**Fiction Fans**

Description

**Fiction Fans** is a web application for users to read, write and post fiction. The application collects all fictions from published fiction on the website, and users can read the story by searching the fiction's name or from genres. Users can also give a rate, leave the comment and follow the authors to get updates of their new recent updates.

This application is intended for users who are entertained by reading and writing stories.

Background

Electronic books (E-Books) are very well known because they can be used anywhere and anytime, and it is easy to access by just traveling on the internet. Most people have been using electronic books nowadays such as students, teachers, workers and the authors as well.

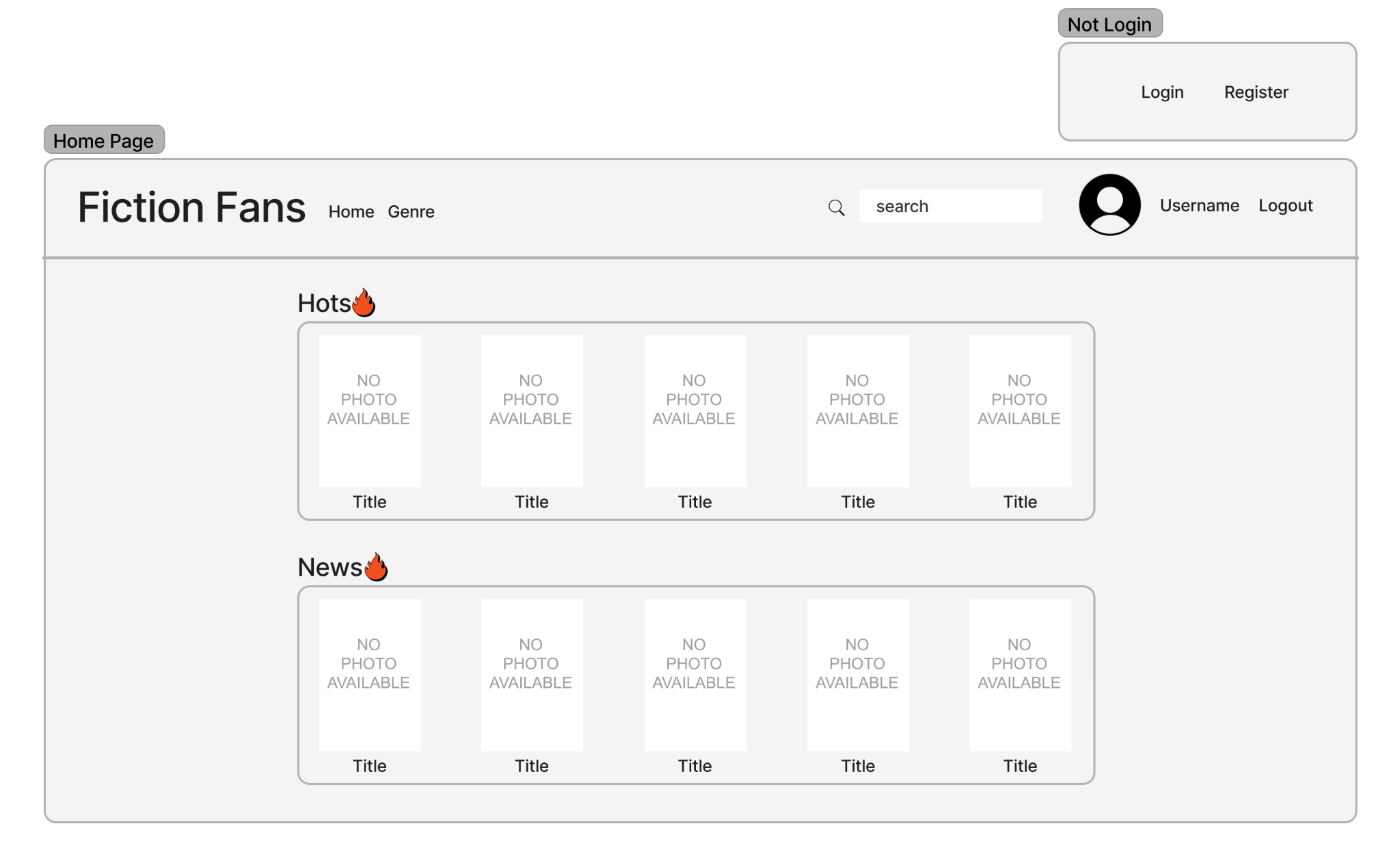
At the present time, many authors use e-books to publish their fiction on the internet because in cloud applications people can access it anytime and anywhere with just an internet connection. Not only does fiction get easier to access but most fiction websites also make readers and writers communicate with each other by providing a rating system and comments. These things are the benefits that writers and readers will get from using our web applications instead of physical books.

