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) Manu Prasad	PES1201801436
2) Goutam G Jaddipal	PES1201801698
3) Chirag ganti	PES1201802051

JANUARY – MAY 2021

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RR CAMPUS,
BENGALURU – 560100, KARNATAKA, INDIA

TABLE OF CONTENTS

Abstract

The purpose of this project is to present a detailed description, Design and working of the knowledge repository system modelled after the existing Wikipedia platform. It will explain the features and purpose of the system, the interface of the system, what the system will do and the constraints under which it must operate. This document will also provide the necessary requirements to develop the system.
The aim of **KnowledgeBox** is to provide an encyclopedia with minimum functions such as write, edit, and delete articles; search bar to search the article as well as to approve newly created articles.

Software Requirements Specification

for

KnowledgeBox

(Knowledge Repository: Wikipedia)

Version 1.0 approved

PES UNIVERSITY

01/02/2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Analysis Model	03-02-2021	ER diagram was changed, added attributes for editors and moderators	0.1
Image storage	04-02-2021	Decided to use base64 based image storage instead of file storage (S3, etc.)	0.2

HTTPS	05-02-2021	Changed requirements to use HTTPS instead of HTTP for security	1.0
Use case diagram	15-02-2021	Changed use case diagrams to use one consolidated diagram with all use cases	1.1

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the knowledge repository system modelled after the existing Wikipedia platform. It will explain the features and purpose of the system, the interface of the system, what the system will do and the constraints under which it must operate. This document will also provide the necessary requirements to develop the system.
The aim of **KnowledgeBox** is to provide an free online encyclopedia which exhibits information about numerous topics in various languages, characterized by extensive diversity. It will help organizations connect people with information and expertise globally via online searchable libraries, creation, development and maintenance of articles, and other elements.

1.2 Intended Audience

This document is intended for the stakeholders, the developers of the software behind the system, the testers and the legal team. It can also be useful to the editors and moderators, who play a key role in the growth of the system. The following sections provide an overall description of the product and information about the user interface, the analysis models, as well as the functional and non-functional requirements and constraints of the system. The document also establishes a context for the technical requirements and specification in the following chapters. This document along with its revisions can be treated as a standalone prerequisite to acquire a comprehensive understanding of the working of the system and its functionalities.

1.3 Product Scope

The system will be an online knowledge repository website that hosts information regarding any and all topics. It is akin to an encyclopedia on the web. The information on a particular topic will be condensed from various sources, and available on a single page along with the appropriate reference materials for further information. This will allow readers and knowledge seekers from across the globe to quickly find extensive and reliable information on any topic, which includes not only the relevant text, but also images pertaining to that topic. At the same time it will also provide the user with a detailed list of references for further research on the topic.

The content hosted on the repository will be crowd-sourced from its users and will be maintained by learned editors and moderated by carefully chosen moderators. Readers, editors and moderators can be anyone from any place in the world regardless of any specifics of the person.

1.4 References

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

"Software Engineering: Principles and Practice", Hans van Vliet 3rd edition Wiley India 2010.

V. Makhija, V. Kumar, A. Tiwari and A. Verma, "Knowledge Management System using CORE Repository," 2018 5th International Symposium on Emerging Trends and Technologies in Libraries and Information Services (ETTLIS), Noida, 2018, pp. 59-64, doi: 10.1109/ETTLIS.2018.8485240.

https://wikipedia.org/

https://www.britannica.com/

2. Overall Description

2.1 Product Perspective

KnowledgeBox will be an online encyclopedia that should serve as the first place to look for information on the web. The topics can be of immense diversity ranging from sports, prominent persons, locations on earth, etc. It will provide the information in various languages such as English, Spanish, Hindi, French, etc. The repository will provide concise and to the point information about the topic without having to read articles or technical papers related to it.

The information will be user-submitted but will guarantee reliability as it will be checked for correctness and recentness by learned editors and moderators. Additionally, all content posted will be backed by a list of references which may be links to other articles on KnowledgeBox or elsewhere, including technical and scientific research papers. Thus, it will provide a single dependable source of quick information on any topic possible.
This system will be standalone, independent of any other systems.

2.2 Product Functions

The following functions will be available from the knowledge repository:

- Reading: any user of the system, whether registered or not, will be able to access all articles hosted on the system. The information pages will be accessible even without logging in.
- Search: any user will be able to search for information related to a topic by typing in keywords, in any supported language, that can identify it. This feature will be available to all users as well.
- Editing: any authorized (registered and logged in) user will be able to add articles or edit any information on the existing articles of the repository. The edits can also be translations from one language to another, or uploading of relevant images. All additions/edits will be validated by one or more moderators.

- Moderating: any authorized moderator will be able to accept or reject the aforementioned additions/edits to a page.

2.3 User Classes and Characteristics

Users consist of Readers, Editors and moderators. The following describes characteristics of each user class.

2.3.1 Readers

- Readers can be anyone from the world who wish to seek knowledge on any of the topics hosted on the website.
- Readers can access any content page on the repository and can use the search function to look for specific pages.
- Frequency of use may vary from only once to several times per day. Every reader will be anonymous and no tracking is done on them.
- The readers will not be expected to have any technical expertise and knowledge of the system itself; they should be able to read the content even without it.
- Readers do not have privileges to edit the pages or moderate edits.

2.3.2 Editors

- Editors are responsible for editing and providing good content for the users. They will be having login credentials for editing a particular page.
- Editors can submit edits which moderators can then accept, reject or suggest changes.
- Editors can also perform the basic actions available to all readers.
- Editors have the privileges to edit pages but cannot accept or reject other editors' changes.
- The editors will be expected to have basic knowledge of writing in their language of choice as well as some familiarity with formatting text (such as headings, bold, italics, links, etc.). They should be able to navigate the page and access suggestions given by the moderators.
- Frequency of use is expected to be higher compared to readers.

2.3.3 Moderators

- They will be moderating the work done by the editors so that users get good content.
- Moderators have the responsibility of reviewing and accepting, rejecting or suggesting changes to the edits made by editors.
- Moderators can additionally perform all of the actions available to readers and editors.
- The moderators will be expected to have a high degree of knowledge regarding how the knowledge repository works including text editing, formatting, edits made by editors, leaving comments on edits, accepting/denying edits, etc.

- Moderators are expected to be the most frequent class of users - upto several times a day.

2.4 Operating Environment

2.4.1 Server Side

2.4.1.1 Hardware

The web application will be hosted on a web server with high speed internet capability. It will listen to incoming requests on the standard Hypertext Transfer Protocol (HTTP) port 80.

2.4.1.2 Software

An Apache web server will be used to serve the required web pages to the incoming requests. A database will be hosted using MySQL.
The following softwares are required to develop the product

- HTML5
- PHP
- CSS3
- Python3
- Mysql, Sqlite3, Firebase firestore
- Django, React & bootstrap4

2.4.2 Server Side

2.4.2.1 Hardware

A physical machine having a monitor screen, mouse and keyboard.

- The monitor screen will be used by the web browser running on the machine to display the information on the web pages.
- While the mouse and keyboard are not a strict necessity for using the website, they may be required to perform some specific actions such as navigating around, searching, etc.
- The mouse and its buttons can be used for interacting with the software environment such as activating areas of data input, open and select options from the context menu, navigating out from the current page.
- The keyboard can be used for entering input data for searching, or for utilizing browser specific keyboard shortcuts.

2.4.1.2 Software

An Operating System running a modern web browser such as Google Chrome, Mozilla Firefox, etc., which supports HTML5, CSS3 and JavaScript.

2.5 Design and Implementation Constraints

2.5.1 Hardware Constraints

The software and the APIs should be able to run on low power commodity servers such as EC2 micro instances as they may be hosted on such servers to reduce cost.

2.5.2 Database constraint

The size of the database utilized depends on the number of articles, images and the number of users.

2.5.3 Operational Constraints

The system is restricted by the operating server based on the maximum number of users it can serve at any given time. If there are a substantial number of users trying to access the website at the same time, then the server may operate slower than usual.

2.5.4 Software constraints

The client software will run best on Google Chrome 10 and above, Mozilla Firefox 4 and above and Internet Explorer 8 and above.

2.5.5 Communication Protocols

HTTP
HTTPS (for security measures)

2.5.6 Memory Constraints

Memory Constraints will be of a concern when the size of the database grows to a considerable size.

2.5.7 User Interface Constraints

The users trying to access the website should possess basic browser navigation skills in order to understand and utilize all the functionalities provided by the system. Using the system however will be simple and intuitive.

2.6 Assumptions and Dependencies

- All the Editors will be moderated by the moderator and once the moderator approves the content then it can be uploaded.
- Editors have to create an account before editing and has to be verified by the moderators.
- The performance of the application is largely dependent on the web server provider - any QoS violation would directly impact the customer experience.

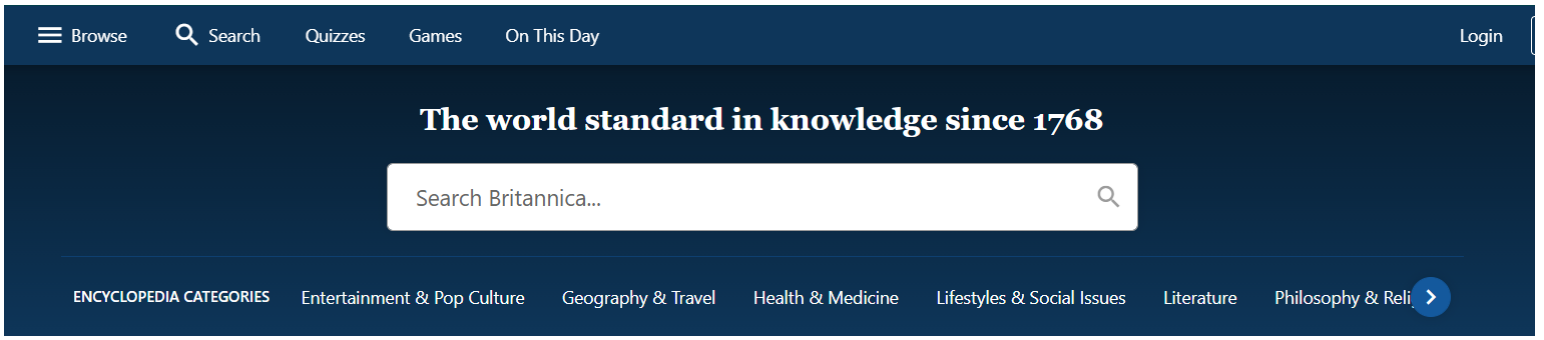
- If many readers access the repository at the same time, due to network traffic the page might load slowly.
- HTTPS is assumed to be secure with end-to-end encryption and hence no other encryption layer is used.
- The database is assumed to be safeguarded from external attacks.

3. External Interface Requirements

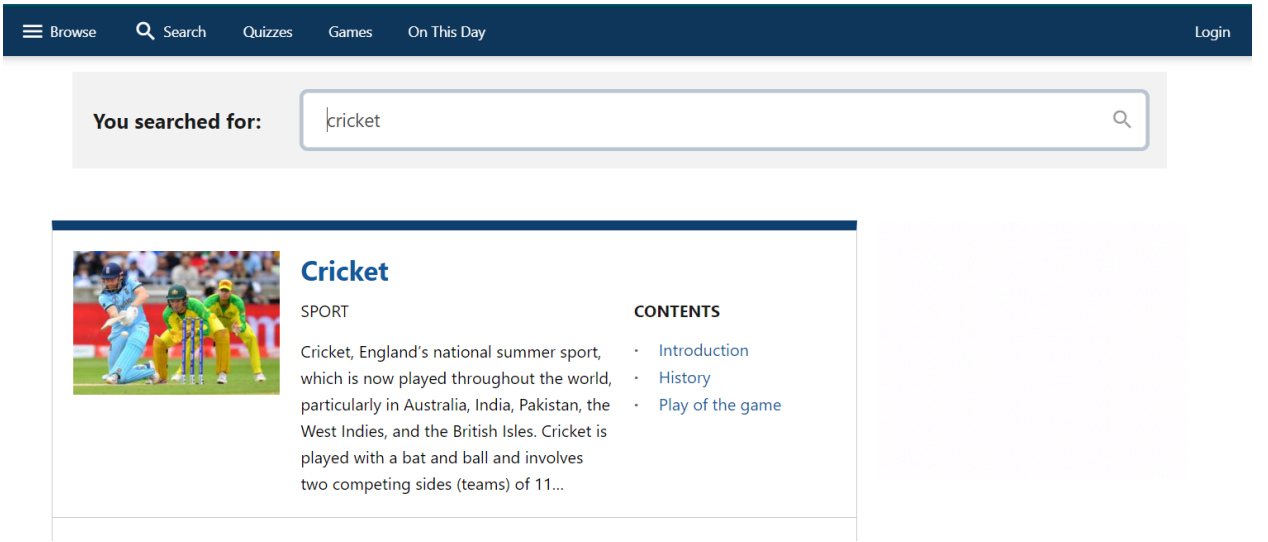
3.1 User Interfaces

User Interface consists of a dynamic Web page.Web page consisting of following features.

