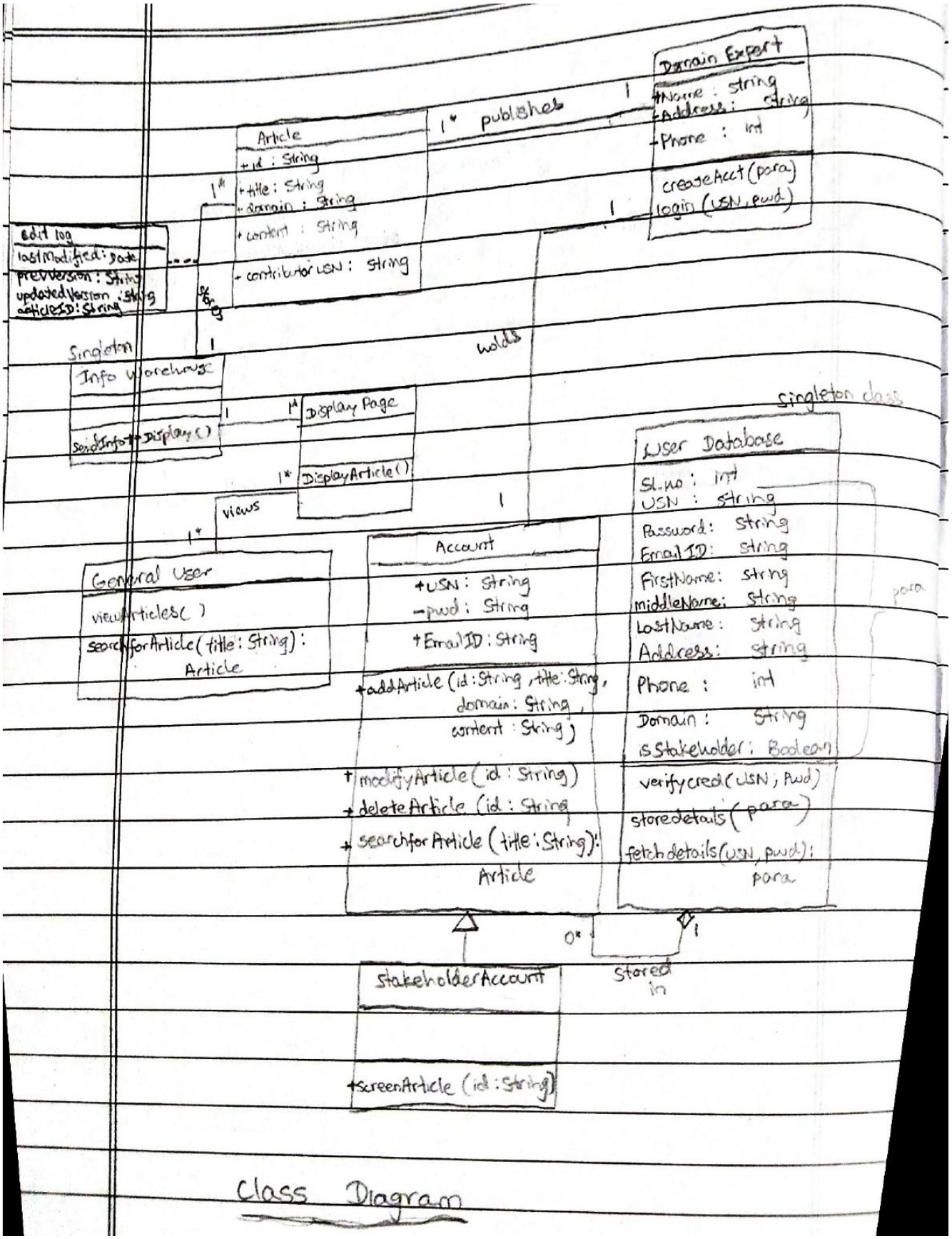
- User Interface for viewing and retrieving information - Typically requires one person month. For a team of two the effort is reduced to half a month.
- Information addition interface - The effort to complete the whole task demands two person months. The same two person team who work on the Information retrieval UI can work on this in the second month.
- User Authentication - This may require one and a half person months and is handled independently by the third team member.
- Deployment and Testing - After one and a half months the application is ready for deployment and the whole team of three can finish deployment and testing in one month.

6: Create the Gantt chart for scheduling the defined tasks.

https://drive.google.com/file/d/1ygxpJkmuG69Dmr4_uAi4VEIwB7JUnp0P/view?usp=sharing

DESIGN DIAGRAMS

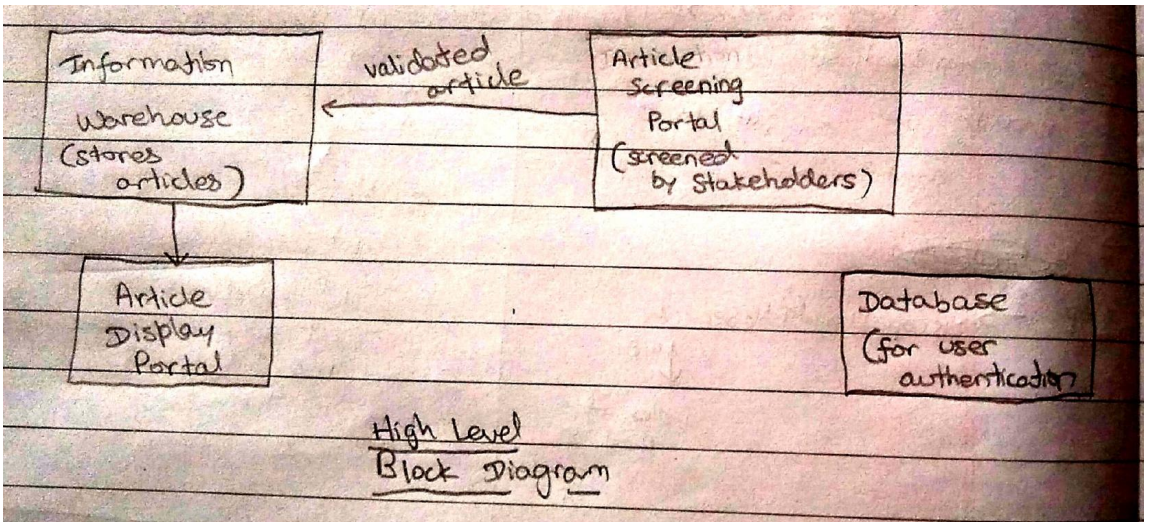
Class Diagram (id = Class1)

