HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Hópur 11
Verklegt námskeið 1
T-113-VLN1
Design report

Tristan Snær Danielsson
Jeremias Borjas Tablante
Arnór Eyþórsson
Viktor Yngvi Ísaksson
Kristófer Kristján Erlendsson

Table of contents:

Foreword

Introduction

This design report serves as the main reference document for the development of the e-Match Bookings Records System, created for RU's Glorious e-Sport Extravaganza. Its purpose is to give the development team — and any future developer who reads it — a complete, structured overview of how the system is designed, how it should function, and how the implementation must follow the 3-layer architecture required by the course.

The document is organized so that each part of the system can be understood independently but also together as one complete blueprint. Developers should use this report throughout the entire project cycle: during planning, coding, testing, and verification. Each section has a specific role:

The System Overview explains the fundamental purpose and scope of the system.

The Requirements section defines what the system must do (A-requirements) and what it should or could do (B/C-requirements).

Each requirement is expanded with detailed Use Cases and Happy Paths that describe exactly how the system should behave from the user's perspective.

UI Wireframes show the structure and flow of the text-based interface.

Class Diagrams describe the internal system architecture and how data moves between components.

The Testing Methods section links requirements to how they are verified during development.

Finally, Other Work Documents contain supporting material used throughout coding and testing.

The report is designed to be read both sequentially and as a reference manual. Developers may jump directly to the section relevant to the task they are working on — such as UI building, data-layer mapping, or implementing specific features — while instructors can evaluate whether the implementation matches the design.

All team members must keep this document updated as the project evolves. Any change in logic, structure, or UI must also be reflected in the design report so it always matches the codebase. This ensures clarity, consistency, and a smooth development process.

System Overview

The purpose of this system is to manage and run a single-elimination tournament with 16 registered teams. Also, as a B requirement, a double elimination and last man standing tournament. The system must allow the organizer to create a new tournament, load the initial team list from a text-based file, schedule all matches, record match results, and automatically update the tournament bracket after each completed game.

The tournament progresses through predefined knockout rounds:

Round of 16 – 8 matches

Quarterfinals – 4 matches

Semifinals – 2 matches

Final – 1 match

This structure produces a total of 15 matches.

The system maintains the full tournament state throughout the event. After each match result is entered, the system updates the bracket, moves the winning team into the next available slot in the next round, and displays an updated scoreboard. Users should be able to view both completed matches and upcoming matches at any time.

The system uses a text-based user interface and requires minimal input from the user. Match outcomes are entered manually by the organizer, and the system ensures that the tournament schedule remains valid according to the predefined rules (e.g., no team can play two matches at the same time, all matches must occur within the tournament's date range, and server availability must be respected).

By the end of the tournament, the system will have produced a complete and consistent tournament record, including the final champion and updated results stored in text-based files.

Base Flow – Match Result Processing

The core flow of using the system should be:

Organizer should:

1. Create a tournament name.
2. Import 16 team names from a file
3. Create a tournament schedule.

Captain:

1. Register his players by inputting his team name (should already be in memory from organizer import)
2. Modify player info by player ID (should already exist in memory).

Player:

1. Modify personal info by ID.

Spectator:

1. View upcoming matches.
2. View completed matches.

Start of Tournament:

The system loads 16 teams from a .txt data source.

The teams are placed into the Round of 16 bracket.

Match Execution

Each match result is registered by user input as the organizer.
The losing team is eliminated from the score board immediately.

Scoreboard Update

After each match, the winner is placed in the next available slot in the next round.
The losing team is removed from the bracket.
The system prints the full updated bracket, allowing the user to track tournament progression.

Continuous Progression

This process repeats for:

8 matches (Round of 16)
4 matches (Quarterfinals)

2 matches (Semifinals)
1 match (Final)

Tournament Completion

After the final match, the system prints the champion.

Team Contract

Team Contract – 3-Week Project

Working Hours & Attendance:

Project Duration: 3 weeks
Agreed Working Days: Mon-Fri.
Daily Start Time: 10:00
Daily End Time: 17:00 or later if needed

Attendance Expectations:

- Everyone shows up on time (within 30 minutes).
- Notify the team 1 hour in advance if late/absent.

Rotating Roles & Responsibilities:

Daily note logging, diary:
- A diary is kept and filled out at the beginning of every day.

Communication Rules:

- Primary communication tool: Discord.
- Communicate respectfully and professionally.

Work Quality & Deadlines:

- Deadlines decided as a group.
- Deliver work on time and to agreed quality.
- Ask for help early if stuck.

Conflict Resolution

1. Discuss openly as a group.
2. Suggest and agree on solutions.
3. If unresolved, contact the instructor.

Accountability:

Everyone agrees to follow this contract.
Repeated failure will be addressed by the team
and escalated to the instructor if needed.

Signatures:

Name: Jeremias Borjas Tablante

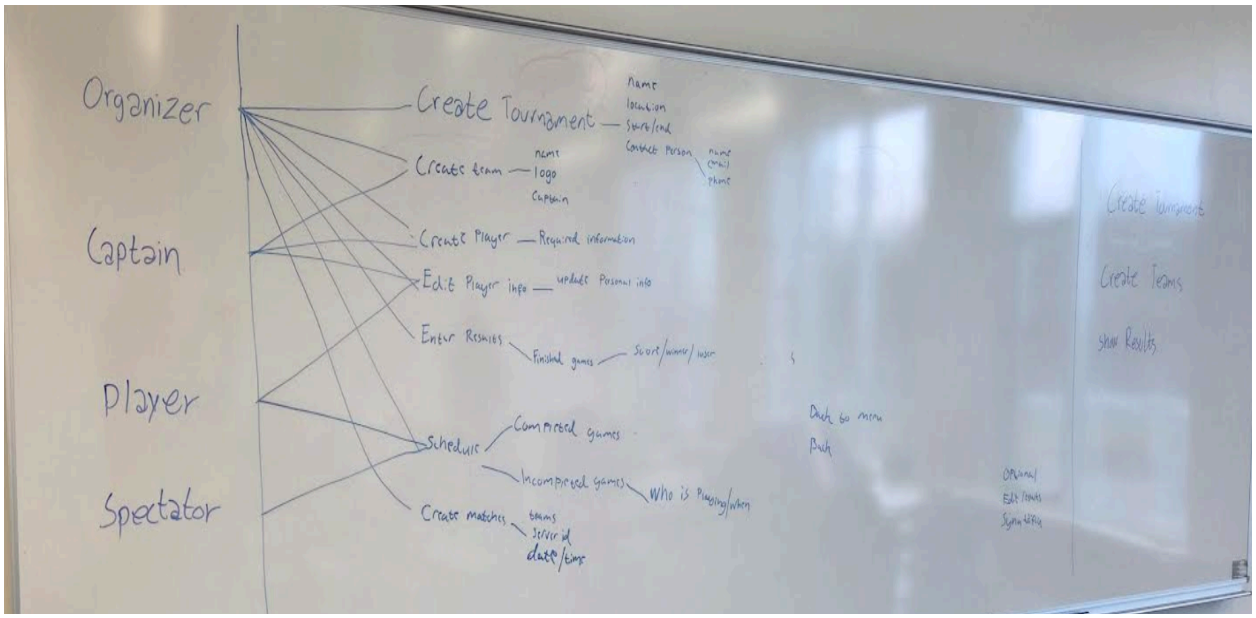
Name: Arnór Eyþórsson

Name: Tristan Snær Danielsson

Name: Viktor Yngvi Ísaksson

Name: Kristófer Kristján Erlendsson

Initial Mapping



Requirements

List of requirements A

Functional requirements

Number	Description	User group	Priority	Additional info
R1	As a Organizer or the team captain i want to be able to create a new player so that i can add the player to a team.	Organizer, Team captain	A	Player must belong to exactly one team, team leader can only add to his own team, but the organizer can create and add player to any team.
R2	As the team captain i want to be able to edit the players personal information.	Team Captain	A	Captain can only edit players in own team.
R3	As the organizer i want to be able to change the team captain so that the teams can have a different captain if they want to.	Organizer	A	Captain must be a player in the team.
R4	As the organizer i want to be able to create a new tournament with the predefined team list so that the tournament can be scheduled.	Organizer	A	Requires name, start date, end date, venue or location and contact person, predefined teams.txt file.

R5	As a user i want to be able to view the standings so that i can see how the tournament is going.	All users	A	Completed matches show results and incompleted show who plays and when.
R6	As the organizer i want the system to register a team automatically so that when a tournament is created the predefined list of teams will automatically be registered without manual input.	Organizer	A	16 teams minimum in the bracket tournament system.
R7	As the organizer i want to be able to generate a schedule based on the type of tournament so that the tournament can begin.	Organizer	A	The schedule must respect server limitations.
R8	As the organizer i want to make sure no team is scheduled to play more than one game at a time so that the tournament can run correctly.	Organizer	A	Implemented during match scheduling (R10) to validate time and slot conflicts.
R9	As the organizer i want the system to assign predefined dates, time and unique server IDs to each match so that all matches are properly scheduled for a 3 day period.	System	A	Server availability.

R10	As the organizer i want to enter match results so that the tournament standings are updated.	Organizer	A	Update bracket and standings.
R11	As a user i want to see the team information so that i can see players in each team.	All users	A	View team names, handles and name of players.
R12	As the organizer, I want to generate a double elimination tournament so that teams get a second chance even if they lose once.	Organizer	B	The tournament shall consist of a Winner Bracket and a Loser Bracket. Teams that lose once move to the loser bracket. Teams that lose two matches are eliminated from the tournament.The winner of the loser bracket plays against the winner of the winner bracket.
R13	As the team i want to be able to get points if they win the tournament so that i can keep track of our wins on tournaments.	System	B	Tournament winner gets 1 point.
R14	As a spectator, captain or a player i want to be able to view team clubs information so that i can see their stats.	All users	B	Clubs will have their own points that will be composed of the sum of each team.

				Not all teams will have clubs.
R15	As the organizer, I want to generate a last team standing tournament so that teams get a second chance even if they lose once.	Organizer	B	The tournament shall be held with the number of teams that have entered and the winner will be awarded a point.
R16	As a user i want to be able to view the players profile so that i can see information about the player.	All users	B	Only name, handle and team visible for public.

Non functional requirements

Number	Description	Priority	Additional Info
N1	The UI should be text based.	A	Required
N2	3 tiers architecture.	A	UI,Logic,Data layers
N3	Data will be stored in text based files.	A	CSV or JSON

Use cases

Blueprint.

Use case name:	
Number:	
Priority:	
User story:	
Precondition:	
Basic flow:	
Alternative flow:	
Post condition:	
Actor(s):	
Author(s):	

Use case name:	**Create a new player**
Number:	R1
Priority:	A
User story:	As a Organizer or the team captain i want to be able to create a new player so that i can add the player to a team.
Precondition:	The user is The organizer or Team captain. The team must exist before registering a new player.
Basic flow:	1. The user selects create new player from the menu. 2. The system shows a list of available teams. 3. The user selects a team for the new player. 4. The system asks for the required information (Name, date of birth, address, phone, email, social media URL, handle (username)). 5. The system validates the information. 6. The system assigns the player to the team and stores the information. 7. The system confirms the player has been registered.
Alternative flow:	1A. Invalid input. - Error message saying required information is not filled.

