# Interim Reflective Report for Software Engineering Group Project

2021/2022 Semester 2
Leisure Town

*2022.5.13*

Lincheng Shi
1927978
A-14

## Table of Contents

# Introduction

## Background

In this fast food era, fierce competition leads to people's increasing pressure. A simple blog platform for sharing life can help them slow down and relieve stress. The main social products on the market now have a common problem: paying too much attention to business goals. Therefore, we aim to build a blog platform called "Leisure Town", which has a simple and clear interface, focuses on sharing life, and can express personal ideas about the same hobbies. Because our products do not have complex functions and gorgeous interfaces, it may be difficult to attract young people. Our main target users are between 25-50 who have great life pressure.

## Overview of The Completed Works

In sprint 1, we mainly complete the login and registration functions. Both pages use a background image and a dialog box. In the registration box, you must enter user name, nickname and password, and their length cannot exceed 10 characters. At the same time, the user name cannot duplicate the existing one. Otherwise, different tips will appear. In the login box, you must enter the corresponding user name and password. The length requirements are the same as above. After successful registration or login, it can directly jump to the main page. JWT technology and redis technology are also used to make a double guarantee when registering or logging in. We also set up a login interceptor to make unified login judgment.

In sprint 2, we mainly completed part of the home page and logged out. In the home page, there is a navigation bar at the top, the slogan of our web application at the bottom, and a list of articles in the middle. In the article list, you can see the title, abstract, author, tag and creation time of the article. When logging out, the front end can clear the token in JWT, while the back end is responsible for clearing the information in redis.

In sprint 3, we mainly completed the rest of the home page, viewing article details, commenting and publishing articles. The rest of the home page includes the number of readings per article and the hottest town tags. The number of readings will increase with the increase of clicks. The hottest town tag shows the tags in the top six. The article details page displays the article content, category, tags, comments, etc. Published articles refer to a markdown editor (the reference is in the appendix 1) to write articles. After completing the abstract, tags and article category, click "publish article" to publish successfully and jump to the home page. The system will also automatically check your spelling mistakes when commenting and writing an article abstract. Publishing articles and comments can only be done after logging in.

Up to sprint 3, "Leisure Town" can mainly log in and register, view articles and post articles. I think the software has many strengths and weaknesses. For strengths, its interface and functions are not complex, and there is little information to fill in when registering. Therefore, it is easy to use and has high learnability. Besides, the green article tags and article category are very eye-catching, so that viewers can clearly understand what the article is about. The display of reading times can also let users understand the popularity of their shared content, and pave the way for us to develop the hottest article section later. Most notably, you can use an editor when writing articles. Bold, segmentation, inserting pictures, using Emoji, immersive reading and so on can be easily realized. This greatly deepens the interest and readability of the article. For weaknesses, when the user forgets the password, there is no measure to find the password. Some contents are in English, while other contents are in Chinese, which is also a less friendly point. In addition, more tags and article categories need to be added to meet the needs of people with different hobbies. Finally, if there are too many articles on the home page, the slogan of our product will appear only when the scroll bar rolls to the article on the last page.

# Software Processes

## Overview of Agile

Agile method is a kind of software process with human centered, iterative, incremental and flexibility. This method has become popular due to globalization and the rapidly changing business environment. When using Agile method, development teams tend to focus on the software itself rather than documentation. Because requirements change rapidly and unpredictably, the team can make appropriate compromises on software quality and requirements. Besides, delivering the system as soon as possible so that users can find more clear and accurate real requirements and face-to-face conversation are also the characteristics of Agile method. These also can be summarized in the agile manifesto. Of course, Agile method still has many challenges. For example, some organizations find it difficult to accept this informal definition process. Also, some people may not like this kind of high participation because of their personality. These make it difficult to achieve.

## Application of Agile in The Project

In this project, our team roughly divided 4 people to do the front end and 5 people to do the back end (including database), but the division of labor is not fixed. After completing their own work, the product owner and scrum master will also appropriately help the development team to do some tasks. Before the whole project starts, we first determine the theme of the project, and then plan the PBI of the first three sprints. Before the start of each sprint, we improve the PBI of the next sprint, and then tasks will be assigned to members. These tasks include the problems left over from the previous sprint. In sprint, the front and back end communicate actively, and everyone works independently and helps each other. At the end of each sprint, we do some tests. After the three sprints, the first release of our products came out. I think our group has achieved human centered, incremental, iterative and pay more attention to the software itself.

There are four main problems we encountered during the application of Agile. The first one is we seldom do daily scrum because everyone has different time arrangements for themselves and their studies are heavy. Maybe some members did part of the task today, and other members made up the task the next day. Therefore, I made a suggestion to the team members: at the end of each day, members who have made progress in the task just send a message in the WeChat group to report. The second problem is that the product owner has arranged specific testers to test each sprint, but they don't seem to pay much attention to this part. Sometimes they don't do it carefully, sometimes they ignore this step. I value this part very much, so I usually do an in-depth test and provide the bug to all members. The third problem is that the development of sprint 1 is relatively slow. Because we have little exposure to real agile development, we need a lot of learning and comparison. My idea is to reduce the PBI of the sprint 1 and appropriately leave some tasks for later sprints. After all, our efficiency will be higher and higher after being proficient. The last problem is an inevitable problem: we just can develop our software online. Each of us is separated into different places, so this has a certain impact on our development, such as reducing efficiency. What I can do is to actively guide members to communicate.

During my project, I found some areas that need to be improved when we use Agile method. Due to our lack of experience, the progress of sprint 1 is very slow. I hope to add another week to study and make decisions before sprint 1. In addition, considering that some people may not invest much energy in the testing part, I think we can set another position in the team. Because the workload of only testing may be smaller than that of others, it is better for this position to be able to design some icons by himself and carefully complete the testing of finished product. Finally, although we allocate tasks according to the same estimated time, the time for members to complete tasks varies greatly. This is because the efficiency and expertise of members are different and the task allocation is random. Therefore, I propose to assign tasks according to the technical level of members.

### Comparing Agile to Other Software Processes

Plan-driven process is also a popular software process. In this software process, all process activities are planned in advance. Users generally need to see the "fully completed" products before they can judge whether they meet the expectations. A typical model is the waterfall model. In this model, after one phase is completed, the next phase can be started. At the same time, documentation is important and iteration is expensive. In the agile process, the plan is incremental and it is easy to change the process to reflect the change of user needs. A typical model is incremental development.

## Individual Work

### Sprint 1 Overview

In sprint 1, our aims and objectives are as follows:

1. build the springboot framework
2. complete PBIs of login and registration and front-end of homepage

We have two PBIs in this sprint. They are #1 Login and #2 Registration. Other details can be seen in the appendix 2. In this sprint, we lack experience, so the PBIs and tasks of sprint 1 listed here are not initial, but updated.

#1 can be divided into 7 tasks: 1.1 UI – login page, 1.2 Front-end api, 1.3 Front-end script, 1.4 Router – to main page, 1.5 Login interceptor, 1.6 Back-end api coding, 1.7 Testing. #2 can be divided into 7 tasks: 2.1 UI – registration page, 2.2 Front-end api, 2.3 Front-end script, 2.4 Router – to main page, 2.5 Back-end api coding, 2.6 Database initialization, 2.7 Testing.

## Personal Contribution to Sprint 1

In sprint 1, I first learned a lot of basic knowledge. For example, how to connect the front and back ends, how to connect the database, something about springboot and postman. Then I wrote the code of jumping from the login page to the registration page. Since I need to conduct tests and other people's pages haven't been designed yet, I simply wrote two pages: login and registration. When learning the basic knowledge, I watched all the videos on learning mall and found a lot of information on the Internet. I learned that we need to conduct cross domain configuration when connecting the front and back ends. Because the front and back ends are two different servers. To connect to the database, we only need to write our URL, username, password and driver in the configuration file. When writing jump code, I only need to write a jump function to jump to the corresponding page file, and then call this function with "onclick" between <body> and </body>. During the test, I found the code of login and registration page on the Internet, and made simple modifications to test my jump function.