- A search box for the user to search a particular topic.
- Login option for the editors on top right corner of web page
- A tab below the search box having topics mentioned such as sports, business, economics, history etc.
- Users will be able to enter the topic of their interest or can select a particular topic from the topics tag.



Sample search bar image



Sample image for search result

3.2 Software Interface

The product will be communicating at two specific points in the architecture:

1. Backend and frontend
2. Backend and database - this will be handled by the database connector and hence the details are not required.

The communication between backend and frontend will take place with the help of REST APIs. The following REST APIs are designed.

- `GET /page/<page title>`
    - This is used to access specific pages
    - The only parameter is the URL encoded name of the page title
    - The response is the entire page content serialised in a JSON if it exists, or a 404 error if it does not exist
    - The structure of the JSON is as follows:
    - ```
      {
            "title": "Page title here",
            "contents": [
                  {"type": "text", "heading 1": "content 1"},
                  {"type": "text", "heading 2": "content 2"},
                  {"type": "image",
                   "image": "base64...",
                   "caption": "caption here"},
                  ...
            ]
      }
      ```
    - This JSON will then be rendered by the front end into the page
    - Images are serialised into base64 format and stored as text. The frontend is responsible for rendering the base64 encoded images.

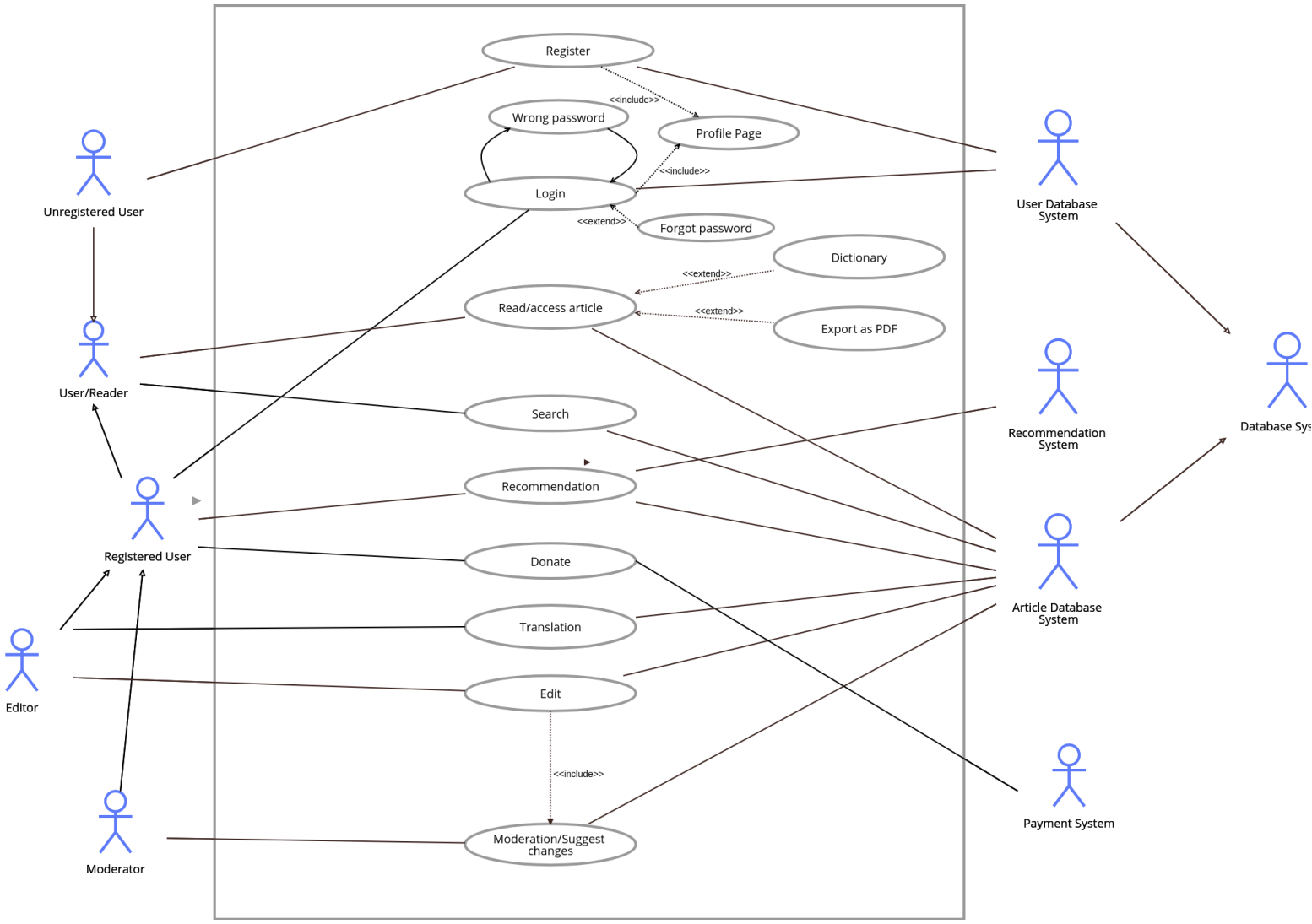
- `POST /page/<page title>/edit`
    - This is used to edit the particular page. The user has to be authorized before using this.
    - The request must have the page contents as a JSON. The structure is the same as the response returned by the previously described endpoint.
    - User's identification is inferred from the authentication
    - An edit ID is returned
- `POST /page/<page title>/edit/<edit id>/comment`
    - This endpoint is used to post a comment on an edit
    - The request must send the following in JSON serialized form:
    - ```
      {
            "text": "<comment here>"
      }
      ```
    - A comment ID is sent back as the response
- `GET /page/<page title>/edit/<edit id>`
    - This endpoint is used to view a particular edit along with the comments
    - If the edit exists, the page contents are returned as in the first endpoint along with comments.

    - {
        "title": …
        "contents": …
        "comments": [
            { "id": "...", "text": "..." },
            ...
        ]
      }
- POST /page/<page title>/edit/<edit id>/respond
    - This endpoint is used to accept or reject an edit
    - The request must have the following as a JSON
    - {
        "accept": true // or false
      }

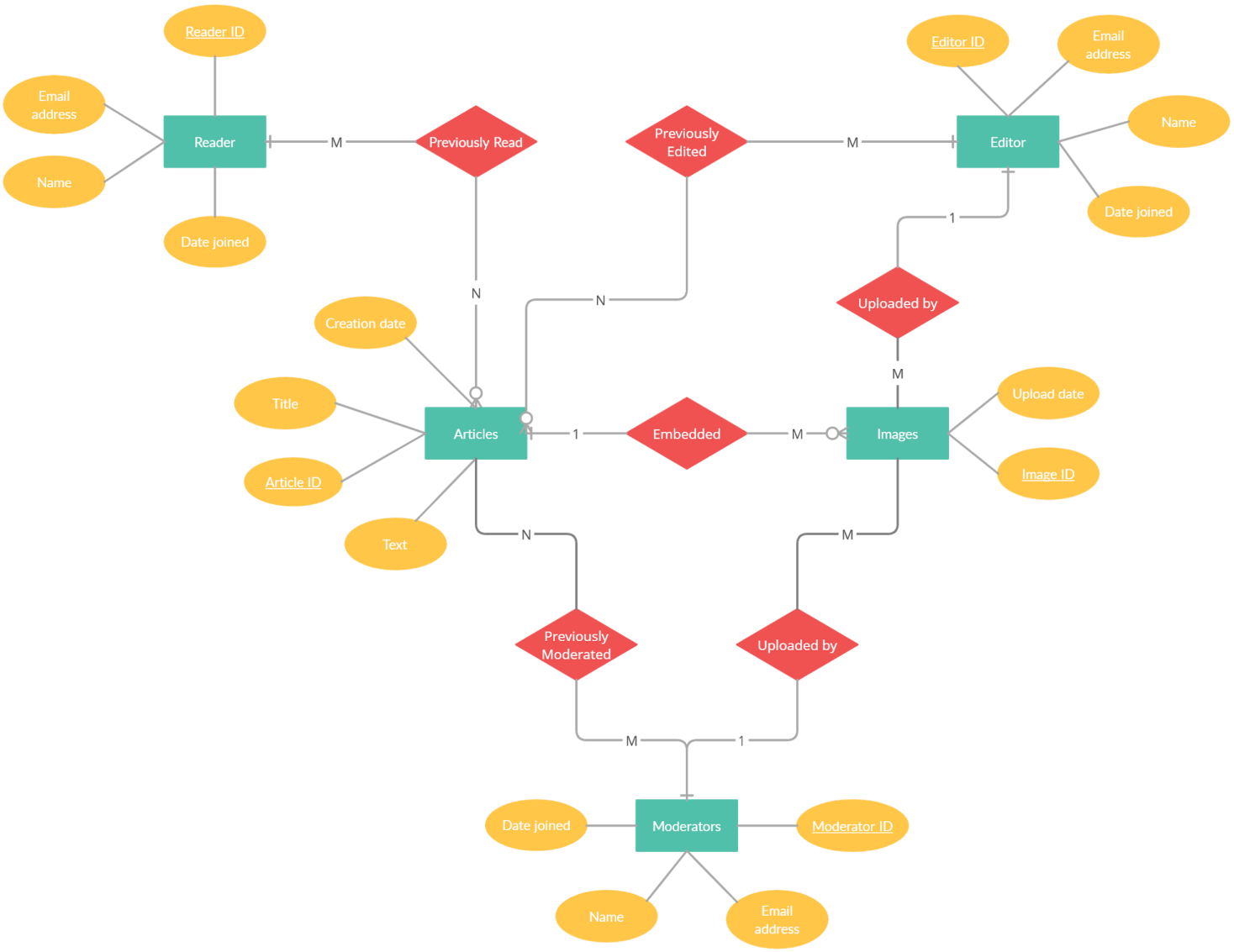
3.3 Communication Interfaces

- The client software will run best on Google Chrome 10 and above, Mozilla Firefox 4 and above and Internet Explorer 8 and above.
- The software product can be accessed from any device such as PC, laptop and phone.
- Communication Protocols:
    - HTTP
    - HTTPS (for security measures)
- The images which will be stored in the database will be base-64 encoding.
- The necessary encryption will be provided by HTTPS.
- The software does not require high data transfer rate and can be used even on slower internet speeds.
- Synchronisation is not required since there is no real time communication.
- Messages between the frontend and the backend are communicated through REST APIs and are described in the previous section.

4. Analysis Models



Use-Case Diagram for the system



The Entity Relationship Diagram for the system

5. System Features

5.1 Reading or accessing knowledge pages

5.1.1 Description and Priority

This is the feature that allows every user - anonymous or authorized - to access any page in the knowledge repository. The priority for this feature is high since it is the most important part of the product.

- Benefits: 9
- Penalty: 1
- Cost: 5
- Risk: 1

5.1.2 Stimulus/Response Sequences

Users can either access the pages using the direct URL linked from elsewhere on the internet or they can use the search feature described below to obtain a direct link to a specific page.
The only key needed to access the page is the URL which has the title in URL encoded form with a prefix.
When the user accesses the page using the URL, the page contains the title at the top and content below it. The content can include images and other media.

5.1.3 Functional Requirements

REQ-1: The entire page must be rendered. All headings, sub-headings and content must be correctly rendered on all types of devices (such as mobile phones). Any content being cut-off or inaccessible will be a failure of the system's most basic feature.

REQ-2: Invalid URL must show a "Not Found" page with links to the search feature to look for the correct page.

REQ-3: Images must be rendered correctly. Articles can include images which must be visible on all types of devices (mobile phones, etc.). In case an image cannot be shown on a particular device, it must be replaced with a warning image and the alt-text must be shown.

5.2 Searching for articles or pages

5.2.1 Description and Priority

This feature allows users to search for specific pages using keywords related to the topic. This feature has a medium priority.

- Benefits: 7
- Penalty: 6
- Cost: 6
- Risk: 1

5.2.2 Stimulus/Response Sequences

On all web pages of the system, there will be a search bar at the top right. The user will enter their query in the search box. After submitting the query it is sent to the backend server. The

server will look for the query words in the entire database and return links to the pages that match the given query. These links are called the search results.
The returned links are then rendered by the frontend and the user is shown an overview of all the matching articles.

5.2.3 Functional Requirements

REQ-4: A query containing words in the page title must match the correct page. A basic requirement of the feature is that if the user's query contains the title or words in the title of any page then that page must be returned as one of the search results.

REQ-5: If the query does not match any pages, an appropriate message must be shown along with instructions for the user to better phrase their query.

REQ-6: The search results must link to the pages directly and optionally include a part of the page contents.

5.3 Editing a page

5.3.1 Description and Priority

The editing feature allows authorized users to submit edits to any page in the system. The edits are then reviewed by the moderators.