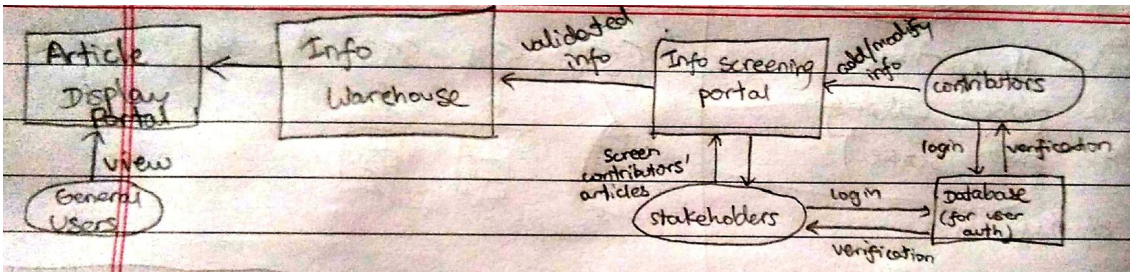


Class Diagram

High Level Block Diagrams

id = Arch1

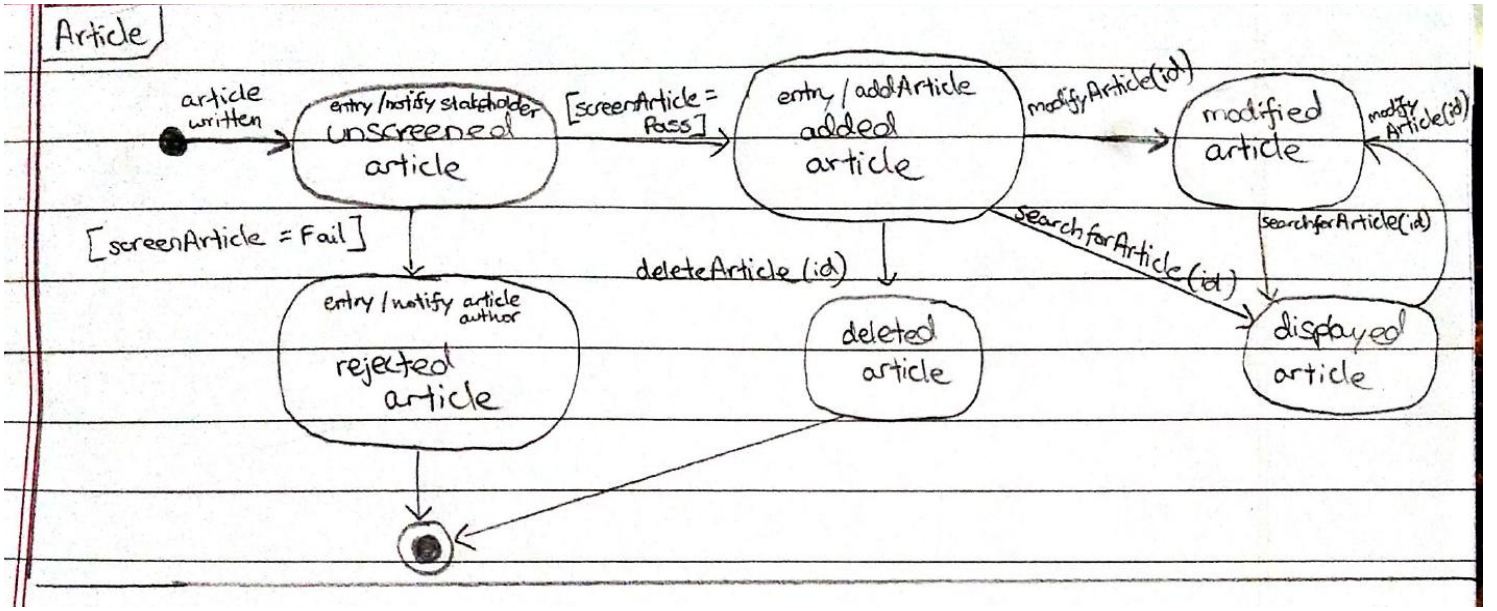




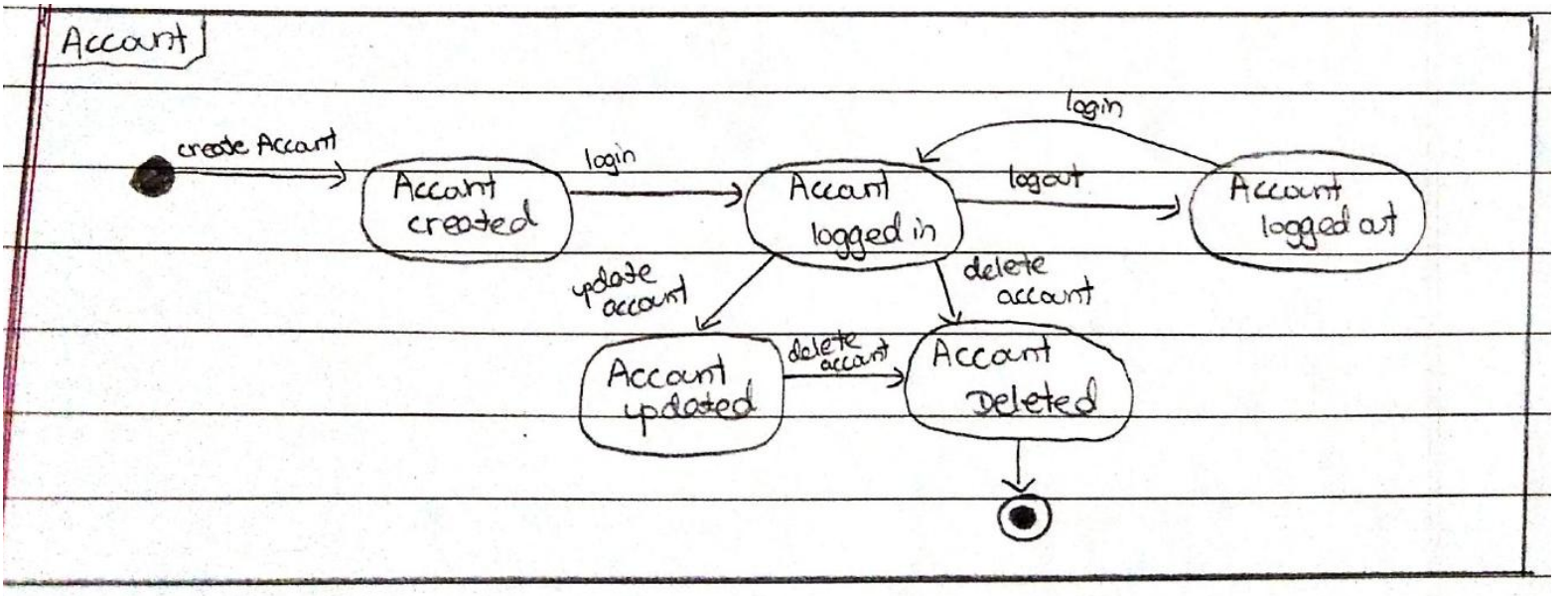
id = Arch2

State Diagrams

id = State1



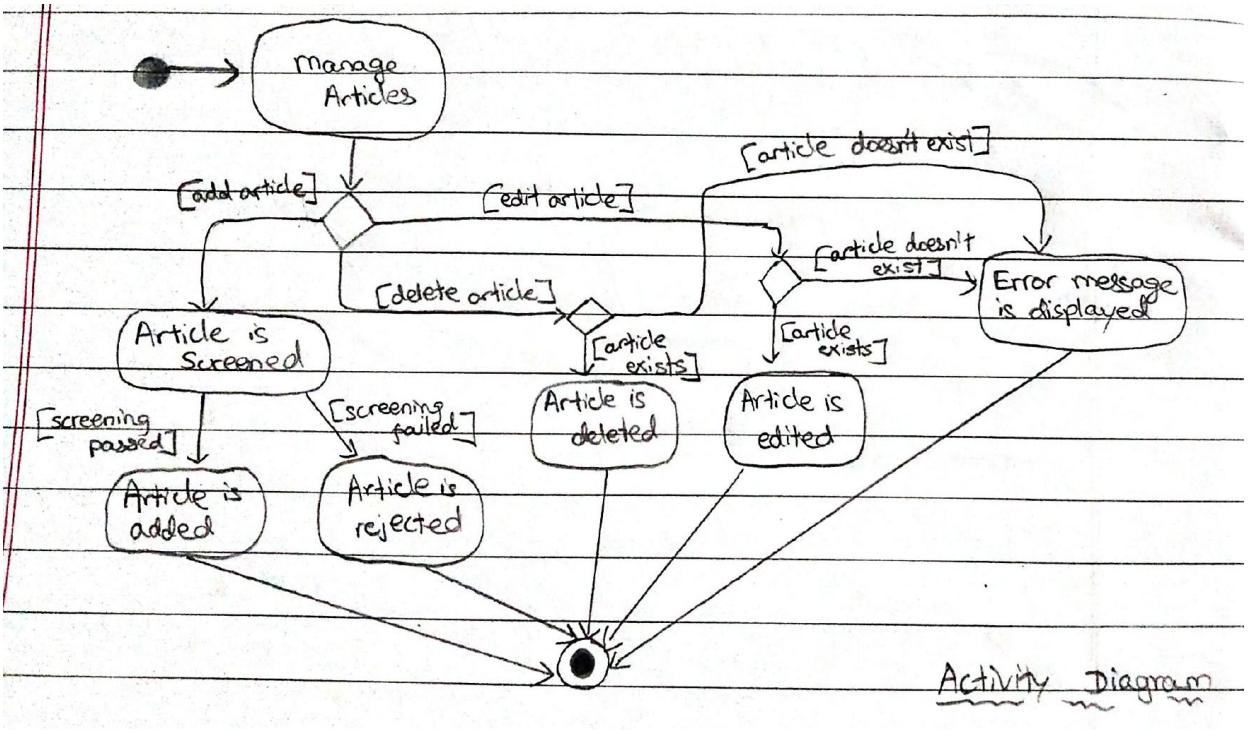
id = State2



Activity Diagrams

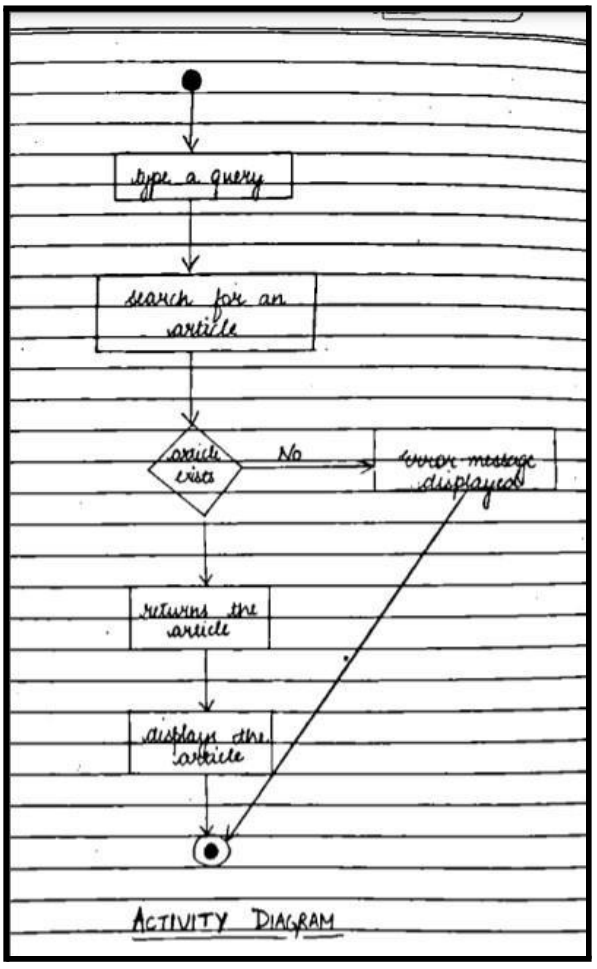
For ‘Manage Articles (add, edit, delete)’ use case

