IBM Lotus Symphony

COS 730 - Assignment 2

Robyn C. Hancock
u19068035

23 May 2024

Logo [1]

1 Introduction

1.1 History

IBM Lotus Symphony is a legacy office suite that consists of three office applications: a word processor, a spreadsheet program and a presentation program [2].

IBM Lotus Symphony was first launched in September, 2007, as a free suite of programs for Windows, Mac and Linux computers, and positioned itself as a competitor to Microsoft Office ([3], [4]).

IBM Lotus Symphony contains all the major functions of office suite software, including common functions, such as a spell checker, templates for the presentation application and charts for the spreadsheet application ([5], [6], [4]).

A key strength of the software was that it used Open Document format (ODF), allowing the software to read and work with documents of various formats ([7], [4]), as opposed to Microsoft Office, which restricted file types to a single system, specifically the Microsoft Office format [3]. This Open Document format (ODF) allowed for great flexibility in what documents can be worked with [8], whilst improving user-friendliness, as it was reported that many office workers spent unnecessary time attempting to convert documents to a compatible version for other office suite software [7]. Another strength, was that each of the three applications (word processor, spreadsheet and presentation application) could be accessed from a single-window interface [5].

A weakness of the IBM Lotus Symphony software was that it could not compete with Microsoft Office which dominated the desktop, with 80 percent of organisations using Microsoft Office ([9], [6]). Additionally, IBM Lotus Symphony requires a large amount of system resources [6], whilst not being as complete in terms of applications available compared to other office suites [5].

IBM announced in January 2012 that version 3.0.1. of IBM Lotus Symphony will be the last version based on the OpenOffice code base, with IBM contributing the Lotus Symphony code to the Apache Software Foundation [10]. Effectively, ending IBM Lotus Symphony.

The industry that uses the software is any person or organization that uses Microsoft Office or other office application suites. This industry is still the same today as it was in 2007 when the software was released ([11], [4]).

A potential improvement, would be to take the software suite entirely web-based, whilst maintaining the Open Document Format (ODF), and expanding it to include recent format types and extensions that have emerged since the software was discontinued in 2012. Additionally, the software will be modified to work with cloud storage, to improve portability, as the user would be able to store documents in the cloud.

1.2 Purpose

To provide the basic features of an office suite, particularly the word processor, the spreadsheet application and the presentation application, as a web-based system for users of all types.

1.2.1 The Business Need

Office suites are an important part of day-to-day life for many people around the world. Many of the current office suites available require either a user account or need to be installed on a device. The new IBM Lotus Symphony aims to provide the basic features of an office suite completely in a web-based format without the need to have a user account. Businesses also require flexibility when it comes to document formats and extensions.

1.3 Scope

The scope of the project would be to create a web-based office suite providing a word processor, a spreadsheet application and a presentation application.

1.4 Vision

To revitalize the legacy system IBM Lotus Symphony as a web-based system offering the same functionality as the old system whilst modernizing the system by including natural language processing capabilities and cloud-interoperability features.

1.5 Objectives

- To update the legacy system as a web-based office suite providing a word processor, a spreadsheet application and a presentation application.
- To incorporate cloud access to allow for the retrieval and storing of documents in cloud applications, such as Google Drive.
- To include Natural Language Processing functionality to improve spelling, grammar and writing for the users.
- To provide an easy to use web-based application designed to offer an alternative to application-based office suites.

2 User Characteristics

2.1 The Casual User

The Casual User is a user that does not often use an office suite. These users are not interested in complex tools, but rather the core tools and systems to ensure that the work that they need to complete, is completed. An example of the casual user would be a mechanic.

Purpose
- The user would use the system to perform basic word document processing, basic presentation viewing and creation, and basic spreadsheet management.

Educational Level
- No to limited formal and/or informal education.
- A basic level of education of computer usage.

Experience
- No to limited experience with office suite software.

Expertise
- Any level of expertise - from completely new to computers to having a basic understanding of computers.

Technical Skills
- Limited technical skills.
- Skill set limited to operating a computer and using the base tools/-functions (for example: copy, paste, print) of office suite software.

2.2 The Active User

The active user often uses office suite applications in their day-to-day activities. These users have a good understanding of how to use a computer and office suite applications. An example of this user would be someone in the field of marketing.

Purpose
- To use software to create, view and modify documents, create charts and view trends in spreadsheets, and to create well-designed presentation slides.

Educational Level
- Some formal education, with a good level of confidence in using office suite software.

Experience
- Previous experience using computers and office suite software.

Expertise
- This user would have a good understanding of how to use a computer and would have in-depth knowledge of how to use office suite software to achieve most of their day-to-day activities.

Technical Skills
- Moderate to high technical skills.
- Skill set is such that the user can work comfortably on any computer or software, particularly office suite software.

3 User's Story

US 1 Opening A Document

User Story As a manager of a large team, I receive many documents daily from different people using different office suite software. I want to be able to open any of these documents, regardless of the software they were made in and view these document.

Acceptance Criteria When the user selects a document to open, the system should open the document, regardless of the document extension.

US 2 Creating A Document

User Story As a user, I want to easily create any kind of document, with templates or without.

Acceptance Criteria When the user chooses to create a new document, they should be prompted to create a blank document or to create a document made using a template (derived from a list of templates). The document should be created and immediately open.

US 3 Saving Documents

User Story As a user, I want to be able to store my documents anywhere - on my device, in the cloud, etc. I also want to be able to save any documents as a PDF.

Acceptance Criteria The system should allow the user to save any document on local storage or in the cloud. The system should also allow the user to save any document as a PDF.

US 4 Editing Documents

User Story As a user, I want to be able to easily edit any document. I want to be able to add images, changes font styles, etc. I want to have complete control over how the document looks.

Acceptance Criteria The system should allow the user to edit any part of the document, including text and images.

US 5 Printing Documents

User Story As a user, I want to be able to print any part of my documents. I want to be able to choose which pages get printed, or if the whole document needs to be printed.

Acceptance Criteria The system should allow the user to print a document or a part of a document. The system should allow the user to select which parts of the document to print.

US 6 Fixing Errors

User Story As someone who writes a lot, I want any errors, such as typos, to be detected by the software. I want this software to give me easy options to fix any errors.

