



Documentation: Release v1 - Beastopia

Product-Owner: Sylvan Giese
Scrum-Master: Ali Kobeissi
Developer: Armin Kozik
Celina Werkmeister
Leon Gierschner
Luis Schrader
Jonas Graunitz

Date: May 23, 2023

Contents

1	Introduction	4
2	Implementation	5
2.1	Comparison Wireframes and Screenshots	5
2.1.1	Login	5
2.1.2	Registration	8
2.1.3	Main Menu	11
2.1.4	Ingame	15
2.1.5	Pause Menu	16
2.1.6	Edit Profile	17
2.1.7	Direct Messages	21
2.1.8	Create Group	23
2.1.9	Edit Group	24
2.2	Technical Implementation	25
2.2.1	Model-View-Controller Pattern	25
2.2.2	Third Party Libraries	26
3	Project documentation	29
3.1	Scrum	29
3.1.1	Scrum structure	29
3.2	First sprint	30
3.2.1	Jira stories	30
3.2.2	Additional Jira Tasks and Bugs	32
3.2.3	Sprint retrospective	33
3.3	Second sprint	36
3.3.1	Jira stories	36
3.3.2	Additional Jira Tasks and Bugs	39
3.3.3	Sprint retrospective	40
3.4	Lessons learned	42
3.5	Story overview	43
4	Appendix	44
4.1	Comparison Summer and Dark theme	44
4.2	Acronyms	47

List of Figures

1	Login Wireframe	5
2	Login Screen Implementation	6
3	Login Screen when the inputs do not meet the requirements . . .	7
4	Login Screen when the user may log in	7
5	Login Screen if username or password is wrong	8
6	Registration Wireframe	8
7	Registration Screen	9
8	Registration Screen if inputs are too short	10
9	Registration Screen when username is already taken	10
10	Registration Screen when successfully registered	11
11	Main Menu Wireframe	11
12	Main Menu Implementation	12
13	User Search Wireframe	13
14	User Search Implementation	13
15	Friends on the users friend list	14
16	Ingame Wireframe	15
17	Ingame Implementation	15
18	Pause Menu Wireframe	16
19	Pause Menu Implementation	17
20	Edit Profile Wireframe	17
21	Edit Profile Implementation	18
22	Editing one's password	19
23	Editing language and theme	19
24	Delete User Implementation	20
25	Direct Messages Wireframe	21
26	Direct Messages User Wireframe	22
27	Direct Messages Implementation	22
28	Create Group Wireframe	23
29	Edit Group Wireframe	24
30	Modified Model-View-Controller (MVC) pattern	25
31	Burndown chart first sprint	35
32	Burndown chart second sprint	42

List of Tables

1	Unfinished tasks and stories in first sprint	33
2	Unfinished tasks and stories in second sprint	40
3	Overview of estimated and required time for every story	43

1 Introduction

This documentation reports the state of development of Beastopia. Beastopia is a multiplayer top-down 2D adventure Role-Play-Game (RPG). As software company LUMNIX, we are developing Beastopia, whereby the project is commissioned by Dead Bird Society.

LUMNIX is a freshly founded small start up company, that specialises in building gaming software. The team consists of seven people of which all are related to computer science. The team members differ from each other regarding skills what leads to an overall broader skill set. Our project management utilizes the Scrum framework to work efficient and agile. That enables us to react to changes and improve quality consistently through reflection and by defining lessons learned.

The software project Beastopia aims to deliver a multiplayer top-down 2D adventure RPG. The game is an Open World game where monsters can be caught and used to battle with other players in a turn-based fighting mechanic. Players can join a region with other players and go on a journey, either alone or together with friends.

Caught or encountered monsters will be tracked enabling quantifiable metrics e.g. monster catch or seen completion. Monsters are categorized as different types, making them more or less effective in fights, e.g. a water monster will damage a fire monster more than a grass monster, which gives fights a strategic dimension. Furthermore, items deepen this strategic dimension, enabling players to capitalize on using them at the right moment.

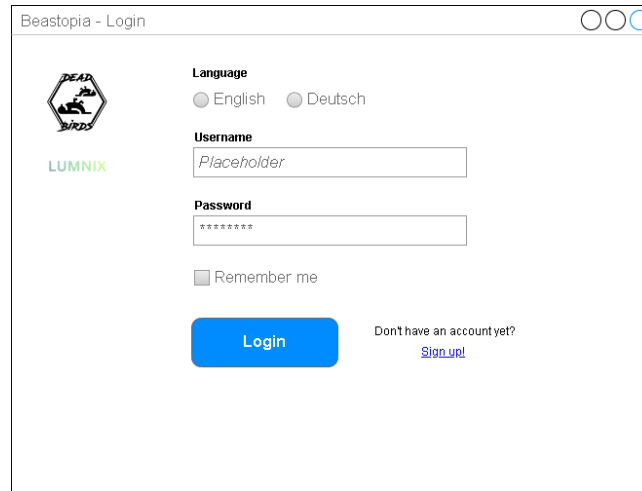
The world of Beastopia is filled with visitable spots, Non-Playable-Character (NPC's) and free roaming monsters, giving rise to an immersive world in which players can dive in and enjoy the experience.

2 Implementation

This chapter will elaborate the idea of the product through wireframes. The shown elements of the wireframes are put in contrast with the actual implementation. Moreover the user interaction and the implemented error cases will be outlined. For simplicity, the screenshots show their dark theme counterpart, being the summer theme. A direct comparison between both themes can be seen in chapter 4.1. Since English is the main language of the application, the screenshots of this chapter show the application in English.

2.1 Comparison Wireframes and Screenshots

2.1.1 Login



The wireframe shows a login window titled "Beastopia - Login". On the left is a logo with a hexagon containing a silhouette of a person and the text "DEAD" above and "BIRDS" below, with "LUMNIX" in green below it. To the right, there is a "Language" section with radio buttons for "English" (selected) and "Deutsch". Below this are input fields for "Username" (with a "Placeholder" text) and "Password" (with "*****" text). A "Remember me" checkbox is below the password field. At the bottom left is a blue "Login" button. To its right is the text "Don't have an account yet?" followed by a blue "Sign up!" link. The window has standard OS window controls (minimize, maximize, close) in the top right corner.

Figure 1: Login Wireframe

Upon opening the application on a device, the initial screen presented to the user is the login screen. In order to log in, the user is prompted to input their username and password in the respective fields provided. Additionally, the user has the option to select their preferred language for the entire application, with the available options being English and German. To expedite future login attempts, the user can utilize the Remember me tickbox to store their login credentials. All interface elements are clearly labeled and stacked on top of each other, as depicted in Figure 1.

The login button is situated below the Remember me tickbox and to the left of the Sign up link, and upon clicking, it validates the user's input and logs them into the application. For new users without an account, the Sign up link provides a quick registration process.

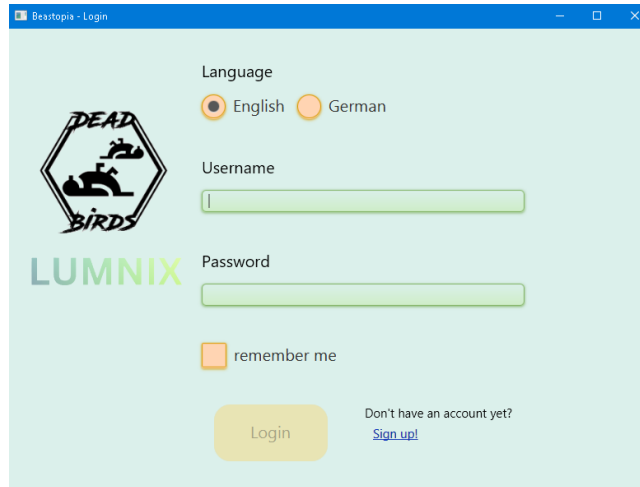
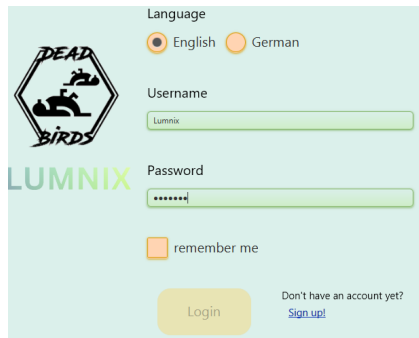



Figure 2: Login Screen Implementation

The present chapter showcases several implementation screenshots that are colored in the primary theme of the application, being a summer palette. There are two available designs for the app: the Summer theme and the Dark theme. The Summer theme reflects the colors of nature during summer and features a greenish background, orange radio buttons and tick boxes, greenish gradients for text fields, password fields and submenus, yellow buttons, and shadow effects to enhance depth perception. The Dark theme, as the counterpart, depicts elements in a darker tone.

The Login Screen implementation shown in Figure 2 sticks to the previously described wireframe. Users may select their preferred language via the English or German radio buttons. English is the main language to comply with internationalization standards. The Sign up hyperlink appears in a lighter color, and the Login button is grayed out until input is entered in the Username and Password fields. Further details about the Login implementation can be found in the following screenshots.


 A login form for 'DEAD BIRDS LUMNIX'. It includes a language selector (English selected, German unselected), a username field containing 'Lumix', and a password field containing '*****'. Below the password field is a 'remember me' checkbox which is unchecked. At the bottom is a yellow 'Login' button and a link 'Don't have an account yet? Sign up!'.

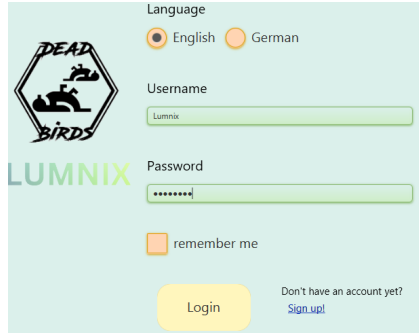
(a) Login Screen when input password too short


 A login form for 'DEAD BIRDS LUMNIX'. It includes a language selector (English selected, German unselected), an empty username field, and an empty password field. Below the password field is a 'remember me' checkbox which is unchecked. At the bottom is a yellow 'Login' button and a link 'Don't have an account yet? Sign up!'.

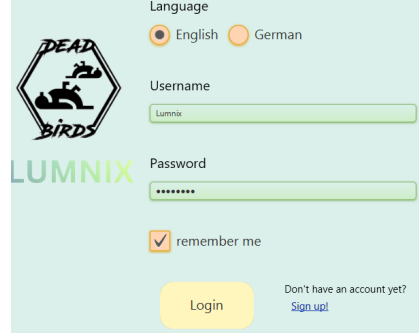
(b) Login Screen when no username entered

Figure 3: Login Screen when the inputs do not meet the requirements

As previously described in the implementation of the Login screen, displayed in Figure 2, it was mentioned that the Login button is rendered in a grayed-out state. This behavior occurs for two possible reasons: either the password input from the user is considered too short, as illustrated in Figure 3a, or the user has left the Username text field empty, as depicted in Figure 3b.


 A login form for 'DEAD BIRDS LUMNIX'. It includes a language selector (English selected, German unselected), a username field containing 'Lumix', and a password field containing '*****'. Below the password field is a 'remember me' checkbox which is unchecked. At the bottom is a yellow 'Login' button and a link 'Don't have an account yet? Sign up!'.