In the first week of this sprint, we actually did a lot of repetitive work because all members were inexperienced. Almost everyone wrote the front-end code of the login and registration page. My page "jump" didn't give effective help to the team in the follow-up. In the second week of this sprint, I didn't do much practical work. I just helped our team establish the environment and did a small amount of back-end code for login. I spent most of my time studying the front-end and back-end knowledge by myself, which also laid a great foundation for me to help the front-end correct a small number of bugs and complete the back-end code in the later development. This is the hardest sprint.

## Sprint 2 Overview

In sprint 2, our aims and objectives are as follows:
1. improve the functions and UI design in sprint 2
2. complete 4 PBIs of blog list, tags, authors and logout

We have 4 PBIs in this sprint. They are #3 View blogs (main page interface), #4 Blog tags, #5 Blog authors and #6 Log out. Other details can be seen in the appendix 2.

#3 can be divided into 7 tasks: 3.1 UI – main page + blog component, 3.2 Front-end api, 3.3 Front-end script, 3.4 Router – to main page, 3.5 Back-end api coding, 3.6 Database initialization, 3.7 Testing. #4 can be divided into 6 tasks: 4.1 UI – tags in the blog component, 4.2 Front-end api, 4.3 Front-end script, 4.4 Back-end api coding, 4.5 Database initialization, 4.6 Testing. #5 can be divided into 5 tasks: 5.1 UI – authors in the blog component, 5.2 Front-end api, 5.3 Front-end script, 5.4 Back-end api coding, 5.5 Testing. #6 can be divided into 5 tasks: 6.1 UI – log out, 6.2 Front-end api, 6.3 Front-end script, 6.4 Back-end api coding, 6.5 Testing.

## Personal Contribution to Sprint 2

In sprint 2, I mainly completed five things. I write the back-end code for the home page's article list display function, create part of tables related to the article list in the database, combine the front and back end, do almost all test and fix a small bug where the article list could not show the author. In back-end code, my interface url is “/articles”, request method is “post”, request parameters are “page(int)” which means current page and “pageSize(int)” which means the number of pages displayed. It returns json data. Article class and Tag class are automatically associated with the database through MyBatisPlus. The Mapper related to these two classes inherits the related BaseMapper. The data required by the front end is stored in the VO package. Return the data of article list in ArticleController. Query the article list page by page in ArticleService. ArticleServiceImp implements ArticleService. This includes top ranking, descending by time, and the transfer of corresponding data in the “Copy(Article article)” method. The small bug is “author” has “.nickname” in front-end file “ArticleItem.vue”.

In this sprint, I think I have made great efforts and basically have nothing bad to do. With the knowledge learned in the previous sprint, I can not only complete the back-end code of my own task and fix all bugs, but also help the front-end find bugs and make modifications. I also did almost all tests outside my own tasks and actively communicated with the team members.

## Sprint 3 Overview

In sprint 3, our aims and objectives are as follows:
1. improve the functions and UI design in previous Sprints
2. complete 5 PBIs of view count, trendy tags, blog detail, comment and posting blogs
3. focus on report writing

We have 5 PBIs in this sprint. They are #7 Blog view count, #8 Trendy tags, #9 Blog full text, #10 Comment and #11 Post blogs. Other details can be seen in the appendix 2.

#7 can be divided into 6 tasks: 7.1 UI – view count in the blog component, 7.2 Front-end api, 7.3 Front-end script, 7.4 Back-end api coding, 7.5 Database refinement, 7.6 Testing. #8 can be divided into 5 tasks: 8.1 UI – in the main page, 8.2 Front-end api, 8.3 Front-end script, 8.4 Back-end api coding, 8.5 Testing. #9 can be divided into 6 tasks: 9.1 UI – blog detail page, 9.2 Front-end api, 9.3 Front-end script, 9.4 Router – to full text page, 9.5 Back-end api coding, 9.6 Testing. #10 can be divided into 6 tasks: 10.1 UI – comment, 10.2 Front-end api, 10.3 Front-end script, 10.4 Back-end api coding, 10.5 Database initialization, 10.6 Testing. #11 can be divided into 7 tasks: 11.1 UI – post blog in the main page, 11.2 UI – interface for writing blogs, 11.3 Front-end api, 11.4 Front-end script, 11.5 Routers, 11.6 Back-end api coding, 11.7 Testing.

## Personal Contribution to Sprint 3

In sprint 3, I mainly completed four things. I write back-end code for article details, test and list all functions, write a report for the production environment section and add an account and some articles in database. In back-end code, interface url is “/articles/view/{id}”, request method is “post”, request parameter is “id(long)” which means article id (path parameter). It also returns json data. The general development process is similar to that in sprint 2. The main difference is that in ArtcileServiceImp, association queries are performed according to BodyId and CategoryId. Functions I list can be seen in appendix 3. The functions list helps us a lot in later coursework.

In this sprint, after I tested and listed all the functions, I actively communicated with the team members and skillfully completed my own code. I also found the relevant materials of markdown editor for the members, which was accepted and used by the members. The bad thing is that I didn't find all the bugs and I don't actively help the team members fix bugs after completing my own task.

## Conclusion

In short, this report mainly shows my personal perspective on our project and my individual work. In all three sprints, our team members are very united and work together towards the goal of "Leisure Town". Although the first sprint encountered some bottlenecks, we all succeeded in overcoming them and accelerated the progress in the subsequent sprint.

## Lesson Learnt and Future Work

In the future, we will fix the bugs left over from the first three sprints. These bugs include that Emoji cannot be displayed in comments, the page cannot be refreshed automatically after adding comments, the number of comments cannot be displayed correctly, the floor order of comments is inverted, and some pictures cannot be successfully updated in the article details. At the same time, add more functions to our software, such as classifying blogs according to categories or tags, trendy blogs, searching blogs through titles, viewing and changing personal information, liking blogs. Some of these have been completed in sprint 4. Based on the evaluation, I will speed up the development in the future to free up more time to help other members, learn more knowledge to develop new functions, find a person who specializes in testing and propose to the team members to assign tasks according to the technical level of the members.

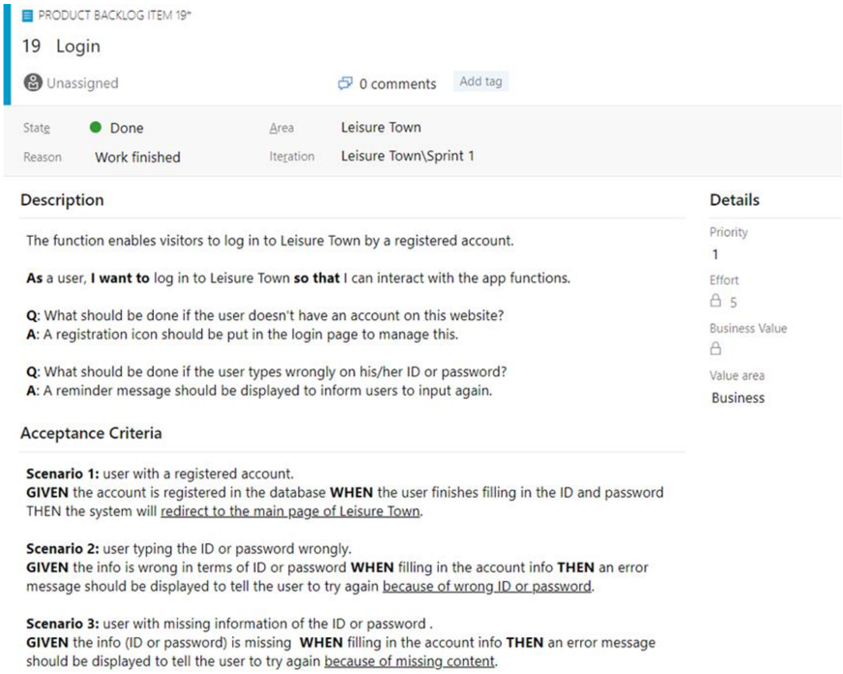
I learned many lessons. Some parts have been explained in personal contribution to sprint 1. In addition to these, I also learned how to use GitHub to share code, HTML page design and build back-end projects. I learned to use GitHub through class content, CSDN and BiliBili which shared by team members. HTML page design is that I find the source code of the login registration page, compare it with the generated page, and modify some code to see the effect. I searched on the Internet and compared a lot of materials and classroom content to learn. The construction of the project is also a key learning experience, because it is the most basic step and a troublesome step. I put forward many useful suggestions for project construction, such as looking for the relevant code of cross domain configuration on the official website. Through this team project, I learned the basic process of agile development, the importance of actively communicating with team members and increased my practical experience. At the same time, it also improved my programming thinking ability.