Acceptance Criteria The system should detect any spelling or grammatical errors. The corrections to these errors should be presented to the user, who should then be able to accept or decline these corrections.

4 Functional Requirements

FR-1 Opening an Existing Document

- FR-1.1 The system will allow the user to open an existing document found in local/device storage.
- FR-1.2 The system will allow the user to open an existing document found in cloud storage.
- FR-1.3 The system should allow the user to open an existing document regardless of format or file extension.

FR-2 Creating a New Document

- FR-2.1 The system will allow the user to create a blank document.
- FR-2.2 The system will allow the user to create a document using a template design.

FR-3 Editing a Document

- FR-3.1 The system will allow the user to modify existing text in the document.
- FR-3.2 The system will allow the user to add new text to the document.
- FR-3.3 The system will allow the user to remove text from the document.
- FR-3.4 The system will allow the user to format the document (for example: adding italics).
- FR-3.5 The system will allow the user to add images/figures to the document.
- FR-3.6 The system will allow the user to modify images/figures in the document.
- FR-3.7 The system will allow the user to remove images/figures from the document.

FR-4 Saving a Document

- FR-4.1 The system will allow the user to save the document to local/device storage.
- FR-4.2 The system will allow the user to save the document to cloud storage.
- FR-4.3 The system will allow the user to save the document as a PDF to local/device storage.
- FR-4.4 The system will allow the user to save the document as a PDF to cloud storage.

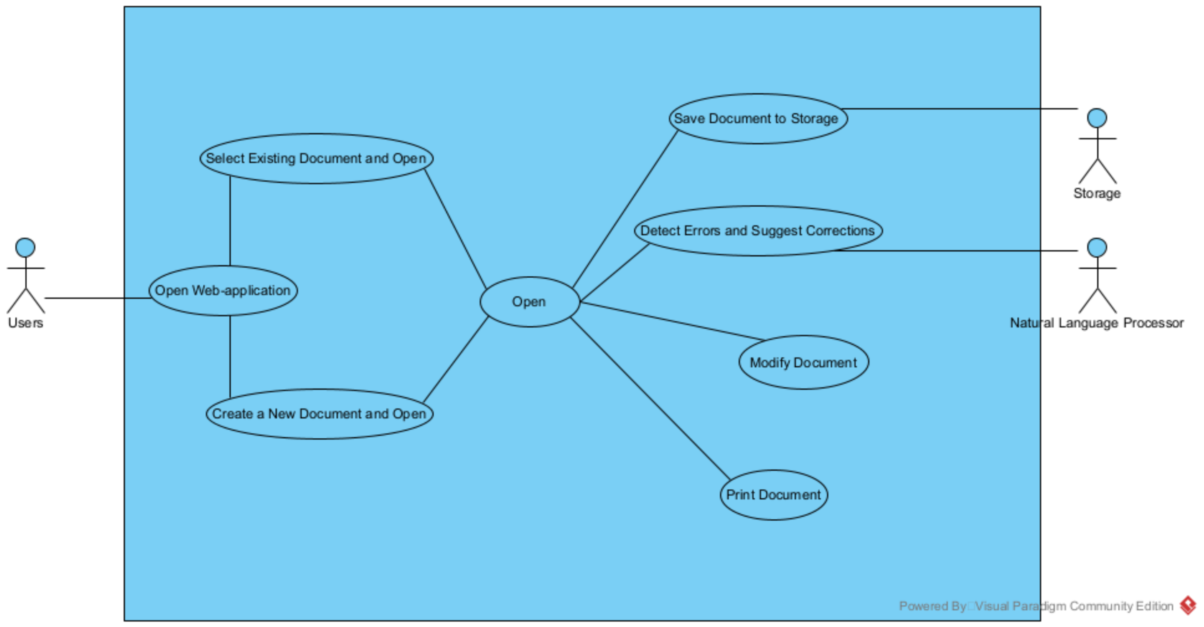
FR-5 Reviewing a Document

- FR-5.1 The system will detect grammatical errors in the document and suggest corrections to the user.
- FR-5.2 The system will detect spelling errors in the document and suggest corrections to the user.

FR-6 Printing a Document

- FR-6.1 The system will allow the user to print the document.

5 Use Case Diagrams



Users
Open Web-application
Select Existing Document and Open
Create a New Document and Open
Open
Save Document to Storage
Detect Errors and Suggest Corrections
Modify Document
Print Document
Storage
Natural Language Processor
Powered By Visual Paradigm Community Edition

Figure 1: High-Level Use Case Diagram

5.1 Subsystem 1 - Accessing Documents

U1 Opening an Existing Document
Allows the user to open an existing document stored either in local storage or in cloud storage.

U2 Creating a New Document
Allows the user to create a new document.

5.2 Subsystem 2 - Saving Documents

U3 Save Document to Local Storage
Allows the user to save a document to local storage (e.g., the document will be saved to the laptop's hard-drive).

U4 Save Document to Cloud Storage
Allows the user to save the document to cloud storage.

U5 Convert Document to PDF and Save to Local Storage
Allows the user to convert the current document to a PDF and save it to local storage.

U6 Convert Document to PDF and Save to Cloud Storage
Allows the user to convert the current document to a PDF and save it to cloud storage.

5.3 Subsystem 3 - Modifying Documents

U7 Edit Text
Allows the user to modify any text in the document.

U8 Edit Images/Figures
Allows the user to modify any images and figures in the document.

U9 Format Document
Allows the user to format the document.