(a) Requirements for input match


 A login form for 'DEAD BIRDS LUMNIX'. It includes a language selector (English selected, German unselected), a username field containing 'Lumix', and a password field containing '*****'. Below the password field is a 'remember me' checkbox which is checked. At the bottom is a yellow 'Login' button and a link 'Don't have an account yet? Sign up!'.

(b) Selection of "Remember me"

Figure 4: Login Screen when the user may log in

Once the login form requirements are met, the Login button becomes activated, indicating that the necessary data has been filled in correctly, as shown in Figure 4a. In addition, Figure 4b illustrates a Remember me checkbox, which allows the user to store their login credentials for subsequent use, thereby reducing the time required for future logins and enhancing the overall user experience. Following the activation of the Login button, the validation process begins to ensure the accuracy of the entered data and to enable successful login.

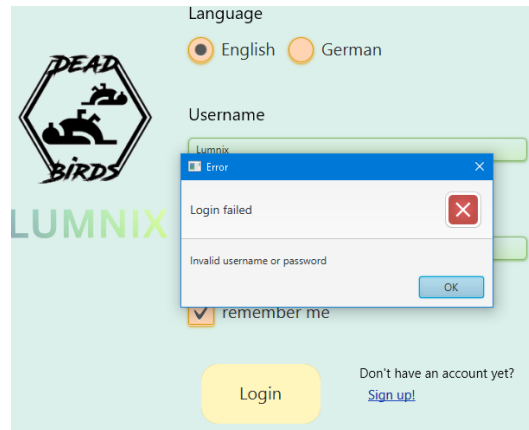


Figure 5: Login Screen if username or password is wrong

If the input data provided by the user doesn't match any account data stored on the server, a popup box is displayed indicating that the login attempt failed due to an invalid username or password, as presented in Figure 5. However, if the login credentials match the data stored on the server, the Main Menu is displayed by the application.

2.1.2 Registration

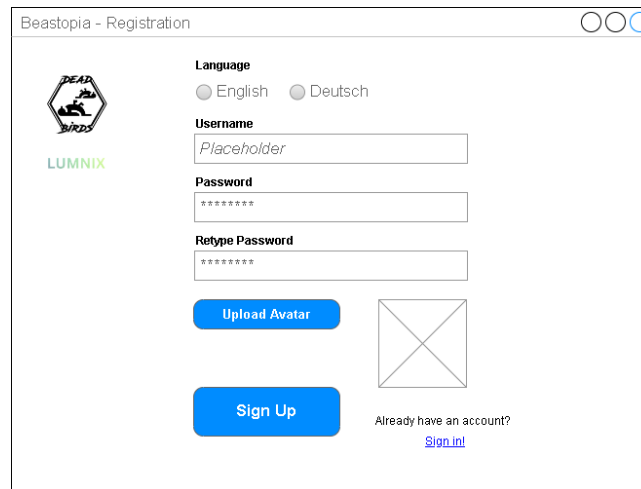


Figure 6: Registration Wireframe

The registration wireframe follows a design similar to the login wireframe with the items being labeled and stacked on top of each other. The user must provide

a unique username, a secure password, and select a preferred language for the application. In addition, the user may personalize their profile by selecting an avatar as depicted in Figure 6. As a security measure, the user must re-enter their password. Once all inputs have been provided, the user may create their account by clicking on the Sign Up button. The input data is validated and if successful, the account is created.

If the user already has an account, they may switch back to the login screen using the Sign in link.

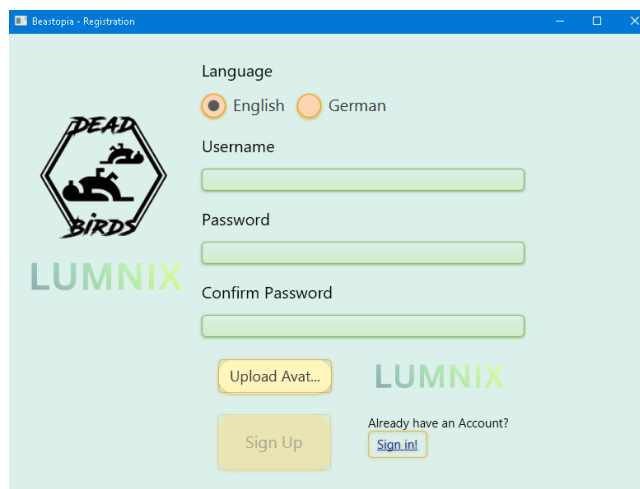
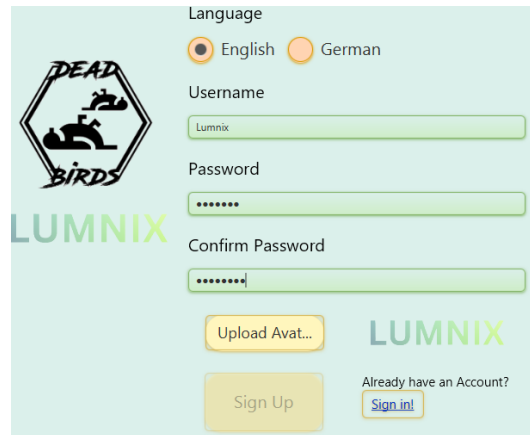
The image shows a web browser window titled "Beartopia - Registration". The background is a light teal color. On the left side, there is a logo consisting of a black hexagon with a white silhouette of a bear and the words "DEAD" and "BIRDS" in a stylized font. Below the logo, the word "LUMNIX" is written in a green, sans-serif font. On the right side, there is a registration form. It starts with a "Language" section with two radio buttons: "English" (selected) and "German". Below this are three input fields: "Username", "Password", and "Confirm Password". Each input field has a green border. Below the "Password" field is a yellow button labeled "Upload Avat...". Below the "Confirm Password" field is a yellow button labeled "Sign Up". To the right of the "Sign Up" button, there is a link that says "Already have an Account?" followed by a blue "Sign in" link. The word "LUMNIX" is also written in green next to the "Sign in" link.

Figure 7: Registration Screen

Once the Sign up hyperlink is clicked on the Login screen, the user is redirected to the registration form as presented in Figure 7. The registration form is designed similarly to the login form, however, it offers the additional functionality of uploading a custom avatar for the user's profile.

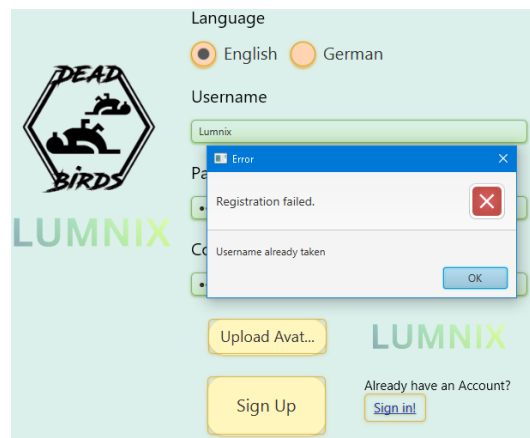


The image shows a registration form for 'LUMNIX'. On the left is a logo with a bird silhouette and the text 'DEAD BIRDS' and 'LUMNIX'. The form includes a 'Language' section with radio buttons for 'English' (selected) and 'German'. Below are input fields for 'Username' (containing 'Lumnix'), 'Password' (filled with dots), and 'Confirm Password' (also filled with dots). There is an 'Upload Avat...' button and a 'Sign Up' button which is disabled. To the right of the 'Sign Up' button is a link 'Sign in!' preceded by the text 'Already have an Account?'.

Figure 8: Registration Screen if inputs are too short

Figure 8 illustrates that the Sign Up button is disabled due to insufficient input length, an empty Username field, or an excessively long Username. The user is thus unable to register for an account until these conditions are met. A similar constraint applies when the entered passwords do not match.

However, when the input requirements are met, the Sign Up button becomes active, allowing the user to proceed with the account creation process.



This image shows the same registration form as Figure 8, but with an error dialog box overlaid. The dialog box has a title bar 'Error' and a close button. It contains the text 'Registration failed.' and 'Username already taken' with a red 'X' icon. An 'OK' button is at the bottom right of the dialog. The background form is partially obscured but the 'Sign Up' button remains disabled.

Figure 9: Registration Screen when username is already taken

In the scenario depicted in Figure 9, when the chosen username is already in use, the registration process encounters an error and a dialog box is displayed to notify the user of the failure. It is important to note that the input fields will retain their values even after the user acknowledges the error by clicking the

OK button in the dialog box. This allows the user to easily make modifications to the existing input without having to re-enter the entire registration form.

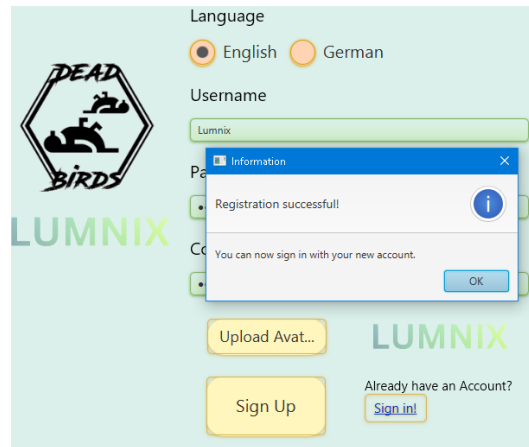


Figure 10: Registration Screen when successfully registered

If the registration process meets all the required conditions, a dialog box will be displayed by the application, as illustrated in Figure 10, to inform the user of the successful completion of the process. Subsequently, the user will be redirected to the Login Screen without any further action.

2.1.3 Main Menu

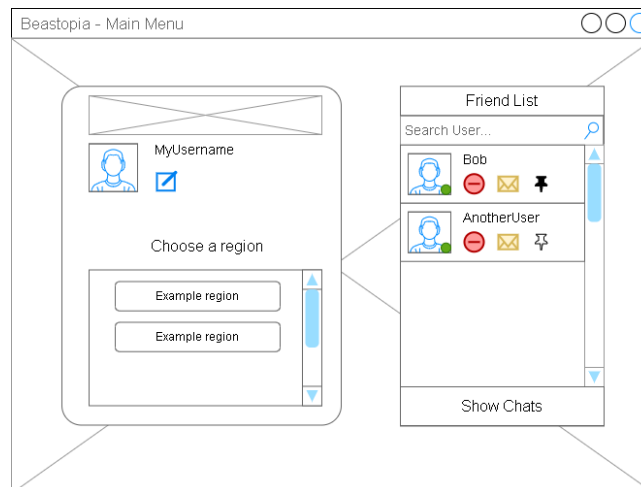


Figure 11: Main Menu Wireframe

The Main Menu serves as the primary interface for accessing the application's features. A placeholder for a banner, a personalized avatar, and the user's name are displayed on the left side of the screen, as illustrated in Figure 11. The Profile Edit Menu can be accessed via an edit icon to adjust profile information, and a list of regions is provided for the user to join and start the game.

