comp90082-2024-si-wombat

- SiLA - Team Wombat - Home Page
- Check List
- Test Account

SiLA - Team Wombat - Home Page

Welcome to the SiLA - Team Wombat

Project Description:

This project aims to develop an Online Manuscript Submission and Peer Review Platform for the Studies in Language Assessment (SiLA) journal, operated by the Association for Language Testing and Assessment of Australia and New Zealand (ALTAANZ). The envisioned platform will replace the current email-based manuscript submission and review process with a comprehensive, streamlined, online system. This initiative seeks to enhance the efficiency of manuscript handling, review assignments, editorial decision-making, and communication among authors, reviewers, and editors.

You are welcome to access our system via http://3.27.174.20/login

Stakeholder:

Name	Role	Email
Dr Eduardo Oliveira	Lecturer	duardo.oliveira@unimelb.edu.au
Dr Lucy Sparrow	Subject Coordinator	lucy.sparrow@unimelb.edu.au
Lin Li	Mentor	lin.li10@unimelb.edu.au
Jason Fan	Client	**jinsong.fan@unimelb.edu.au**
John Read	Client	ja.read@auckland.ac.nz
Annemiek	Client	annemiek.huisman@unimelb.edu.au

Development Team:

Photo	Name	Contact
	Donghong Zhuang	donghongz@student.unimelb.edu.au

	Jiayu Yang	jiayuy7@student.unimelb.edu.au
	Jiajin Yang	jiajiny@student.unimelb.edu.au
	Chuyuan Xu	chuyuanx@student.unimelb.edu.au
	Yuxuan Zeng	yuxuzeng@student.unimelb.edu.au
	Yizhi Liao	yizliao@student.unimelb.edu.au

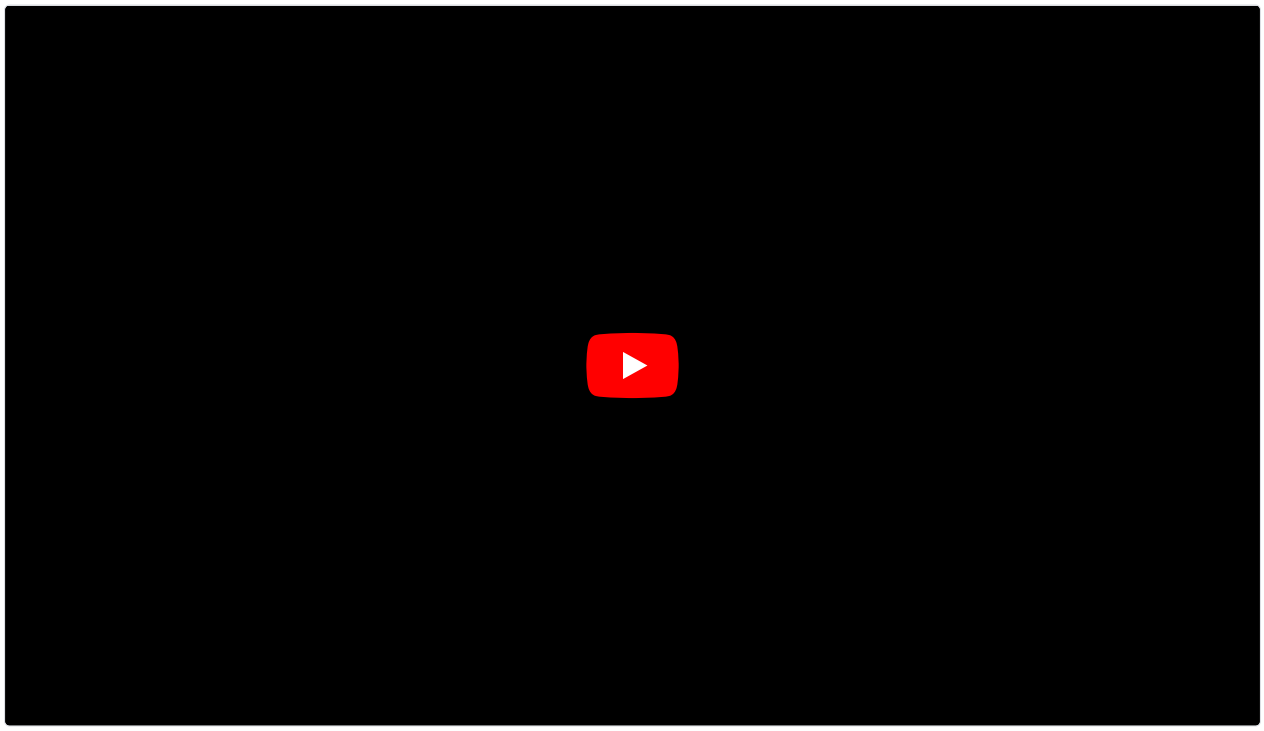
Tools:

Tool	Description	Link
Confluence	Confluence is a collaboration tool developed by Atlassian that provides a platform for project teams to create, share, and manage documents and ideas in a unified space.	Confluence-Wombat
Jira	Jira is a project management tool developed designed to help teams plan, track, and manage agile software development projects.	Jira-Wombat
Github	GitHub is a web-based platform for version control and collaboration, allowing developers to store, manage, and track changes to their code projects using Git.	Github-Wombat
Figma	Figma is a collaborative interface design tool that enables designers to create, share, and iterate on user interface designs in real-time.	Figma-Wombat
Slack	Slack is a communication platform used for seamless collaboration and communication between mentors, clients, and project teams, facilitating real-time messaging, file sharing, and project coordination.	Slack-SiLa
VScode	*Visual Studio Code* is a code editor redefined and optimized for building and debugging modern web and cloud applications.	VScode

Sprint 2 Demo Video

https://youtu.be/m-Q2aA_KJoA?feature=shared

COMP90082-2024-SM1 SI-Wombat Sprint 2 Demo



Project Overview

This section provides an overview of the project which includes SiLA Overview and Background Description and The Project Scope.

Background

The SiLA journal is a renowned international peer-reviewed publication dedicated to the field of language assessment and testing. Despite its significant contributions to academia, SiLA has been operating with an email-based system for managing manuscript submissions and peer reviews. This manual process has become less efficient due to the increasing number of submissions and the intricate demands of the peer review process. The need for an automated, more organized system has become evident to ensure timely and effective management of submissions and reviews.

Goals

Our goal is to establish a fully functional academic journal website that can streamline the process of manuscript submission.

In Scope

1. **User Registration and Management:**

- Implement a secure login system for authors, reviewers, and editors.
- Develop dynamic user profiles that accurately track and display individual submission histories, review assignments, and editorial decisions.

2. **Manuscript Submission:**

- The system is intended to provide a seamless submission process, ensuring that authors can easily upload their work without encountering technical difficulties. A key feature of this platform is the automatic generation and dispatch of confirmation emails to authors immediately after their submission has been successfully received.

3. **Manuscript Tracking and Management:**

- A dashboard for editors/editorial assistants to track manuscript statuses (submitted, under review, revisions required, accepted).
- Assignment of manuscripts to appropriate reviewers based on expertise. The website must ensure the author can not review their own manuscript as a reviewer.

4. **Peer Review System:**

- Double-blind review options - There will be a submission guide for authors, which reminds them that is a double-blind review, and they are not allowed to publish their personal information. In addition, authors will be required to fill out the domain conflicts when they submit their articles.
- Space for reviewers to upload their reports and recommendations.
- Communication module for reviewers to interact with editors/editorial assistants if needed.

5. **Editorial Decision-Making Tools:**

- Facilitates editors in recording verdicts and making decisions on SiLA journal submissions, offering options to accept, request revisions, or reject.
- Proactively updates authors with editorial decisions and detailed feedback, streamlining communication within the manuscript submission process.

6. **Data Security and Confidentiality:**

- Robust data protection measures to ensure the confidentiality of manuscripts and reviews.

- Implement role-based access controls to ensure only authorized users can access relevant data and functionalities. For instance, reviewers should not access other reviewers' comments, and authors should only see information related to their submissions.
- Protect data stored on servers with encryption to prevent unauthorized access.

7. **Scalability and Flexibility:**

- Ensure the platform’s architecture can easily adapt to increasing submissions and user growth
- Maintain a modular system design to facilitate easy updates and the introduction of new features

8. **Integration Capabilities:**

- Establish interfaces for integration with academic indexing and archiving services to broaden the journal’s reach.
- Ensure compatibility with existing academic software ecosystems for streamlined workflows.

9. **Analytics and Reporting:**

- Implement tools for tracking and analyzing submission patterns and review timelines, offering strategic insights.
- Enable customizable reporting features for editors to monitor and improve operational efficiency and user engagement.

Out of Scope

1. **Direct Marketing and Distribution of Published Articles**: Academic platforms focus primarily on the dissemination and archiving of scholarly work rather than on the marketing aspects commonly associated with commercial publications. Marketing efforts like social media campaigns or email promotions may detract from the platform's primary mission of scholarly integrity and impartiality. These activities also require different expertise and resources, which might dilute the platform's focus and investment in academic services.
2. **Financial Management Systems**: Incorporating financial transactions directly into an academic platform can complicate the primary objective of scholarly dissemination with commercial operations such as payment processing and financial tracking. These functions require compliance with various financial regulations and security standards, which can be resource-intensive. Utilizing external systems for these tasks ensures that the platform can focus on academic publishing without the overhead of financial management.
3. **Hosting of Non-Academic Content**: The main goal of an academic platform is to support the research community by providing a repository for scholarly articles. Introducing non-academic content, such as personal blogs or commercial advertisements, could undermine the platform's credibility and distract from its educational purpose. Keeping the content strictly academic ensures that the platform remains a dedicated resource for researchers and scholars.

Requirements

This section provides the detailed requirements of the project, including the system personas, user story, Do-Be-Feel, our Goal model, and Prototype and Client Requirements.

Persona

This section introduces 4 types of persona based on the discussion from the client meeting, which are author, reviewer, editor, and web administrator. Those 4 persons are also the major stakeholders in the SiLA project.

Do-Be-Feel

This section shows the motivation model for our major stakeholders which is listed above and includes their roles, expectations, and emotions.

Goal model

This section contains the Goal Model for the stakeholders which includes their roles, expectations, and emotions.

User Story

This section lists all possible user stories that our development team planned, it provides detailed illustrations and justification for the user story which includes story point, size estimation, and priority (importance).

Prototypes

This section displays the prototypes of the SiLA project that are designed on Figma. The prototypes show the home page, login page, register page, dashboard for author, editor, and reviewer, submission page, reviewing page, and chat box.

Client Requirement

This section lists the client requirements that we collect from client meetings, and we list our solution for each requirement.

Personas

The personas section mainly discusses the personas in project SiLa. From the client's description and development team discussion, there are four major stakeholders: Author, Reviewer, Editor, and Web administrator. The following pages will detailly exhibit those four personas.

Author

Nicholas Brea - Author

"Publishing the article on SiLa is one of my biggest goals in my academic life, and I'm always working hard on it."

Age: **25**
Work: **PhD candidate in the University of Melbourne**
Family: **Married**
Location: **Melbourne, Vic**

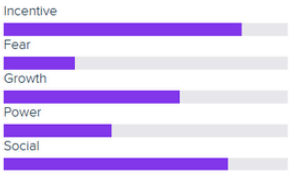
Goals

- Publishing the Article on his focused area.
- Get feedback from experienced reviewers and editors to improve writing skills

Major Responsibilities

- Include research and studying for the focused area and writing the journal publishing on the SiLa.

Motivation



Physical, Social, and Technological Environment:

- Currently, study at the University of Melbourne on Campus.
- Studying and working in a research group with a professor.
- Mainly stayed in the university laboratory for the experiment, and wrote the journal in the library.

Frustration:

- Difficulty in navigating the manuscript submission interface, leading to confusion about how to properly submit manuscripts and supporting documents.
- Experiencing delays in receiving confirmation emails after manuscript submission, causing uncertainty about the submission status.
- Lack of clear communication or updates on the manuscript review process, resulting in anxiety and frustration about waiting times.

Reviewer

Snow - Reviewer

"Efficiency, clarity, and confidentiality in the review process not only save time but also uphold the integrity and quality of scholarly work."

Age: **30**
Education: **Professor**
Work: **PhD candidate in the University of Melbourne**
Family: **Not married yet**
Location: **Melbourne, Vic**

Goals

- Reviewers aim to easily access and manage their assigned manuscripts through a user-friendly dashboard that displays their current review tasks, deadlines, and statuses.
- They seek to communicate effectively with editors and authors, providing clear and constructive feedback through the platform. This includes the ability to ask clarification questions and receive updates or responses.

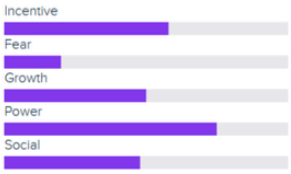
Major Responsibilities

- Conduct objective reviews of manuscripts submitted to SiLA.
- Provide constructive feedback to authors to improve the quality of their research and presentation.
- Make recommendations to the editorial team regarding the suitability of manuscripts for publication.

Physical, Social, and Technological Environment:

- Works primarily from home.
- Engages in scholarly discussions via academic forums, social media groups dedicated to language assessment, and peer collaboration.
- Familiar with using online manuscript submission and review systems and experienced with various digital communication platforms for collaborating with colleagues and participating in virtual conferences.

Motivation



Frustration:

- Challenges in matching their expertise with the assigned manuscripts, resulting in a longer review process or difficulty in providing thorough reviews.
- Difficulties in using the platform for uploading reports and recommendations due to a non-intuitive interface.
- Inadequate communication tools for discussing specific issues with editors or editorial assistants, leading to inefficiencies in the review process.

Editor

KK - Editor

"I will bring your manuscript to life and ensure our journal is the premier choice for showcasing your work."

Age: 40
Education: **Professor**
Work: **PhD candidate in the University of Melbourne**
Family: **Not married yet**
Location: **Melbourne, Vic**

Goals

- ensuring that all submitted manuscripts are being processed properly.
- to effectively manage the peer review process by selecting and inviting qualified reviewers.
- to make well-informed editorial decisions on manuscripts based on peer review feedback.

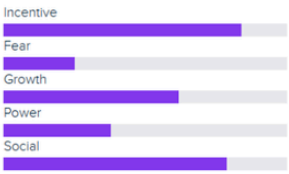
Major Responsibilities

- Manuscript Assessment: evaluate the submitted manuscripts, this includes a check for plagiarism and compliance with the submission guidelines.
- Peer Review Management: oversee the peer review process, including selecting and inviting appropriate reviewers, and monitoring the review progress.
- Decision Making: make decisions on manuscripts based on peer review feedback, including accept, request revisions, or reject.

Physical, Social, and Technological Environment:

- Physical environment: editors work in office settings equipped with computers and necessary software for manuscript handling and communication.
- Social environment: integrate with authors, reviews, publishers and editorial board.
- Technological environment: Editors use specialized software and online platforms for manuscript submission, peer review management, plagiarism checks, and editorial decision-making.

Motivation

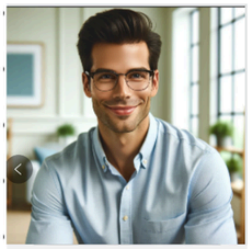


Frustration:

- Overseeing the entire submission and review process manually, including tracking manuscript statuses and coordinating between authors and reviewers, which can be time-consuming and prone to errors.
- Making editorial decisions without an efficient way to compile and view all reviewer reports and recommendations in one place.
- Challenges in maintaining data security and confidentiality, especially when handling sensitive information or feedback.

Web Administrator

Ken - Web Administrator



"Ensuring the SiLA platform runs smoothly and securely is my top priority. It's about creating a safe, efficient environment where researchers and editors can focus on their work, knowing the technical side is taken care of."

Age: **35**
Education: **Bachelor**
Work: **Bachelor**
Family: **Married**
Location: **Melbourne, Vic**

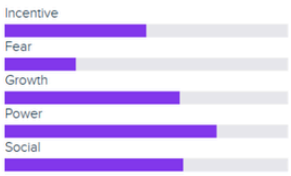
Goals

- Maintain high availability and reliability of the SiLA platform.
- Protect sensitive data and ensure the platform meets the latest security standards.
- Efficiently manage user accounts and permissions, providing prompt support when issues arise.

Major Responsibilities

- include the daily maintenance of the site, ensuring data security, managing user accounts, and providing technical support to both the editorial team and users.

Motivation



Physical, Social, and Technological Environment:

- Works primarily from home, but also maintains close communication with user groups.
- Regularly interacts with users and the editorial team to address their technical inquiries and needs.
- Utilizes a variety of tools for network monitoring, security analysis, and user management, staying abreast of technological advancements and security trends.

Frustration:

- Ensuring the platform's scalability and flexibility to accommodate an increasing number of submissions without compromising performance.
- Implementing robust data protection measures that adequately safeguard manuscript and review information against breaches.
- Continuously updating and adding new features to the platform based on user feedback, which requires staying ahead of technological advancements and user needs.

DO-BE-FEEL

Roles	Function Goal	Quality Goal	Emotional Goal
Editor	• Ensuring that all submitted manuscripts are being processed properly. • To effectively manage the peer review process by selecting and inviting qualified reviewers. • To make well-informed editorial decisions on manuscripts based on peer review feedback.	• Ensure accuracy and thoroughness in editorial decisions	• Feel confident and empowered in managing the editorial process
Reviewer	• Reviewers aim to easily access and manage their assigned manuscripts through a user-friendly dashboard that displays their current review tasks, deadlines, and statuses. • Reviewers seek to communicate effectively with editors and authors, providing clear and constructive feedback through the platform. This includes the ability to ask clarification questions and receive updates or responses.	• Ensure fairness and integrity in the peer review process	• Feel engaged and respected as a valued contributor
Author	• Easily submit manuscripts and track their status • Publishing the Article on his focused area. • Get feedback from experienced reviewers and editors to improve writing skills	• Ensure visibility and transparency in the submission process	• Feel supported and informed throughout the publication journey
Administrator	• Maintain high availability and reliability of the SiLA platform. • Protect sensitive data and ensure the platform meets the latest security standards. • Efficiently manage user accounts and permissions, providing prompt support when issues arise.	• Ensure reliability and confidentiality of user data	• Feel empowered and in control of the platform's operations

Editor:

1. Efficiently manage manuscript submissions and peer review process:
    a. Streamline the process of receiving, assigning, and tracking manuscripts.
    b. Facilitate seamless communication between authors, reviewers, and the editorial team.
    c. Enable quick decision-making on manuscript acceptance/rejection

Reviewer：

1. Provide comprehensive and constructive feedback on manuscripts:
    a. - Conduct thorough evaluation of manuscript content, methodology, and relevance.
    b. - Offer clear and actionable suggestions for improvement.
    c. - Assess manuscripts impartially and with attention to detail.
2. Ensure fairness and integrity in the peer review process:
    a. - Maintain anonymity in reviewing process where required.
    b. - Avoid bias and conflicts of interest in evaluating manuscripts.

c. - Uphold ethical standards and guidelines set by the journal.

3. Feel engaged and respected as a valued contributor:
    a. - Receive recognition for expertise and contributions.
    b. - Appreciate clear guidelines and expectations for the review process.
    c. - Experience satisfaction in contributing to the advancement of scholarly work.

Author:

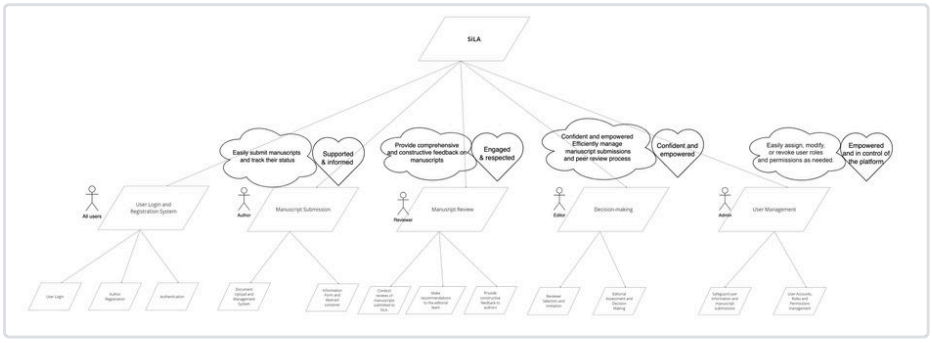
1. Easily submit manuscripts and track their status:
    a. - Navigate an intuitive submission interface.
    b. - Receive timely updates on manuscript progress and review status.
    c. - Access submission history and feedback for reference. |
2. Ensure visibility and transparency in the submission process:
    a. - Understand submission guidelines and requirements clearly.
    b. - Have access to submission history and review reports.
    c. - Receive prompt notifications on editorial decisions.
3. Feel supported and informed throughout the publication journey:
    a. - Receive guidance and assistance in preparing and submitting manuscripts.
    b. - Experience responsiveness and professionalism from the editorial team.
    c. - Celebrate achievements and milestones in the publication process.

Administrator

1. Ensure reliability and confidentiality of user data:
    a. - Safeguard user information and manuscript submissions from breaches or leaks.
    b. - Comply with data protection regulations and industry standards.

GOAL MODEL

This page contains the Goal Model for the stakeholder which include their roles, expectation to do, and emotions.



User Story

In this project, we estimate the **Story Point** by the time cost of the task. In each task, we estimate the time cost and assign the story point based on the three scale: **1 point** for the task that could done within 1 hour. **2 points** for the task that should spend half a day (4 hours). **4 points** for the task that took one day (8 hours or longer). By adding all the task's story points in a user story, we can get the story point for the user story.

For Size Estimation, we have 3 scales - **small, medium,** and **large** as follows:

Small - the user story with. story points less or equal to 5.

Medium - the user story with. story points larger than 5 and less or equal to 10.

Large - the user story with. story points larger than 10.

For **MoSCoW priority**, the '**Must have**' user story is the mandatory user story in this project, detailly all the 'must have' user stories are the key function or base function of the SiLA project. The missing 'Must Have' user story will cause the incomplete of the project, and might affect the development of other related functions.

The '**Should have**' user story is the user story that is important but not mandatory. Unlike the 'Must have' user story, The missing 'Should Have' user story might not cause the incomplete of the project, and might affect the development of other related functions. But, the missing 'Should Have' will greatly reduce the user experience.

Could-have: These items would be nice to have but are not essential. Still less important than the two preceding categories, these elements are considered a third-level priority. If including them will have negative consequences on cost or meeting deadlines, they should be omitted. It is only when they don't negatively affect other project elements that they should be included.
Will not Have (this time): These items are not essential and can be excluded from the project without jeopardizing its success. Being the lowest priority category, omitting them won't hurt the project and they can be included when project conditions are more favorable.

Epic ID	Epic	Epic Start Date	Epic Due Date	User story ID	User story	AS	I Want To	So that	Story Point	Size Estimation	MoSCoW Priority	Justification
SW-84	Home Page	23-Mar-24	01-Apr-24	SW-16	Home page for All Users	a user	access a welcoming home page	it can ensuring a positive first impression and ease of use	10	Medium	Will not Have	Size Estimation: It's complex enough to require significant design and development time but doesn't involve extensive backend logic complexities.

												MoSCoW priority: The home page is the entry point to the platform, crucial for engaging users right from their first visit.
SW-122	Database Design and Setup	23-Mar-24	25-Apr-24	SW-123	Database for Account system	an administrator	create a robust database schema for the account system	we can securely store and efficiently manage user information, roles, permissions, and sensitive data, supporting the dynamic needs of our platform as it grows.	14	Large	Must have	Size Estimation: the basic component of our system, so that it should take a long time to design and implement. MoSCoW priority: the users' account information must store in a reliable database.
				SW-124	Database for Manuscripts	an administrator	create a database schema that efficiently organizes, stores, and retrieves manuscripts, along with their submission details, review statuses, and associated metadata	authors can submit their work, reviewers can assess submissions, and editors can make informed decisions.	20	Large	Must have	Size Estimation: the basic component of our system, so that it should take a long time to design and implement. MoSCoW priority: the articles' relevant information must store

												in a reliable database.
SW-8	Account Management	01-Apr-24	11-Apr-24	SW-2	Login page for different user types	an user	log in to the system as my specific user group	access different kinds of functions of the website tailored to my role	6	Medium	Must have	Size Estimation: a simple page for user to login will be just fine. MoSCoW priority: the author/reviewer/editor can only use the functions of the website once they have logged into the system.
				SW-5	Register Page for Author	an author	a straightforward registration process	I can create an account and start submitting manuscripts	6	Medium	Must have	Size Estimation: a simple page for author to registe a new account will be fine. MoSCoW priority: Without an account identification, the author could not login to the system.
SW-49	Submit System	12-Apr-24	22-Apr-24	SW-30	Dashboard page for Author to Start New Submission	an author	submit the manuscripts to the journal in an efficient manner	I can publish my article via the system	10	Medium	Must have	Size Estimation: the main function of the system, should take a relatively long time to complete. MoSCoW priority: the

												journal use this website to gather article from authors, this is the only portal for author to submit their manuscripts.
SW-45	Review System	20-Apr-24	03-May-24	SW-14	Review page for Reviewer	A Reviewer	be able to securely log in to the online submission and review platform, see all the relevant comments to articles, submit my own comments, and contact the editors conveniently	I can access and review manuscripts assigned to me, participate fully in the peer review process, provide comprehensive feedback, and communicate to the editors directly and efficiently.	8	Medium	Must have	Size estimation: should take a relatively long time to design the review page and implement the functions. MosCow priority: the basic functionality for reviewers to review an article and give the comment.
SW-50	Decision-Making System	27-Apr-24	04-May-24	SW-135	Decision make page for Editor	An Editor	decide whether to accept an article or not	I can guarantee the quality of the journal.	5	Medium	Must Have	Size Estimation: could reuse some functions from review page, it only adds decision make function. MoSCoW priority: the basic functionality for the editor to decide whether to accept an

												article or not.
SW-36	Dashboard System	01-May-24	18-May-24	SW-26	Dashboard - Reviewer - Review and Score	A Reviewer	have a centralized dashboard to track manuscript statuses (submitted, under review, revisions required, and accepted) and to show my assigned tasks	I can efficiently manage the review process and stay organized with my responsibilities.	12	Large	Must have	Size Estimation: It is a sizable task due to its complexity and requires significant resources. MoSCoW priority: Since it is for efficient task management and communication among team members.
				SW-28	Dashboard - Reviewer - History	A Reviewer	have a dashboard to display all edited manuscripts	I can access and review them easily, facilitating collaboration and ensuring efficient processing.	8	Medium	Must have	Size Estimation: Developing these feature is a sizable task due to its complexity and requires significant resources. MoSCoW Priority: It is essential for facilitating collaboration among team members and ensuring efficient review processes.
				SW-17	Dashboard page for Author	An Author	have a user-friendly page that	I can edit and change my personal information.	8	Medium	Should have	Size Estimation: Creating a user-friendly page for

							displays the information					displaying and editing personal information involves moderate effort. MoSCoW priority: As it enhances user experience by allowing authors to easily manage their personal information, though it's not critical for immediate functionality.
				SW-18	Dashboard page for Editor	An Editor	have a dedicated section to record the number of those decisions (accept, revise, reject)	my decisions are clearly communicated to the authors.	8	Medium	Must have	Size Estimation: It needs design the interface and implement to display data on the dashboard. MoSCoW priority: t should show all the data about articles in the dashboard, which is also the requirement of client.
SW-51	Communication System	01-May-24	18-May-24	SW-118	Automatic confirmation emails for	A author	receive the confirmation email	I can ensure I submit the article successfully.	5	Medium	Should have	Size estimation: it should design the

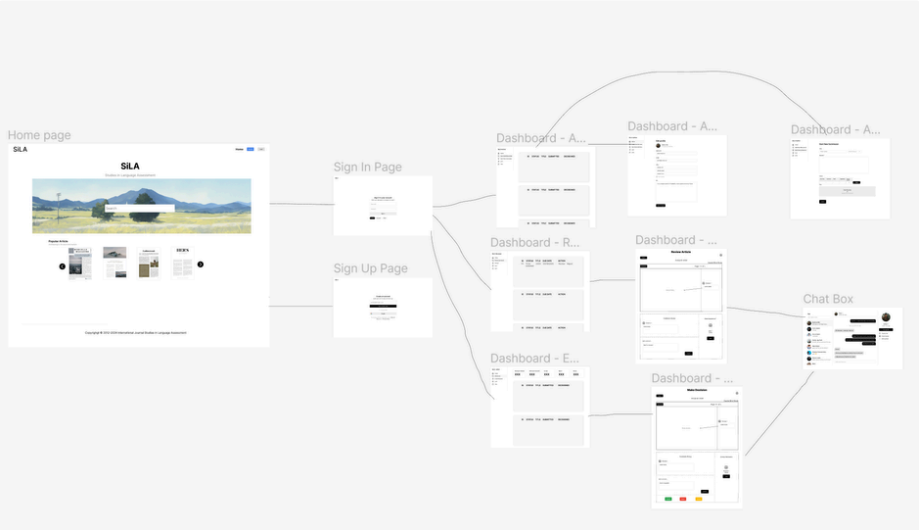
					successful submission		from the SiLA				email format and integrate with submission system. MoSCoW Priority: important function which also be required by client.
				SW-67	Chat Functionality for Reviewer-Editor Communication	A reviewer or editor	be able to engage in text-based communication within a dedicated pop-up chat window on the platform	I can discuss manuscripts, share feedback, and clarify queries directly and efficiently.	10	Medium	Should have	Size estimation: chatting each other should design the interface and the backend environment. MoSCoW Priority: important function because authors need to get feedback from reviewer.
SW-140	Navigation Component	04-May-24	18-May-24	SW-141	Role-Based Dynamic Navigation for logged-in users	A logged-in user	dynamically adjust based on my permissions and role (author, reviewer, editor)	I can easily access modules specific to my tasks, including Submission, Profile, Review, Decision-Making, and Communication, enhancing my efficiency and	8	Medium	Should have	Size Estimation: a simple toolbar that navigates the user through the website. MoSCoW priority: The navigation component is important but it is not vital to the system.

								experience on the platform.				
SW-116	Data Security	04-May-24	19-May-24	SW-76	Data Security and Confidentiality	a system administrator	implement robust data protection measures for manuscript and review confidentiality	sensitive data is safeguarded from unauthorized access or disclosure, instilling user trust and upholding review process integrity.	8	Medium	Should have	Size Estimation: It involves moderate effort due to the complexity of encryption protocols, access controls, and audit trails. MoSCoW Priority: As it enhances data security and instills user trust, although it's not critical for immediate functionality.