	2A. The username already exists. - Redirecting the user to pick a new username.
Post condition:	The player has been registered. The player is assigned to a team.
Actor(s):	Organizer, Team Captain
Author(s):	Tristan

Use case name:	**View player profile**
Number:	R2
Priority:	B
User story:	As a user i want to be able to view the players profile so that i can see information about the player.
Precondition:	The player exists and is assigned to a team.
Basic flow:	1. The user selects spectator from menu. 2. The user selects view teams. 3. The system asks the user to input the player id. 4. The system displays the player profile with name, handle, stats.
Alternative flow:	1A. player id not in the system. - Ask for a valid player id.
Post condition:	The user has viewed the players public information.
Actor(s):	All users
Author(s):	Tristan

Use case name:	**Edit players information**
Number:	R3
Priority:	A
User story:	As the team captain i want to be able to edit the players personal information.
Precondition:	The user is the team captain and only change information of his own

	teammates. The player is registered. The team captain can only change players information in his own team.
Basic flow:	1. The player is already registered. 2. User selects captain. 3. Selects edit specific player. 4. User gets a list of teams. 5. The captain pick his own team. 6. The system show a list of players on the chosen team. 7. The user picks a player. 8. Edits info. 9. Validates info and saves changes if user selects “go back to main menu” or “go back”.
Alternative flow:	1A. Missing data. - The user needs to fill all the required information. 2A. invalid input. - Error message saying to only pick a team from the list. - Error message saying to pick a player of the shown list.
Post condition:	The captain can update his players information.
Actor(s):	Captain
Author(s):	Tristan

Use case name:	**Change the team captain**
Number:	R4
Priority:	A
User story:	As the organizer i want to be able to change the team captain so that the teams can have a different captain if they want to.
Precondition:	The team exists. The team has at least one player.
Basic flow:	1. The user clicks change captain from the menu. 2. The system shows a list of teams. 3. The user clicks the team they want to change. 4. The system shows a list of players and the current captain. 5. The user selects the player will be the new captain. 6. The system validates that the chosen player belongs to the selected team. 7. The system confirms the change and updated the data and

	redirects to the main menu.
Alternative flow:	1A. There is no player in the team. - Shows an error saying that at least one player needs to be assigned to the team.
Post condition:	The team has a new captain assigned to their team. The data has been updated.
Actor(s):	The organizer
Author(s):	Tristan

Use case name:	**Create a new tournament**
Number:	R5
Priority:	A
User story:	As the organizer i want to be able to create a new tournament with the predefined team list so that the tournament can be scheduled.
Precondition:	The tournament name does not already exist.
Basic flow:	1. Click create tournament. 2. The system shows a list of possible type of tournament. 3. The user chooses between a knockout tournament or double elimination. 4. The system requests required information: tournament name, location/venue, start and end date, contact person(name, email, phone). 5. The user inserts the information. 6. The system validates the data and that the duration is exactly 3 days, the end date is two days after the start date. 7. The system looks like this - Day 1 start date - Day 2 start date + 1 day - Day 3 start date + 2 days 8. The system automatically calls R8 to register the teams. 9. The system calls the teams text file and registers all 16 teams into the tournament. 10. The system shows that the tournament has been created.

Alternative flow:	1A. invalid name. - Error message saying the team name already exists and to pick a different name. 2A. End date is not exactly two days after the start date. - Message saying the tournament must be 3 days long. 3A. missing information. - Missing the required information, asks to fill in the required information.
Post condition:	The tournament has been created and is in the three day structure. 16 teams are registered. Tournament is ready to generate the schedule. This use case automatically triggers R7.
Actor(s):	Organizer
Author(s):	Tristan

Use case name:	**View tournament schedule**
Number:	R6
Priority:	A
User story:	As a user i want to be able to view the standings so that i can see how the tournament is going.
Precondition:	The tournament has been created. At least 16 valid teams have been registered. Schedule must be generated.
Basic flow:	1. The user selects view tournament schedule. 2. The system shows a list of tournaments. 3. The user picks a tournament. 4. The system displays the tournament information (name, date, location). 5. The system shows both completed matches and incomplete matches. 6. If the match is incomplete show who is about to play and when. 7. The system displays the current tournament standings.
Alternative flow:	1A. No matches are scheduled. - Show a message saying there are no matches scheduled yet. 2A. No results have been entered. - The system displays the match schedule without standings. 3A. Tournament not found.

	- Error message saying that no tournament is found and return to the tournament selection.
Post condition:	All users can view current standings and the tournament schedule.
Actor(s):	All users
Author(s):	Tristan

Use case name:	**Automatic team registration into tournament**
Number:	R7
Priority:	A
User story:	As the organizer i want the system to register a team automatically so that. when a tournament is created the predefined list of teams will. automatically be registered without manual input.
Precondition:	The tournament exists. The predefined teams text file exists. The tournament schedule has not been generated yet.
Basic flow:	1. The system reads the team list text file. 2. The system validates that there are exactly 16 teams. 3. The system makes sure there are no duplicate of teams. 4. The system automatically registered the 16 teams into the tournament. 5. The system confirms the 16 teams have been registered.
Alternative flow:	1A. more or less than 16 teams in the text file. - Make sure that there are 16 team in the text file. 2A. Duplicate teams found. - Make sure no team has the same team name.
Post condition:	16 teams have been registered into the tournament.
Actor(s):	Organizer
Author(s):	Tristan

Use case name:	**Generate Tournament schedule**
Number:	R8

Priority:	A
User story:	As the organizer i want to be able to generate a schedule based on the type of tournament so that the tournament can begin.
Precondition:	The tournament exists. 16 teams have been registered. No schedule has been generated yet.
Basic flow:	1. The system show a list of tournaments to generate a schedule for. 2. The user selects which tournament to generate a schedule for. 3. The system checks if 16 teams have been registered. 4. The system randomly pairs the 16 teams to 8 different matches. 5. The system creates a knockout bracket with the first round of 8 then quarterfinals 4 games and then semifinal 2 games and then the final game. 6. The system stores the schedule in a text file. 7. Display generated bracket.
Alternative flow:	1A. Less than 16 teams registered. - Message saying 16 teams must be registered to generate schedule. 2A. the schedule already exists. - Then just a message saying a schedule has already been created.
Post condition:	Knockout schedule created. The tournament is ready to start.
Actor(s):	Organizer
Author(s):	Tristan

Use case name:	**Prevent teams playing two matches at the same time**
Number:	R9
Priority:	A
User story:	As the organizer i want to make sure no team is scheduled to play more than one game at a time so that the tournament can run correctly.
Precondition:	The tournament schedule is created.
Basic flow:	1. The system checks if the team is scheduled for more than one match at a time. 2. If both teams pass the check then the match is scheduled. 3. If one or both teams are already assigned the system selects a different matchup.

Alternative flow:	1A. No valid matchups. - Show an error and abort scheduling.
Post condition:	No team appears on more than one match at a time.
Actor(s):	Organizer
Author(s):	Tristan

Use case name:	**Assign date, time and server to matches**
Number:	R10
Priority:	A
User story:	As the organizer i want the system to assign predefined dates, time and unique server IDs to each match so that all matches are properly scheduled for a 3 day period.
Precondition:	The tournament has been created with a valid start and end date. The end date is exactly two days after the start date. The system looks like this - Day 1 start date - Day 2 start date + 1 day - Day 3 start date + 2 days Knockout schedule generated. A predefined list of time slots for each day exists. A predefined list of servers exists.
Basic flow:	1. The system goes through all scheduled matches. 2. The system assigns matches from the round of 16 to day 1. 3. Quarter finals for day 2. 4. Semi final and final to day 3. 5. For each match has to select the next available time slot for that day and an available server. 6. The system makes sure that no server is assigned to two matches at the same time and no teams are scheduled in the same time slots. 7. The system confirms that the full schedule has been created.
Alternative flow:	1A. Not enough time slots. - Error not enough time slots for all matches, make sure that we have enough time slots for the matches. 2A. Not enough servers. - Error not enough servers for the selected time. 3A. tournament dates were invalid. - Tournament must be 3 days long.

Post condition:	Every match has been assigned a calendar date based on the tournament start date, time slot and a unique server id. The schedule is ready to start the tournament.
Actor(s):	Organizer
Author(s):	Tristan

Use case name:	**Enter match results**
Number:	R11
Priority:	A
User story:	As the organizer i want to enter match results so that the tournament standings are updated.
Precondition:	The tournament has been created. The knockout schedule exists. At least one match were the results have not been entered.
Basic flow:	1. The organizer chooses enter match results. 2. The system shows a list of ongoing tournaments. 3. The user picks a tournament. 4. The system shows a list of unfinished matches. 5. The organizer selects a match from the list. 6. The system displays the two teams in the match. 7. The organizer puts in the score for the match. 8. The system calculates the winner based on the score. 9. The system validates the input and saves the result. 10. The system moves the winning team to the next round. 11. The system updates the tournament bracket.
Alternative flow:	1A. No unfinished matches exist. - All matches have already been completed. 2A. invalid input. - Invalid input please input a valid score one team has to be the winner.
Post condition:	The match is marked as completed. The winner is moved to the next round. The brackets are updated.
Actor(s):	Organizer
Author(s):	Tristan

Use case name:	**View team information**
Number:	R12
Priority:	A
User story:	As a user i want to see the team information so that i can see players in each team.
Precondition:	The team exists.
Basic flow:	1. User selects view teams. 2. The system displays a list of the teams. 3. User selects a team. 4. The system displays a list of the team,with the players and the captain and all player handles.
Alternative flow:	1A. Team not found. - Enter a valid name.
Post condition:	User has viewed team information.
Actor(s):	All users
Author(s):	Tristan

Use case name:	**Double elimination tournament**
Number:	R13
Priority:	B
User story:	As the organizer, I want to generate a double elimination tournament so that teams get a second chance even if they lose once.
Precondition:	The tournament is created. All teams are registered. The tournament type is set to Double elimination.
Basic flow:	1. Organizer selects Generate Double Elimination Schedule 2. System creates a Winner Bracket. 3. System creates a Loser Bracket. 4. Teams start in Winner Bracket.

	5. When a team loses a match for the first time, they move to Loser Bracket. 6. When a team loses a second match, they are eliminated from the tournament. 7. The system schedules matches in both brackets. 8. The winner of the Loser Bracket plays against the Winner Bracket finalist. 9. The system displays the full double elimination structure.
Alternative flow:	1A. Not enough teams. - At least 4 teams need to be registered to create a double elimination tournament.
Post condition:	A double elimination tournament structure is created. Both brackets are ready for match scheduling.
Actor(s):	Organizer
Author(s):	Tristan

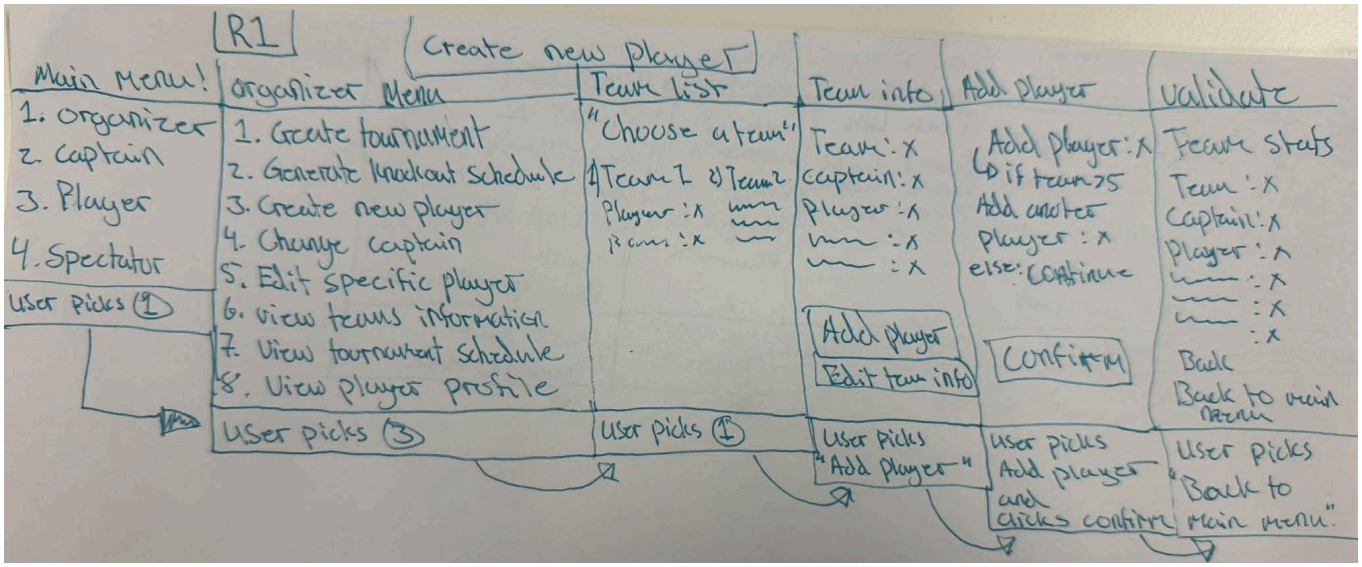
Use case name:	**Point system**
Number:	R14
Priority:	B
User story:	As the team i want to be able to get points if they win the tournament so that i can keep track of our wins on tournaments.
Precondition:	Tournament exists. All matches have been completed. A final winner has been picked.
Basic flow:	1. The final match results have been entered. 2. The system identifies the tournament winner. 3. The system gives the winning team 1 point. 4. The system saves the updated team data and keeps track of the winners. 5. The updated point is visible in the team statistics.
Alternative flow:	1A. tournament has not been finished. - Points can not be given if the tournament is not finished.
Post condition:	The winning team of the tournament receives one point.
Actor(s):	System