On the right side of the screen, the Friend List submenu can be found, containing a search bar for users registered on the server, a list of friends currently added by the user, and corresponding friend cards. Each friend card exhibits the friend's avatar, an online/offline status icon, a button for removing the friend from the list, and a button for initiating a chat message. Moreover, the option to pin a friend card to the top of the list is available, signifying a "best friend" behavior for swift interaction.

Lastly, a button located at the bottom of the friend list provides access to the Direct Messages screen for chatting functionality.

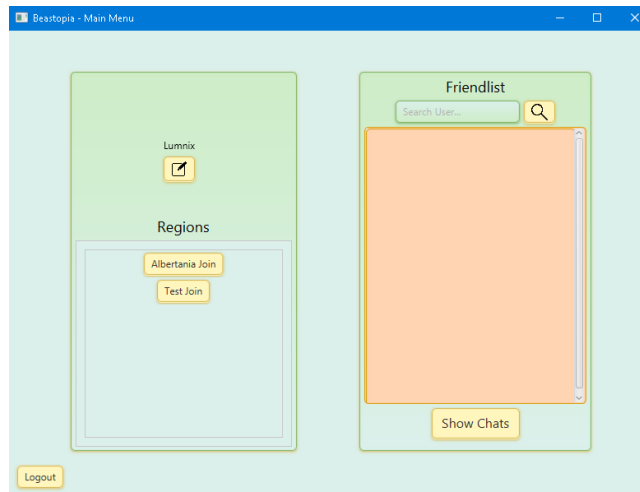


Figure 12: Main Menu Implementation

Figure 12 displays the Main Menu implementation that comprises of two submenus, in accordance with the wireframe showcased in Figure 11. The left submenu contains the application's banner, the user's name along with their custom avatar, and an edit icon for modifying profile information. Additionally, a list of regions that the user can join is available, with further development planned in future releases. The right submenu displays the user's friend list, indicating all added friends, and a button at the bottom enables the chat feature. An additional feature of the application is the Logout button located in the bottom left corner, allowing users to log out and login using alternative

credentials without closing and reopening the entire application.

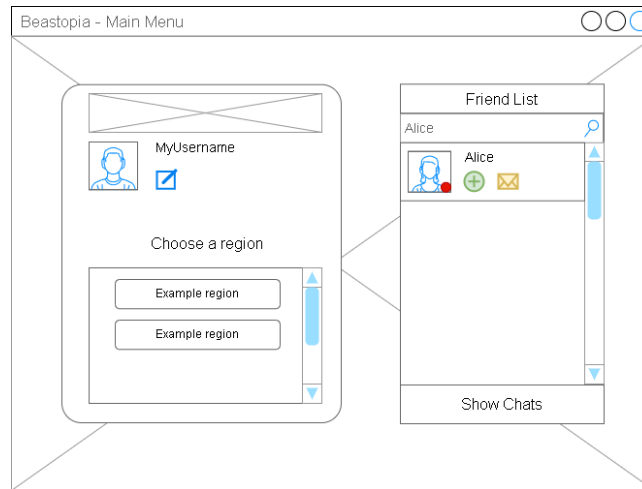


Figure 13: User Search Wireframe

Figure 13 depicts a user search scenario that differs from Figure 11. Specifically, when a username is entered in the search bar, the server searches for a matching user and displays their info card in the list. The remove icon is replaced by an add icon to indicate that the user can add the searched person to his/her friend list. Notably, the pin function is hidden in this context.

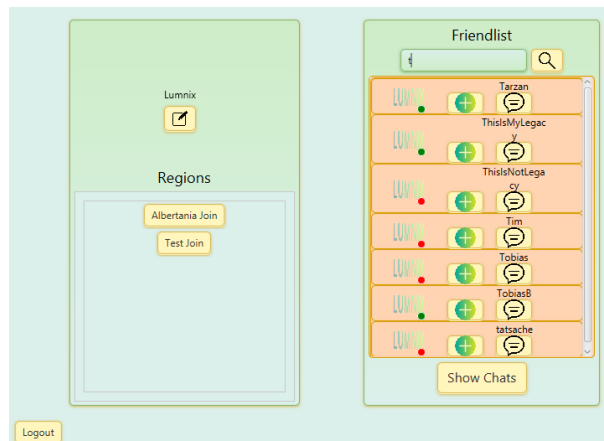


Figure 14: User Search Implementation

The visual representation of the User Search feature of the friend list is depicted in Figure 14. Upon entering any input in the Search bar, the system automatically initiates a search for usernames that contain the entered characters, except for the user's own username, as displayed in the figure. The Search icon is substituted with a refresh icon, enabling the user to refresh the current username search. As per the User Search wireframe, the user is offered the option to add a searched user, compose a direct message, and pin a user if already added to the friend list. However, it should be noted that the pin button is not displayed when the user performs a search for other users.

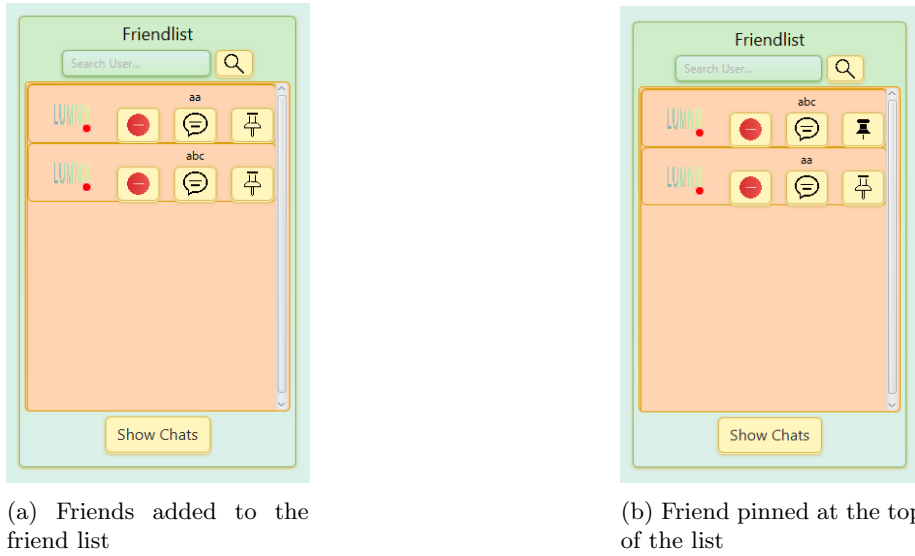


Figure 15: Friends on the users friend list

The process of adding a friend to a user's friend list is initiated by clicking the green add symbol located on a user's information card, as illustrated in Figure 15a. After a user has been added to the friend list, they can be pinned to the top of the list by clicking the pin icon located next to the direct messages icon. Figure 15b depicts a friend that has been pinned indicated by the change of the pin icon.

2.1.4 Ingame

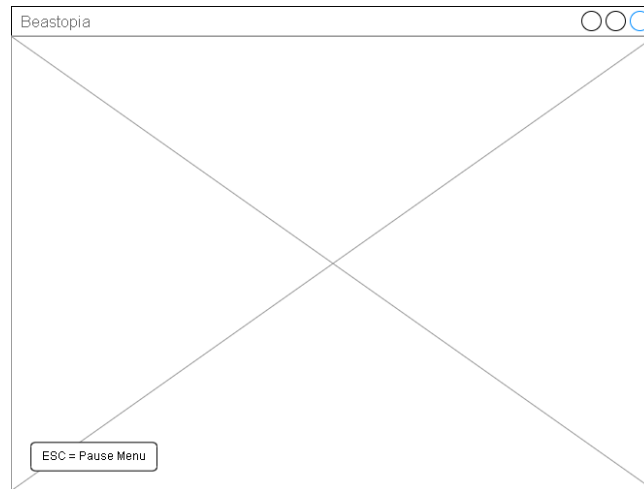


Figure 16: Ingame Wireframe

By selecting a region from the Main Menu, the user is directed to the Ingame screen, which will host the game logic implemented in later releases adding functionalities to actually be able to play the game. As illustrated in Figure 16, the Pause Menu can be accessed by pressing the ESC key on the keyboard.

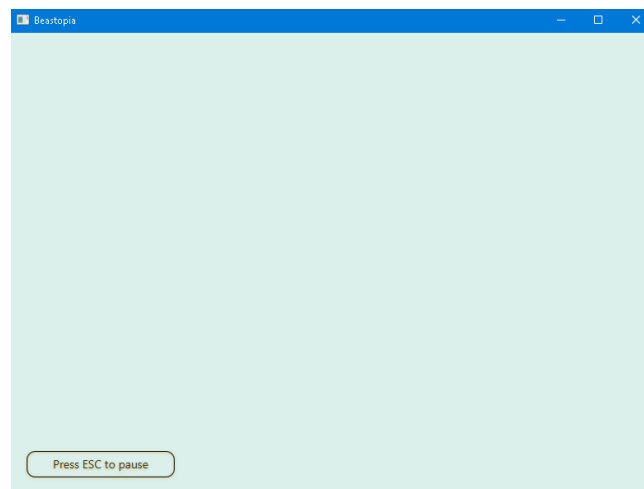


Figure 17: Ingame Implementation

The implementation of the Ingame screen is the same as its wireframe counter-

part, thus no big changes have been made during development as one can see in Figure 17.

2.1.5 Pause Menu

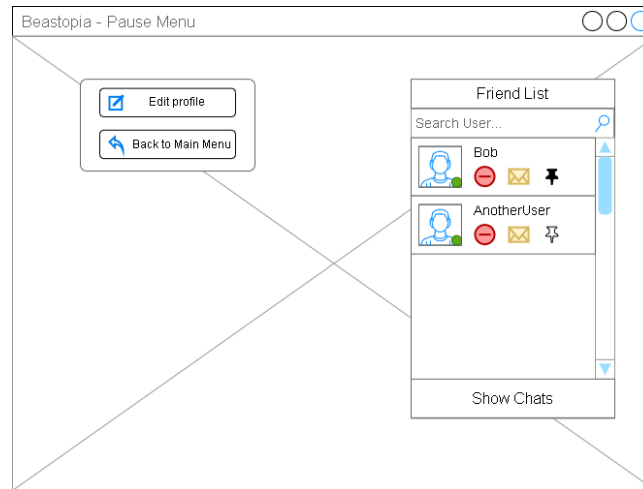


Figure 18: Pause Menu Wireframe

Figure 18 portrays the Pause Menu wireframe. The Pause Menu comprises two sub menus, which are labeled and placed on the left and right sides of the screen. The left sub menu features two buttons, one for redirecting the user to the Profile Edit Menu to edit his/her profile information, such as the username, password or avatar, and the second button functions as a navigation tool, allowing the user to return to the Main Menu. On the right side of the screen, the friend list of the user is displayed, including all its functionalities as outlined in Figure 11.