id = A1



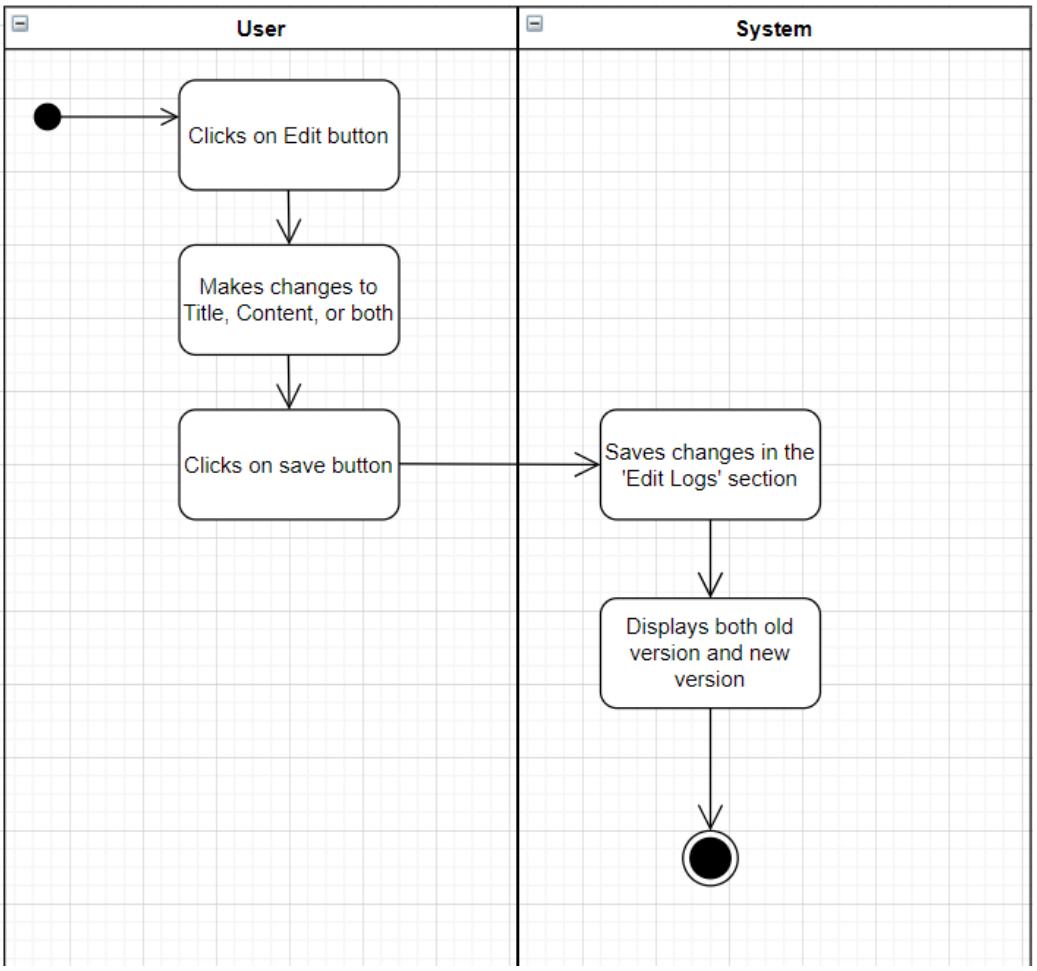
For ‘Search Based on Keyword’ use case

id = A2



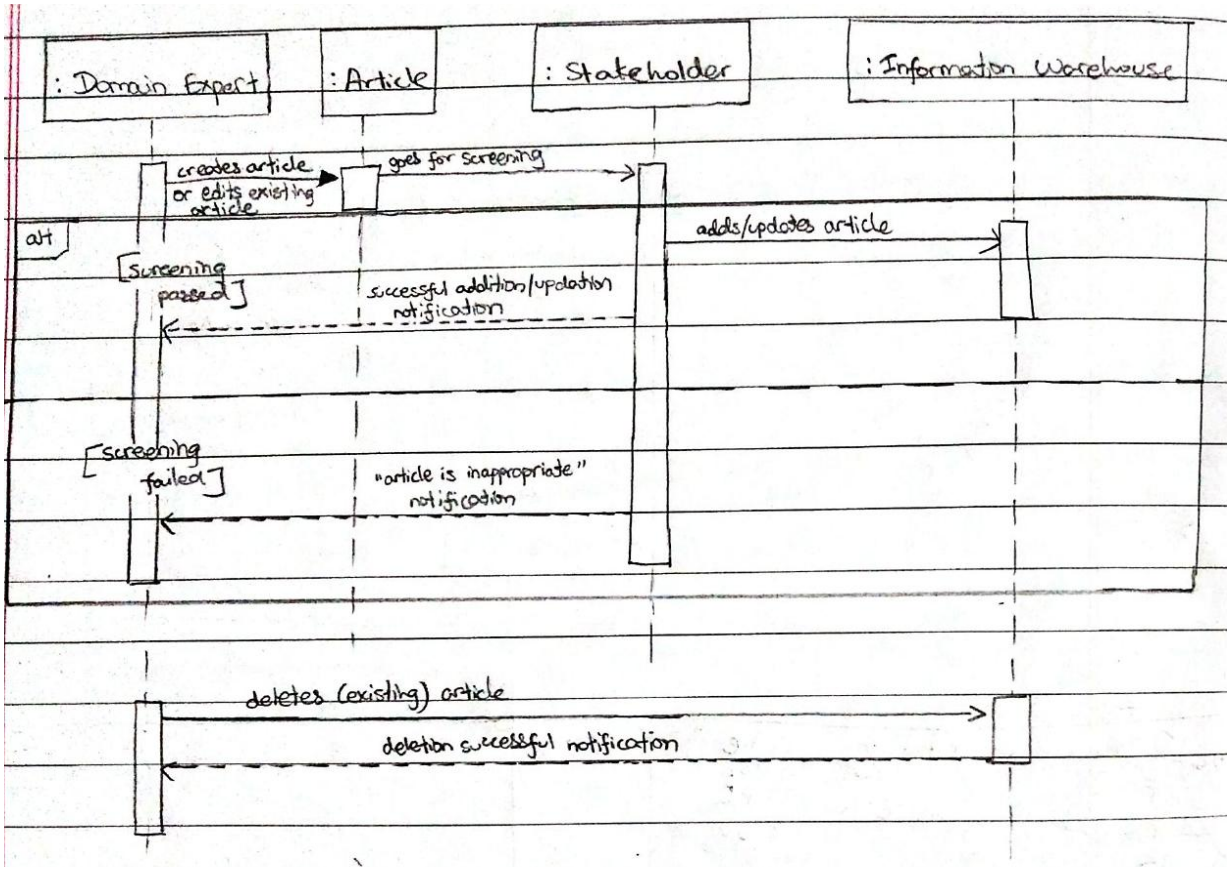
For ‘Edit Logs’ use case

id = A3



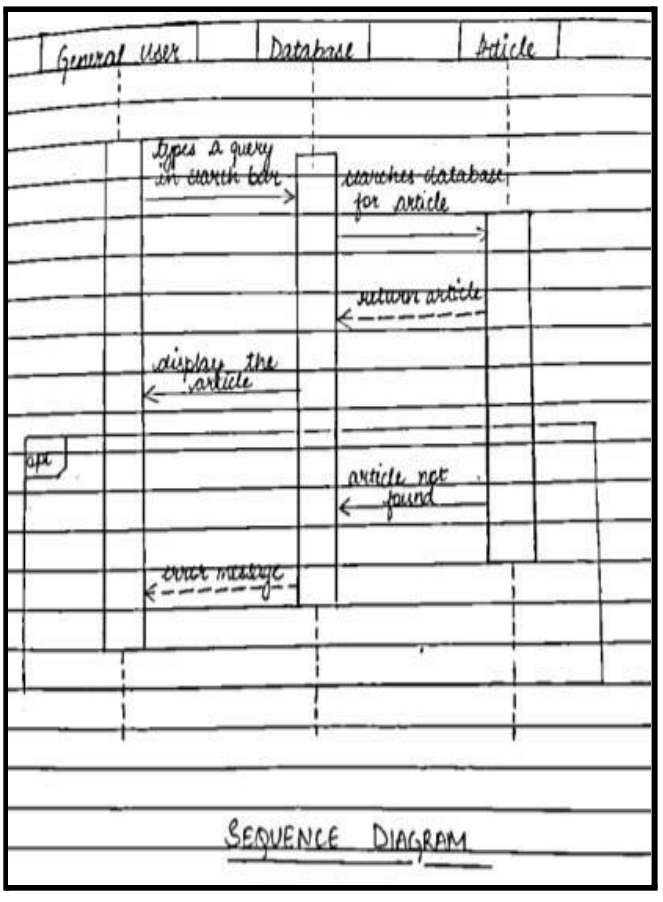
Sequence Diagrams

For ‘Manage Articles (add, edit, delete)’ use case
id = S1



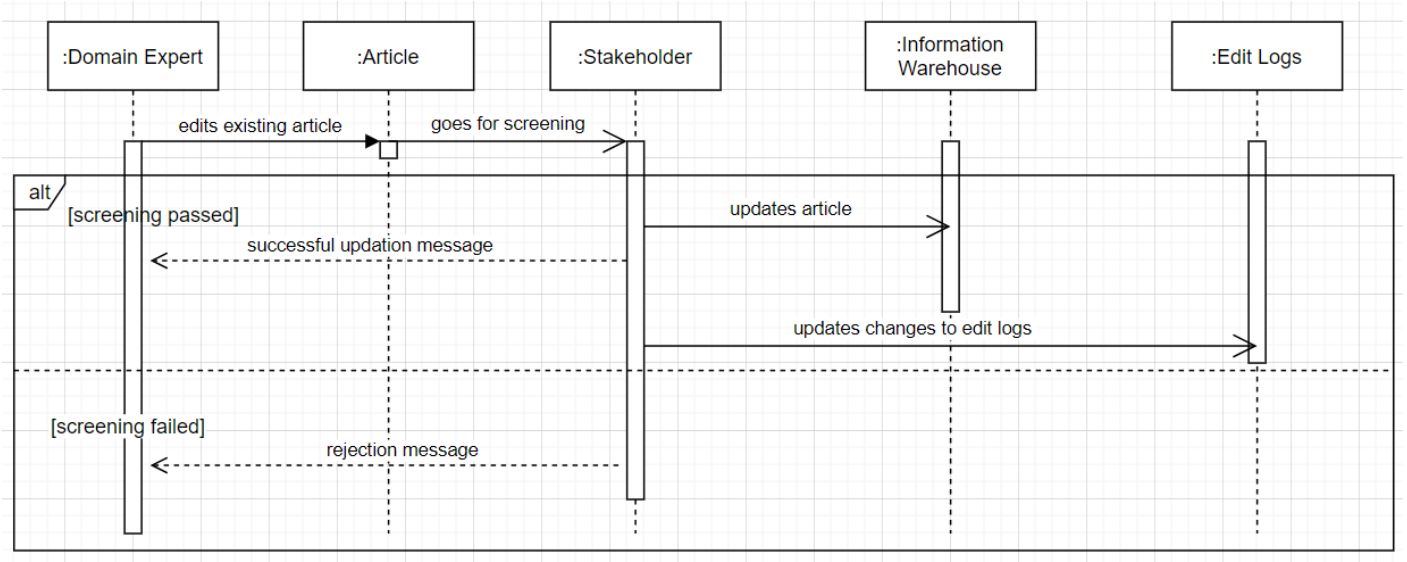
For ‘Search Based on Keyword’ use case

id = S2



For ‘Edit Logs’ use case

id = S3



MODULE DESCRIPTION

1. User Authentication System
   This module implements both signup and login.
   The signup form includes fields such as username, name, domain, is stakeholder (checkbox to indicate whether the account is for a stakeholder or domain expert), password, and password confirmation. If the credentials don't conform to the standards of the fields, an appropriate error message gets displayed, asking the user to fill it out again. Upon successful signup, the system keeps the user signed in, so that they don't need to login immediately after signing up.
   The login module takes the username and password, and if they are valid, it will sign the user in. Else, a "user not found" error gets displayed.

2. Manage Articles (add, edit, delete)
   This module's features are available only to account holders.
   An account holder may add an article by giving it a title, a domain, and some content, all of which are required fields. If any one is not filled out, it won't let the user save it, and will prompt them to fill out that field.
   An account holder may edit any article written by them. The timestamp associated with that article gets updated only if any changes are made to the title or content. If the article is simply saved without making any modifications, the timestamp doesn't change.
   An account holder may also delete any of the articles written by them. Upon doing so, it will disappear from the list of all their articles.

3. Search Based on Keyword
   This module allows users to browse through all the articles. This includes both articles created by them, and by other users.
   This feature can be used by both account holders and non account holders.
   The search may be based on author, title, domain, or content, case insensitively. The resulting articles will be displayed in the order of creation.
   If the search doesn't yield any results, an appropriate message is displayed.
   The search query must be smaller than 50 characters.
   The feature also saves keywords searched for in the past, and will display them as suggestions when the user starts to type them again.

4. Edit Logs
   Each time an article is modified, the changes get saved under the Edit Logs section. This can include a change in title, a change in content, or a change in both. Both the old and new versions get displayed, along with the time of modification.

