Requirements and Analysis Document for
Smurfinator

Jonatan Cederberg, Erik Green Blomroos,
Isac Jason, Luqas Lundahl,
Noel Wiman Ohlsson

15/12-2023
Version 1.0

1. Introduction

The purpose of Smurfinator is to be a game of guessing, where the program either guesses a smurf, given information collected from the user by multiple forms of questions, or lets the user guess the name of a smurf based on the increasing amount of information revealed by the program. The program also features an account based user system, where users can contribute to the program's collection of smurfs by creating new ones, as well as featuring user-unique achievements, tracking the amount of discovered Smurfs. Smurfinator will also feature a leaderboard, showing the amount of contributions of each user in a sorted order, as well as a separate leaderboard showing smurfs sorted by amount of times found.

1.1 Definitions, acronyms and abbreviations

- Smurfinator - The main “game”/ also the name for the entire project
- Character - The thing the application is trying to guess
- Trait - Character has multiple traits (example: happy, magical, leader…)
- Questions - A question that corresponds to a trait, can either be answered by yes/ no or by a number between 1-10 (or “don’t know”).
- User - A user that the player is logged in to that tracks the users game contributions
- Contributions - The amount of characters a user has added to the program
- Leaderboard - Rankings entities based on corresponding measurable integer values.
- Collection - Pages of characters showing which ones has been guessed through playing.
- WindowHandler - The class that handles the fxml windows, takes action when buttons are pressed and displays new information when it comes from the model or controller.

2. Requirements

2.1 Definitions

Epics: More abstract User stories that are too large to fit in a sprint.

User Story: A natural word description of a feature
Traits: Traits that are associated with characters. EX “Glasses”
Questions: Questions that are associated with traits to. Ex “Does you character wear glasses” is associated with Glasses
Contributions: Amount of contributed characters that the player has added to the game.

2.2 Epics

- I as a customer want to have a program that can guess characters depending on the data given by the user
- I as a customer want to have a login feature that can login users as well as use those users to show different stats in the game.

2.3 User Stories

The time estimates written are the estimated time it would take to complete these tasks if someone works on the task continuously for a couple of hours a day.

User Story 1:
Estimated time 2 days
Description:
I as a user want many different types of questions to be asked

Tasks:
- Write many different types of questions for different traits.
- Write many different types of traits

Confirmation:
- There are a number of relevant questions that are connected to traits.

User Story 2:
Estimated time 5 days
Description:
I as a user want the be able to answer questions and that the program can guess on a character based on those answers

Tasks:
- There should be yes/no questions that update the results when answered
- The program should guess on characters depending on what the user has answered

Confirmation:
- The program should ask questions
- The user should be able to answer the questions
- The program should save the input of the questions
- The program should make a guess based on those answers

User Story 3:
Estimated time 2 days

Description:
I as a user should be able to login to a account

Tasks:

- The program should hash the password before saving it.
- The program should match the name and password with a database and return a user if it exists.

Confirmation:

- The password is saved hashed
- The program returns the correct user when a name and password is entered

User Story 4:
Estimated time 1 day
Description:
I, as a customer, want an account where I can save different attributes that are used in the application. And i want that information to be updated when the program is run.
Tasks:

- Create a list of Contributed Characters,password, name, amount of contributions.
- Update the user when action happen in the program

Confirmation:

- You can get this information from the user
- The attributes are updated when appropriate action happens in the game

User Story 5:
Estimated time 1 day
Description:
I as a user do not want to answer to contradicting questions
Tasks:

- Irrelevant questions should not be asked and instead removed
- Questions should only be removed if the player answers yes

Confirmation:

- Irrelevant questions are not asked, Ex if the question “Is you character a boy” it should not then ask if it it is a girl

User Story 6:
Estimated time 2 days
Description:
I as a customer want to be able to create users and use them in the program
Tasks:

- You should be able to create a user and save it to the list of users
- You should be able to get the users from outside the program

- Create database that can save users.

Confirmation:
- You can get users from outside the program
- You can save users to a file outside of the program
- Users that are retrieved from a file should have the correct information

User Story 7:
Estimated time 2 days
Description:
I as a user want to be able to answer a question in the range 1-10
Tasks:
- Create a type of question where you can answer in a range from 1-10
- Make it possible for the view to view questions that can be answered with a number between 1 and 10

Confirmation:
- The user can answer questions in a range of 1-10
- The view can display these questions

User Story 8:
Estimated time 2 days
Description:
I as a customer want the program to display an image and a name that is unique for each guess.
Tasks:
- Make it so that each guess is associated with a character that has and image and name connected to it.
- Create a name, image for each character.

Confirmation:
- When a character is guessed it shows a image and name of the character guessed.

User Story 9:
Estimated time 2 days
Description:
I as a user want to be able to add a new character if the one i am looking for does not exist
Tasks:
- Add the option to create a character if a guess is wrong
- The program must continuously ask questions until all have been answered
- After the previous point, the user must give a name
- The program should then create and save a new character with the new name and askedTraits as traits.
- After a character is added, the active user's contributions must increase by 1.