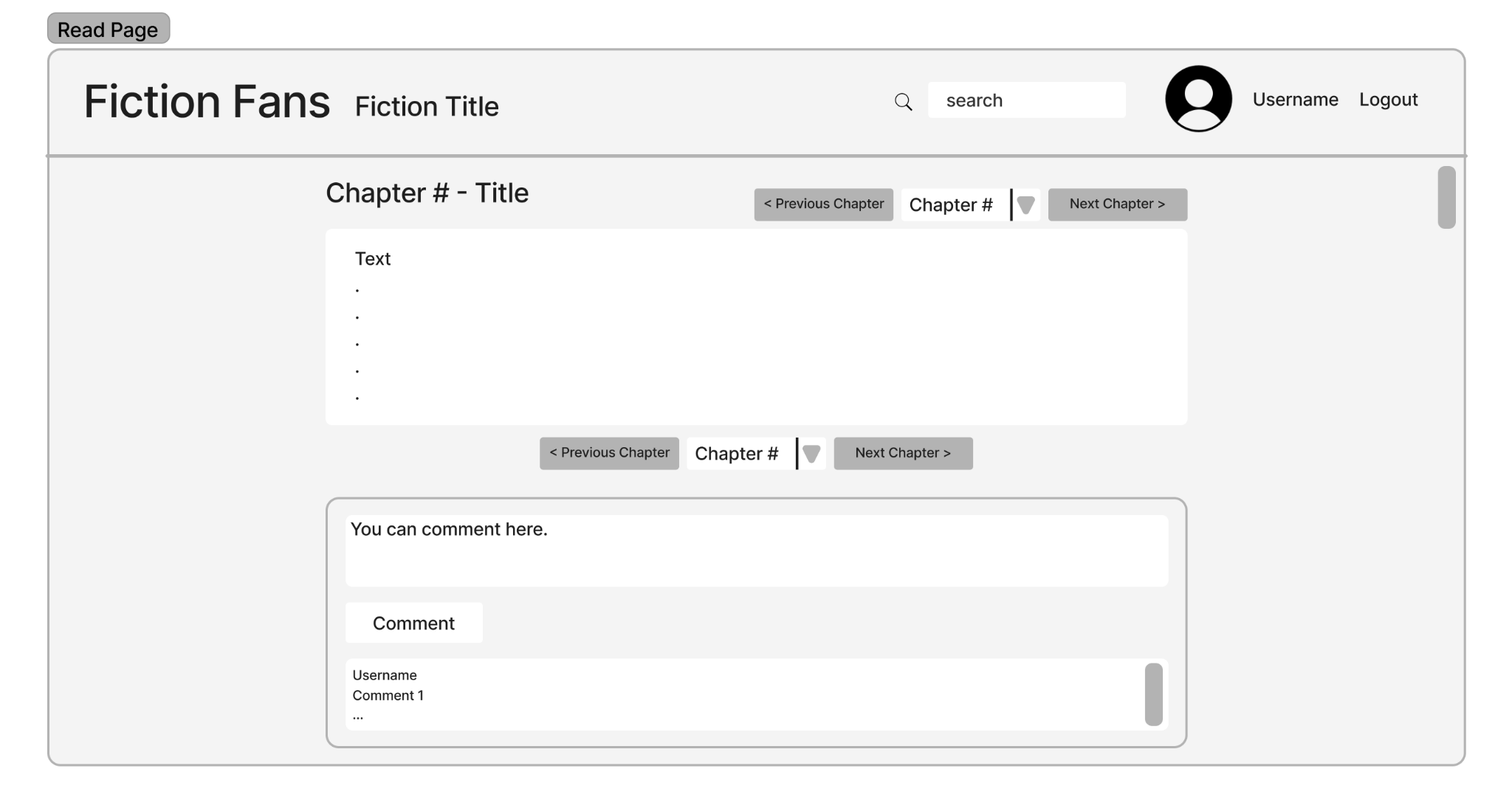
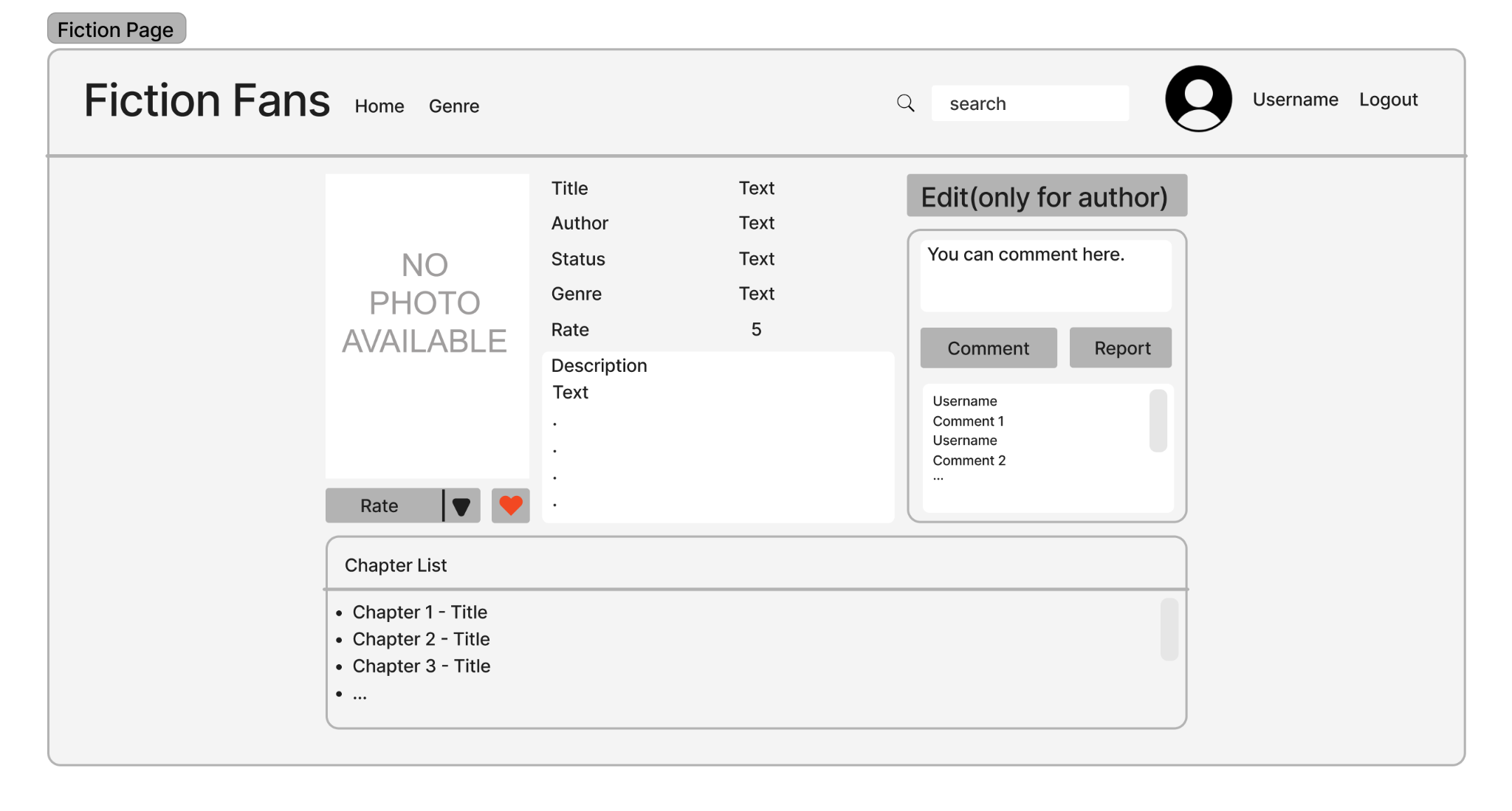
For these reasons, our group intends to create a web application to create a community by letting the writers share their own fictions with their own style freely and the readers share their own opinion about the story. This would allow new writers to practice writing stories for free.