TEST CASES

For ‘Manage Articles (add, edit, delete)’ use case (with User Authentication System)

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
UT1-01	Sign up feature	To test the sign up feature, by creating a new user	-Django server must be running (python manage.py runserver) -Access to a web browser	-Go to localhost in browser (type localhost:8000 in URL bar) -Click on Sign up button on landing page -Enter credentials (Username, Name, Domain, Is stakeholder, Password, Password Confirmation)	Username = TestUser, Name = test_name, Domain = test domain, Is stakeholder (checkbox) = unchecked, Password = user1pwd123_@, Password Confirmation = user1pwd123_@	It should redirect to the registered page, with the words “SUCCESSFUL REGISTRATION!!! Congratulations, test_name!! You are now a domain expert! :)”	It redirects to the registered page, with the words “SUCCESSFUL REGISTRATION!!! Congratulations, test_name!! You are now a domain expert! :)”	Pass
UT1-02	Sign up feature	To test if proper error message is displayed when invalid credentials are entered during sign up, ie, anything which violates the sign up guidelines given on the sign up page, for example, when Password and Password Confirmation don’t match	-Django server must be running (python manage.py runserver) -Access to a web browser	-Go to localhost in browser (type localhost:8000 in URL bar) -Click on Sign up button on landing page -Enter credentials (Username, Name, Domain, Is stakeholder, Password, Password Confirmation)	Username = User2, Name = Name2, Domain = test domain, Is stakeholder (checkbox) = checked, Password = user2pwd123_@, Password Confirmation = abcdefg123	There should be an error message getting displayed at the top of the page, saying “Oops. Something went wrong. Enter valid details!”	“Oops. Something went wrong. Enter valid details!” gets displayed at the top of the page	Pass
UT1-03	Login feature	To login using credentials of user just created	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up	-Go to localhost in browser (type localhost:8000 in URL bar) -Click on Login button on landing page -Enter credentials (Username, Password)	Username = TestUser, Password = user1pwd123_@	It should redirect to the home page	It gets redirected to the home page	Pass
UT1-04	Login feature	To try logging in with invalid credentials (mismatched/non-existent usn/pwd)	-Django server must be running (python manage.py runserver) -Access to a web browser	-Go to localhost in browser (type localhost:8000 in URL bar) -Click on Login button on landing page -Enter credentials	Username = TestUser, Password = abc123	There should be an error message getting displayed at the top of the page, saying “User not found! Invalid login!”	“User not found! Invalid login!” gets displayed at the top of the page	Page

				(Username, Password)				
UT1-05	Login feature	To try and access certain authorised pages without logging in	-Django server must be running (python manage.py runserver) -Access to a web browser	-in the URL bar, type either of the URLs in the next cell to try and access the home page and add article page, respectively. The same can also be done for the URLs of editing/deleting specific articles	http://localhost:8000/home_page/ http://localhost:8000/post/new/	Because the user has not logged in, and the actions of viewing the account home_page, adding, editing, and deleting articles can only be done by an account holder (domain expert/ stakeholder), it redirects to the login page	It correctly redirects to the login page	Pass
UT1-06	Add article feature	To test and see if articles are successfully getting added	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up -Account logged into through the steps in UT1-03	-In the home page, click on "+ Add an Article" -Enter title, domain, and content for the article and click save	Title = My First Article!, Domain = some domain, Content = This is the content for my first article	It should redirect to the article's detail page, which displays the article title, the author (usn of the author), the domain, date/time of creation, and the content, along with buttons to edit and delete the article	It redirects to the detail page with everything in the expected results being displayed	Pass
UT1-07	Add article feature	To try and save a new article without entering title or content	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up -Account logged into through the steps in UT1-03	-In the home page, click on "+ Add an Article" -Don't enter title or content (or both) and click save	Title = Second Article, and leave Content blank Or Content = This is some content, And leave Title blank Or Leave both fields blank	An error should be prompted at the field(s) left blank, saying "Please fill out this field.", And it shouldn't get saved until they are filled	The correct error message gets displayed	Pass
UT1-08	Articles listing feature	To check if correct articles are being displayed under My Articles	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up -Account logged into through the steps in UT1-03	-In the home page, click on the My Articles button	N/A	Only the articles created by this user must be displayed in the order of creation. Upon clicking on any of these titles, all the details (article title, the author (usn of the author), domain, date/time of	It displays all the correct details	Pass

						last modification, and the content, along with buttons to edit and delete the article) should be displayed		
UT1-09	Articles listing feature	To check if correct articles are being displayed under Browse	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up	-In the home page, click on the Browse button (if logged in) -In the landing page, click on the Browse button (if not logged in)	N/A	All articles, irrespective of author, get displayed here. When any of the titles are clicked,the details are displayed, and the option to edit/delete them are only visible if the current user is logged in as the author of that article (Basically, one can edit/delete only one's own articles, and not the articles of any other user)	It is correctly displaying everything as described in the previous cell	Pass
UT1-10	Edit article feature	To edit an existing article, and see if it gets updated	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up -Account logged into through the steps in UT1-03	-In the home page, click on the My Articles button (the titles of all the articles created by this user get displayed) -Click on the title of any of the articles (ex- the one created in UT1-06) -When it gets redirected to the article's display page, click on the edit button -Change the title, content, or both and click save	Title = My First Article! (updated), Content remains the same	It should redirect to the article's detail page, which displays the article title, the author (usn of the author), domain, date/time of updation/ editing, and the content, along with buttons to edit and delete the article. The updated Title/Content should be displayed, not the old version	It redirects to the detail page with everything in the expected results being displayed	Pass
UT1-11	Edit article feature	To test out the timestamp modification during article editing	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up	-In the home page, click on the My Articles button (the titles of all the articles created by this user get displayed) -Click on the title of any of the articles	N/A	It should redirect to the article's detail page, which displays the article title, the author (usn of the author), domain, date/time, and the content,	All details are displayed correctly, and the timestamp does not get updated	Pass

			-Account logged into through the steps in UT1-03	(ex- the one created in UT1-05) -When it gets redirected to the article's display page, click on the edit button -DON'T change anything (make sure the title and content remain the same) and click save		along with buttons to edit and delete the article. The timestamp shouldn't get updated to the current time, as there were no modifications in the title/content. The timestamp should continue to be what it was before executing this test case (it only gets updated when there's been a change in the title or content)		
UT1-12	Delete article feature	To check if articles get deleted	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up -Account logged into through the steps in UT1-03	-In the home page, click on the My Articles button (the titles of all the articles created by this user get displayed) -Click on the title of any of the articles (ex- the one created in UT1-06) -When it gets redirected to the article's display page, click on the delete button	N/A	It should get redirected to the My Articles page, and the article just deleted should not show up in the list	It correctly redirects, and the deleted article is not being displayed	Pass
UT1-13	Article listing feature	Upon addition/ updation/ deletion, to check if the change gets reflected in the list of articles	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up -Account logged into through the steps in UT1-03	-Add/ Edit/ Delete articles using steps in above test cases -Click on Browse/ My Articles on home page.	Steps similar to above test cases for adding, editing, deleting	Upon clicking on Browse/ My Articles, the change should be reflected in the list	It gets reflected	Pass
UT1-14	Logout feature	To check if the logout feature works correctly	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up	-In any of the pages after logging in, click the Logout button	N/A	It should redirect to the landing page	Gets correctly redirected	Pass

			-Account logged into through the steps in UT1-03					
UT1-15	Cascade feature of User deletion	To check if when a user is deleted, their posts get deleted, too	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up	-Log into the admin site using the URL in the next cell, and enter superuser credentials -Navigate to Users, and delete a user (ex- User1) (by selecting the checkbox next to the usn, and clicking on the delete action)	http://localhost:8000/admin/login/?next=/admin/	The user should get deleted, and in the app, their posts should no longer be displayed	User1's posts are not longer in the list	Pass

For 'Search Based on Keyword' use case

Test Case ID	***Name of Module***	***Test case description***	***Test steps***	***Test data***	***Expected result***	***Actual Result***	***Test Result***
UT2-01	Searching an article	Search for an article present in the database	1.Open the website 2.Search for the query 3.Click Search	Search for a query	Should display the article	Displays the article	Pass
UT2-02	Query range < 50	Testing the maximum length constraints	1.Open the website 2.Search for the query 3.Click Search	Input more than 50 characters in the search bar	An error message should be displayed	An error message is displayed	Pass
UT2-03	Blanked search bar is submitted	Does search work if no keywords are	1.Open the website 2.Click on Search button	Without typing a query click on search	Search shouldn't go through. It should	Search doesn't go through. Correct	Pass

		entered?			say “Please fill out this field.”	error message displayed	
UT2-04	Case sensitivity	Test that search works for capital letters	1.Open the website 2.Search for the query 3.Click Search	Searching for a query in all caps	Matched query should be displayed	Matched query is displayed	Pass
UT2-05	Special characters and number sensitivity	Does search work for numbers & special characters	1.Open the website 2.Input numbers 3.Click Search	Input numbers or special characters in the search bar	Respective article should be displayed else displays appropriate error message	Respective article is displayed else displays appropriate error message	Pass
UT2-06	Content search	Searches for words inside a document	1.Open the website 2.Search for the query 3.Click Search	Input a query	Matched article must be displayed in the results in order of their creation	Matched article is displayed in the results in order of their creation	Pass
UT2-07	Maintains history	Does the search box keep the history of previously searched keywords?	1.Open the website 2.Click on the Search bar	Start typing something that was previously searched	History should be saved, and the query should appear as a suggestion	Query appears as a suggestion	Pass
UT2-08	Ordering of resulted articles	In which order the search result is organized	1.Open the website 2.Search for the query 3.Click Search	Input a query that is featured in multiple articles	Query should be displayed in the order they were created	Queries are displayed in the order of their creation	Pass
UT2-0	Auto	Does the	1.Open the	Input a	Relevant	No	Fail

9	suggestion availability	website have an auto suggest feature	website 2.Search for the query 3.Click Search	query	Suggestions should be displayed	suggestions are displayed, other than history	