Confirmation:
- You must be able to create a new character with a picture and name

- The new character must have the trait values that have been answered during the game
- User contributions must increase by 1

User Story 10:
Estimated time 3 days
Description:
As a customer, I want to be able to see a leaderboard that can switch between different categories

Tasks:
- Leaderboard listing category information
- Create a controller and view that communicates with to display the leaderboard.

Confirmation:
- You must be able to browse between categories
- You should be able to load a category and see it

User Story 11:
Estimated time 2 days
Description:
As a user, I want to be able to browse between different categories in the leaderboard
Tasks:
- Category that lists Users by most contributions
- Category that lists by number of times a character has been found
- Characters that are placed far down on the leaderboard must have a higher rarity
- To be updated when new information is added

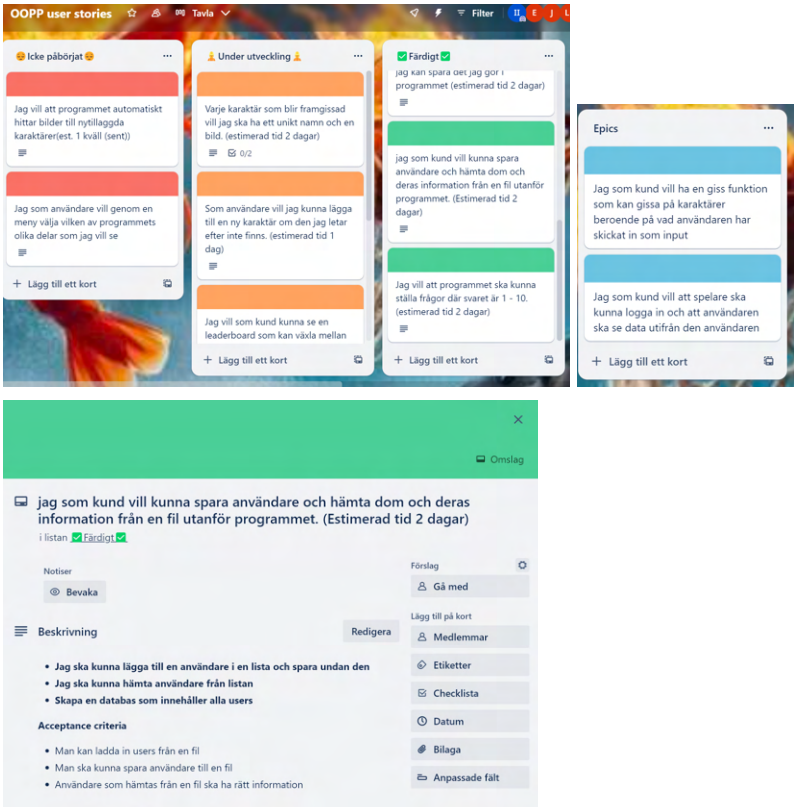
Confirmation:
- When new information about the content of the categories is updated, they must be updated
- There are different categories that you can switch between. (Ex the bridge pattern)

User Story 12:
Estimated time day
Description:
I, as a user, want to select through a menu which of the different parts of the program I want to see
Tasks:

- skapa en meny med olika knappar för olika delar
- koppla knapparna så att dom går till olika delar av programmet

Confirmation:

- There should be some sort of menu that you can navigate around



Picture of user stories.

2.2 Definition of Done

A user story is done if it fulfills all of the following conditions:

- Passed all tests
- Is merged in git with all (if any) conflicts resolved
- Meet all acceptance criteria
- Is accepted by another group member
- Has been tested and survived user feedback*