Author(s):	Tristan

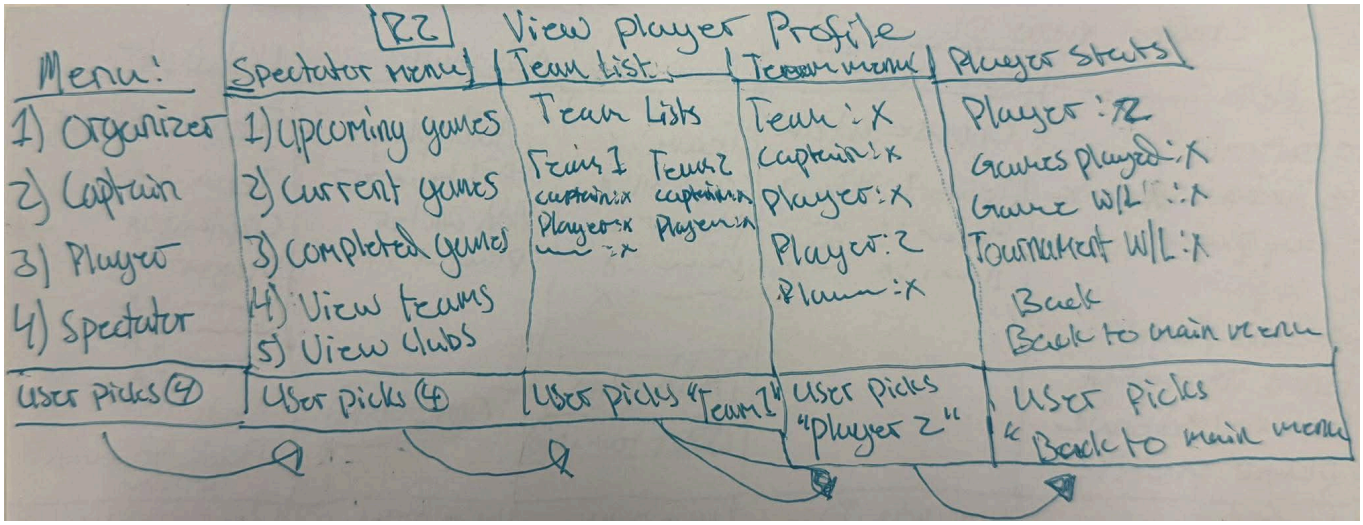
Use case name:	**View team clubs**
Number:	R15
Priority:	B
User story:	As a spectator, captain or a player i want to be able to view team clubs information so that i can see their stats.
Precondition:	The team club exists.
Basic flow:	1. The user clicks view teams information. 2. The system shows two options view teams or team clubs. 3. The user clicks view club. 4. The system shows the list of clubs.
Alternative flow:	1A. No clubs found. - Message saying no clubs found.
Post condition:	User has viewed team clubs information.
Actor(s):	All users
Author(s):	Tristan

Happy paths

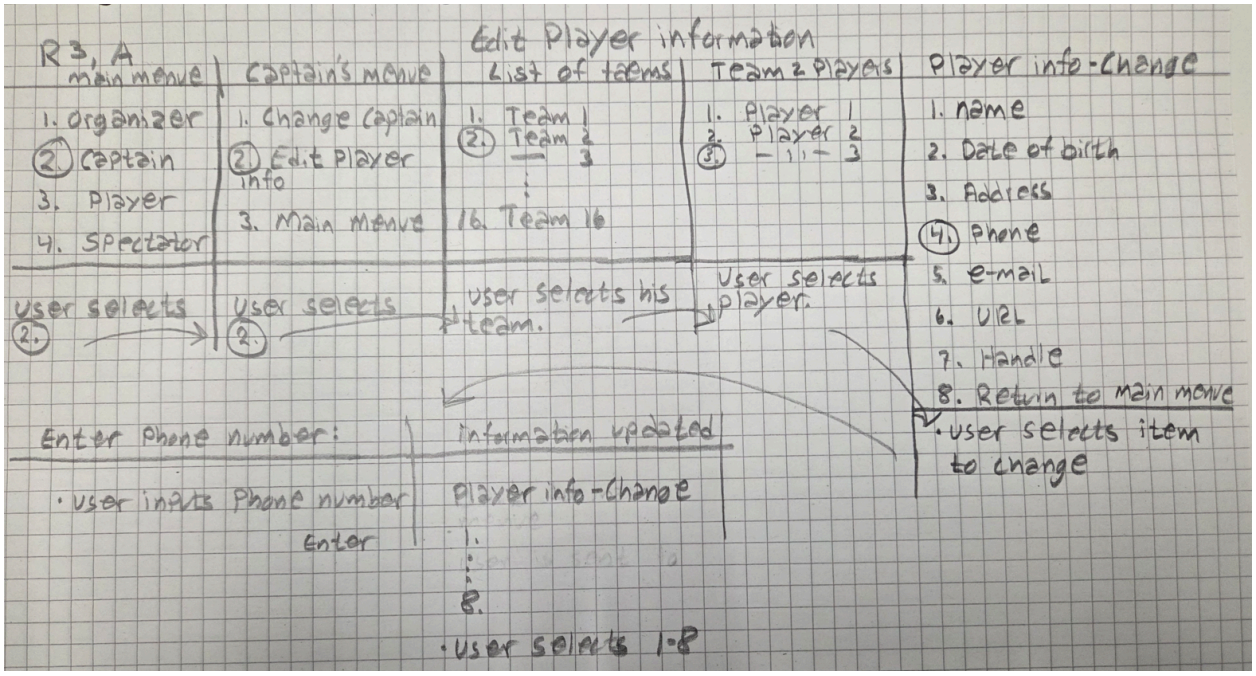
R1 : Create a new player



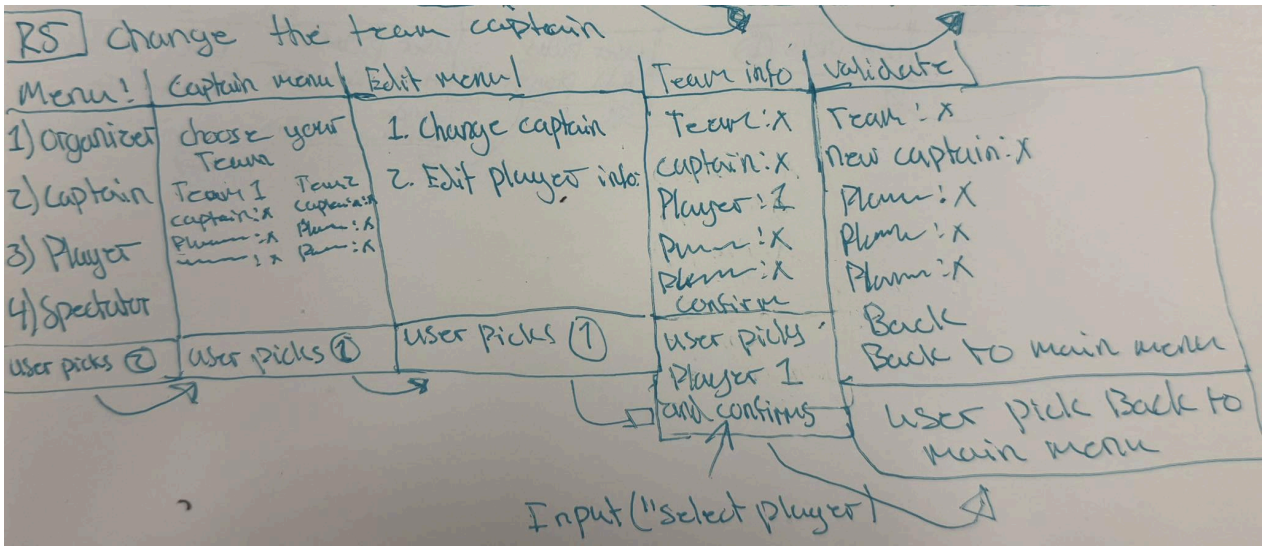
R2 : View player profile



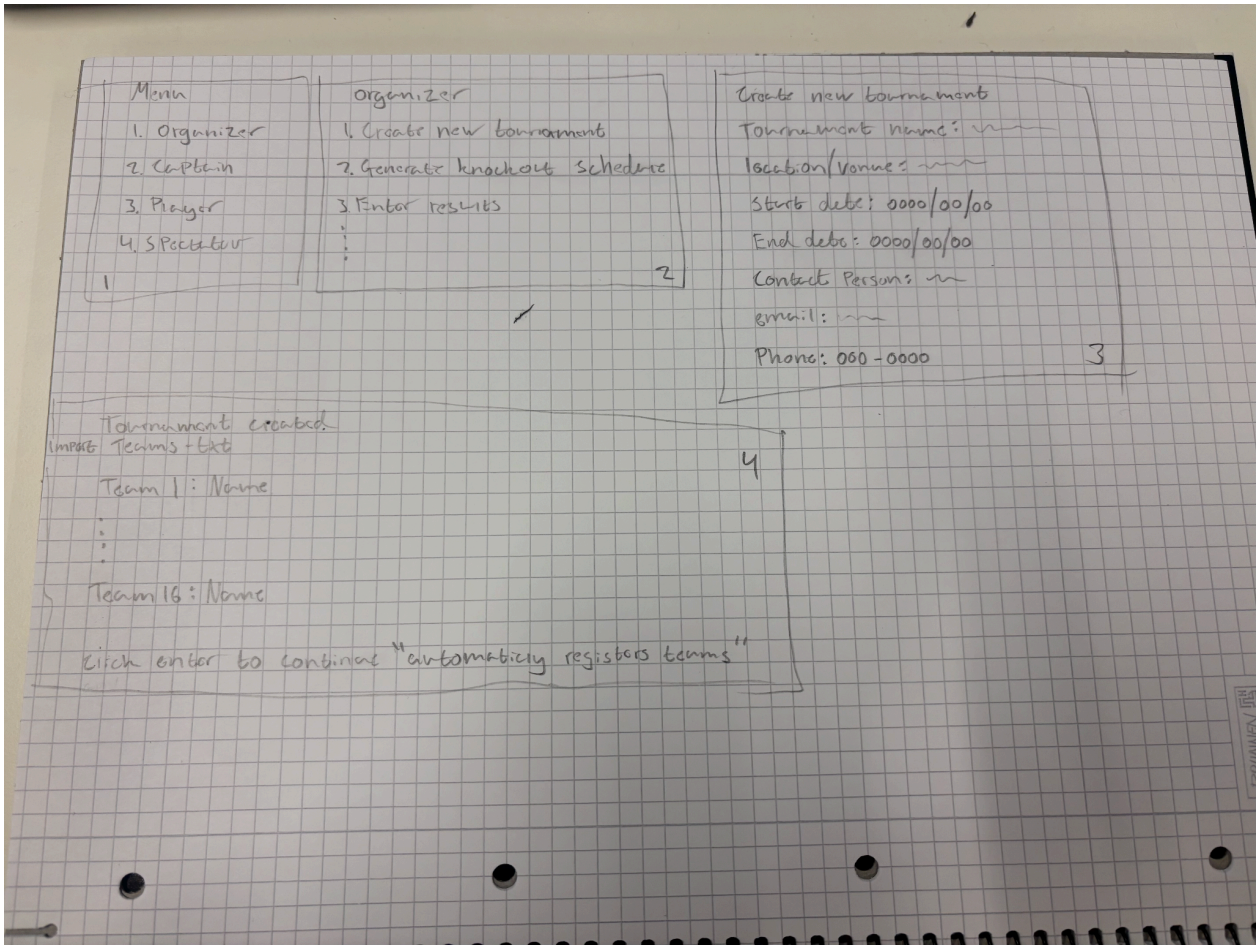
R3 : Edit Player Information.



