Software Requirements Specification

for

OpenRead Web-Application

**Version 4.1**

**Prepared by**

**Anthony Gilrandy Theo- 2201734814**

**Dewi Puspita Tanurezal - 2201740546**

**Devita - 2201755131**

**Steven - 2201852132**

**Bina Nusantara University**

**2020**

Table of Contents

Table of Contents 2

Revision History 3

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.2.1 Visitor Use Case 3

2.2.2 Reader Use Case 5

2.2.3 Writer Use Case 8

2.3 User Classes and Characteristics 9

2.4 Operating Environment 9

2.5 User Documentations 9

2.6 Non-function requirements 10

3. External Interface Requirements 11

3.1 User Interfaces 11

3.2 Hardware Interfaces 17

3.3 Software Interfaces 17

3.4 Communications Interfaces 18

4. System Features 18

4.1 Register Account 18

4.2 Login 19

4.3 Search Story 19

4.4 Write Comments 19

4.5 Update Comments 20

4.6 Rate Story 20

4.7 Edit Profile 21

4.8 Change Password 21

4.9 Write Story 22

4.10 Edit Story 22

5. Other Nonfunctional Requirements 23

5.1 Performance Requirements 23

5.2 Safety Requirements 23

5.3 Security Requirements 23

5.4 Software Quality Attributes 24

5.5 Business Rules 24

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Steven | 13-10-2020 | Added Product Function (2.2) | 1.1 |
| Steven | 14-10-2020 | Added System Features (Chapter 4) | 1.2 |
| Steven | 27-11-2020 | Revised System Features (4.10 & 4.11)   * Takedown Story * Ban User   Added New System Features (4.1 & 4.2)   * Register Account * Login | 2.0 |
| Steven | 29-11-2020 | Revised User Interfaces (3.1)   * Login Account * Side Menu   Added Nonfunctional Requirements (Chapter 5) | 3.0 |
| Steven | 11-12-2020 | Revised ERD   * Added Entity Rating * Added Entity Story\_Genre * Added Entity Genre * Minor changes to attribute’s name | 3.1 |
| Steven | 13-12-2020 | Revised ERD   * Removed Entity Complaint * Removed Entity Admin   Removed Admin Management  Removed Admin Use Case  Removed System Feature related to admin (4.9, 4.10, 4.11) | 4.0 |
| Steven | 17-12-2020 | Revised ERD  Revise Use Case Diagram  Revised Data flow chart context diagram  Revised User Interface Design  Added Use Case Rate Story, Edit Profile, and Change Password  Added content for External Interface Requirements | 4.1 |

# Introduction

## Purpose

The purpose of this document is to build a web-application for people to share and read stories.

## Document Conventions

|  |  |
| --- | --- |
| *DB* | *Database* |
| *ER* | *Entity Relationship* |

## Intended Audience and Reading Suggestions

This project is a prototype for storytelling website and it is restricted within the university premises. This project are made under the guidance of university professors. This project is useful for developer who want to make website storytelling and as well as the user of the website.

## Product Scope

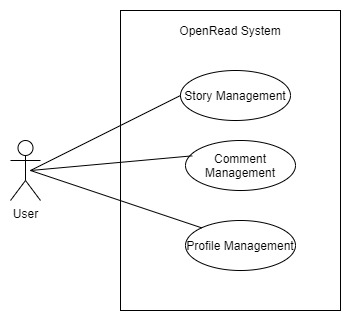
The purpose of OpenRead is that we provide a platform for people to read and share stories. We will have a database server that support to store writer’s stories as well as reader’s thoughts. Above all, We aim to create a community around stories for both amateur and established writers.

## References

* IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.
* Roger S. Pressman. (2015). Software engineering: a practitioners approach. 08. McGraw - Hill Higher Education. NY.
* Ian Sommerville. (2016). Software Engineering TENTH edition. Pearson
* https://www.cse.msu.edu/~chengb/RE-491/Papers/SRS-BECS-2007.pdf

# Overall Description

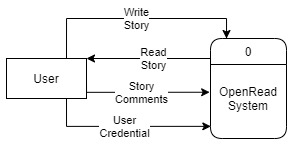
## Product Perspective



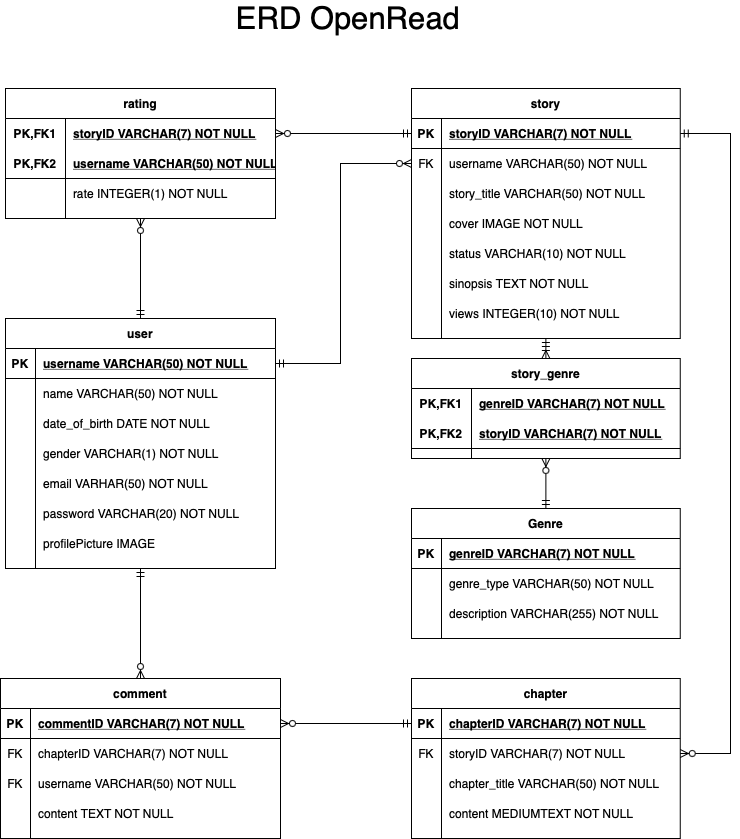
OpenRead is a platform as a website for amateur or professional writers to create stories that can be read by readers without the need to go through a publisher. Readers can search stories by genre and title to find whatever story they want to read. Writers can create stories, determine its genre and divide it into several chapters.

## Product Functions

Data Flow Context Diagram:



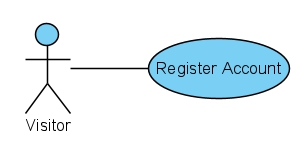
Entity-Relationship model :



### Visitor Use Case

**Use case : Register Account**

**Diagram:**



**Brief Description:**

The user will register their information and create their account. This action can be done on register page.

**Initial Step-by-Step Description:**

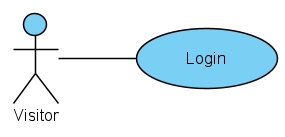
Before this use case can be initiated, the user has already accessed the register page.

1. The visitor inputs their username, email, password, and date of birth.
2. The visitor presses Sign Up button.
3. If the register is successful, the visitor will be logged in and redirected to the home page.
4. If one of the inputs is failed to be validated, the visitor must correct one of the fields before the account can be created.

**Xref:** Section 4.1, Register Account

**Use case : Login**

**Diagram:**



**Brief Description:**

The visitor can login based on the user credentials that he/she has registered. After he/she has login, his/her status will change to User.

**Initial Step-By-Step Description**

Before this use case can be initiated, the reader has already register and his/her account details is recorded in the database.