2.3 User Interface

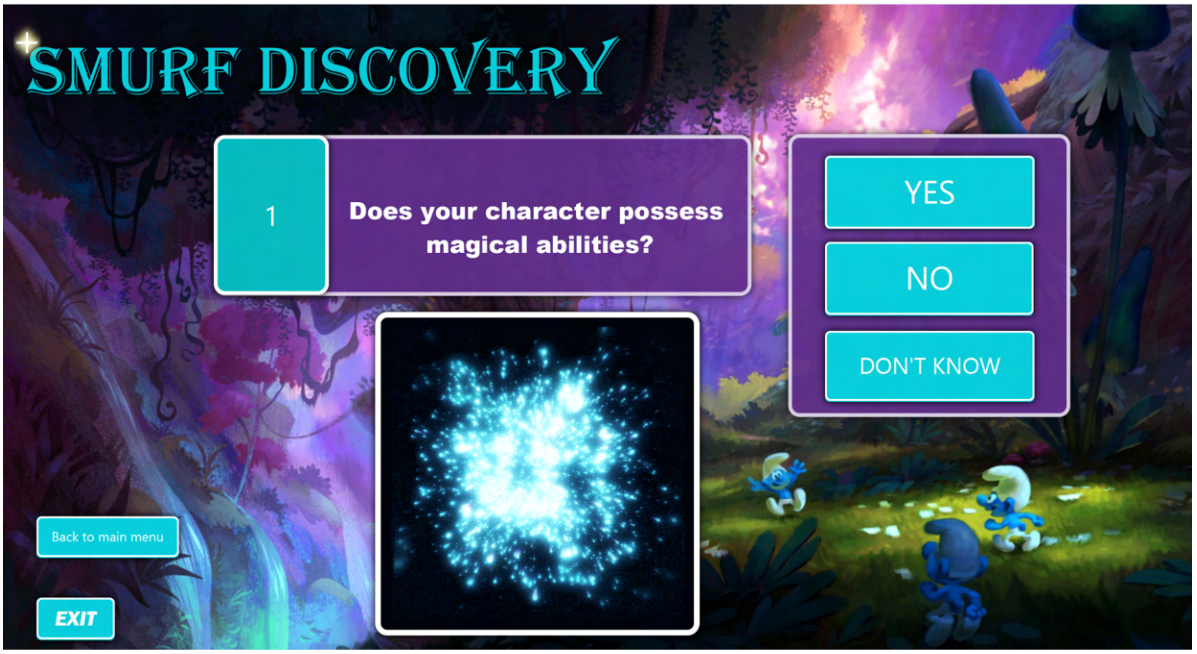
The current version of the program contains five different views, the login screen, the smurfinator game, the main menu, the achievements and the leaderboard. The first screen that the player is prompted with is the login page, where the user has to login to be able to play the game. If the user does not have an account they can create one by pressing on the button that displays “create new user”. When the user is logged in the main menu is displayed where they can navigate to the different parts of the program. If the play button is pressed the smurfinator main game starts and the player can start guessing. If the program makes a guess on a character the player can either confirm that it was the correct character, return to the main menu or create a new character. When answering questions the player can choose between the answers yes,no or don’t know if it is a multiple choice question or a number between 1 and 10 if it is a ranged question.

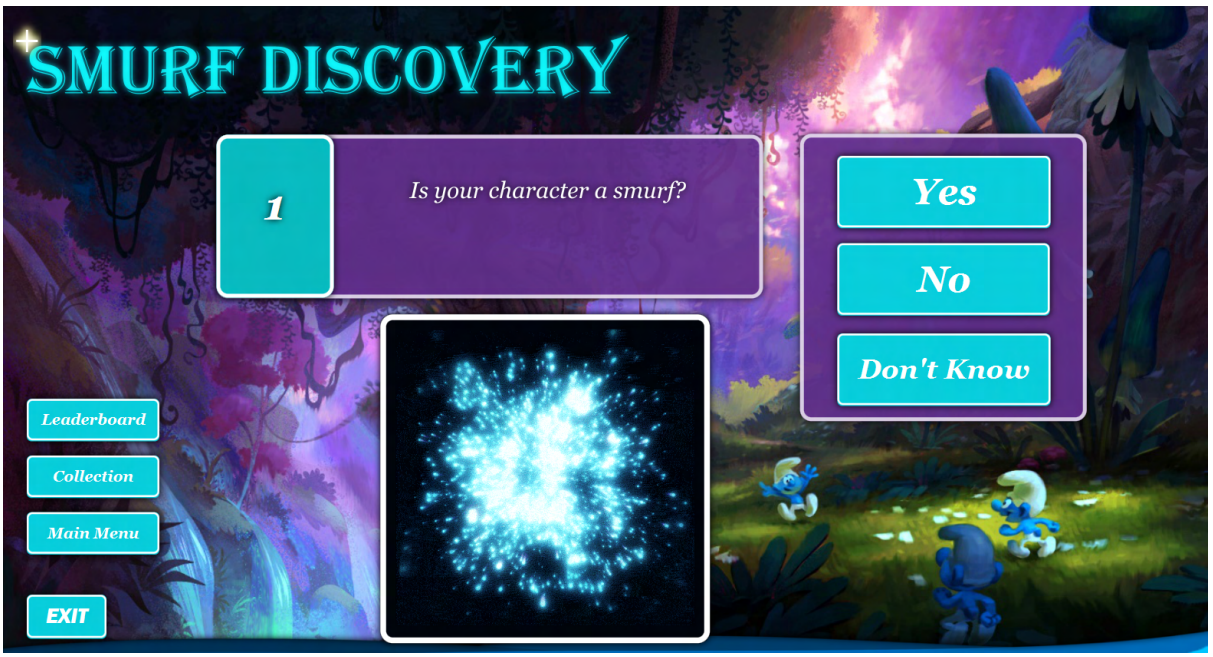
If the player presses the leaderboard button the leaderboard screen is shown where the player can decide what kind of category to be displayed, which will be displayed in the middle of the screen.

The last screen is the achievements screen which goal is to show the player what characters that it has found through playing the game. The characters are blacked out until they have been found at which point they will be “revealed”.