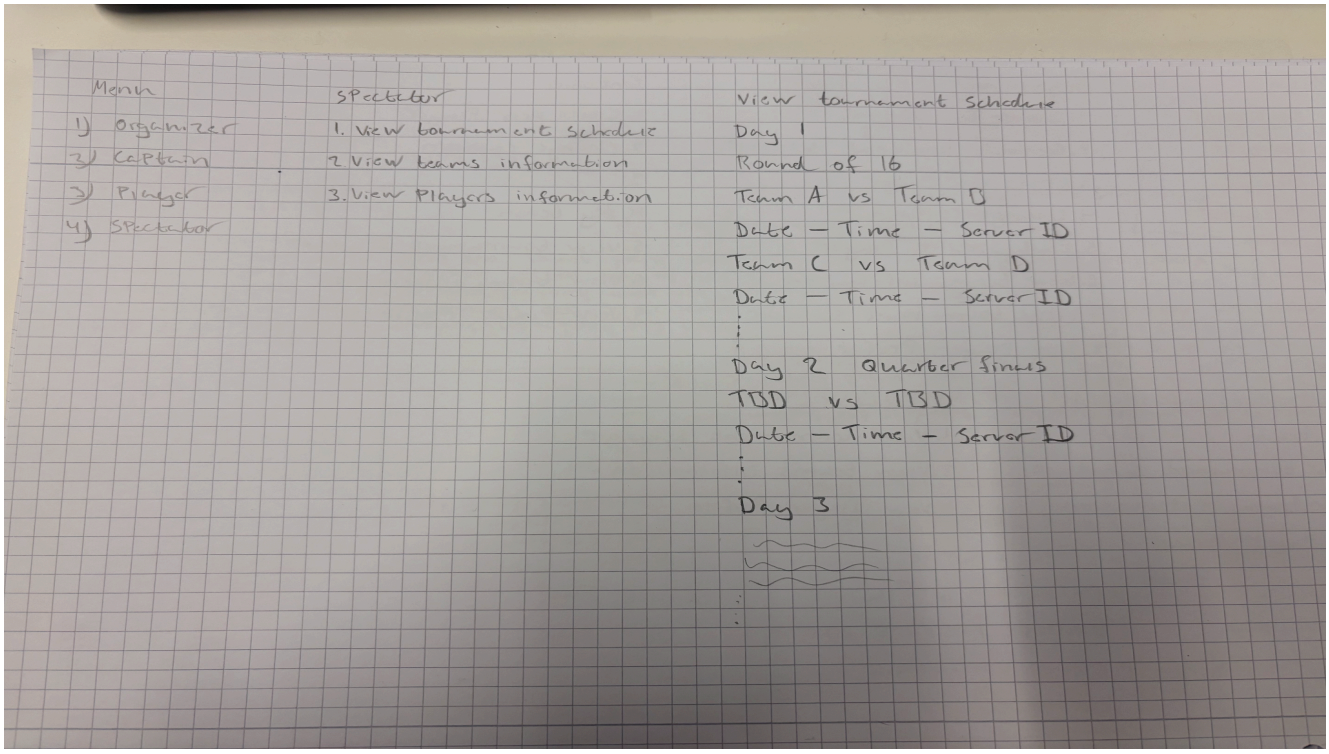
R5 : Change the team captain



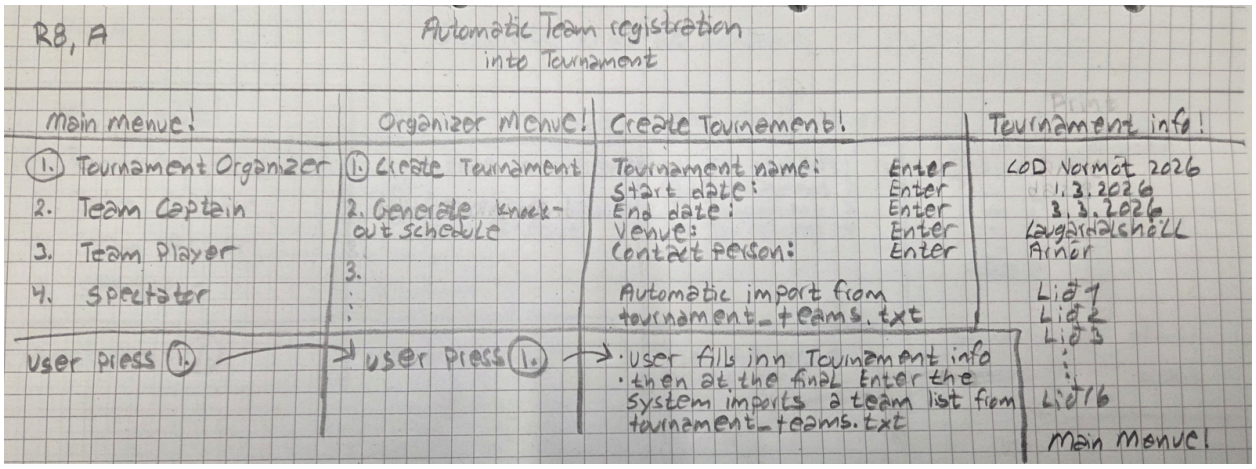
R6


Menu
1. Organizer
2. Captain
3. Player
4. Spectator
1
Organizer
1. Create new tournament
2. Generate knockout schedule
3. Enter results
2
Create new tournament
Tournament name:
location/venue:
Start date: 0000/00/00
End date: 0000/00/00
Contact Person:
email:
Phone: 000-0000
3
Tournament created.
Import Teams + txt
4
Team 1: Name
Team 16: Name
Click enter to continue "automatically registers teams"

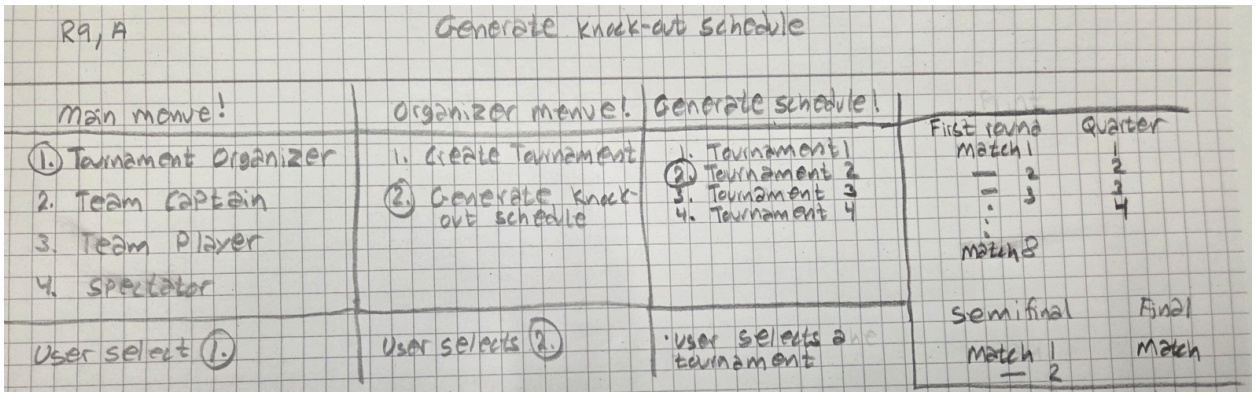
R7 : View tournament schedule.



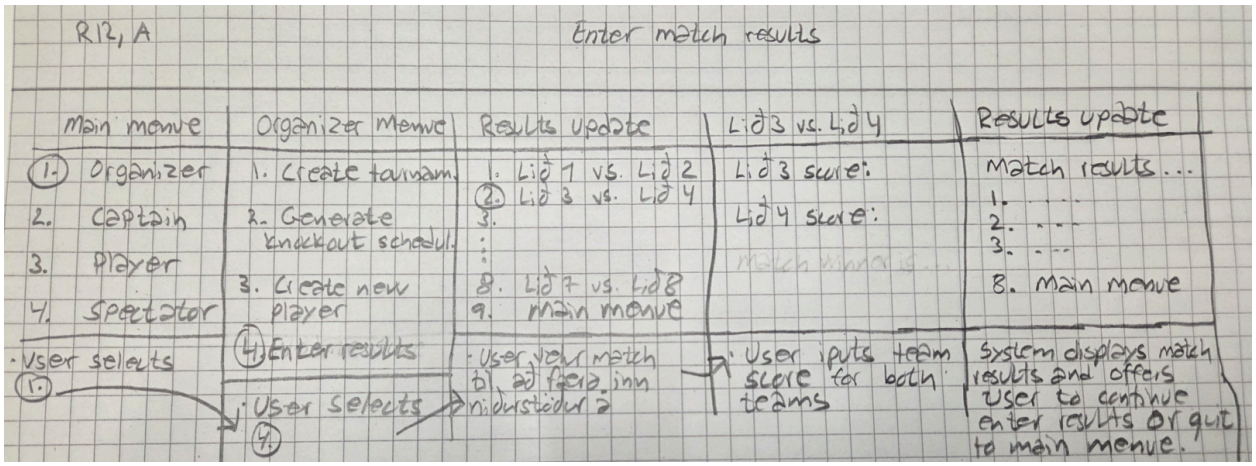
R8 Automatic Team registration into tournament.



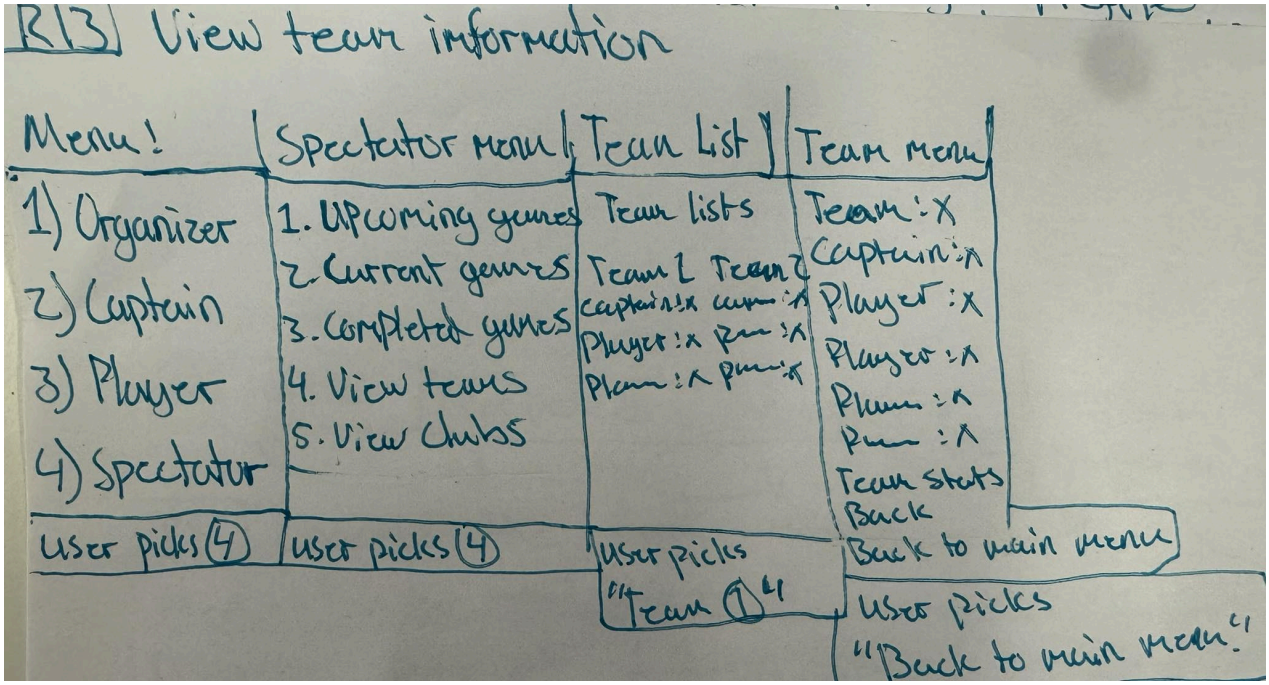
R9 Generate knockout schedule.



R12 Enter match results.



R13 : View team information.



Prototype testing

https://www.figma.com/proto/e2TnVINb6vn1wr7fad0YvS/Verklegt-1-flow?node-id=21-371&t=Xzsene8fpX8Hbwjl-0&scaling=min-zoom&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=21%3A371 - figma prototype

Prototype testing,

user is asked to create a tournament.

User is asked to edit player info as the captain

User is asked to view completed tournaments

==========

RU E-Sport Tournament System

==========

1. Organizer
2. Team Captain
3. Player
4. Spectator
0. Exit

Not for user display.