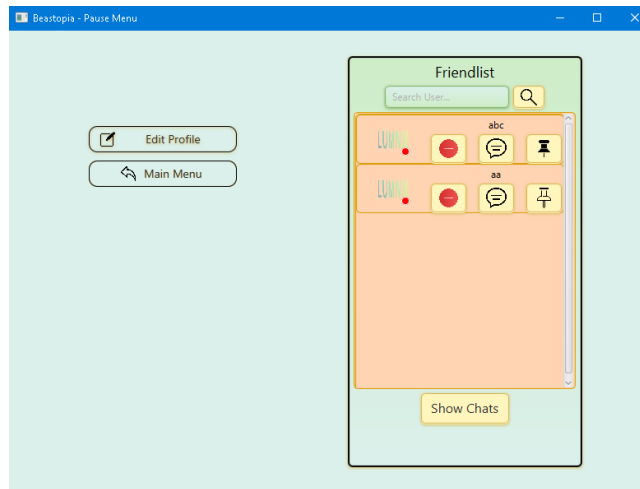


Figure 19: Pause Menu Implementation

As depicted in Figure 19, the Pause Menu implementation displays no significant alterations and is akin to the one illustrated in Figure 18, with the two buttons Edit Profile and Back to Main Menu situated on the left, and the user's friend list on the right. The Pause Menu is accessed via the ESC key on the keyboard, as detailed in the Pause Menu wireframe.

2.1.6 Edit Profile

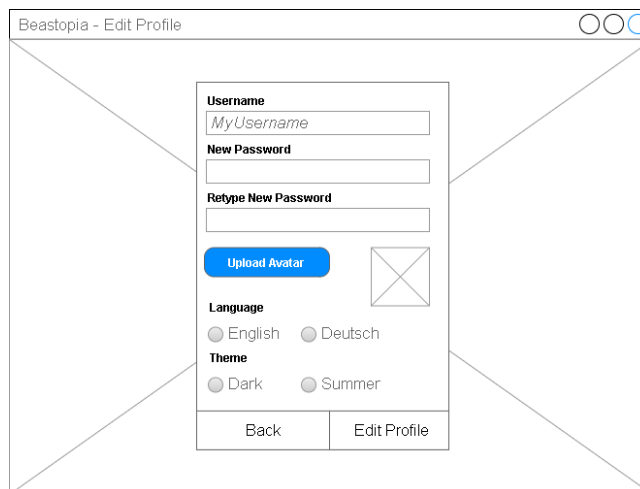


Figure 20: Edit Profile Wireframe

The wireframe displayed in Figure 20 depicts the Edit Profile Menu, which enables users to modify their profile information. The menu consists of labeled profile options stacked on top of each other. The user can select a new password and confirm it for security reasons, choose a language using radio buttons located under the Language label, and upload a new avatar by clicking on the Upload Avatar button. The user can also change the application's theme to either Dark or Summer.

If a user has made an error while changing any of the profile information, the Back button can be clicked to return to the previous menu. After clicking on the Edit Profile button, the changes are verified and updated.

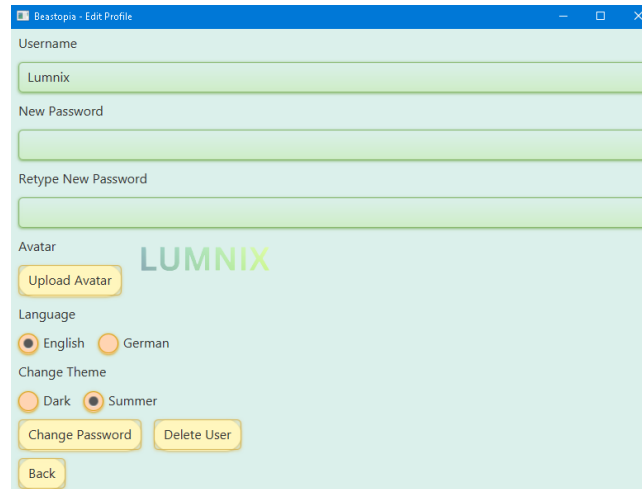
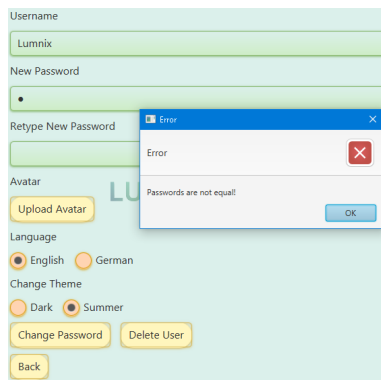
The image shows a web browser window titled "Beartopia - Edit Profile". The background is a light blue gradient. The form contains the following elements: a "Username" label above a text input field containing "Lumnix"; a "New Password" label above an empty text input field; a "Retype New Password" label above an empty text input field; an "Avatar" label next to a large "LUMNIX" logo, with an "Upload Avatar" button below it; a "Language" label above two radio buttons for "English" (selected) and "German"; a "Change Theme" label above two radio buttons for "Dark" and "Summer" (selected); a "Change Password" button and a "Delete User" button; and a "Back" button at the bottom left.

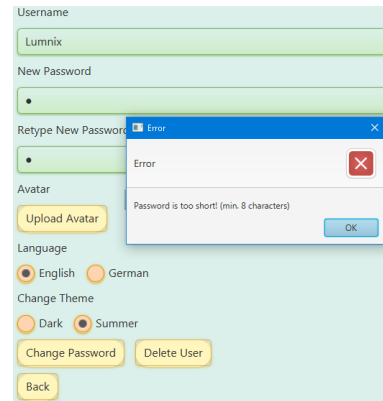
Figure 21: Edit Profile Implementation

The Edit Profile menu has undergone several modifications, as depicted in Figure 21. In addition to the existing functionality of setting a new password, confirming the password, selecting a new avatar, and adjusting the theme and language of the application, the arrangement of elements has been slightly altered from the wireframe displayed in Figure 20. Notably, the Edit Profile button has been replaced with a Change Password button, enabling users to update their password. All other changes made to the profile are immediately applied without requiring a separate button. Furthermore, a new option has been introduced adjacent to the Change Password button, allowing users to delete their entire account. The Upload Avatar section now includes a label called Avatar to enhance menu consistency. The Back button is positioned below the Change Password and Delete User options.

The ability to change one's username will be added in the upcoming release.



(a) Edited passwords do not match



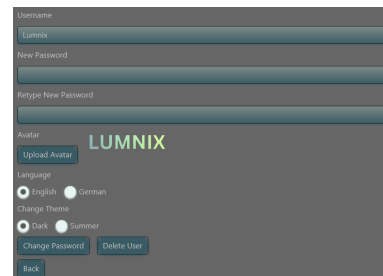
(b) Edited password is too short

Figure 22: Editing one's password

In Figure 22a, an error is displayed indicating that the password update process was unsuccessful due to the passwords not matching. Furthermore, Figure 22b illustrates that the user's password input is too short, falling below the required minimum length of eight characters.



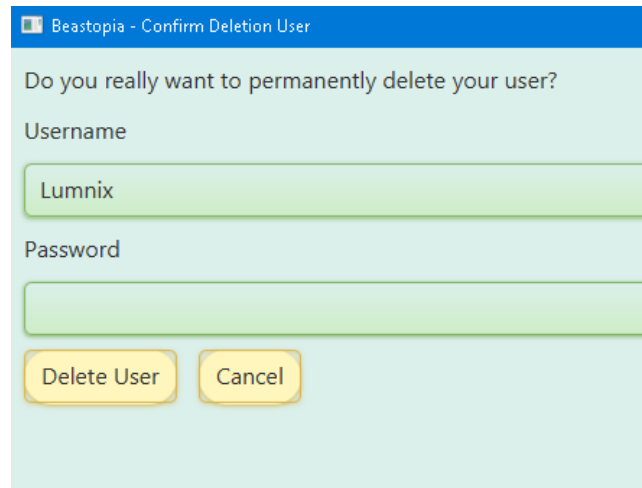
(a) Language changed to German



(b) Theme changed to Dark

Figure 23: Editing language and theme

Figure 23a illustrates the change of the language for the application to German and one can see in Figure 23b that the theme was changed to the already mentioned Dark theme. A click on one of the radio buttons triggers a change that is immediately applied to the application.



Beastopia - Confirm Deletion User

Do you really want to permanently delete your user?

Username

Lumnix

Password

Delete User Cancel

Figure 24: Delete User Implementation

As mentioned in the explanation of Figure 21, the user has the ability to delete their account. If the user may decide to delete their account for the reason being that he/she is unsatisfied with said account, he/she can click on the Delete User button. A new screen will guide the user through the deletion process as seen in Figure 24. For confirmation, the user has to type in his/her password.

2.1.7 Direct Messages

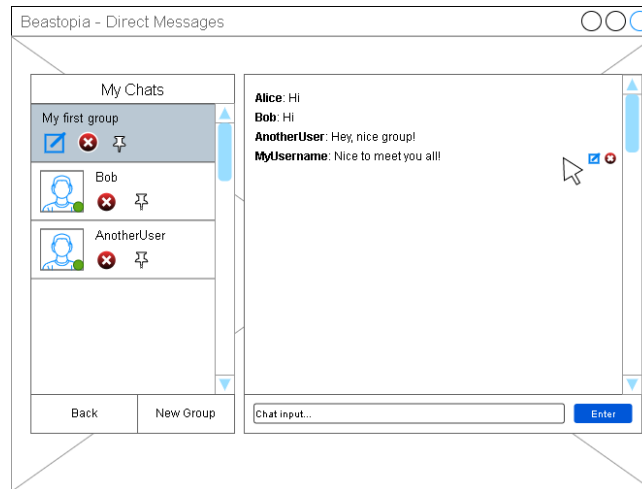


Figure 25: Direct Messages Wireframe

Figure 25 displays the wireframe illustrating the Direct Messages screen, which enables users to interact with each other through a chat functionality. The left side of the screen features a sub menu that exhibits all of the opened chats initiated by the user. The chat card that is selected is presented with a distinct background color and provides the name of the user, the selected avatar, and the current status. In the case of a group, the name of the group is shown alongside the icons. Three icons are shown for each chat information card, or two in the case of an individual user. For groups, an edit icon is available for modifying the group information, such as the name and the added users of the group. This icon is only displayed for users who own groups. The second icon enables the user to delete the whole chat conversation, including groups. When a user deletes a group chat, he/she is removed from that specific group. The third and final icon functions as a pin chat feature, which functions similarly to the pin friends feature in the friend list. The bottom of the chat card list has two buttons, labeled Back for returning to the previous menu and New Group for creating a new group.

The right side of the screen features the chat history of already written chat messages in that specific chat, as well as newly incoming messages. Users may edit or remove messages they have written in that chat by hovering over the desired message and clicking on the corresponding edit or delete icon. If the user wishes to add a new chat message, he/she may use the Chat input box located below the chat history. Once the message has been written, it can be sent by either clicking on the Enter button or pressing the Enter key on the keyboard.

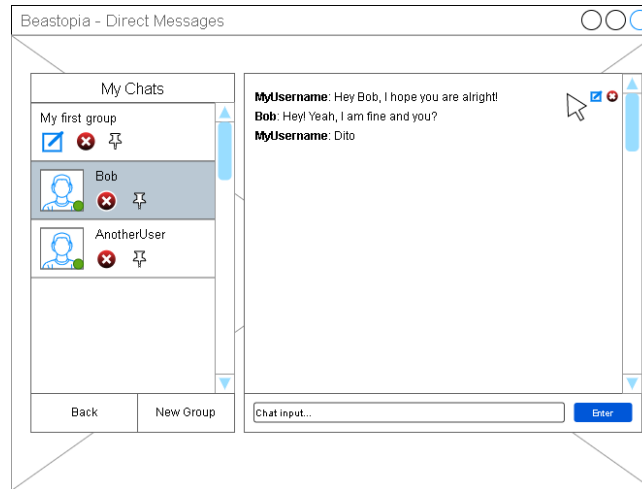


Figure 26: Direct Messages User Wireframe

The behavior of the Direct Messages screen is illustrated in Figure 25, while Figure 26 depicts the same behavior, but specifically for displaying the chat history and private messages exchanged between two users.