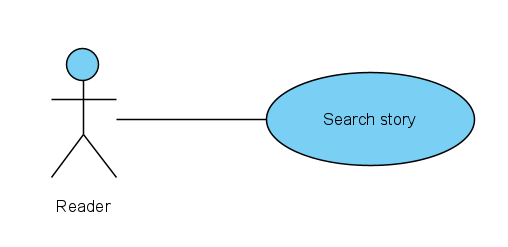
1. The visitor enters his/her username/emails and passwords
2. The system will validate whether the credentials match.
3. The visitor logins successfully.
4. The status of the visitor change to User.

**Xref:** Section 4.2, Login

### Reader Use Case

**Use case: Search story**

**Diagram:**

****

**Brief description:**

The reader searches a story they want to read. They can search by the title of the story or by the author name.

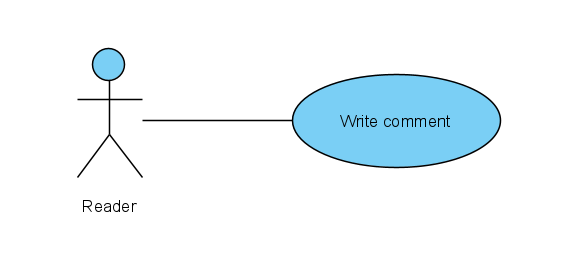
**Initial step-by-step description:**

Before this use case initiated, the reader has accessed the home page.

1. The reader clicks the search button at the top of the website home page.
2. The reader writes down the name of the story or the author name specifically.
3. The system will find the story or author name related to what the reader search for.
4. If there are any story or author name related, the system will show the story. If there are not, the system will write “The story or author name not found” and show another story available.

**Xref:** Section 4.3, Search Story

**Use Case: Write Comments**

**Diagram:**

**Brief Description:**

The reader can comment his/her thoughts from other people’s stories, this action can be done on the comment section.

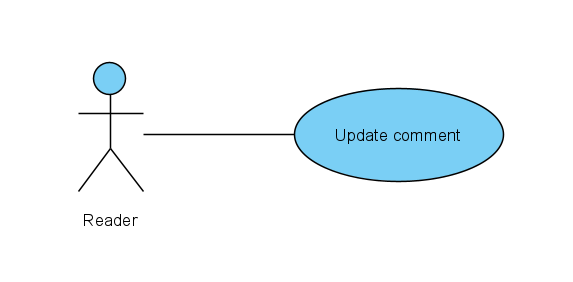
**Initial Step-By-Step Description:**

Before this use case can be initiated, the reader has already accessed a story page.

1. The reader selects add button near comment section.
2. The reader types his/her comments.
3. The reader posts.
4. The system asks for the reader to review his/her comments to make sure it doesn’t contain any inappropriate content.
5. The reader chooses to continue/review.
6. The system reads the comment and posts it at the comment section.

**Xref:** Section 4.4, Write Comments

**Use Case: Update Comments**

**Diagram:**

**Brief Description:**

The reader can edit his/her comment that he/she posted; this action can be done on the comment section.

**Initial Step-By-Step Description:**

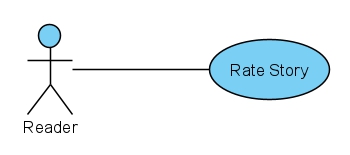
Before this use case can be initiated, the reader has already access a story page and posted a comment at the exact same story page.

1. The reader selects the pencil button of the comment he/she wishes to update.
2. The reader updates his/her comments.
3. The reader posts.
4. The system asks for the reader to review his/her comments to make sure it doesn’t contain any inappropriate content.
5. The reader chooses to continue/review.
6. The system reads the comment and renews it at the comment section.

**Xref:** Section 4.5, Update Comment

**Use Case : Rate Story**

**Diagram :**

****

**Brief Description :**

Reader can rate story from 1 to 5. This action can be done on the page where user can select the chapter as well as rate story.

**Initial Step-By-Step Description**

Before this use case can be initiated, the reader has already logged in his/her account.

1. User clicks on the amount of stars that he/she wants to give. 1 star mean the rating will be 1. Maximum will be 5 stars.
2. User clicks send.
3. The rating will be recorded to the database.
4. The corresponding story’s rating will be changed according to the new rating.

**Xref:** Section 4.6, Rate Story

**Use case: Edit Profile**

**Diagram:**

**Diagram

Description automatically generated**

**Brief description:**

The user opens their own profile page and edits their information. After the user is done with editing their information, they can either cancel or save their changes.

**Initial step-by-step description:**

Before this use case initiated, the reader has logged in and accessed the user’s own profile page.

1. The user clicks the edit profile button.
2. The user changes their information as necessary.
3. The user clicks the Save changes button.
4. If user changes their username or email, the system will check whether the new username or new email has already been used.
5. If the changes are successfully saved, the user will be redirected back to their own profile page.

**Use Case: Change Password**

**Diagram:**

**Diagram

Description automatically generated**

**Brief Description:**

The user opens their own profile page and change the password. After the user is done with changing their password, they can either cancel or save their changes.

**Initial Step-By-Step Description:**

Before this use case initiated, the reader has logged in and accessed the user’s own profile page.

1. The user clicks the change password button.
2. The user fills their current password (old password), desired new password, and re-type the new password.
3. The user clicks Save button.
4. The system will do several validations:
   1. The old password input must be equal to user’s current password.
   2. The confirm password input must be equal to the new password input.
   3. The new password input must not be equal to user’s current password.
5. If the system successfully validates user’s input, the system will hash the new password and updates the new password in the database.
6. The system redirects user to their own profile page.

### Writer Use Case

**Use Case: Write Story**

**Diagram**

Diagram

Description automatically generated

**Brief Description:**

The writer will create story they want to write. This action can be done on writer story page.

**Initial Step-by-Step Description:**

Before this use case can be initiated, the writer has already accessed the writer story page.

1. The writer selects the create story button.
2. The writer enters story title and genre.
3. After entering title and genre, system automatically adds one new chapter to the story.
4. Writer enters title for the chapter and its content.
5. The writer reviews the story title, chosen genre, chapter and its content.
6. The writer chooses to publish the story.
7. The system publishes to story on to the website.

**Xref:** Section 4.7, Write Story

**Use Case: Edit Story**

**Diagram**

Diagram

Description automatically generated

**Brief Description:**

The writer will edit story they have created. This action can be done on writer story page.

**Initial Step-by-Step Description:**

Before this use case can be initiated, the writer has already accessed the writer story page and has created at least one story.

1. The writer selects the edit story on the story he/she wants to edit.
2. The writer updates title or genre of the story.
3. The writer chooses a chapter he/she wants to edit.
4. The writer updates title or content of the chapter.
5. The writer reviews the story title, chosen genre, chapter and its content.
6. The writer chooses to accept the story’s edit.
7. The system republishes the story on to the website.

**Xref:** Section 4.8, Edit Story

## User Classes and Characteristics

There are 2 basic users – User

* User is both the reader and the writer. They are allowed to read stories as well as uploading stories for people to read. The user should be able to do the following functions:

1. Upload stories
2. Read stories
3. Comment on stories

## Operating Environment

The software will operate with all modern operating systems that has access to a browser (google chrome, firefox, safari, internet explorer), while the server will be operating under the Linux operating systems.

## User Documentation

For user documentation and information, head to section 4 (System Features) to understand further about the system. There will also be a user manual attached.

## Non-functional requirements

* The website must be compatible with google chrome, firefox, and internet explorer.
* The database’s query will run on the database administrator’s PC and will contain an SQL database. SQL Server Management Studio is already installed on the PC and it has a Windows operating system as well.