# Appendix

1. Markdown editor reference:
https://gitee.com/dsnull/mavonEditor/
https://blog.csdn.net/cnds123321/article/details/109112408

2. Details of PBIs:

#1 Login



PRODUCT BACKLOG ITEM 19*

19 Login

Unassigned 0 comments Add tag

| | | | |
|---|---|---|---|
| State | ● Done | Area | Leisure Town |
| Reason | Work finished | Iteration | Leisure Town\Sprint 1 |

**Description**

The function enables visitors to log in to Leisure Town by a registered account.

**As** a user, **I want to** log in to Leisure Town **so that** I can interact with the app functions.

**Q**: What should be done if the user doesn't have an account on this website?
**A**: A registration icon should be put in the login page to manage this.

**Q**: What should be done if the user types wrongly on his/her ID or password?
**A**: A reminder message should be displayed to inform users to input again.

**Acceptance Criteria**

**Scenario 1:** user with a registered account.
**GIVEN** the account is registered in the database **WHEN** the user finishes filling in the ID and password THEN the system will redirect to the main page of Leisure Town.

**Scenario 2:** user typing the ID or password wrongly.
**GIVEN** the info is wrong in terms of ID or password **WHEN** filling in the account info **THEN** an error message should be displayed to tell the user to try again because of wrong ID or password.

**Scenario 3:** user with missing information of the ID or password .
**GIVEN** the info (ID or password) is missing **WHEN** filling in the account info **THEN** an error message should be displayed to tell the user to try again because of missing content.

**Details**

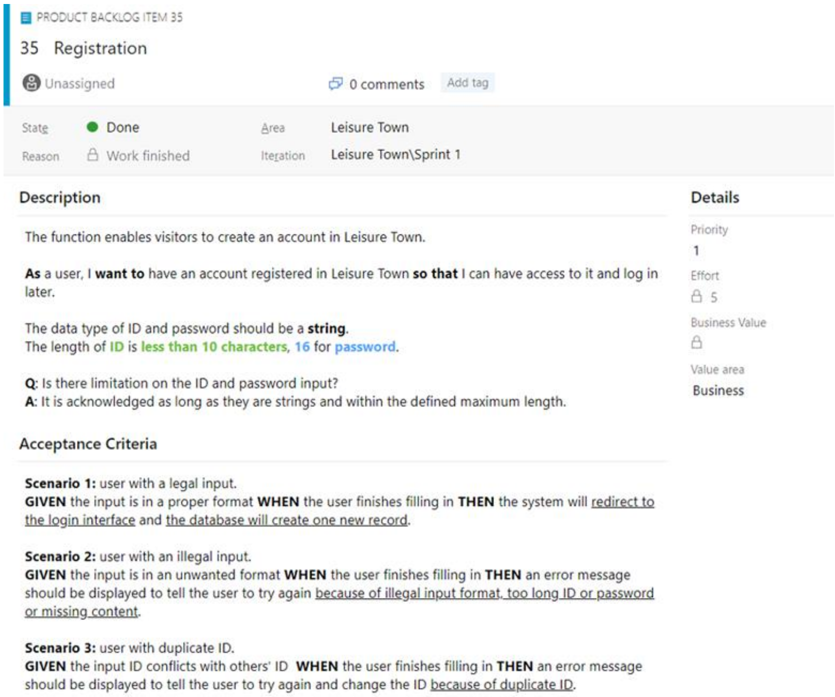
Priority
1

Effort
5

Business Value

Value area
Business

#2 Registration



PRODUCT BACKLOG ITEM 35

35 Registration

Unassigned 0 comments Add tag

| | | | |
|---|---|---|---|
| State | ● Done | Area | Leisure Town |
| Reason | Work finished | Iteration | Leisure Town\Sprint 1 |

**Description**

The function enables visitors to create an account in Leisure Town.

**As** a user, I **want to** have an account registered in Leisure Town **so that** I can have access to it and log in later.

The data type of ID and password should be a **string**.
The length of **ID** is **less than 10 characters**, **16** for **password**.

**Q**: Is there limitation on the ID and password input?
**A**: It is acknowledged as long as they are strings and within the defined maximum length.

**Acceptance Criteria**

**Scenario 1:** user with a legal input.
**GIVEN** the input is in a proper format **WHEN** the user finishes filling in **THEN** the system will redirect to the login interface and the database will create one new record.

**Scenario 2:** user with an illegal input.
**GIVEN** the input is in an unwanted format **WHEN** the user finishes filling in **THEN** an error message should be displayed to tell the user to try again because of illegal input format, too long ID or password or missing content.

**Scenario 3:** user with duplicate ID.
**GIVEN** the input ID conflicts with others' ID **WHEN** the user finishes filling in **THEN** an error message should be displayed to tell the user to try again and change the ID because of duplicate ID.

**Details**

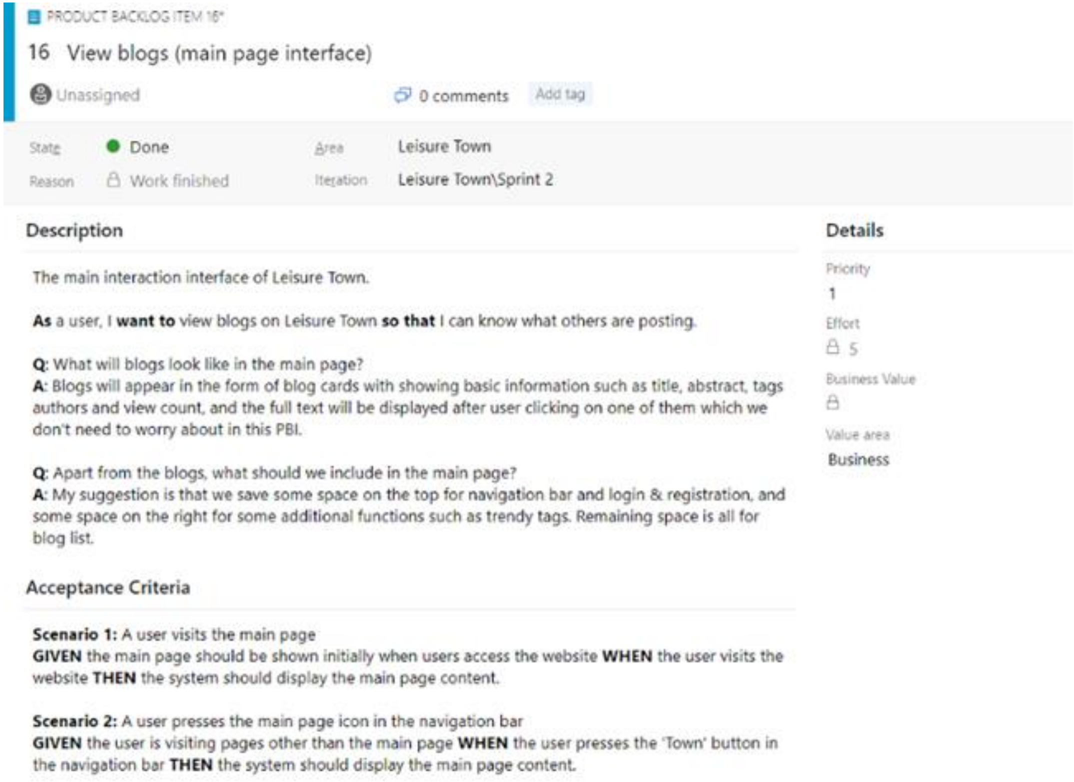
Priority
1

Effort
5

Business Value

Value area
Business

# #3 View blogs (main page interface)



PRODUCT BACKLOG ITEM 16*

## 16 View blogs (main page interface)

Unassigned

0 comments

| | | | |
|---|---|---|---|
| State | ● Done | Area | Leisure Town |
| Reason | Work finished | Iteration | Leisure Town\Sprint 2 |

### Description

The main interaction interface of Leisure Town.