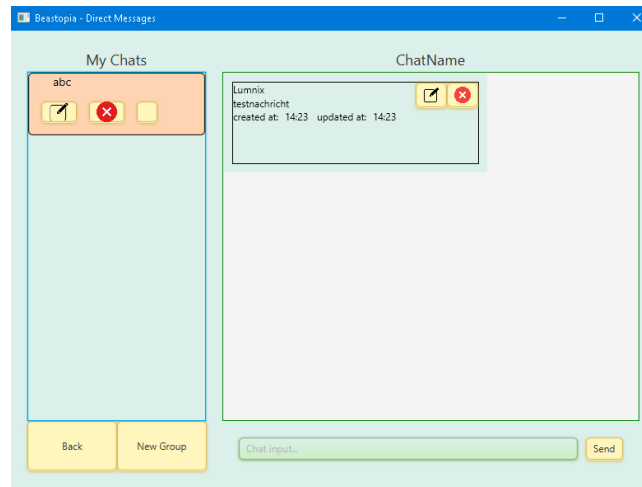


Figure 27: Direct Messages Implementation

The Direct Messages implementation is presented in Figure 27, which closely resembles the wireframe depicted in Figure 25. However, there are some minor

differences between the wireframe and the screenshot. For instance, the label for the list of the user's chats has been placed outside its box. Additionally, when a user clicks on one of their chats, a new label appears above the chat history displaying the name of the selected chat. This helps to distinguish the selected chat and accounts for the difference in height between the chat history and the list of chats. Furthermore, the button for posting a new message has been labeled differently to better describe its purpose.

One can also see, that a chat was started with a user. The buttons in the chat information card are placed like the one's in the wireframe and in the chat bubble the user can see what message was typed and when the chat was created and updated at.

2.1.8 Create Group

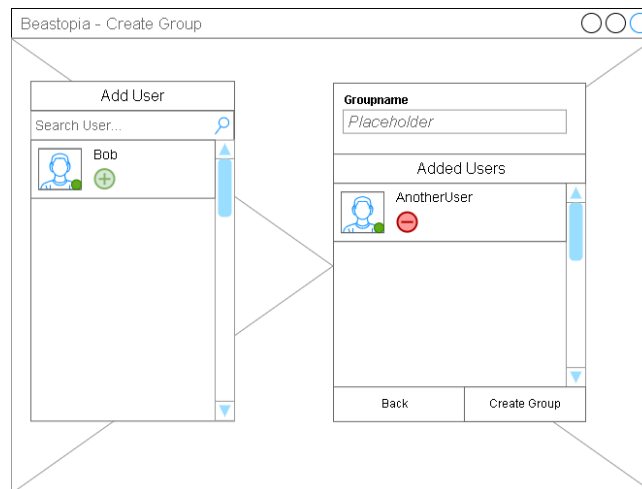


Figure 28: Create Group Wireframe

When a user selects the option to Create Group on the Direct Messages screen, the application presents the user with the Create Group screen, which is depicted in Figure 28. The Create Group screen consists of two submenus: The left submenu shows a list of potential group members, which primarily includes the user's friends. The Search bar can also be used to find users who are registered on the server. To add a user to the group, the user must click on the green plus icon next to the user's information card.

On the right side of the screen, the user can see the current group name, a list of users who have already been added to the group, and two buttons: Back and Create Group. The Back button returns the user to the previous menu, while the Create Group button creates the group with the added users. Users who have already been added to the group can be removed from the Added

Users list by clicking on the red minus icon located below the user's name on the information card.

2.1.9 Edit Group

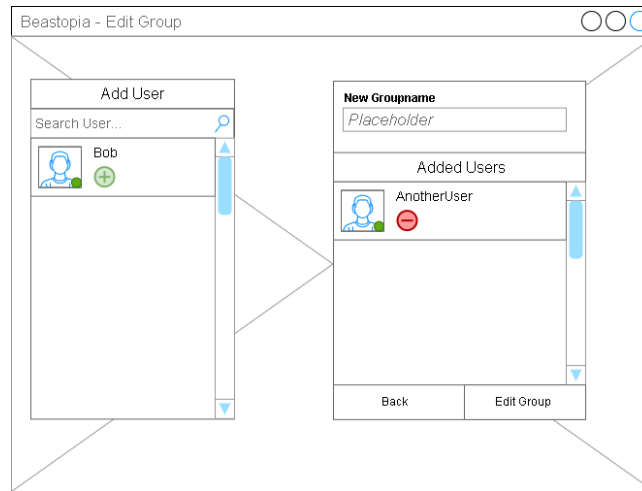


Figure 29: Edit Group Wireframe

The wireframe shown in Figure 29 illustrates the functionality of the Edit Group screen, which is similar to the Create Group screen shown in Figure 28 with the exception that it allows the user to modify the group's name by changing the text displayed beneath the New Groupname label, as well as add or remove users from the group. Upon making the desired changes, the user can either confirm them by selecting the Edit Group button or return to the previous menu by clicking the Back button.

Missing features

The implementation part of some of the features like the group options or deletion of chats do not exist but will be added in the following release. The reasons for this are stated in chapter 3.2.3 and chapter 3.3.3 respectively.

2.2 Technical Implementation

This section provides an overview of the technical concept behind the development of Beastopia. It begins with a brief introduction to the applied software pattern and then discusses the third-party libraries that were used during the development process.

2.2.1 Model-View-Controller Pattern

Software patterns are commonly used design conventions that are applied to structure code syntactically and semantically. When used correctly, the code should remain maintainable and expandable.

Beastopia is designed by utilizing a slightly modified MVC pattern. MVC divides software into three main Components: Model, View and Controller, whereby the modified MVC introduces a fourth component known as Service, as shown in Figure 30. Each component takes over its own responsibilities.

Model: Stores data within the application. Model refers to data structure and relations between data. Information is stored within objects of those structures. Although every other component is handling data, model is managing data that is viewable and changeable by the user [7].

View: Defines the appearance of the User Interface (UI). That includes visualization of data and control elements [7].

Controller: Manages logic behind the scenes. The Controller handles data, such as preparing data before being displayed, or input caused by the user, whereby direct changes in data do not happen. Furthermore it orchestrates view changes caused by any kind of trigger [7].

Service: Service introduces an abstraction layer between controller and model, allowing the controller to manipulate data through service functionality, ultimately making controller slimmer and improving testability.

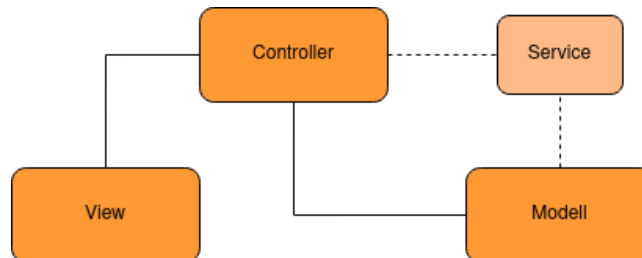


Figure 30: Modified MVC pattern

2.2.2 Third Party Libraries

Beastopia is written in Java 17 [4]. Java is a versatile high level, cross platform object-oriented programming language. Due to its popularity a vast and thriving ecosystem has been developed over the years providing numerous useful libraries, that enhance development capabilities. In the following, we will list some of the libraries used to develop Beastopia.

JavaFX

Building a UI can be tedious. Keeping Nodes in position, data and labels updated can be challenging. But it does not have to be. JavaFX is an open source Java-based framework for developing Graphical-User-Interface (GUI) that provides a wide range of functionality to simplify development [6]. JavaFX is distributed under GNU General License (GPL) which allows the creation of both open source and commercial applications. However if JavaFX is extended, the licence has to be passed on.

The view is constructed as scene graph inside a main stage in JavaFX, nodes in that graph can represent element used to input and represent data. Creating those views is done in a declarative way, using FXML markup language which emphasizes the graph structure.

In the development of Beastopia, JavaFX is used to design the front end of the application, binding data to that front end and implementing styles such as our Dark and Summer theme. In addition all built-in functions of Java and other libraries can be utilized due to the reason that JavaFX is based on Java.

Retrofit

Retrofit is a type-safe HTTP client for Java. Developed by Square and licensed under Apache License version 2.0 [3]. Retrofit models Representational State Transfer (REST) endpoints as Java interfaces by describing these endpoints through annotations. This results in interfaces that easy to read and consume within our code base. Furthermore it takes over serialization and deserialization by utilizing other converter, such as Jackson ¹.

Beastopia utilizes Retrofit to create, send and receive HTTP responses. Due to the abstracting capabilities of Retrofit, developer do have a faster way to define new endpoints and using already existing endpoints implemented by other developers.

¹JSON processor for Java

Dagger

Dagger is fully static, compile-time dependency injection framework also developed by Square and licensed under Apache License version 2.0 [2]. Dagger provides functionality to unwrap dependencies in Java classes, instead of writing classes that depend on each other in a hardcoded way. We tell objects what what dependencies they need through annotations. As a result Dagger generates a lot of boilerplate code that handles those injections and provides factories that generates those required objects.

Instead of manually creating dependent objects and passing them down from object to object, Beastopia utilizes Dagger for dependency injection. This approach ultimately slims down the code base, making it more readable and testable.

RxJava

RxJava is a library for composing asynchronous and event-based programs by using observable sequences [5]. Licensed under the Apache License version 2.0. RxJava brings reactive programming to the table, enabling us to compose asynchronous and event-based functions based on so called observables. Observables are the key concept behind RxJava. They represent a stream of data or events on which various operations can be applied on. Such as filtering, transforming, combining or merging. Furthermore RxJava introduces schedulers, which specifies on what thread the functions should be performed.

Beastopia utilizes RxJava to handle asynchronous events, such as server requests. Due to its asynchronous response, RxJava observes requests on another thread and provides results as soon as the request is done.

JUnit

JUnit is a popular unit testing framework [1], licensed under the Eclipse Public License v2.0. JUnit provides a simple and effective way to write unit test. Through annotations it is possible to declare test classes and within those classes, unit tests. Furthermore JUnit provides different assertion functions to catch bugs within the logic of the code. The modularity of JUnit makes Test-Driven-Development (TDD)² possible.

Beastopia is firmly tested throughout the development. All services, as shown in Figure 30, are tested with unit test, making Beastopia less bug prone. Especially if different developers work on the same code base the testing prevent bugs from creeping in.

²Writing tests, such that the test declares the behavior of the desired function before writing the actual function. Hence unit tests.