5.4 Subsystem 4 - Printing Documents

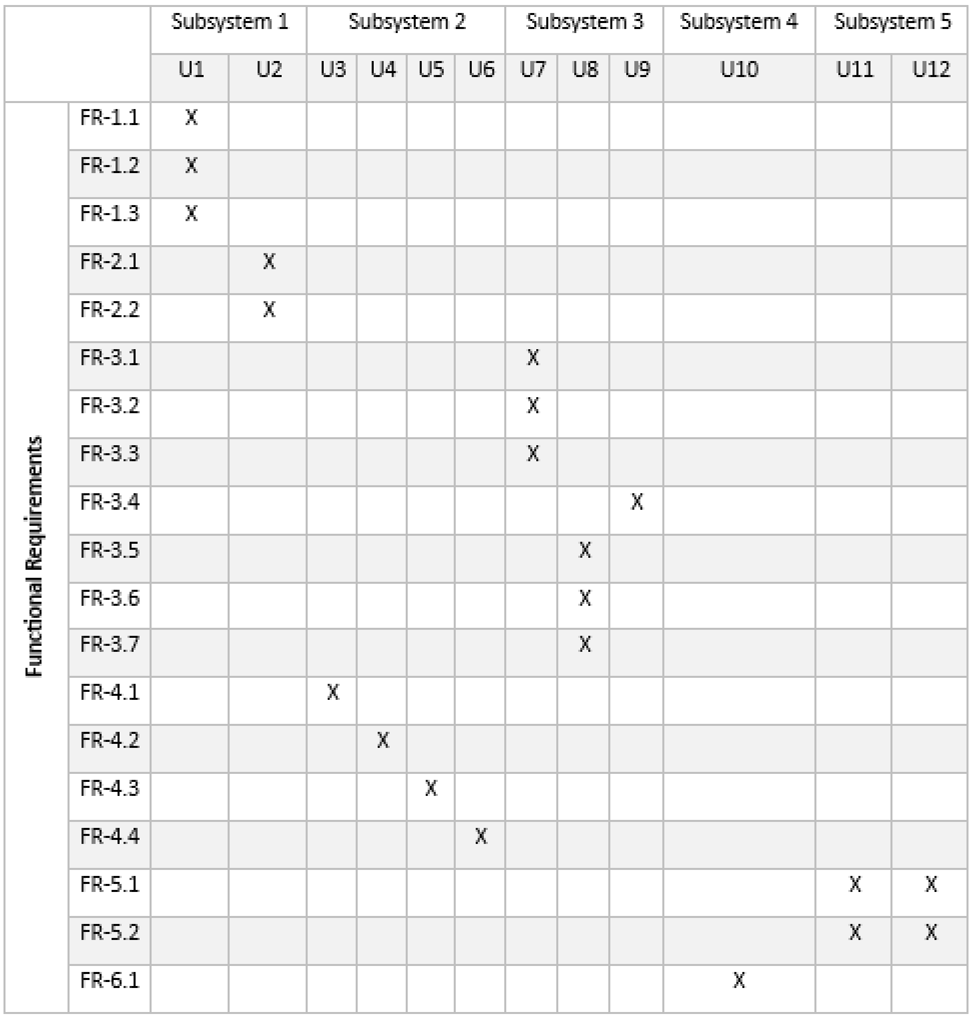
U10 Print Document
Allows the user to print any document.

5.5 Subsystem 5 - Spelling and Grammar Detection

U11 Recommend and Accept Spelling and Grammar Corrections
Allows the user to view spelling and grammar corrections offered by the system and accept these corrections.

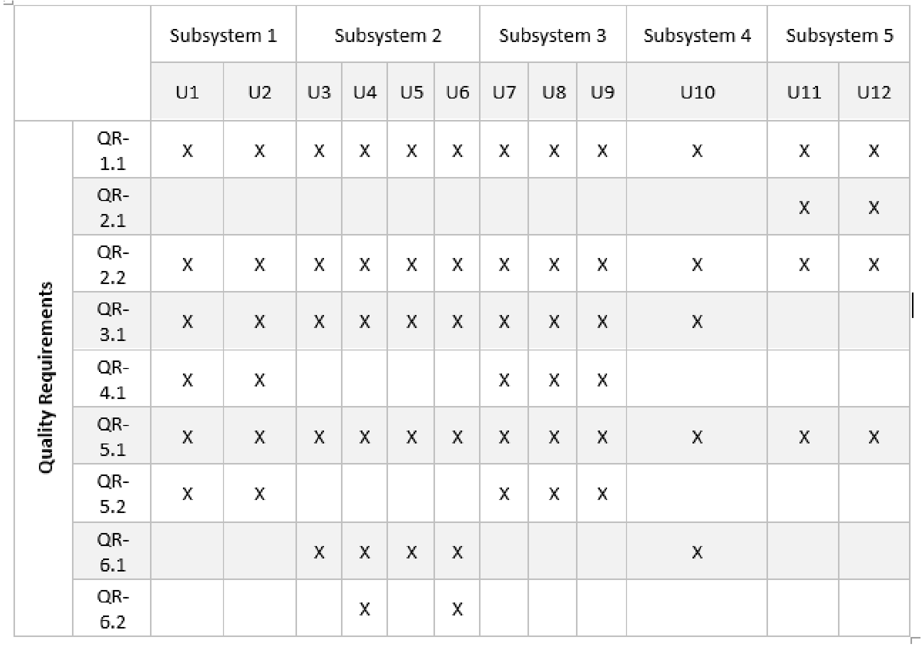
U12 Recommend and Decline Spelling and Grammar Corrections
Allows the user to view spelling and grammar corrections offered by the system, and decline these corrections.

6 Traceability Matrix



		Subsystem 1		Subsystem 2				Subsystem 3			Subsystem 4	Subsystem 5	
		U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12
Functional Requirements	FR-1.1	X											
	FR-1.2	X											
	FR-1.3	X											
	FR-2.1		X										
	FR-2.2		X										
	FR-3.1							X					
	FR-3.2							X					
	FR-3.3							X					
	FR-3.4									X			
	FR-3.5								X				
	FR-3.6								X				
	FR-3.7								X				
	FR-4.1			X									
	FR-4.2				X								
	FR-4.3					X							
	FR-4.4						X						
	FR-5.1											X	X
	FR-5.2											X	X
	FR-6.1										X		

Figure 2: Functional Requirements Traceability Matrix



		Subsystem 1		Subsystem 2				Subsystem 3			Subsystem 4	Subsystem 5	
		U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12
Quality Requirements	QR-1.1	X	X	X	X	X	X	X	X	X	X	X	X
	QR-2.1											X	X
	QR-2.2	X	X	X	X	X	X	X	X	X	X	X	X
	QR-3.1	X	X	X	X	X	X	X	X	X	X		
	QR-4.1	X	X					X	X	X			
	QR-5.1	X	X	X	X	X	X	X	X	X	X	X	X
	QR-5.2	X	X					X	X	X			
	QR-6.1			X	X	X	X				X		
	QR-6.2				X		X						