**As** a user, I **want to** view blogs on Leisure Town **so that** I can know what others are posting.

**Q**: What will blogs look like in the main page?
**A**: Blogs will appear in the form of blog cards with showing basic information such as title, abstract, tags authors and view count, and the full text will be displayed after user clicking on one of them which we don't need to worry about in this PBI.

**Q**: Apart from the blogs, what should we include in the main page?
**A**: My suggestion is that we save some space on the top for navigation bar and login & registration, and some space on the right for some additional functions such as trendy tags. Remaining space is all for blog list.

### Acceptance Criteria

**Scenario 1:** A user visits the main page
**GIVEN** the main page should be shown initially when users access the website **WHEN** the user visits the website **THEN** the system should display the main page content.

**Scenario 2:** A user presses the main page icon in the navigation bar
**GIVEN** the user is visiting pages other than the main page **WHEN** the user presses the 'Town' button in the navigation bar **THEN** the system should display the main page content.

### Details

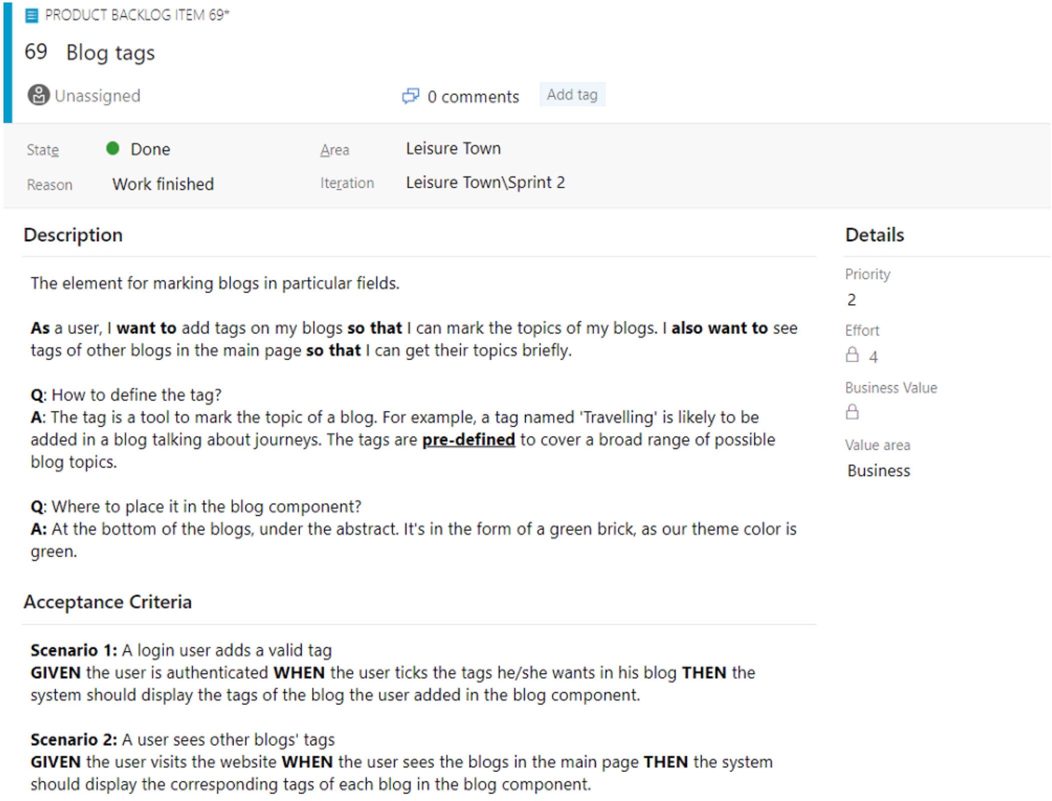
Priority
1

Effort
5

Business Value

Value area
Business

# #4 Blog tags



PRODUCT BACKLOG ITEM 69*

## 69 Blog tags

Unassigned

0 comments

| | | | |
|---|---|---|---|
| State | ● Done | Area | Leisure Town |
| Reason | Work finished | Iteration | Leisure Town\Sprint 2 |

### Description

The element for marking blogs in particular fields.

**As** a user, I **want to** add tags on my blogs **so that** I can mark the topics of my blogs. I **also want to** see tags of other blogs in the main page **so that** I can get their topics briefly.

**Q**: How to define the tag?
**A**: The tag is a tool to mark the topic of a blog. For example, a tag named 'Travelling' is likely to be added in a blog talking about journeys. The tags are **pre-defined** to cover a broad range of possible blog topics.

**Q**: Where to place it in the blog component?
**A:** At the bottom of the blogs, under the abstract. It's in the form of a green brick, as our theme color is green.

### Acceptance Criteria

**Scenario 1:** A login user adds a valid tag
**GIVEN** the user is authenticated **WHEN** the user ticks the tags he/she wants in his blog **THEN** the system should display the tags of the blog the user added in the blog component.

**Scenario 2:** A user sees other blogs' tags
**GIVEN** the user visits the website **WHEN** the user sees the blogs in the main page **THEN** the system should display the corresponding tags of each blog in the blog component.

### Details

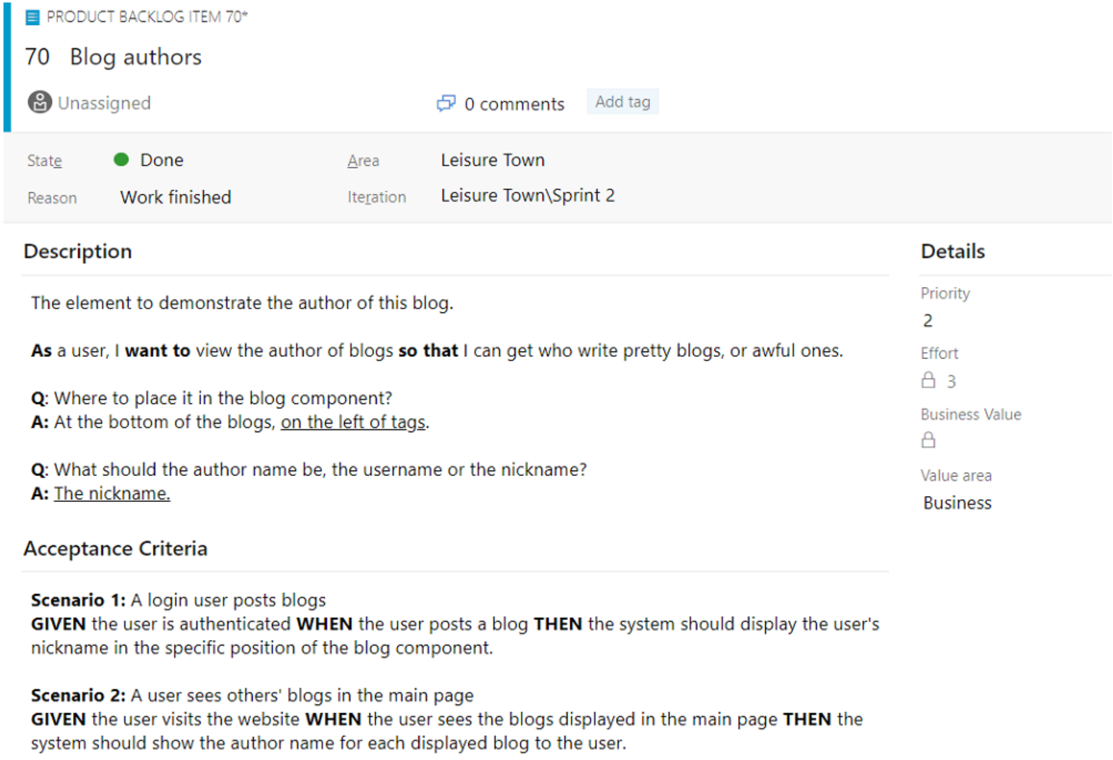
Priority
2

Effort
4

Business Value

Value area
Business

## #5 Blog authors



PRODUCT BACKLOG ITEM 70*

70 Blog authors

Unassigned

0 comments

Add tag

| | | | |
|---|---|---|---|
| State | ● Done | Area | Leisure Town |
| Reason | Work finished | Iteration | Leisure Town\Sprint 2 |