# External Interface Requirements

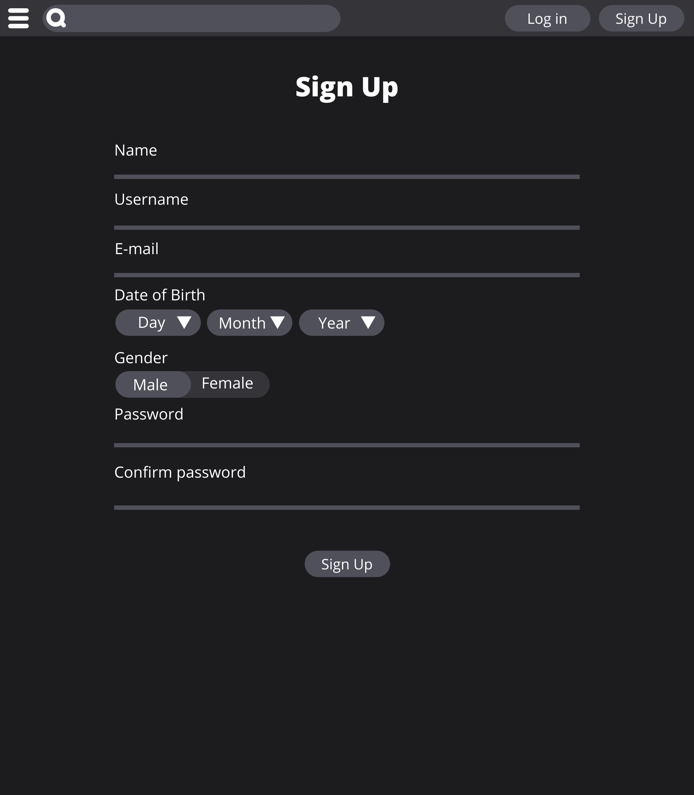
## User Interfaces

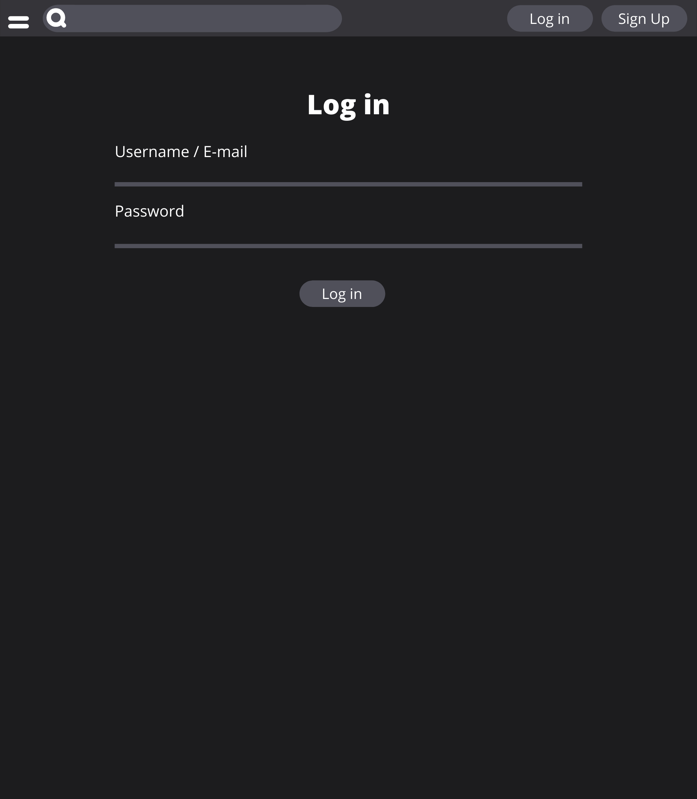
Home Side Menu (User)