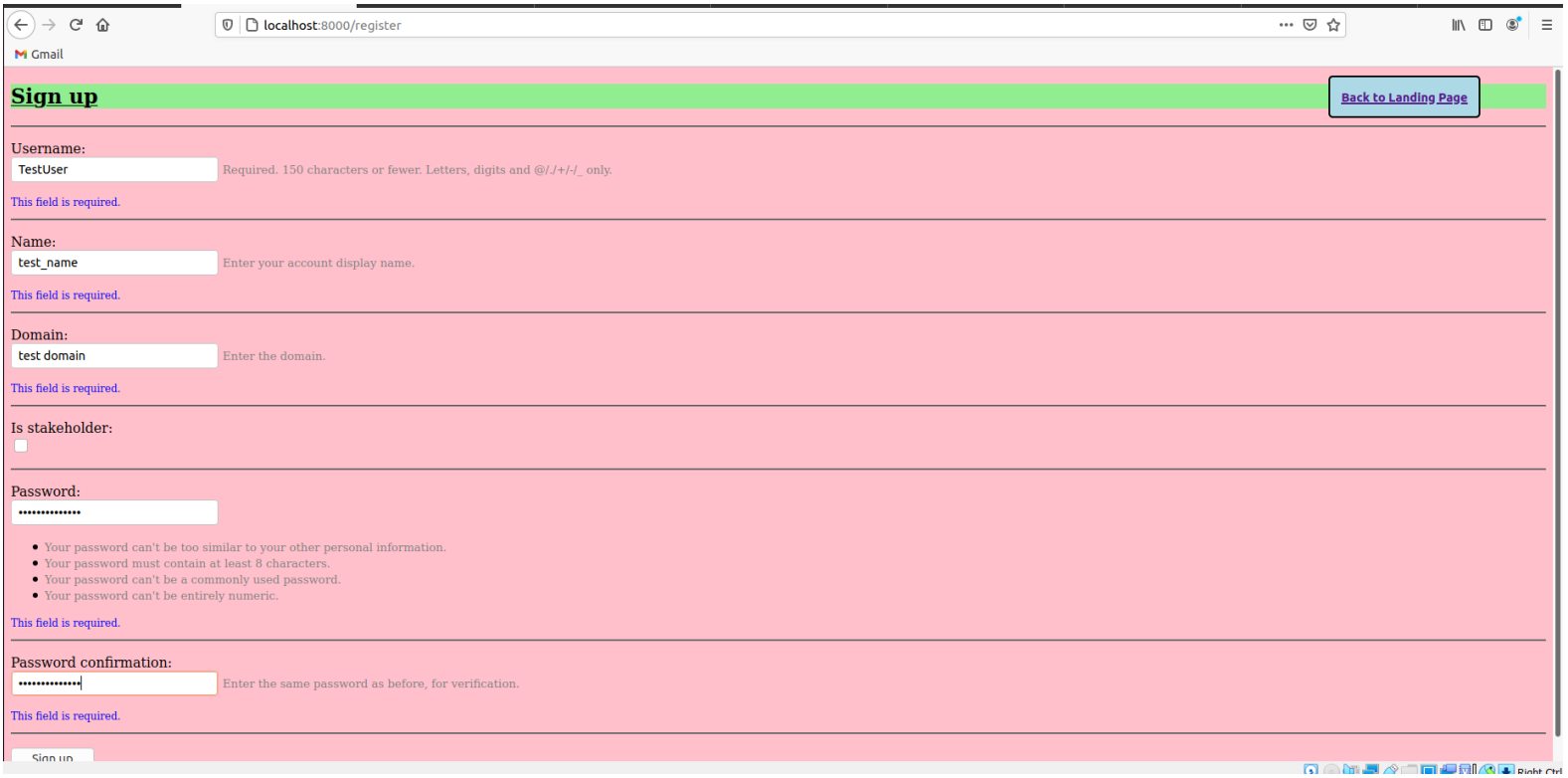
For ‘Edit Logs’ use case

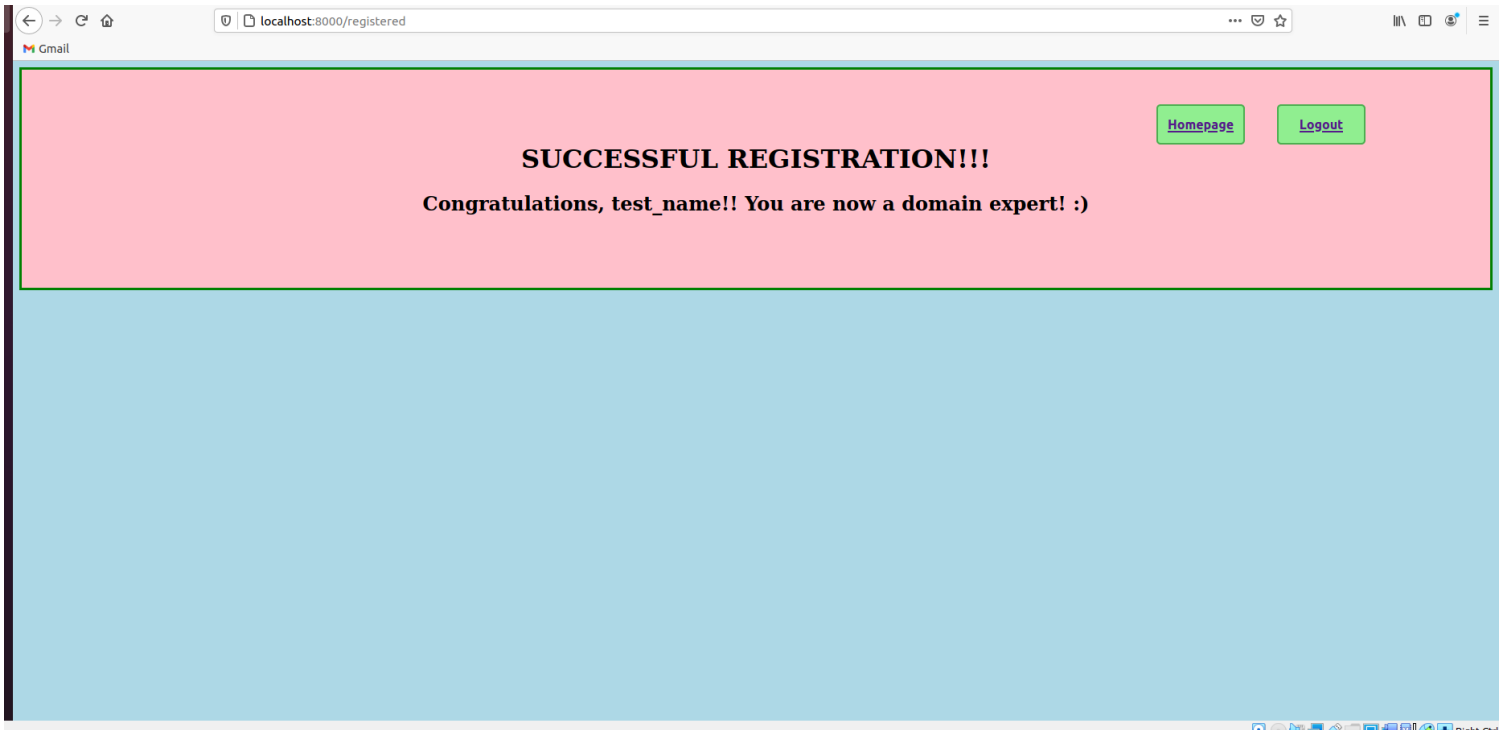
Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
UT3-01	Edit Logs	Change only title, see if it gets added to Edit logs	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up -Account logged into through the steps in UT1-03 -Add an article with the title “title” and content “some content”	-Edit the article. Change the title to “title updated” Refer to UT1-10	Title = “title updated”	-The changes should get reflected. -In Edit Logs, there should be an entry for this article, showing both old and new titles, and “no change in content”, with the modification timestamp	Everything gets displayed as expected	Pass
UT3-02	Edit Logs	Change only content, see if it gets added to Edit logs	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up -Account logged into through the steps in UT1-03	-Edit the previously created article. Change the content from “some content” to “content updated” Refer to UT1-10	Content = “content updated”	-The changes should get reflected. -In Edit Logs, there should be an entry for this article, showing both old and new contents, and the title displayed as it is, with the modification timestamp	Everything gets displayed as expected	Pass
UT3-03	Edit Logs	Change both title and content, see if it gets added to Edit logs	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up -Account logged into through the steps in UT1-03	-Edit the previously created article. Change the content from “content updated” to “content updated again”, and the title from “title updated” to “title updated again” Refer to UT1-10	Title = “title updated again” Content = “content updated again”	-The changes should get reflected. -In Edit Logs, there should be an entry for this article, showing both old and new contents, and old and new titles, with the modification timestamp	Everything gets displayed as expected	Pass

UT3-04	Edit Logs	Change nothing, see if it gets added to Edit logs	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up -Account logged into through the steps in UT1-03	-Go to the previously created article, click on the edit button -Don't change anything -Click on the save button	N/A	There should be no entry this time in Edit Logs	There is no new entry	Pass
UT3-05	Edit Logs	Check the order in which the Edit Logs entries appear	-Django server must be running (python manage.py runserver) -Access to a web browser -A user account created previously through sign up -Account logged into through the steps in UT1-03	-Go to Edit Logs	N/A	The entries should appear in decreasing order of modification time (article modified last should appear at top)	The order is as expected	Pass