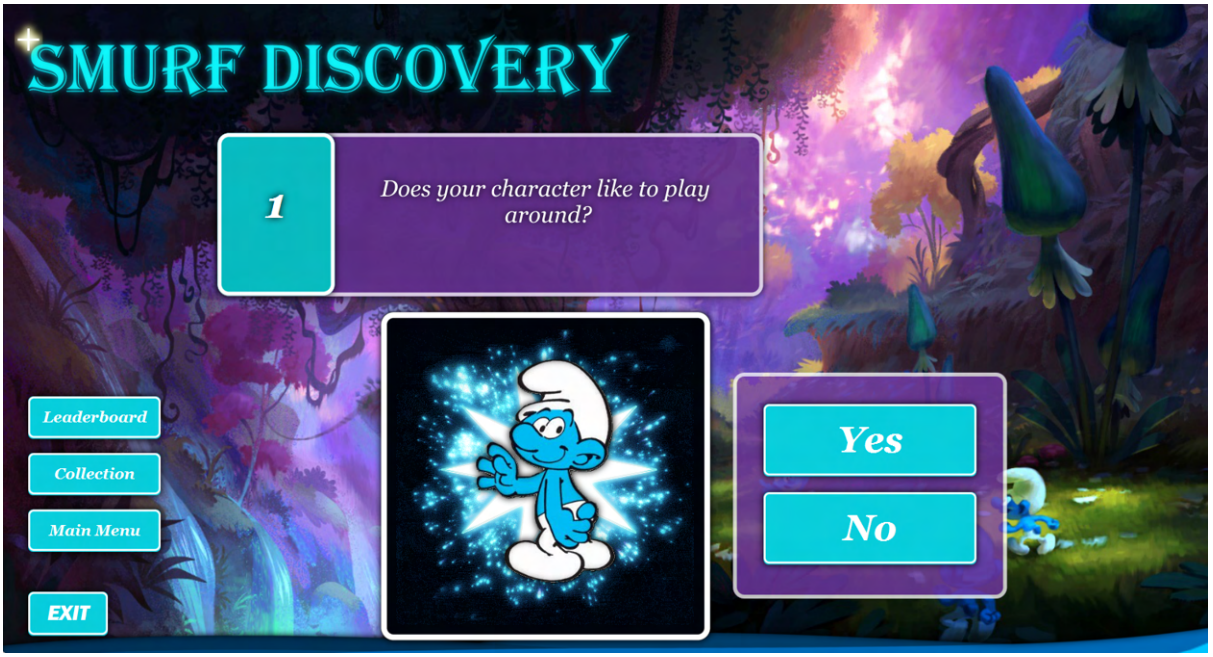
Figma prototype of the smurfinator view

Implementation of the smurfinator view

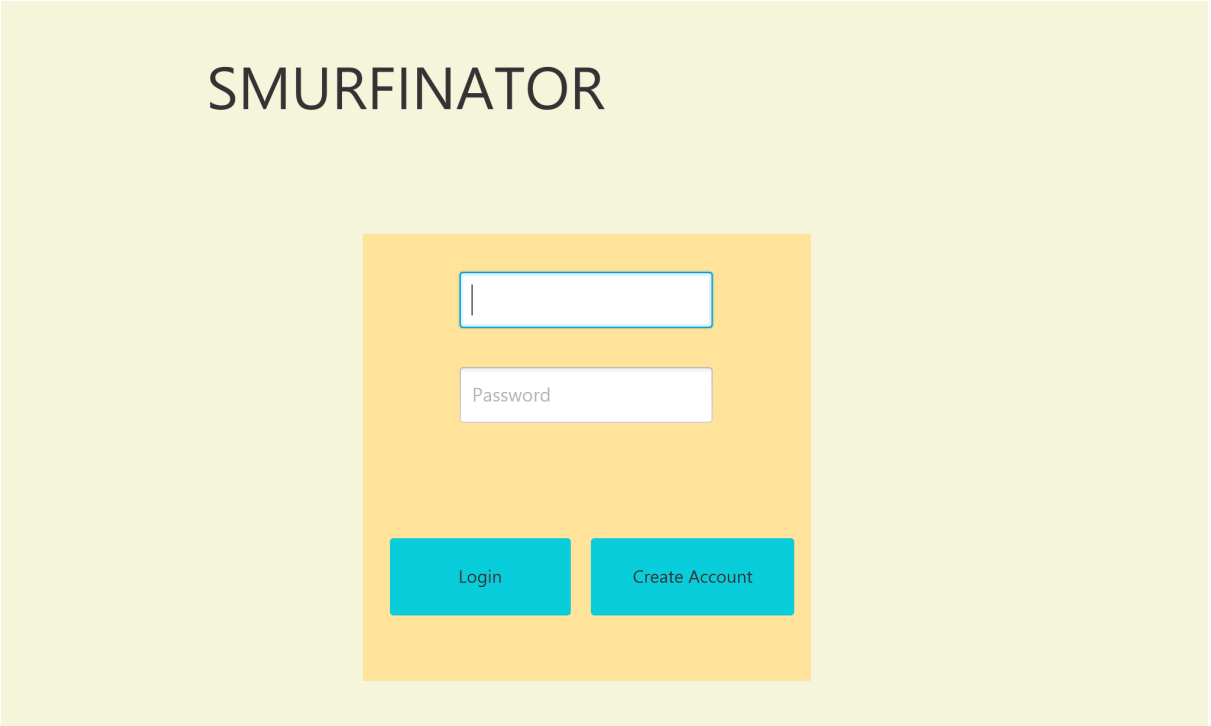