Figure 3: Quality Requirements Traceability Matrix

7 Class/Component Domain Modeled Diagram

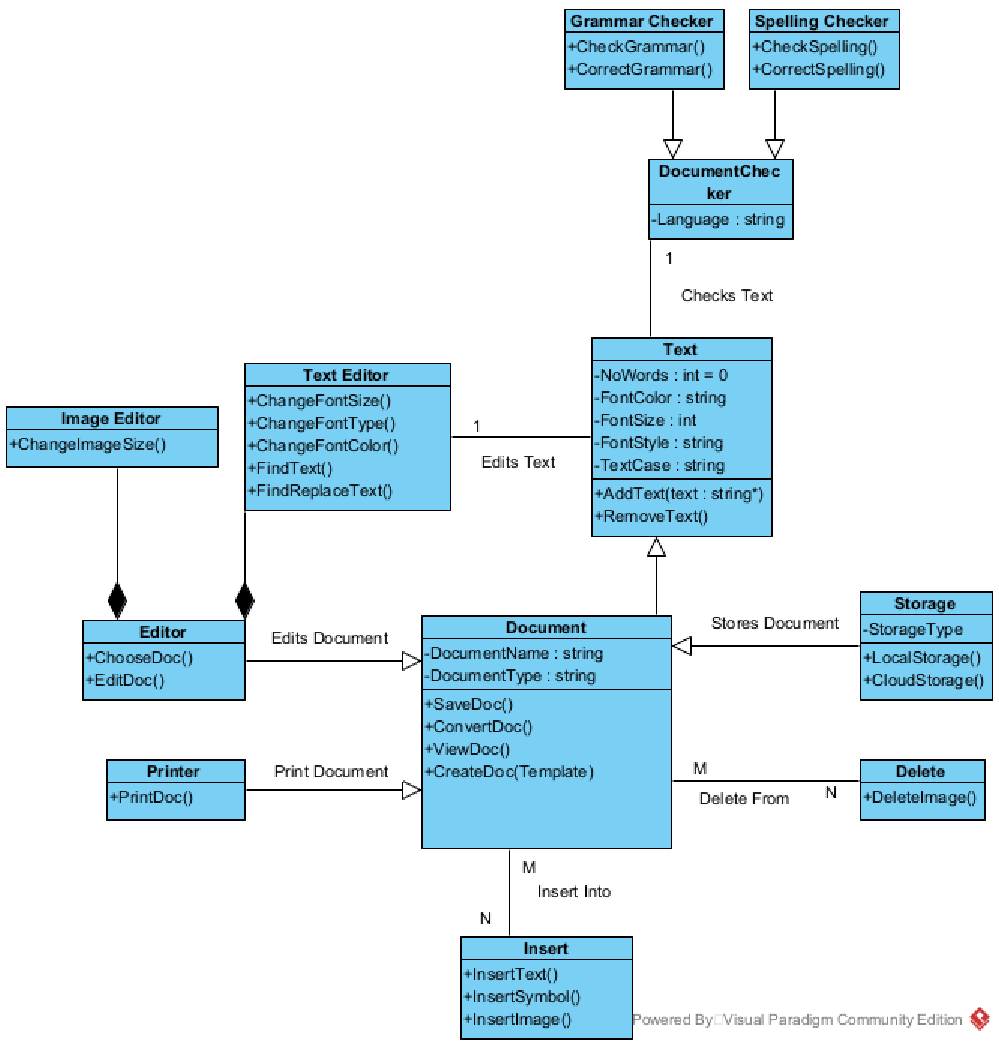


Figure 4: Domain Model

The Document component is the main component. It is primarily made of text. The Document component uses the Insert, Delete, Print and Storage components to achieve the inserting, deleting, printing and storing functionalities respectively.

The Insert component adds text, symbols and images to the document. The Delete component removes images from the document. The Storage component stores the type of storage that the document uses, and stores the document in either the cloud or on local storage. The Print component takes the document and sends the document to a connected printer to be printed.

The Text component is dependent on the Document component. The Text

Editor component is used by the Text component to change the characteristics (e.g., font style) of the text.

The DocumentChecker Component has two child component - the GrammarChecker and the SpellingChecker components. The DocumentChecker uses the Text component to as input for checking and any corrections.

The Editor component is used by the Document component. The Editor component has two child components - ImageEditor and TextEditor.

8 Non-Functional Requirements

8.1 Quality Requirements

QR-1 Usability

- QR-1.1 The system should be easy to use regardless of the user's prior experience with office suite software.
- QR-1.2 The system should have an effectiveness percentage of 75%, as calculated by the formula:

  $NumberofTaskscompletedsuccessfullydividedbytheTotalNumberofTasksundertakenx100$

  [12]

QR-2 Maintainability

- QR-2.1 The system should comply with modern best practices.
- QR-2.2 The system should be able to easily be updated to ensure that it works with each new version of all major web browsers 95% of the time.
- QR-2.3 The system should be easily maintainable, such that adding new functionality should not damage existing functionality 95% of the time.

QR-3 Interoperability

- QR-3.1 The system should be able to interact with document, spreadsheet and presentation files regardless of the format or file extensions of the documents.
- QR-3.2 The system should be able to retrieve and save a document to any storage location either on cloud storage or local device storage.
- QR-3.3 The system should be able to interact with any document format and storage location 90% of the time.

QR-4 Performance

- QR-4.1 The system should load and display the interface upon initialization within 5 seconds.
- QR-4.2 The system should be able to retrieve and open any document within 30 seconds.
- QR-4.3 The system should be able to save any document to either local or cloud storage within 60 seconds.

QR-5 Portability

- QR-5.1 The system should be fully functional on all major web browsers 95% of the time.
- QR-5.2 The system should be consistent in layout and performance regardless of browser or screen size 95% of the time.

QR-6 Security

QR-6.1 The system should conform to web-security best practices 99% of the time.

QR-6.2 The system should be able to work alongside any cloud storage security requirements 99% of the time.