Prototypes

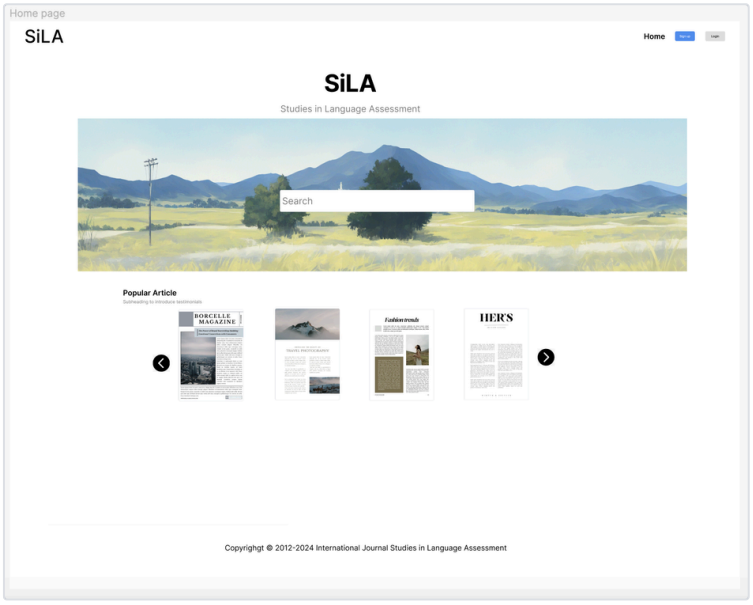
This Section displays the prototypes of SiLA:

Flow of the Prototypes:

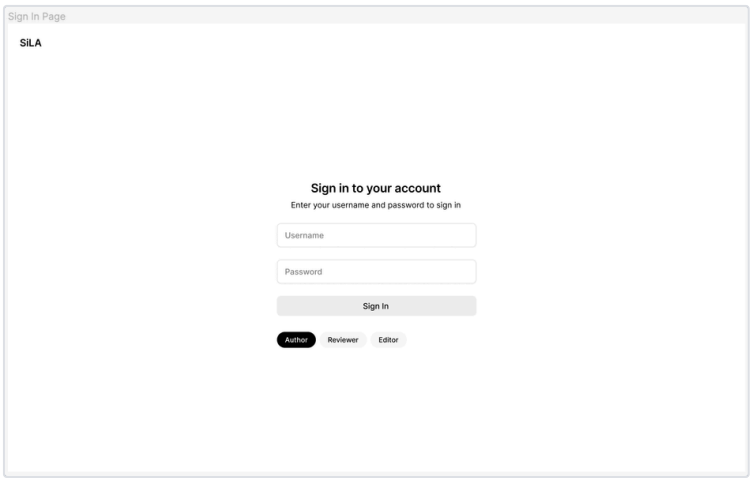


Page Detail

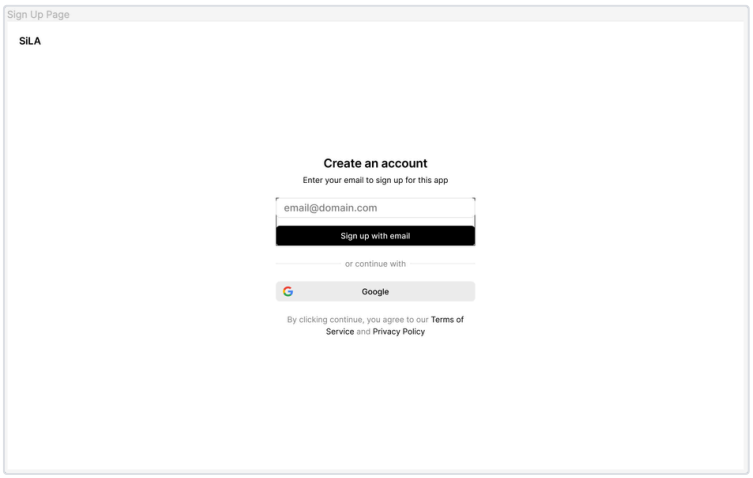
Home page



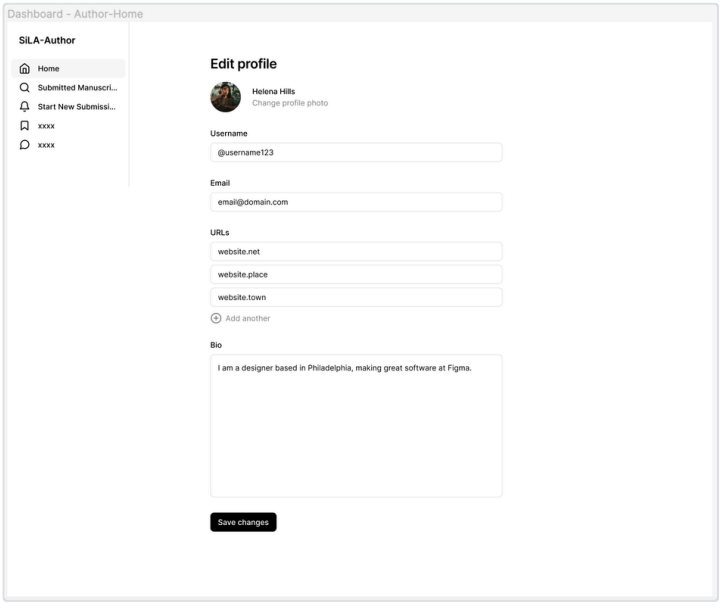
Sign In page



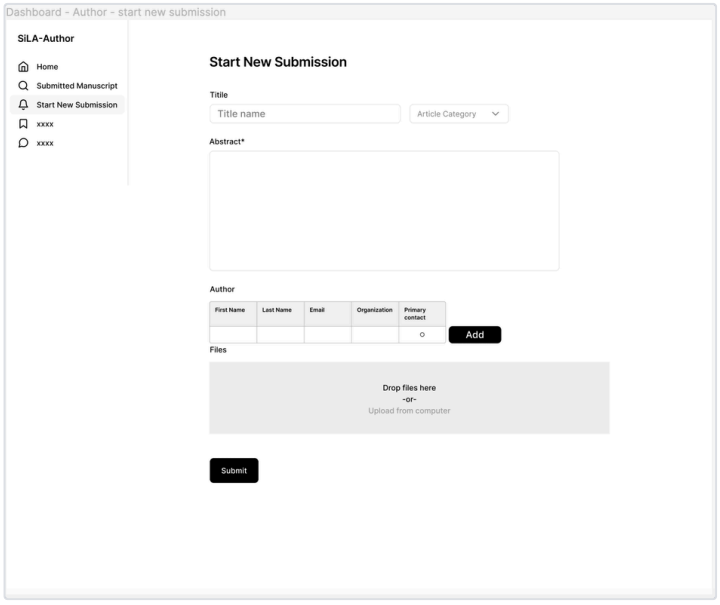
Sign Up page



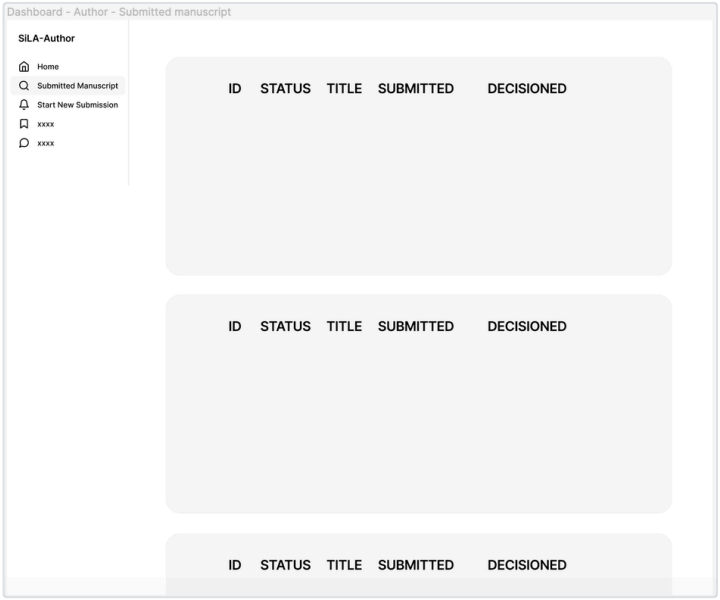
Dashboard for Author home



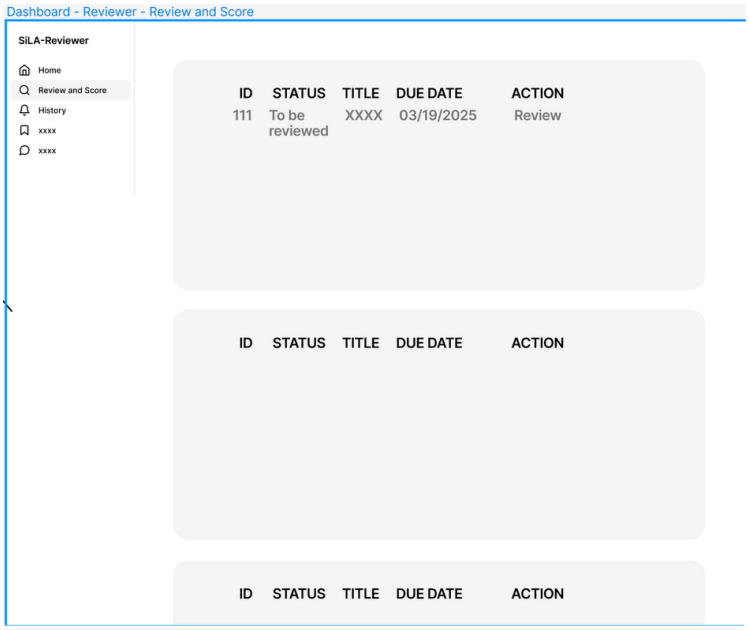
Dashboard for Author - start new submission



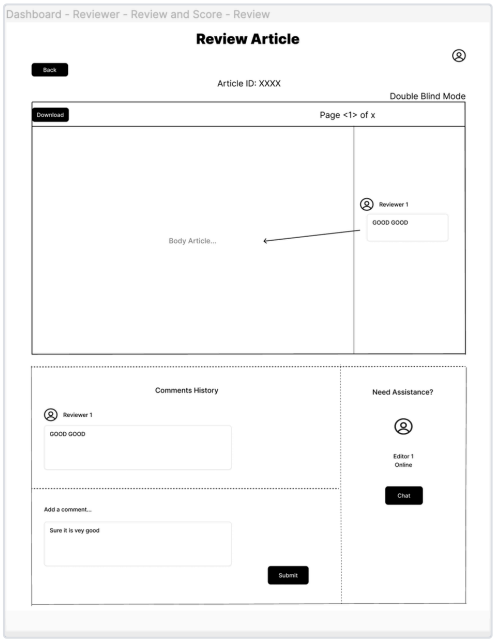
Dashboard for Author - Submitted manuscript



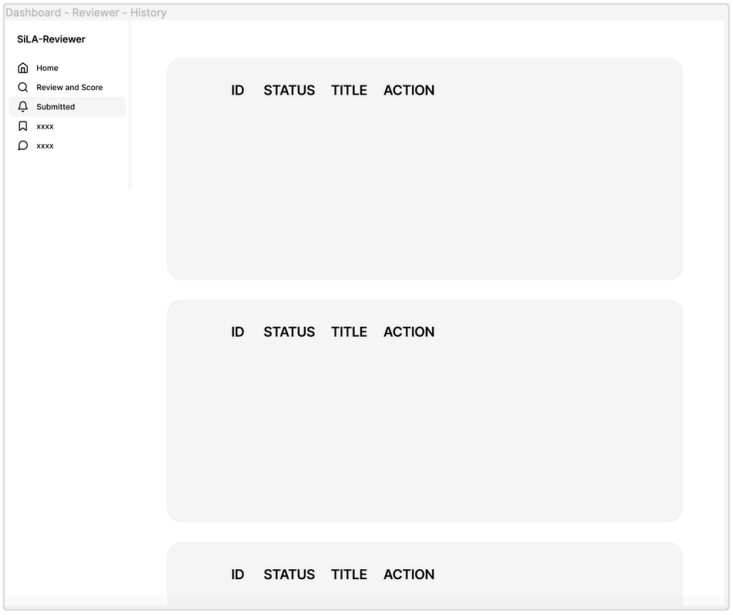
Dashboard for Reviewer - Review and Score



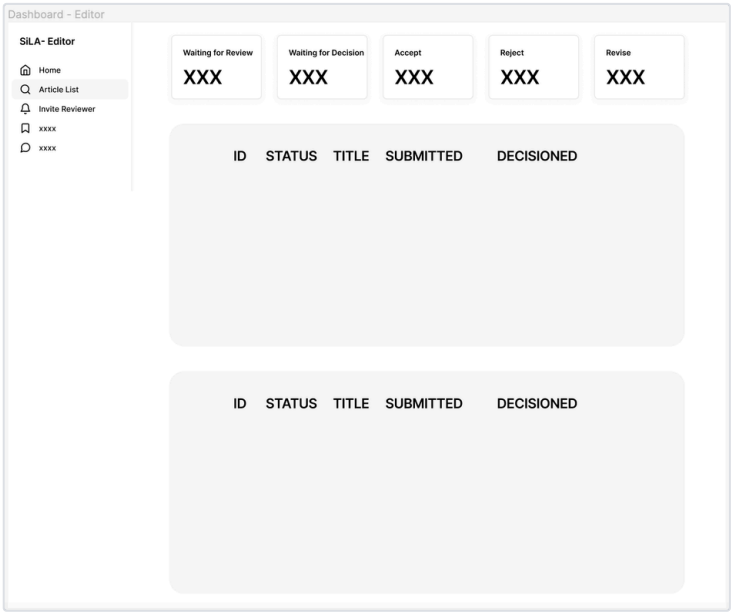
Dashboard for Reviewer - Review and Score - Review



Dashboard for Reviewer - History



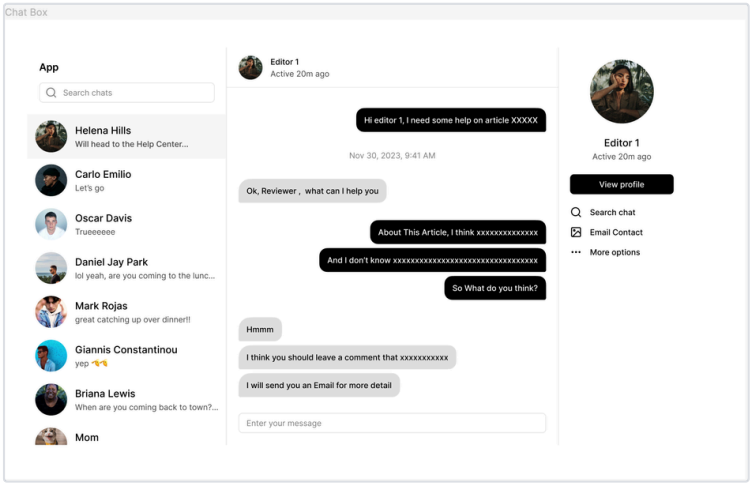
Dashboard for Editor



Dashboard for Editor - Decision Page


Dashboard - Editor - Decision page
Make Decision
Back
Article ID: XXXX
Double Blind Mode
Download
Page <1> of x
Reviewer 1
0000 0000
Body Article...
Comments History
Reviewer 1
0000 0000
Add a comment...
Sure it is very good
Submit
Accept
Reject
Revise
Contact Reviewers
Reviewer 1
Online
Chat

Chat Box


Chat Box
App
Search chats
Helena Hills
Will head to the Help Center...
Carlo Emilio
Let's go
Oscar Davis
Trueeeeee
Daniel Jay Park
lol yeah, are you coming to the lunc...
Mark Rojas
great catching up over dinner!!
Giannis Constantinou
yep
Briana Lewis
When are you coming back to town?...
Mom
Editor 1
Active 20m ago
Hi editor 1, I need some help on article XXXXX
Nov 30, 2023, 9:41 AM
Ok, Reviewer , what can I help you
About This Article, I think xxxxxxxxxxxxx
And I don't know xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
So What do you think?
Hmmm
I think you should leave a comment that xxxxxxxxxx
I will send you an Email for more detail
Enter your message
Editor 1
Active 20m ago
View profile
Search chat
Email Contact
More options

Client Requirement

This section lists the client requirements that we collect at the client meeting. The client requirement is one of the most important guidelines for the project development.

Ver.1

The version 1 requirement provided a brief development guideline for the SiLA project. This section lists client requirements including account roles, avoiding self-review, special issue submission, Admin role, double-blind submission, decision on double-blind submission, reviewer template, and reviewer requirement.

Ver.2

The version 2 requirement provided a more detailed development requirement for each page and role which includes the Manuscript submission page, Editors’ dashboard, Authors’ dashboard, and Reviewers’ dashboard.

Client Requirement ver.2

The version 2 requirement provided a more detailed development requirement for each page and role which includes the Manuscript submission page, Editors' dashboard, Authors' dashboard, and Reviewers' dashboard.

General comments:

1. Reviewers will be signed up by editors/editorial assistant
2. User details during sign-up (all mandatory): first name, last name, email address, institution/affiliation (users should be able to change these themselves at any given time)

Manuscript submission page

- Include progress bar: Author details (see Q1 below) – Submission details (see Q2-5) – Conflict of interest (Q6) – Attach files (Q7)
- Note to authors at top of the submission page:

You will need the following documents to complete your submission:

- Full names and affiliations of all authors
- Anonymized manuscript
- Appendices (if applicable)
- Supplementary materials (if applicable)
- Fields marked * below should be mandatory

1. Author Details
    a. Author 1: first name*, last name*, affiliation*, academic title*, ORCID, q: is this the corresponding author (tickbox)?
    b. Author 2: idem
    c. Etc. (up to 8?)
2. Type of submission (drop-down menu)*
3. Title of paper*
4. Abstract* (include word count if possible)
5. Acknowledgments and/or funding information (text box)
6. Conflict of interest declaration* (yes/no tick box, we can provide you with a generic declaration)
    a. If yes, please specify
7. Attach files, dropdown menu with the following options:
    a. Anonymized manuscript (incl. tables, figures, and appendices)
    b. Other, add a text box for authors to specify what the document is

Editors' dashboard

1. Receive notification of new submission
2. Review submission: desk reject or external review
    a. If desk rejects: editors should be able to include comments, authors will receive outcome + comments from editors once released
    b. If external review: editors will approach potential reviewers within the platform
3. Send anonymized paper to external reviewers
4. Receive a notification when the review is in
5. Receive notification when both reviews are in ('both reviews have now been submitted)
6. Decide: accept (ACC), accept with minor revisions (AR), revise&resubmit (RR), reject (RJ)

7. Release anonymous reviews, decisions, and additional comments to the author(s)
    a. If revisions: specify the deadline
8. Receive a notification when a revised manuscript has been submitted
9. Decide: accept (ACC), accept with minor revisions (AR), revise&resubmit (RR), second-round reviews (RV2)
    a. ACC/AR/RR: send notification + comments to authors, specify deadline
    b. RV2: send revised manuscript back to reviewers, specify the deadline
10. Receive notification when resubmission/external reviews are in
11. Repeat steps 7-9

Author's dashboard

1. Submission is made (as per above)
2. Receive notification: desk-reject or external review
    a. If desk-reject: should include comments from editors
3. Receive notification when editors have made a decision and have sent through reviews
4. Accept new deadline/request extension/withdraw submission
5. Submit revised manuscript
6. Repeat steps 3-5

Reviewer's dashboard

1. Receive notification of review request
    a. Title of manuscript
    b. Abstract
    c. Proposed deadline
2. Decide: accept, accept but propose a new deadline, decline. They should be able to leave comments as well.
3. If accepted: Receive anonymized manuscript
4. On the review page there should be:
    a. Up the top of the page, the deadline and reviewing guidelines should be visible at all time
    b. Text box with comments for editors
    c. Text box with comments for authors
    d. Tick box (yes/no): Would you be willing to review a revision of this manuscript?
    e. Possibility to add comments to the manuscript and/or download a Word version, work on it, and upload it again
5. Receive a reminder 1 week prior to the deadline
6. If needed, receive a reminder 2 days after the deadline (to allow for different time zones)
7. Reviewers should be able to save progress and come back later

Note: notifications and reminders should be received at email address

Development

Development environment

The tools and development environment will be listed and introduced in this section which includes Confluence, GitHub and Technical Details.

Confluence

The project uses confluence to streamline our document. Confluence is a collaborative workspace where teams can efficiently create, share, and update documents and information in a centralized location.

GitHub

GitHub is a web-based platform for version control using Git, primarily used for computer code. It provides a collaborative environment for developers to host and review code, manage projects, and build software alongside millions of other developers.

The GitHub address of our project is https://github.com/COMP90082-2024-SM1/SI-Wombat RESTRICTED CONTENT

Workflow

- master: The `master` branch is the primary branch where the source code reflects the production-ready state of this project. It is the definitive branch where code is fully tested, stable, and ready to be relseaed to end-users.
- dev: The `dev` branch serves as the active development branch where new features and fixes are integrated and tested. The beanch contains the latest ongooing work and once stable, changes are merged into the `master` branch for release.

Branch Naming Conventions

- Task Branches: Prefix with `task/` followed by a brief description of the feature, e.g., "task/login".
- Bug Fixes Branches: Use `bugfix/` along with the a short description, like bugfix/fix-login-error.

Technical Details

Framework and Programming Language:

- Front-End Development Environment:
  - React: is a popular JavaScript library for building user interfaces. It will be used to create dynamic and responsive UI components for the submission and review platform.
  - TypeScript: is a statically typed superset of JavaScript, providing enhanced tooling and type safety. It will help catch errors early in the development process and improve code maintainability.
- Back-End Development Environment:
  - C#: is a robust and versatile programming language, well-suited for building scalable and performant backend systems. It will be used to implement the server-side logic and data management for the submission and review platform.
  - Dot Net: is a cross-platform framework for building various types of applications, including web applications. It provides libraries and tools for rapid development and deployment of backend services.
- IDE:
  - Visual Studio Code: is a lightweight and versatile code editor with excellent support for React and TypeScript development. It will facilitate efficient coding and debugging workflows.

Additional Tools:

- Version Control: Git
- Hosting Service: AWS
- Database**:** MySQL

Environment Setup:

1. Set up React project using create-react-app.
2. Install necessary TypeScript typings and configurations.
3. Install .NET SDK and Visual Studio Code extensions for C# development.
4. Initialize a new .NET project for backend development.

Collaboration:

- Communication: Utilize Slack for real-time communication among team members.
- Version Control: Use Git for version control, with GitHub as the remote repository.
- Code Review: Implement a code review process using GitHub pull requests.

GitHub Process

We will discuss the method of how we use GitHub on this page.

1. Main Branching Structure

In the beginning, we created a `dev` branch out of `master`.

1.1 Main branch

The `master` branch is used as the main branch where the source code of HEAD always reflects a production-ready state. It is the definitive branch where final releases are cut from, ensuring stability and reliability in our project production environment.

1.2 Developing branch

The `dev` branch, aka the development branch, serves as an active development environment where developers merge new features and changes. This branch acts as a preparation branch before changes are moved to the master branch, allowing for testing and adjustments without affecting the stable version.