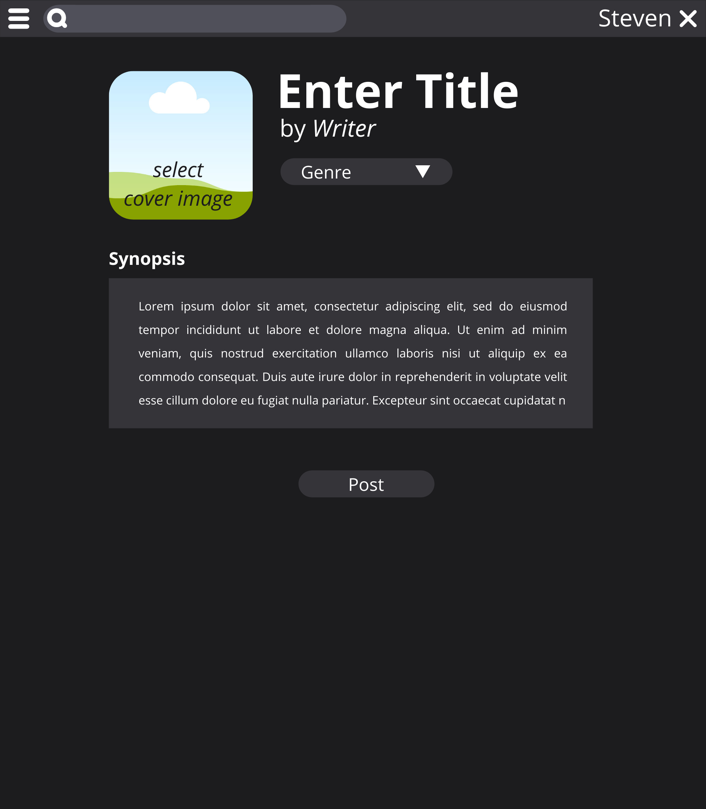


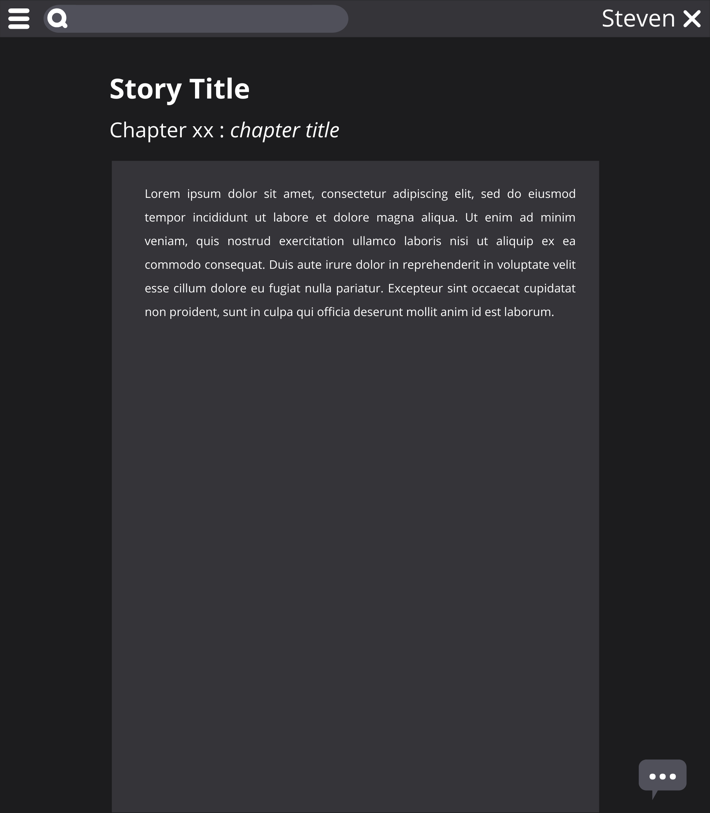


Sign Up Log In

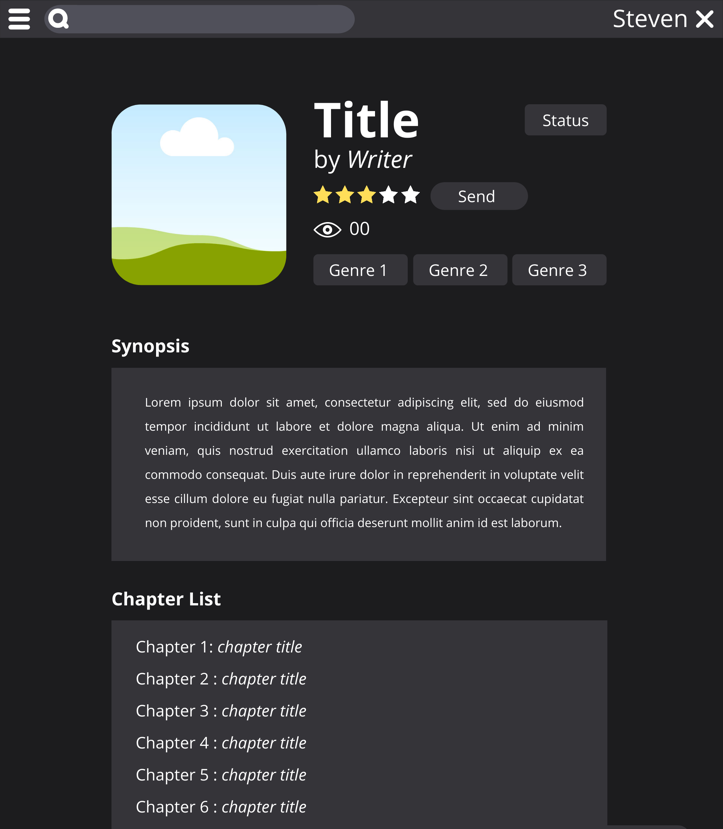
**

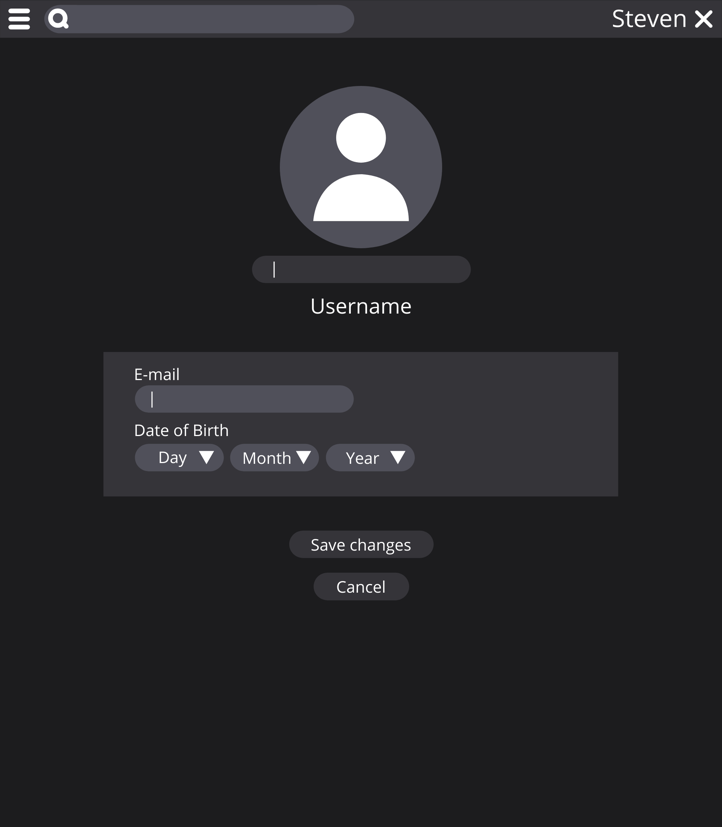
**

** Read Story Write Story

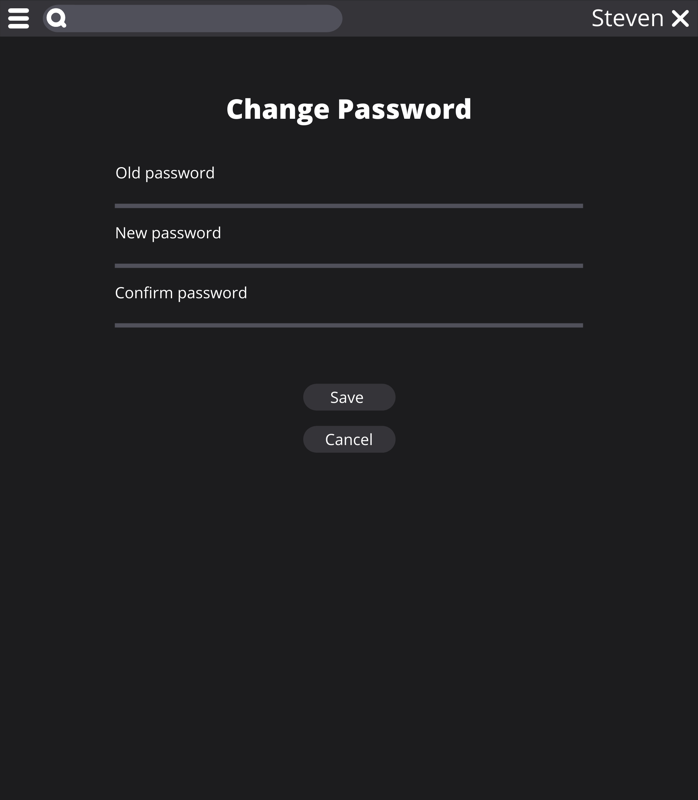
**

Change Profile Choose Chapter from Story

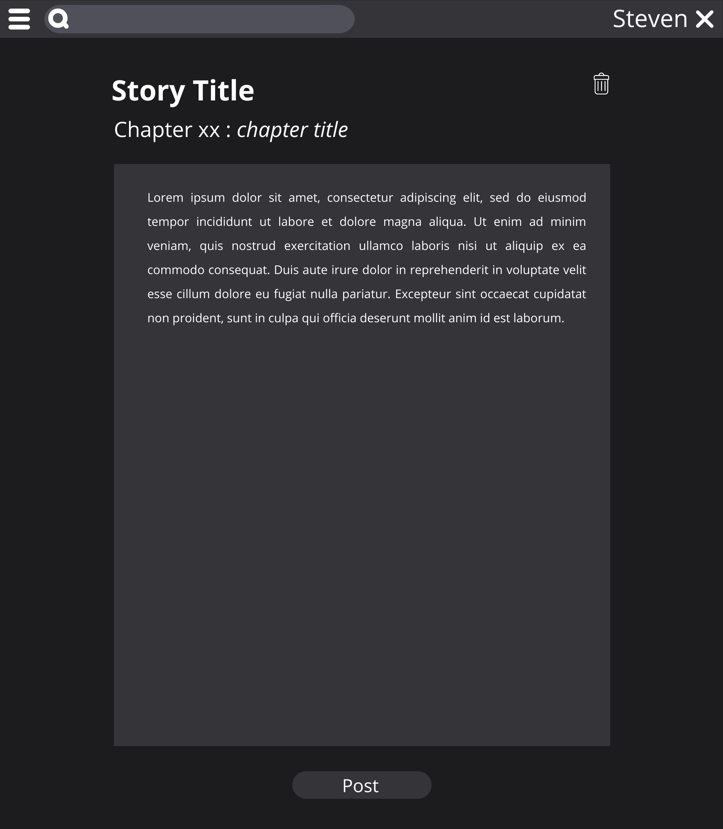