- Benefits: 8
- Penalty: 9
- Cost: 9
- Risk: 9

5.3.2 Stimulus/Response Sequences

An authorized user (editor) can start editing a page by using the edit button on the page. After performing the edits, the user can submit it for review. The user will then wait for the moderator team to review it. The moderators can accept the changes, reject the changes or suggest changes to be made by the user. The user can perform further edits and the cycle of review continues until it is accepted or rejected.
Edits will be shown in a single place where the moderators can easily access them. Reviews are left as comments on the edit and the user will be notified whenever a comment is made.

5.3.3 Functional Requirements

REQ-7: Editing must be text-based with a formatter built into the interface. Text formatting options such as headings, links, bold, italics, etc. must be available to the editor. There must also be an option to

preview the changes and the final page which will allow the editor to ensure that their changes do not break formatting.

REQ-8: Notifications. Moderators must be notified of new edits and editors must be notified of comments on their edits. These notifications can be built into the home page of the product. Alternatively, an email or push notification service can be used.

REQ-9: Changes must be highlighted. To ensure that the moderators can easily figure out what has changed in the page, the interface must highlight all additions and deletions. A comment box must be provided on the same page for the moderator to leave reviews.

5.4 Authentication

5.4.1 Description and priority

Authentication is required to correctly authorize registered users and allow them edit access. Invalid authentication could lead to serious security issues.

- Benefits: 8
- Penalty: 4
- Cost: 6
- Risk: 8

5.4.2 Stimulus/Response sequences

A user has to first register on the registration page by providing their details described in Appendix B. The details are then stored on the database and the user gets a confirmation link in their email inbox. User registration is complete when they open the link.

To log in, the user has to enter their username or email and the password. If the password matches, the user gets access to their account and can then suggest edits as described above.

5.4.3 Functional Requirements

REQ-10: The email address provided by the user must be a valid internet email and the email must be verified before the user can access their newly created account. Verification is done by sending a confirmation link on the user provided email address.

REQ-11: The password must be secure. The user provided password must contain at least 8 characters, one capital letter, one digit and one special character.

REQ-12: Email address and username must be unique across all users. No two users can have the same email address or username.

6. Other Nonfunctional Requirements

6.1 Performance Requirements

The website must be light enough to run on low power devices such as cheap mobile phones, tablets, etc. This is to ensure that people from every walk of life can access it without being restricted by their situation or financial conditions. The content must render correctly and must be readable even on smaller screens.

The backend of the website must be scalable enough to provide the pages quickly (within a few fractions of a second) even under high load. The search functionality must scale well with the total number of pages stored and the performance must not degrade even when there are thousands of pages. To be specific, the search results must be available within a few seconds of submitting the query.

6.2 Safety Requirements

As content is user-submitted, there is a possibility of potentially harmful information being submitted and accepted. To prevent this, the standards for being a moderator and for accepting an edit must be high i.e., edits must be scrutinized heavily, perhaps by multiple moderators before it is accepted.
As the Web Page will be hosted online there is a possibility of database failure. To handle this issue there will be a mechanism to migrate the database so that the repositories are safe during catastrophic events.

6.3 Security Requirements

A security consideration is that only registered users must be able to submit edits. This can be safeguarded by using strong authorisation protocols. To protect passwords in the database from being used by malicious parties in case of a database breach, the passwords have to be hashed with a salt i.e., they must not be stored in plain text.

A privacy concern that users could have is that the pages that they view through the product can be used to profile them and can be used to create an online advertising target out of them. This can be prevented by avoiding any kind of storage (such as cookies) for readers i.e., the pages that they view must not be tracked in any way.

The selection of moderators is important as they control the quality of content on the system. The process must be strict and only trusted people must be allowed to become moderators.

6.4 Software Quality Attributes

- **Robustness** : The user can access the Web page without any delay .
- **Correctness**: Moderators control the quality of content and the correctness as well.

- **Maintainability**: A robust database eases the work of storing the files and images and it's easy to maintain.
- **Flexibility** : The Web page is easy to use by any reader as well as editor.
- **Availability**: The Web page will be accessible and from anywhere and anytime.
- **Simplicity**: The ease of learning is one of the characteristics the team will focus on. There will be no compromise with the quality of content.
- **Usability**: Clean and intuitive user interface.

6.5 Business Rules

- Moderators will have the access to change the content uploaded by the editor for the quality measures.
- There will be multiple moderators to verify the content uploaded by the editor to keep the quality of content as the main criteria.
- Editors having negative votes or suggestions pointed out by moderator will be given warning and in extreme circumstances they can be removed from the editing position.
- Users can provide feedback about the content and based on the feedback the QoS will be enhanced.
- Electing moderators is based on various parameters such as work experience, experience using the product, friendliness, etc.

7. Other Requirements

All content submitted to the system must be licensed using a public license such as Creative Commons. This allows content to be publicly accessible while maintaining rights to it. Editors must agree to this license before submitting any changes.

The system will also absolve itself of any legal responsibility for the content as it is user-submitted and subject only to the rules of the platform.

The moderators should be qualified enough to assist the work done by editors so that there is no compromise in quality of content.

Appendix A: Glossary

- **Key Points:**
  - Editors
  - Moderators
  - Readers
  - search option
  - Responsive Web page
- **Abbreviations:**
  - QoS: Quality of Services
  - HTTP(S): Hypertext transfer protocol(secure)
  - HTML: Hypertext markup language
  - CSS: Cascaded style sheet
  - REST: REpresentational State Transfer

- API: Application Programming Interface
- GET, POST: HTTP words
- JSON: JavaScript Object Notation

Appendix B: Field Layouts

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

Sample sheet with information required to register a Reader

Field	Length	Data Type	Description	Is Mandatory
Reader's Name	50	String	Name of the reader	Y
Age	3	Integer	Age of the user	N
Reader's ID	20	Numeric	System generated id	Y
Reader's email address	30	Email/String	email address of the Reader	Y
Date joined	-	Datetime	Date when the Reader created the Reader account	Y

Sample sheet with information required to register the Editor

Field	Length	Data Type	Description	Is Mandatory
Editor's Name	50	String	Name of the editor who will be uploading the content.	Y
Age	3	Integer	Age of the user	N
Editor's ID	20	Numeric	System generated id	Y
Editor's email address	30	Email/String	email address of the Editor	Y
Date joined	-	Datetime	Date when the editor created the editor account	Y

Sample sheet with information required to register a Moderator

Field	Length	Data Type	Description	Is Mandatory
Moderator's Name	50	String	Name of the moderator who will validate editors' additions/edits	Y
Age	3	Integer	Age of the user	N
Moderator's ID	20	Numeric	System generated id	Y
Moderator's email address	30	Email/String	email address of the Moderator	Y
Date joined	-	Datetime	Date when the Moderator created the Moderator account	Y

Sample sheet with information about an article

Field	Length	Data Type	Description	Is Mandatory
Article Title	100	String	Title of the article	Y
Article ID	20	Numeric	System generated id	Y

Article Text	-	JSON	The text details to be displayed for the article which includes the subheadings	Y
Date first added	-	Datetime	Date when the article was first added to the repository	Y

Sample sheet with information about an image

Field	Length	Data Type	Description	Is Mandatory
Image caption	100	String	Caption of the image	N
Image ID	20	Numeric	System generated id	Y
Date first added	-	Datetime	Date when the image was first added to the repository	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 Case ID
1	REQ-1	Page text must be rendered correctly					
2	REQ-2	“Not Found” page on invalid URL					
3	REQ-3	Images must be rendered correctly					
4	REQ-4	Query in page title must match					
5	REQ-5	Error handling when no pages found					
6	REQ-6	Search results must link					
7	REQ-7	Text-based editor with formatting					
8	REQ-8	Notifications					
9	REQ-9	Changes must be highlighted in edits					
10	REQ-10	Email address must be valid					

11	REQ-11	Password must match constraints					
12	REQ-12	Unique email and username					

Project Planning

for

KnowledgeBox

(Knowledge Repository: Wikipedia)

PES UNIVERSITY

11/02/2021

Table of Contents

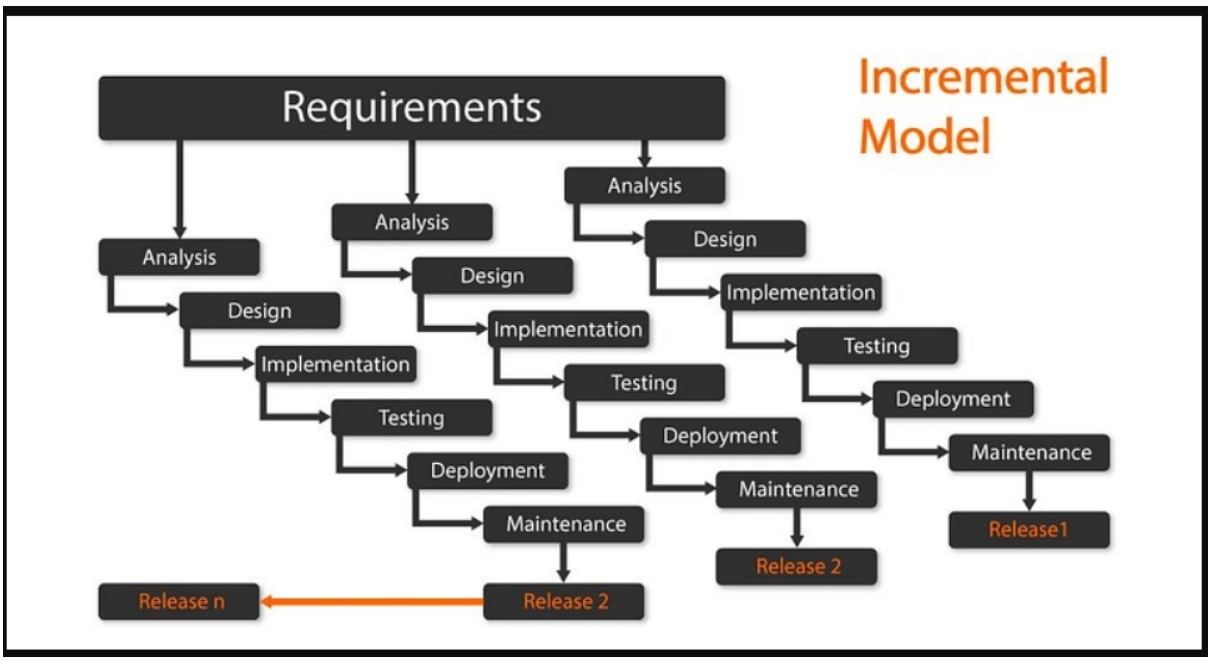
1. Lifecycle

We have chosen the **Incremental model** of software development. In this model, the software is broken down into multiple modules and each module is done in steps of requirement analysis, design, implementation and testing. This model is suitable for our project since it can be broken down into four separate modules:

- Reader
    - This includes login and registration of users, storing and accessing of articles in and from the database
- Editor
    - This module consists of the frontend and backend to support editing of articles and submitting changes
- Moderator
    - The moderator module includes adding support for a thread chain of comments between a moderator and an editor.
- Search
    - This module consists of the frontend and backend to support searching of articles based on interest. We will be performing machine learning and data analytics to support the suggestion feature in our product which will suggest the topics based on user previous history and interest.

These modules are arranged in order of priority i.e., the reader has the highest priority and search has the lowest.
Advantages of following the incremental model are:

- Early release of the product: the software can be deployed for public use after the first module i.e., the reader module is finished. This allows users to have early access to the product and the demand for subsequent features increases. It also facilitates a head start into the market and makes the users more familiar with the product which can also help figure out bugs early.
- Building the software in an incremental manner makes debugging and testing easy as the core modules are released to customers in the early phases which helps in identifying bugs.
- Splitting the functionalities of the products into different phases and identifying their independence from other features allows for a parallel workflow that improves efficiency.
- Changes can be made throughout the development cycle



2. Tools

2.1. Planning Tool

1. The tool used for project planning will be ***ClickUp*** *(https://clickup.com/)* because of the following reasons:

- Easy to use
- Supports collaboration even in a large team
- Goals and OKRs (Object and Key results) can be tracked
- Project features can be tracked with priority for each feature
- Time tracking support

2. The creation of the Gantt chart will be done using ***Office Timeline Online*** *(https://online.officetimeline.com/)* for the following reasons:

- Simple and easy to use
- Provides great flexibility
- Collaboration is easy as it is cloud enabled
- Assigning tasks to team members is effortless
- Provides tracking of milestones

2.2. Design Tool

The design tools that will be used for the project are:

- ***AdobeXD*** will be used to prototype the user interface and gauge the user experience.
- ***Figma*** is used to design and prototype the user interfaces of the web and mobile applications

2.3. Version Control Software (VCS)

Git and ***GitHub*** will be used for version control for the following reasons:

- Git is the most popular version control tool and hence has good support.
- Git is stable and gets frequent updates.
- On Git, development can take place simultaneously on different branches by different teams.
- Git is fast even for very large projects. Specific commits can be tagged for release.
- GitHub enables the team to push changes in the form of Pull Requests (PRs) which allows all members of the team to review the changes before being merged into the main branch.
- GitHub is feature-rich and provides a good remote for git even for private projects or organisations.