9 System Architecture

9.1 Architectural Patterns

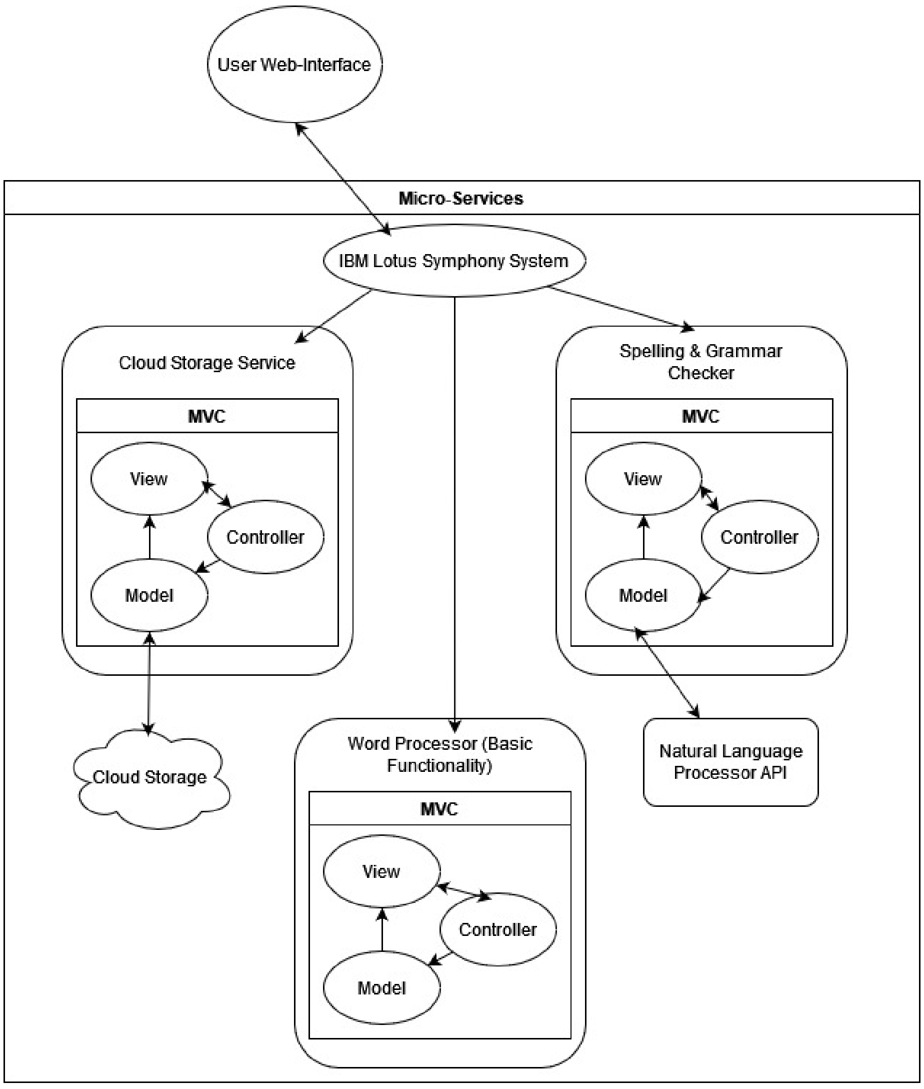


Figure 5: Architectural Diagram - Microservices with MVC

The system uses two architectural patterns: Microservices and Model-View-Controller (MVC).

The Microservices architecture is an architectural style where the system is structured as a collection of independent components or services. Each component performs a single function (e.g., provide the connection to save documents to cloud storage in the case of IBM Lotus Symphony) [13]. An important characteristic of Microservices is that each component is completely independent, and as such can be developed, and deployed without affecting any of the other components within the system [13].

Some of the benefits of Microservices are as follows:

Resilience Due to the independent nature of each component, should a single component fail, the remaining components will not be affected. Thus increasing the system's resilience to failure [13].

Deployability Each component can be independently deployed. This also increases the system's modifiability, as each component can be modified and deployed independently [13], [14].

The Model-View-Controller (MVC) architecture is an architectural pattern that is well-known for its usability [14]. The MVC architecture is comprised of three layers. The Model layer represents the business logic [15]. Additionally, the Model layer maintains the data, this is connected to components such as databases [16].

The View layer is the presentation layer [15]. It generates the user interface [16].

The Controller layer manages the flow between the View and Model layers [15]. It enables to interaction between the View and the Model [16].

Some of the benefits of the MVC architecture are as follows:

Reusability The components are reusable [16].

Maintainability The separation of the components means that the system is easy to maintain [16].

9.2 Architectural Styles

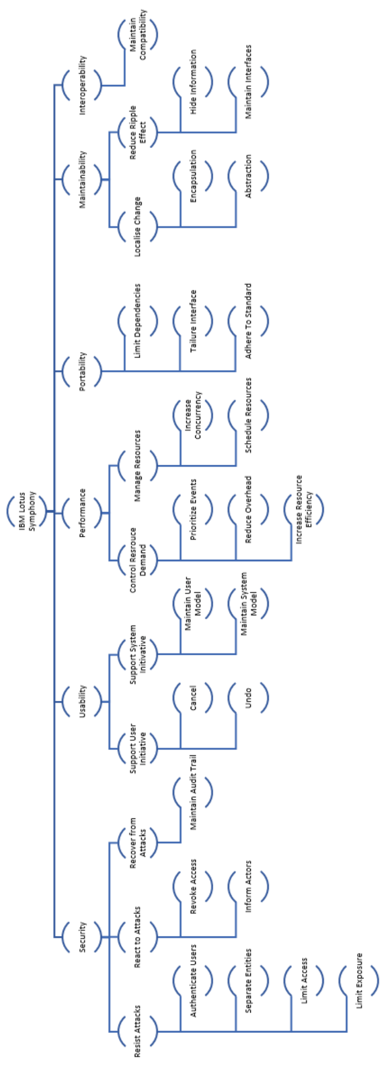


Figure 6: Architectural Tactics

Tactics include Manage Resources (Performance), Support User Initiative (Usability), Resist Attacks (Security), Reduce Dependencies (Portability), Localise Change (Maintainability) and Maintain Compatibility (Interoperability).

9.3 Architectural Constraints

C1 Open-Source

- The system is limited to being completely open-source, this includes all code produced and technologies used.