**

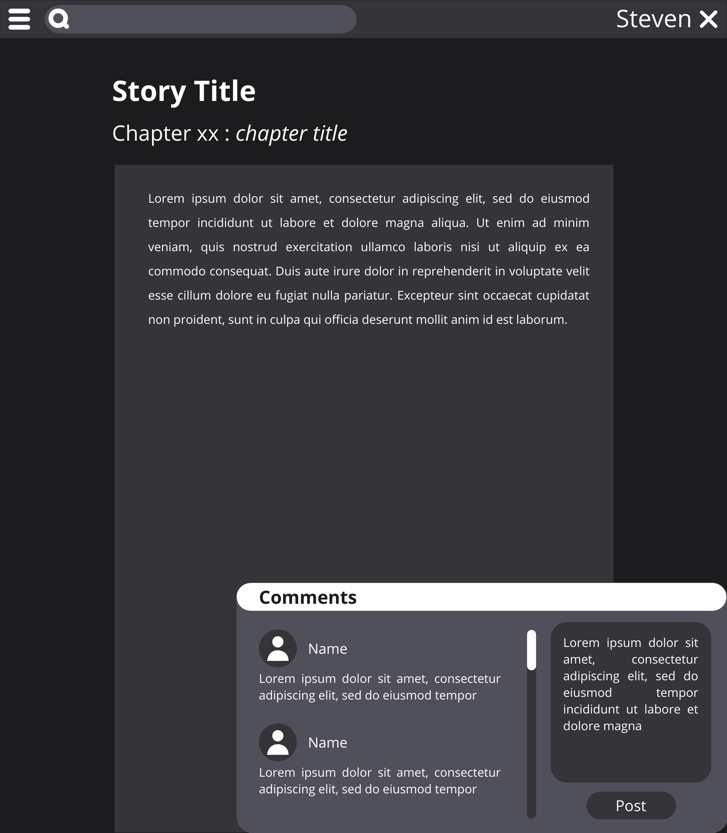
**

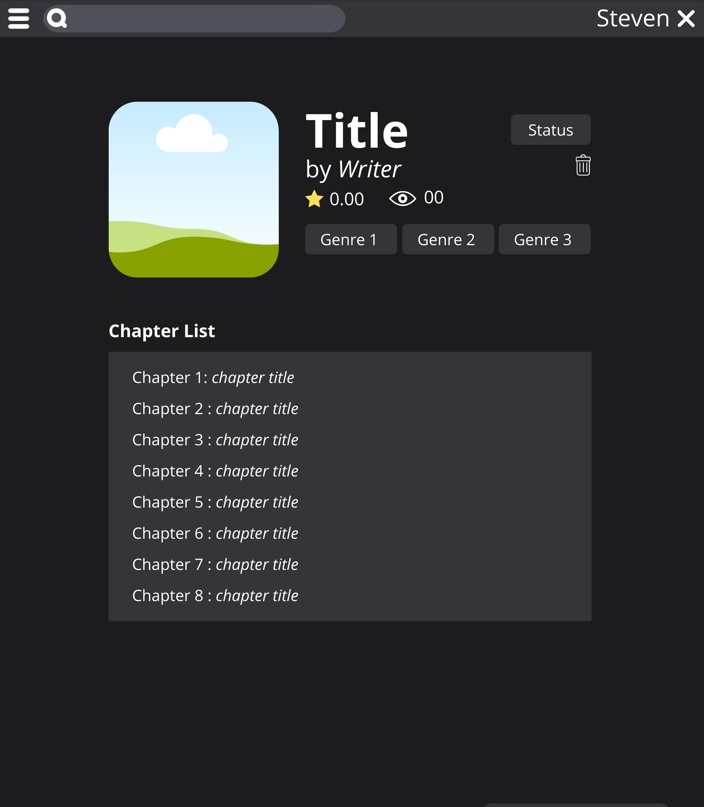
Add Chapter Change Password

**

Menu for comment (User) Edit Story

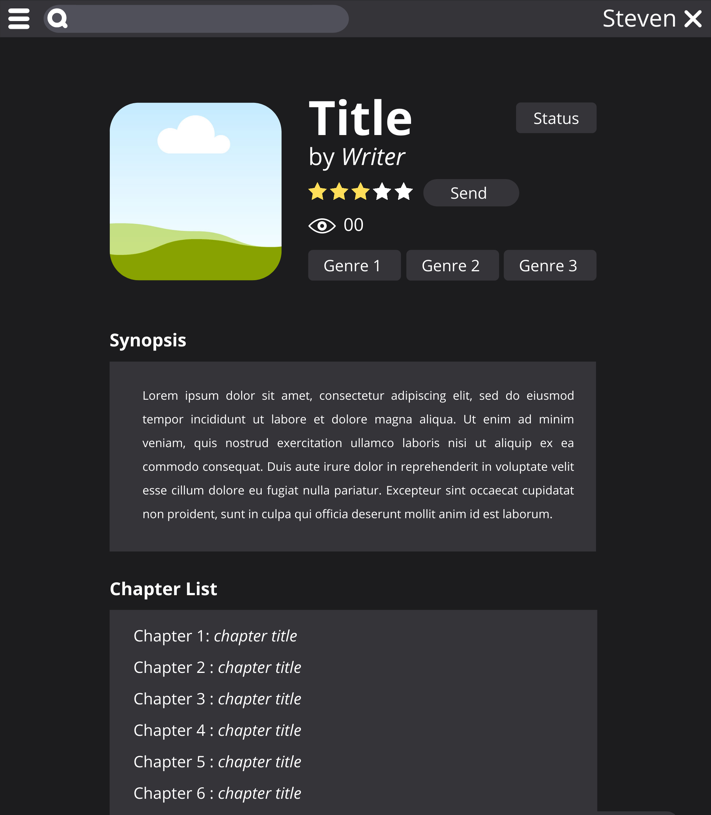
**

**

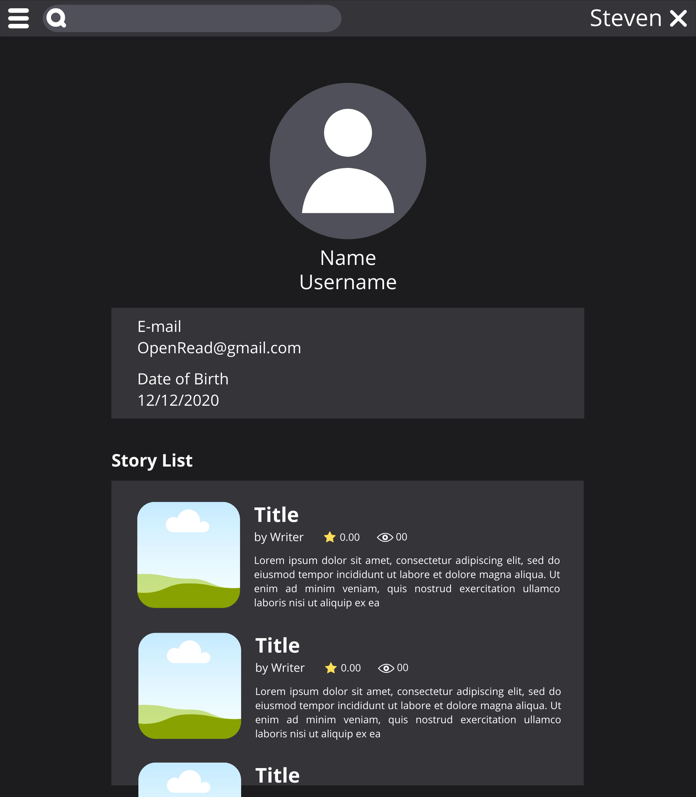
**Add Chapter for existing story Menu to choose which chapter to edit