2.4. Development tool

The Software will be developed in different code editors using different frameworks whose details are as follows:

- The code will be written in ***VScode*** which is feasible and easy to use as a code editor with different functionalities such as auto fill and moving from one file to another is easier as it provides split screen functionality.
- The alternative code editor is ***Sublime text***. Which can be used if we want to work on different frameworks at once.
- The frameworks which we will be using in building the software are:
    - ***React JS***: It is a frontend framework with rich features and functionality which eases the work of developers.
    - ***React Native***: It is a native mobile framework for both android and ios which has all the features of ReactJs.
    - ***Bootstrap***: It is a popular and open-source HTML, CSS and JavaScript framework which can be used for building responsive websites that are mobile friendly as well.
    - ***Django***: It is a high-level Web framework for Python that promotes rapid development and clean, pragmatic design. It follows the model-template-views (MTV) architectural pattern which enables effortless abstraction.
    - ***Flask*** for API: Flask is a micro web framework for Python. It is easy to use and is good for building web APIs quickly. Each route or endpoint in the API will be a simple python function.
    - ***Firebase*** and ***firestore*** for database: Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

2.5. Bug Tracking software

GitHub Issues will be the bug-tracking software used. It was chosen for the following reasons:

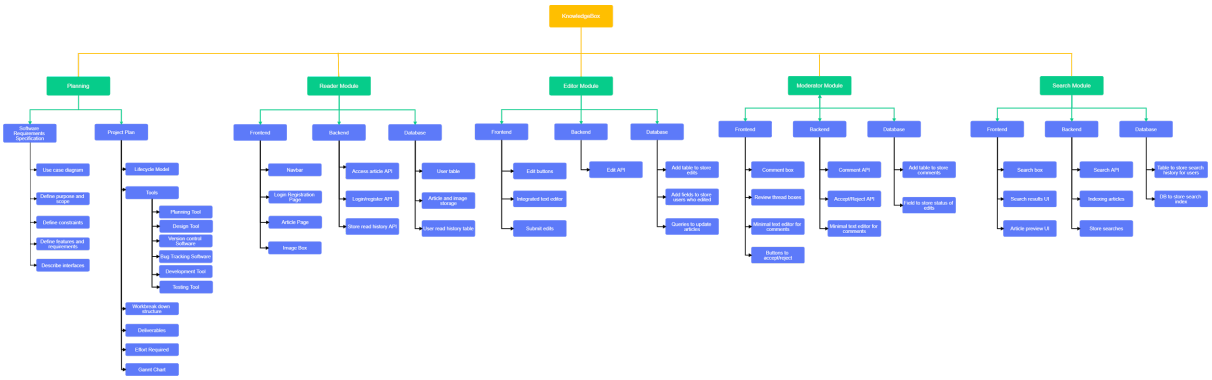
- Easy to create new issues and find old issues
- Issue templates can be made which allows the bug reporter to provide all required information at the time of creation
- Issues can be tagged with any number of predefined or custom tags
- Issues can be assigned to a specific person
- A milestone can be added to issues (such as target version)

2.6. Testing tool

Tests will be written using relevant testing frameworks such as ***Mocha*** (for JavaScript) and ***PyTest*** (for python). These tests will then be run on automated continuous integration (CI) servers using the GitHub Actions platform.
GitHub Actions workflows will be created to ensure that tests are run on every commit and failed tests are easily shown by the GitHub Actions dashboard.

3. Work Breakdown Structure

Click here for an enlarged view



3.1. Reading

Reading is the task done by readers who can be anyone from the world who wish to seek knowledge on any of the topics hosted on the website. Readers can access any content page on the repository and can use the search function to look for specific pages. The reader has an option of either login and continue or without login and continue reading. Once the reader is logged in there is a functionality of storing the history in the database so that once the reader logs in next time the software will be able to recommend the topics which the user is interested in. This feature is unique to the product.We will be performing

machine learning models for recommendation. Users will be able to provide suggestions as well so that the quality of content can be improvised.

3.2. Editing

The editing of the articles on the repository will be done by a class of users called as Editors. They are responsible for addition and editing of articles on the website to provide good content for the users. Unlike the readers, it is necessary for the editors to be registered and logged in.
To provide the appropriate functionalities for the editors, the following work structure will be followed:

- Provide the ability to do anything that a registered reader can including searching and browsing of articles to read
- A feature to access the previous articles read by the user
- A feature to access the previous articles edited by the user
- A template to suggest adding of a new article to the repository, which has the provisions to add text, multiple headings/subheadings, images, etc.
- A template to suggest edits to any of the existing articles in the repository, which has the provisions to edit the existing headings/subheadings, text, images, or even add new ones.
- A thread to communicate with the moderators who will then accept, reject or suggest changes for the work done by the editors. The editors can then perform further updates to incorporate the changes suggested to them, if any.

3.3. Moderating

The moderating of the articles i.e., the review of the additions and editions suggested by the editors, will be done by a class of users called Moderators. They have the option to accept or reject the addition/edition proposals by the editors. To provide the appropriate functionalities for the moderators, the following work structure will be followed:

- Provide the ability to browse or search for articles to read
- A feature to access the previous articles read by the user
- A feature to access the previous articles moderated by the user
- A mechanism to review the changes or updates that editors bring to the repository.
- A thread to communicate with the editors on an article
- For reviewing an article, the moderators should be given the following options
    - Accept the additions/editions and incorporate the same into the repository, which will make the updates public. This closes the communication thread, which can be reopened by the editor to perform updates if needed.
    - Reject the additions/editions. This also closes the communication thread, after which the editor who proposed the changes will not be able to reopen the thread to perform further updates.
    - Suggest changes which the editor can incorporate

3.4. Searching

Searching is a unique feature in KnowledgeBox. In this feature the user will be able to search any topic which he/she is interested in. This feature is provided to all the users

irrespective of whether the reader is logged in or not. But the readers who logged in, this feature will help in searching as the product will be providing auto fill option based on readers history.
This feature will be worked by all the team members, this feature will be implemented after reading, editing and moderating stages of the product.

4. Deliverables

4.1. Frontend

The Frontend, also known as User side will consist of all the details which will provide an interactive UI to all the readers. The Frontend will consist of user login, search feature and navigation to different web pages. The core of the frontend will be categorised as **build.** The features such as user login and search option we have to build it from scratch as there is no framework which provides the feature of search option. Django has a user login and logout feature but there will be overhead if a large number of users registers. The framework which will be used in the frontend is Reactjs and Flask for API. In Reactjs everything is categorised as **build.**It will consist of classes and functions which will be rendered based on the functionality such as navigation.

The styling required for the product is categorised as a **reused component**. Bootstrap4 will be used for styling which is a famous tool for styling among development of softwares. It is categorised as build because it consists of classes which will be called in the html components and can be used directly. The alternative of Bootstrap4 is material UI which is suitable for reactjs projects. It is an open-source project that features React components that implement Google's Material Design.

4.2. Database

User information (reader, editor, moderator) as well as the articles and associated images will be stored in a Firebase firestore database. Firebase Cloud Firestore is a scalable NoSQL Database provided by the Firebase service. Data is stored on the database without any SQL schema and hence minimal setup is required. This will be a **reused component** since the database requires minimal setup and everything that is required to be built is in the database client i.e, the web API server.

4.3. API Server

The server for the web API (application programming interface) will be implemented using Flask. The API will have many endpoints or routes as described in the requirements specification - routes for login, signup, accessing specific pages, editing a page and moderating an edit. These routes will have to be built and tested. Each route will be a function implemented in python. Hence, this is classified as a **build component**.
This API server will be deployed on heroku through the Gunicorn WSGI (Web Server Gateway Interface) layer. The deployment will be a **reused component** since no building is

required. The deployment can also be dockerized so that it can be reused across platforms (such as GCP, AWS, etc.).

5. Effort Required

5.1. Read and login features

This feature consists of login logout and recommendation based on search history. The estimated time period required is 15-20 days. This phase will start with a login feature followed by search and recommendation as this feature is dependent upon feature number 4.

5.2. Edit Feature

This feature is estimated to be completed within 10-15 days from starting it.

5.3. Moderate Feature

This feature includes accepting/rejecting of edit proposals and leaving comments on edits by the moderators. It is estimated that this feature will require 10-12 days.

5.4. Search Feature

The search feature includes the search bar and the backend for indexing and looking up keywords. This feature is estimated to be completed in 15-18 days.

ESTIMATION OF EFFORT

Project	a_i	b_i	c_i	d_i
Organic	3.2	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

● **Type of Project:** This project can be classified as “Organic” as the problem is well understood. Additionally, as our project is based on Wikipedia, we have a complete grasp and reference of the expected deliverables. The choice of the Organic model is also supported by the fact that our team size is small.
● **Lines of code** ~ 3 KLOC
● **Effort (E)** = a $(KLOC)^b$
= 2.4 $(3)^{1.05}$
= 7.60 person months
● **Duration (D)** = c $(E)^d$ = 2.5 $(7.6)^{0.38}$ = 5.40 months
● **Team strength (N)** = 3

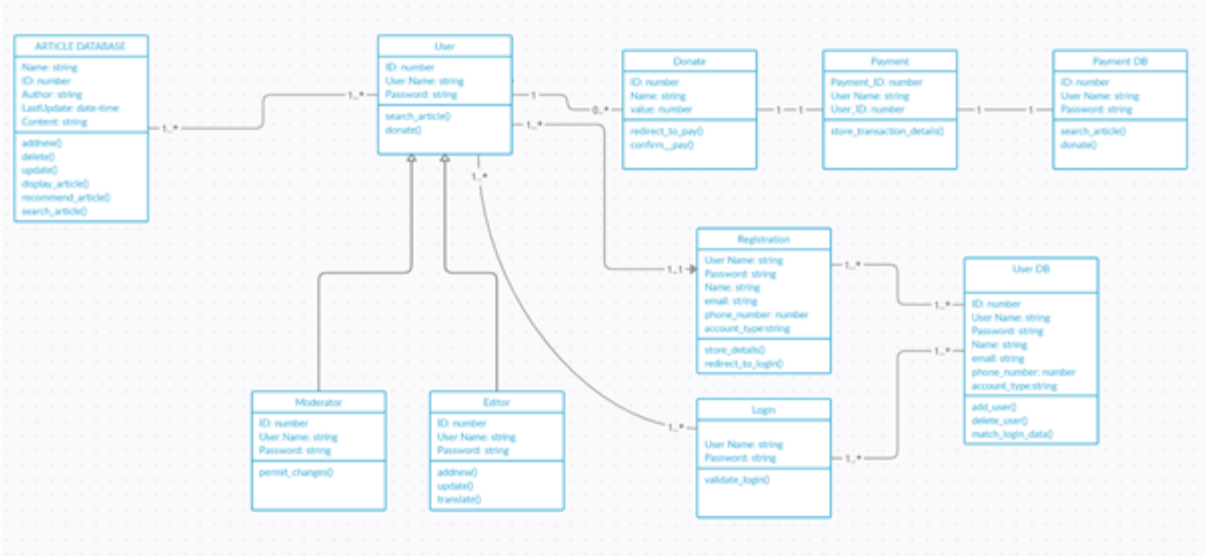
Based on the above calculations : The project can be completed in ~1.8 months if the three of us work parallely.

6. Gantt Chart

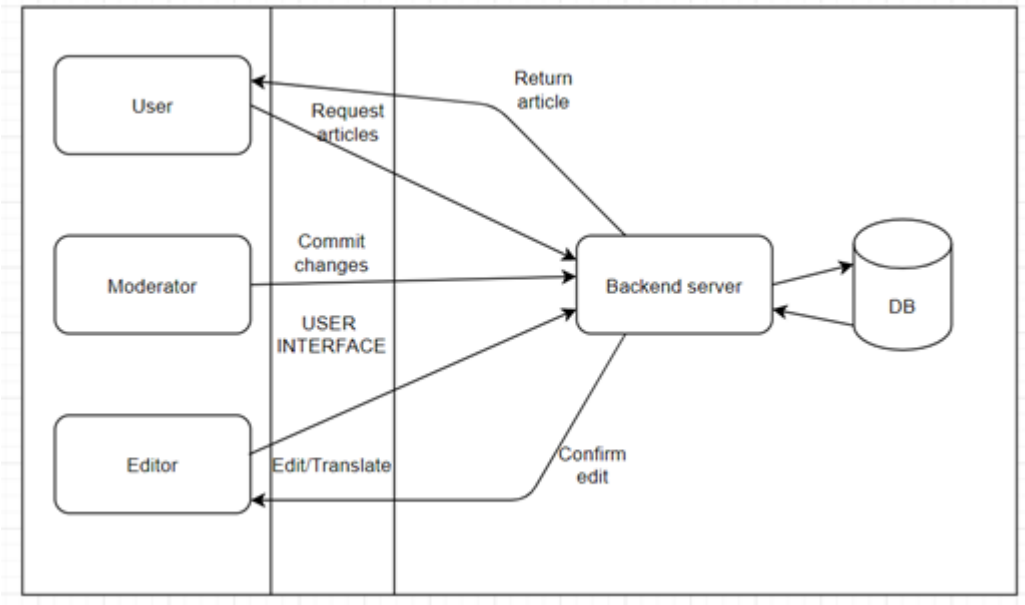
As the model we have chosen is an incremental model, the development of each of the modules shown above in the Gantt chart will include the **requirement analysis, design, implementation and testing phases**.