C2 Limited Size

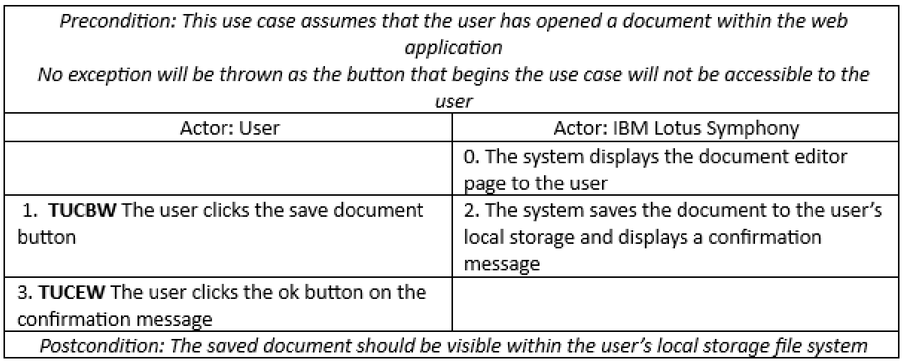
- The system is designed to operate on web-browsers, thus the size of the system on the client-side must be limited to avoid the system lagging.

C3 Limited Resources

- Due to the limited resources, such as time, equipment available, the limitations of open source technologies, etc., the system was not able to be implemented to its full capability. Thus, only parts of the system that offered key functionality was developed, and some parts outsourced to existing components developed by others.

9.4 Actor-System Interaction Models

9.4.1 Subsystem 2 - Use Case 3



Precondition: This use case assumes that the user has opened a document within the web application
No exception will be thrown as the button that begins the use case will not be accessible to the user

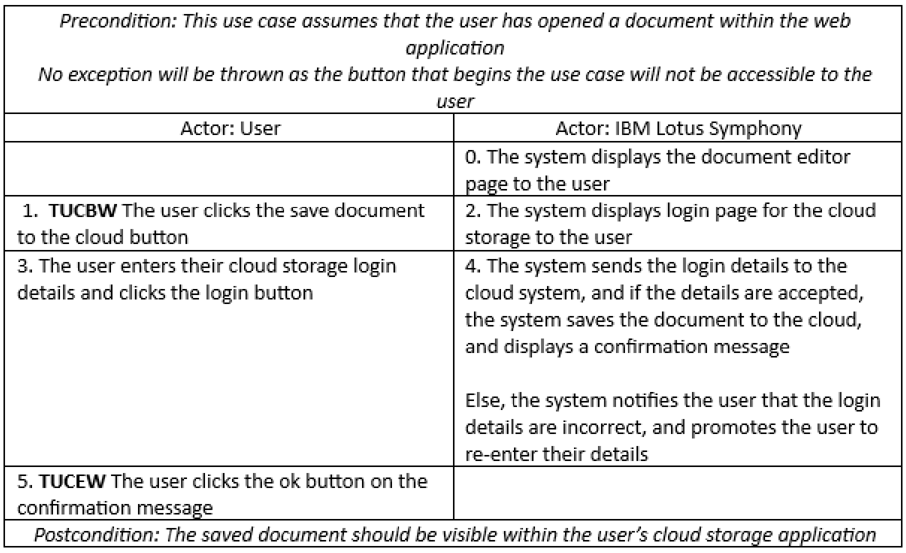
Actor: User	Actor: IBM Lotus Symphony
	0. The system displays the document editor page to the user
1. **TUCBW** The user clicks the save document button	2. The system saves the document to the user's local storage and displays a confirmation message
3. **TUCEW** The user clicks the ok button on the confirmation message	

Postcondition: The saved document should be visible within the user's local storage file system

Figure 7: Actor-System Interaction Model of U3

Use Case 3 allows the user to save a document to local storage.

9.4.2 Subsystem 2 - Use Case 4



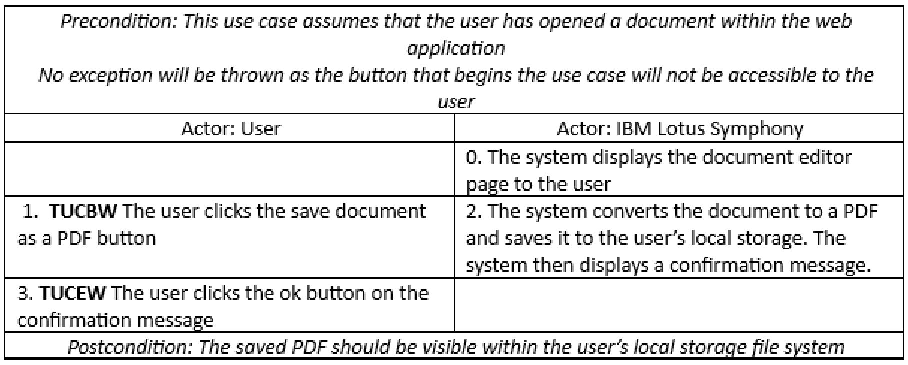
Precondition: This use case assumes that the user has opened a document within the web application
No exception will be thrown as the button that begins the use case will not be accessible to the user

Actor: User	Actor: IBM Lotus Symphony
	0. The system displays the document editor page to the user
1. **TUCBW** The user clicks the save document to the cloud button	2. The system displays login page for the cloud storage to the user
3. The user enters their cloud storage login details and clicks the login button	4. The system sends the login details to the cloud system, and if the details are accepted, the system saves the document to the cloud, and displays a confirmation message Else, the system notifies the user that the login details are incorrect, and promotes the user to re-enter their details
5. **TUCEW** The user clicks the ok button on the confirmation message	

Postcondition: The saved document should be visible within the user's cloud storage application

Figure 8: Actor-System Interaction Model of U4

Use Case 4 allows the user to save a document to cloud storage.