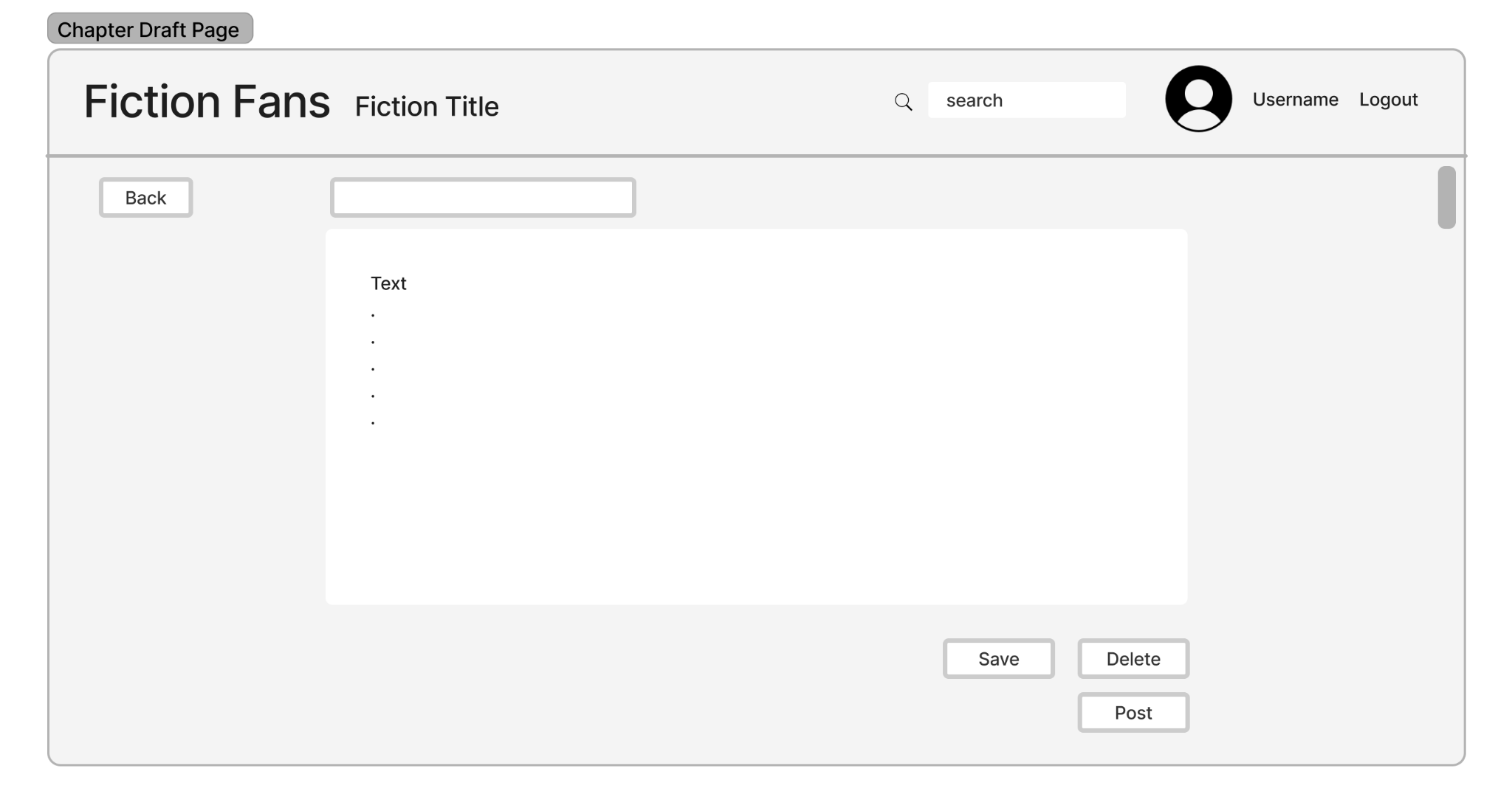
Features

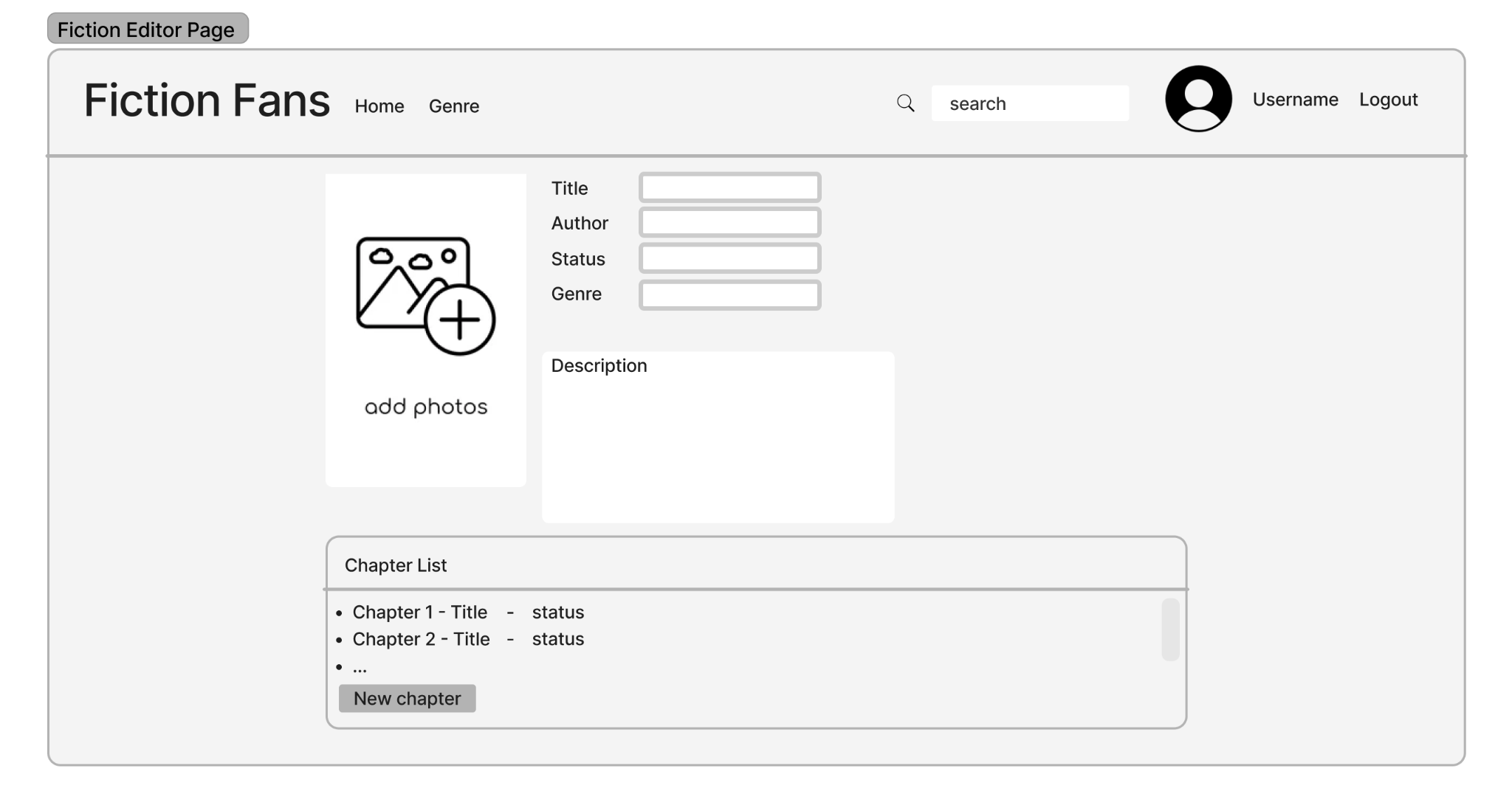
* User authentication system for member register, login and logout.
* Anyone can read published fiction.
* Users have a profile page
  + Username
  + All published stories
  + All drafts
* Home pages contain
  + Hot stories which show the top 5 fictions by user rating.
  + 5 New fictions which were recently published.
  + Search tab for searching stories by fiction’s name.
  + Link for redirect to
    - User page
    - Genres that link to each index page of that genre.
* Users can create or edit their own make-up fiction and publish on the website.
* Users can add or edit genres to the fiction.
* Users can add Illustrations into their stories.
* Users also can create drafts of their stories. Only the owner can access the drafts.
* Users can report fictions that are plagiarized to the admin, then the admin will look at reports and decide to remove those stories or not.
* The fictions can be categorized by type of the fiction referring to genre such as horror, comedy, romantic, action, etc. And its length such as series and one shot.
* Users can add fictions to their favorite and receive notification by email directly when the favorite fiction has new updates.
* Only registered users can rate the fiction, leave a comment, and report the fiction. Users cannot rate their own fiction.
* In the stories, the creator page will have character minimum in stories to prevent a too short sentence.