DESIGN DIAGRAMS :

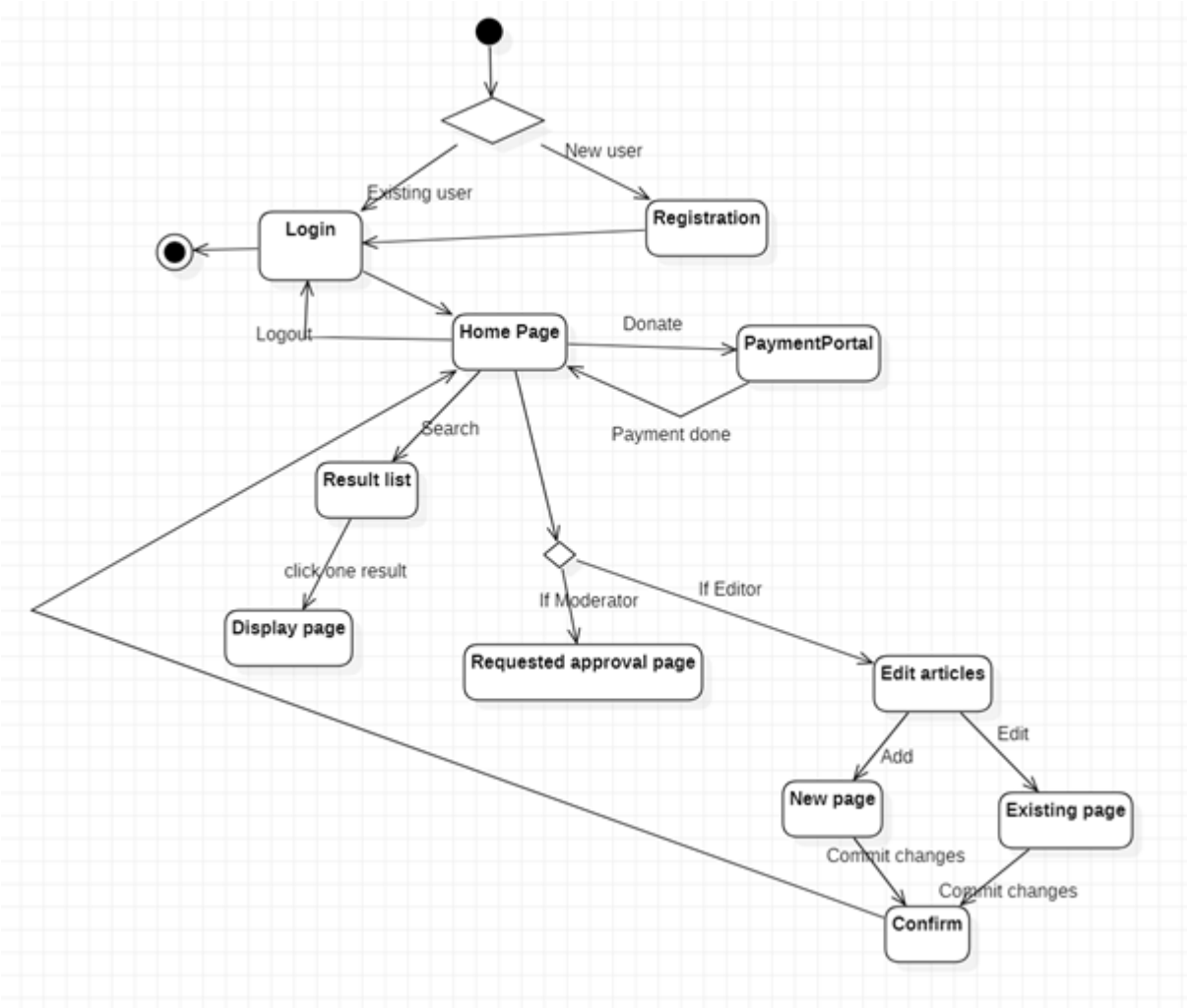
Class Diagram:



Generic System Architecture:

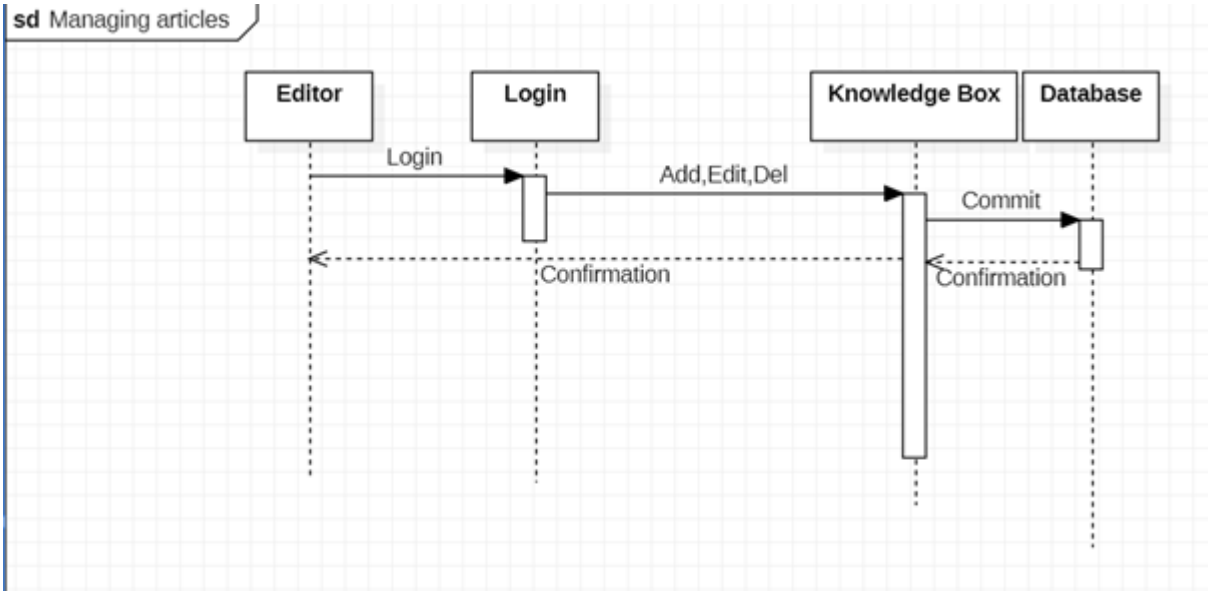


State Diagram

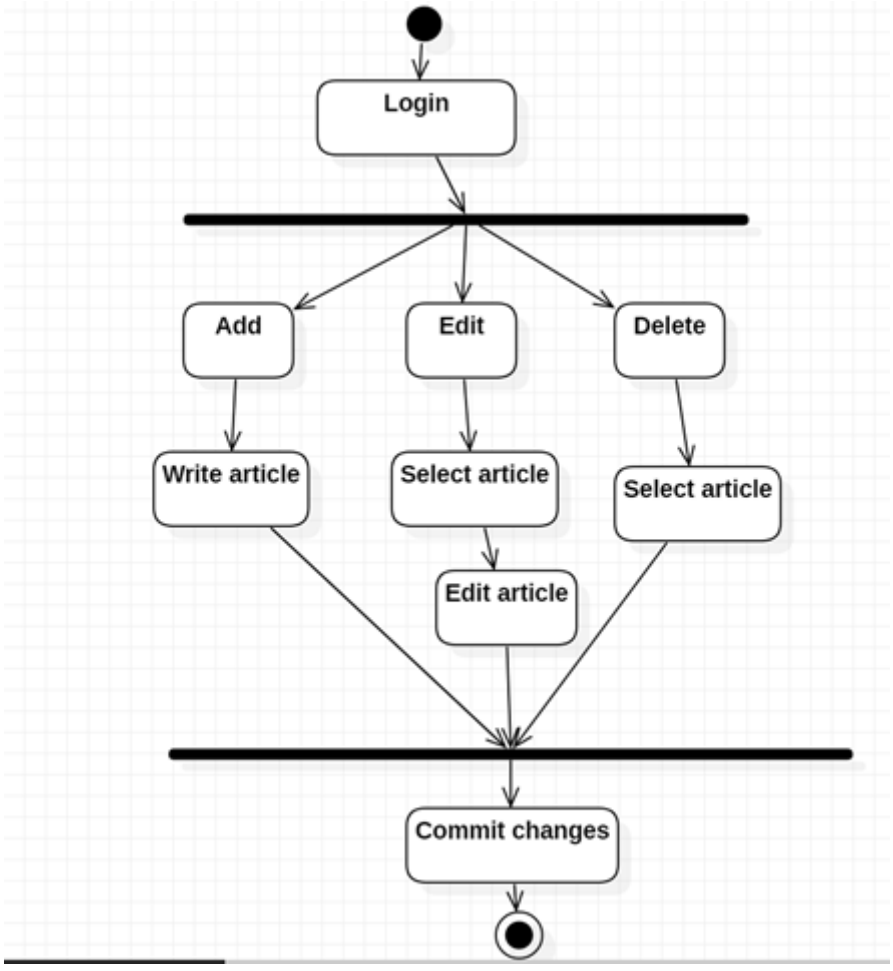


1.Managing Articles

Sequence Diagram

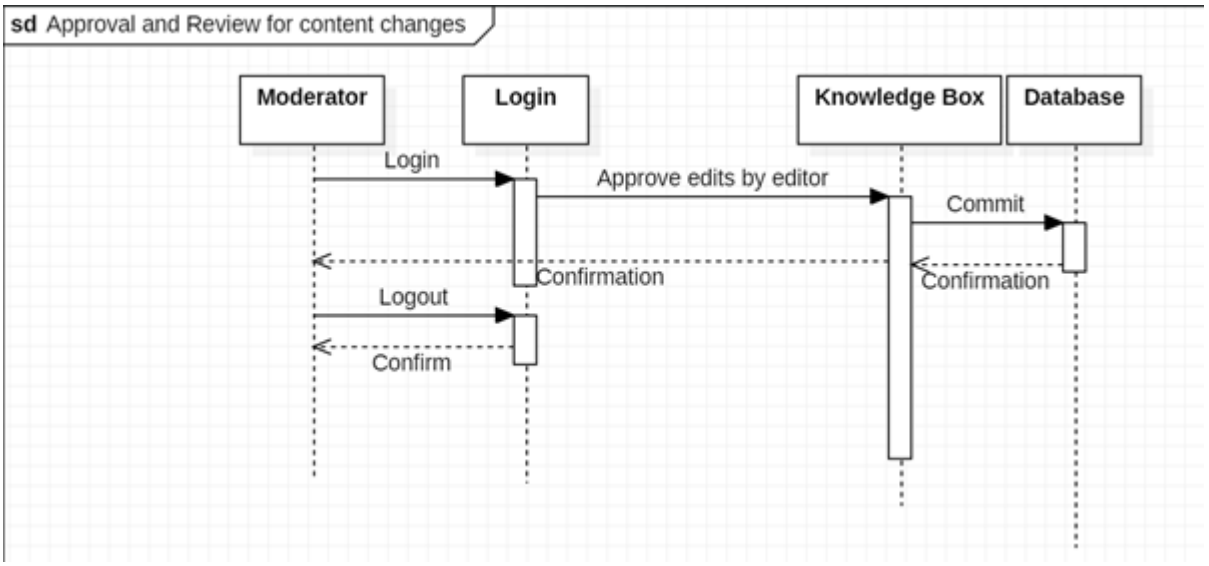


Activity Diagram

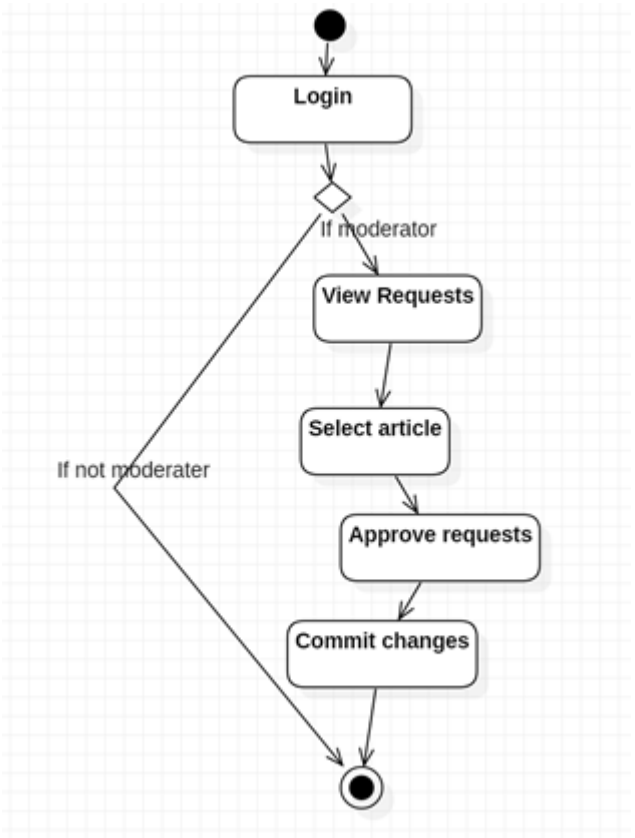


2.Approval and Review for content changes

Sequence Diagram

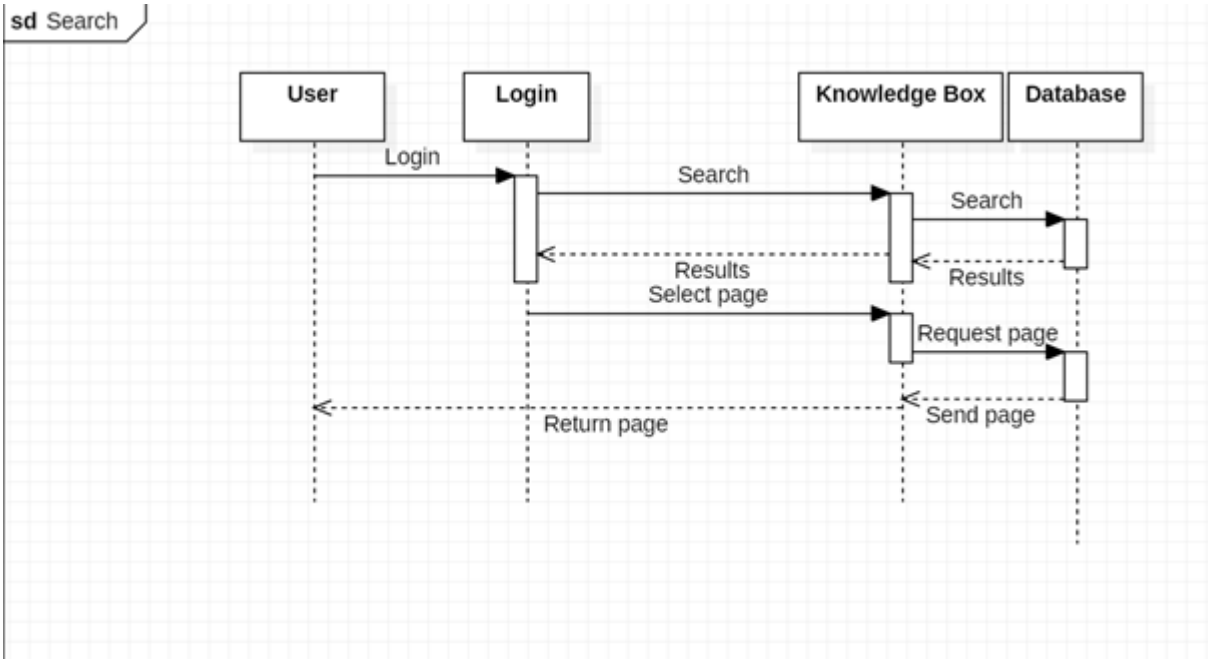


Activity Diagram

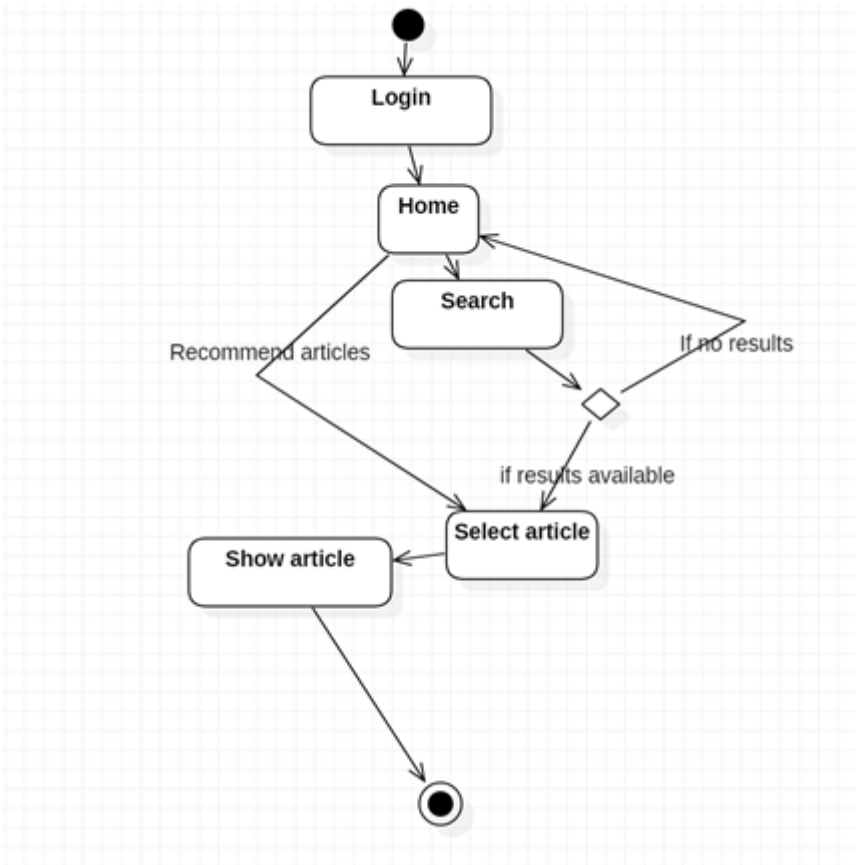


3.Search for Articles

Sequence Diagram



Activity Diagram



Module Description

1. Manage Articles

This module allows the user to create,edit,delete articles. This module is implemented using the python web framework django.
Edit: User clicks on the desired article and clicks on edit, he can make changes which on saving, will be sent for approval.
Create: User can click on create new page option on sidebar of webpage and input title and description and save it. It will then be sent to review.
Delete: Users can delete a desired article.

2. Search Articles

Searching is a unique feature in KnowledgeBox. In this feature the user will be able to search any topic which he/she is interested in. This feature is provided to all the users irrespective of whether the reader is logged in or not. But the readers who logged in, this feature will help in searching as the product will be providing auto fill option based on readers history.

3. Approve Articles

The moderating of the articles i.e., the review of the additions and editions suggested by the editors, will be done by a class of users called Moderators. They have the option to accept or reject the addition/edition proposals by the editors.

TEST CASES

Knowledge Repository

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
UT-01	homepage	To test homepage	Access to Chrome Browser	1: Navigate to website	none	homepage visible as the default page	homepage visible as the default page	Pass
UT-02	homepage	To view articles on homepage	Access to Chrome Browser	1: Navigate to website 2.click on any article	none	article should be visible	article visible	Pass
UT-03	search bar	to search existing article	Access to Chrome Browser	1: Navigate to homepage 2: type article name in search bar	article name	search must return the exact article	return the exact article	Pass
UT-04	search bar	to search non-existing article	Access to Chrome Browser	1: Navigate to homepage 2: type article name in search bar	article name	search should not return any article	doesn't return any article	Pass
UT-05	search bar	prefix search	Access to Chrome Browser	1: Navigate to homepage	article name	should return prefix	returns prefix matches articles	Pass

				2: type article name in search bar		matched articles		
UT-06	new page	To test the create new page function	Access to Chrome Browser	1: Navigate to homepage 2: click on create new page	none	redirect to creation page	redirects to creation page	Pass
UT-07	new page	To test the create new page function	Access to Chrome Browser	1: Navigate to homepage 2: click on create new page	input article	must allow to write an article	allows to write article	Pass
UT-08	save article	To test the save article function	Access to Chrome Browser	1: Navigate to homepage 2: click on create new page 3:click on save	input article	must send data to article approve	sends article to approve	Pass
IT-01	homepage	to test if non approved article is visible	Access to Chrome Browser	1: Navigate to homepage 2:after submitting article, goto homepage	none	non approved article should not be in home	non approved article in not visible in home	Pass
UT-09	homepage	to test where the page redirects when an article is submitted	Access to Chrome Browser	1: Navigate to homepage 2:press submit	none	submit should redirect to homepage	submit redirects to homepage	Pass
UT-10	edit article	To test the edit article function	Access to Chrome Browser	1: Navigate to homepage 2:click on the article you want to edit 3:edit article	edit article	should allow to edit article	allows to edit article	Pass
UT-11	edit article	to test the save edit function	Access to Chrome Browser	1: Navigate to homepage 2:click on the article you want to edit 3:edit article	none	should allow to save edits	allows edit save	Pass
IT-02	edit article	To if article is visible in edit article page	Access to Chrome Browser	1: Navigate to homepage 2:click on the article you want to edit 3:edit articles	none	article must be visible to the user	article is visible to the user	Pass
IT-03	edit article	to test if edited article is in approve page	Access to Chrome Browser	1: Navigate to homepage 2:click on the article you want to edit 3:edit articles 4:goto approve articles	edit article	edited article must be in approve page	edited article is visible in approve article	Pass

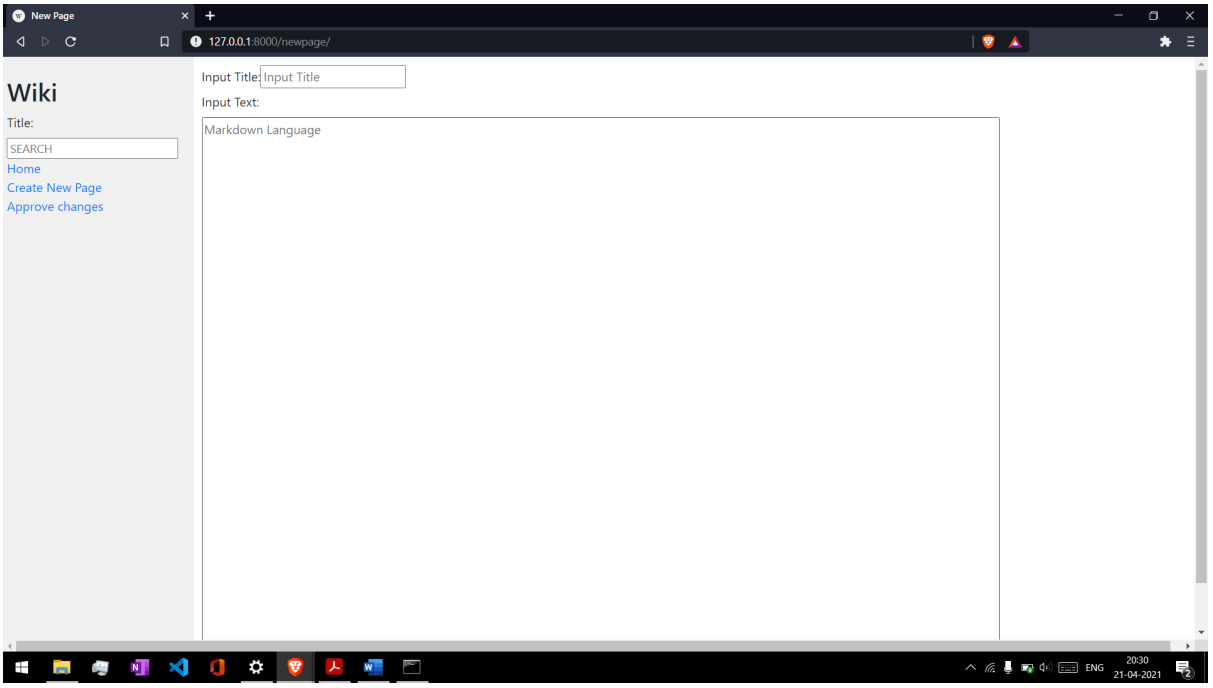
IT-04	edit article	To Test if home page has the older approved version	Access to Chrome Browser	1: Navigate to homepage 2:click on the article you want to edit 3:edit articles 4:homepage	edit article	Home page Must have older approved version of the article	Home page has older approved version of article	Pass
UT-12	delete article	To test if delete article function	Access to Chrome Browser	1: Navigate to homepage 2:click on the article you want to delete 3:delete article	none	must allow to delete article	allows to delete article	Pass
UT-13	delete article	To test if deleted article is visible in homepage	Access to Chrome Browser	1: Navigate to homepage 2:click on the article you want to delete 3:delete article 4.goto homepage	none	deleted article must not be shown on homepage	deleted article is not shown on homepage	Pass
UT-14	delete article	To test if deleted article is visible on approve page	Access to Chrome Browser	1: Navigate to homepage 2:click on the article you want to delete 3:delete article 4.goto approve page	none	deleted article should not be shown on approval page	deleted article is not shown on approval page	Pass
UT-15	Approve changes	To Test View Articles Not approved	Access to Chrome Browser	1: Navigate and Homepage 2: Click on Approve changes	none	must return the list from entries2 folder	returns the list from entries2 folder.	Pass
UT-16	Approve changes	To review the content	Access to Chrome Browser	1: Navigate to the website 2: Click on Approve changes 3. Click on an article to review	none	User should be able to review the content	content visible	Pass
UT-17	Approve changes	Check the approve functionality	Access to Chrome Browser	1: Navigate to the website 2: Click on Approve changes 3.Click on article 4.Click on approve	none	The page must not be visible, i.e. removed from un-approved folder	Approved page not in Approve changes page	Pass