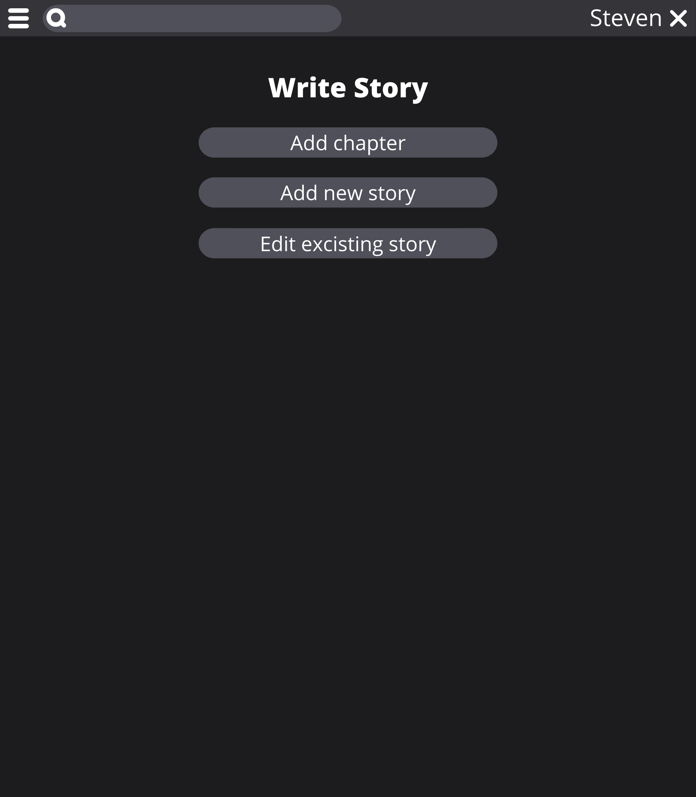
**

Menu to choose which chapter to read Menu to choose which story to edit

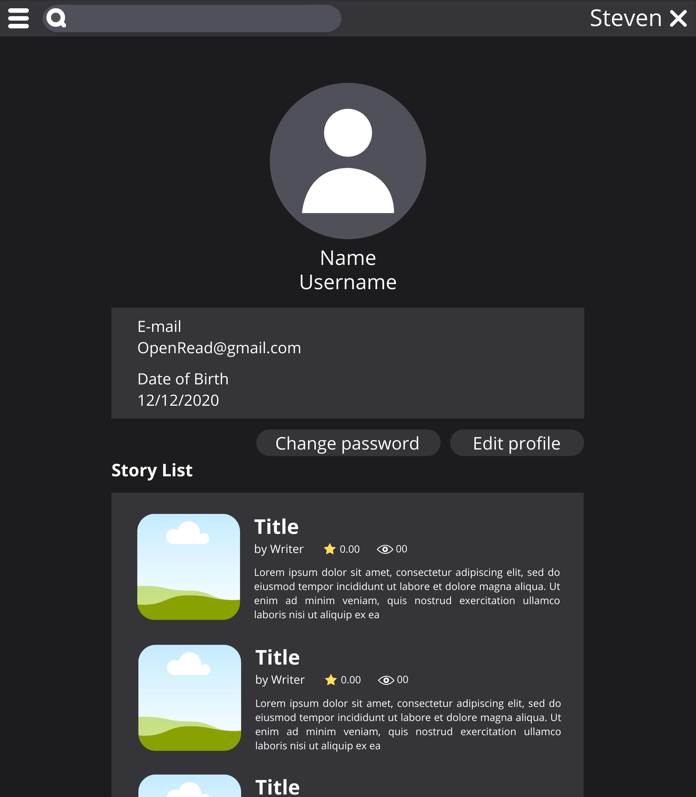
**

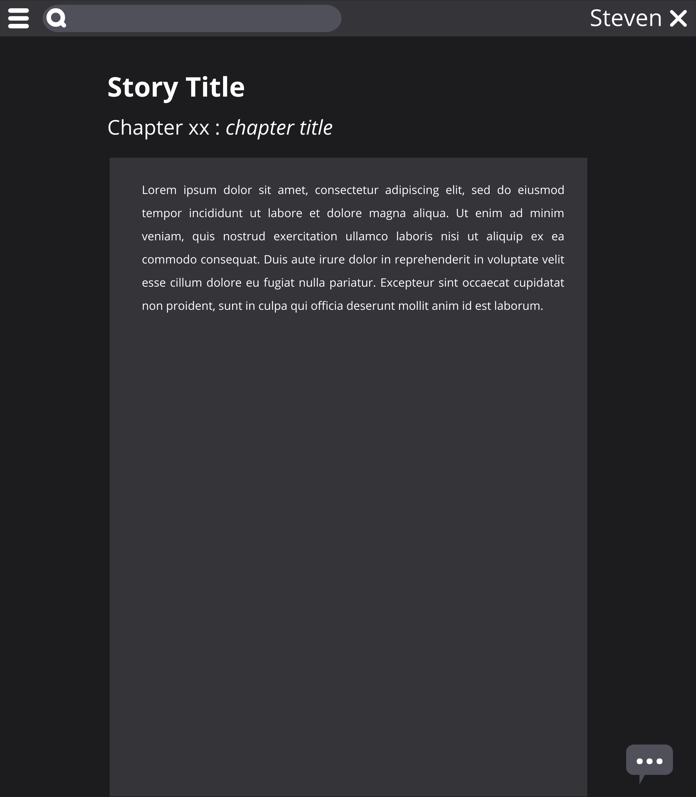
Menu to write story User visiting other user’s profile

**

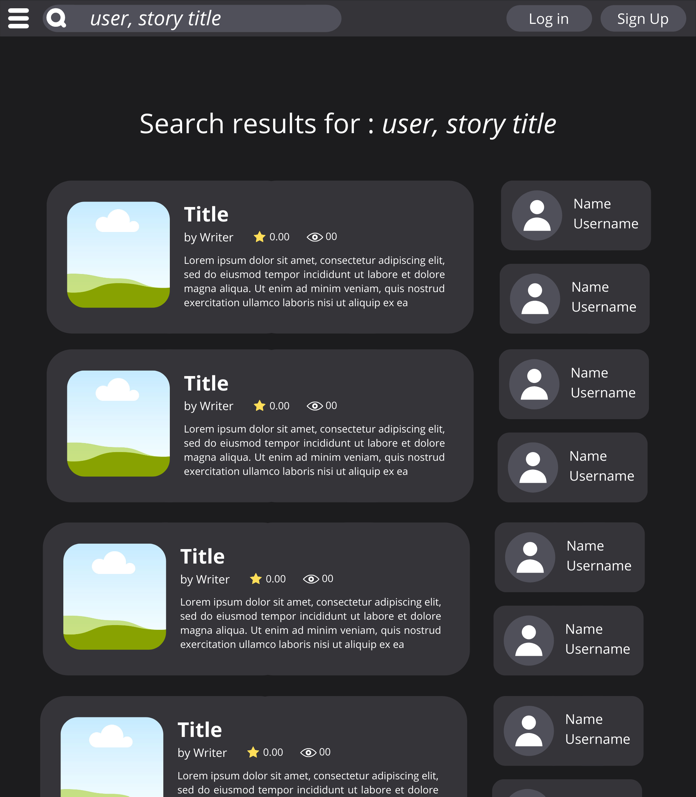
**

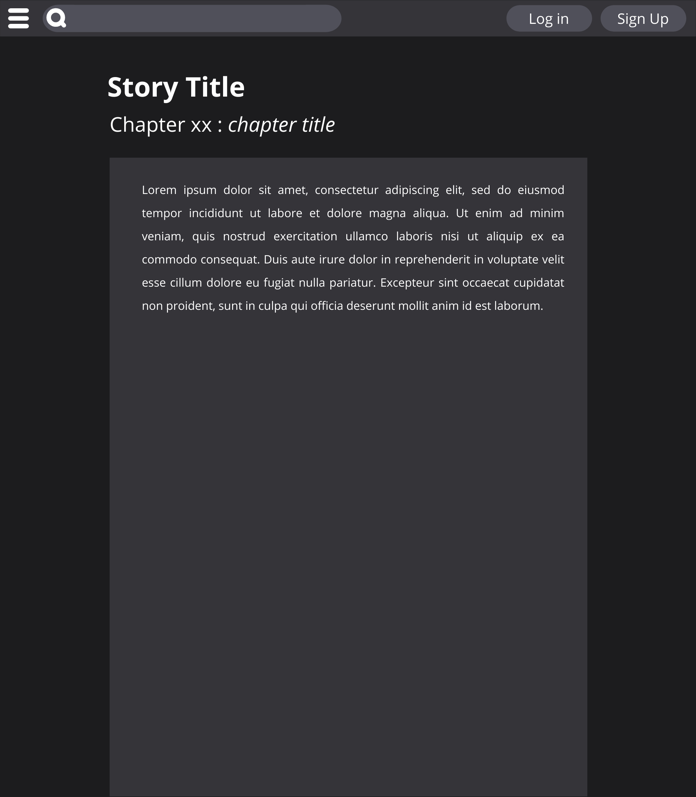
User’s Profile Page Read Story (User)