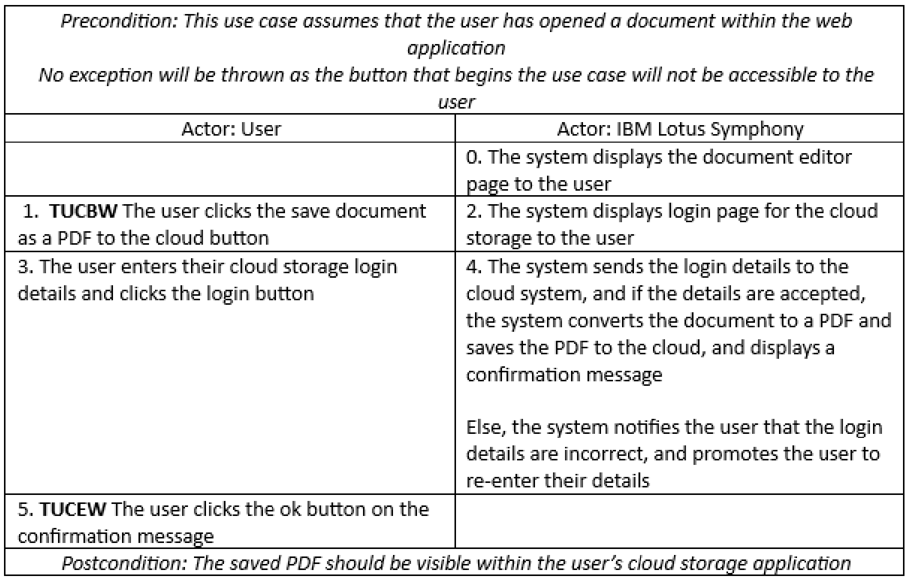
Precondition: This use case assumes that the user has opened a document within the web application *No exception will be thrown as the button that begins the use case will not be accessible to the user*	
Actor: User	Actor: IBM Lotus Symphony
	0. The system displays the document editor page to the user
1. **TUCBW** The user clicks the save document as a PDF button	2. The system converts the document to a PDF and saves it to the user's local storage. The system then displays a confirmation message.
3. **TUCEW** The user clicks the ok button on the confirmation message	
Postcondition: The saved PDF should be visible within the user's local storage file system	

Figure 9: Actor-System Interaction Model of U5

9.4.3 Subsystem 2 - Use Case 5

Use Case 5 allows the user to convert a document to PDF and save the PDF to local storage.

9.4.4 Subsystem 2 - Use Case 6



Precondition: This use case assumes that the user has opened a document within the web application *No exception will be thrown as the button that begins the use case will not be accessible to the user*	
Actor: User	Actor: IBM Lotus Symphony
	0. The system displays the document editor page to the user
1. **TUCBW** The user clicks the save document as a PDF to the cloud button	2. The system displays login page for the cloud storage to the user
3. The user enters their cloud storage login details and clicks the login button	4. The system sends the login details to the cloud system, and if the details are accepted, the system converts the document to a PDF and saves the PDF to the cloud, and displays a confirmation message Else, the system notifies the user that the login details are incorrect, and promotes the user to re-enter their details
5. **TUCEW** The user clicks the ok button on the confirmation message	
Postcondition: The saved PDF should be visible within the user's cloud storage application	

Figure 10: Actor-System Interaction Model of U6

Use Case 6 allows the user to convert a document to PDF and save the PDF to cloud storage.

9.5 Technology Choices and Decisions

9.5.1 Angular

What is it

- Angular is a Typescript development platform [17].
- It is a component-based framework designed for building web applications [17].

Pros

- Angular applications are faster to develop due the component-based architecture [18].
- Comes with a built-in Command-Line Interface (CLI) to allow for quick generation of components [18].
- Designed to support mobile web and native applications [18].
- Maintainability. Due to the components being decoupled from each other, components can be easily maintained [18].
- Unit-test friendly [18].
- Allows for visually appealing and easy to use User Interfaces [18].

Cons

- Complex and verbose system [18]. A single component could be comprised of multiple files, and development is highly repetitive [18].
- Steep learning curve [18]. Comprised of large modules, dependencies, etc. [18].

Reasoning

- Angular automatically makes the application follow the MVC architecture, by dividing the web-app into three parts [18].
- Supports mobile web applications, which is what the system is being designed for [18].
- Easily maintainable structure and code layout [18].

Alternatives

- React
- Vue

9.5.2 GitHub Project Board

What is it

- A GitHub Project board is an adaptable and flexible tool used for planning and tracking work, particular work related to software development [19].
- Users can manage and track tasks with task lists, labels and milestones [20].

Pros

- Can add unlimited custom fields to tasks to allow for full control over the project board [20].
- Can convert tasks into a GitHub issue [20].
- Automated workflow (e.g., when an issue is finished, the task associated with it is automatically moved to the Complete column of the project board) [20].
- No limit to number of users working on the project board [20].
- Free to use [20].

Cons

- Cannot comment on a task unless it has been converted to a Github issue [20].
- No referencing between tasks [20].
- Cannot import tasks from other project management tools [20].

Reasoning

- Easy to use, due to prior experience with it
- Can track tasks related to programming easily, as it is connected to the project code repository.
- Can access on any device with an internet connection.
- Easy to customize project board
- Free to use

Alternatives

- Asana
- Microsoft Excel

10 Implementation

IBM Lotus Symphony will be implemented as a service. It will be implemented as an online office suite that will not require any user to have an account.