Select user type: 1

--------- Organizer Menu ---------

1. Create Tournament
2. Generate Tournament Schedule
3. Enter Match Results
4. Change Team Captain
5. View Tournament Schedule
6. View Teams
0. Back

Not for user display:

Select option: 1

Enter tournament name:

Enter start date (DD/MM/YYYY):

Enter end date (DD/MM/YYYY):

Enter location:

Enter contact email:

Enter contact phone:

Not for user display:

Enter tournament name: RU Cup 2025

Enter start date (DD/MM/YYYY): 12/05/2025

Enter end date (DD/MM/YYYY): 14/05/2025

Enter location: Reykjavik University

Enter contact email: ru@ru.is

Enter contact phone: 555-1234

✔ Tournament created successfully!

Crimson Dragons

Silver Wolves

Neon Ninjas

Iron Titans

Shadow Ravens

Frost Giants

Obsidian Owls

Solar Serpents

Lunar Lions

Voltage Vipers

Arcane Arrows

Blaze Brigade

Phantom Phantoms

Cyber Cobras

Pixel Pirates

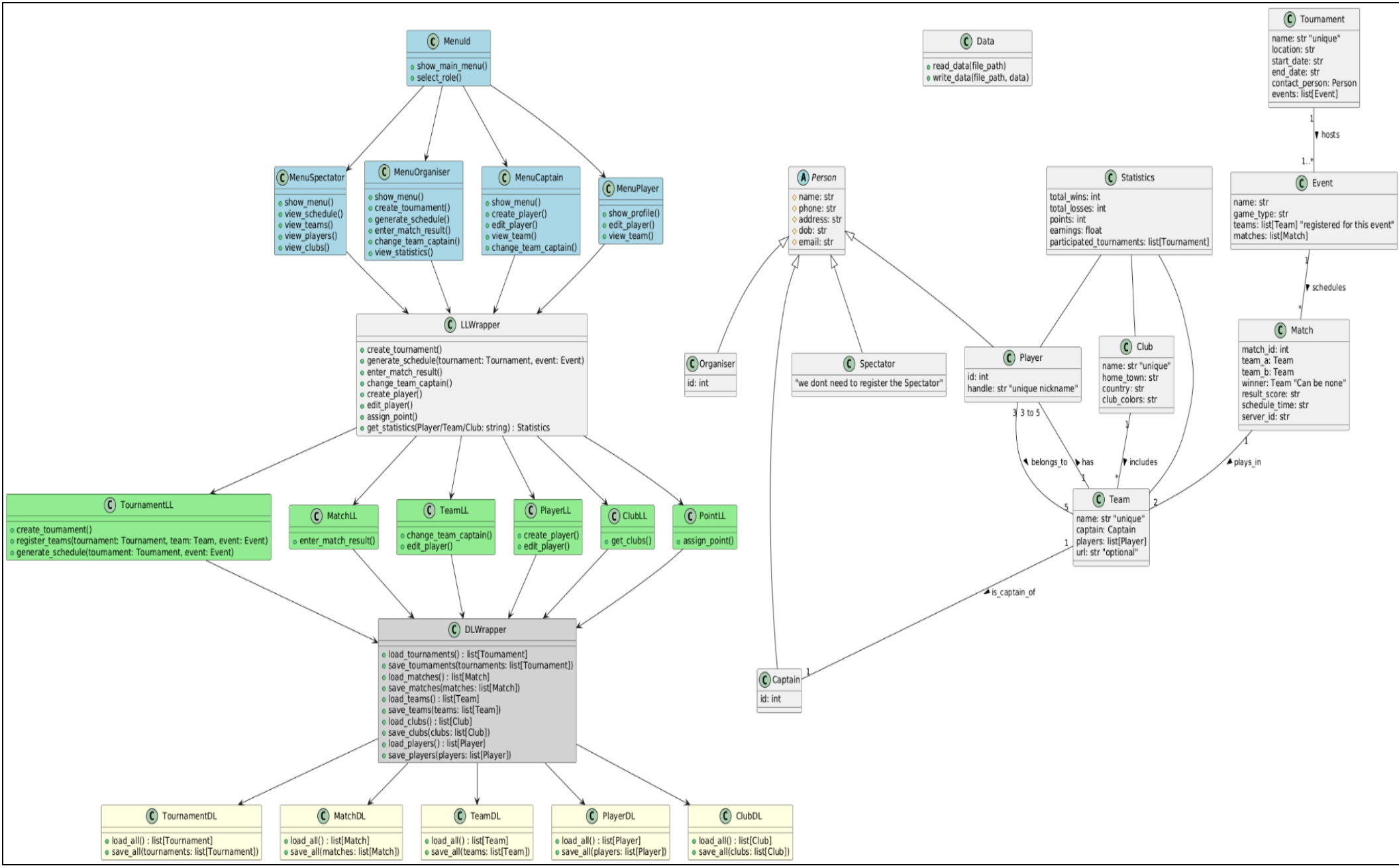
Nova Knights

User group analysis

Group Name	Who (Background)	Main Goals	Abilities / Limitations	Equipment	Usage Frequency & Duration	System Skill	Importance Level	Group Size
Administrator / Organizer	Age: 18–99Gender: AllEducation: Master/PhD in administration Computer Skills: Medium–High	Access backend to create and edit events and plans	Can fully manage administration and organization with some restrictions	Smartphone / Laptop with web access	Frequently during tournament setup10–60 min per session	Moderate–High	Very Important	4
Team Captain	Age: 18–99Gender: AllEducation: AnyComputer Skills: Low–Medium	Manage team data, edit players, change team name	Can edit team-related data only, no access to organizer or schedule edits	Smartphone / Laptop with web access	Occasionally10–60 min per session	Low	Very Important	16
Player	Age: 18–99Gender: AllEducation: None neededComputer Skills: Medium–High	Access and edit personal info, view schedule	Can only edit their own data, no access to others' data	Smartphone / Laptop with web access	Not often1–10 min per session	Low	Not Very Important	4

Spectator / Public User	Age: 1–99Gender: AllEducation: AnyComputer Skills: Low–Medium	View tournament schedule, team and player status	View-only access, no editing	Smartphone / Laptop with web access	Sometimes to very often10–60 min per session	Low	Not Very Important	Unlimited

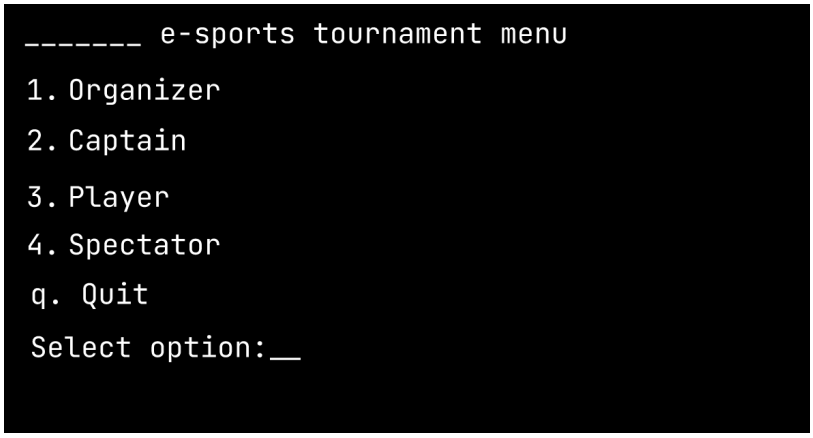
Class diagram



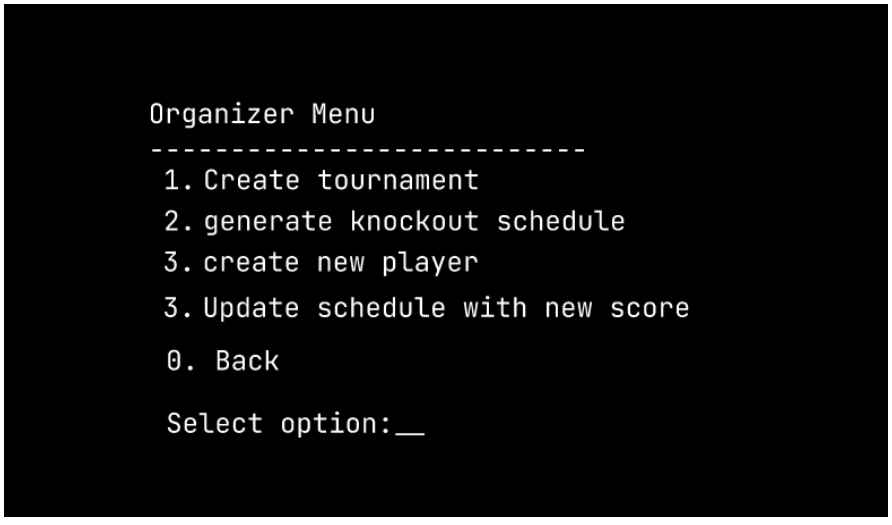
UI design

This section shows the layout and structure of the system's text-based user interface. The wireframes illustrate how users navigate the menus, enter information, and view tournament data. They serve as a visual guide for how the final console-based UI should look and help ensure consistency during implementation.