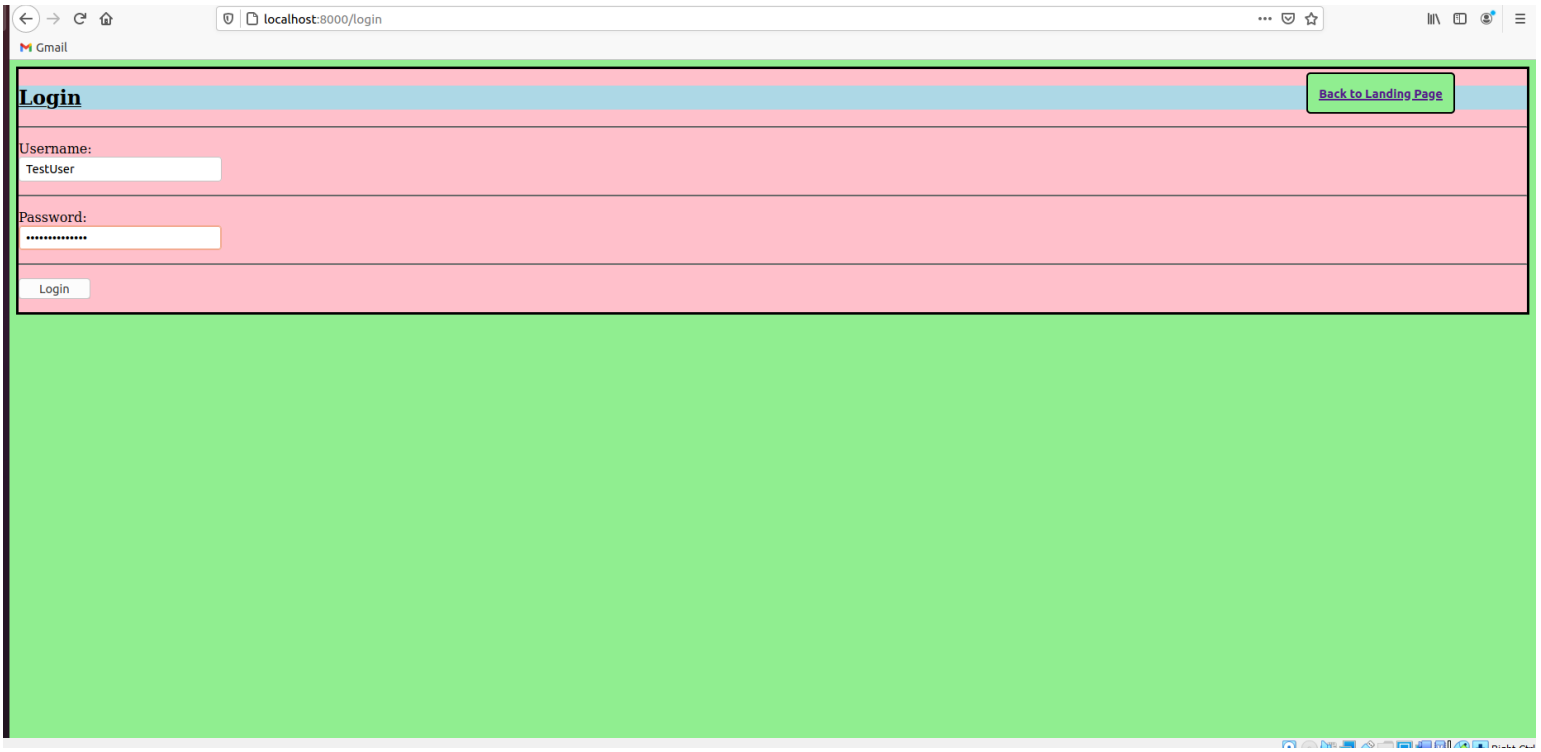
SCREENSHOTS OF OUTPUT

UT1-01

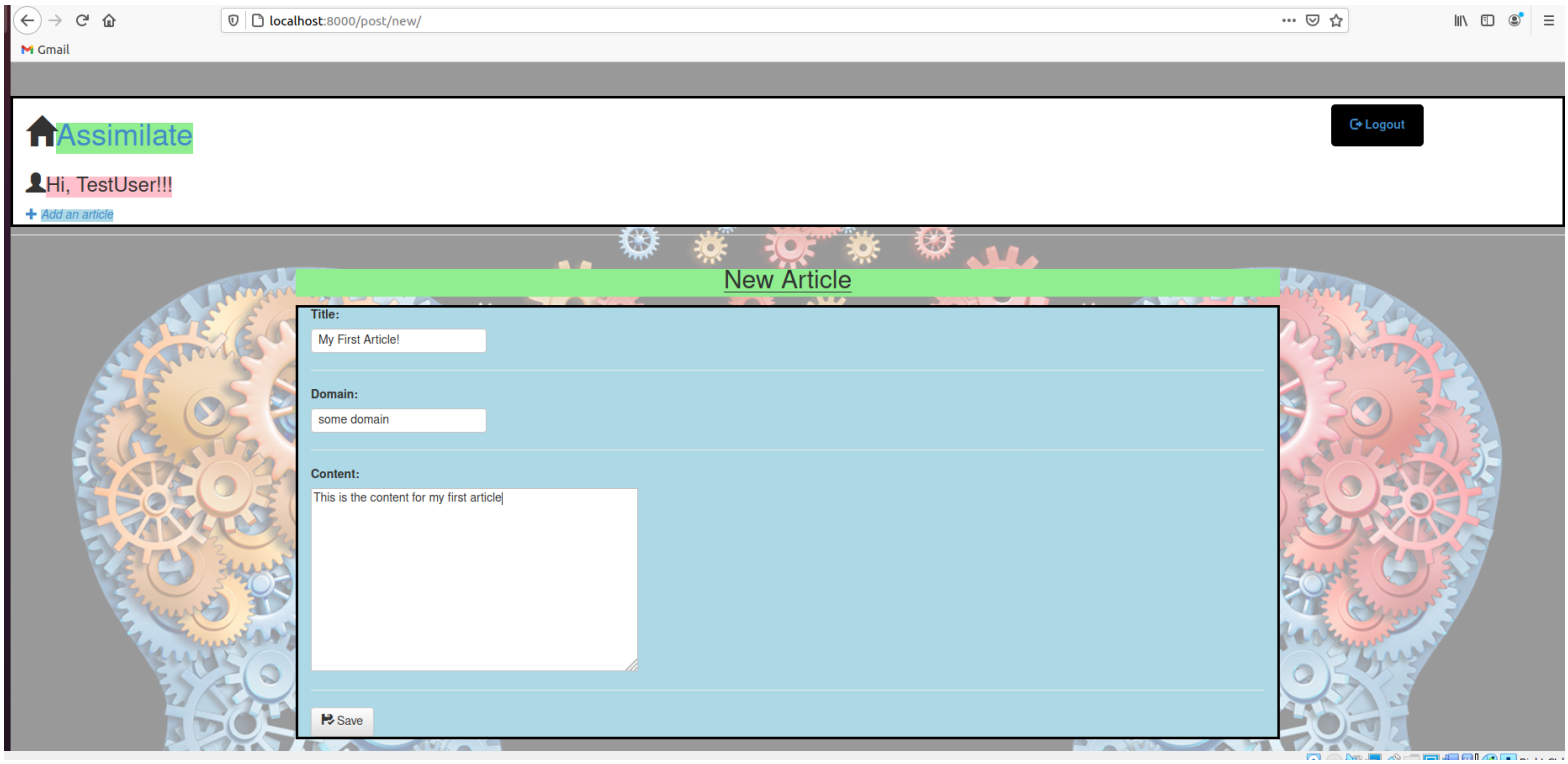


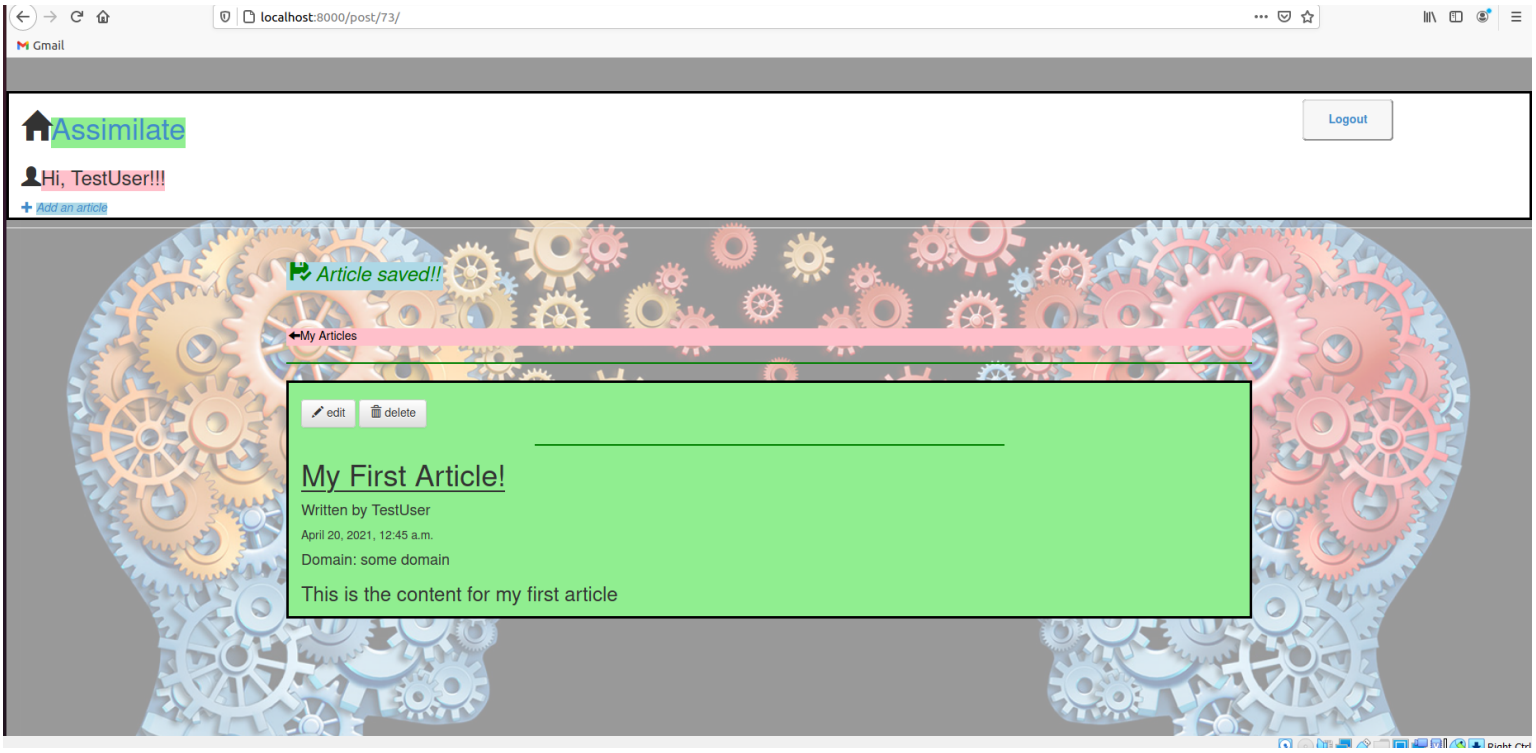


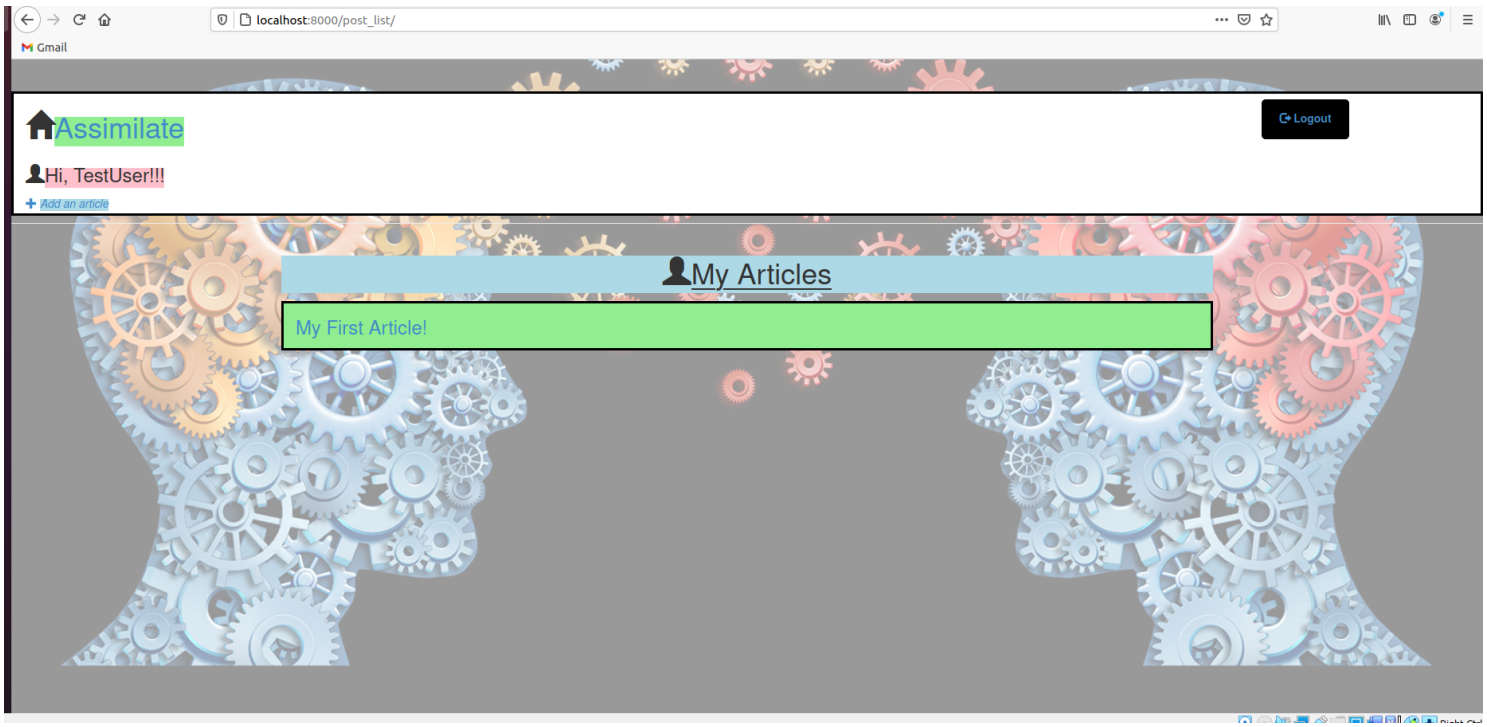
UT1-03


localhost:8000/login
Gmail
Login
Back to Landing Page
Username:
TestUser
Password:
Login