2. Branching Strategy

We break the project into several small features. For each new feature, we create a new branch for it such as `` `auth-manuscipts-userinfo-relevant-backend-endpoint` ``. Codes relevant to this new feature will be developed in this new branch. In order to distinguish between frontend and backend environment. The branch corresponding to the front-end will be denoted by “frontend“, and “backend“ for backend.

Once a new feature is tested and done successfully, a pull request will then be created for code reviewing. The reviewers will be able to leave comments for that pull request on the GitHub website, a detailed explanation of `pull request` is introduced in the following chapter. If all reviewers approve, the corresponding branch will be deleted from GitHub, both locally and remotely.

3. Structure of Pull Request:

We have a doc structure for Pull Request. The following picture illustrates an example of Pull Request.

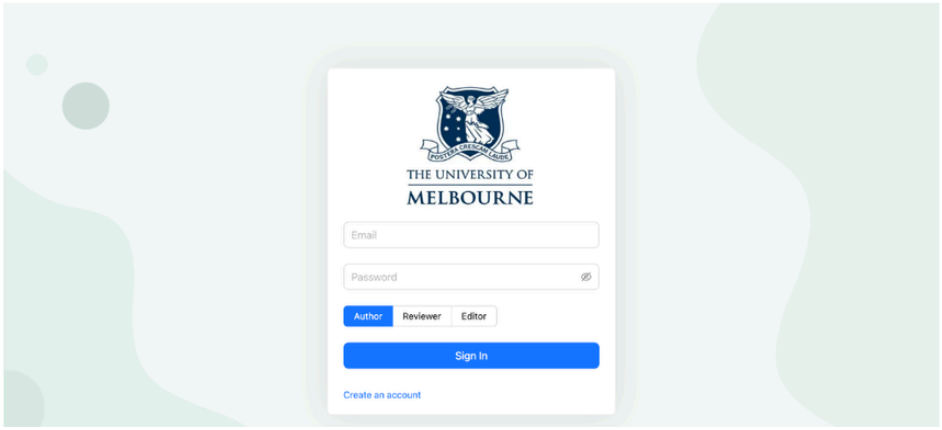
GG2T commented 2 weeks ago • edited

Close

Close issue

- [x] [Login Page for different user types] https://wombat123.atlassian.net/browse/SW-2?atlOrigin=eyJpIjoiZGEzODMwM2NmZDRkNDI0ZGI5NTQ1NzFjNTIyY2M4NDIiLCJwIjoiaiJ9
- [x] [Register page for author] https://wombat123.atlassian.net/browse/SW-5?atlOrigin=eyJpIjoiNGUyYWNkOTIxMDc5NDViYWIwNzhlMzhiYTBhOTMyYTAiLCJwIjoiaiJ9
- [x] [Login and Register api endpoint]

Screenshot



Source Code Structure

For this project, we developed the front-end and back-end separately, this page introduces the main functionalities of each document in the system.

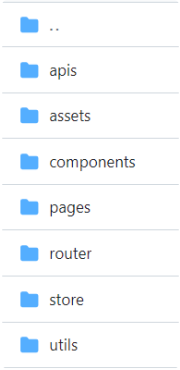
1. The root directory of the project

The root directory contains two documents, which are `docs` and `src`. The docs file consists of different kinds of documentation for this project, including confluence files and images.

2. Front-end files system

Codes associated with front-end are included in `src/front_end/sila-frontend`, where the `public` file stores the static resource and `src` contains the source code.

The following image reveals the detailed directory structure of `src/front_end/sila-frontend/src`.

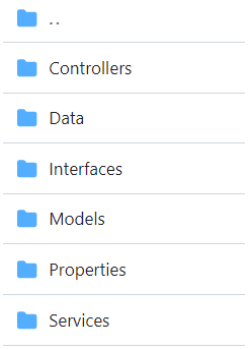


Folder	Functionality
`apis`	This directory contains all the API-related code. It is where the front-end interacts with back-end services, handles API requests and manages responses.
`assets`	Static files such as images, fronts and any other files that are part of the visual resources of the project can be found here.
`components`	This holds all the React components that we use to build the user interface. Each component is a piece of the UI that can be reused throughout the application.
`pages`	It includes the files that represent the different pages or routes in the application. Typically, each file corresponds to a different view that users can navigate to.
`router`	In this directory, we configure the routing for the application, defining how to navigate between the pages/components based on the URL in the browser and hyperlinks.
`store`	This is used for state management such as Redux, it will contain actions, reducers, and the store configuration.
`utils`	Contains utility functions and helper code that can be used across different parts of the project. This is where we will place generic functions that are not tied to any specific

	component or page.

3. Back-end files system

The root directory of back-end is `/src/back_end/SiLA-Backend/SiLA-Backend`.



Folder	Functionality
`Controllers`	This directory contains controller classes that handle client requests, process them (possibly using services), and return responses. They act as an intermediary between the `Models` and `Views` in an MVC architecture.
`Data`	Code that interacts with your database, such as database context configurations, migrations, and data seeding files.
`Interfaces`	This folder is for interface definitions, which provide blueprints for the classes. They help to define contracts within the code and ensure that certain classes provide specific public methods.
`Models`	Contains classes that represent the data structures used in the application. These models are typically mirrored by database tables.
`Properties`	This directory usually includes configuration files, such as `launchSettings.json`, which configures project behaviours and environment variables.
`Services`	In this folder, it will place the business logic of the application. Services encapsulate business logic, data access, and computations, and are used by the controllers.

APIs

The below table is the API we have for the SiLA project.

Endpoint	Method	Description	Request Content-Type	Response Codes	Parameters	Body
/Auth/register	POST	Register a new user.	application/json, text/json, application/*+json	200: Success	-	RegisterModel
/Auth/login	POST	Log in an existing user.	application/json, text/json, application/*+json	200: Success	-	LoginModel
/Auth/logout	POST	Log out a user by invalidating their session token.	-	200: Success	id (query), token (query)	-
/Manuscripts/uploadfile	POST	Upload a manuscript file.	multipart/form-data	200: Success	-	file: binary
/Manuscripts/submit	POST	Submit a manuscript with metadata.	application/json, text/json, application/*+json	200: Success	-	ManuscriptSubmissionModel
/User/{userId}	GET, PUT	Retrieve or update user details.	application/json, text/json, application/*+json for PUT	200: Success	userId (path)	UserDTO for PUT
/User/AuthorDashboard/{userId}	GET	Access a specific user's author dashboard.	-	200: Success	userId (path)	-

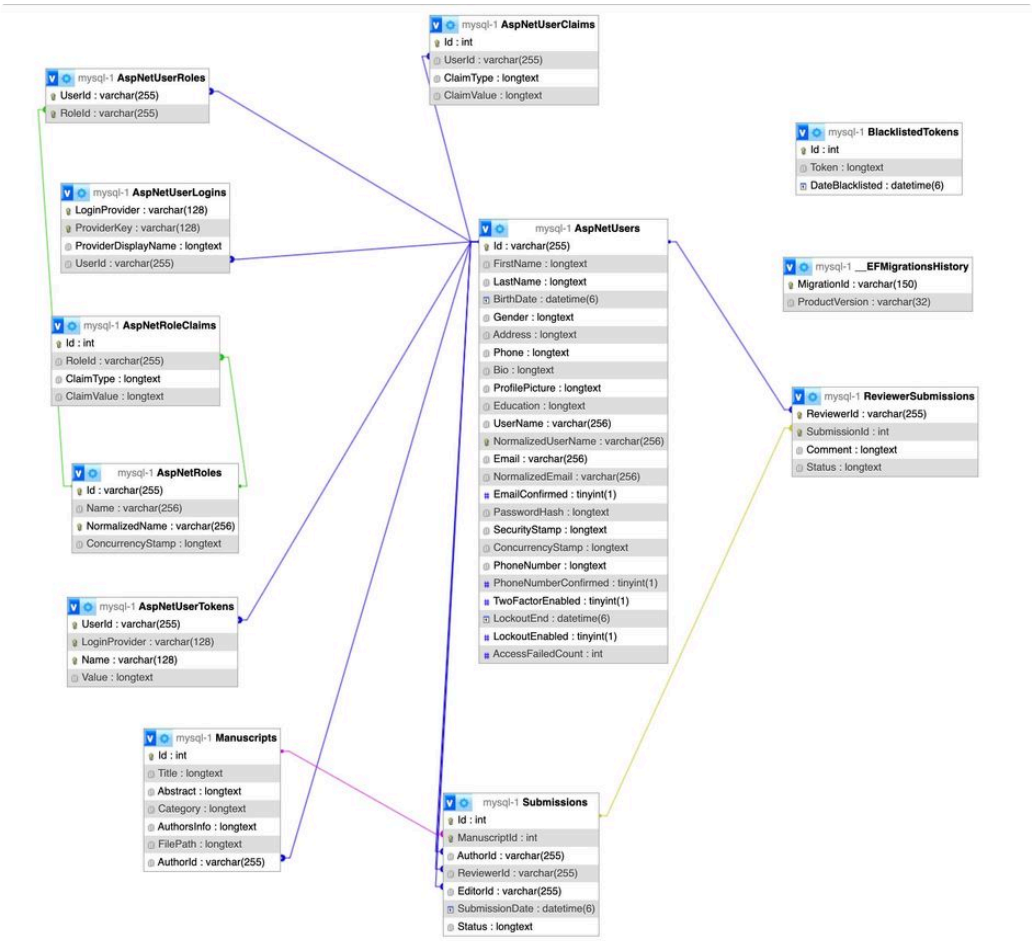
Additional Notes:

- **Parameters**: 'userId' is a path parameter required for the GET and PUT requests under '/User/{userId}' and '/User/AuthorDashboard/{userId}'.
- **Body**: Describes expected schema objects which are linked to detailed schemas defined under components. 'RegisterModel', 'LoginModel', 'ManuscriptSubmissionModel', and 'UserDTO' are the main models used.
- **Content-Type**: Indicates the type of data that should be in the request body. Multiple types are supported for flexibility.

This table is designed to provide a clear and concise overview, making it easier to understand the API's capabilities and requirements without delving into the full complexity of the original Swagger-generated documentation. If needed, more specific details like security requirements, exact schema properties, and additional response messages could be further expanded upon in supplementary documentation or detailed API guides.

Database Development

ER(Entity–relationship) Diagram



Explanation:

This Entity Relationship Diagram (ERD) depicts a part of a manuscript review system's database design, explaining the entities involved and their relationships as follows:

- **AspNetUsers**: This table holds the users in the system, containing essential information such as the unique identifier, username, email, password hash, etc.
- **AspNetRoles**: This is the roles table, defining various user roles such as editor, author, reviewer, etc.
- **AspNetUserRoles**: It's a many-to-many relationship table between users and roles, indicating which users belong to which roles.
- **AspNetUserClaims**: Stores additional claim information for users, which can be used to hold extra attributes of a user.
- **AspNetUserLogins**: Maintains user login information, especially concerning third-party login providers.
- **AspNetUserTokens**: Used to store user token information, like security or authentication tokens.
- **AspNetRoleClaims**: Used to store specific permission claims for roles, aiding in defining the capabilities of a role.
- **BlacklistedTokens**: This appears to be a table used to store tokens that are invalid or have been revoked, important for managing security and user sessions.
- **Submissions**: Represents the table for submission records, likely including manuscripts or reports submitted to the review system.
- **Manuscripts**: Represents the entity for manuscripts or documents, storing title, abstract, category, authorship information, etc.

- **ReviewerSubmissions**: Represents the relationship between reviewers and the manuscripts they review, storing reviewer feedback and status information.
- **EFMigrationsHistory**: Records the database migration history for Entity Framework, important for maintaining the changes in the database structure.

From this ERD, it can be seen that the database design of the review system pays significant attention to user authentication and role authorization, while also focusing on the submission and review processes of manuscripts. Relationships between entities are manifested through foreign key connection tables, ensuring data consistency and integrity.

GPT Code Review

In Sprint 2, our team conducted a detailed review of several key documents and components of the project. Here’s a breakdown of what was reviewed and the feedback received:

Time Reviewed

- About now, we tried about 30 times, but it failed a lot.

Selection Criteria

- We decide to review the code for the key features from the front and back end. The quality of key features could greatly affect the user experience. The GPT could provide some valuable commons that help us improve the code structure which greatly improves the project quality.

Our Decision

- Once the code is reviewed, the group will discuss the GPT suggestion together, and decide rather accept or reject the common. The following section “GPT Code Review [File Name]“ will detailly show the example of GPT Review with our decision.

Documents and Components Reviewed:

- Author Dashboard - The interface that allows authors to view and manage their submissions.
- Login - The authentication mechanism for users to access their specific roles and functions.
- Submission - The process and interface for authors to submit their manuscripts.
- Reviewer Editor FormModal - The forms used by reviewers and editors to provide feedback or make decisions on manuscripts.
- ManuscriptsController - The backend controller manages the requests related to manuscripts.
- Submission Service - The service handling the business logic for manuscript submissions.

Feedback Received:

- Author Dashboard - We will improve code comments and error handling as suggested, and consider pagination based on data growth.
- Login - Refining error handling and enhancing comments. We'll explore modular error management to simplify the logic.
- Submission - Acknowledged the need for better comments and simplification of PDFUploader components. Robust error handling will be prioritized.
- Reviewer Editor FormModal - Useful suggestions on improving documentation and component abstraction. We'll implement enhanced state management and validation.
- ManuscriptsController - Add detailed comments and break down complex lines for clarity. Enhanced error handling for external operations is planned.
- Submission Service - Restructuring database operations and improving API documentation for clarity and security.

GPT Instruction

Please evaluate the {code} below.
Use the following checklist to guide your analysis:

1. Documentation Defects:
   a. Naming: Assess the quality of software element names.
   b. Comment: Analyze the quality and accuracy of code comments.
2. Visual Representation Defects:
   a. Bracket Usage: Identify any issues with incorrect or missing brackets.

b. Indentation: Check for incorrect indentation that affects readability.
c. Long Line: Point out any long code statements that hinder readability.

3. Structure Defects:
   a. Dead Code: Find any code statements that serve no meaningful purpose.
   b. Duplication: Identify duplicate code statements that can be refactored.
4. New Functionality:
   a. Use Standard Method: Determine if a standardized approach should be used for single-purpose code statements.
5. Resource Defects:
   a. Variable Initialization: Identify variables that are uninitialized or incorrectly initialized.
   b. Memory Management: Evaluate the program's memory usage and management.
6. Check Defects:
   a. Check User Input: Analyze the validity of user input and its handling.
7. Interface Defects:
   a. Parameter: Detect incorrect or missing parameters when calling functions or libraries.
8. Logic Defects:
   a. Compute: Identify incorrect logic during system execution.
   b. Performance: Evaluate the efficiency of the algorithm used.
   Provide your feedback in a numbered list for each category. At the end of your answer, summarize the recommended changes to improve the quality of the code provided.

File List

- src/front_end/sila-frontend/src/pages/Author/Home/index.js
- src/front_end/sila-frontend/src/pages/Login/index.js
- src/front_end/sila-frontend/src/pages/Author/StartNewSubmission/index.js
- src/front_end/sila-frontend/src/pages/Editor/FormModal/index.js
- src/back_end/SiLA-Backend/SiLA-Backend/Controllers/ManuscriptsController.cs
- src/back_end/SiLA-Backend/SiLA-Backend/Services/SubmissionService.cs

GPT Code Review Author Dashbord

File: /SI-Wombat/src/front_end/sila-frontend/src/pages/Author/Home/index.js

GPT feedback

ChatGPT 4

ChatGPT

Evaluation of Provided Code:

1. **Documentation Defects:**
   - **a. Naming:**
     - The naming of the function `get_articles_by_author` and the variable `articles` are clear and describe their purposes well. However, `data` in the component could be more specifically named to reflect its role or contents.
   - **b. Comments:**
     - The comments are minimal and could be improved for clarity, especially regarding the logic of the API call and data handling.
2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - No issues with bracket usage were noted in the provided code snippet.
   - **b. Indentation:**
     - Indentation appears consistent and adheres to common practices, facilitating readability.

ChatGPT 4

array, are quite long and could benefit from line breaks for enhanced readability.

3. **Structure Defects:**
   - **a. Dead Code:**
     - There doesn't seem to be any dead code; all parts of the code serve a purpose.
   - **b. Duplication:**
     - The code shows no obvious duplication, and functionality is encapsulated in reusable functions and components.
4. **New Functionality:**
   - **a. Use Standard Method:**
     - The code uses React and Ant Design standards effectively; no changes are recommended here.
5. **Resource Defects:**
   - **a. Variable Initialization:**
     - All variables are initialized properly. The use of `useState` and `useEffect` hooks are appropriately used for managing state and side effects.
   - **b. Memory Management:**
     - The component does not manually handle memory, and given the context of a React application, this is typically managed by the framework.

ChatGPT 4

   - **a. Check User Input:**
     - There is no direct user input handling in the snippet provided; it assumes the API and navigational functions receive correct input.
7. **Interface Defects:**
   - **a. Parameter:**
     - The parameter passing in `get_articles_by_author` and API calls seem appropriate, although the snippet assumes correct API functionality.
8. **Logic Defects:**
   - **a. Compute:**
     - The logical structures seem correct, though the snippet lacks error handling for API calls which could be improved.
   - **b. Performance:**
     - The code uses efficient React patterns such as `useState` and `useEffect` for state management and side effects. However, the lack of pagination or lazy loading in data fetching might impact performance with large datasets.

Summary of Recommended Changes:

- Improve comments to describe more clearly what each major block of code is doing, particularly around API calls and data processing.
- Consider breaking longer lines within the `columns` array for better readability.
- Implement error handling for the API call within `useEffect` to manage cases where the fetch fails or returns unexpected results.
- Add checks for user input where relevant, especially if this code interfaces with broader parts of the application where user input is possible.

- Consider implementing pagination or lazy loading for the article data if performance becomes a concern with large datasets.

Evaluation of Provided Code:

1. **Documentation Defects:**
   - **a. Naming:**
     - The naming of the function `get_articles_by_author` and the variable `articles` are clear and describe their purposes well. However, `data` in the component could be more specifically named to reflect its role or contents.

✅ Accept, we modified the naming conventions as per the GPT suggestions.

1. **b. Comments:**
   - The comments are minimal and could be improved for clarity, especially regarding the logic of the API call and data handling.
2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - No issues with bracket usage were noted in the provided code snippet.
   - **b. Indentation:**
     - Indentation appears consistent and adheres to common practices, facilitating readability.
   - **c. Long Line:**
     - The code is generally well formatted, but some lines, particularly within the `columns` array, are quite long and could benefit from line breaks for enhanced readability.

✅ Accept, we plan to break long lines into several short lines.

1. **Structure Defects:**
   - **a. Dead Code:**
     - There doesn't seem to be any dead code; all parts of the code serve a purpose.
   - **b. Duplication:**
     - The code shows no obvious duplication, and functionality is encapsulated in reusable functions and components.
2. **New Functionality:**
   - **a. Use Standard Method:**
     - The code uses React and Ant Design standards effectively; no changes are recommended here.
3. **Resource Defects:**
   - **a. Variable Initialization:**
     - All variables are initialized properly. The use of `useState` and `useEffect` hooks are appropriately used for managing state and side effects.
   - **b. Memory Management:**
     - The component does not manually handle memory, and given the context of a React application, this is typically managed by the framework.
4. **Check Defects:**
   - **a. Check User Input:**
     - There is no direct user input handling in the snippet provided; it assumes the API and navigational functions receive correct input.

❌ Reject. We handle user's input at back-end portal. It doesn't be displayed at here.

1. **Interface Defects:**
   - **a. Parameter:**

    - The parameter passing in `get_articles_by_author` and API calls seem appropriate, although the snippet assumes correct API functionality.
2. **Logic Defects:**
    - **a. Compute:**
        - The logical structures seem correct, though the snippet lacks error handling for API calls which could be improved.

✅ Accept. We plan to write relevant code to handle error for API calls.

1. **b. Performance:**
    - The code uses efficient React patterns such as `useState` and `useEffect` for state management and side effects. However, the lack of pagination or lazy loading in data fetching might impact performance with large datasets.

❌ Reject. It is not necessary since we have a lot functions need to implement.

Summary of Recommended Changes:

- Improve comments to describe more clearly what each major block of code is doing, particularly around API calls and data processing.
- Consider breaking longer lines within the `columns` array for better readability.
- Implement error handling for the API call within `useEffect` to manage cases where the fetch fails or returns unexpected results.
- Add checks for user input where relevant, especially if this code interfaces with broader parts of the application where user input is possible.
- Consider implementing pagination or lazy loading for the article data if performance becomes a concern with large datasets.

GPT Code Review Login

File: /SI-Wombat/src/front_end/sila-frontend/src/pages/Login/index.js

GPT feedback

ChatGPT 4

ChatGPT

Evaluation of Provided Code:

1. **Documentation Defects:**
   - **a. Naming:**
     - The naming convention used in the code (`fetch_login`, `on_finish`, etc.) is descriptive and indicates the function's purpose clearly.
   - **b. Comments:**
     - The code lacks detailed comments that could help explain complex sections, particularly the error handling logic within the `on_finish` function.
2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - Brackets are used correctly throughout the code.
   - **b. Indentation:**
     - The code maintains consistent indentation, aiding readability.
   - **c. Long Line:**
     - The error handling within the `on_finish` function could be refactored to avoid deeply

ChatGPT 4

   - **a. Dead Code:**
     - No dead code is evident in the snippet provided.
   - **b. Duplication:**
     - There is no significant duplication in the code; the structure is streamlined and functional.
4. **New Functionality:**
   - **a. Use Standard Method:**
     - The code appropriately utilizes React and Redux patterns for managing state and actions, suggesting that standard methods are already being used.
5. **Resource Defects:**
   - **a. Variable Initialization:**
     - Variables are initialized correctly, using React hooks (`useDispatch`, `useNavigate`) effectively.
   - **b. Memory Management:**
     - The React framework manages memory, and the use of hooks like `useDispatch` does not suggest any issues with memory leaks in this snippet.
6. **Check Defects:**

ChatGPT 4

   for both presence and format (email validation).
7. **Interface Defects:**
   - **a. Parameter:**
     - Parameters in function calls are correctly used, such as passing `values` to the `fetch_login` dispatch action.
8. **Logic Defects:**
   - **a. Compute:**
     - The logic to handle different error scenarios is present, but it can be enhanced by refining error messages and handling more specific exceptions.
   - **b. Performance:**
     - Performance concerns are minimal in the provided code. However, the error handling could be optimized to streamline execution and reduce complexity.

Summary of Recommended Changes:

- Add more detailed comments, especially explaining the error handling logic in the `on_finish` function.
- Consider refactoring the error handling to reduce complexity and improve readability, possibly by using a more modular approach or helper functions.
- Introduce more specific error handling for anticipated common issues, such as network errors or specific API failure modes, to provide users with more informative feedback.
- Ensure consistent style and format across all code, particularly in how errors and states are managed to maintain a clean and maintainable codebase.

Evaluation of Provided Code:

1. **Documentation Defects:**

- **a. Naming:**
  - The naming convention used in the code (`fetch_login`, `on_finish`, etc.) is descriptive and indicates the function's purpose clearly.

✅ Accept, we modified the naming conventions as per the GPT suggestions

- **b. Comments:**
  - The code lacks detailed comments that could help explain complex sections, particularly the error handling logic within the `on_finish` function.

✅ Accept

2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - Brackets are used correctly throughout the code.
   - **b. Indentation:**
     - The code maintains consistent indentation, aiding readability.
   - **c. Long Line:**
     - The error handling within the `on_finish` function could be refactored to avoid deeply nested conditional statements, which make lines long and complex.

❌ Reject, the deep nested conditional statements increase readability of the code

3. **Structure Defects:**
   - **a. Dead Code:**
     - No dead code is evident in the snippet provided.
   - **b. Duplication:**
     - There is no significant duplication in the code; the structure is streamlined and functional.
4. **New Functionality:**
   - **a. Use Standard Method:**
     - The code appropriately utilizes React and Redux patterns for managing state and actions, suggesting that standard methods are already being used.
5. **Resource Defects:**
   - **a. Variable Initialization:**
     - Variables are initialized correctly, using React hooks (`useDispatch`, `useNavigate`) effectively.
   - **b. Memory Management:**
     - The React framework manages memory, and the use of hooks like `useDispatch` does not suggest any issues with memory leaks in this snippet.
6. **Check Defects:**
   - **a. Check User Input:**
     - The form validation using Ant Design's `Form.Item` rules effectively checks user input for both presence and format (email validation).
7. **Interface Defects:**
   - **a. Parameter:**
     - Parameters in function calls are correctly used, such as passing `values` to the `fetch_login` dispatch action.
8. **Logic Defects:**

- **a. Compute:**
  - The logic to handle different error scenarios is present, but it can be enhanced by refining error messages and handling more specific exceptions.

✅ Accept, we will enhance the exceptions handling

- **b. Performance:**
  - Performance concerns are minimal in the provided code. However, the error handling could be optimized to streamline execution and reduce complexity.

✅ Accept, we will enhance the exceptions handling to streamline our code

Summary of Recommended Changes:

- Add more detailed comments, especially explaining the error handling logic in the `on_finish` function.
- Consider refactoring the error handling to reduce complexity and improve readability, possibly by using a more modular approach or helper functions.
- Introduce more specific error handling for anticipated common issues, such as network errors or specific API failure modes, to provide users with more informative feedback.
- Ensure consistent style and format across all code, particularly in how errors and states are managed to maintain a clean and maintainable codebase.

GPT Code Review - Submission

File: src/front_end/sila-frontend/src/pages/Author/StartNewSubmission/index.js

GPT Feedback

ChatGPT

Evaluation of Provided Code:

1. **Documentation Defects:**
   - **a. Naming:**
     - The names of variables and functions (`setAuthorsInfo`, `uploadedFilePaths`, `submit`) are mostly clear and convey their purposes well. However, the name `normFile` could be more descriptive.
   - **b. Comment:**
     - The code snippet has minimal comments, especially lacking in areas that perform complex operations like file handling and state management. More comments would improve maintainability and understanding.
2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - Brackets are used correctly and consistently in the React component and functions.
   - **b. Indentation:**
     - The indentation is consistent throughout, which helps maintain readability.
   - **c. Long Line:**
     - Some lines, especially within the `Form.Item` definitions, are quite long. Breaking these down would enhance readability.
3. **Structure Defects:**
   - **a. Dead Code:**
     - No dead code is evident in this snippet.
   - **b. Duplication:**
     - The usage of multiple `PDFUploader` components with similar functionality could be refactored into a single component that is rendered multiple times with different props.
4. **New Functionality:**
   - **a. Use Standard Method:**
     - The usage of Ant Design components and React patterns is standard and appropriate. The structured approach to form handling and state management adheres to React best practices.
5. **Resource Defects:**
   - **a. Variable Initialization:**
     - All variables are initialized appropriately. The use of React's `useState` for state management ensures variables are not left uninitialized.
   - **b. Memory Management:**
     - React handles memory management well, and there are no obvious leaks in this snippet. The use of local state and controlled components is correctly implemented.
6. **Check Defects:**
   - **a. Check User Input:**
     - The form validation through Ant Design's rules is a robust method for checking user input. This ensures that inputs like title, category, and abstract are not only present but adhere to specific requirements.
7. **Interface Defects:**
   - **a. Parameter:**
     - Parameters to functions like `onFileListChange` and `onFileUploaded` are correctly used. The code could benefit from documentation specifying the expected structure of these parameters.
8. **Logic Defects:**
   - **a. Compute:**
     - The logic implemented for file handling, data aggregation, and submission appears correct, but it could be enhanced by handling potential asynchronous errors more robustly.