10.1 Deployment Model

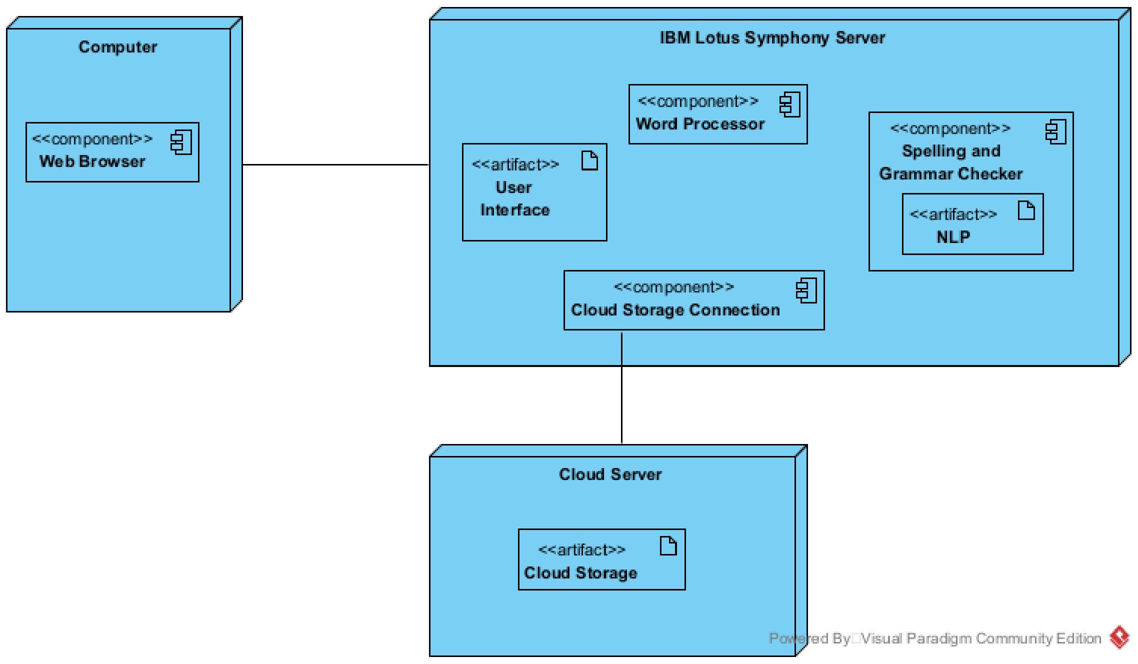

Computer
<<component>> Web Browser
IBM Lotus Symphony Server
<<component>> Word Processor
<<artifact>> User Interface
<<component>> Spelling and Grammar Checker
<<artifact>> NLP
<<component>> Cloud Storage Connection
Cloud Server
<<artifact>> Cloud Storage
Powered By Visual Paradigm Community Edition

Figure 11: Deployment Diagram

The Microservices architecture sits within the IBM Lotus Symphony Server. The user accesses the system through the User Interface on their web-browsers. Each component within the Microservices architecture contains the MVC architecture, not shown within the diagram.

10.2 Technical Manual

Technical Manual

10.3 User Manual

User Manual As part of the User Manual, a word-processor developed by Syncfusion was used to show functionality [21].

References

[1] Softonic, *Ibm lotus symphony for mac*, https://ibm-lotus-symphony-mac.en.softonic.com/mac?ex=RAMP-1768.1, (Accessed on: 22/03/2024).

[2] IBM, "An overview of ibm lotus symphony," IBM, One Rogers Street, Cambridge, MA 02142, USA, White Paper, Apr. 2008. [Online]. Available: https://public.dhe.ibm.com/software/in/lotus/IBM_productivity_tools_overview.pdf.

[3] K. Fiveash, *Ibm hopes open office is symphony to your key-tapping fingers*, https://www.theregister.com/2007/09/19/ibm_office_symphony_open_source/, (Accessed: 11-03-2024).

[4] IBM, "Ibm lotus symphony software," IBM, One Rogers Street, Cambridge, MA 02142, USA, Tech. Rep., Jun. 2009. [Online]. Available: https://public.dhe.ibm.com/software/in/lotus/products/symphony/LOF14003USEN.pdf.

[5] Softonic, *Ibm lotus symphony for windows*, https://ibm-lotus-symphony.en.softonic.com/, (Accessed: 12-03-2024).

[6] B. Popa, *Ibm lotus symphony*, https://www.softpedia.com/get/Office-tools/Office-suites/IBM-Lotus-Symphony.shtml, (Accessed: 12-03-2024).

[7] M. Levitt, "Ibm lotus symphony: A step-by-step approach to finding open suite spots in your organisation," IDC, 5 Speen Street, Famingham, MA 01701 USA, White Paper, Oct. 2008. [Online]. Available: https://public.dhe.ibm.com/software/in/lotus/products/symphony/IDC_Lotus_Symphony_Nov2008.pdf.

[8] M. Garbett, "Smarter collaboration for strategic impact: Optimising collaboration cost structures," IBM, Tech. Rep., 2010. [Online]. Available: https://public.dhe.ibm.com/software/id/lcty/Smarter_Collaboration_Strategic_Impact.pdf.

[9] S. McLeish, *Microsoft office still owns the desktop, future of staroffice unclear*, https://www.zdnet.com/article/microsoft-office-still-owns-the-desktop-future-of-staroffice-unclear/, (Accessed: 12-03-2024).

[10] G. Clarke, *Ibm calls time on symphony openoffice fork*, https://www.theregister.com/2012/01/30/ibm_openoffice_apache/, (Accessed: 12-03-2024).

[11] IBM, "Why ibm lotus symphony? why free?" IBM, One Rogers Street, Cambridge, MA 02142, USA, Tech. Rep., Jul. 2008. [Online]. Available: https://public.dhe.ibm.com/software/my/collaboration/downloads/LOB14006-USEN-00_US_Busbrochure.pdf.

[12] J. Mifsud, *Usability metrics - a guide to quantify the usability of any system*, https://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/, (Accessed on: 19/05/2024).

[13] AWS, *Microservices*, https://aws.amazon.com/microservices/, (Accessed on: 22/05/2024).

[14] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice, 4th Edition*. Carnegie Mellon University: Pearson Addison-Wesley, 2022, (Accessed on: 22/05/2024).

[15] JavaTPoint, *Mvc architecture in java*, `https://www.javatpoint.com/mvc-architecture-in-java`, (Accessed on: 22/05/2024).

[16] Z. Svirca, *Everything you need to know about mvc architecture*, `https://towardsdatascience.com/everything-you-need-to-know-about-mvc-architecture-3c827930b4c1`, (Accessed on: 22/05/2024).

[17] Angular, *What is angular?* `https://angular.io/guide/what-is-angular`, (Accessed on: 22/05/2024).

[18] Altexsoft, *The good and bad of angular development*, `https://www.altexsoft.com/blog/the-good-and-the-bad-of-angular-development/`, (Accessed on: 22/05/2024).

[19] GitHub, *About projects*, `https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects`, (Accessed on: 22/05/2024).

[20] D. B. Tips, *Github projects review*, `https://deluxeblogtips.com/github-projects-review/`, (Accessed on: 22/05/2024).

[21] Syncfusion, *Blazor word processor - a wysiwyg document editor component*, `https://www.syncfusion.com/blazor-components/blazor-word-processor`, (Accessed on: 26/05/2024).