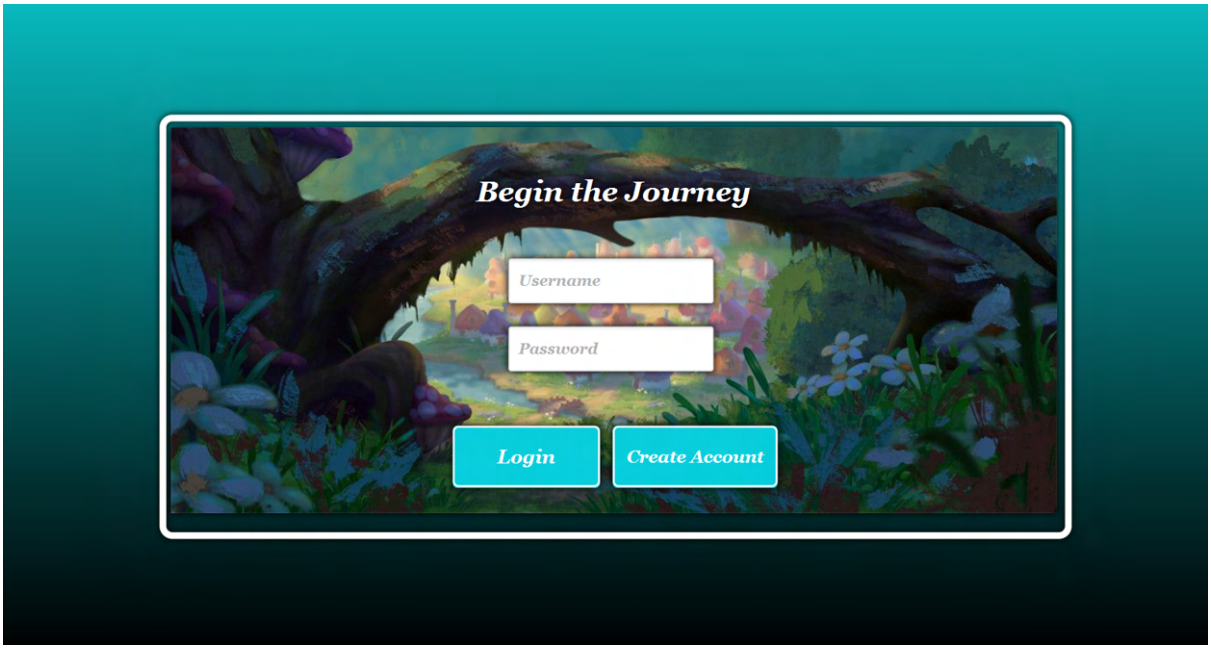
Most recent implementations of the smurfinator view



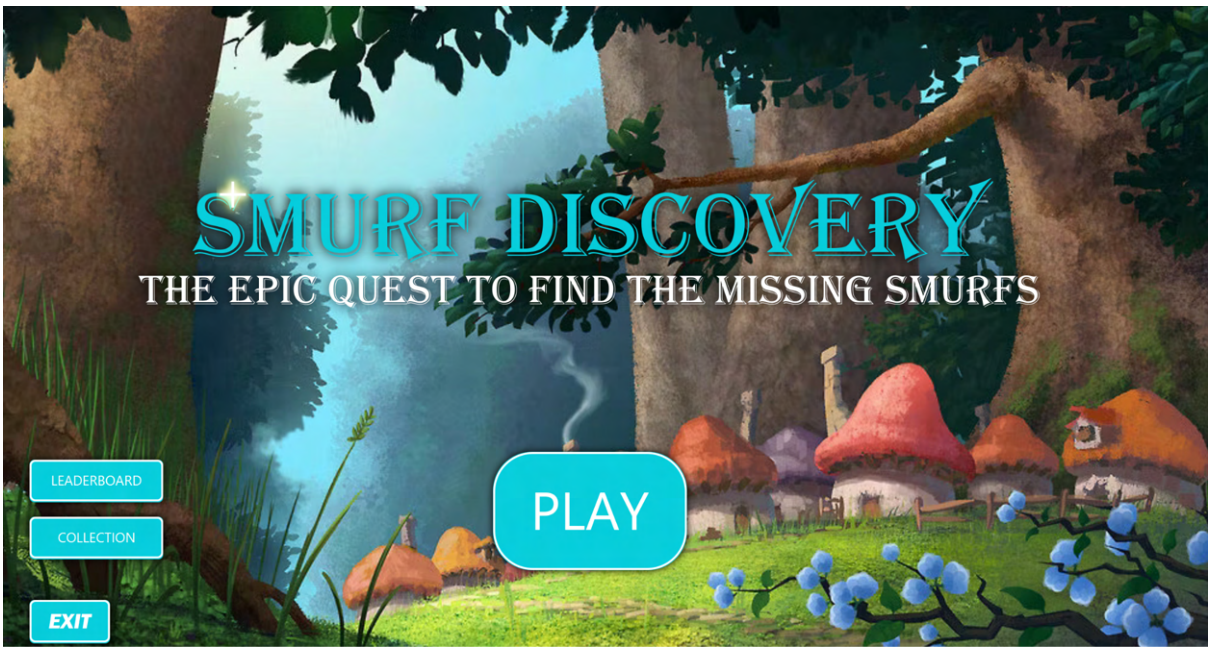
The view when a smurf have been discovered