Mockito

Mockito is one of the most valued mocking framework and licensed under the Massachusetts Institute of Technology (MIT) license. When testing software components that have dependencies reaching beyond their own scope, a different approach is needed compared to unit testing. The key concept to handle dependencies in tests is mocking. Mocking means, defining objects that simulate functional dependencies. By giving these mocks a fixed behavior, the dependent code can be tested without invoking the actual dependencies. This allows for isolated and controlled testing of the target code, enhancing testability and facilitating more comprehensive testing scenarios.

Due to the reason that dependency injection pattern is used to develop Beastopia, several software components incorporate dependencies. To ensure proper functionality Mockito is used to test those components.

3 Project documentation

This chapter will provide a detailed examination of the project process. We will introduce the Scrum framework and review the two sprints that took place in Release one. Additionally, we will discuss stories and independent tasks as main part of this chapter. Finally we will elaborate those stories and tasks that did not finish within the time frame of Release one.

3.1 Scrum

Good project management is crucial for the success of any project, including software projects. However, software projects often require more frequent adaptations than other types of projects. Consequently, conventional project management approaches may not be well-suited for software projects.

A more suitable approach for managing software projects is agile project management, with Scrum being a popular framework in this regard. Scrum is a framework that encompasses values, principles, and methods to facilitate effective software development team organization and structure, ultimately enabling the achievement of project goals [8]. Scrum follows an iterative approach, allowing for continuous refinement and adjustment of the development process based on feedback and lessons learned. This iterative nature of Scrum promotes flexibility, adaptability, and responsiveness to changing requirements throughout the project lifecycle.

3.1.1 Scrum structure

In Beastopia’s development, one release encompasses a total duration of four weeks. This four-week period is divided into two sprints, with each sprint lasting two weeks. This division allows for a focused and time-boxed approach to development, where specific goals and tasks are assigned to each sprint. By breaking down the release into sprints, the development team can work in smaller, manageable iterations, ensuring continuous progress and the ability to adapt and adjust as needed throughout the development process.

A sprint did consist of:

- One Refinement
- Two Weeklys
- Two Hackathons

Whereby Refinements were used to estimate the workload proposed by the stories. Every developer could propose an estimation, based on that a median was concluded and the story was assessed. Weeklys did provide a platform to update the whole team regarding what was done in the passed week and Hackathons did allow to work collectively on open tasks and reach agreements on group decisions.

3.2 First sprint

Sprint one was conceptualized to implement Authentication, Lobby and a fraction of Ingame functionality. Officially sprint one was intended to be from April 24, 2023, until May 7, 2023. However, due to the sprint start on April 24 and the meeting with the customer on the same day it could not start on April 24. Following the meeting, our Product Owner (PO) started writing stories based on the customers expectations. Workload and corresponding time of the stories where estimated by the development team in an refinement meeting. Based on those stories and estimations, tasks has been derived at the hand of our Scrum Master (SM). These tasks where created to break down the work required to achieve the features outlined in the stories.

3.2.1 Jira stories

STP23L-2 – Login

The first story of the sprint delineates the login interface, which manifests upon a user’s initiation of the application or after registering. The login interface prominently features a form, wherein a user is required to input their unique identifying information. The user is then prompted to specify their preferred language for the application. Upon completion of the requisite inputs, the user proceeds to trigger the Login button, which initiates the authentication process. Upon successful authentication of the input user credentials, the user is granted access and redirected to the Main Menu.

STP23L-4 – Registration

The fundamental objective of registration entails the provision for a user to create a unique account within the application. Upon selecting the Sign up link exhibited on the login interface, the user is directed to the registration screen. This screen showcases the requisite user credentials, including but not limited to, the username and password. Moreover, the user is afforded the option of selecting an avatar and specifying their preferred language. Upon furnishing the necessary inputs and selecting the Sign Up button, the validation procedure commences, and with successful completion, the user account is formally established. Thereafter, the user may log in after being redirected to the login interface.

STP23L-8 – Choose Region

In the event that a user has successfully authenticated and gained access, the initial interface that manifests is the Main Menu, also denoted as the lobby. This screen affords the user with an exhaustive synopsis of their friend list, as well as a comprehensive catalog of the available regions for potential exploration. Upon the user's selection of a region from the aforementioned list, they are instantaneously transported to the specified region, wherein the In-game screen is exhibited for the commencement of gameplay.

STP23L-10 – Find user

The friend list encompasses a distinctive functionality that enables a user to conduct a search query for any user that is presently registered on the server. This feature facilitates seamless interaction between users, allowing them to establish a connection with their desired user, either by adding them to their friend list or by dispatching them a direct message.

STP23L-11 – Add friend

As delineated in STP23L-10, a user is empowered with the capability to execute a search for other users within the server. Upon the successful discovery of the desired user, the interface provides an array of options for engaging with the specific user. One such option is to add the user to one's friend list by selecting the green add symbol that is prominently exhibited within the user's information card. Once the user is appended to the friend list, future interaction with said user is considerably simplified.

STP23L-12 – Friends status

In the event that a user conducts a search or has already added some friends to their friend list, the information card of each user presents a small circular symbol that is situated in the bottom right corner of their respective avatars. The symbol serves to indicate the online status of the user and is denoted by one of two possible status options: Online, signified by a green circle, indicating that the user is currently logged in to the server, or Offline, represented by a red circle, which signifies that the user is presently inactive.

STP23L-13 – Pin a friend

The act of pinning a user causes their information card to be persistently displayed at the top of the friend list, thereby facilitating ease of access and interaction. The pinned users are sorted by a principle, in which a newly pinned user occupies the topmost slot in the friend list.

STP23L-18 – Pause game

Upon selection of a region by the user, the In-game screen is displayed, which exhibits a small notification located at the bottom left corner, which provides a means for the user to access the Pause Menu (accessible by pressing the ESC key on the keyboard). The user can activate the Pause Menu by pressing the ESC key, which then presents a display of the user's friend list, in addition to options to modify one's profile information and navigate back to the Main Menu.

3.2.2 Additional Jira Tasks and Bugs

STP23L-5 – Prepare Project structure

Preparing the project structure was one of the first task to handle, where a developer went over the project and extended the structure by adding a folder hierarchy, dependencies and package names.

STP23L-6 – Data model

At first the idea was to develop a data model containing relations between objects within Beastopia. However as development went on, we realized that a fixated data model did not make much sense and generated too much boilerplate code. Hence, the idea was scrapped and we decided to use Java records instead.

STP23L-7 – Implementing Dagger

As explained in section 2 Dagger was an essential part in developing Beastopia. Implementing Dagger is a task that was designed to implement Dagger components early on, making it possible to use dependency injection right from the start.

STP23L-53 – Internationalization (I18N)

As agreed upon, internationalization was a feature that needed to be implemented in Beastopia. The needed infrastructure within the code to realize such a language change was established with this task. Additionally, already existing text elements were translated and added to the dictionary.

STP23L-57 – Server API Services

With Server API Services a developer did implement the communication to the server. The functionality was encapsulated within services, making them easy to use for other developers.

3.2.3 Sprint retrospective

Retrospectively, sprint one opposed different difficulties, mostly structural problems such as:

- Time-wise problems
- Ticket design problems
- Ticket dependencies

Furthermore as freshly founded company and with Beastopia as our first project the team had to adapt, meaning that everybody found themselves in an new team construct. Beyond that, uncertainties came up at the beginning regarding working together and working with Scrum. Nevertheless, after a short time period tasks were processed and Pull-Request (PR) started to roll out. A solid communication was established between members of the team laying a foundation for further teamwork.

At the end of sprint one five stories had been not finished, as shown in the following table:

Story/Task	Title	Reason
STP23L-18	Pause game	missing test
STP23L-8	Choose Region	missing test
STP23L-10	Find user	dependencies
STP23L-11	Add friend	dependencies
STP23-L-60	Save login credentials	added late to first sprint

Table 1: Unfinished tasks and stories in first sprint

Reasons that led to this discrepancy will be elaborated here. Generally said all reasons listed in the beginning of 3.2.3 also apply here, however there is more to it.

STP23L-18 - Pause game

Pause game did have dependencies in the beginning, at least the final implementation was depending on Ingame, whereby Ingame final implementation was depending on Choose Region. That led to a chained dependencies that had to be

finished in order, yet work was done partially on the View and Controller components. Although the View and Controller components were finalized quickly, after Ingame was realized, it took time to create the test. Due to late incoming of information on "How to test" the testing developers attention was shifted towards other tasks and had to implement the test later. Hence, with only the test-task still undone this story had to be carried into sprint two.

STP23L-8 - Choose Region

Most of the work for Choose Region was finalized in the first week of sprint one. However, as mentioned before testing was not introduced yet. The assigned developer continued with other tasks, which took more time than expected. That ultimately led to missing capacities to finish testing Choose Region. Hence, testing Choose Region had to be carried into sprint two.

STP23L-10 - Find User

Find User was a story that depended on the basic structure of the main menu and friend list. Those were incorporated throughout the first half of sprint one. In addition Find User was also dependent on server communication, although that was finished at the beginning of the second half of sprint one, the assigned developer on Find User was already working on something different. At the end of sprint one the developer was working on Find User but did not finish in time. Hence, Find User had to go into sprint two.

STP23L-18 - Add Friend

As for Add Friend, the reason that it got carried over into sprint two is due to dependencies on other stories and the lack of developer capacities in sprint one. It was not started in sprint one.

STP23-L-60 - Save login credentials

Save login credentials was added late to the first sprint. That was simply a mistake, instead of adding a new task other tasks should have been reassigned to another developer. For instance Add Friend could have been reassigned. That mistake led ultimately to taking over two stories into the second sprint.

The burndown chart in Figure 31 represents the issue count over the period of the first sprint. A closer look reveals that during the sprint three issues were added, which are:

- STP23-L-53 Internationalization (I18N)
- STP23-L-57 Server API Services
- STP23-L-60 Save login credentials

As mentioned before, adding Save login credentials was a mistake. However adding Internationalization and Server API Services was essential to create a

better code base. Meaning other developer could use provided functionality, such as the server API. Furthermore with implementing Internationalization, it was possible to create and maintain a language package for View components from the beginning, eliminating the work that had to be done later on.

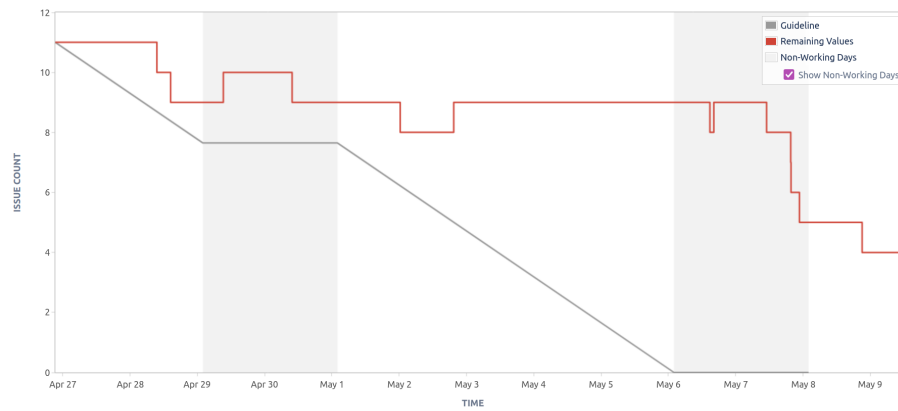


Figure 31: Burndown chart first sprint

3.3 Second sprint

The second sprint in Release one was aiming to implement chat functionality and finalizing Lobby and Ingame components. Additionally independent tasks such as refactoring, Internationalization functionality and web socket implementation were planned. Considering the feedback received from the first sprint, the underlying tasks of stories were designed different in order to improve workflow and reduce dependencies. However, based on the written stories dependencies could not be reduced to zero.