**



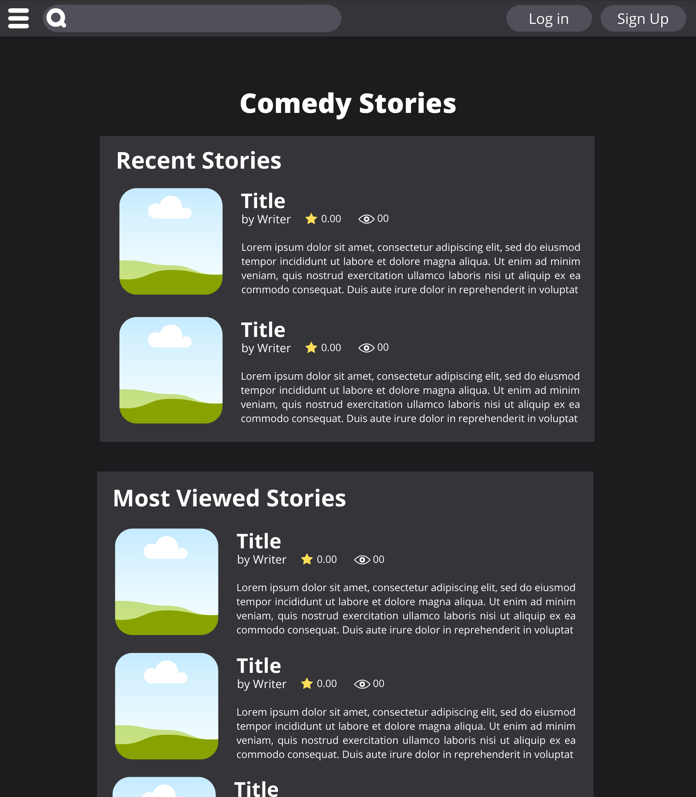
Read Story (Visitor) Search Results (Visitor)

**

**

Side Menu Home Page (Visitor) Stories for selected genre





## Hardware Interfaces

**Minimum Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| **Client Side** | | | |
|  | Processor | RAM | Disk Space |
| Google Chrome | 1 GHz CPU | 128 MB | 100 MB |

|  |  |  |  |
| --- | --- | --- | --- |
| **Server Side** | | | |
|  | Processor | RAM | Disk Space |
| Apache Web Server | 1 x 1.75 GHz CPU | 1.75 GB RAM | 40 GB |
| MySQL Database | 2 x 1.75 GHz CPU | 1.75 GB RAM | 40 GB |

**Recomended Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| **Client Side** | | | |
|  | Processor | RAM | Disk Space |
| Google Chrome | 2 GHz CPU | 256 MB | 200 MB |

|  |  |  |  |
| --- | --- | --- | --- |
| **Server Side** | | | |
|  | Processor | RAM | Disk Space |
| Apache Web Server | 2 x 1.75 GHz CPU | 3.5 GB RAM | 40 GB |
| MySQL Database | 2 x 1.75 GHz CPU | 3.5 GB RAM | 40 GB |

## Software Interfaces

The software architecture must follow Model-View-Controller design pattern. The application must also follow 4-tier architecture which includes Repository, Service, Controller, and View. Definition for each layer of architecture is as follow:

* Repository layer serves as a communicator between the application and the database. Query, Transactions, and other database related logic must be applied at this layer.
* Service layer is where all application’s business logics are defined. This is where we apply the use cases and implement it into code. Service layer will either receive data or none then writes or reads such data into the repository layer.
* Controller layer is the middleman between client side and server side. It receives HTTP Request, validate the data if it were given, send it to the service, and returns a response.
* View layer serves as the presentation of the application. Because the application nature is that of a web application, this is where HTML, style and Javascript is located. View layer also determines what to show or what to not show based on the data given from the controller.

Client-side user must be able to open the application in all these following web browsers: Microsoft Edge, Firefox, Safari, Google Chrome. There is no limitation on chosen Operating System if it could open at least one of the stated web browsers. The application must be able to run with Apache Web Server with any suitable Operating System. The application must use MySQL Database as its data storage.

For development-end these requirements must be met:

* Programming Language: PHP 7.3/8.x
* Web Server: Apache
* Database: MySQL
* Development Environment: XAMPP 8.0
* Source Code Editor: Visual Studio Code
* PHP Dependency Manager: Composer 2.0.8
* PHP Framework: Laravel 8.0
* Node Package Manager (npm) 6.14.10

Libraries that will be used for development:

* Laravel/ui v3.1
* Laravelcollective/html v6.2
* Front-end CSS Framework: Bootstrap v4.5.3
* jQuery v3.5.1

## Communications Interfaces

The communication architecture must follow the client-server model. Communication between the client and server should utilize a REST-compliant web service and must be served over HTTP Secure (HTTPS). The client-server communication must be stateless.

# System Features

This website supports all types of web browsers. We are using simple electronic forms for writing story, writing comment, reading story, etc.

* 1. **Register Account**

|  |  |
| --- | --- |
| **Use Case Name** | **Register Account** |
| **XRef** | Section 2.2.1, Register Account |
| **Trigger** | The user presses “register” button at the navbar or opens register page. |
| **Preconditions** | 1. The user must not be logged in 2. There is no existing username or email on the database |
| **Basic Paths** | 1. The visitor inputs their username, email, password, and date of birth. 2. The visitor presses Sign Up button. 3. If the register is successful, the visitor will be logged in and redirected to the home page. |
| **Alternative Paths** | – |
| **Postconditions** | The visitor’s account is created. |
| **Exception Paths** | If there is an existing account with same username input or email input, the register will fail and the visitor must input another username or email. |
| **Other** | None |

* 1. **Login**

|  |  |
| --- | --- |
| **Use Case Name** | **Login** |
| **XRef** | Section 2.2.1, Login |
| **Trigger** | The visitor selects the login button |
| **Preconditions** | 1. The visitor has registered. |
| **Basic Paths** | 1. The visitor enters his/her username/emails and passwords 2. The system will validate whether the credentials match. |
| **Alternative Paths** | - |
| **Postconditions** | Visitor is logged in and status changed to User |
| **Exception Paths** | If the visitor enters incorrect password or the username/email he/she enters didn’t exist in the database, the login process failed and the system will ask the visitor to enter again his/her credentials. |
| **Other** | None |

## Search Story

|  |  |
| --- | --- |
| **Use case name** | **Search story** |
| **Xref** | Section 2.2.2, Search Story |
| **Trigger** | The reader clicks the search button |
| **Preconditions** | The reader has accessed the website home page |
| **Basic paths** | 1. The reader writes down the name of the story or the author name specifically 2. The system will find the story or author name related to what the reader search for. 3. If there are any story or author name related, the system will show the story. If there aren’t the system will write “The story or author name not found” and show another story available |
| **Alternative paths** | None |
| **Postconditions** | Show the story |
| **Exception paths** | The reader can abandon the operation at any time |
| **Other** | None |

