French - Azerbaijan University

Software Engineering

LOVE LETTER GAME PROJECT

Group members: Nadir Abdullayev

Rufat Huseynov

Majid Qurbanli

Gulshan Mustafayeva

Javid Guliyev

_____ Mayl 20, 2020 _____

• What was the purpose?

The purpose of the course project is to be provided with the knowledge of software engineering methodology and the skills to apply it. The particular project is not the goal in itself; rather, it serves as a vehicle to apply the knowledge and to develop the skills. As previously stated, the goal of this project is to create a game that is played by 2 or more people using the necessary steps. The implementation of the project requires two mandatory modules: core software and GUI.

• How did we solve different things, Which technologies were used, Which problems we faced

In Front-End part

In this part of work, before get down to business , Bootstrap was learned, which is a framework to help to design websites faster and easier, including HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels, etc. It also gives you support for JavaScript plugins. **jQuery** is also used for its main **advantages** such as : light weight comparing to other **javascript** libraries. it has a wide range of plugins available for various specific needs. it is easier for a designer to learn **jQuery** as it uses familiar CSS syntax. Possession of this knowledge helped us to avoid problems in front-end part.

In Back-End part

The Back-end was written on Node.js, which is a Javascript run-time environment that helps in the execution of JavaScript code server-side. It is an open–source cross-platform JavaScript that helps in the development of real-time network application. The main reason why we chose exactly Node.js is that it has advantage such as : it offers the writing the server-side applications in the JavaScript. This allows us to write both the front-end as well as the back-end web application in JavaScript using a runtime environment. And that's why no need to
use any other server-side programming language. It also makes the deployment of the web applications simpler because almost all the web browsers support JavaScript. Because of benefits towards online multiplayer games we used socket.io that allows the interaction in the game, the data is sent to the server, and the server broadcasts it to the other players in the game.

About Game's logic

Firstly, when / is accessed, the player gets a fake game screen with the Love Letter logo, then after 2 seconds, everything becomes blurry and a modal appears simultaneously. The modal asks for the username. After the username is entered and the button is clicked, the modal's content changes (fade out-in) to Welcome <nickname> with two buttons:

1. **Create a game room**
2. **Enter a game room**

Create a game room changes the content of the modal to "How many

players you want in the game?” and “Create!” button. Then the content of the modal changes to a table that has the list of the players. From that point, the room chat is accessible. If no players are left in the room, the room gets eliminated.

Enter a game room changes the content of the modal to “Enter the room code:” and input with a button “Enter!” below it. After that, the modal goes away and the player sits and waits until everyone’s connected. (mb add a feature to close the room to open a new one w a different number of people or even AI?).

We might add a modal history so that it would be possible to go back to the previous modals (only possible if a room is not entered). It would use a swiper and the history would be saved simply as an array. If the player goes back too much and then goes forward to something new, then the forward history is first cleared, and only then the new modal window is created and appended to the swiper. The modal would have arrows just like Instagram has them to the sides of its posts.After all the players are in the room, the game waits until everyone texts */ready* in the chat to begin. The Chat and the Action Window are both pivoted to
the center of the screen. *The Action Window* will appear when an action is required. *The cards in hand* won’t be draggable but rather clickable. That saves dev time and actually is a good UX decision. Again, on the Action Window, you will choose the player you want target or any other action that is required. After the interaction with the Action Window has ended, the chosen card will slide down (go behind the screen) and stay there until the turn of the current player begins again. When a card from the deck gets drawn, it merely slides down (off- screen). All animations will be in CSS because they don’t require much processing power and they are not simultaneous which ensures that they will not lag. If the game is 1v1 and the other player leaves the game, the game ends (saving dev time).

How the chat works:

Each room has its own chat array. When a user is connected, they receive all the messages in the chat and show the chat for 10 seconds. After 10 seconds, the chat gets hidden. After the initial chat get, all messages are transferred using sockets and the array is not sent again but only updated in the room on the server. All new messages appear in this manner: Html is added. After that, the height of it goes from 0 to the normal height with 0.2s duration.

How the game works:

If previously a played had left the game, then only a player with the same username as theirs is accepted . If the game has not started yet, then the room modal is shown *Room modal* will only show a table of the player nicknames and statuses, as well as the code of the room. Under the table will be a button *Ready*/*Not Ready* depending on the state of the player. The host will get *Start the game* button when all the players are ready (the host will not have the Ready button but rather only have Start the game button which will be disabled unless all the players are ready (obviously except the host)). On the backend: clicking start the game will only send to the server the same thing as all other players do: ready. The server will check after each status change whether all the players are ready and will start

the game. If the host leaves the room, then another player will become the host or if there are no players left, then the room will be removed. If the game is already started and it's not a break time, then *Wait modal* will be shown. It will only say that the game is still going, please wait. It will also show the player's nicknames and tokens.

Creating a room: When a player creates a room the following happens: A room with a unique *room code* is created and the player is connected to it. The roomCode is sent to the host and at the front: The roomCode is added to the URL of the user (it's written as uppercase but it doesn't matter what the size is) The *Room modal* is shown to the user The chat is shown to the user

When someone disconnects:

- If the game was on a break (in between rounds or before starting/after ending of the game), then the player simply gets removed from the room
- If the game is going, then the round ends, no one wins that round, the break starts, *Disconnected modal is shown* (At seems <username> has disconnected
The round is prematurely ended
No one wins the round
OK button). Clicking OK button shows the *Break modal.*

Break modal:

Same as the room modal, except for the host it shows start new round button (for start modal it was start the game button)

- **UML Class diagram:**

• What did we gain working on this project

This project is designed to put into practice the knowledge and techniques acquired during courses However, it can also allow us to experiment with new topics, techniques and tools if they are more relevant to the task, are more interesting for the future career, or simply more fun. For example we acquired with new thinks that we didn’t know before, some of us learned new

language JS, others improve their knowledge.

• Which improvements can be done in the future

The game is NOT for mobile (at least for now, not enough time for it), after about 1200x500, the screen stops getting smaller and scrolls appear . In the future the game can be avaible for both computer and phone screen.