- **b. Performance:**
  - The performance seems adequate for the given functionality. However, improving the efficiency of state updates, especially with large data sets or files, could be beneficial.

Summary of Recommended Changes:

- Enhance commenting throughout the code, especially in complex sections like file upload handling and JSON data preparation for submission.
- Break down longer lines within `Form.Item` and other lengthy expressions to improve readability.
- Consider refactoring the usage of multiple `PDFUploader` components into a single, more generic component that can be reused with different properties.
- Ensure robust error handling in asynchro .s operations to manage exceptions more effectively.
- Document function parameters and expected data structures to enhance clarity and maintainability, particularly for functions handling external interactions or complex data.

Evaluation of Provided Code:

1. **Documentation Defects:**
   - **a. Naming:**
     - The names of variables and functions (`setAuthorsInfo`, `uploadedFilePaths`, `submit`) are mostly clear and convey their purposes well. However, the name `normFile` could be more descriptive.

Accept, we modified the naming conventions as per the GPT suggestions.

- **b. Comment:**
  - The code snippet has minimal comments, especially lacking in areas that perform complex operations like file handling and state management. More comments would improve maintainability and understanding.

Accept.

2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - Brackets are used correctly and consistently in the React component and functions.
   - **b. Indentation:**
     - The indentation is consistent throughout, which helps maintain readability.
   - **c. Long Line:**
     - Some lines, especially within the `Form.Item` definitions, are quite long. Breaking these down would enhance readability.

Accept, we plan to break long lines into several short lines.

3. **Structure Defects:**
   - **a. Dead Code:**
     - No dead code is evident in this snippet.
   - **b. Duplication:**
     - The usage of multiple `PDFUploader` components with similar functionality could be refactored into a single component that is rendered multiple times with different props.

❌ Reject. We may improve it when all necessary functions implement.

4. **New Functionality:**
    - **a. Use Standard Method:**
        - The usage of Ant Design components and React patterns is standard and appropriate. The structured approach to form handling and state management adheres to React best practices.
5. **Resource Defects:**
    - **a. Variable Initialization:**
        - All variables are initialized appropriately. The use of React's `useState` for state management ensures variables are not left uninitialized.
    - **b. Memory Management:**
        - React handles memory management well, and there are no obvious leaks in this snippet. The use of local state and controlled components is correctly implemented.
6. **Check Defects:**
    - **a. Check User Input:**
        - The form validation through Ant Design's rules is a robust method for checking user input. This ensures that inputs like title, category, and abstract are not only present but adhere to specific requirements.
7. **Interface Defects:**
    - **a. Parameter:**
        - Parameters to functions like `onFileListChange` and `onFileUploaded` are correctly used. The code could benefit from documentation specifying the expected structure of these parameters.
8. **Logic Defects:**
    - **a. Compute:**
        - The logic implemented for file handling, data aggregation, and submission appears correct, but it could be enhanced by handling potential asynchronous errors more robustly.

✅ Accept. We will improve it in next sprint.

- **b. Performance:**
    - The performance seems adequate for the given functionality. However, improving the efficiency of state updates, especially with large data sets or files, could be beneficial.

❌ Reject. We may improve it when all necessary functions implement.

Summary of Recommended Changes:

- Enhance commenting throughout the code, especially in complex sections like file upload handling and JSON data preparation for submission.
- Break down longer lines within `Form.Item` and other lengthy expressions to improve readability.
- Consider refactoring the usage of multiple `PDFUploader` components into a single, more generic component that can be reused with different properties.
- Ensure robust error handling in asynchronous operations to manage exceptions more effectively.
- Document function parameters and expected data structures to enhance clarity and maintainability, particularly for functions handling external interactions or complex data.

GPT Code Review Reviewer Editor FormModal

File: src/front_end/sila-frontend/src/pages/Editor/FormModal/index.js

GPT Feedback

ChatGPT 4

ChatGPT

Evaluation of Provided Code:

1. **Documentation Defects:**
   - **a. Naming:**
     - Names like `handleOk`, `handleReviewerChange`, and `removeReviewer` are clear and descriptive of their functions. `selectedReviewerIds` aptly describes the state it manages. Overall, the naming convention is solid.
   - **b. Comment:**
     - The code lacks comments, which could enhance understanding, especially explaining the logic behind reviewer selection and the roles of various handlers.
2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - Brackets are used correctly throughout the component.
   - **b. Indentation:**

ChatGPT 4

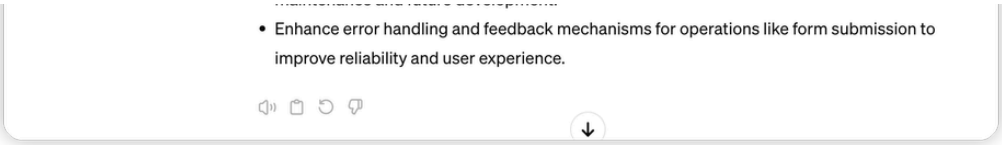
     - Some lines, particularly those involving JSX properties, are somewhat long but remain readable. Careful formatting helps maintain clarity.
3. **Structure Defects:**
   - **a. Dead Code:**
     - There is no evident dead code in the snippet.
   - **b. Duplication:**
     - The functionality for selecting reviewers and displaying them could be more streamlined or abstracted to reduce the complexity within the component.
4. **New Functionality:**
   - **a. Use Standard Method:**
     - The component uses React and Ant Design components in a standard way. The use of `useState` for managing state and `Form` for handling submissions aligns with common practices.
5. **Resource Defects:**
   - **a. Variable Initialization:**
     - All variables and states are initialized appropriately, ensuring no uninitialized states that could lead to bugs.

ChatGPT 4

     - React's handling of state and component lifecycle should adequately manage memory without leaks, given proper use of hooks and component unmounting.
6. **Check Defects:**
   - **a. Check User Input:**
     - The form validates the content field, requiring input before submission. However, further validations for email formats or reviewer selections could enhance robustness.
7. **Interface Defects:**
   - **a. Parameter:**
     - Parameters for functions like `onOk` and `onCancel` are used correctly, but it might benefit from documentation detailing their expected signatures and behaviors.
8. **Logic Defects:**
   - **a. Compute:**
     - Logic for handling state updates and submissions is correct, though the component might handle exceptions or errors in input more explicitly.
   - **b. Performance:**
     - The performance is likely adequate for most use cases, though the rendering efficiency could be impacted by frequent state updates in large reviewer lists.

Summary of Recommended Changes:

- Add comments to the code to explain the purpose of functions and the logic, especially where complex interactions like reviewer management occur.
- Consider abstracting the reviewer selection and list display into a separate component or using more sophisticated state management techniques to simplify the `FormModal`.
- Implement more robust validation for reviewer emails or selection patterns to ensure data integrity and user experience.
- Document the expected function parameters and behaviors for `onOk` and `onCancel` to aid in maintenance and future development.



Evaluation of Provided Code:

1. **Documentation Defects:**
   - **a. Naming:**
     - Names like `handleOk`, `handleReviewerChange`, and `removeReviewer` are clear and descriptive of their functions. `selectedReviewerIds` aptly describes the state it manages. Overall, the naming convention is solid.
   - **b. Comment:**
     - The code lacks comments, which could enhance understanding, especially explaining the logic behind reviewer selection and the roles of various handlers.

✅ Accept, we will add comments to our code.

2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - Brackets are used correctly throughout the component.
   - **b. Indentation:**
     - Indentation is consistent and aids in the readability of the code.
   - **c. Long Line:**
     - Some lines, particularly those involving JSX properties, are somewhat long but remain readable. Careful formatting helps maintain clarity.
3. **Structure Defects:**
   - **a. Dead Code:**
     - There is no evident dead code in the snippet.
   - **b. Duplication:**
     - The functionality for selecting reviewers and displaying them could be more streamlined or abstracted to reduce the complexity within the component.

✅ Accept, we will adjust the code, and make it clearer.

4. **New Functionality:**
   - **a. Use Standard Method:**
     - The component uses React and Ant Design components in a standard way. The use of `useState` for managing state and `Form` for handling submissions aligns with common practices.
5. **Resource Defects:**
   - **a. Variable Initialization:**
     - All variables and states are initialized appropriately, ensuring no uninitialized states that could lead to bugs.
   - **b. Memory Management:**
     - React's handling of state and component lifecycle should adequately manage memory without leaks, given proper use of hooks and component unmounting.
6. **Check Defects:**
   - **a. Check User Input:**
     - The form validates the content field, requiring input before submission. However, further validations for email formats or reviewer selections could enhance robustness.

✅ Accept, we will add the email verification and enhance reviewer selections.

7. **Interface Defects:**
    - **a. Parameter:**
        - Parameters for functions like `onOk` and `onCancel` are used correctly, but it might benefit from documentation detailing their expected signatures and behaviors.
8. **Logic Defects:**
    - **a. Compute:**
        - The logic for handling state updates and submissions is correct, though the component might handle exceptions or errors in input more explicitly.
    - **b. Performance:**
        - The performance is likely adequate for most use cases, though the rendering efficiency could be impacted by frequent state updates in large reviewer lists.

Summary of Recommended Changes:

- Add comments to the code to explain the purpose of functions and the logic, especially where complex interactions like reviewer management occur.
- Consider abstracting the reviewer selection and list display into a separate component or using more sophisticated state management techniques to simplify the `FormModal`.
- Implement more robust validation for reviewer emails or selection patterns to ensure data integrity and user experience.
- Document the expected function parameters and behaviours for `onOk` and `onCancel` to aid in maintenance and future development.
- Enhance error handling and feedback mechanisms for operations like form submission to improve reliability and user experience.

GPT Code Review ManuscriptsController

File: src/back_end/SiLA-Backend/SiLA-Backend/Controllers/ManuscriptsController.cs

GPT feedback

ChatGPT 4

ChatGPT

Evaluation of Provided Code:

1. **Documentation Defects:**
   - **a. Naming:**
     - The names used in the code (`UploadFile`, `Submit`, `GetAuthorDashboard`) are descriptive and clearly indicate their functionalities. Variables like `_submissionService` and `_amazonS3` also appropriately describe their purposes.
   - **b. Comments:**
     - There are no comments in the code snippet. Adding comments would help improve the understanding of the architecture and logic, particularly around the handling of AWS S3 operations and the role-specific authorizations.
2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - Brackets are used correctly and consistently throughout the code.
   - **b. Indentation:**
     - The code uses consistent indentation, which helps in maintaining readability.

ChatGPT 4

Breaking these into multiple lines could improve readability.

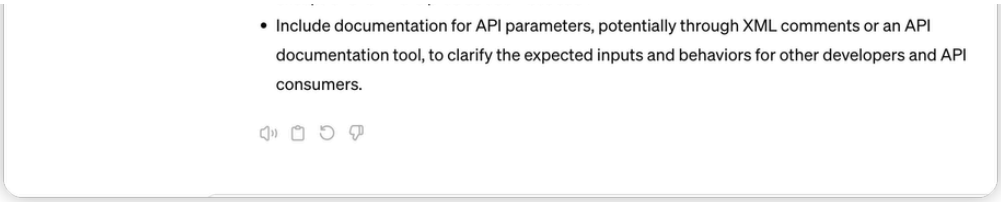
3. **Structure Defects:**
   - **a. Dead Code:**
     - There appears to be no dead code in the provided snippet.
   - **b. Duplication:**
     - The code does not show significant duplication. Each method has a clear, distinct responsibility.
4. **New Functionality:**
   - **a. Use Standard Method:**
     - The use of standard libraries and frameworks like ASP.NET Core, Amazon S3 SDK, and dependency injection indicates a standardized approach. The code adheres to modern practices in software development.
5. **Resource Defects:**
   - **a. Variable Initialization:**
     - All variables are appropriately initialized. Constructor dependency injection is used effectively for service and configuration management.
   - **b. Memory Management:**

ChatGPT 4

6. **Check Defects:**
   - **a. Check User Input:**
     - The validation for null or empty file in `UploadFile` and the check for a successful response in `Submit` demonstrate good practices in user input validation.
7. **Interface Defects:**
   - **a. Parameter:**
     - The parameters in API methods appear correct. However, documenting the expected types and constraints for these parameters would enhance interface clarity.
8. **Logic Defects:**
   - **a. Compute:**
     - The logic seems appropriate for the intended functionalities. Each method handles its specific task based on the success or failure of operations.
   - **b. Performance:**
     - The code could potentially improve in performance by handling exceptions more granularly, especially in network or AWS S3 operations, to avoid unnecessary retries or operations.

Summary of Recommended Changes:

- Add detailed comments throughout the code to explain the function of each component and method, especially for complex operations like file handling and error management.
- Consider breaking long lines, particularly those constructing error messages and URLs, to enhance code readability and maintainability.
- Implement more detailed error handling in AWS S3 operations to differentiate between types of exceptions for more precise user feedback.



Evaluation of Provided Code:

1. **Documentation Defects:**
   - **a. Naming:**
     - The names used in the code (`UploadFile`, `Submit`, `GetAuthorDashboard`) are descriptive and clearly indicate their functionalities. Variables like `_submissionService` and `_amazonS3` also appropriately describe their purposes.
   - **b. Comments:**
     - There are no comments in the code snippet. Adding comments would help improve the understanding of the architecture and logic, particularly around the handling of AWS S3 operations and the role-specific authorizations.

✅ Accept. We have added some comments on critical code snippet.

2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - Brackets are used correctly and consistently throughout the code.
   - **b. Indentation:**
     - The code uses consistent indentation, which helps in maintaining readability.
   - **c. Long Line:**
     - There are a few long lines, especially in error messages and URL constructions. Breaking these into multiple lines could improve readability.

✅ Accept.

3. **Structure Defects:**
   - **a. Dead Code:**
     - There appears to be no dead code in the provided snippet.
   - **b. Duplication:**
     - The code does not show significant duplication. Each method has a clear, distinct responsibility.
4. **New Functionality:**
   - **a. Use Standard Method:**
     - The use of standard libraries and frameworks like ASP.NET Core | Open-source web framework for .NET Core, Amazon S3 SDK, and dependency injection indicates a standardized approach. The code adheres to modern practices in software development.
5. **Resource Defects:**
   - **a. Variable Initialization:**
     - All variables are appropriately initialized. Constructor dependency injection is used effectively for service and configuration management.
   - **b. Memory Management:**
     - The use of `using` statements with AWS S3's `TransferUtility` suggests proper management of unmanaged resources, ensuring there are no memory leaks.

✅ Accept.

6. **Check Defects:**

- **a. Check User Input:**
  - The validation for null or empty file in `UploadFile` and the check for a successful response in `Submit` demonstrate good practices in user input validation.

7. **Interface Defects:**
   - **a. Parameter:**
     - The parameters in API methods appear correct. However, documenting the expected types and constraints for these parameters would enhance interface clarity.

✅ Accept. Will do more documentation on API methods parameters.

8. **Logic Defects:**
   - **a. Compute:**
     - The logic seems appropriate for the intended functionalities. Each method handles its specific task based on the success or failure of operations.
   - **b. Performance:**
     - The code could potentially improve in performance by handling exceptions more granularly, especially in network or AWS S3 operations, to avoid unnecessary retries or operations.

❌ Reject. Consider our time constrain, we should put more effort on something more valuable.

Summary of Recommended Changes:

- Add detailed comments throughout the code to explain the function of each component and method, especially for complex operations like file handling and error management.
- Consider breaking long lines, particularly those constructing error messages and URLs, to enhance code readability and maintainability.
- Implement more detailed error handling in AWS S3 operations to differentiate between types of exceptions for more precise user feedback.
- Include documentation for API parameters, potentially through XML comments or an API documentation tool, to clarify the expected inputs and behaviors for other developers and API consumers.

GPT Code Review SubmissionService

File: src/back_end/SiLA-Backend/SiLA-Backend/Services/SubmissionService.cs

GPT Feedback

ChatGPT 4

ChatGPT

Evaluation of Provided Code:

1. **Documentation Defects:**
   - **a. Naming:**
     - The naming convention used for methods (`SubmitAsync`, `GetAuthorDashBoardAsync`), classes, and variables is generally clear and descriptive, indicating their functions effectively.
   - **b. Comment:**
     - The provided code snippet lacks comments. Adding explanatory comments, especially for complex operations like transactions and error handling, would improve code maintainability and understanding.
2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - Brackets are used correctly and consistently in the code snippet.
   - **b. Indentation:**
     - The indentation is consistent and appropriate, aiding in the readability of the code.
   - **c. Long Line:**

ChatGPT 4

these into multiple lines could improve readability.

3. **Structure Defects:**
   - **a. Dead Code:**
     - No dead code is evident in the provided snippet. Each line and variable serve a purpose.
   - **b. Duplication:**
     - The method structure seems optimized without unnecessary duplication. However, each database operation is followed by `SaveChangesAsync()`, which might be more efficiently handled with a single call after all changes if they are meant to be a part of the same logical transaction.
4. **New Functionality:**
   - **a. Use Standard Method:**
     - The use of standard methods and patterns like EF Core for database operations, ASP.NET Identity for user management, and transactions for integrity is evident, which is good practice.
5. **Resource Defects:**
   - **a. Variable Initialization:**
     - All variables and services are properly initialized through dependency injection, reflecting good practices in resource management.

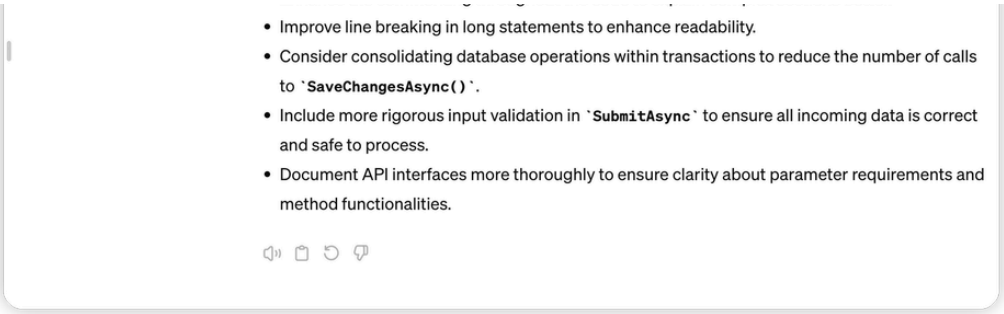
ChatGPT 4

preventing memory leaks.

6. **Check Defects:**
   - **a. Check User Input:**
     - User input seems to be validated at some level, as seen with the user check in `GetAuthorDashBoardAsync`. However, additional validations, especially on the `model` data in `SubmitAsync`, might be necessary to ensure data integrity before processing.
7. **Interface Defects:**
   - **a. Parameter:**
     - The parameters used in method calls appear correct, but the absence of interface documentation may obscure the expected types and values, which could be improved with XML comments or similar documentation techniques.
8. **Logic Defects:**
   - **a. Compute:**
     - The logic for submitting data and fetching dashboard data appears sound, but detailed error handling and transaction management could be more robust.
   - **b. Performance:**
     - The performance could potentially be impacted by multiple `SaveChangesAsync()` calls within a transaction. Consolidating these into a single call might improve performance.

Summary of Recommended Changes:

- Enhance the commenting throughout the code to explain complex sections better.



Evaluation of Provided Code:

1. **Documentation Defects:**

- **a. Naming:**
  - The naming convention used for methods (`SubmitAsync`, `GetAuthorDashBoardAsync`), classes, and variables is generally clear and descriptive, indicating their functions effectively.
- **b. Comment:**
  - The provided code snippet lacks comments. Adding explanatory comments, especially for complex operations like transactions and error handling, would improve code maintainability and understanding.

2. **Visual Representation Defects:**
   - **a. Bracket Usage:**
     - Brackets are used correctly and consistently in the code snippet.
   - **b. Indentation:**
     - The indentation is consistent and appropriate, aiding in the readability of the code.
   - **c. Long Line:**
     - Some lines, particularly error messages and LINQ queries, are quite long. Breaking these into multiple lines could improve readability.

✅ Accept. Broke them into multiple lines.

3. **Structure Defects:**
   - **a. Dead Code:**
     - No dead code is evident in the provided snippet. Each line and variable serve a purpose.
   - **b. Duplication:**
     - The method structure seems optimized without unnecessary duplication. However, each database operation is followed by `SaveChangesAsync()`, which might be more efficiently handled with a single call after all changes if they are meant to be a part of the same logical transaction.

✅ Accept. We have removed unnecessary database operations.

4. **New Functionality:**
   - **a. Use Standard Method:**
     - The use of standard methods and patterns like EF Core for database operations, ASP.NET Core | Open-source web framework for .NET Identity for user management, and transactions for integrity is evident, which is good practice.
5. **Resource Defects:**
   - **a. Variable Initialization:**
     - All variables and services are properly initialized through dependency injection, reflecting good practices in resource management.
   - **b. Memory Management:**
     - The use of `using` for transactions ensures that resources are properly disposed of, preventing memory leaks.

6. **Check Defects:**
    - **a. Check User Input:**
        - User input seems to be validated at some level, as seen with the user check in `GetAuthorDashBoardAsync`. However, additional validations, especially on the `model` data in `SubmitAsync`, might be necessary to ensure data integrity before processing.

Accept. Will add data integrity check before processing.

7. **Interface Defects:**
    - **a. Parameter:**
        - The parameters used in method calls appear correct, but the absence of interface documentation may obscure the expected types and values, which could be improved with XML comments or similar documentation techniques.

Accept. Will do more XML comments documentation on API endpoints.

8. **Logic Defects:**
    - **a. Compute:**
        - The logic for submitting data and fetching dashboard data appears sound, but detailed error handling and transaction management could be more robust.
    - **b. Performance:**
        - The performance could potentially be impacted by multiple `SaveChangesAsync()` calls within a transaction. Consolidating these into a single call might improve performance.

Accept.

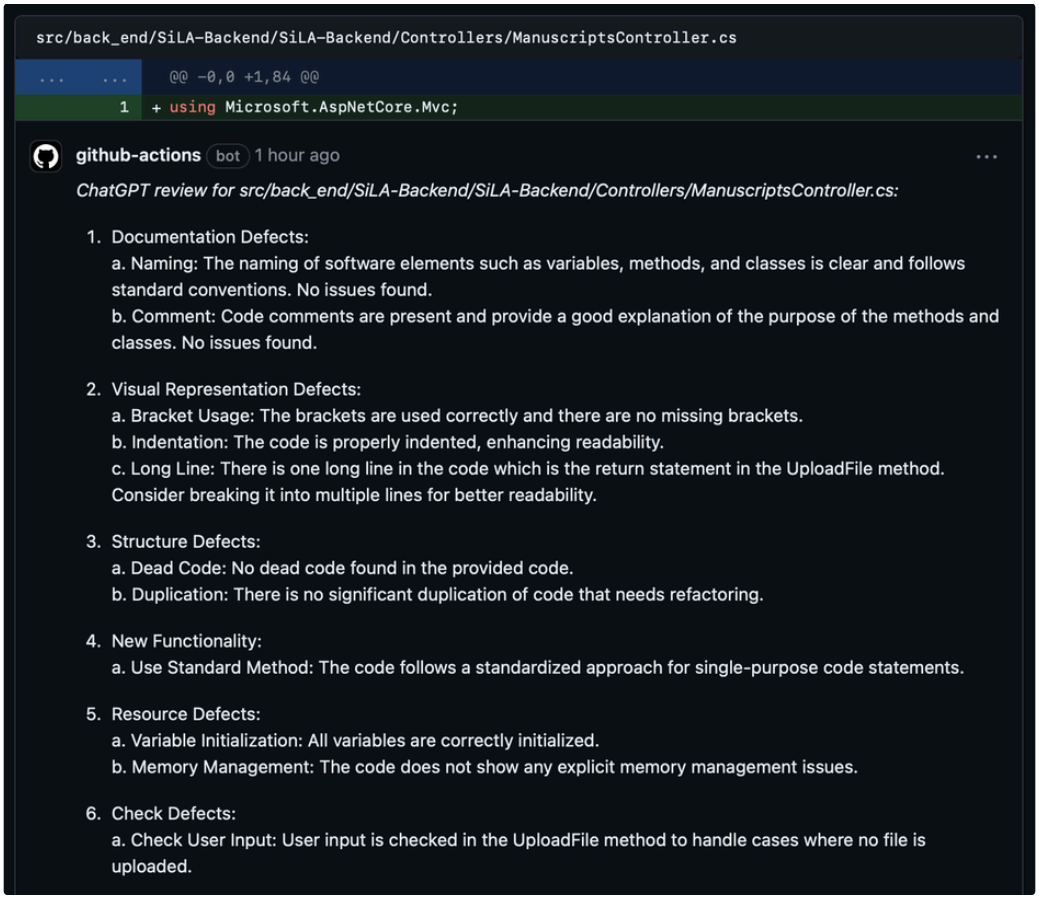
Summary of Recommended Changes:

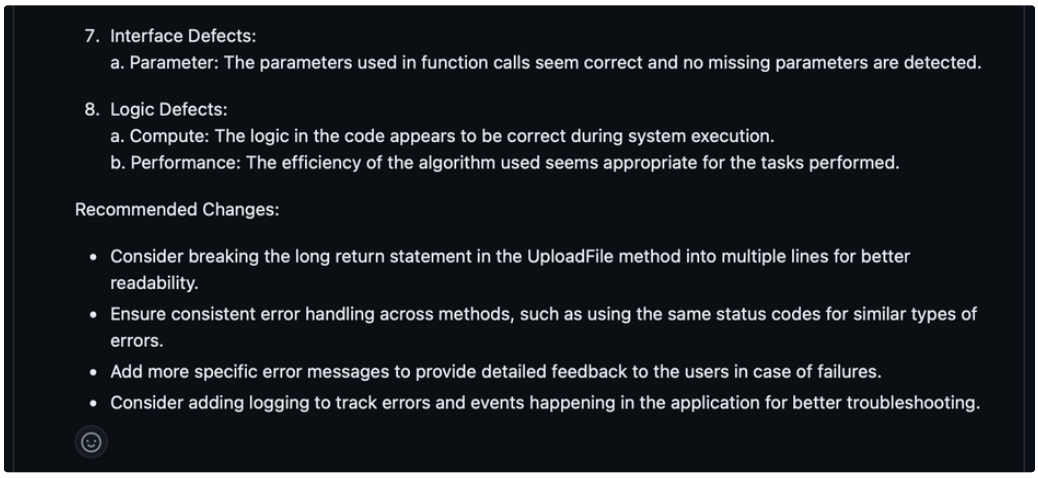
- Enhance the commenting throughout the code to explain complex sections better.
- Improve line breaking in long statements to enhance readability.
- Consider consolidating database operations within transactions to reduce the number of calls to `SaveChangesAsync()`.
- Include more rigorous input validation in `SubmitAsync` to ensure all incoming data is correct and safe to process.
- Document API interfaces more thoroughly to ensure clarity about parameter requirements and method functionalities.