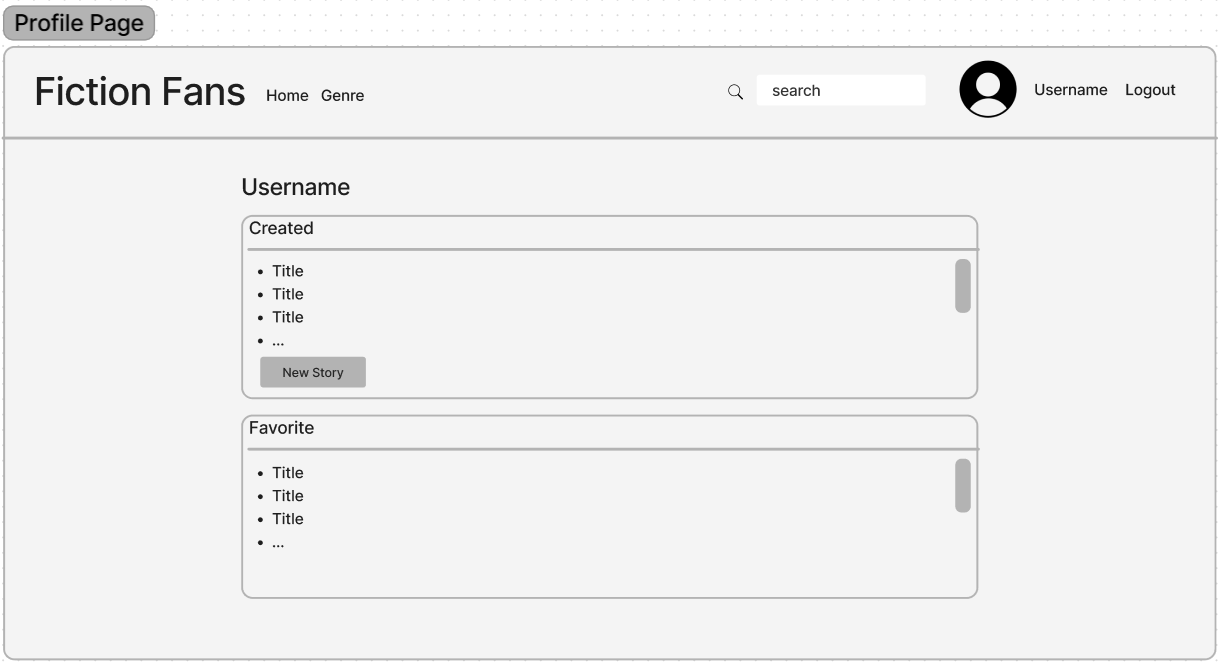
Mock - up











Team members

| Name | Student ID | Github |
| --- | --- | --- |
| Chinapat Rattanapirom | 6310546317 | chinapat317 |
| Pichaiyuth Uraisay | 6310546023 | inwpolol |
| Tanakorn Kullapatchawarit | 6310546325 | caneledip |
| Wongsathorn Deekaoropkun | 6310545353 | KwangSensei |