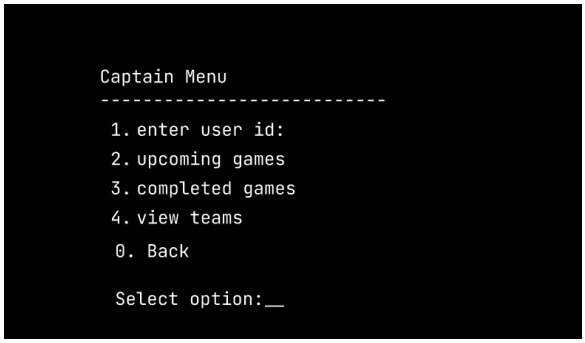
Main menu

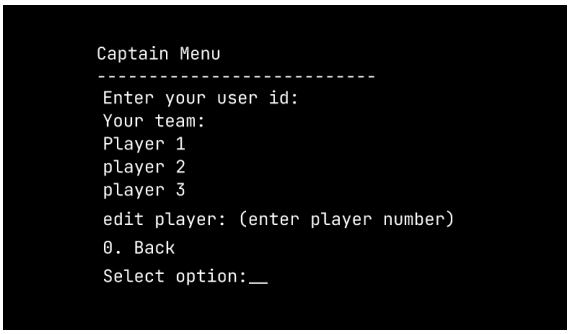


Organizer menu

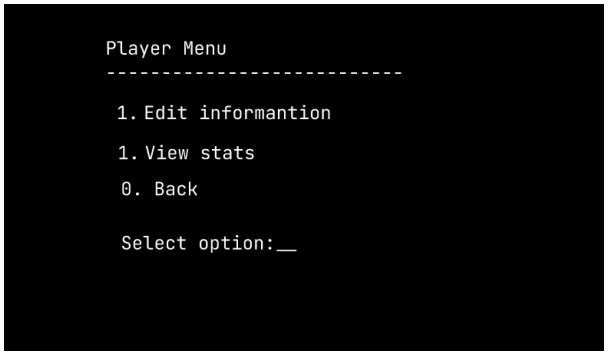


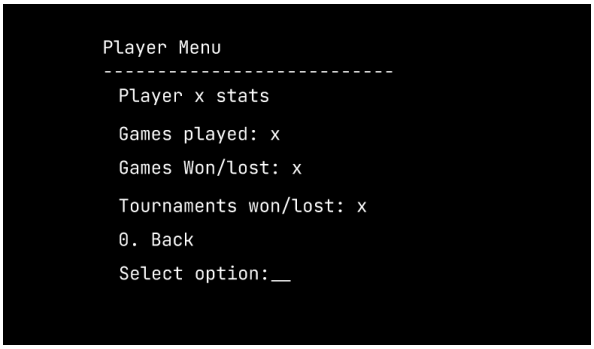
Captain menu



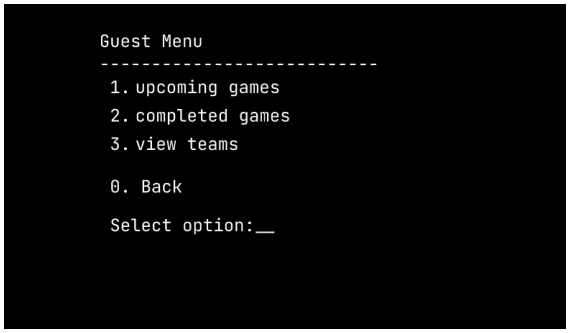


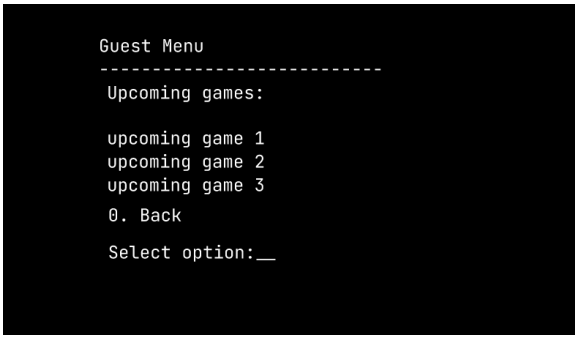
Player menu

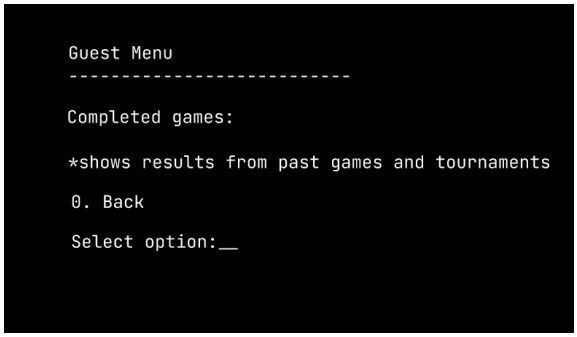


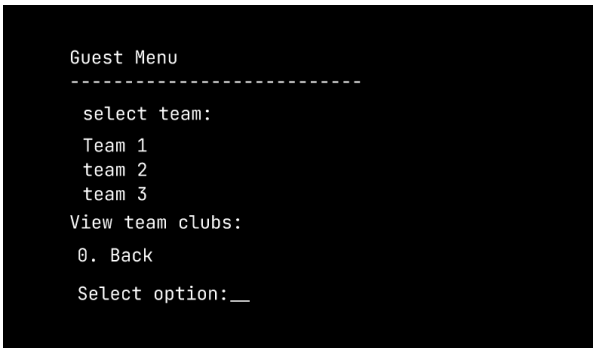


Guest menu



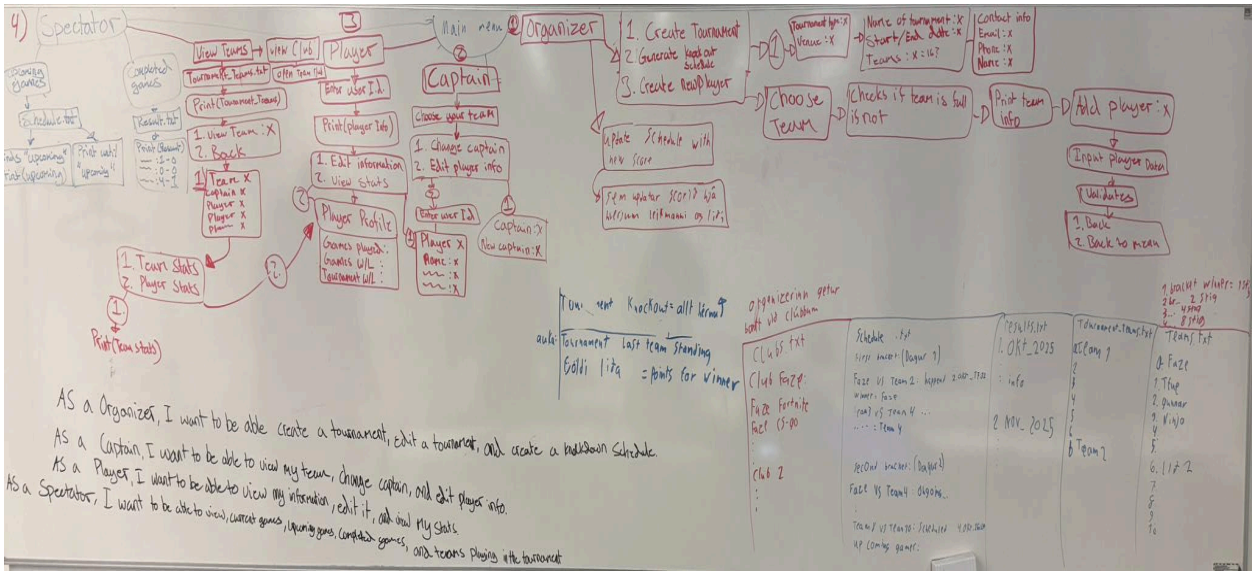


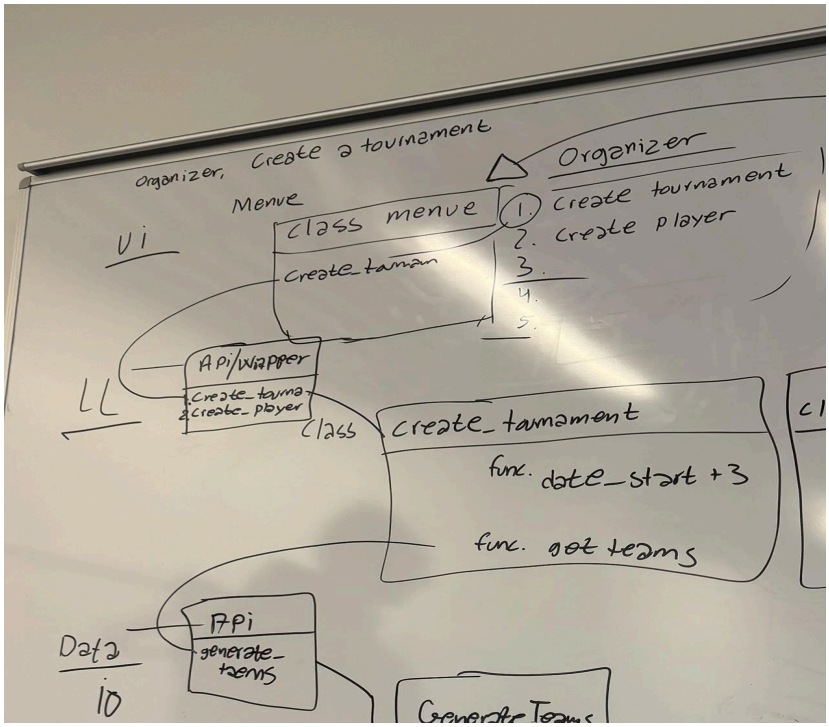




Testing methods

- **Lo-fi Testing**
  Lo-Fi testing is when we test very simple, rough prototypes (like paper sketches or basic wireframes) to check ideas, user flow, and concept.





- **Think-Aloud Testing**
  User talks while using the system, reveals thoughts and problems

- **Pilot Testing**
  Pilot testing is a small-scale trial run of your test or system with a few users, before doing full user testing, to check if everything works correctly

Test methods

1. Create a new tournament
2. Edit information on a player in your team
3. See statistics on completed games

- **Functional Testing**
  Requirements for what the user should be able to do in the system and what not.
- **Validation / Error Testing**
  Checks if the system is doing what its supposed to do, and purposely testing mistakes and trying to break system
- **Usability Testing**
  How easy the interface is to navigate
- **User Testing**
  User Testing is when real users from your target group interact with your product or prototype in a realistic scenario to see how well it works for them. It focuses on how they use it, what works, what confuses them, and whether the product meets their needs.
- **Non-Functional Testing**

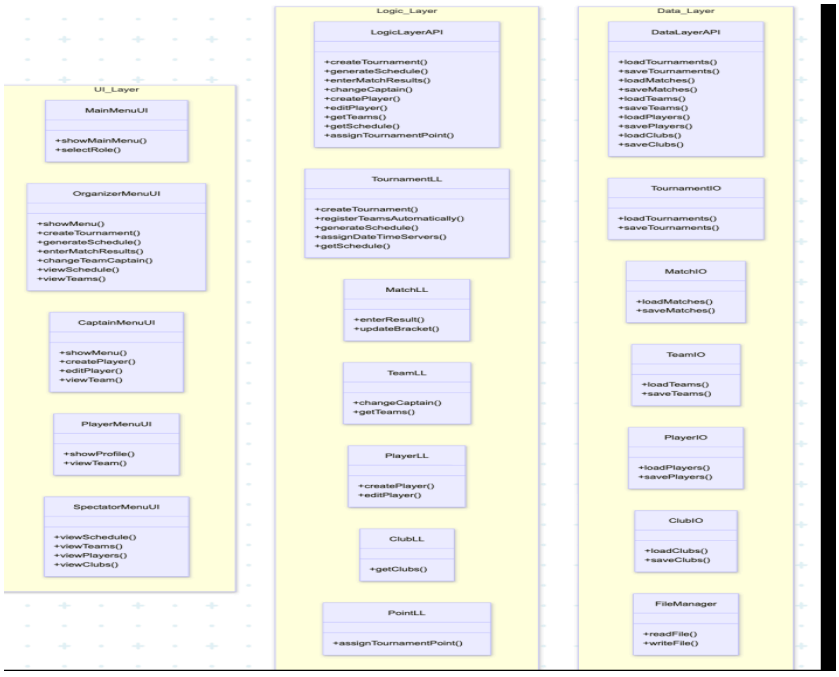
Non-functional testing checks **how well** a system works rather than **what it does**.

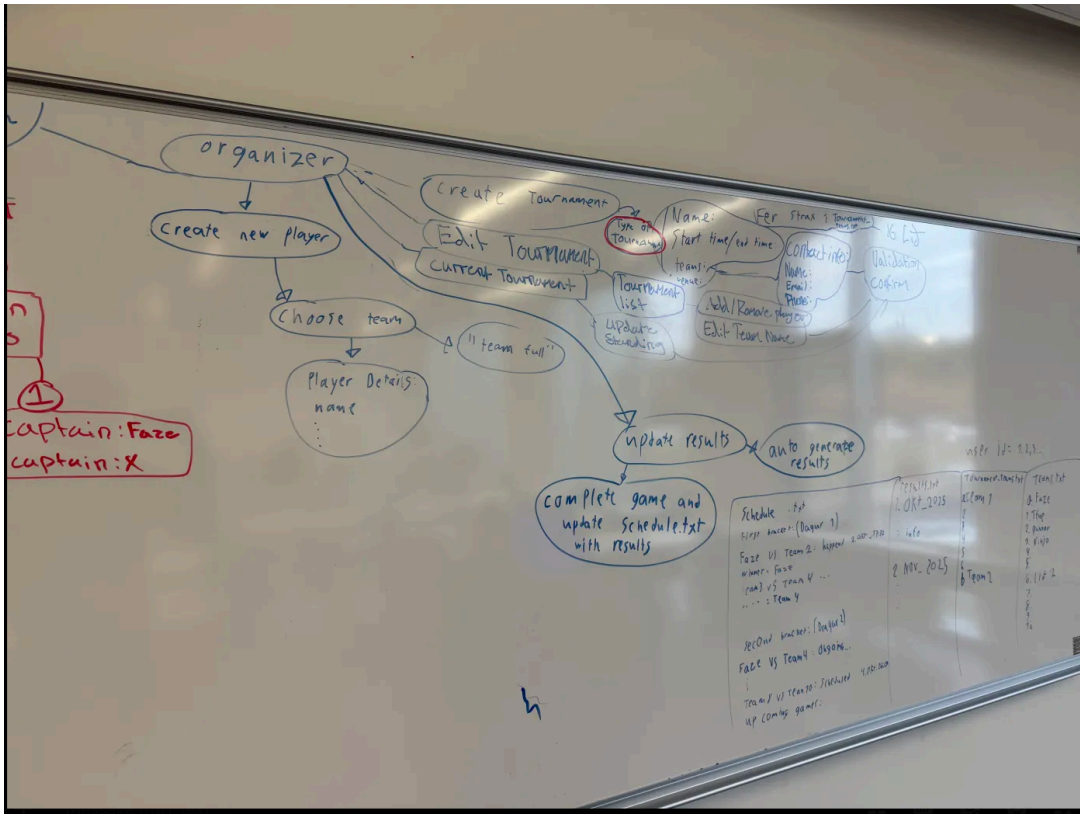
Req.	Testing Types	Example User Interaction (Scenario)
R1 – Create new player	Think-aloud, Functional, Validation/Error, Usability	Organizer → Create new player → Select Team 1 → Add Player → Enter data → Save
R2 – View player profile	Think-aloud, Usability, Functional, Non-functional (data retrieval), Validation/Error	Spectator → View teams → Select Team 1 → Select Player 2 → View profile
R3 – Edit player information	Think-aloud, Usability, Functional, Non-functional (data retrieval), Validation/Error	Captain → Edit player → Team 2 → Select Player 3 → Change phone number → Save
R4 – Change team captain	Functional, Non-functional (data retrieval), Validation/Error, Think-aloud	Captain → Team 2 → Change captain → Select Player 1 → Confirm