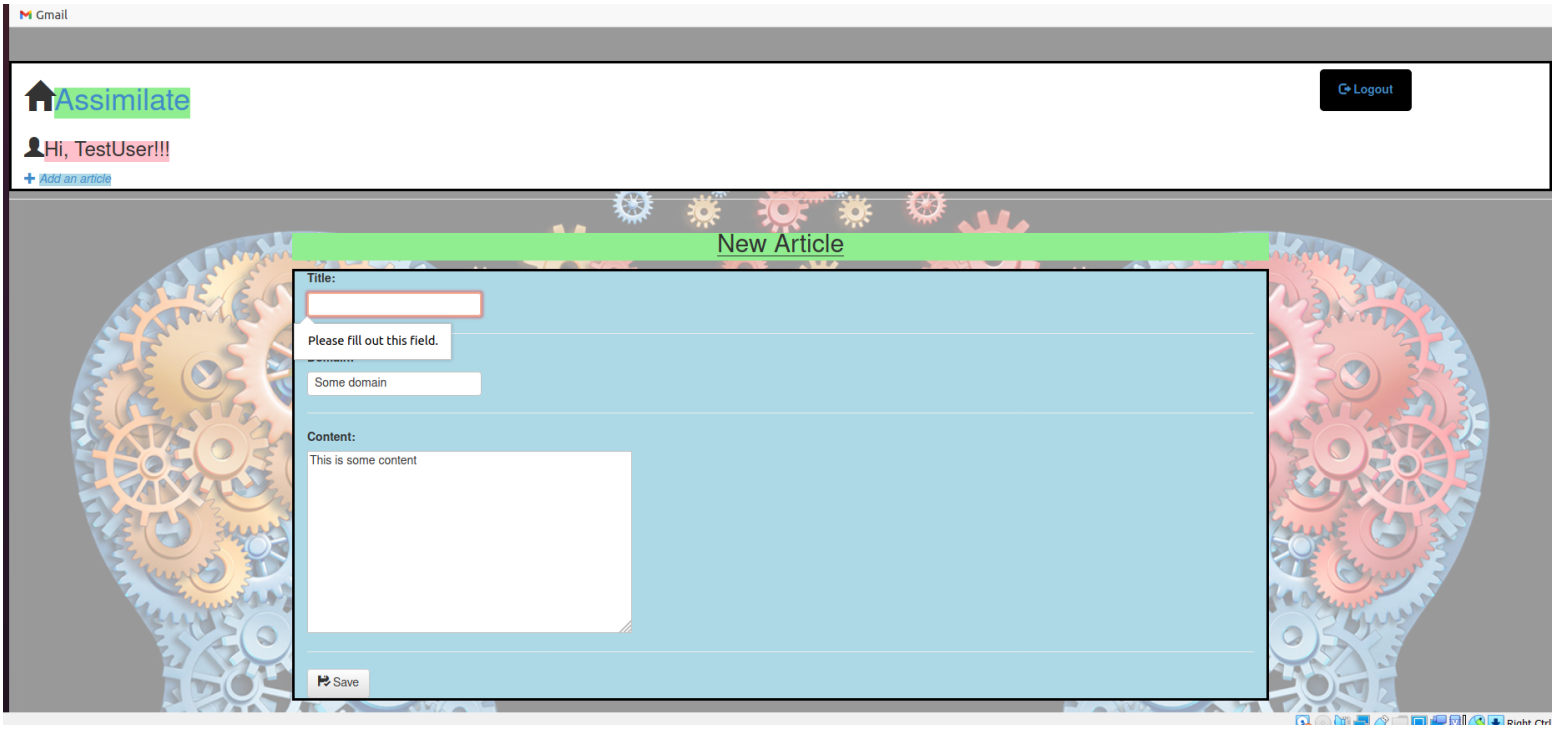
UT1-06


localhost:8000/post/new/
Gmail
Assimilate
Logout
Hi, TestUser!!!
Add an article
New Article
Title:
My First Article!
Domain:
some domain
Content:
This is the content for my first article
Save

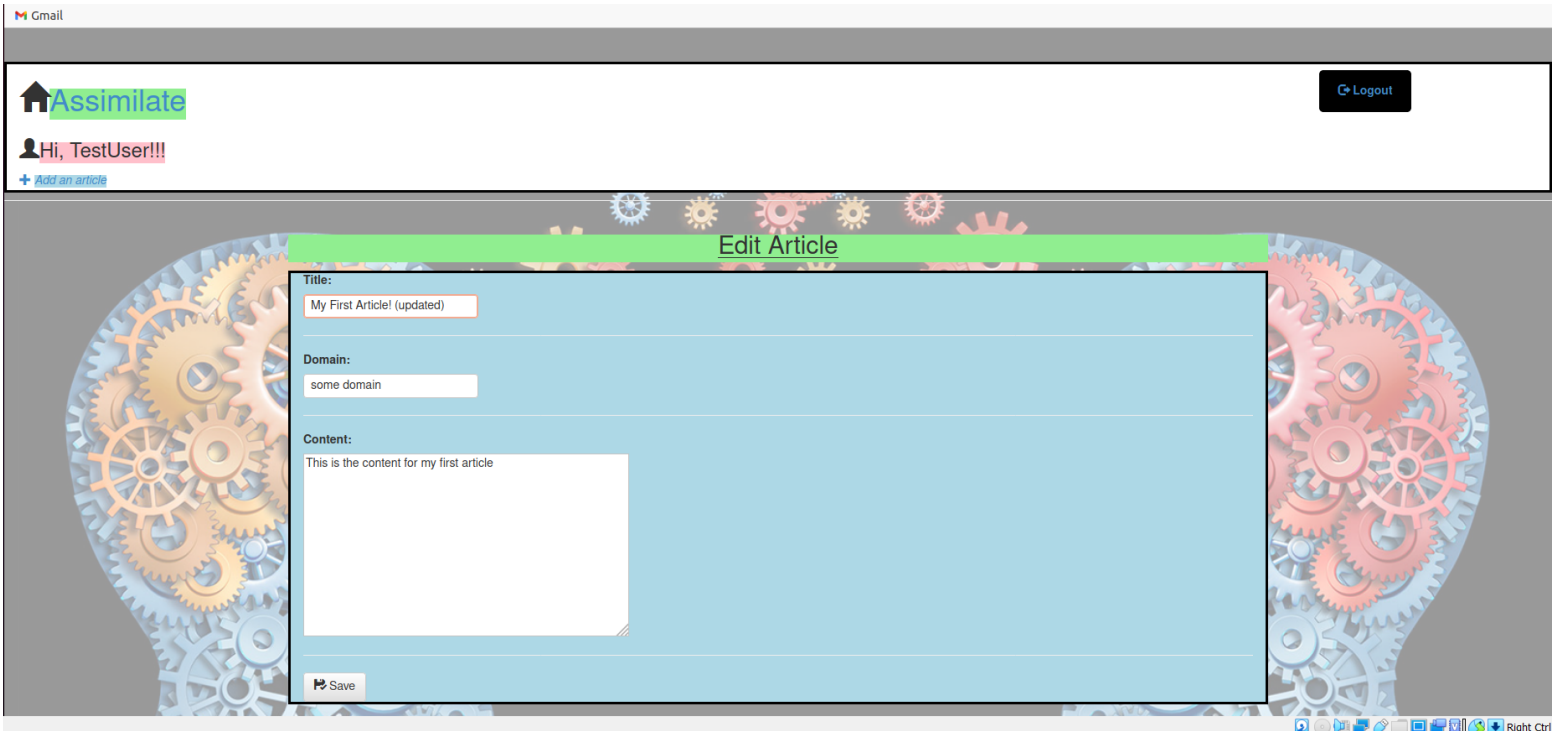

localhost:8000/post/73/
Gmail
Assimilate
Logout
Hi, TestUser!!!
Add an article
Article saved!!
My Articles
edit
delete
My First Article!
Written by TestUser
April 20, 2021, 12:45 a.m.
Domain: some domain
This is the content for my first article


localhost:8000/post_list/
Gmail
Assimilate
Logout
Hi, TestUser!!!
Add an article
My Articles
My First Article!

UT1-07


Gmail
Assimilate
Logout
Hi, TestUser!!!
Add an article
New Article
Title:
Please fill out this field.
Some domain
Content:
This is some content
Save

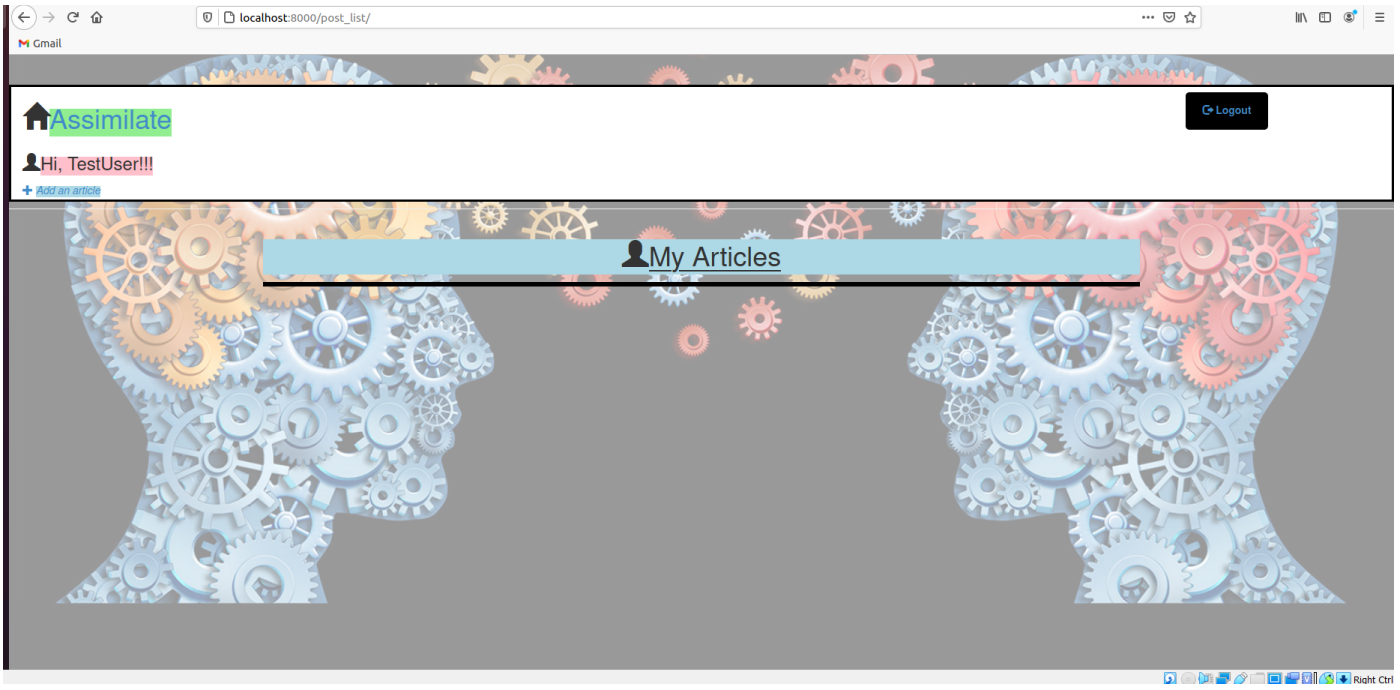
UT1-10


Gmail
Assimilate
Logout
Hi, TestUser!!!
Add an article
Edit Article
Title:
My First Article! (updated)
Domain:
some domain
Content:
This is the content for my first article
Save