Development plan

| Iteration dates | Major Work and Goal(s) | Milestone(\*) |
| --- | --- | --- |
| 10 Oct - 16 Oct | Written features list and detailed user stories for main uses of project evaluator app. Mockup of screens. Domain model design in UML. | Main user stories written.  All docs on project wiki. |
| 17 Oct - 30 Oct | * In model file create django fiction application * Create story.html template to show a fiction text and picture * Create **class Story** in **model.py** * Create **class StoryView** in **view.py** to change from .txt file into string and display on story page. * Create url.py and url path for **class StoryView**. | A web application can show fiction on the web page in the correct format. |
| 31 Oct - 6 Nov | * Add command in setting.py for user authenticate * Create an **editor.html** template for writers to write a story on it and save it. * In view.py create **class** **EditorView** to receive post request from writer and keep it in .txt file and save picture to firebase picture cloud. | * A web application can login and let users add fiction from fiction creator pages. * Creator can add illustrations. |
| 7 Nov - 13 Nov | * Create **profile.html** template for display user information, own fiction list and favorite fiction list. * Create class rate for linking story rating and story model together * Create **fiction\_info.html** page for displaying fiction information eg, genres, chapters. * Create **class FictionView** in **view.py** for display fiction information. | * Web application has user profile page to show after login * Users who login can rate fiction which he/she reads and fiction rating will show in the fiction info page. |
| 14 Nov - 20 Nov | * Create **class IndexView** for display list of fiction that match genre. * Create a report method in **view.py** to receive post requests from users when they report inappropriate fiction. | * Web application has index pages for each genre. * Users can report unapropriate fiction. |
| 21 Nov - 27 Nov | * Create notification method in class **EditorView** for sending notification email to following user. | Users will get notification via registered email. |
| 28 Nov - 4Dec | * To be added | Deployed the web application on to the server. |

Major risks

* The team members do not have enough knowledge. We may take too much time for researching so our project may not finish with required features.
* We are still not sure how to find the suitable tools for our features. In the website, we will keep the story in a text file when we use the open method in the app and may have problems with formatting.

Software process

Our team will use an iterative and incremental process with iteration length of 7 weeks, starting on Monday and ending Friday.

Team will meet every Friday.

Retrospective will be held on Friday. A summary of the retrospective with a plan for process improvement will be recorded in project wiki.

Implementation

* Python
* Django framework
* HTML
* CSS

Process Components and Tools

* Configuration management: Use github to store source code and github workflow to manage and maintain code to always be deployable.
* Documentation: The iteration plan, design document, summary of idea on retrospective, and other useful documents will be available in project wiki.
* Issue Tracking: Github for issue tracking.
* Communication: The team will be exchanging information, and also meeting, via discord. The key information each exchange will be pinned so that it can be looked up easily.
* Task Management: Our team will use the Django framework for the development and also use HTML and CSS to decorate the UI of the application.
* Visibility: in the project README.md on Github, include links to the task board, documentation (esp. iteration plans), issue tracking (Github), and communication channel.
* Team Roles:
  + Project owner/ Scrum master: Chinapat Rattanapirom
  + There is no definite role for our team members. However, maybe in some part of development, if there is a part, which our team member is good at, we will let that team member work on that task.
* Quality Assurance: We have team members who are used to reading webnovels on online platforms so we can let him be a QC member. Also, when the demo of the application is finished, it will be lent to some users who are used to reading stories on online websites to try it out and review.
* Continuous Integration Server: Github Action
* Deployment: (To be decided which cloud service to use) (Edit: We use pythonanywhere for deployment.)