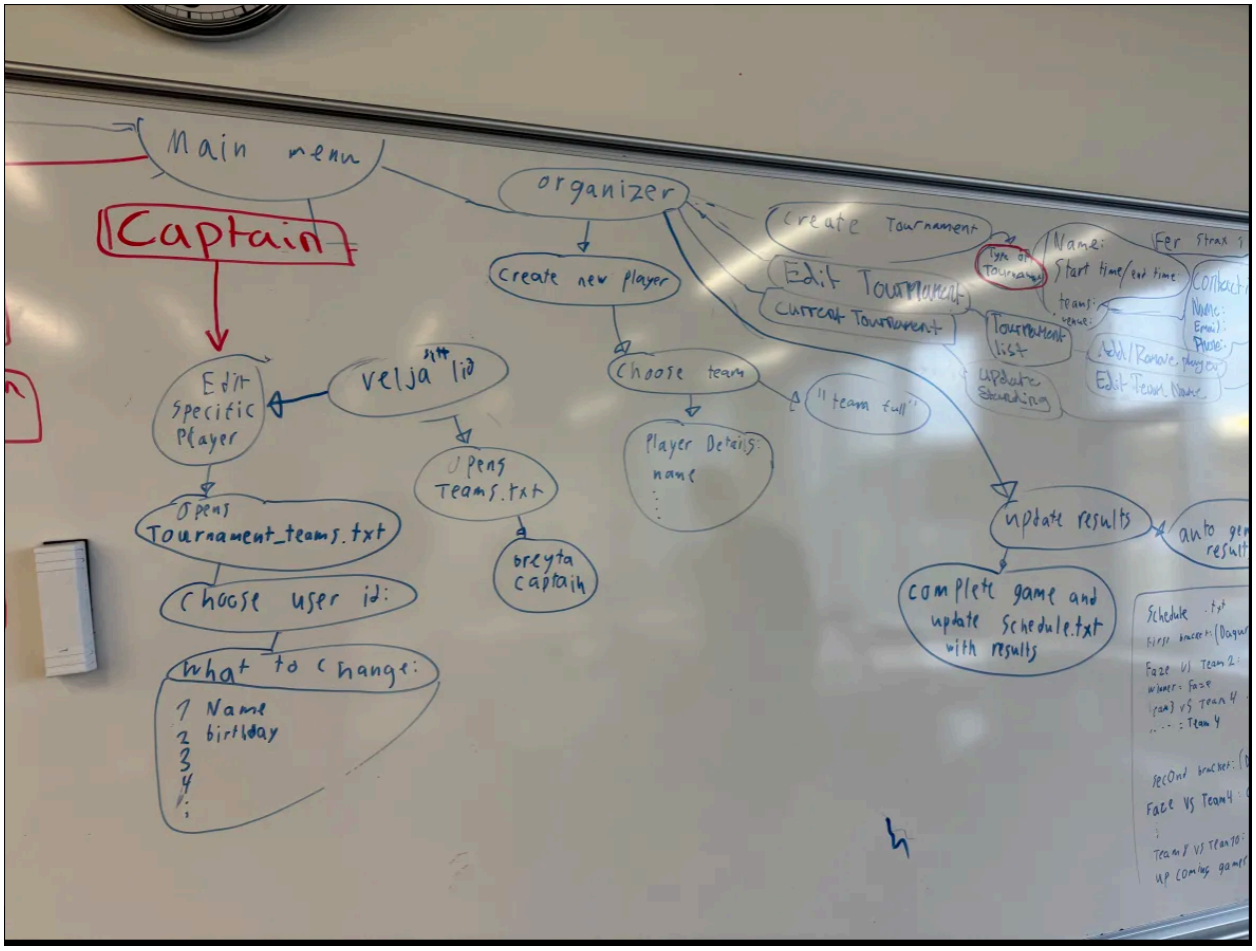
R5 – Create new tournament	Think-aloud, Usability, Functional, Non-functional (generation), Validation/Error	Organizer → Create tournament → Enter data → Save → Tournament created
R6 – View tournament schedule	Think-aloud, Usability, Functional, Non-functional (schedule generation), Validation/Error	Spectator → View tournament schedule → Display schedule
R7 – Automatic team registration into tournament	Think-aloud, Usability, Functional, Non-functional (data retrieval/assignment), Validation/Error	Organizer → Create tournament → Enter data → Print/generated tournament with auto teams
R8 – Generate tournament schedule	Think-aloud, Usability, Functional, Non-functional (data generation), Error testing	Organizer → Generate schedule → Select tournament → Auto-generate schedule
R9 – Prevent teams playing two matches at the same time	Non-functional, Validation (conflict checking), Error testing	System checks when generating schedule and blocks overlapping matches
R10 – Assign date, time and server to matches	Non-functional, Validation (correct assignment), Error testing	Organizer → Generate schedule → Auto-assign date/time/server → Display
R11 – Enter match results	Think-aloud, Usability, Functional, Non-functional (data saving), Validation/Error	Organizer → Enter results → Submit match details → Confirm and save
R12 – View team information	Think-aloud, Usability, Functional, Non-functional (data retrieval), Validation, Error testing	Spectator → View teams → Select Team 4 → View information
R13 – Double elimination tournament	Usability, Functional, Non-functional (generate winner/loser bracket), Validation, Error testing	System manages elimination brackets automatically for tournaments
R14 – Point system	Non-functional (point generation), Validation (correct scoring), Error testing	System auto-assigns win/loss points based on results
R15 – View team clubs	Think-aloud, Usability, Functional, Non-functional (club data retrieval), Validation, Error testing	Spectator → View clubs/teams → Select club → Display club information

Other work documents

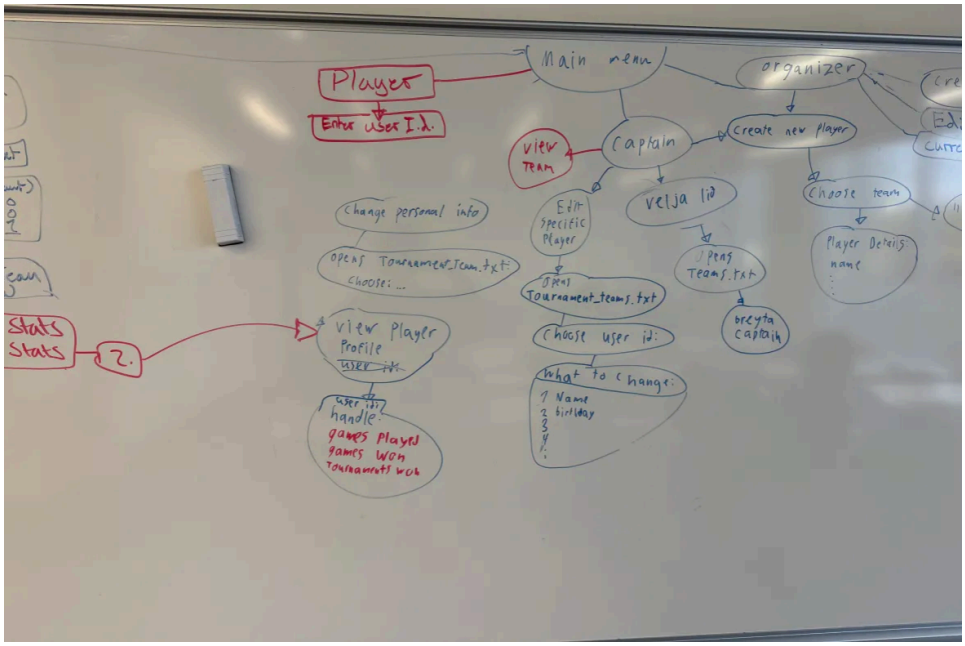
Unnið í blöndu af flæði, valmyndum, klösum og samspili við gagnasöfn.

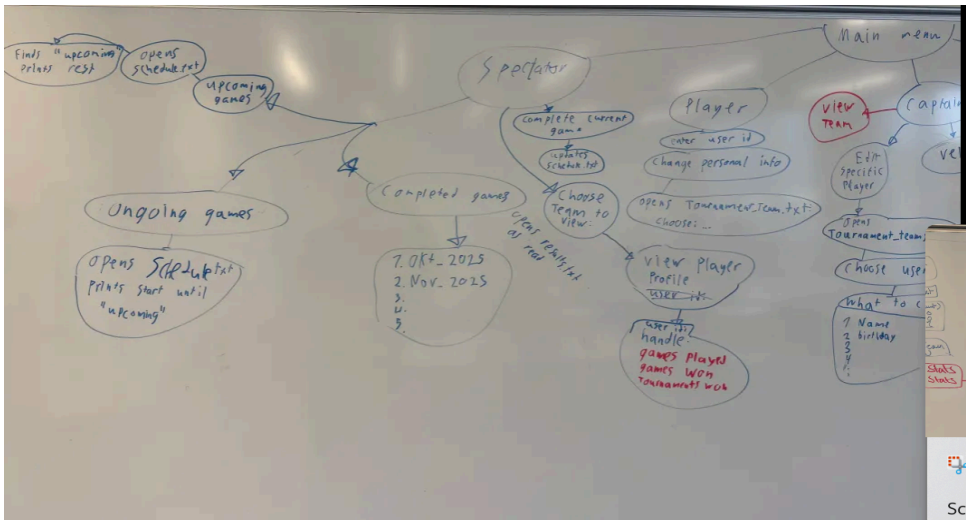


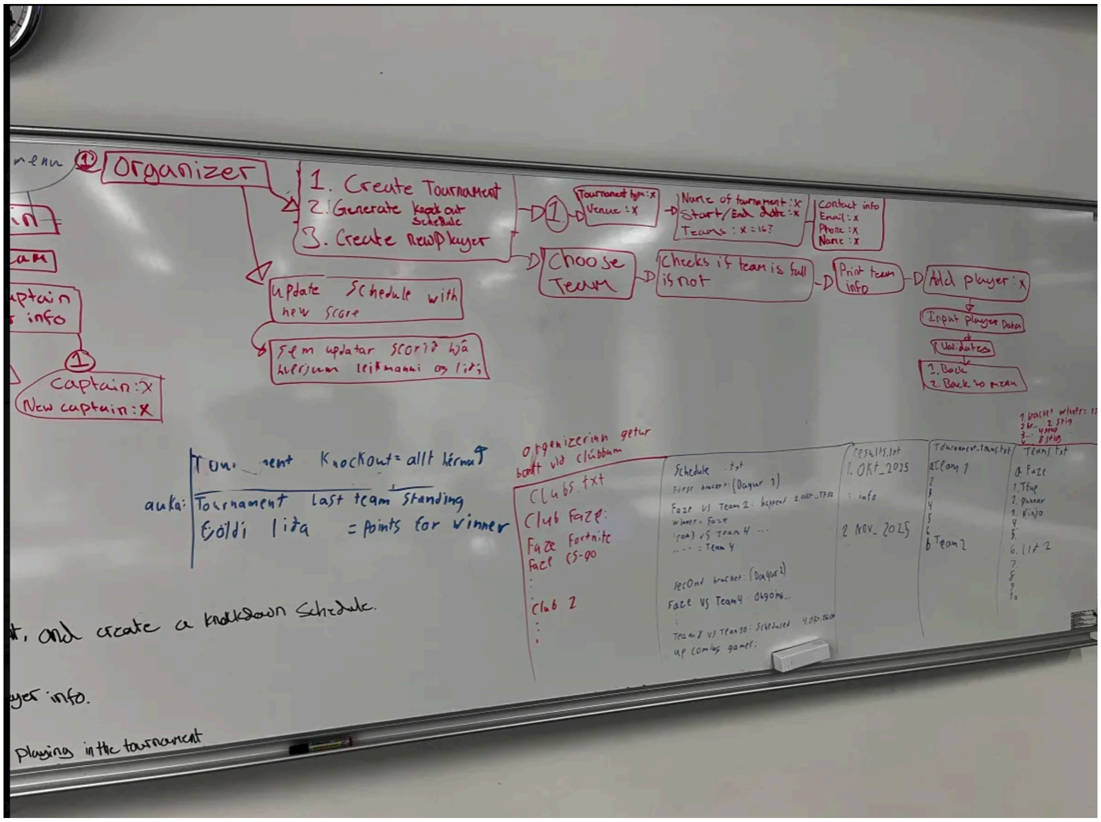

organizer
Create Tournament
Edit Tournament
Current Tournament
Create new player
Choose team
"team full"
Player Details
name
captain: Faze
captain: X
update results
auto generate results
complete game and update Schedule.txt with results