**Description**

The element to demonstrate the author of this blog.

**As** a user, I **want to** view the author of blogs **so that** I can get who write pretty blogs, or awful ones.

**Q**: Where to place it in the blog component?
**A:** At the bottom of the blogs, on the left of tags.

**Q**: What should the author name be, the username or the nickname?
**A:** The nickname.

**Acceptance Criteria**

**Scenario 1:** A login user posts blogs
**GIVEN** the user is authenticated **WHEN** the user posts a blog **THEN** the system should display the user's nickname in the specific position of the blog component.

**Scenario 2:** A user sees others' blogs in the main page
**GIVEN** the user visits the website **WHEN** the user sees the blogs displayed in the main page **THEN** the system should show the author name for each displayed blog to the user.

**Details**

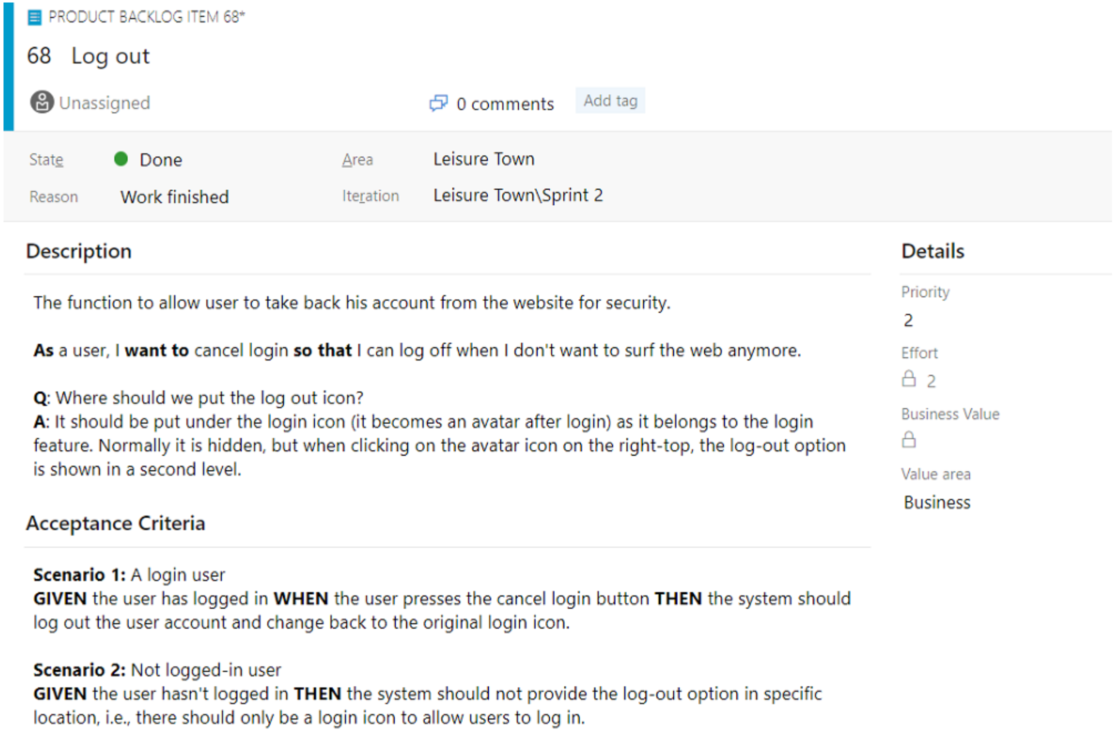
Priority
2

Effort
🔒 3

Business Value
🔒

Value area
Business

## #6 Log out



PRODUCT BACKLOG ITEM 68*

68 Log out

Unassigned

0 comments

Add tag

| | | | |
|---|---|---|---|
| State | ● Done | Area | Leisure Town |
| Reason | Work finished | Iteration | Leisure Town\Sprint 2 |

**Description**

The function to allow user to take back his account from the website for security.

**As** a user, I **want to** cancel login **so that** I can log off when I don't want to surf the web anymore.

**Q**: Where should we put the log out icon?
**A**: It should be put under the login icon (it becomes an avatar after login) as it belongs to the login feature. Normally it is hidden, but when clicking on the avatar icon on the right-top, the log-out option is shown in a second level.

**Acceptance Criteria**

**Scenario 1:** A login user
**GIVEN** the user has logged in **WHEN** the user presses the cancel login button **THEN** the system should log out the user account and change back to the original login icon.

**Scenario 2:** Not logged-in user
**GIVEN** the user hasn't logged in **THEN** the system should not provide the log-out option in specific location, i.e., there should only be a login icon to allow users to log in.

**Details**

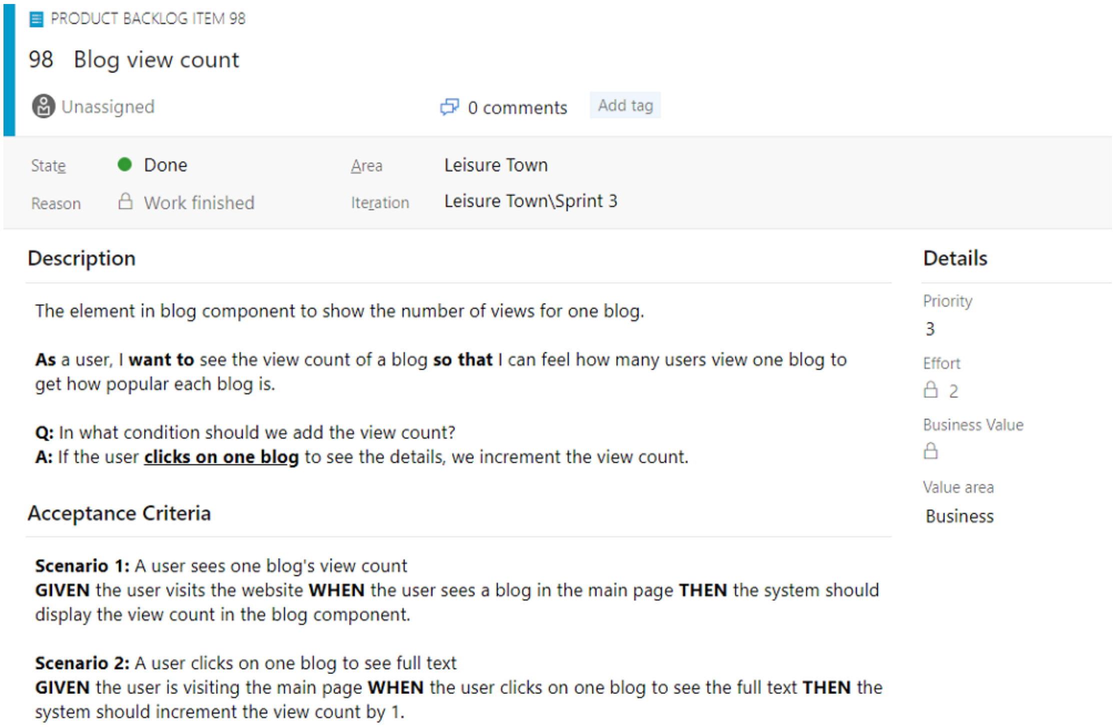
Priority
2

Effort
🔒 2

Business Value
🔒

Value area
Business

## #7 Blog view count



PRODUCT BACKLOG ITEM 98

98 Blog view count

Unassigned

0 comments

Add tag

| | | | |
|---|---|---|---|
| State | ● Done | Area | Leisure Town |
| Reason | Work finished | Iteration | Leisure Town\Sprint 3 |

**Description**

The element in blog component to show the number of views for one blog.