GPT Code Review Pull Request Version

In the above section, we directly use ChatCPT due to the UsageLimit.

This Section is an example of how we can use this GPT Code Reviewer During Pull Request.





Sprint

In this section, it explains the progress of the SiLA project.

1. Sprint 1 (Design)
2. Sprint 2 (Development)
3. Sprint 3 (Development)

Sprint 1

Sprint Goal:

The sprint goal for Sprint 1 is to establish a solid foundation for the project by focusing on comprehensive documentation, analysis of requirements, setting up the development environment, and initial planning for subsequent sprints.

Sprint Duration:

4 Mar - 22 Mar

What has been done in the Sprint?

1. Held the first group meeting with teammates to get to know each other and familiarize with the project.
2. Setup project documentation in Confluence, including project overview, background, goals, DO-BE-FEEL list, and GOAL MODEL.
3. Created personas based on research and analysis, documented on a Miro Board.
4. Set up and structured JIRA for task management.
5. Updated the README file with project information.
6. Confirmed the analysis of requirements and ensured consistency with the project scope.
7. Established user stories and organized backlog items in JIRA.
8. Drafted a preliminary plan for Sprints 2 and 3, discussing requirements, technologies, and infrastructure.
9. Scheduled meetings and ensured all meetings are recorded in Confluence.
10. Created a GitHub repository with structured directories, documentation, and a baseline tag for Sprint 1.

Sprint 1 Planning

Objective:

Establish a solid foundation for the project by focusing on comprehensive documentation, analysis of requirements, setting up the development environment, and initial planning for subsequent sprints.

- **Get to know team members**: Hold the first group meeting with teammates, get to know with each other, and get familiar with the project (read the project document and sprint guidelines) together. **Assigned to**: @Yizhi Liao

- **Setup Project Documentation in Confluence**: Background description, client goals, motivation
  - Have a page that includes the project overview, background, and goals. **Assigned to**: @Donghong Zhuang
  - Create a DO-BE-FEEL list and GOAL MODEL. **Assigned to**: @jiajiny
  - Find Personas that are based on the research done by the development team and the discussion with industry partners. **Assigned to**: @Yuxuan Zeng
  - Analyze those into a Miro Board. **Assigned to**: @Jiayu Yang

- **Development Environment**
  - Set up and structure JIRA for task management. **Assigned to**: @Chuyuan Xu
  - Schedule Jira Training Session. **Assigned to** @Yizhi Liao
  - Update the README file with project information. **Assigned to**: @Donghong Zhuang

- **Analysis of requirements (User Stories or Use Cases)**
  - Confirm that the analysis of requirements has been performed on most of the existing requirements and mark them as reviewed. **Assigned to**: **Everyone**
  - Ensure that the new set of requirements is consistent with the project scope, fully covers the new capabilities required by the client, and is well-documented/structured/organized on Confluence. **Assigned to**: **Everyone**
  - Clarify that requirements can be documented in the form of user stories or use cases, supplementary specifications of design/implementation/deployment requirements. Also, make it explicit what is out of scope to define the scope boundary more clearly. **Assigned to**: **Everyone**
  - Design prototypes in the Figma. **Assigned to**: **Everyone**
  - Set up user stories. In the Confluence Space and Jira. **Assigned to**: **Everyone**

- **Planning for Next Sprints**
  - Draft a preliminary plan for Sprints 2 and 3, discussing requirements, technologies, and infrastructure. **Assigned to**: @jiajiny
  - Estimate and prioritize requirements, organizing backlog items in JIRA. **Assigned to**: @Yuxuan Zeng

- **Meetings**
  - Schedule meetings(Mentor, Client, and Weekly stand up). **Assigned to**: @Jiayu Yang
  - Ensure all meetings are recorded in Confluence. **Assigned to**: @Yizhi Liao

- **Set Up GitHub Space**:
  - Create a GitHub repository, which contains two folders(*docs* and *src*) and a README file. **Assigned to**: @Yizhi Liao
  - Update the repository regularly, including the README file. **Assigned to**: @Yizhi Liao

- Generate a release tag before the end of sprint 1. **Assigned to**: @Yizhi Liao

Deliverables

- Completed and reviewed project documentation (overview, background, goals, DO-BE-FEEL list, GOAL MODEL, personas)
- Documented analysis of requirements with user stories generated and refined
- Fully organized Confluence space and structured Trello/JIRA for project management
- A clear plan for Sprints 2 and 3 with estimated and prioritized requirements
- Updated GitHub repository with a structured directory, necessary documentation, and a baseline tag for Sprint 1

Sprint 1 Retrospective

Participants:

- Jiayu Yang
- Donghong Zhuang
- Jiajin Yang
- Yizhi Liao
- Yuxuan Zeng
- Chuyuan Xu

Related Meetings:

StandUp Meeting 10/03/2024

StandUp Meeting 13/03/2024

StandUp Meeting 20/03/2024

Mentor Meeting 06/03/2024

Mentor Meeting 13/03/2024

Mentor Meeting 20/03/2024

Client Meeting 07/03/2024

What worked well	What didn't go well	What should we try doing next
The team effectively collaborated on tasks	Some tasks took longer than expected	Improve task estimation
Clear plan for subsequent sprints drafted	Some difficulty in estimating and prioritizing requirements	Improving estimation and prioritization processes
Regular scheduling of meetings and recording of minutes	Limited familiarity with some tools and technologies	Providing training sessions for tool usage
All planned meetings occurred as scheduled	Miscommunication regarding task responsibilities leading to slight delays	Enhance communication channels
The team achieved 100% attendance at all scheduled meetings		

Sprint 1 Review

Project Documentation and Confluence Setup

- **Everyone** took participate in establishing project documentation on Confluence, including an overview of the project, background, and goals, ensuring that all team members could access and understand the core information of the project.
- A DO-BE-FEEL list and goal model were developed to clarify the vision and objectives of the project.
- Personas that match the project requirements were identified based on research conducted by the development team and discussions with industry partners. These were organized into a Miro Board, providing a human-centric perspective for subsequent design and development efforts.

Project Management and Jira Usage

- @Donghong Zhuang also took charge of setting up and structuring Jira for task management, and scheduled a Jira training session to ensure that team members could effectively utilize this tool.
- @Yizhi Liao updated the README file to provide project information in the project's GitHub repository, ensuring that all visitors could quickly understand the basics of the project.

Requirement Analysis and User Stories

- All team members jointly confirmed the analysis of existing requirements and marked them as reviewed, ensuring the completeness and accuracy of the requirement analysis.
- We also ensured that a new set of requirements was consistent with the project scope and was detailed, documented, and organized on Confluence, explicitly defining the project scope boundary and the requirements for design/implementation/deployment through user stories and use cases.

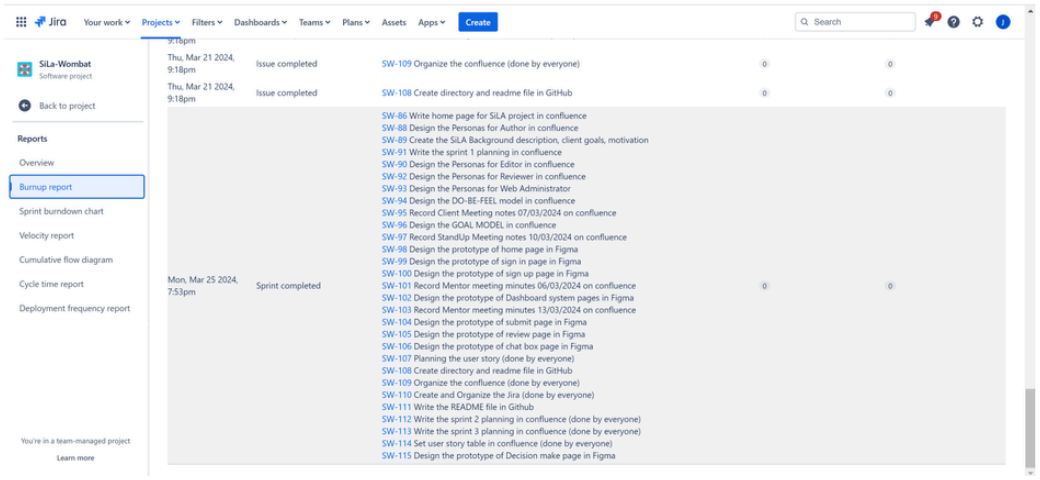
Planning and Meetings

- A preliminary plan for Sprints 2 and 3 was drafted, discussing requirements, technologies, and infrastructure.
- Meetings with mentors, clients, and the team were scheduled regularly, ensuring all meetings were recorded on Confluence.

GitHub Repository Setup

- @Yizhi Liao was responsible for creating a GitHub repository containing docs and src folders along with a README file and regularly updating the repository, including generating a release tag before the end of Sprint 1.

Sprint 1 Complete



Sprint 2

Sprint Goal:

The sprint goal for Sprint 2 is to establish a robust foundation for the submission and review platform by implementing essential components such as user authentication, registration, dashboard creation, and manuscript management functionalities.

Sprint Duration:

23 Mar - 02 May

What will be done in the Sprint?

1. Implement database schema for the account system (SW-123).
2. Design and implement the database schema for manuscripts (SW-124).
3. Develop a login page with authentication for different user types (SW-2).
4. Create a registration page for authors (SW-5).
5. Develop a dashboard page for authors to start new submissions (SW-30).
6. Design and implement a home page accessible to all users (SW-16).
7. Implement a review page for reviewers (SW-14).
8. Develop a decision-making page for editors (SW-135).

Sprint 2 Planning

User story and Sprint backlog

We assign the task to the team members based on their interests and willingness. Also, we considered member's coding skills.

User story ID	User story	Story Point	Size Estimation	MoSCoW Priority	Sub-task	Acceptance Criteria	Assign to
SW-16	Home page for All Users: As a user, I want a home page that intuitively guides me to the platform's main features, such as signing up, logging in, searching for documents, and discovering popular content, so that it can ensuring a positive first impression and ease of use.	10	Medium	Will Not Have	SW-60: Design the home page layout incorporating essential elements	The home page layout is clear and easy to understand. And the functionality is complete.	@Chuyuan Xu
					SW-61: Develop the functionality for the search bar	All the buttons on search bar could navigate us to correct page. Also, the design is clear and easy to understand.	@Chuyuan Xu
					SW-62: Create a dynamic section that displays popular content	The popular content could be displayed correctly to the user. And make sure the user can access those contents.	@Chuyuan Xu
					SW-64: Integrate sign-up and login buttons with clear calls to action, directing users to the respective forms or pages.	The button is easy to find on the home page. Onece click the button, the user should enter the login or sign-up page correctly.	@Chuyuan Xu
SW-123	Database for Account system: As an administrator, I want to create a robust database schema for the account system, so that we can securely store and efficiently manage user information, roles, permissions, and sensitive data, supporting the dynamic needs of our platform as it grows.	14	Large	Must have	SW-125: Design a comprehensive database schema catering to users, roles, permissions, sensitive information.	Database schema includes separate tables for users, roles, permissions, and sensitive information with all relationships clearly defined	@Yuxuan Zeng
					SW-126: Implement relationships between tables to efficiently manage user information, role assignments, permissions, sensitive information.	all database tables are correctly linked via foreign keys, and queries joining these tables execute efficiently without anomalies	@Yuxuan Zeng
					SW-127: Develop secure storage procedures for sensitive information like passwords using best practices for hashing and salting.	Passwords and other sensitive information such as manuscript files are securely stored using advanced hashing and salting techniques	@Yuxuan Zeng
					SW-128: Create scripts for routine database operations (e.g., adding or updating users, changing	Adding, updating users, and changing roles execute successfully without errors and these changes should	@Yuxuan Zeng

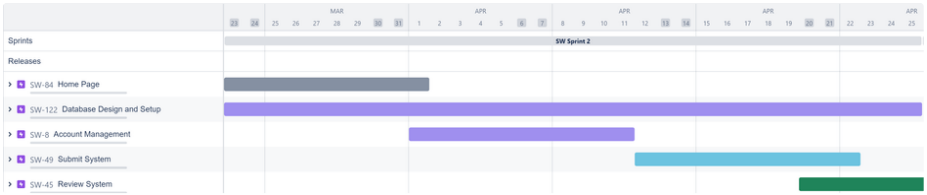
					roles) to streamline account management.	be reflected in the database	
SW-124	Database for Manuscripts: As a platform administrator, I need to create a database schema that efficiently organises, stores, and retrieves manuscripts, along with their submission details, review statuses, and associated metadata, so that authors can submit their work, reviewers can assess submissions, and editors can make informed decisions.	20	Large	Must have	SW-129: Design tables for storing manuscripts, including fields for manuscript ID, title, abstract, submission date, author ID, status (e.g., submitted, under review, accepted, rejected), and any other relevant information.	Create a `Manuscripts` table with columns for manuscript ID, title, abstract, submission date, author ID, and status. Ensure columns are non-nullable where necessary and correctly typed. Link `author_id` to an `Authors` table via foreign key.	`@Yuxuan Zeng`
					SW-130: Create related tables to handle metadata such as keywords, topics, author affiliations, and submission files (linking to stored files if not kept directly in the database).	Establish tables for keywords, topics, author affiliations, and submission files, each with a primary key and a `manuscript_id` foreign key for linking back to the `Manuscripts` table.	`@Yuxuan Zeng`
					SW-131: Implement tables for reviews and reviewer assignments, capturing review ID, manuscript ID, reviewer ID, review date, comments, recommendations, and any confidentiality flags.	Implement a `Reviews` table with fields for review ID, manuscript ID, reviewer ID, review date, comments, recommendations, and a confidentiality flag. Set up foreign keys for `manuscript_id` and `reviewer_id` to ensure proper referencing.	`@Yuxuan Zeng`
					SW-132: Define relationships between authors, manuscripts, and reviews to enable efficient data retrieval and updates, ensuring integrity constraints and referential actions are appropriately applied.	Ensure all foreign keys in the database correctly reference the primary keys of their related tables, with appropriate cascade actions for update and delete operations.	`@jiajiny`
					SW-133: Ensure the design supports queries for manuscript tracking, reviewer assignments, and editorial decision-making, with considerations for performance optimization and future scalability.	Confirm the database supports efficient and accurate queries for manuscript tracking, reviewer assignments, and editorial decision-making, and can handle future scalability.	`@jiajiny`

SW-2	Login page for different user types: As a user (author, reviewer, editor), I want to log in to the system as my specific user group, so that I can access different kinds of functions of the website tailored to my role.	6	Medium	Must have	SW-34: Implement elements that allow users to select their role (author, reviewer, editor) before logging in.	The user must be able to select a role when log into the system.	@Yizhi Liao
					SW-42: Implement a simple form for email/username and password.	The user should be able to input a username and password when log into the system.	@Yizhi Liao
					SW-43: Design a login page that is welcoming and intuitive.	The login page must be visually appealing, user-friendly and provide clear guidance for users to easily access their account.	@Yizhi Liao
SW-5	Register Page for Author: As a potential author, I want a straightforward registration process, so that I can create an account and start submitting manuscripts.	6	Medium	Must have	SW-22: An input field allows the user to enter their email.	The field should allow user to enter the email and check the email is in correct format.	@Donghong Zhuang
					SW-23: A detailed information page with multiple input fields to collect user information.	Multiple input fields on the page correctly ask user's information.	@Donghong Zhuang
					SW-24: Add a Sign-Up button.	Once the sign-up button, the sign-up information will be sent back to the back-end, and an account for the author will be successfully created.	@Donghong Zhuang
SW-30	Dashboard page for Author to Start New Submission: As an author, I want to submit the manuscripts to the journal in an efficient manner, so that I can publish my article via the system.	10	Medium	Must have	SW-56: The input fields include manuscript title and type.	Both manuscript title and type are present on the user interface, are properly labelled, non-empty requirements are applied.	@Jiayu Yang
					SW-70: An "Add" button to add the author information of a specific manuscript	It should add another row where author information can be entered. The maximum number of authors is 8.	@Jiayu Yang
					SW-68: The user can dray the file and drop it into the "Drop files area", which will then automatically prepare the file for the upload progress.	The system automatically initiates the preparation process for the upload progress and then returns a progress message.	@Jiayu Yang
					SW-71: This button will then submit all the information to the system.	The submit button, when clicked, all the information is stored in JSON format and submits all the entered information to the backend system.	@Jiayu Yang
					SW-85: The submission system should be scalable to accommodate	The submission system must demonstrate scalability by efficiently handling a simulated load of up to 10 times the current average	@Jiayu Yang

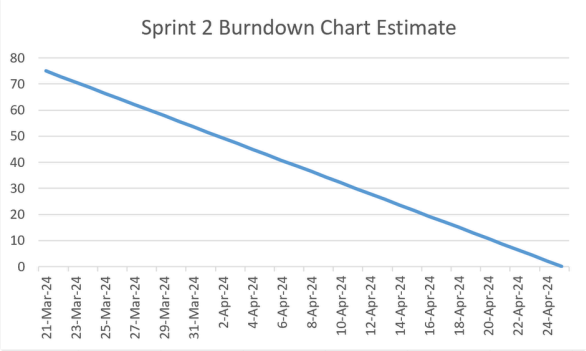
					increasing submissions.	number of daily submissions without significant performance degradation.	
SW-14	Review page for Reviewer: As a reviewer for SiLA, I want to be able to securely log in to the online submission and review platform so that I can access manuscripts assigned to me for review. I wish I can see all the relevant comments to this article and submit my own comment. I also want to contact with the editors conveniently when I need it.	8	Medium	Must have	SW-21: There is a download selection for the reader to download the article.	The reviewer should be able to download the article.	@jiajiny
					SW-29: There is a label that displays whether the article is double blind mode or not. If it is, then it shows the double-blind mode, otherwise it will show the author's name.	The paper will be displayed to reviewer in the correct format.	@jiajiny
					SW-33: The page has a field to show all the comments related to this article.	All the comments related to this article are correctly shown and well organized on the page.	@jiajiny
					SW-31: Reviewers can submit reviews to the article.	Once click the submit, the reviewer's comment will be saved to the backend, and the editor and author should able to see the comment.	@jiajiny
					SW-37: There is a editors list for the reviewers can communicate with. For each editor, there is a chat button for each editor to let the reviewer directly jump into the communication system with that editor.	This section should allow the reviewer to contact editors.	@jiajiny
SW-135	Decision making page for Editor: As an editor, I want to decide whether to accept an article or not, so that I can guarantee the quality of the project.	5	Medium	Must Have	SW-136: Reuse the comment history function, add a comment function, double-blind mode and download article function which are implemented in SW-14.	1. The system must incorporate the ability to reuse and display comment history 2. The system must include functionalities to add comments, enable a double-blind mode for user interactions, and allow users to download articles.	@Jiayu Yang
					SW-137: A reviewers list, which contains a chat button for each reviewer to let the editor communicate with them directly. The chat button will jump to the chat box function.	1. The system must provide a list of reviewers, each accompanied by a chat button that enables direct communication between the editor and the reviewer. 2. The chat button should redirect the	@Jiayu Yang

						user to a chat box function, facilitating immediate and intuitive interaction.	
					SW-138: There are three selections for the editor to decide for an article which are accept, reject or revise.	The system must provide the editor with three clear and functional options for article decisions: “Accept“, “Reject“, and “Revise“.	@Jiayu Yang

Sprint 2 Milestones



Sprint 2 Burndown Chart



Sprint 2 Retrospective

Participants:

- Jiayu Yang
- Donghong Zhuang
- Jiajin Yang
- Yizhi Liao
- Yuxuan Zeng
- Chuyuan Xu

Related Meetings:

StandUp Meeting 27/03/2024

StandUp Meeting 10/04/2024

StandUp Meeting 17/04/2024

StandUp Meeting 24/04/2024

StandUp Meeting 01/05/2024

Mentor Meeting 27/03/2024

Mentor Meeting 10/04/2024

Mentor Meeting 17/04/2024

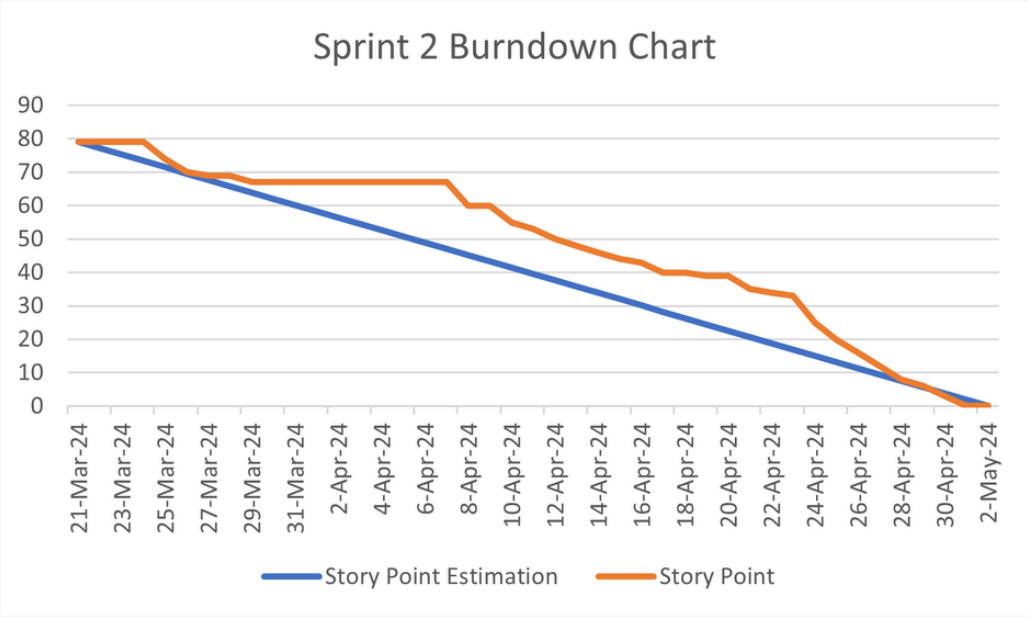
Mentor Meeting 24/04/2024

Mentor Meeting 01/05/2024

Client Meeting 28/03/2024

Sprint 2 Burndown Chart：

This Burndown Chart shows the team’s working process in Sprint 2. Due to the mid-break from Mar 30 - Apr 07, we have a week that we do not have any progress. Because we just started programming in sprint 2, each team member needed time to get familiar with the development process and the code work, which caused our progress in the first few weeks to be slower than planned. As we became more familiar with the project development, our work speed gradually increased, and in the final week, we caught up with the planned progress with great efficiency and successfully completed the sprint 2 objectives.



What worked well	What didn't go well	What should we try doing next
The team collaborated effectively, showing strong teamwork in completing tasks.	Testing phases for some features might not have been as thorough as necessary, leading to bugs or issues not being caught early.	Implement refined estimation strategies to better predict task durations.
Weekly meetings were held on schedule	Task prioritization was occasionally problematic, especially when unexpected issues arose.	Develop clearer criteria for task prioritization to handle disruptions more effectively.
The team consistently achieved the objectives set for each meeting, maintaining high efficiency.	Some parts of the codes lack comments.	Add more comments that improve the readability.
Adjustments to the project were made promptly based on weekly feedback from the mentor.	The structure of the code is not very clear.	Modify the structure of the code file to better find the code.
The project's progress kept pace with the timeline, avoiding any significant delays.		

Sprint 2 Review

Project Documentation and Confluence Setup

- All team members contributed to updating the project documentation on Confluence, including detailed information on the progress and goals achieved during Sprint 2. This ensured that all team members had access to the latest updates and could understand the development progress.
- We refined our DO-BE-FEEL list and goal model to align with the progress made during Sprint 2, ensuring that the project vision and objectives remained clear and focused.

Project Management and Jira Usage

- @Donghong Zhuang continued to manage Jira, structuring it to reflect the tasks and progress of Sprint 2. This included updating task statuses and ensuring that all team members could track the project's progress.
- @Yizhi Liao updated the README file in the project's GitHub repository to reflect the latest developments, providing up-to-date project information for all visitors.

Backend Development

- Set up the backend environment and tested the usability of the Restful APIs.
- Created the Login, Register, and Logout APIs to handle user authentication.
- Constructed essential database tables: User, Article, Submission, SubmissionReview, ArticleAuthor, ChatSession, and Address.
- Developed the Manuscript and User APIs to support the backend functionalities.

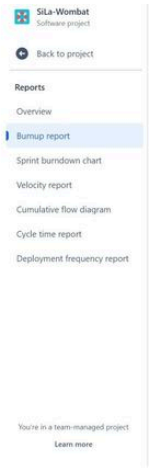
Frontend Development

- Developed the login and register pages using the React framework and connected them to the backend through the Internet.
- Implemented critical UI components, including the author dashboard, article submission page, article detail page, user profile page, and reviewer dashboard.
- Implemented the base role page and logout function.
- Explored various open-source UI frameworks and decided to use "Antd" for its robustness and flexibility.

Deployment and Documentation

- Deployed the backend code to the cloud and launched the backend system using Docker, ensuring scalability and reliability.
- Continued updating the Sprint 2 documentation in Confluence, providing detailed records of the tasks completed and the progress made.
- Merged the branches developed for Sprint 2 into the main branch, deleted the development branches for Sprint 2, and started preparing for Sprint 3’s branches.
- Recorded a Sprint 2 review video to document the progress and demonstrate the functionalities implemented.
- Updated the GitHub README file to reflect the latest developments and provide clear instructions for contributors.