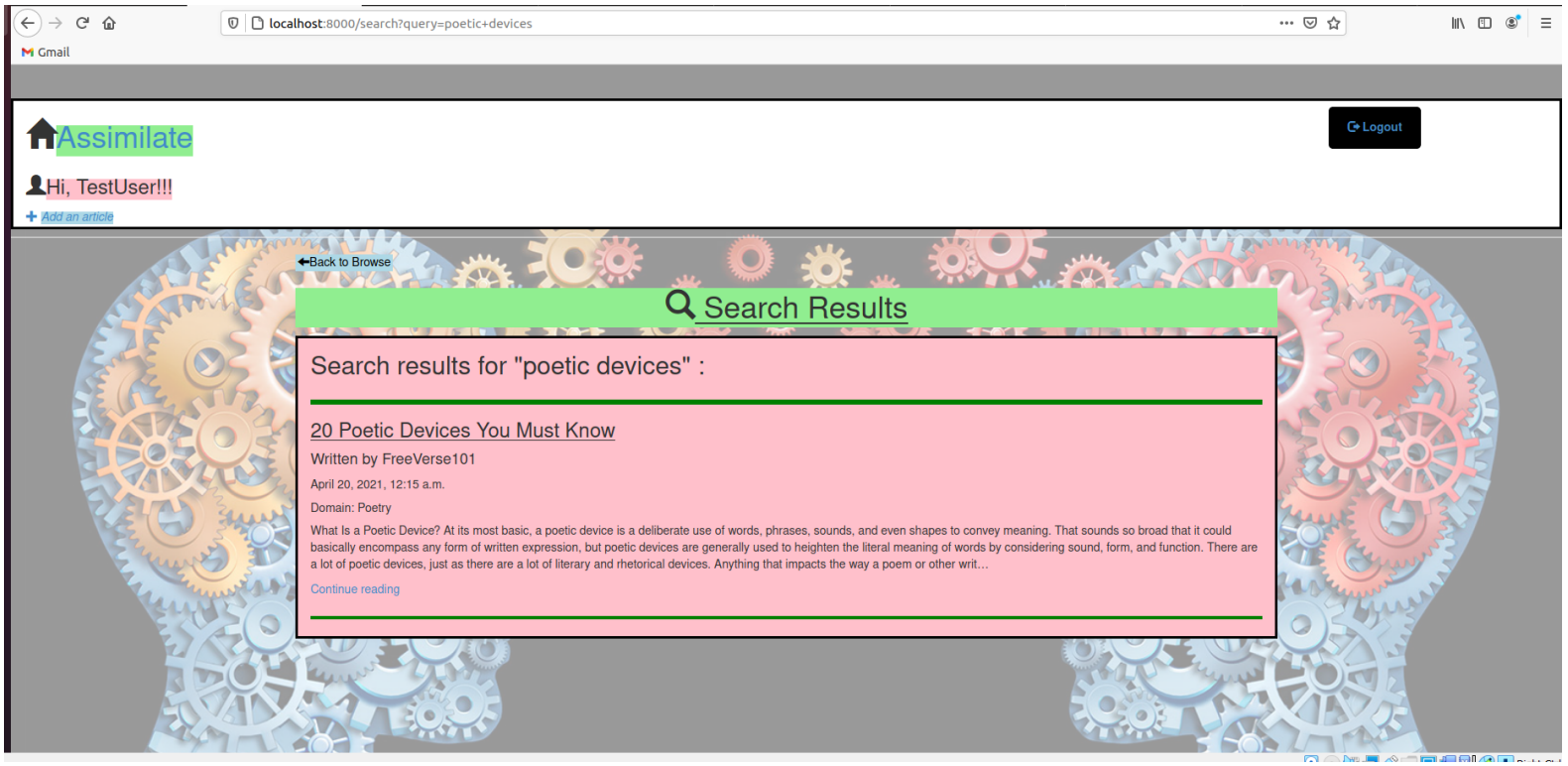

OOAD_lab [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Firefox Web Browser
Apr 20 1:00 AM
Assimilate-landing
Assimilate
Preferences
Gmail - Email from Go
Gmail - Email from Go
WT-Project_SmartBlo
knowledge picture - G
online tool to make a
localhost:8000/post/73/
Gmail
Assimilate
Logout
Hi, TestUser!!!
Add an article
Changes saved!!
My Articles
edit
delete
My First Article! (updated)
Written by TestUser
April 20, 2021, 1 a.m.
Domain: some domain
This is the content for my first article
Type here to search
1:00 AM
20-Apr-21
Right Ctrl

UT1-12


localhost:8000/post/73/
Gmail
Assimilate
Logout
Hi, TestUser!!!
Add an article
My Articles
edit
delete
My First Article! (updated)
Written by TestUser
April 20, 2021, 1 a.m.
Domain: some domain
This is the content for my first article
Right Ctrl


localhost:8000/post_list/
Gmail
Assimilate
Logout
Hi, TestUser!!!
Add an article
My Articles

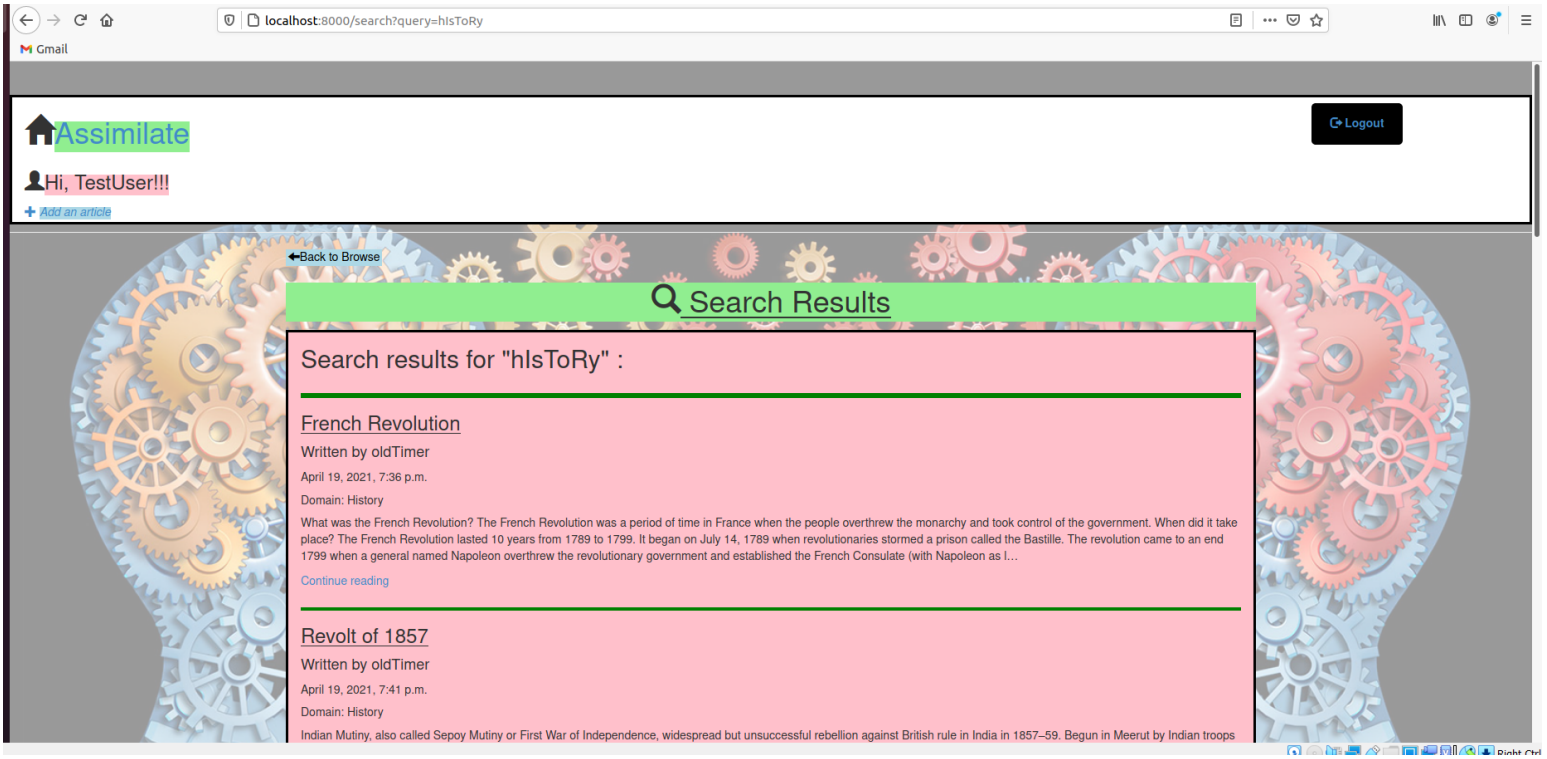
UT2-01


localhost:8000/search?query=poetic+devices
Gmail
Assimilate
Logout
Hi, TestUser!!!
Add an article
Back to Browse
Search Results
Search results for "poetic devices" :
20 Poetic Devices You Must Know
Written by FreeVerse101
April 20, 2021, 12:15 a.m.
Domain: Poetry
What Is a Poetic Device? At its most basic, a poetic device is a deliberate use of words, phrases, sounds, and even shapes to convey meaning. That sounds so broad that it could basically encompass any form of written expression, but poetic devices are generally used to heighten the literal meaning of words by considering sound, form, and function. There are a lot of poetic devices, just as there are a lot of literary and rhetorical devices. Anything that impacts the way a poem or other writ…
Continue reading

UT2-02


Gmail
Assimilate
Logout
Hi, TestUser!!!
Add an article
Back to Browse
Search Results
Search results for "This search query has more than 50 characters. Search will fail." :
No search results
Your search query: This search query has more than 50 characters. Search will fail. was too long.
Suggestions:
Try fewer keywords.

UT2-04


localhost:8000/search?query=hIsToRy
Gmail
Assimilate
Logout
Hi, TestUser!!!
Add an article
Back to Browse
Search Results
Search results for "hIsToRy" :
French Revolution
Written by oldTimer
April 19, 2021, 7:36 p.m.
Domain: History
What was the French Revolution? The French Revolution was a period of time in France when the people overthrew the monarchy and took control of the government. When did it take place? The French Revolution lasted 10 years from 1789 to 1799. It began on July 14, 1789 when revolutionaries stormed a prison called the Bastille. The revolution came to an end 1799 when a general named Napoleon overthrew the revolutionary government and established the French Consulate (with Napoleon as I…
Continue reading
Revolt of 1857
Written by oldTimer
April 19, 2021, 7:41 p.m.
Domain: History
Indian Mutiny, also called Sepoy Mutiny or First War of Independence, widespread but unsuccessful rebellion against British rule in India in 1857–59. Begun in Meerut by Indian troops

UT2-07

UT3-01 - UT3-03