## Write Comments

|  |  |
| --- | --- |
| **Use Case Name** | **Write Comment** |
| **XRef** | Section 2.2.2, Write Comments |
| **Trigger** | The reader selects add button |
| **Preconditions** | The reader is on a story page and the reader had logged in. |
| **Basic Paths** | 1. The reader types his/her comments. 2. The reader posts. 3. The system asks for the reader to review his/her comments to make sure it doesn’t contain any inappropriate content. 4. The reader chooses to continue/review. 5. The system reads the comment and posts it at the comment section. |
| **Alternative Paths** | In step 4, if the reader chooses to review, the reader can edit his/her comments and goes back to step 2. |
| **Postconditions** | Comment is posted. |
| **Exception Paths** | The reader can abandon writing comments anytime. |
| **Other** | None |

## Update Comments

|  |  |
| --- | --- |
| **Use Case Name** | **Update Comment** |
| **XRef** | Section 2.2.2, Update Comment |
| **Trigger** | The reader selects pencil button |
| **Preconditions** | The reader is on a story page and the reader had logged in. |
| **Basic Paths** | 1. The reader updates his/her comments. 2. The reader posts. 3. The system asks for the reader to review his/her comments to make sure it doesn’t contain any inappropriate content. 4. The reader chooses to continue/review. 5. The system reads the comment and renews it at the comment section. |
| **Alternative Paths** | In step 4, if the reader chooses to review, the reader can edit his/her comments and goes back to step 2. |
| **Postconditions** | Comment is edited and renewed. |
| **Exception Paths** | The reader can abandon updating comments anytime. |
| **Other** | None |

## Rate Story

|  |  |
| --- | --- |
| **Use Case Name** | Rate Story |
| **XRef** | Section 2.2.2, Rate Story |
| **Trigger** | The user clicks the stars on the story that he/she wants to rate. |
| **Preconditions** | 1. The reader is on a story page and the reader had logged in. |
| **Basic Paths** | 1. User clicks on the amount of stars that he/she wants to give. 1 star mean the rating will be 1. Maximum will be 5 stars. 2. User clicks send. |
| **Alternative Paths** | - |
| **Postconditions** | The rating will be recorded to the database.  The corresponding story’s rating will be changed according to the new rating |
| **Exception Paths** | - |
| **Other** | None |

## Edit Profile

|  |  |
| --- | --- |
| **Use case name** | **Edit Profile** |
| **Xref** | Section 2.2.2, Edit Profile |
| **Trigger** | The user opens the edit profile button |
| **Preconditions** | The reader has logged in and accessed their own profile page |
| **Basic paths** | 1. The user clicks the edit profile button. 2. The user changes their information as necessary. 3. The user clicks the Save changes button. 4. If user changes their username or email, the system will check whether the new username or new email has already been used. |
| **Alternative paths** | At step 3, if user decided to not change their information, they can click Cancel button and the system will abandon any changes. After that, the user will be redirected back to their own profile page. |
| **Postconditions** | If the changes are successfully saved, the user will be redirected back to their own profile page. |
| **Exception paths** | At step 4, if the system finds an existing username or email, the system will return an error saying that “Username/Email has already used”. |
| **Other** | None |

## Change Password

|  |  |
| --- | --- |
| **Use Case Name** | **Change Password** |
| **XRef** | Section 2.2.2, Change Passowrd |
| **Trigger** | The user clicks the change password button |
| **Preconditions** | The reader has logged in and accessed their own profile page |
| **Basic Paths** | 1. The user clicks the change password button. 2. The user fills their current password (old password), desired new password, and re-type the new password. 3. The user clicks Save button. 4. The system will do several validations:    1. The old password input must be equal to user’s current password.    2. The confirm password input must be equal to the new password input.    3. The new password input must not be equal to user’s current password. 5. If the system successfully validates user’s input, the system will hash the new password and updates the new password in the database. |
| **Alternative Paths** | At step 3, if user decided to not change their password, they can click Cancel button and the system will abandon any changes. After that, the user will be redirected back to their own profile page. |
| **Postconditions** | If the changes are successfully saved, the user will be redirected back to their own profile page. |
| **Exception Paths** | At step 4, if the system finds any of the inputs fail the validation, the system will return an error to the form. |
| **Other** | None |

## Write Story

|  |  |
| --- | --- |
| **Use Case Name** | **Write Story** |
| **XRef** | Section 2.2.3, Write Story |
| **Trigger** | The writer selects create story button |
| **Preconditions** | The writer is on the create story page and writer has logged in |
| **Basic Paths** | 1. The writer selects the create story button 2. The writer enters story title and genre 3. After entering title and genre, system automatically adds one new chapter to the story. 4. Writer enters title for the chapter and its content. 5. The writer reviews the story title, chosen genre, chapter and its content. 6. The writer chooses to publish the story. 7. The system publishes to story on to the website. |
| **Alternative Paths** | In step 5, if the writer chooses to review the story’s information, the writer can edit his/her decision and goes back to step 2.  In step 5, if the writer chooses to review the chapter’s information or content, the writer can edit his/her decision and goes back to step 4. |
| **Postconditions** | Story is published on the website |
| **Exception Paths** | The writer can cancel story creating at any time |
| **Other** | None |

## Edit Story

|  |  |
| --- | --- |
| **Use Case Name** | **Edit Story** |
| **XRef** | Section 2.2.3, Edit Story |
| **Trigger** | The writer selects create story button |
| **Preconditions** | The writer is on the create story page and writer has logged in |
| **Basic Paths** | 1. The writer selects the edit story on the story he/she wants to edit. 2. The writer updates title or genre of the story. 3. The writer chooses a chapter he/she wants to edit. 4. The writer updates title or content of the chapter. 5. The writer reviews the story title, chosen genre, chapter and its content. 6. The writer chooses to accept the story’s edit. 7. The system republishes the story on to the website. |
| **Alternative Paths** | In step 3, if the writer chooses to create a new chapter into the story, the system will provide a new chapter form.   1. Writer enters title for the chapter and its content. Goes to step 5.   In step 5, if the writer chooses to review the story’s information, the writer can edit his/her decision and goes back to step 2.  In step 5, if the writer chooses to review the chapter’s information or content, the writer can edit his/her decision and goes back to step 4. |
| **Postconditions** | Story is republished on the website |
| **Exception Paths** | The writer can cancel story editing at any time |

# Other Nonfunctional Requirements

## Performance Requirements

* OpenRead can be accessed via browser with respond time less than 5 second.
* The database will be gone through the normalization process so there are no redundant data.
* The application is designed to handle lots of user simultaneously.
* The application can process records up to 100 for every transaction.

## Safety Requirements

* There will be regular database backups to ensure that user data, both stories and others, are safe.
* When there is an error in inserting, updating, and deleting transactions, we can rollback.
* Implement an exception handling system so as not to break the application when an error occurs.

## Security Requirements

* User’s data will be stored properly and classified, especially for password data will be encrypted
* Provide an information statement to the user, that this system is based on an online / website that is safe, secure, reliable stores and guarantees member data will be safe
* Each feature can be used by users according to their privileges
* User input data will be validated and sanitized before being entered into the database.

## Software Quality Attributes

* Correctness: the application is made according to the needs where this application can be a container for expressing user ideas in the form of written works.
* Efficiency: Application efficient in terms of coding and efficient in terms of operation by the user
* Integrity: The application has an integrated security system to prevent unauthorized abuse.
* Flexibility: Applications can be modified to suit the user's wishes in the future.
* Maintainability: The application is easy to do maintenance or changes if there is a problem with the application.
* Reliability: the application can be used without failure.
* Testability: Applications have been tested to ensure that they meet the requirements
* Usability: the application can be used easily because the appearance of the application is neat and simple.

## Business Rules

To make the OpenRead application easier and more convenient to use, we will update the system according to user suggestions.