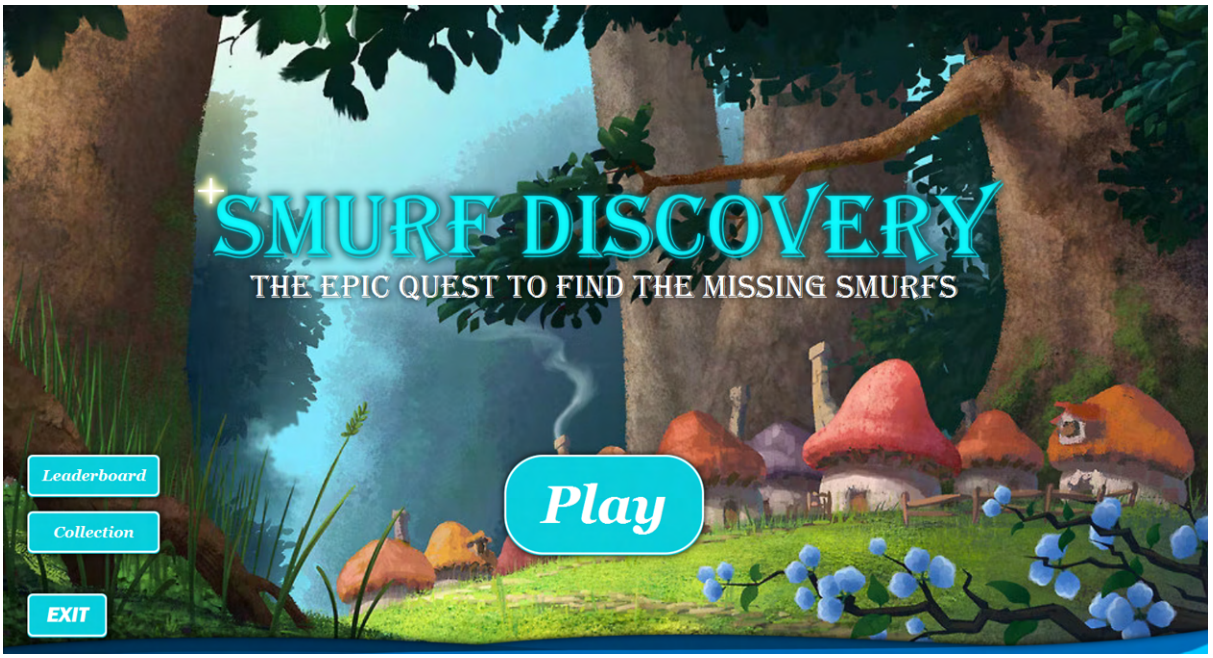
Implementation of login screen



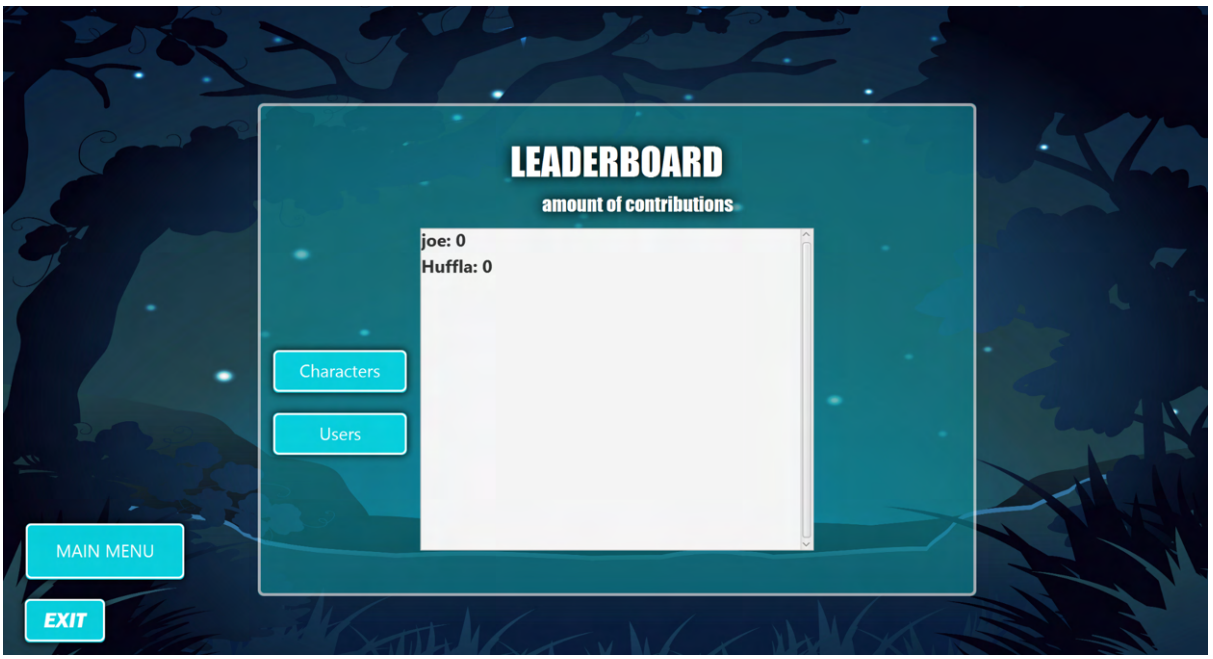
Most recent implementation of the login screen