Sprint 2 Complete



Date	Event	Issue	Completed	Scope
Mon, Mar 25 2024, 7:54pm	Sprint started	SW-5 Register Page for Author SW-14 Review page for Reviewer SW-30 Dashboard page for Author to Start New Submission SW-2 Login page for different user types SW-124 Database for Manuscripts SW-123 Database for Account system SW-16 Home page for All Users SW-135 Decision make page for Editor	0	75
Wed, Apr 24 2024, 7:43pm	Added to sprint	SW-145 Backend for Auth API endpoints	0	75
Wed, Apr 24 2024, 7:53pm	Estimate updated	SW-145 Backend for Auth API endpoints	0	75 → 89
Wed, May 01 2024, 6:10pm	Added to sprint	SW-150 Backend for Manuscripts submission API endpoints	0	89
Wed, May 01 2024, 6:16pm	Estimate updated	SW-150 Backend for Manuscripts submission API endpoints	0	89 → 103
Wed, May 01 2024, 8:56pm	Issue completed	SW-145 Backend for Auth API endpoints	0 → 14	103
Wed, May 01 2024, 8:56pm	Issue completed	SW-150 Backend for Manuscripts submission API endpoints	14 → 28	103
Wed, May 01 2024, 8:56pm	Issue completed	SW-123 Database for Account system	28 → 42	103
Wed, May 01 2024, 8:56pm	Issue completed	SW-124 Database for Manuscripts	42 → 62	103
Wed, May 01 2024, 8:56pm	Issue completed	SW-5 Register Page for Author	62 → 68	103
Wed, May 01 2024, 8:57pm	Issue completed	SW-2 Login page for different user types	68 → 74	103
Wed, May 01 2024, 8:57pm	Issue completed	SW-30 Dashboard page for Author to Start New Submission	74 → 80	103
Wed, May 01 2024, 8:58pm	Issue completed	SW-14 Review page for Reviewer	80 → 88	103
Wed, May 01 2024, 8:58pm	Issue re-opened	SW-14 Review page for Reviewer	88 → 80	103
Wed, May 01 2024, 8:59pm	Issue completed	SW-135 Decision make page for Editor	80 → 85	103
Wed, May 01 2024, 9:36pm	Issue completed	SW-16 Home page for All Users	85 → 95	103
Mon, May 06 2024, 5:10pm	Removed from sprint	SW-14 Review page for Reviewer	95	103 → 95
Mon, May 06 2024, 5:11pm	Added to sprint	SW-14 Review page for Reviewer	95	95 → 103
Mon, May 06 2024, 5:11pm	Issue completed	SW-14 Review page for Reviewer	95 → 103	103
Mon, May 06 2024, 5:11pm	Sprint completed	SW-5 Register Page for Author SW-14 Review page for Reviewer SW-30 Dashboard page for Author to Start New Submission SW-2 Login page for different user types SW-124 Database for Manuscripts SW-123 Database for Account system SW-16 Home page for All Users SW-135 Decision make page for Editor SW-145 Backend for Auth API endpoints SW-150 Backend for Manuscripts submission API endpoints	103	103

Sprint 3

Sprint Goal:

The goal for Sprint 3 is to develop advanced features such as the dashboard for authors, reviewers, and editors, implement a chat box functionality, and enable automatic email notifications.

Sprint Duration:

03 May - 24 May

What will be done in the Sprint?

1. Develop the dashboard interface for authors, reviewers, and editors.
2. Implement functionality to display manuscript statuses on the dashboard.
3. Design and integrate the chat box feature for communication between users.
4. Set up automated email notifications for manuscript submissions, reviews, and editorial decisions.
5. Conduct testing and debugging to ensure the proper functioning of the developed features.
6. Document the implemented functionalities and update project documentation as necessary.
7. Conduct user acceptance testing (UAT) to gather feedback and make necessary adjustments.
8. Prepare for deployment and release planning for the completed features.

Sprint 3 Planning

User story and Sprint Backlog

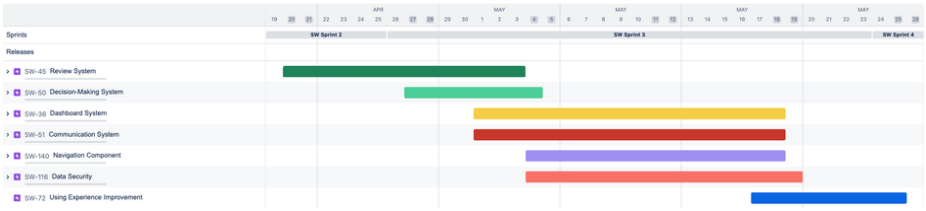
We assign the task to the team members based on their interests and willingness. Also, we considered member's coding skills.

User story ID	User story	Story Point	Size Estimation	MoSCoW Priority	Sub Task	Acceptance Criteria	Assign to
SW-26	Dashboard - Reviewer - Review and Score: As a Reviewer, I want to have a centralized dashboard that not only allows me to track the status of manuscripts through various stages of the review process (including those in the following stages: submitted, under review, revisions required, and accepted) but also acts as a to-do list showing my assigned tasks. This will enable me to have a comprehensive overview of my responsibilities and the progress of manuscripts, so that I can prioritize my work efficiently, and enhance communication with other team members.	12	Large	Must have	SW-57 Implement Reviewer Dashboard Layout	The reviewer dashboard layout should clearly list the articles of those assigned to the current reviewer.	@Yuxuan Zeng
					SW-44 Track Manuscript Status	When the status of an article changes, the reviewer dashboard must display the latest article status.	@Yuxuan Zeng
					SW-46 Enable Reviewer Access to Manuscript Reviewing Page	The reviewer should be able to leave comments to the article in the editing page.	@Yuxuan Zeng
SW-28	Dashboard - Reviewer - History: As a Reviewer I require a functionality to display all manuscripts that have undergone editing. This feature will enable team members to easily access and review the edited manuscripts, facilitating collaboration and ensuring that all edited content is readily available for further processing or review.	8	Medium	Must have	SW-58 Implement Review Dashboard Layout	The Review Dashboard layout should be implemented with a clear, user-friendly interface that displays all essential review metrics and allows for easy navigation and interaction.	@Yuxuan Zeng
					SW-48 Display Edited All The Edited Manuscripts	The system should display all edited manuscripts in a well-organized list, ensuring that each manuscript shows its status.	@Yuxuan Zeng
SW-17	Dashboard page for Author: As an author, I want a user-friendly page that displays the information, so that I can edit and change my personal information.	8	Medium	Should have	SW-52 A page shows the user's detailed information	The Profile page is connected to the back-end database and displays the user's information correctly.	@jiajiny
					SW-53 A button that click to modify user's information	Once click the button, the author should able to modify their information.	@jiajiny
					SW-54 A page to modify user's personal information	All the information sections except the Email will become input field which allows user	@jiajiny

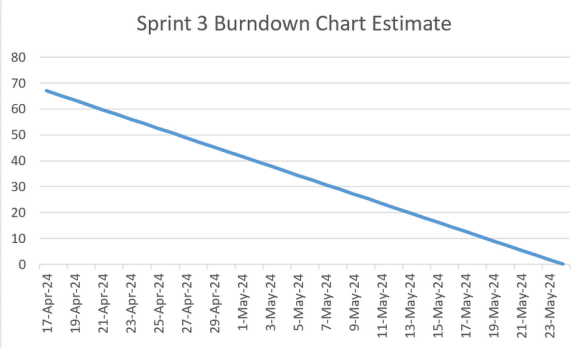
						to change their personal information.	
					SW-55 A button to save the change	Once the button is clicked, all the changed information is saved and sent to the back-end database.	@jiajiny
					SW-139 A page shows all the manuscripts submitted by the current author	This page shows a list of manuscripts that the author submitted which includes an article title, deadline, and editor's decision.	@jiajiny
SW-18	Dashboard page for Editor: As an editor, I want a dedicated section to record the number of those decisions (accept, revise, reject) so that my decisions are clearly communicated to the authors.	8	Medium	Must have	SW-38 A Dashboard Metrics for Editor's Decision-Making	Include relevant metrics such as submission status, reviewer feedback, acceptance rates, publication timelines and display dashboard for each status.	@Jiayu Yang
					SW-41 Display Manuscript Details in Editor's Dashboard	Ensure the displayed manuscript details are up-to-date and accurately reflect the current state of each submission. The display layout should be clear, organized, and easily navigable for editors.	@Jiayu Yang
SW-118	Automatic confirmation emails for successful submission: As an author, I want to receive the confirmation email from the SiLA, so that I can ensure I submit the article successfully.	5	Medium	Should have	SW-119 Create Email Content Template	The Email Content is well formatted and in a formal tone.	@Donghong Zhuang
					SW-120 Integrate Email Sending Function into Submission Process	The Email is successfully sent to user.	@Donghong Zhuang
SW-67	Chat Functionality for Reviewer-Editor Communication: As a reviewer or editor, I want to be able to engage in text-based communication within a dedicated pop-up chat window on the platform, so that I can discuss manuscripts, share feedback, and clarify queries directly and efficiently.	10	Medium	**Will Not Have**	SW-75 Design a user-friendly chat interface that can be accessed as a pop-up window on the platform	The chat interface should be designed as a user-friendly pop-up window, easily accessible from the platform.	@jiajiny
					SW-77 Develop backend services to handle real-time messaging, including storing and retrieving chat histories.	The backend services should support real-time messaging with the ability to store and retrieve chat histories efficiently, ensuring reliable message delivery and quick retrieval times.	@jiajiny

					SW-78 Implement frontend functionality for displaying recent chats and chat history within the pop-up window.	The frontend functionality should enable the display of recent chats and complete chat history within the pop-up window.	@jiajiny
SW-141	Role-Based Dynamic Navigation for logged-in users: As a logged-in user, I want the navigation system to dynamically adjust based on my permissions and role (author, reviewer, editor), so that I can easily access modules specific to my tasks, including Submission, Profile, Review, Decision-Making, and Communication, enhancing my efficiency and experience on the platform.	8	Medium	Should have	SW-142 Design a dynamic navigation menu that adapts based on the user's role	Author, Editor and Reviewer each have their own navigation bar based on their role	@Yizhi Liao
					SW-143 Code the logic to display navigation options conditionally based on the logged-in user's role and permissions.	Display different navigation options and interfaces based on the logged-in user's role and permissions.	@Yizhi Liao
					SW-144 Integrate the dynamic navigation component into the platform's UI	Integrate the dynamic navigation component into the platform's UI to reflect user role-based permissions accurately.	@Yizhi Liao
SW-76	Data Security and Confidentiality: As a system administrator, I need to implement robust data protection measures to ensure the confidentiality of manuscripts and reviews. This includes implementing encryption protocols, access controls, and audit trails to safeguard sensitive data from unauthorized access or disclosure. By prioritizing data security, we can instill trust in our users and uphold the integrity of the review process.	8	Medium	Should have	SW-81 Implement Encryption for Manuscripts and Reviews.	The User must use token to access the article. The URL without a token could not be used for download the Article.	@Chuyuan Xu
					SW-82 Set Up Access Controls and Permissions	The user only can access the page of their role. Access to any page must require a token.	@Chuyuan Xu
					SW-83 Conduct Security Audit and Testing.	The security audit and testing should comprehensively evaluate the system for vulnerabilities, ensuring all identified issues are documented and resolved.	@Chuyuan Xu

Sprint 3 Milestones



Sprint 3 Burndown Chart



Sprint 3 Retrospective

Participants:

- Jiayu Yang
- Donghong Zhuang
- Jiajin Yang
- Yizhi Liao
- Yuxuan Zeng
- Chuyuan Xu

Related Meetings:

StandUp Meeting 08/05/2024

StandUp Meeting 15/05/2024

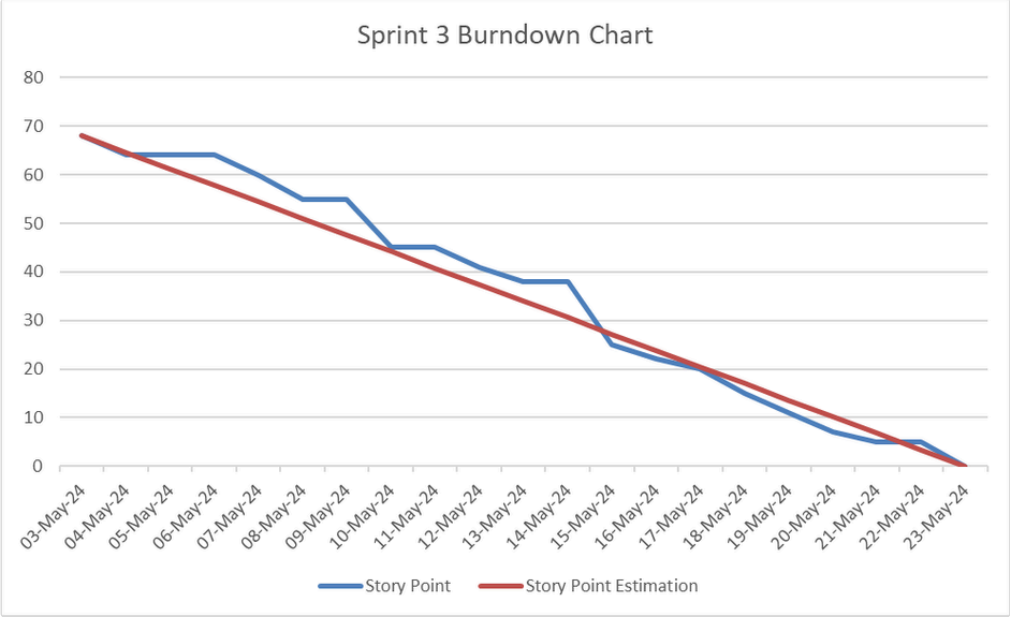
Mentor Meeting 08/05/2024

Mentor Meeting 15/05/2024

Client Meeting 08/05/2024

Sprint 3 Burndown Chart

This Burndown Chart shows the team's working process in Sprint 3. After the development in sprint 2, the team members have become very familiar with the project's workflow and the code work, so our development speed is almost consistent with the predicted rate. Since we need to do the final presentation on 5/22, we had almost completed the development of all functionalities by that time.



What worked well	What didn’t go well	What should we try doing next
The team have completed all features of sprint3 as per the client’s requirements.	The system testing took the team a lot of time.	The team should streamline the testing process.
The code has been reconstructed for clarification.	Problems or changes with third-party services or libraries the project depended on caused obstacles during integration and development.	The dependency should be carefully checked in sprint 4.
The system has deployed in the server, and working perfectly.	Time estimates for tasks were inaccurate, leading to the need for overtime work to meet deadlines, increasing team stress.	The team should engage all team members in the estimation process. Developers, testers, and other stakeholders can provide valuable insights and identify potential risks early on.
The project's progress adhered to the timeline, successfully avoiding any significant delays.		
The team tested all the features developed and found no bugs		

Sprint 3 Review

Project Documentation and Confluence Setup

- All team members contributed to updating the project documentation on Confluence, including detailed information on the progress and goals achieved during Sprint 3. This ensured that all team members had access to the latest updates and could understand the development progress.
- We refined our Status Progress diagram during Sprint 2, ensuring that the project vision and objectives remained clear and focused.

Project Management and Jira Usage

- @Donghong Zhuang continued to manage Jira, structuring it to reflect the tasks and progress of Sprint 2. This included updating task statuses and ensuring that all team members could track the project's progress.
- @Yizhi Liao updated the README file in the project's GitHub repository to reflect the latest developments, providing up-to-date project information for all visitors.

Backend Development

- Created the author review APIs, reviewer review APIs, and editor review APIs to handle user authentication.
- Constructed essential database tables: User, Article, Submission, SubmissionReview, ArticleAuthor, ChatSession, and Address.
- Create the Email notification feature.

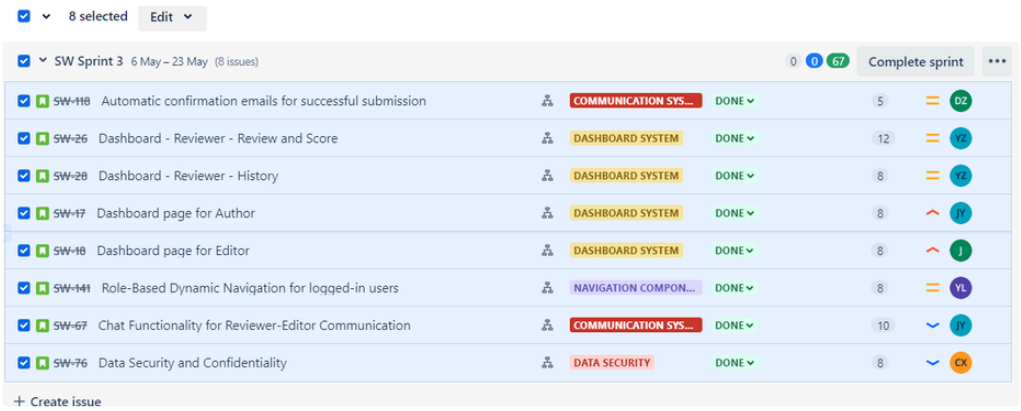
Frontend Development

- Developed the login and register pages using the React framework and connected them to the backend through the Internet.
- Implemented critical UI components, including a reviewer dashboard, review page, and editor dashboard (assign article/view article). Editor final decision page, invite reviewer, assign the article to the reviewer.
- Explored various open-source UI frameworks and decided to use "Antd" for its robustness and flexibility.

Deployment and Documentation

- Deployed the backend code to the cloud and launched the backend system using Docker, ensuring scalability and reliability.
- Continued updating the Sprint 3 documentation in Confluence, providing detailed records of the tasks completed and the progress made.
- Merged the branches developed for Sprint 3 into the main branch, deleted the development branches for Sprint 3, and started preparing for Sprint 4’s branches.
- Deploy the project into a live production environment accessible through a website.
- Updated the GitHub README file to reflect the latest developments and provide clear instructions for contributors.

Sprint 3 Complete


8 selected
Edit
SW Sprint 3 6 May – 23 May (8 issues)
0 0 67
Complete sprint
SW-110 Automatic confirmation emails for successful submission COMMUNICATION SYS... DONE 5
SW-26 Dashboard - Reviewer - Review and Score DASHBOARD SYSTEM DONE 12
SW-28 Dashboard - Reviewer - History DASHBOARD SYSTEM DONE 8
SW-17 Dashboard page for Author DASHBOARD SYSTEM DONE 8
SW-18 Dashboard page for Editor DASHBOARD SYSTEM DONE 8
SW-141 Role-Based Dynamic Navigation for logged-in users NAVIGATION COMPON... DONE 8
SW-67 Chat Functionality for Reviewer-Editor Communication COMMUNICATION SYS... DONE 10
SW-76 Data Security and Confidentiality DATA SECURITY DONE 8
Create issue

Sprint 4

Sprint Goal:

The goal for Sprint 4 is to deploy the Product and create the release.

Sprint Duration:

03 May - 07 June

What will be done in the Sprint?

1. Deploy the Project to the Client’s server.
2. Record a Demo video for the Project.
3. Sending the Zip file of the Project to the client.

Add label

Sprint 4 Planning

Objective

Delivering the Project to the client and helping them deploy the Project. **(Notice: Sprint 4 does not have any coding task, so there is no User story, Sprint Backlog, or burndown chart.)**

- **Client Meeting**:
  - Having a client meeting with our client for the project delivery.
  - Detailly showing the project to the client and help them understand this project.
  - Help them deploy the project to their server if they need.
- **Demo Video**: generated a 3-5mins (max) video demonstrating their product.

Deliverables

- Holding a client meeting for the project delivery, deployment, and explanation.
- Recording a demo video to demonstrate.

Product

This Section shows all the Web pages for this Product. You can access our Project by following the URL.

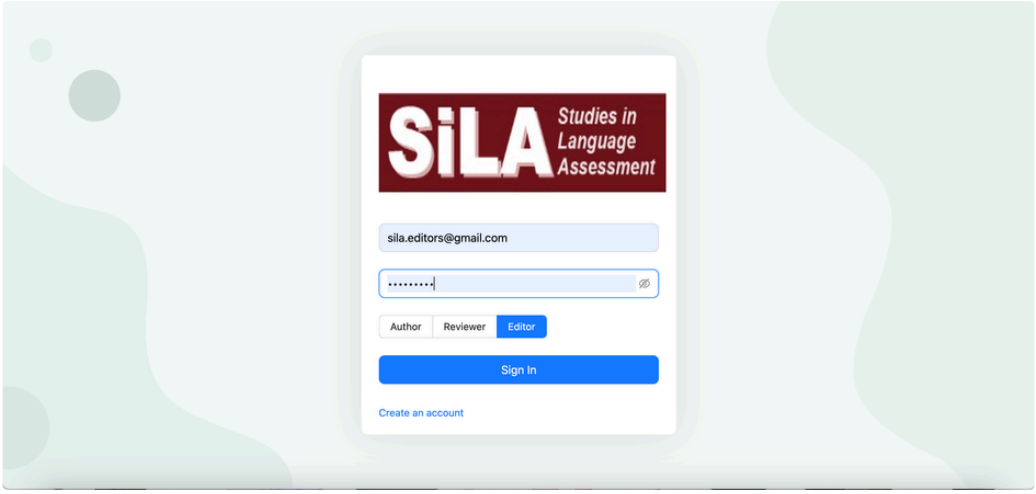
URL: http://3.27.174.20/login

Login and Regester

This Section is the Login and Register Page

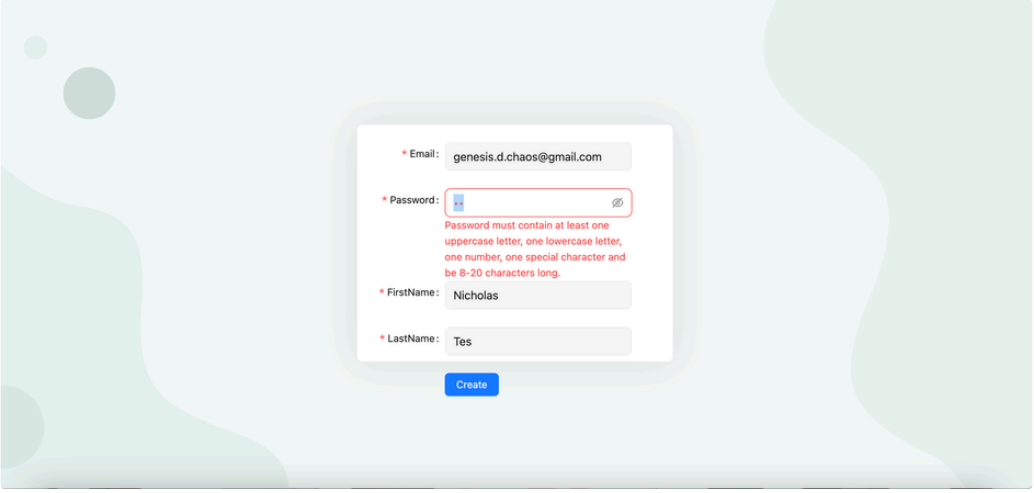
Login Page

The User can log in to the website by entering their email and password. There are 3 different Roles that the author can choose which are Author, Reviewer, and Editor. In addition, The create an account button navigates the user to the register page.



Regester Page

The user can register their email on the registration page by entering their email, password, and name, as shown in the image. The password must include uppercase letters, lowercase letters, numbers, and special characters.

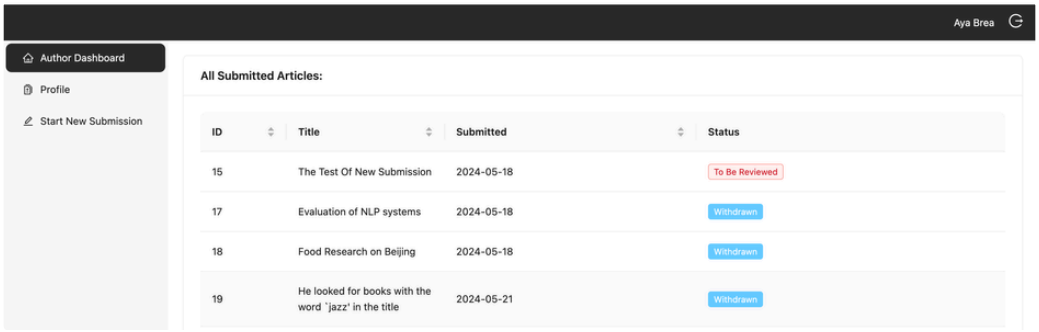


Author Page

This Section shows the web for the Author

Author Dashbaord

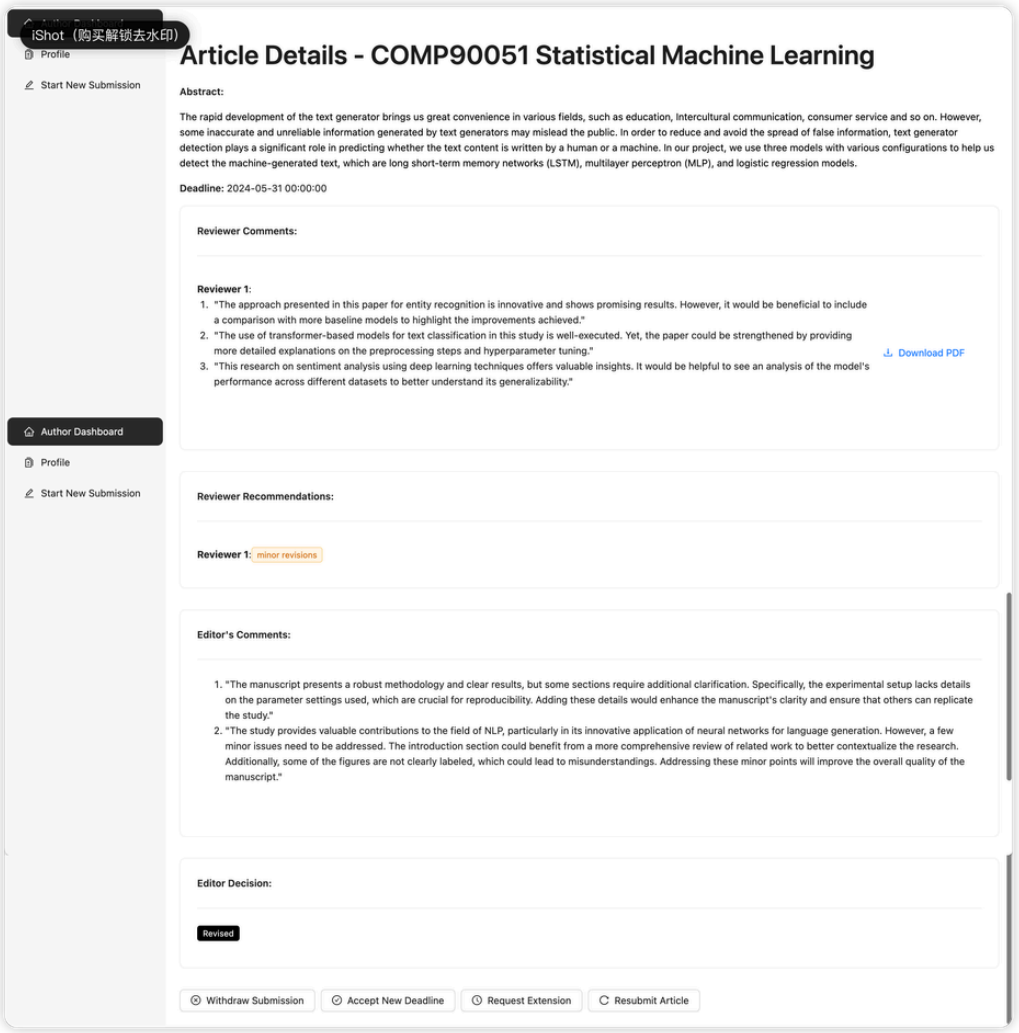
On the author dashboard page, authors can see all their submitted articles and the current status of each article. The author can click any row to access the article detail page for the chosen article. The left navigation bar allows authors to access the profile page and the submission page. The top right corner allows the author to log out.