3.3.1 Jira stories

STP23L-14 – Show all chats

When the user navigates to either the Main Menu or the Pause Menu, they are provided with an option to display a comprehensive overview of all their previously initiated chats. This functionality is accessible via the "Show Chats" button, situated at the bottom of the friend list. Upon selection of the Show Chats button, the application displays a new screen named Direct Messages, which contains details of the user's opened chats.

STP23L-16 – Show specific chat

The user has the ability to send messages directly to friends on their friend list or to users that have been searched using the Search bar, which also triggers the opening of the Direct Messages screen. By clicking on the yellow envelope icon displayed on the user's info card within the friend list or on a searched user's info card, the user can access the Direct Messages screen and initiate a chat with that particular user.

STP23L-19 – Profile edit menu

Users are afforded the capability to access a menu allowing the modification of their personal profile information. This feature can be accessed either by navigating to the Pause Menu and pressing the ESC key to unveil the Edit profile button or by visiting the Main Menu and selecting the edit icon positioned below the username. Upon selection of either option, the user is redirected to the Edit Profile screen, allowing the option to update their personal information provided during registration.

STP23L-24 – Write chat message

In order to initiate a new chat message, a user is required to enter the intended chat message into the designated chat input box. Once the message is complete, the user must press the Enter key on their keyboard or click on the Enter button in order for the message to be transmitted to the intended user.

STP23L-25 – Create group

When a user intends to initiate a group chat, he/she can create a new group by selecting the New Group button available on the Direct Messages screen. Upon selecting the New Group option, the user is directed to the Create Group screen which displays the list of users and group information. To add members to the new group, the user can search for users and include them by clicking on the green plus symbol on the user's info card. Subsequently, the new group members are displayed below the Added Users label. The user can then specify a name for the new group, thereby establishing its identity, and ultimately generate the group by selecting the Create Group button.

STP23L-26 – Edit group

The ability to edit groups is facilitated through clicking on the edit icon located on the group chat's information card present in the Direct Messages screen. However, this action is solely available to users who created that specific group. Subsequent to clicking on the edit icon, the user is directed to a comparable screen akin to the Create Group screen as delineated in STP23L-25. The user is then afforded the opportunity to rename the group, remove users by clicking on the red minus symbol located beneath the Added Users section, and add new users to the group by clicking on the green plus symbol situated on a user's information card. Following these modifications, the user may click the Edit Group button to affirm the alterations and update the group accordingly.

STP23L-27 – Delete chat message

To delete a previously sent chat message, a user can interact with the messaging interface in the following way: Firstly, the user must navigate to the chat in question and then locate the particular message that they wish to remove. Hovering the cursor over the targeted message will reveal a pair of symbols to the right-hand side of the message. The user should then click on the symbol consisting of a white cross on a red background to initiate the deletion process. Prior to the message being permanently deleted, a confirmation dialogue will be displayed to ensure the user's intention to delete the message.

STP23L-28 – Delete chat

The user can initiate the deletion of an entire chat and automatic withdrawal from a group by clicking on the white cross with the red background, located on the chat information card on the Direct Messages screen. Upon triggering the delete action, a confirmation popup is presented to the user, and if confirmed, the chat will be removed and the user will be removed from the group, in the case of a group chat.

STP23L-29 – Pin chat

To prioritize a chat on top of a user's chat list in the Direct Messages screen, the user can utilize the pin icon located in a chat's info card. Activating the pin icon ensures the specified chat remains at the top of the list, similar to pinning a friend on a friend list as previously discussed in STP23L-13.

STP23L-31 – Edit profile information

In case of dissatisfaction with one's profile information, users can access the Edit profile menu via the Main Menu or Pause Menu. The Edit Profile menu facilitates the user to modify various profile attributes such as username, password, avatar, and language preferences. The changes to the profile attributes can be made independently and are confirmed by clicking on the Edit Profile button.

STP23L-60 – Save login credentials

On the login screen, a user is presented with a login form which can be filled out to access the application. An optional feature available to the user is the ability to retain the entered login credentials for subsequent use, by selecting the Remember Me checkbox. This feature significantly augments the ease-of-use of the application.

STP23L-61 – Change app theme

The application offers users the option to select between two distinct color themes. To access this feature, the user must navigate to the Profile Edit Menu and choose between Summer for the light theme or Dark for the dark theme. Once a selection has been made, the color palette of the application will be adjusted accordingly, adding a level of visual diversity to the user experience.

3.3.2 Additional Jira Tasks and Bugs

STP23L-96 – Revisit Main Menu

Main Menu was created within the story Login, however it was just designed to contain a region- and friend list. This task did expand the main menu to also contain the username, an avatar, edit profile button and a place holder.

STP23L-92 – General Refactoring and Testing

Throughout the first sprint some testing was left out, due to not considering them while designing subtasks within relating stories. Additionally the code base was growing and some functionality was starting to being tedious to maintain. Both these aspects had to be improved with this task.

STP23L-99 – Own username can be found via Friendlist

While testing the application we noticed that the own user can be found within the user search. That was issued with this bug ticket.

STP23L-91 – Revisit fxml's

In transition from the first sprint to the second sprint some FXML's did stand out because they where not scaleable, which was fixed with this task.

STP23L-95 – Add I18N to Window Title

Internationalization was already implemented, however we missed to include the window titles. That was touched with this task.

STP23L-97 – Pause Menu can't be closed

This bug issues the problem of users being able to open the pause menu from within the in-game screen but not being able to go back.

STP23L-98 – Pin function shown for non-friends

Pinning is a feature that is reserved for friends only. While testing we noticed that not only friends can be pinned but also all other users. That was fixed with this bug ticket.

STP23L-63 – Websocket

Implementing web sockets was completed with this task, a developer added basic web socket functionality to Beastopia, making it easier for other developer to use this functionality and expand it.

STP23L-93 – Finalize language change

The infrastructure of Internationalization was already implemented. With this task the functionality behind the language change option within the login and

edit screen was realized.

STP23L-120 – AppTest when App closed with German language

The majority of tests did break when the application was closed and German was chosen as language, due to the assertions done in English. That was issued with this bug ticket.

STP23L-118 – Closing Application

During testing we noticed that closing the application does not close all threads accordingly. This bug ticket issues this problem.

STP23L-120 – Make friendlist scaleable

Friend list was not expanding and contracting accordingly to the parent view, hence this a bug ticket was issued to tackle the problem.

3.3.3 Sprint retrospective

In hindsight, the second sprint faced the same structural issues as those listed in the first sprint. The unfinished stories that has been carried over from the first sprint were finished rather quickly. However, the second sprint presented complex dependencies between stories that could not be entirely resolved.

At the end of the second sprint eleven tasks, stories and bugs in total had not been finished. As following table shows:

Story/Task/Bug	Title	Reason
STP23L-14	Show all chats	missing test
STP23L-16	Show specific chat	missing test
STP23L-24	Write chat message	missing test
STP23L-25	Create group	dependencies
STP23L-26	Edit group	dependencies
STP23L-27	Delete chat message	dependencies
STP23L-29	Pin chat	dependencies
STP23L-31	Edit profile information	missing test
STP23L-59	Edit chat message	dependencies
STP23L-100	Make friendlist scaleable	missing capacity
STP23L-159	Return from EditProfile	late Bug

Table 2: Unfinished tasks and stories in second sprint

Due to the mentioned structural problems and the additional reasons listed in Table 2, we were forced to carry those tasks over into release two. In the following we will elaborate what circumstances led to that.

STP23L-14/16/24/31

Taking a closer look reveals that STP23L-14/16/24/31 were just missing tests. These tests did not receive a high priority at the end of release one. Due to the fact that the test coverage had already been met and there were still pending features to be implemented, the decision was made to prioritize the development of those features. Last but not least, LUMNIX needs to reinvent its strategies on how to act on sudden changes in the scope of the sprint during the sprint.

STP23L-25 - Create group

Create groups was dependent on the whole chat frontend infrastructure, that pushed the starting condition way back. Furthermore the assigned developer was constantly working. Which concludes that capacity in general was missing.

STP23L-26 - Edit group

As in Create group, Edit group had also hard dependencies, which led to delays and ultimately resulting in taking Edit group over into Release two.

STP23L-27/29/59

Regarding Delete chat message, Pin chat and Edit chat message stories, all three of them had dependencies on other Stories, leading to chain dependencies. Which led to the circumstance to carry those Stories over into Release two.

STP23L-100 - Make friendlist scaleable

Make friendlist scaleable is a Bug that was identified towards the end of the sprint. Due to the lack of immediate resources to address the bug, it was not resolved during the sprint and was carried over to Release two.

STP23L-159 - Return from EditProfile

This bug came in late at the release day, so initially it was not realistic to finish it at the same day.

Figure 32 displays the Burndown chart of the second sprint. It is clear that work has been done continuously. Unfortunately, there were delays in different stories due to chain dependencies and missing developer capacities. During the two week period, bugs were found and taken into the sprint.

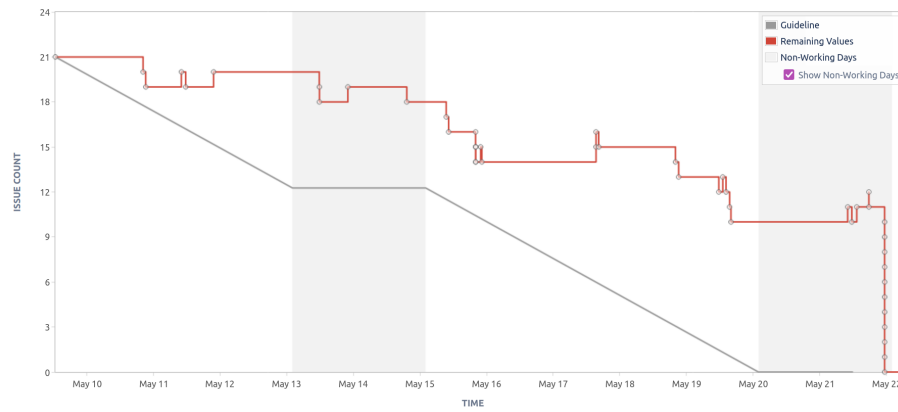


Figure 32: Burndown chart second sprint

3.4 Lessons learned

Recapping release one reveals certain areas where there were shortcomings, which LUMNIX can consider as valuable lessons learned and make improvements accordingly. One of them would be to design stories with fewer dependencies, as this could help optimizing sprints and make them more well-thought-out. Additionally, it is evident that developer resources need to be expanded, as there was not enough coding power to effectively execute the sprints.