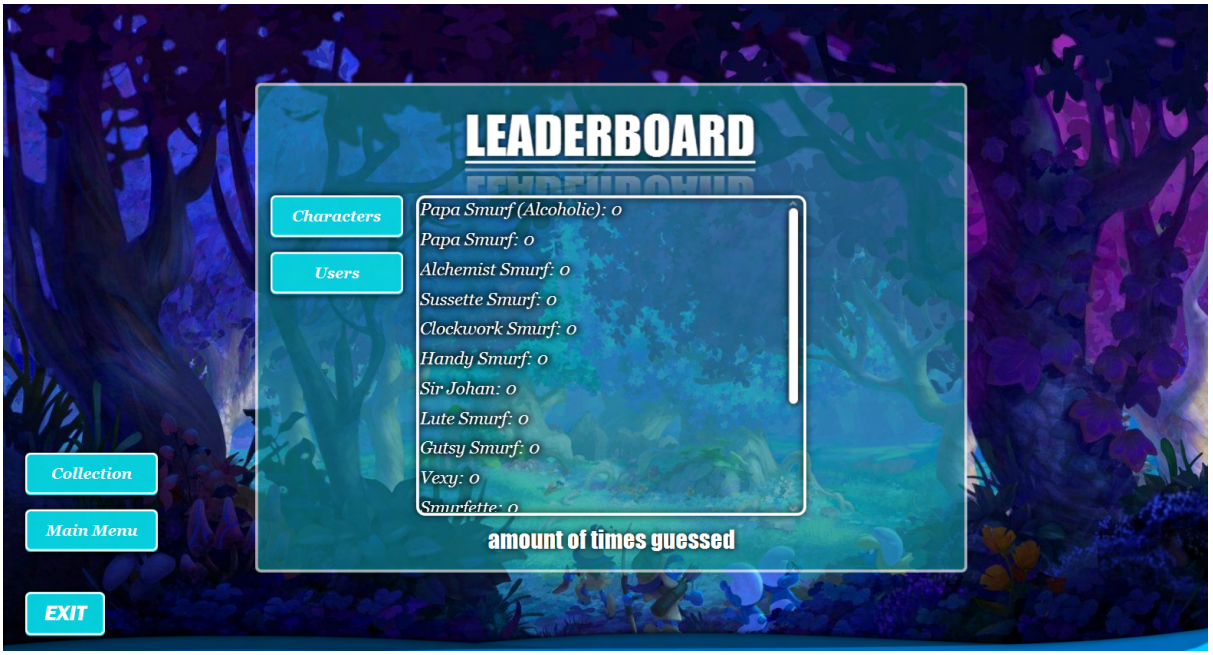
Implementation of main menu



Most recent implementation of main menu



Implementation of leaderboard



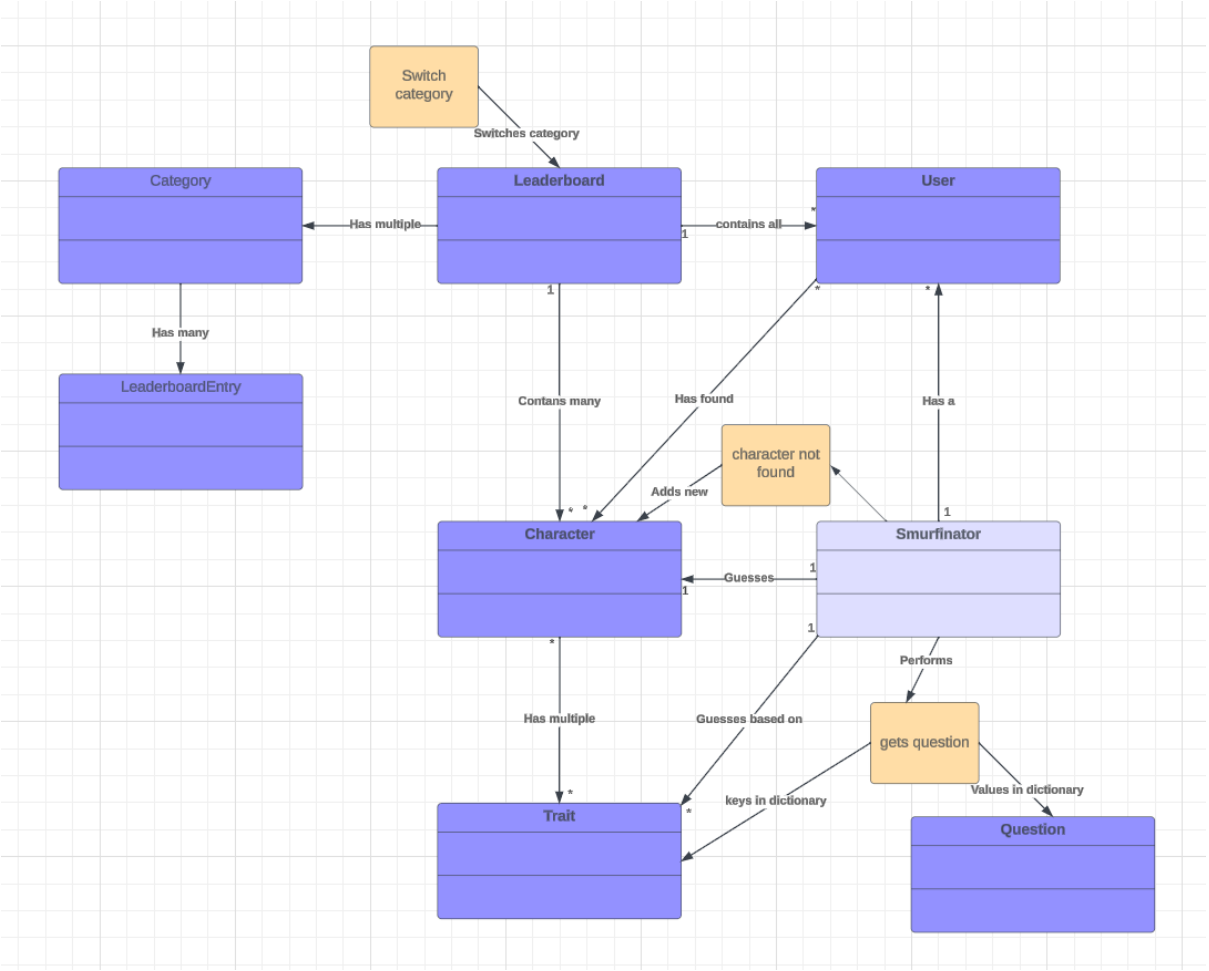
Most recent implementation of leaderboard

Implementation of achievements

Most recent implementation of achievements

3 Domain model

Below is the domain model that represents a high level overview of the application.



Picture of the domain model!

3.1 Character

This class is responsible for representing each character that the program can guess on. It contains the name of the character, the path to the image representing the character aswell as the traits that represents the character. It has methods for getting these different attributes. It also contains an attribute for the amount of times a character has been guessed on which keeps track of how rare it is.

3.2 User

This class represents an account in the game which contains a name as well as a password and the amount of contributions that the user has added. Since it represents an object and not a handler the only methods it contains are getters and one increase method that increases the number of contributions.

3.3 Traits

Traits are the class that represents simple traits such as “Animal” or “intelligent”. It also contains a list of Traits that are considered to be opposite of that trait.

3.4 Smurfinator

Smurfinator is the main model for the part of the program that handles the logic behind the guess system as well as handles the communication between the view and takes input from the controller.

3.5 Leaderboard

Leaderboard is the main model that handles the leaderboard logic, it communicates with the logic view and takes signals from the logic controller. Manages which category which is shown by changing states of the leaderboard.

3.6 Question

Question contains the text and type of the question that is asked. They are the values in a dictionary where the traits are the keys in smurfinator.

3.7 Leaderboard Entry

An entry that categories are displayed in the leaderboard. Can contain users or characters.

3.8 Category

An object that represents the entries of the leaderboard specified to a category. Can contain user leaderboard entries or character leaderboard entries.

4 References

4.1 Production Tools

- Github: Used to share code between each member with Git.

- Lucidchart: Used to create our UML chart and our domain model.
- IntelliJ/VSCode: IDE used to write the code.
- Trello: Application used to manage userstories.
- Figma: Application used to create first digital prototype.

4.2 Communication Tools

- Discord: Communication between members.
- Slack: Communication between others.

4.3 External Libraries

- JavaFX
- Maven