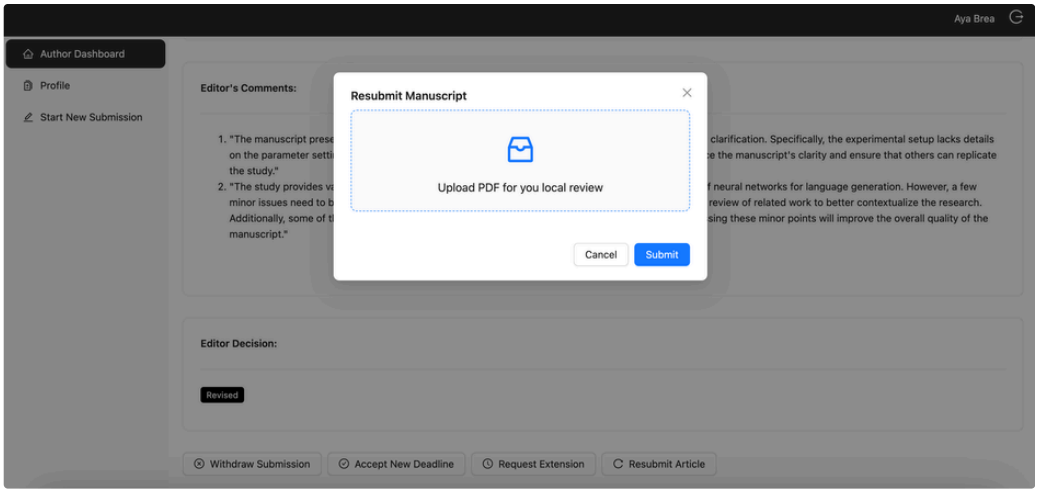


Article Detail

On the article detail page, authors can see the basic information about their article and the deadline (if the article is in revised status). If reviewers have commented on the article, the author can view the comments for the author. Additionally, the author can download the reviewer comment PDFs (if submitted). If the editor has provided comments, the author can see the editor's comments as well. The author can withdraw the article at any time by clicking the Withdraw button. If the article is in revised status, the author will see three additional buttons: Accept New Deadline, Request Extension (can only be used once), and Resubmit Article.

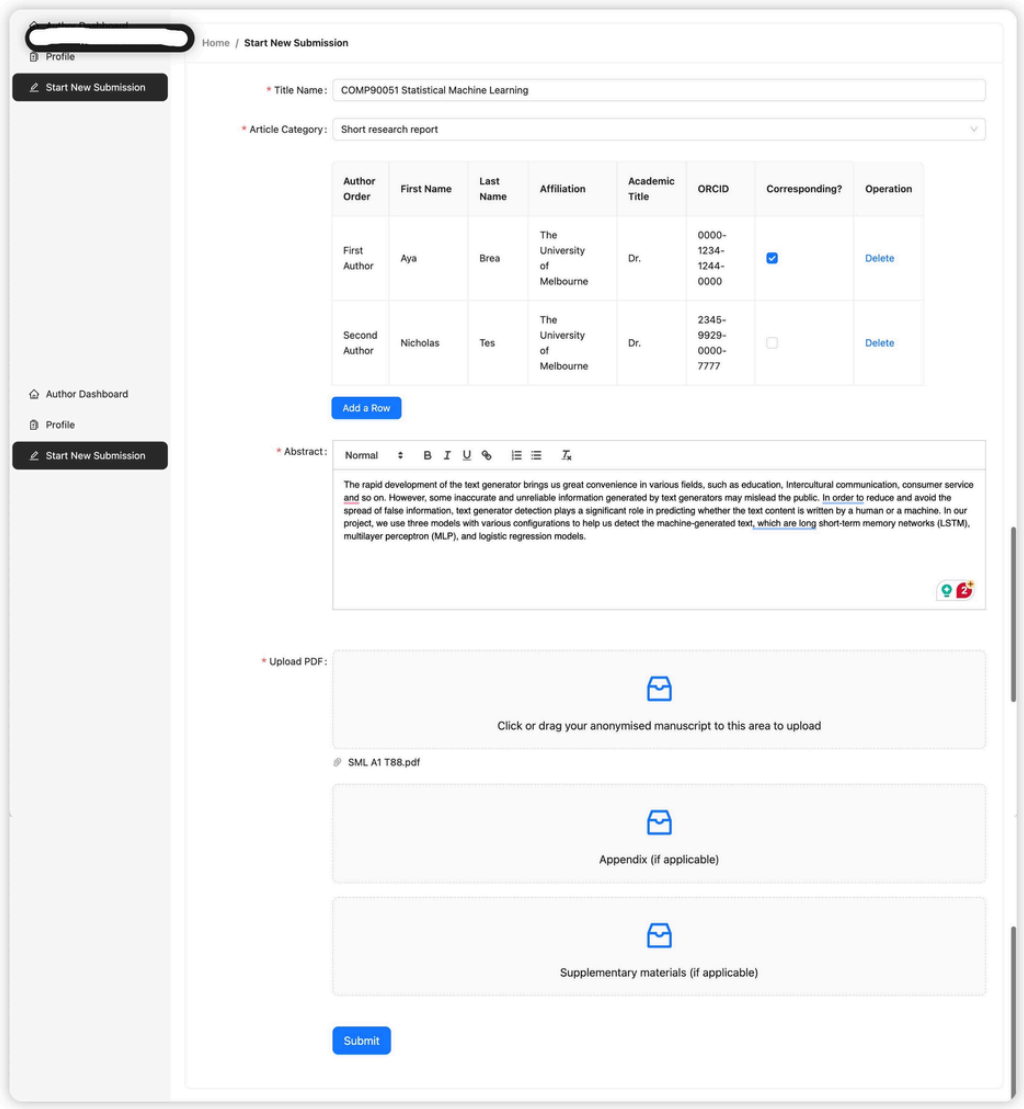
In the resubmit window, the author can upload their revised article.




Aya Brea
Author Dashboard
Profile
Start New Submission
Editor's Comments:
1. "The manuscript prese
clarification. Specifically, the experimental setup lacks details
on the parameter setti
e the manuscript's clarity and ensure that others can replicate
the study."
2. "The study provides va
f neural networks for language generation. However, a few
minor issues need to b
review of related work to better contextualize the research.
Additionally, some of t
sing these minor points will improve the overall quality of the
manuscript."
Resubmit Manuscript
Upload PDF for you local review
Cancel
Submit
Editor Decision:
Revised
Withdraw Submission
Accept New Deadline
Request Extension
Resubmit Article

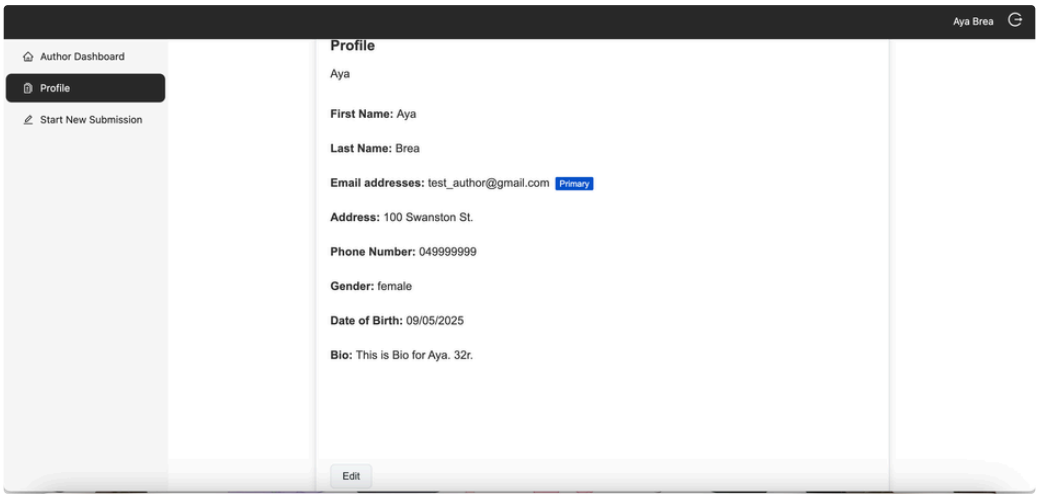
Article Submission

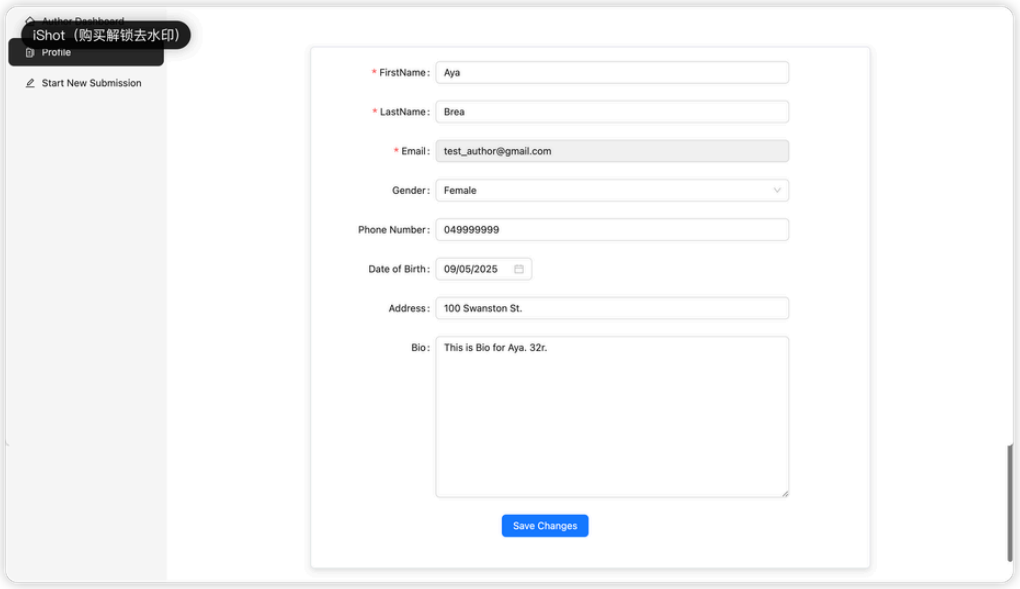
Authors can submit new articles on the Start New Submission Page. Authors need to enter their article title, abstract, and author information (more authors can be added using the Add a Row button). Then, authors can upload their article PDF (mandatory), appendix, and supplementary materials (optional).



Profile

On the profile page, authors can see their name, email, gender, birthday, phone number, and bio. Authors can also edit their personal information on this page.





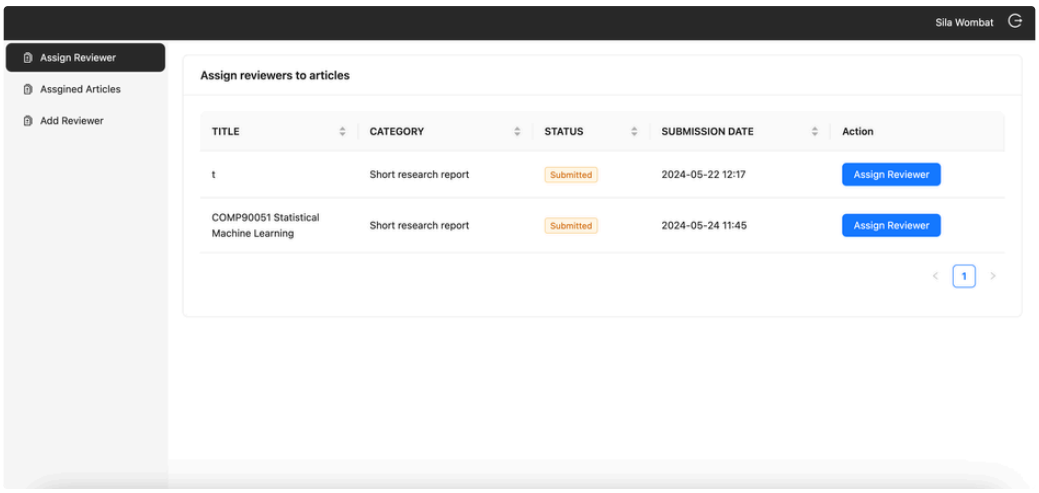
Editor Page

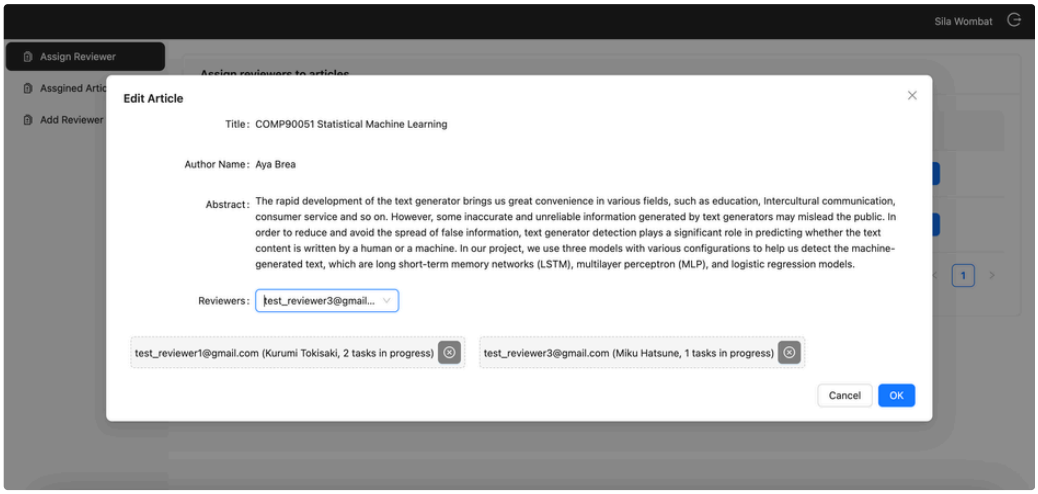
This section shows the Web Page for Editor

Assign Reviewer

On this page, all uploaded articles that have not yet been assigned to a reviewer can be seen. By clicking the Action button, the editor can assign the review task to one or more reviewers.

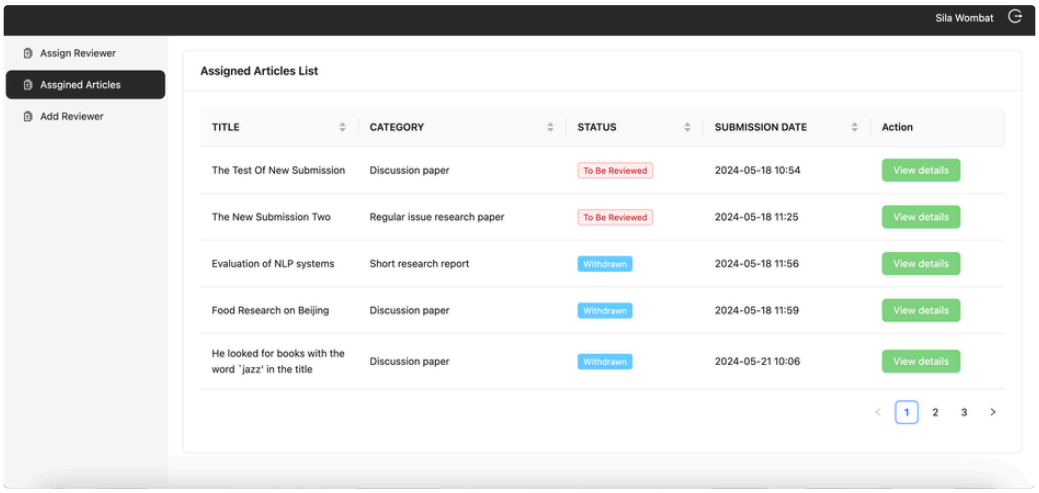
In the assign window, the editor can see the article title, the first author, and the abstract. Then, the editor can select the reviewer(s) to assign. When the editor selects any reviewer, they can see how many articles that reviewer is currently reviewing.





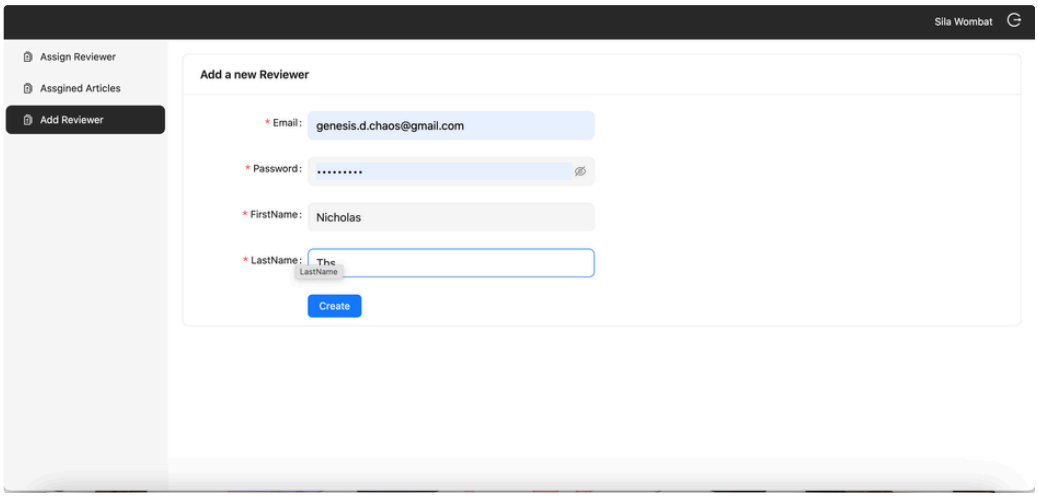
Assigned Articles

On the Assigned Articles page, the editor can see all articles that have been assigned to reviewers. By clicking the View Details button, the editor can access the details page of the article.



Add Reviewer

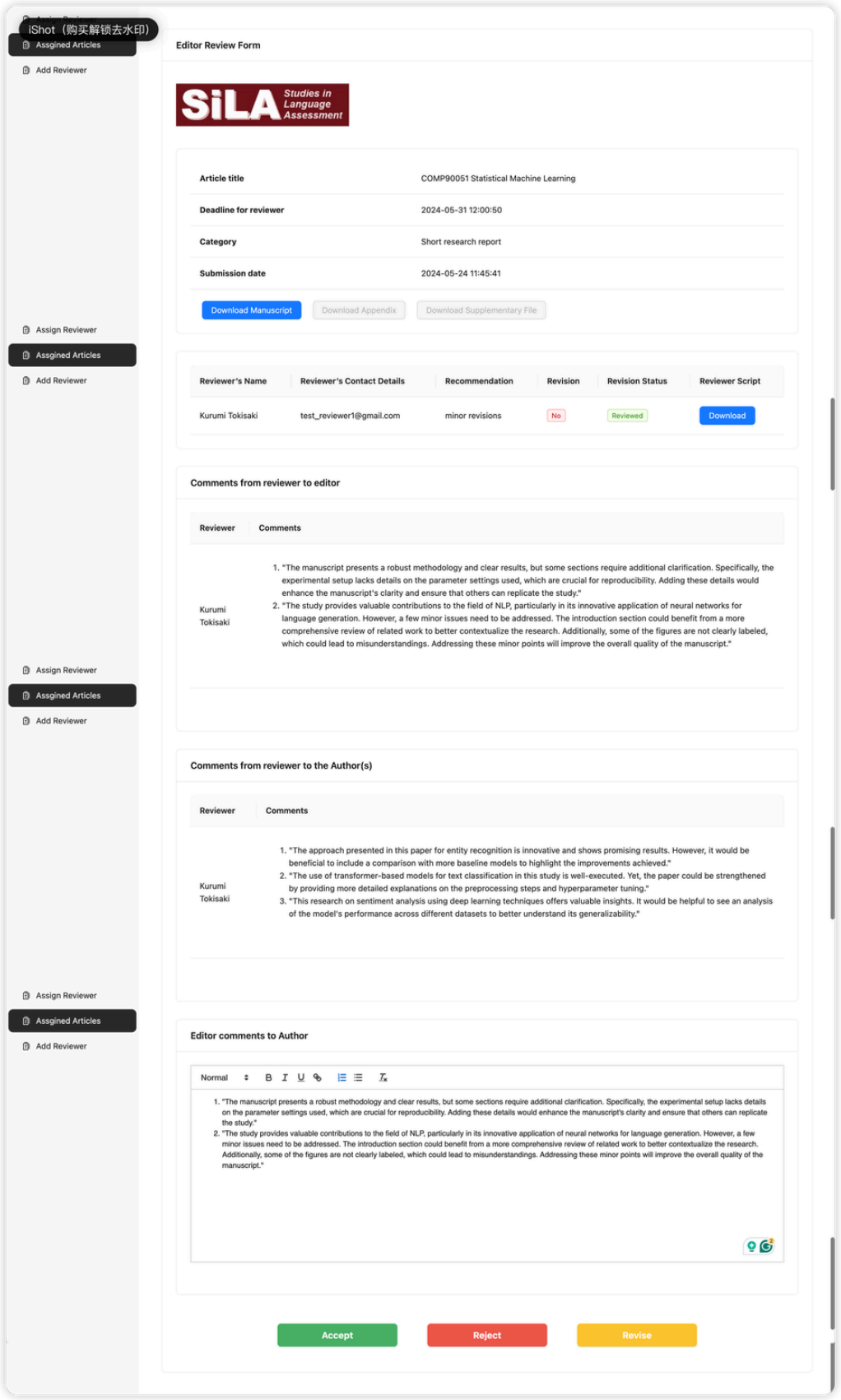
The editor can invite others to become reviewers through the Add Reviewer page. On this page, the editor needs to enter the invitee's email, password, and name to complete the invitation. The reviewer will receive an invitation email.

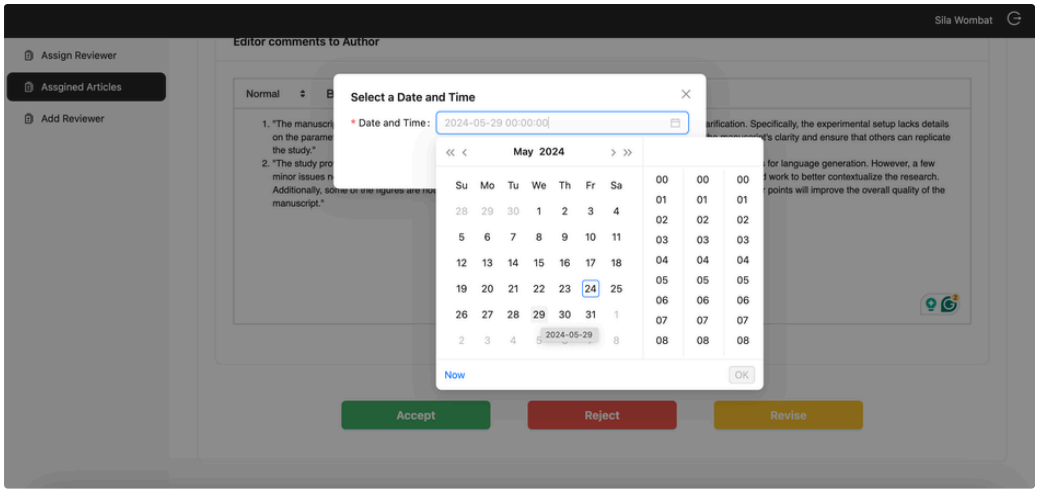


Editor Decision

On the editor decision page, the editor can view detailed information about the article and download the article, appendix (if available), and supplementary files (if available). If the article is assigned to reviewers, the editor can also see the detailed information of the reviewers. If reviewers have commented on the article, the editor can view the comment for the editor (optional) and the comment for the author (mandatory). Additionally, the editor can download the reviewer comment PDFs (if submitted). Once all reviewers have submitted their comments, the editor can make a final decision on the article: Accept, Reject, or Revise.

When the editor selects "Revise," the editor will set a submission deadline for the article.


iShot（购买解锁去水印）
Assgined Articles
Add Reviewer
Assign Reviewer
Assgined Articles
Add Reviewer
Assign Reviewer
Assgined Articles
Add Reviewer
Assign Reviewer
Assgined Articles
Add Reviewer
Editor Review Form
SiLA Studies in Language Assessment
Article title
COMP90051 Statistical Machine Learning
Deadline for reviewer
2024-05-31 12:00:50
Category
Short research report
Submission date
2024-05-24 11:45:41
Download Manuscript
Download Appendix
Download Supplementary File
Reviewer's Name
Reviewer's Contact Details
Recommendation
Revision
Revision Status
Reviewer Script
Kurumi Tokisaki
test_reviewer1@gmail.com
minor revisions
No
Reviewed
Download
Comments from reviewer to editor
Reviewer
Comments
Kurumi Tokisaki
1. "The manuscript presents a robust methodology and clear results, but some sections require additional clarification. Specifically, the experimental setup lacks details on the parameter settings used, which are crucial for reproducibility. Adding these details would enhance the manuscript's clarity and ensure that others can replicate the study."
2. "The study provides valuable contributions to the field of NLP, particularly in its innovative application of neural networks for language generation. However, a few minor issues need to be addressed. The introduction section could benefit from a more comprehensive review of related work to better contextualize the research. Additionally, some of the figures are not clearly labeled, which could lead to misunderstandings. Addressing these minor points will improve the overall quality of the manuscript."
Comments from reviewer to the Author(s)
Reviewer
Comments
Kurumi Tokisaki
1. "The approach presented in this paper for entity recognition is innovative and shows promising results. However, it would be beneficial to include a comparison with more baseline models to highlight the improvements achieved."
2. "The use of transformer-based models for text classification in this study is well-executed. Yet, the paper could be strengthened by providing more detailed explanations on the preprocessing steps and hyperparameter tuning."
3. "This research on sentiment analysis using deep learning techniques offers valuable insights. It would be helpful to see an analysis of the model's performance across different datasets to better understand its generalizability."
Editor comments to Author
Normal
1. "The manuscript presents a robust methodology and clear results, but some sections require additional clarification. Specifically, the experimental setup lacks details on the parameter settings used, which are crucial for reproducibility. Adding these details would enhance the manuscript's clarity and ensure that others can replicate the study."
2. "The study provides valuable contributions to the field of NLP, particularly in its innovative application of neural networks for language generation. However, a few minor issues need to be addressed. The introduction section could benefit from a more comprehensive review of related work to better contextualize the research. Additionally, some of the figures are not clearly labeled, which could lead to misunderstandings. Addressing these minor points will improve the overall quality of the manuscript."
Accept
Reject
Revise

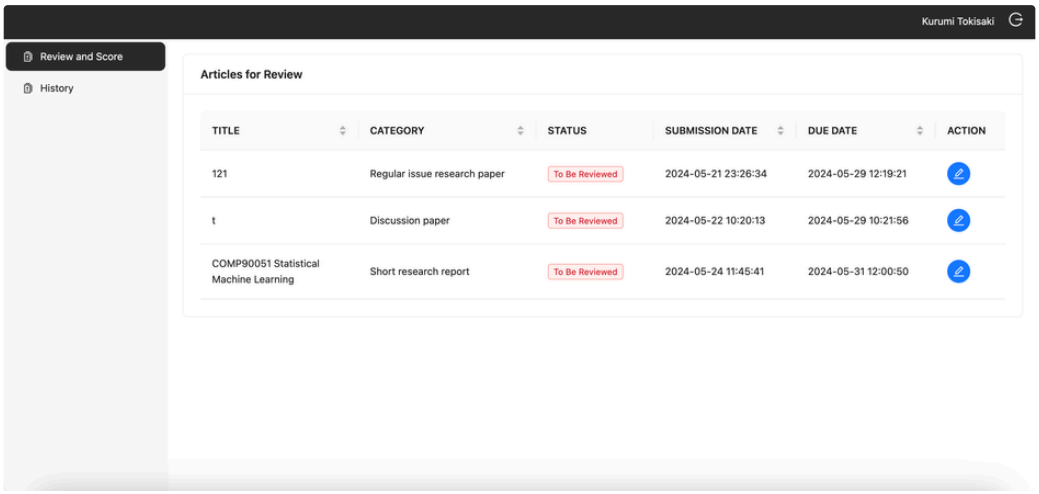

Sila Wombat
Assign Reviewer
Assgined Articles
Add Reviewer
Editor comments to Author
Normal
Select a Date and Time
* Date and Time :
2024-05-29 00:00:00
May 2024
Su Mo Tu We Th Fr Sa
Now
OK
Accept
Reject
Revise

Reviewer Page

This section shows the web page for the reviewer.