3.5 Story overview

Story ID	Title	Estimated time	Time required
STP23L-2	Login	17h	19h 10m
STP23L-4	Registration	4h 3m	3h 45m
STP23L-8	Choose Region	4h	10h 35m
STP23L-10	Find user	3h	4h 3m
STP23L-11	Add friend	2h	2h 20m
STP23L-12	Friends status	1h	1h
STP23L-13	Pin a friend	1h	1h 4m
STP23L-14	Show all chats	12h 30m	18h 54m (+)
STP23L-16	Show specific chat	2h 30m	-
STP23L-18	Pause game	4h	6h 55m
STP23L-19	Profile edit menu	3h	1h 58m
STP23L-24	Write chat message	3h	1h 50m (+)
STP23L-25	Create group	6h	-
STP23L-26	Edit group	3h	-
STP23L-27	Delete chat message	2h 30m	-
STP23L-28	Delete chat	2h 30m	5h
STP23L-29	Pin chat	3h	-
STP23L-31	Edit profile information	3h	7h 18m (+)
STP23L-60	Save login credentials	2h	6h
STP23L-61	Change app theme	4h	42m
Sum		83h 3m	90h 34m

Table 3: Overview of estimated and required time for every story

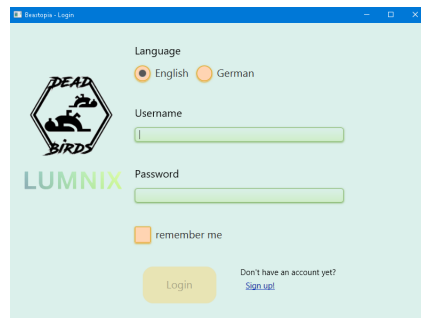
Table 3 presents a comparison between the estimated time and the actual time taken for each story, based on the explanations provided in chapter 3.2.3 and chapter 3.3.3, also including the explanation of stories with a minus sign. The presence of a (+) sign in the respective row of a story indicates that although work has been initiated, there are remaining unfinished tasks associated with that particular story and therefore marks the story as unfinished. These stories will be carried over to the next release.

4 Appendix

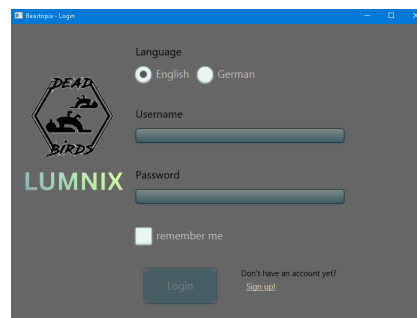
4.1 Comparison Summer and Dark theme

This chapter depicts the direct comparison of both themes, side by side. Some parts of the design are not finished yet. These will be addressed and completed in the upcoming release.

Login

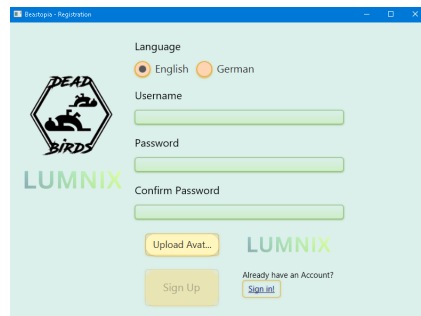


Light theme login form. The background is light blue. The logo on the left consists of a hexagon with 'DEAD' above and 'BIRDS' below a silhouette of a person on a motorcycle, with 'LUMNIX' in green below it. The form includes a language selector with 'English' selected and 'German' as an option. It has input fields for 'Username' and 'Password', a 'remember me' checkbox, a yellow 'Login' button, and a link 'Don't have an account yet? Sign up!'.

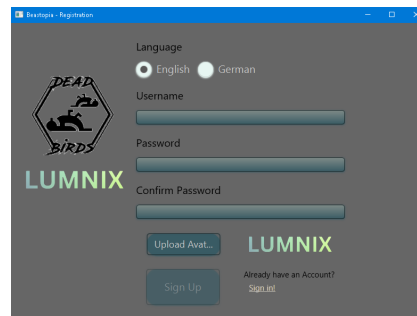


Dark theme login form. The background is dark grey. The logo on the left is the same as the light theme. The form includes a language selector with 'English' selected and 'German' as an option. It has input fields for 'Username' and 'Password', a 'remember me' checkbox, a dark grey 'Login' button, and a link 'Don't have an account yet? Sign up!'.

Registration

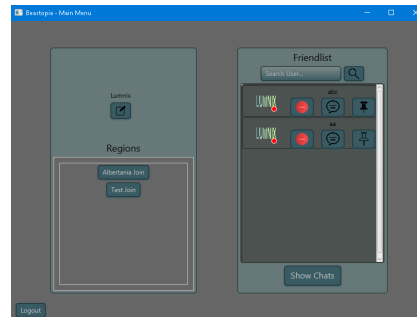
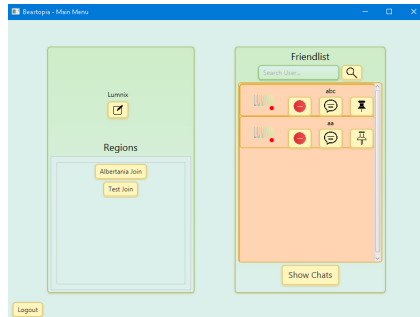


Light theme registration form. The background is light blue. The logo on the left is the same as the light theme login form. The form includes a language selector with 'English' selected and 'German' as an option. It has input fields for 'Username', 'Password', and 'Confirm Password', an 'Upload Avat...' button, a yellow 'Sign Up' button, and a link 'Already have an Account? Sign in!'.

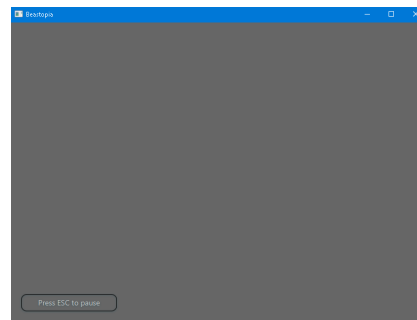
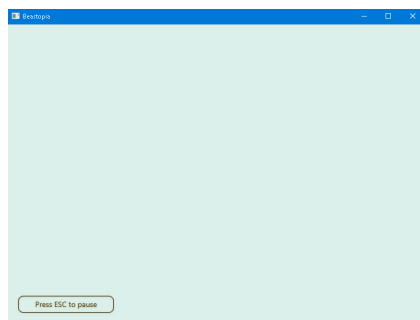


Dark theme registration form. The background is dark grey. The logo on the left is the same as the light theme. The form includes a language selector with 'English' selected and 'German' as an option. It has input fields for 'Username', 'Password', and 'Confirm Password', an 'Upload Avat...' button, a dark grey 'Sign Up' button, and a link 'Already have an Account? Sign in!'.

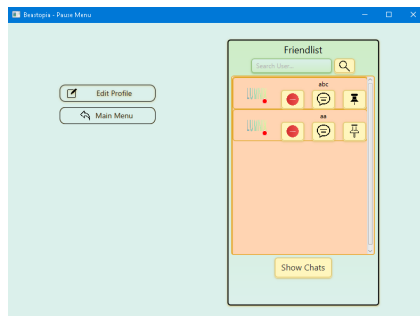
Main Menu



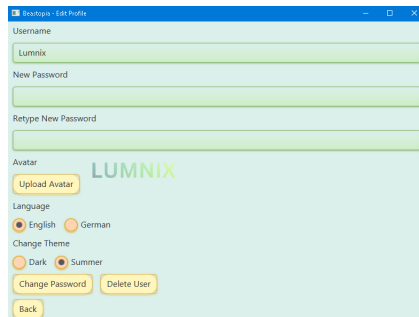
Ingame



Pause Menu



Edit Profile



Username
Lumnix

New Password

Retype New Password

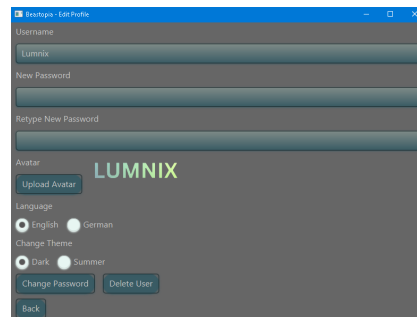
Avatar
LUMNIX
Upload Avatar

Language
☒ English ☐ German

Change Theme
☐ Dark ☒ Summer

Change Password Delete User

Back



Username
Lumnix

New Password

Retype New Password

Avatar
LUMNIX
Upload Avatar

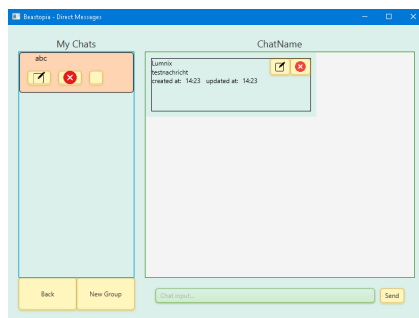
Language
☐ English ☒ German

Change Theme
☐ Dark ☒ Summer

Change Password Delete User

Back

Direct Messages



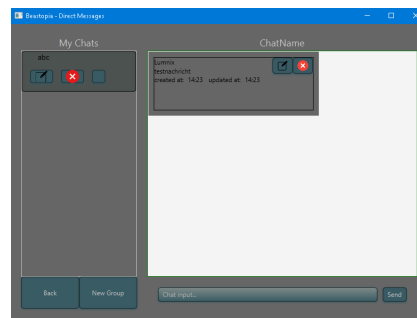
My Chats

ChatName

abc
Lumnix
bestachricht
created at: 14:23 updated at: 14:23

Back New Group

Chat input... Send



My Chats

ChatName

abc
Lumnix
bestachricht
created at: 14:23 updated at: 14:23

Back New Group

Chat input... Send

4.2 Acronyms

PO Product Owner

SM Scrum Master

RPG Role-Play-Game

MVC Model-View-Controller

UI User Interface

PR Pull-Request

GUI Graphical-User-Interface

GPL GNU General License

REST Representational State Transfer

TDD Test-Driven-Development

MIT Massachusetts Institute of Technology

NPC Non-Playable-Character

References

- [1] *A Guide to JUnit 5*. URL: <https://www.baeldung.com/junit-5>. (accessed: 21.05.2023).
- [2] *Dagger Homepage*. URL: <https://dagger.dev/>. (accessed: 21.05.2023).
- [3] *Introduction to Retrofit*. URL: <https://www.baeldung.com/retrofit>. (accessed: 16.05.2023).
- [4] *Java Downloads*. URL: <https://www.oracle.com/java/technologies/downloads/#java17>. (accessed: 14.05.2023).
- [5] *ReactiveX Homepage*. URL: <https://reactivex.io/>. (accessed: 21.05.2023).
- [6] Kishori Sharan. *Learn JavaFX 17 : Building User Experience and Interfaces with Java*. Berkeley, CA : Apress L. P., 2020, 2022. ISBN: 9781484278482.
- [7] Peter Späth. *Beginning Java MVC 1. 0 : Model View Controller Development to Build Web, Cloud, and Microservices Applications*. Berkeley, CA : Apress L. P., 2020, 2020. ISBN: 9781484262801.
- [8] *What is scrum and how to get started*. URL: <https://www.atlassian.com/agile/scrum>. (accessed: 13.05.2023).