UT-18	Approve changes	check the approve functionality	Access to Chrome Browser	1: Navigate to the website 2: Click on Approve changes 3.Click on article 4.Click on approve	none	the approved page should be visible in homepage	approved page is visible in homepage	Pass
UT-19	Approve changes	To Test REdirection to approve changes after approval is done	Access to Chrome Browser	1: Navigate to the website 2: Click on Approve changes 3.Click on article 4.Click on approve	none	Redirect to approve page	redirect to approve page	Pass
ST-01	Complete webpage	To test the wholes system	Access to Chrome Browse	1: Navigate to the website 2. Test every function	article input, edit article	All functions should work as expected	All functions work as expected	Pass

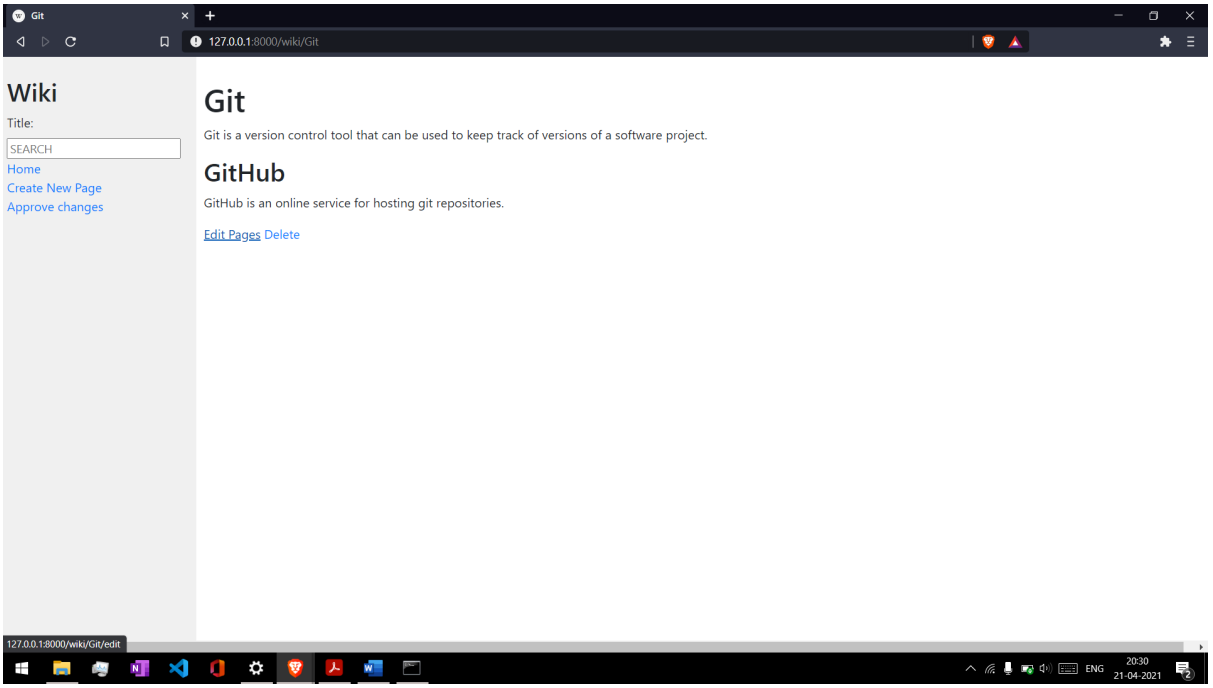
SCREENSHOTS:

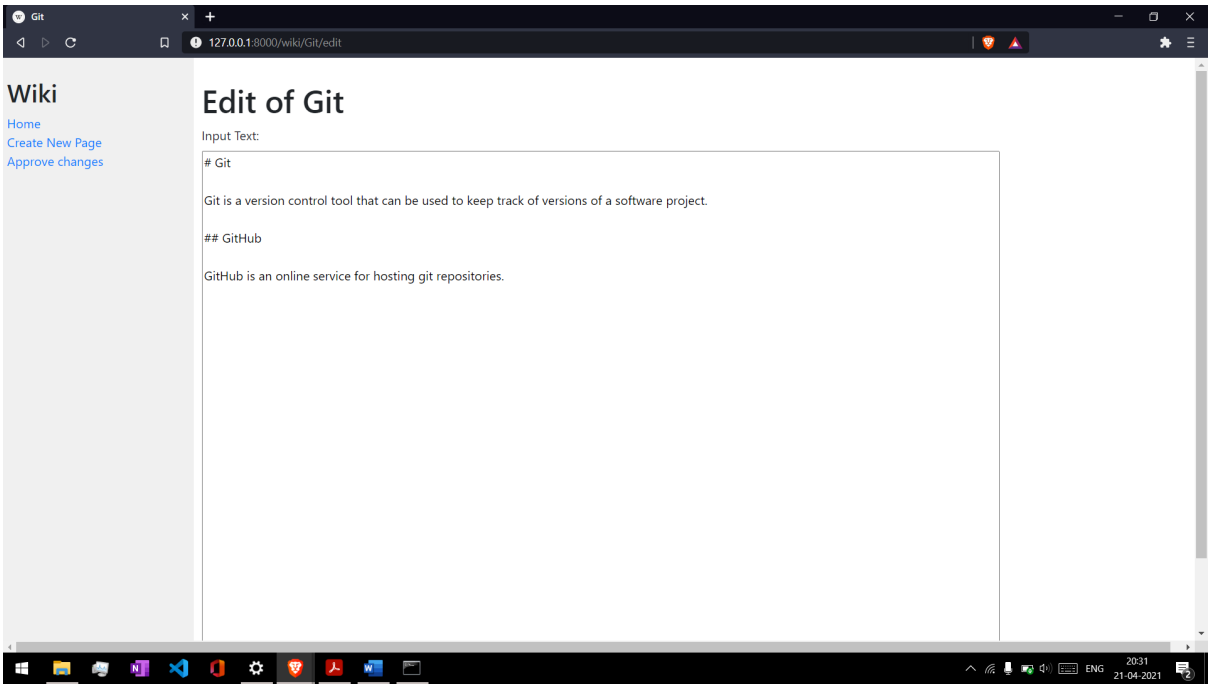
Functionality:

1. Manage articles:
    a. Create:

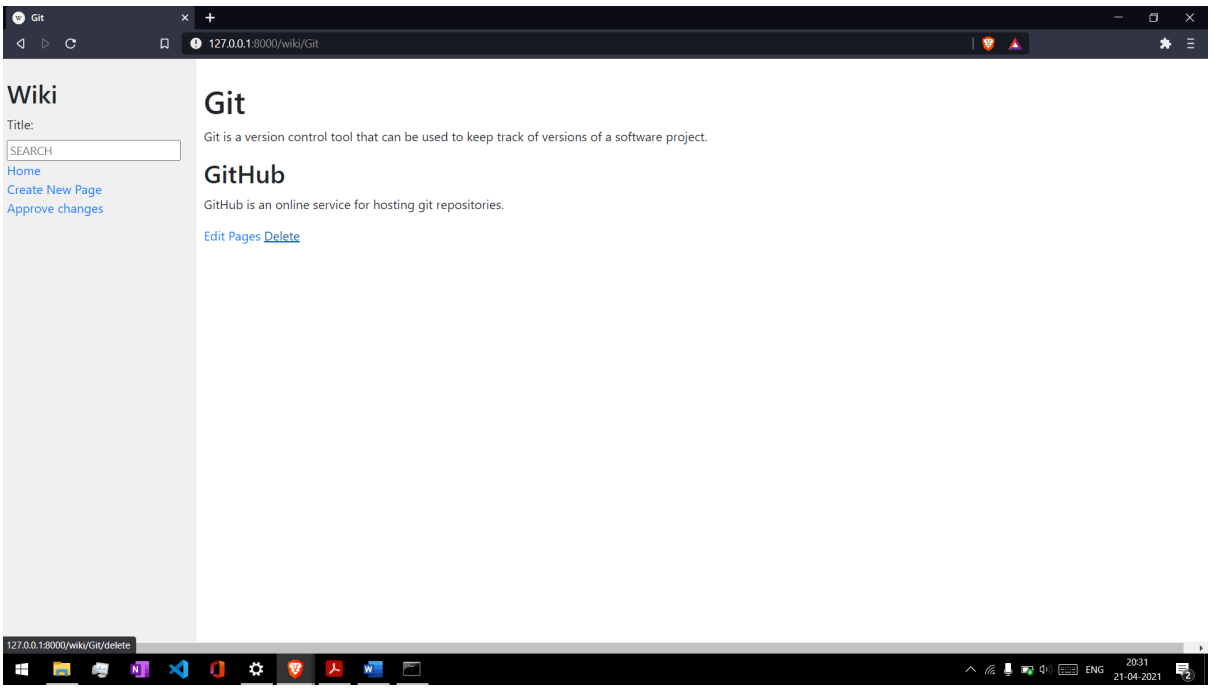


b. Edit:

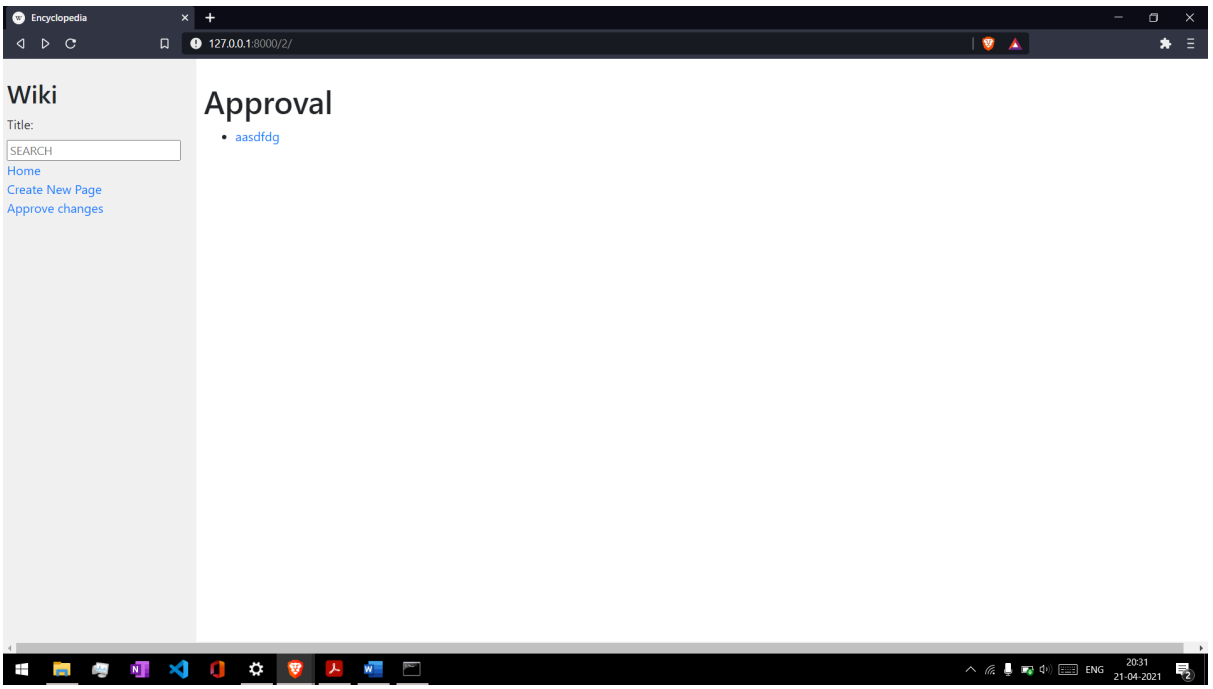


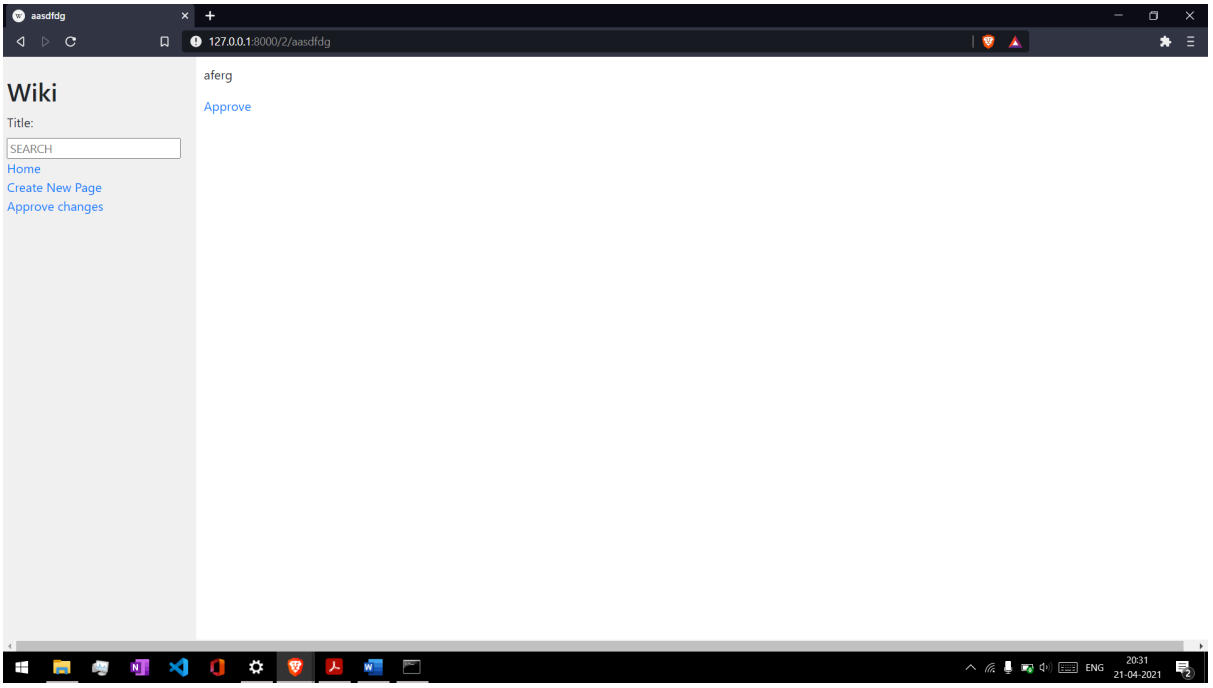


c. Delete:

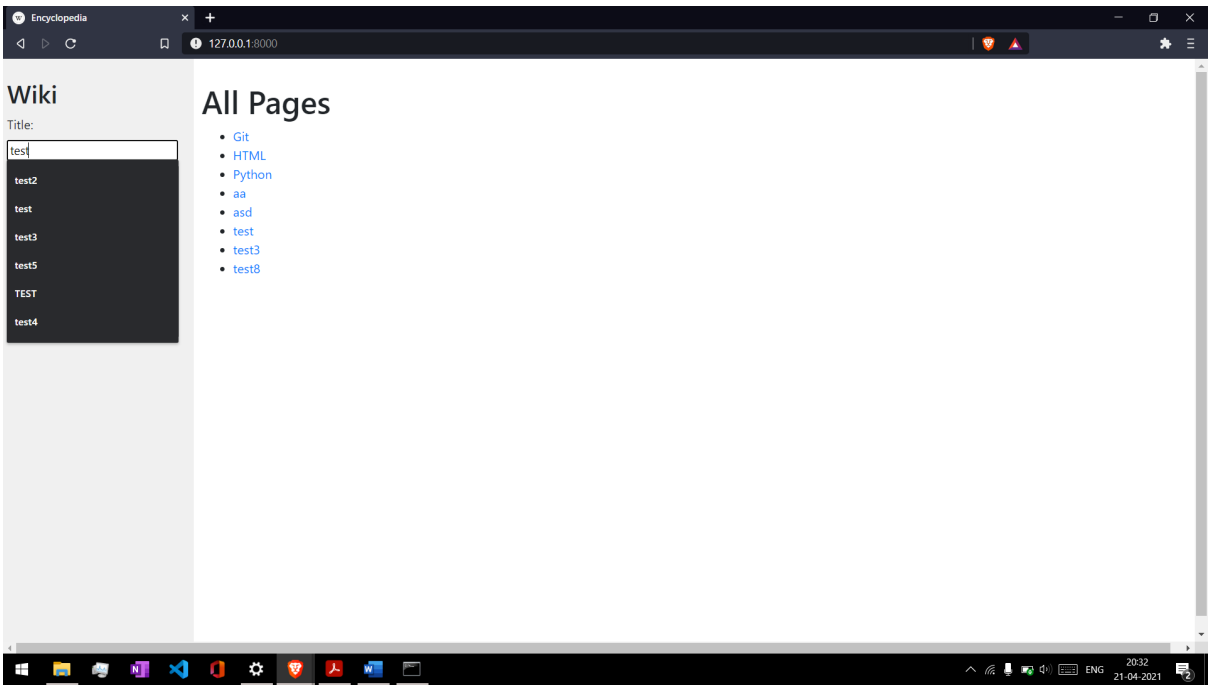


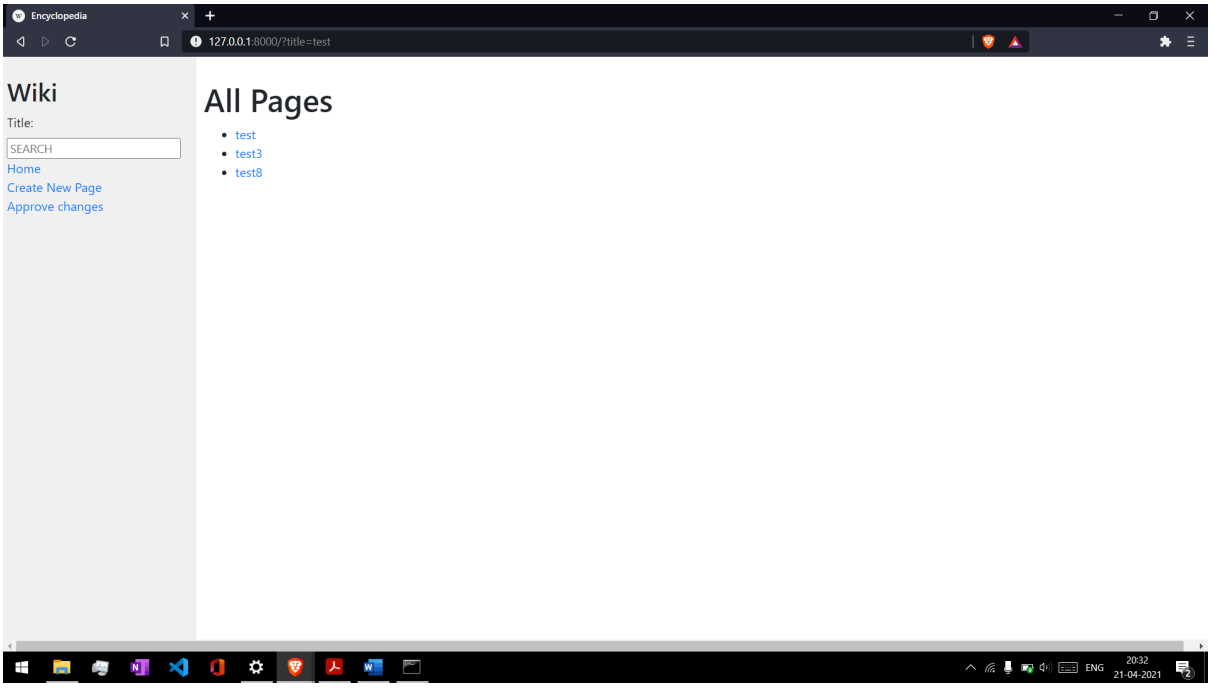
2. Approve changes:



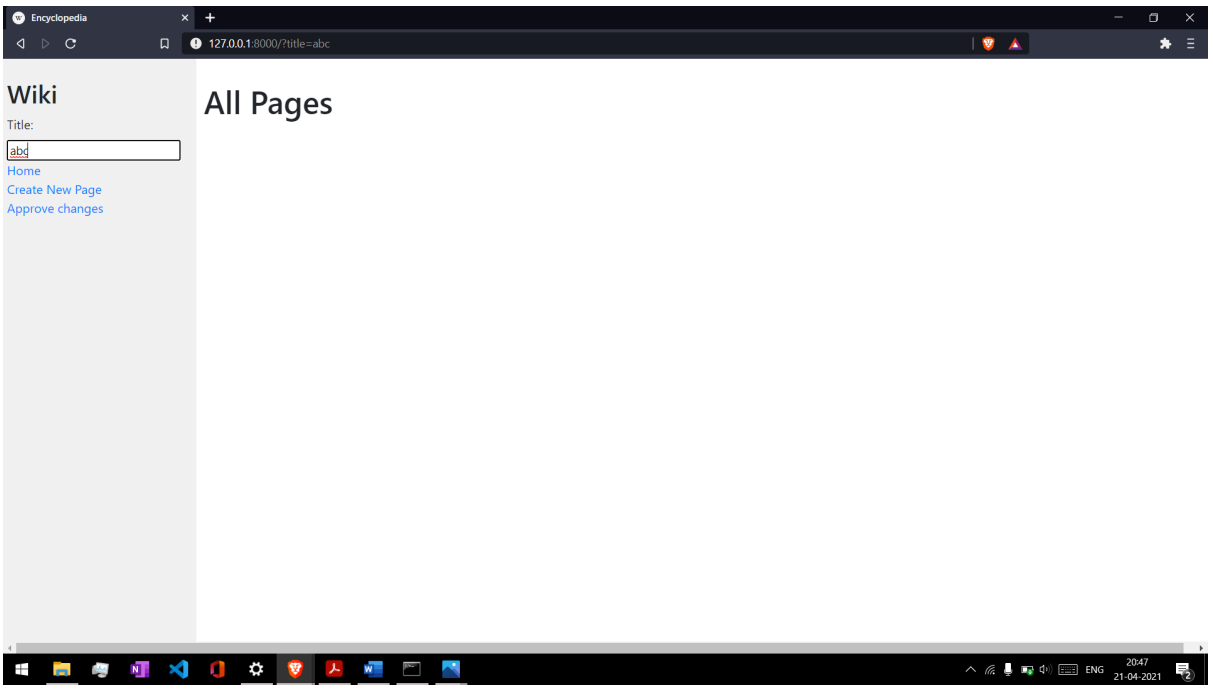


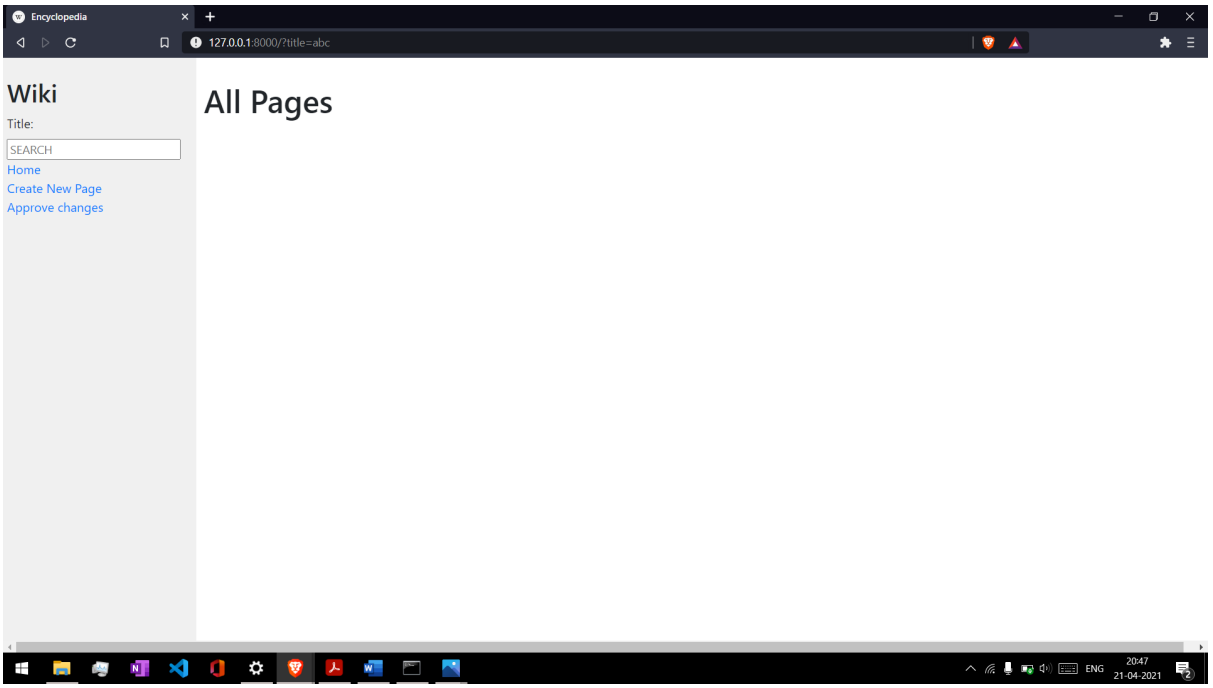
3. Search bar:
    a. successful search:





b. unsuccessful search:


Encyclopedia
127.0.0.1:8000/?title=abc
Wiki
Title:
abc
Home
Create New Page
Approve changes
All Pages
ENG
20:47
21-04-2021


Encyclopedia
127.0.0.1:8000/?title=abc
Wiki
Title:
SEARCH
Home
Create New Page
Approve changes
All Pages
ENG
20:47
21-04-2021