**As** a user, I **want to** see the view count of a blog **so that** I can feel how many users view one blog to get how popular each blog is.

**Q:** In what condition should we add the view count?
**A:** If the user **clicks on one blog** to see the details, we increment the view count.

**Acceptance Criteria**

**Scenario 1:** A user sees one blog's view count
**GIVEN** the user visits the website **WHEN** the user sees a blog in the main page **THEN** the system should display the view count in the blog component.

**Scenario 2:** A user clicks on one blog to see full text
**GIVEN** the user is visiting the main page **WHEN** the user clicks on one blog to see the full text **THEN** the system should increment the view count by 1.

**Details**

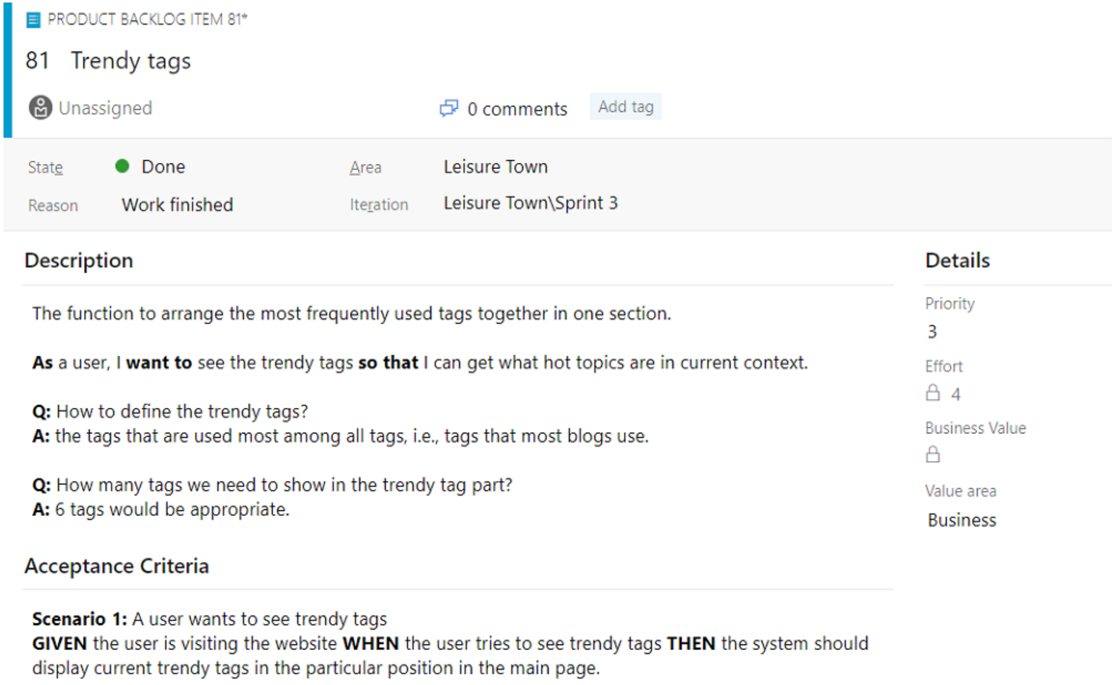
Priority
3

Effort
2

Business Value

Value area
Business

# #8 Trendy tags



PRODUCT BACKLOG ITEM 81*

81 Trendy tags

Unassigned

0 comments

Add tag

| | | | |
|---|---|---|---|
| State | ● Done | Area | Leisure Town |
| Reason | Work finished | Iteration | Leisure Town\Sprint 3 |

**Description**

The function to arrange the most frequently used tags together in one section.

**As** a user, I **want to** see the trendy tags **so that** I can get what hot topics are in current context.

**Q:** How to define the trendy tags?
**A:** the tags that are used most among all tags, i.e., tags that most blogs use.

**Q:** How many tags we need to show in the trendy tag part?
**A:** 6 tags would be appropriate.

**Acceptance Criteria**

**Scenario 1:** A user wants to see trendy tags
**GIVEN** the user is visiting the website **WHEN** the user tries to see trendy tags **THEN** the system should display current trendy tags in the particular position in the main page.

**Details**

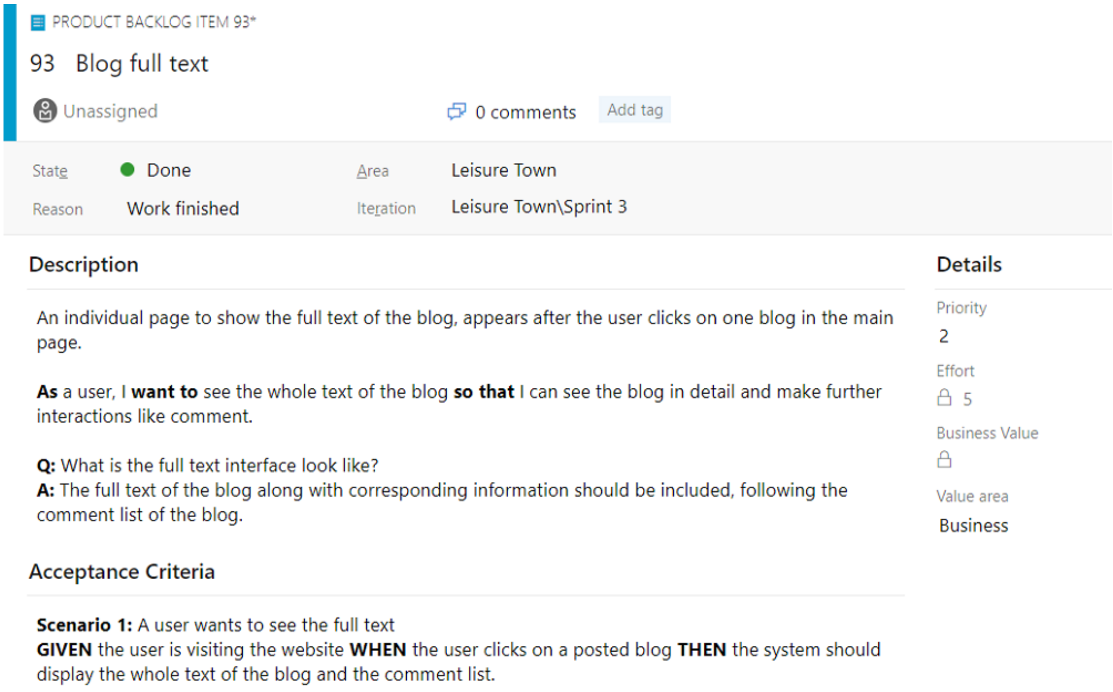
Priority
3

Effort
4

Business Value

Value area
Business

# #9 Blog full text



93 Blog full text

| | | | |
|---|---|---|---|
| State | ● Done | Area | Leisure Town |
| Reason | Work finished | Iteration | Leisure Town\Sprint 3 |

**Description**

An individual page to show the full text of the blog, appears after the user clicks on one blog in the main page.

**As** a user, I **want to** see the whole text of the blog **so that** I can see the blog in detail and make further interactions like comment.