Review and Score

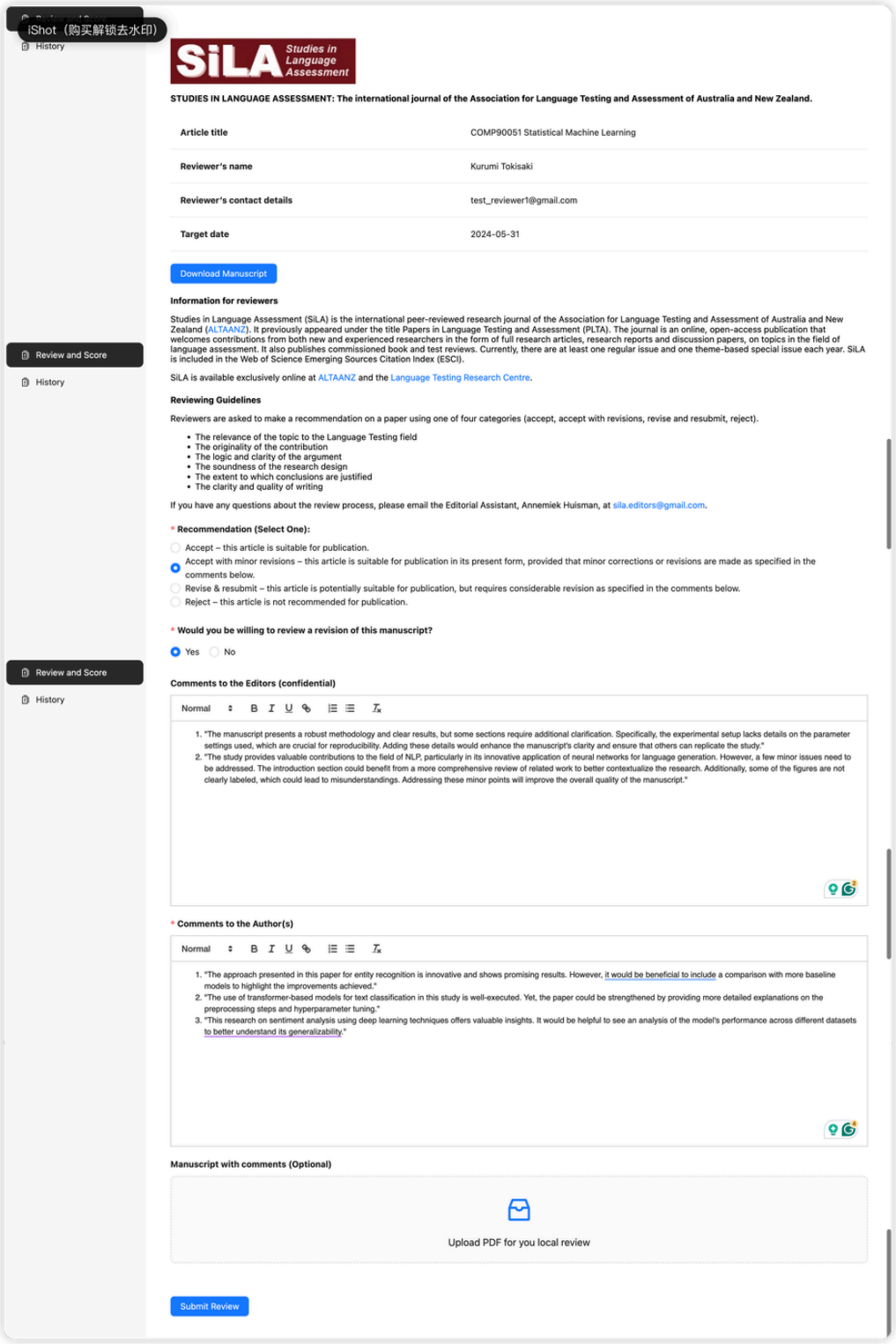
On this page, reviewers can see all the articles assigned to them that have not yet been reviewed, along with the basic information about each article. By clicking the Action button, the reviewer can enter the review page for the selected article.



Review Page

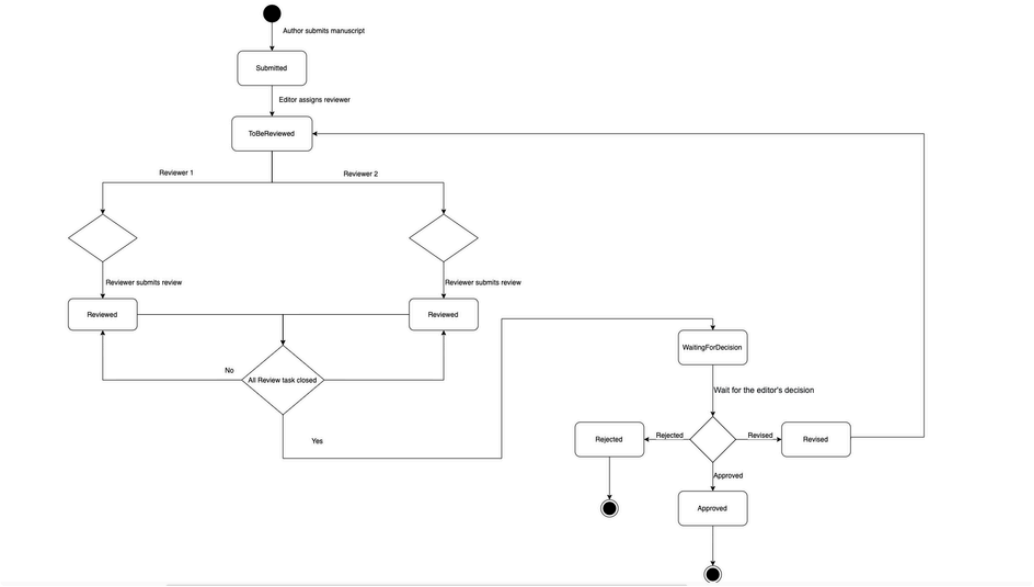
The review page displays some basic information about the reviewer and the article. The Download Manuscript button allows the reviewer to download the manuscript. According to the guidelines on the review page, the reviewer can choose their recommendation for the article: Accept, Accept with minor revisions, Revise & resubmit, Reject.

Then, the reviewer can indicate their willingness to review the revised article. They can leave a comment for the editor (optional) and a comment for the author (mandatory). If the reviewer has a PDF with additional comments, they can upload it in the upload section.


iShot（购买解锁去水印）
History
Review and Score
History
Review and Score
History
Studies in Language Assessment
STUDIES IN LANGUAGE ASSESSMENT: The international journal of the Association for Language Testing and Assessment of Australia and New Zealand.
Article title
COMP90051 Statistical Machine Learning
Reviewer's name
Kurumi Tokisaki
Reviewer's contact details
test_reviewer1@gmail.com
Target date
2024-05-31
Download Manuscript
Information for reviewers
Studies in Language Assessment (SiLA) is the international peer-reviewed research journal of the Association for Language Testing and Assessment of Australia and New Zealand (ALTAANZ). It previously appeared under the title Papers in Language Testing and Assessment (PLTA). The journal is an online, open-access publication that welcomes contributions from both new and experienced researchers in the form of full research articles, research reports and discussion papers, on topics in the field of language assessment. It also publishes commissioned book and test reviews. Currently, there are at least one regular issue and one theme-based special issue each year. SiLA is included in the Web of Science Emerging Sources Citation Index (ESCI).
SiLA is available exclusively online at ALTAANZ and the Language Testing Research Centre.
Reviewing Guidelines
Reviewers are asked to make a recommendation on a paper using one of four categories (accept, accept with revisions, revise and resubmit, reject).
The relevance of the topic to the Language Testing field
The originality of the contribution
The logic and clarity of the argument
The soundness of the research design
The extent to which conclusions are justified
The clarity and quality of writing
If you have any questions about the review process, please email the Editorial Assistant, Annemiek Huisman, at sila.editors@gmail.com.
* Recommendation (Select One):
Accept – this article is suitable for publication.
Accept with minor revisions – this article is suitable for publication in its present form, provided that minor corrections or revisions are made as specified in the comments below.
Revise & resubmit – this article is potentially suitable for publication, but requires considerable revision as specified in the comments below.
Reject – this article is not recommended for publication.
* Would you be willing to review a revision of this manuscript?
Yes
No
Comments to the Editors (confidential)
Normal
1. "The manuscript presents a robust methodology and clear results, but some sections require additional clarification. Specifically, the experimental setup lacks details on the parameter settings used, which are crucial for reproducibility. Adding these details would enhance the manuscript's clarity and ensure that others can replicate the study."
2. "The study provides valuable contributions to the field of NLP, particularly in its innovative application of neural networks for language generation. However, a few minor issues need to be addressed. The introduction section could benefit from a more comprehensive review of related work to better contextualize the research. Additionally, some of the figures are not clearly labeled, which could lead to misunderstandings. Addressing these minor points will improve the overall quality of the manuscript."
* Comments to the Author(s)
Normal
1. "The approach presented in this paper for entity recognition is innovative and shows promising results. However, it would be beneficial to include a comparison with more baseline models to highlight the improvements achieved."
2. "The use of transformer-based models for text classification in this study is well-executed. Yet, the paper could be strengthened by providing more detailed explanations on the preprocessing steps and hyperparameter tuning."
3. "This research on sentiment analysis using deep learning techniques offers valuable insights. It would be helpful to see an analysis of the model's performance across different datasets to better understand its generalizability."
Manuscript with comments (Optional)
Upload PDF for you local review
Submit Review

Status Digram

When the author submits the article, its status changes to "**Submitted.**" Once the editor assigns the article to reviewers, the status changes to "**To Be Reviewed.**" For reviewers, after they review the article, the status changes to "**Reviewed.**" When all reviewers have reviewed the article, the status changes to "**Waiting For Decision.**" In this status, the editor can decide to **Approve, Reject,** or **Revise** the article. Once the editor makes a decision, the article will change to the corresponding status. If the editor selects **"Revise,"** the author needs to upload the revised article, and the status will change back to "**To Be Reviewed,**" restarting the review process. The author can choose to withdraw the article at any time and in any status, after which the article status will permanently become "**Withdrawn.**"



Ethical Considerations

In our software project, ethical considerations play an important role in guiding the development and deployment processes. We are committed to ensuring that our software promotes fairness, respects user privacy, and operates transparently.

Our ethical analysis is based on the "Ethics-Aware Software Engineering" framework, and our team has met the conditions of this framework during development.

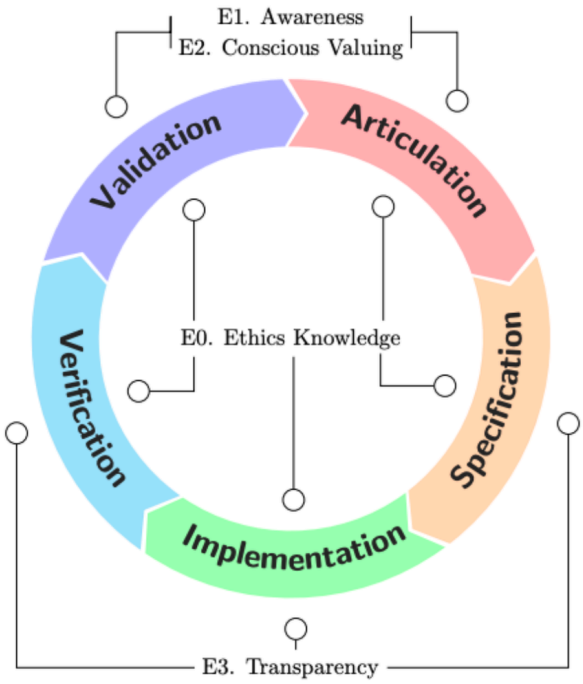


Figure 1: Method for Ethics-Aware Software Engineering

Ethical Issue	Issue Analysis	Potential Impact on Users	How to Resolve?	Team Action
User Privacy & Trust	The submission system must safeguard the confidentiality of the authors' work. Only relevant parties such as assigned reviewers and editors should have access to the manuscript.	If a manuscript is accessed by unauthorized persons, the author's ideas may be leaked prematurely, resulting in intellectual property theft or misuse. Authors may lose trust in our system and no longer want to use it.	Implemented strong authorization and access control mechanisms to ensure that only authorized reviewers and editors have access to the manuscript. The backend and servers we use encrypt all sensitive data to keep it	We regularly check access records to ensure there is no unauthorized access. We conduct periodic security audits and updates to our security protocols to identify and rectify vulnerabilities.

			secure. In addition, we have also set up three different user login identities. Different identities have different levels of permissions, and user passwords are hashed and encrypted to ensure user security.	
Transparency in Review Process	The blind review process must be transparently managed to maintain trust in the fairness and integrity of the publication process.	Lack of transparency in the review process can make authors believe that the review results are unfair, affecting their trust in the system. This perception of unfairness can deter authors from submitting their work and harm the platform's reputation.	Create clear documentation on how the review process works and how decisions are made, which can be audited if needed while preserving the anonymity of the authors and reviewers. In addition, it also conducts double-blind testing to ensure that the review process is transparent and fair.	We regularly update documents to ensure that the review process is always fair and transparent. We gather feedback from users to improve the documentation and review process.
Bias Prevention	The development team must ensure that no inherent biases are introduced in the system that could affect the review and selection process.	Bias in the system can lead to unfair rejection of excellent works, affecting authors' confidence and the system's credibility. Biased review results can distort the publication landscape, impacting the integrity of academic discourse.	Include a feature where the original submission is available alongside the review to check for any biases, and provide the option for the editor to override the review outcomes if necessary. In addition, authors are not allowed to show their names in the article, and reviewers who have no contact with the author will be assigned to	We continually improve our systems to eliminate any possible bias by incorporating regular training sessions on bias awareness for our team. We also implement automated checks to flag potential biases in reviews.

			review the manuscript.	
Device Compatibility	The system should be accessible to users with different devices to avoid any form of digital discrimination.	Incompatibility with certain devices can prevent users from accessing or using our system, resulting in a poor experience and reduced productivity. This can alienate users with older or less common devices, limiting the system's reach and inclusivity.	Optimize the front end for low-resource use and perform complex operations on the server-side. Regularly test the application on different devices and browsers for compatibility.	We continue to conduct compatibility testing and solve problems in a timely manner to ensure that users have a good experience on various devices. We maintain a comprehensive device and browser testing matrix to cover as many scenarios as possible.
Accessibility	Ensure that the website is accessible to users with different abilities and from various technological backgrounds.	If a website does not meet accessibility standards, some users may be unable to use the system, thereby preventing them from submitting manuscripts. This affects their experience.	Adhere to web accessibility standards to ensure that the website is usable by people with disabilities and those using older technologies.	During the development process, we conduct accessibility checks to ensure that every feature complies with standards.
Visual Accessibility	The system should be usable by individuals with visual impairments to ensure equitable access to the system’s functionalities.	If the system is not user-friendly for people with visual impairments, they may not be able to access content, which affects their productivity and satisfaction.	Follow web accessibility guidelines to accommodate users with visual impairments. Implement design features such as high contrast modes and compatibility with screen readers.	We continually update the interface using feedback from user testing, ensuring visual accessibility is a priority in design updates. We ensure that our development team is trained in best practices for visual accessibility.
Data Security	Securely handle the storage and transmission of sensitive data such as personal information and manuscript content.	In the event of a data breach, users' personal information may be misused, leading to identity theft. If the contents of the	Use encryption and secure data handling practices to protect data integrity and confidentiality.	We use safe and reliable back-end and servers to ensure that the system is always safe. Regular security audits and

		manuscript are leaked, intellectual property rights may be lost.		updates to our security protocols are conducted to ensure ongoing protection.

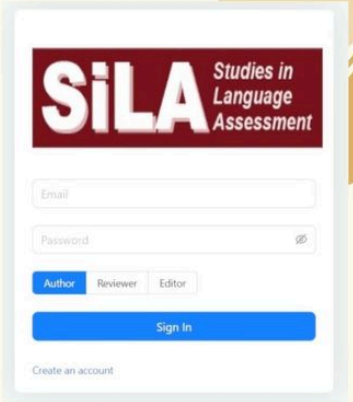
Presentation Slides

Outline

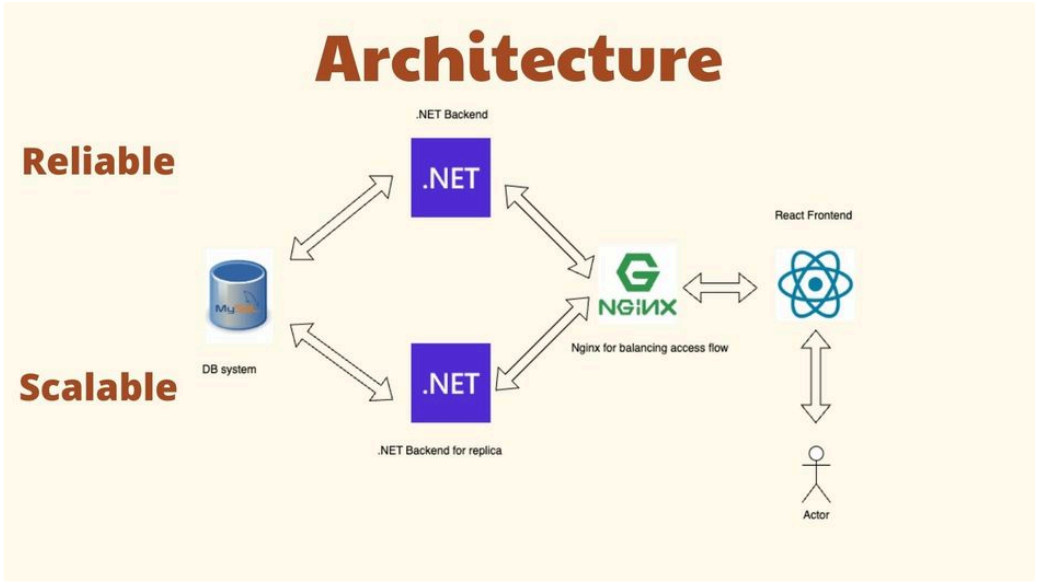
- Background
- Architecture
- Demo
- Process
- Deliverable
- Future Work

Background

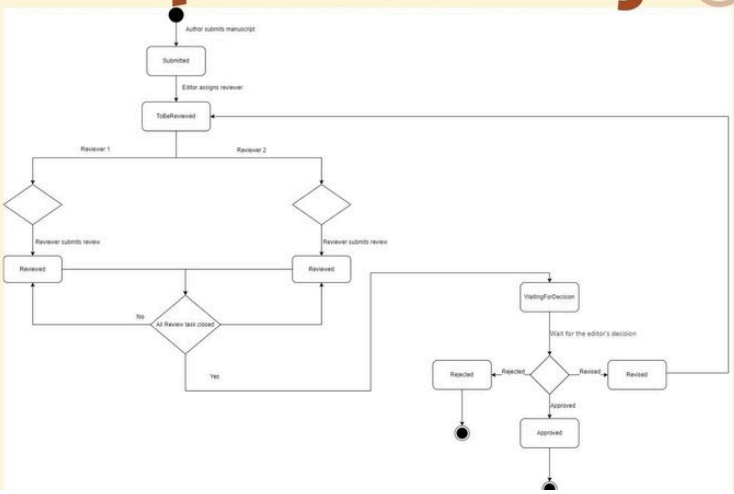
- Develop a website for uploading and review articles for SiLA
- Three login identities




Architecture
.NET
NGINX
MySQL
aws
docker


Architecture
Reliable
Scalable
.NET Backend
.NET
.NET
DB system
NGINX
Nginx for balancing access flow
React Frontend
.NET Backend for replica
Actor

Manuscript Status Diagram



Demo

Project Process

Sprint 2:

- Home page for All Users
- Database for Account system
- Database for Manuscripts
- Login page for different user types
- Register Page for Author
- Dashboard page for Author to Start New Submission
- Review page for Reviewer
- Decision making page for Editor

Question Time

Deliverable

- Zip File - Front-end and Back-end code
- Readme File
- Deploy Servers

Future Work

- Chat box
- Email to Authors
- Delivery

Project Process

Sprint 3:

Dashboard - Reviewer - Review and Score
Dashboard - Reviewer – History
Dashboard page for Author
Dashboard page for Editor
Data Security and Confidentiality
Role-Based Dynamic Navigation for logged-in users

Cyber Security

Approach Used

- **Cross-origin resource sharing**

  Cross-Origin Resource Sharing (CORS) is a protocol that enables web pages to request restricted resources from a server located on a different domain than the domain that delivered the original web page. While web pages can embed cross-origin resources like images, stylesheets, scripts, iframes, and videos without special permissions, certain "cross-domain" requests such as Ajax are blocked by the Same-Origin Policy by default. CORS provides a method for browsers and servers to communicate to ascertain if it is safe to permit the cross-origin request. This approach allows greater flexibility and functionality than same-origin requests alone, yet offers more security than allowing all cross-origin requests indiscriminately.

- **Nginx Proxy Manager**

  Nginx Proxy Manager is a graphical user interface (GUI) tool that simplifies the process of managing Nginx proxy configurations for users who may not be familiar with command-line operations or Nginx's native configuration syntax. This tool is particularly useful for handling the complexities of setting up Nginx as a reverse proxy, load balancer, or for serving static websites, especially in a home or small business environment.

- **JSON Web Token(JWT)**

  One of the most crucial security features of a JSON Web Token (JWT) is its signature. A JWT consists of three parts: Header, Payload, and Signature. The signature is generated using the algorithm specified in the header (such as HMAC SHA256 or RSA), which ensures that the JWT has not been altered during transmission.

  JWTs usually include an expiration time (exp claim). This timestamp specifies when the token expires. Having a short validity period reduces the risk of the token being stolen and misused, because even if the token is compromised, it is only valid for a short duration.

- **AWS Identity and Access Management**

  AWS Identity and Access Management (IAM) is a service provided by Amazon Web Services designed to help administrators securely control access to AWS resources. IAM allows you to manage users, security credentials such as passwords or keys, and permissions to access AWS resources.

  IAM can control what actions a user or group of users can perform on AWS resources. For instance, a permission might allow a user to launch a manuscript instance but not modify it.

Cyber Security Analysis Table

Likelihood in five categories: Rare; Unlikely; Possible; Likely; and Almost Certain.

Impact in four categories: low, medium, high, very high.

Risk ID	Description	likelihood	likelihood description	Impact	Impact Description	Solution
R1	Account Stolen	Likely	It is very possible if the user has a weak password.	very high	The account stolen will greatly threaten user information security. The attacker can access all the information and functions of that account, and cause countless losses to the user.	In the back-end, Setting password complexity requirements. Detailly, the user must set their password that at less 6 characters which include at least one uppercase, one lowercase, one digit, and one special character.
R2	SQL Injection	Likely	SQL injection is a common attack	very high	The hackers may input some SQL statements	We use .Net's Entity Framework Core tool to

			technology for the hackers to attack our website.		and directly send to our back-end to modify our database. It may cause our data leaked or database destroyed.	avoid SQL injection attack. This tool has reliable in-build mechanisms to avoid attacks.
R3	Database Password Leak	Possible	It is possible if our database suffered attacks and the hackers get the data that are stored in database.	very high	If there is no protection method for the passwords that are stored in database, once the hackers stole the data in our database, then they can directly get our accounts' passwords.	We use Salted Hash method to avoid our passwords leak. This method stores a pre-processed hash password instead of storing the password directly to the database.
R4	Manuscript Stolen	Possible	User-uploaded manuscripts are one of the most important data on this website. Attackers targeting the SiLA website have a strong motivation to steal user manuscripts.	high	Manuscripts stolen from can severely impact our website's integrity and trustworthiness. This damages SiLA's reputation and affects its ability to attract both contributors and subscribers.	Using Amazon IAM (Identity and Access Management) to prevent data security. It enhances security by strictly controlling who can access manuscripts on AWS-hosted platforms
R5	Self Reviewing	Unlikely	If the author steals the reviewer's token, he might be able to review his manuscript. However, it is unlikely, since it is hard to get the reviewer's token, and considering the risk of doing that most of the authors do not have the motivation for it.	low	Even if an author can review their own paper by stealing tokens from reviewers and editors, there is still a high likelihood that reviewers and editors will notice the abnormality in the review process and address it promptly.	Using JWT(JSON Web Token) Manager to generate the on-time token, and the token will refresh and update in a short period which make sure other people can not use only one token to review the manuscript.
R6	Communication Attacks	Possible	It is possible to suffer attacks when the front-end communicates with the back-end.	high	When the communications suffer attacks, it may cause data leak. The hackers may use these data to threaten the security of our website.	HTTPS will be used to avoid this kind of attack. HTTPS uses TLS/SSL to encrypt data and also uses message digests to ensure that data has not been tampered with during transmission. HTTPS also includes verification of the identities of both parties in the communication.
R7	Data Lost by Server Crash	Rare	Although our server is stable, there is still a	high	If the server crashed, it means that our data may lost. It has a high impact	We use data backup to prevent server crashes. That is, we will

			slight possibility of crash.		to our website if we lost our data.	automatically back up the data at regular intervals, so that when the server crashes, our data will not be lost.