**Q:** What is the full text interface look like?
**A:** The full text of the blog along with corresponding information should be included, following the comment list of the blog.

**Acceptance Criteria**

**Scenario 1:** A user wants to see the full text
**GIVEN** the user is visiting the website **WHEN** the user clicks on a posted blog **THEN** the system should display the whole text of the blog and the comment list.

**Details**

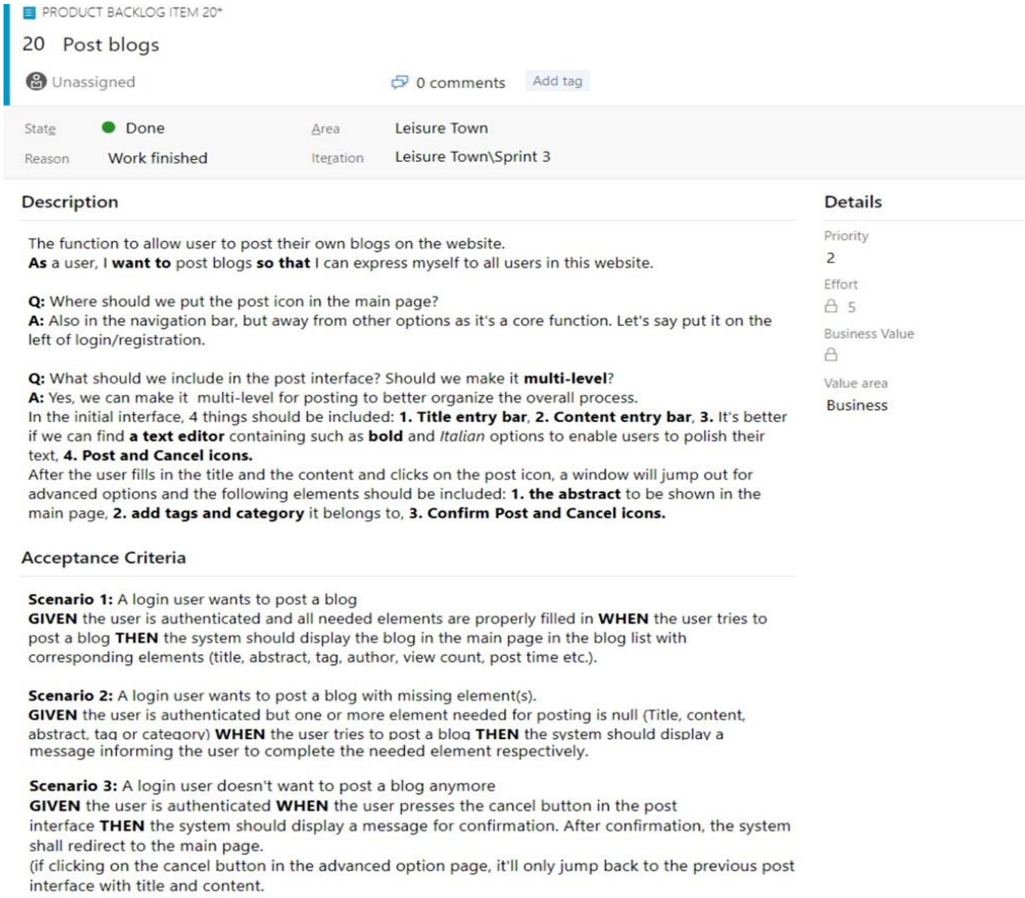
Priority
2

Effort
5

Business Value

Value area
Business

#10 Comment



20 Post blogs

| | | | |
|---|---|---|---|
| State | ● Done | Area | Leisure Town |
| Reason | Work finished | Iteration | Leisure Town\Sprint 3 |

**Description**

The function to allow user to post their own blogs on the website.
**As** a user, I **want to** post blogs **so that** I can express myself to all users in this website.

**Q:** Where should we put the post icon in the main page?
**A:** Also in the navigation bar, but away from other options as it's a core function. Let's say put it on the left of login/registration.

**Q:** What should we include in the post interface? Should we make it **multi-level**?
**A:** Yes, we can make it multi-level for posting to better organize the overall process.
In the initial interface, 4 things should be included: **1. Title entry bar**, **2. Content entry bar**, **3.** It's better if we can find **a text editor** containing such as **bold** and *Italian* options to enable users to polish their text, **4. Post and Cancel icons.**
After the user fills in the title and the content and clicks on the post icon, a window will jump out for advanced options and the following elements should be included: **1. the abstract** to be shown in the main page, **2. add tags and category** it belongs to, **3. Confirm Post and Cancel icons.**

**Acceptance Criteria**

**Scenario 1:** A login user wants to post a blog
**GIVEN** the user is authenticated and all needed elements are properly filled in **WHEN** the user tries to post a blog **THEN** the system should display the blog in the main page in the blog list with corresponding elements (title, abstract, tag, author, view count, post time etc.).

**Scenario 2:** A login user wants to post a blog with missing element(s).
**GIVEN** the user is authenticated but one or more element needed for posting is null (Title, content, abstract, tag or category) **WHEN** the user tries to post a blog **THEN** the system should display a message informing the user to complete the needed element respectively.

**Scenario 3:** A login user doesn't want to post a blog anymore
**GIVEN** the user is authenticated **WHEN** the user presses the cancel button in the post interface **THEN** the system should display a message for confirmation. After confirmation, the system shall redirect to the main page.
(if clicking on the cancel button in the advanced option page, it'll only jump back to the previous post interface with title and content.

**Details**

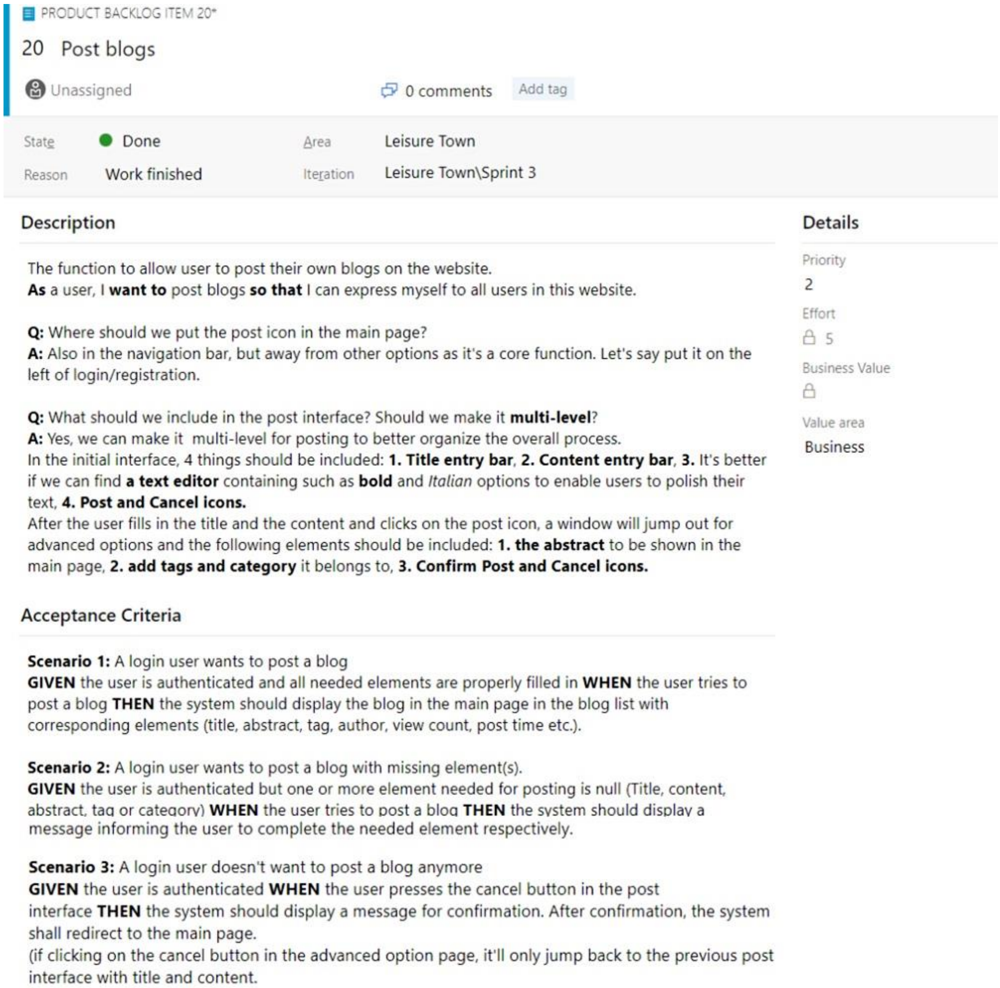
Priority
2

Effort
5

Business Value

Value area
Business

#11 Post blogs



PRODUCT BACKLOG ITEM 20*

20 Post blogs

Unassigned

0 comments Add tag

State ● Done

Reason Work finished

Area Leisure Town

Iteration Leisure Town\Sprint 3

**Description**

The function to allow user to post their own blogs on the website.
**As** a user, I **want to** post blogs **so that** I can express myself to all users in this website.

**Q:** Where should we put the post icon in the main page?
**A:** Also in the navigation bar, but away from other options as it's a core function. Let's say put it on the left of login/registration.

**Q:** What should we include in the post interface? Should we make it **multi-level**?
**A:** Yes, we can make it multi-level for posting to better organize the overall process.
In the initial interface, 4 things should be included: **1. Title entry bar**, **2. Content entry bar**, **3.** It's better if we can find **a text editor** containing such as **bold** and *Italian* options to enable users to polish their text, **4. Post and Cancel icons.**
After the user fills in the title and the content and clicks on the post icon, a window will jump out for advanced options and the following elements should be included: **1. the abstract** to be shown in the main page, **2. add tags and category** it belongs to, **3. Confirm Post and Cancel icons.**

**Acceptance Criteria**

**Scenario 1:** A login user wants to post a blog
**GIVEN** the user is authenticated and all needed elements are properly filled in **WHEN** the user tries to post a blog **THEN** the system should display the blog in the main page in the blog list with corresponding elements (title, abstract, tag, author, view count, post time etc.).

**Scenario 2:** A login user wants to post a blog with missing element(s).
**GIVEN** the user is authenticated but one or more element needed for posting is null (Title, content, abstract, tag or category) **WHEN** the user tries to post a blog **THEN** the system should display a message informing the user to complete the needed element respectively.

**Scenario 3:** A login user doesn't want to post a blog anymore
**GIVEN** the user is authenticated **WHEN** the user presses the cancel button in the post interface **THEN** the system should display a message for confirmation. After confirmation, the system shall redirect to the main page.
(if clicking on the cancel button in the advanced option page, it'll only jump back to the previous post interface with title and content.

**Details**

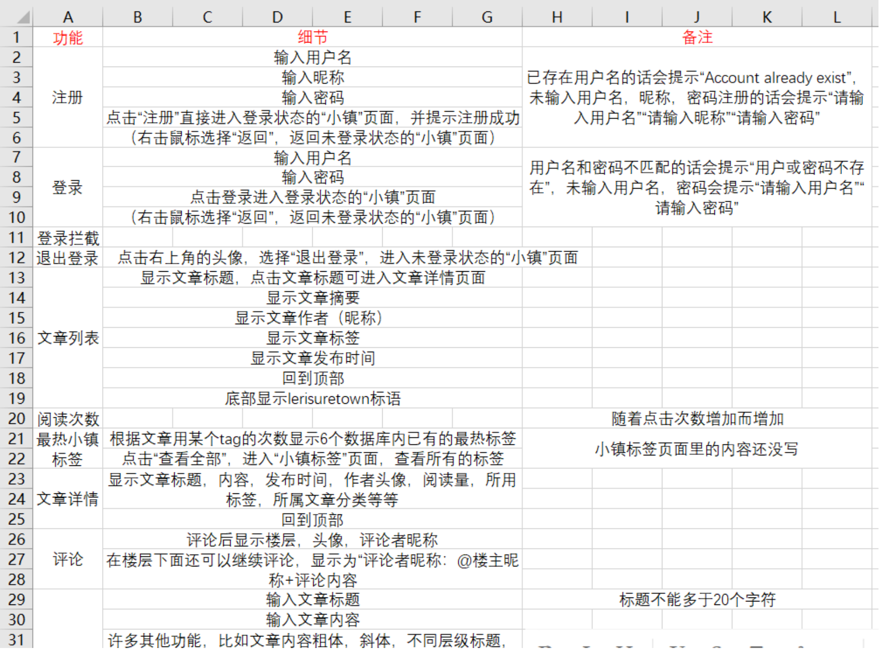
Priority
2

Effort
5

Business Value

Value area
Business

3. Part of functions I list:



| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 功能 | 细节 | | | | | | 备注 | | | | |
| 2 | 注册 | 输入用户名 | | | | | | 已存在用户名的话会提示“Account already exist”，未输入用户名，昵称，密码注册的话会提示“请输入用户名”“请输入昵称”“请输入密码” | | | | |
| 3 | | 输入昵称 | | | | | | | | | | |
| 4 | | 输入密码 | | | | | | | | | | |
| 5 | | 点击“注册”直接进入登录状态的“小镇”页面，并提示注册成功 | | | | | | | | | | |
| 6 | | （右击鼠标选择“返回”，返回未登录状态的“小镇”页面） | | | | | | | | | | |
| 7 | 登录 | 输入用户名 | | | | | | 用户名和密码不匹配的话会提示“用户或密码不存在”，未输入用户名，密码会提示“请输入用户名”“请输入密码” | | | | |
| 8 | | 输入密码 | | | | | | | | | | |
| 9 | | 点击登录进入登录状态的“小镇”页面 | | | | | | | | | | |
| 10 | | （右击鼠标选择“返回”，返回未登录状态的“小镇”页面） | | | | | | | | | | |
| 11 | 登录拦截 | | | | | | | | | | | |
| 12 | 退出登录 | 点击右上角的头像，选择“退出登录”，进入未登录状态的“小镇”页面 | | | | | | | | | | |
| 13 | 文章列表 | 显示文章标题，点击文章标题可进入文章详情页面 | | | | | | | | | | |
| 14 | | 显示文章摘要 | | | | | | | | | | |
| 15 | | 显示文章作者（昵称） | | | | | | | | | | |
| 16 | | 显示文章标签 | | | | | | | | | | |
| 17 | | 显示文章发布时间 | | | | | | | | | | |
| 18 | | 回到顶部 | | | | | | | | | | |
| 19 | | 底部显示lerisuretown标语 | | | | | | | | | | |
| 20 | 阅读次数 | | | | | | | 随着点击次数增加而增加 | | | | |
| 21 | 最热小镇标签 | 根据文章用某个tag的次数显示6个数据库内已有的最热标签 | | | | | | 小镇标签页面里的内容还没写 | | | | |
| 22 | | 点击“查看全部”，进入“小镇标签”页面，查看所有的标签 | | | | | | | | | | |
| 23 | 文章详情 | 显示文章标题，内容，发布时间，作者头像，阅读量，所用标签，所属文章分类等等 | | | | | | | | | | |
| 24 | | | | | | | | | | | | |
| 25 | | 回到顶部 | | | | | | | | | | |
| 26 | 评论 | 评论后显示楼层，头像，评论者昵称 | | | | | | | | | | |
| 27 | | 在楼层下面还可以继续评论，显示为“评论者昵称：@楼主昵称+评论内容 | | | | | | | | | | |
| 28 | | | | | | | | | | | | |
| 29 | | 输入文章标题 | | | | | | 标题不能多于20个字符 | | | | |
| 30 | | 输入文章内容 | | | | | | | | | | |
| 31 | | 许多其他功能，比如文章内容粗体，斜体，不同层级标题， | | | | | | | | | | |