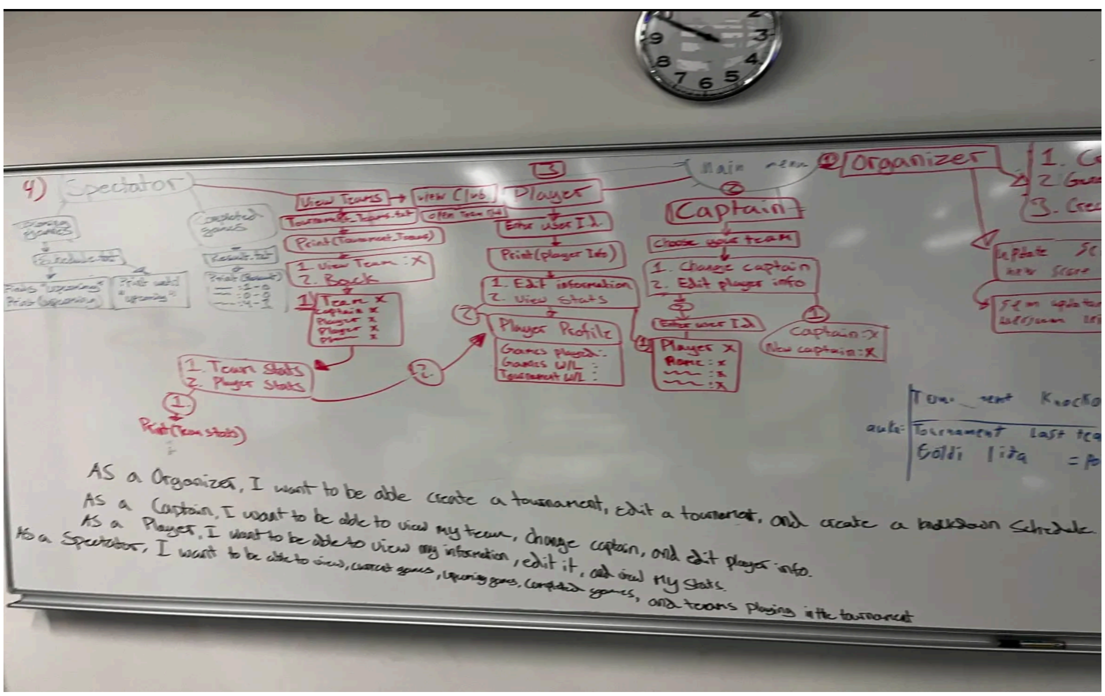

Main menu
Captain
organizer
Create Tournament
Edit Tournament
Current Tournament
Tournament list
Update Standing
Create new player
Choose team
"team full"
Player Details:
name
Edit Specific Player
velja lið
Opens Teams.txt
breyta captain
Opens Tournament_teams.txt
Choose user id:
What to change:
1 Name
2 birthday
3
4
update results
complete game and update Schedule.txt with results
Name:
Start time/end time:
teams:
Add/Remove player
Edit Team Name


Main menu
organizer
Captain
View Team
Create new player
Edit Specific Player
velja lið
Opens Teams.txt
Opens Tournament_teams.txt
breyta captain
Choose user id:
What to change:
1 Name
2 birthday

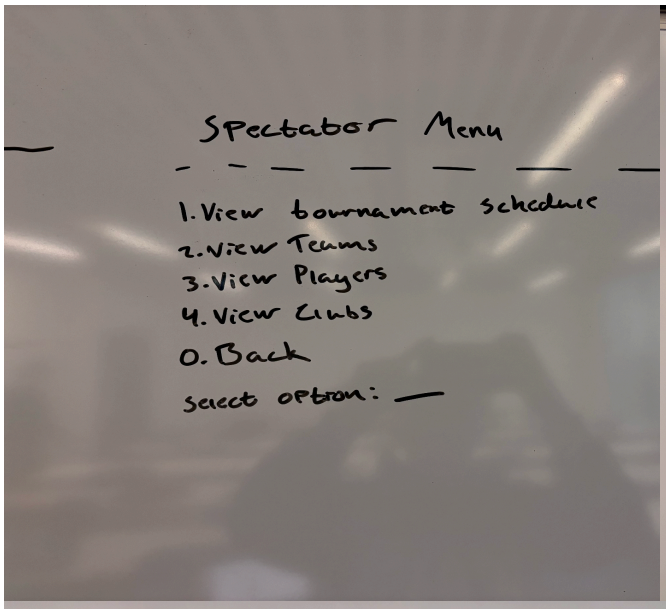

Player
Enter user I.d.
Main menu
organizer
Captain
View TEAM
Create new player
Edit specific Player
velja lid
Choose team
Change personal info
opens Tournament_Team.txt
Choose:
Open Tournament_teams.txt
Open Teams.txt
Player Details
name
breyta captain
Choose user id:
What to change:
1 Name
2 birthday
3
4
5.
Stats
Stats
2.
view Player Profile
USER id:
user id:
handle:
games Played
games Won
Tournaments won

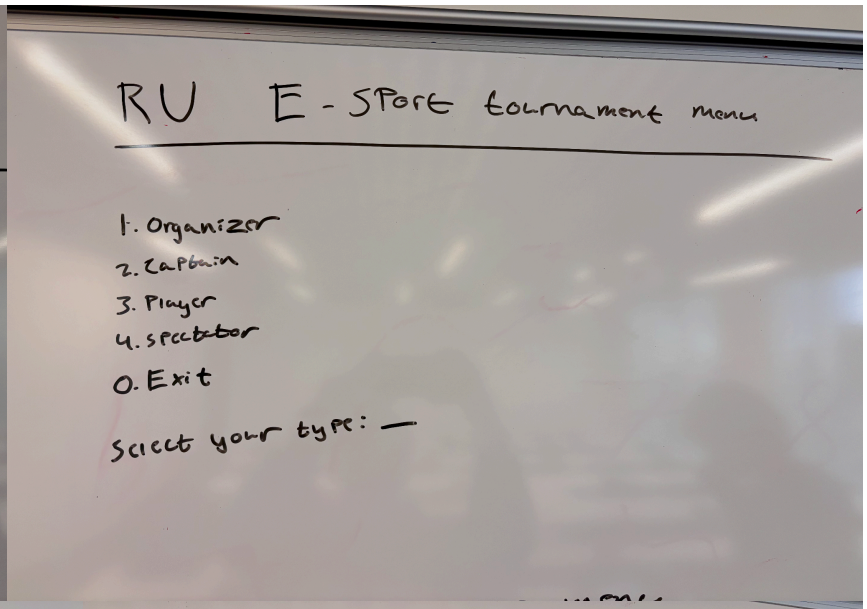

Finds "upcoming" prints rest
opens schedule.txt
upcoming games
Spectator
Main menu
Player
Captain
View TEAM
complete current game
update schedule.txt
enter user id
Change personal info
Edit specific Player
Choose team to view:
opens Tournament_Team.txt
Choose:
Open Tournament_team
Choose user
What to
1 Name
2 birthday
3
4
5.
Ungoing games
Completed games
opens Schedule.txt
prints start until "upcoming"
open results.txt or read
1. Okt_2025
2. Nov_2025
3.
4.
5.
view Player Profile
USER id:
user id:
handle:
games Played
games Won
Tournaments won
Stats
Stats

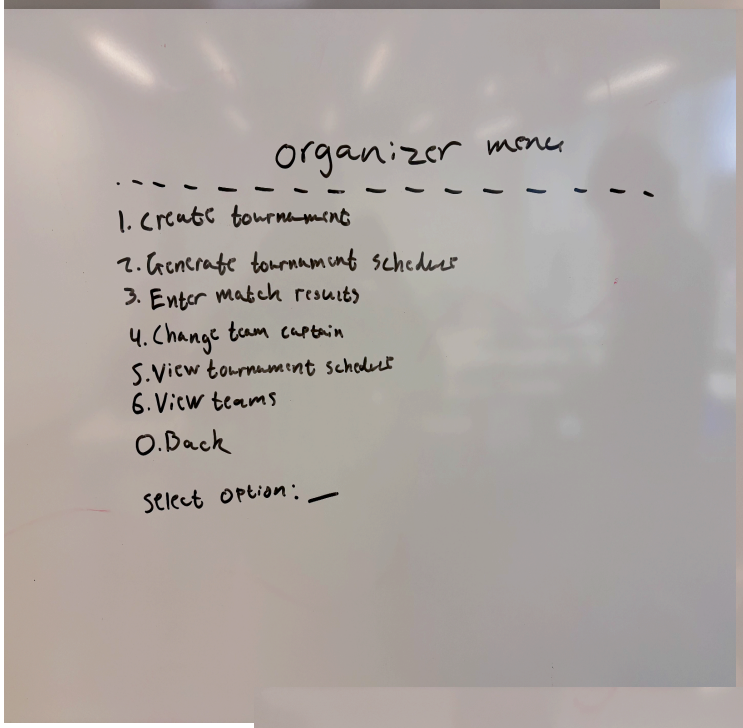

Organizer
1. Create Tournament
2. Generate knock out schedule
3. Create new player
Choose Team
Checks if team is full / is not
Print team info
Add player: X
Input player data
Validates
1. Back
2. Back to menu
Update schedule with new score
Captain: X
New captain: X
Clubs.txt
Club Faze:
Faze Fortnite
Club 2
Schedule.txt
results.txt
Teams.txt
, and create a knockdown schedule.
player info.
playing in the tournament

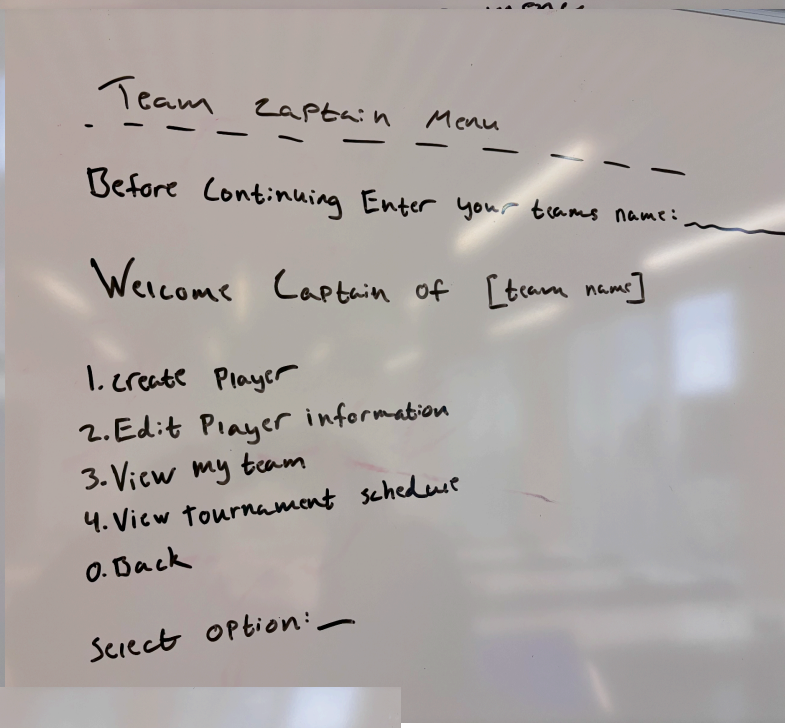

4) Spectator
View Teams
View Club
3 Player
Main menu
1 Organizer
2 Captain
Choose your team
1. Change captain
2. Edit player info
Enter user I.D.
Print (player info)
1. Edit information
2. View stats
Player Profile
1. View Team: X
2. Back
1. Team stats
2. Player stats
Print (team stats)
Captain: X
New captain: X
AS a Organizer, I want to be able create a tournament, edit a tournament, and create a knockdown schedule.
AS a Captain, I want to be able to view my team, change captain, and edit player info.
AS a Player, I want to be able to view my information, edit it, and view my stats.
As a Spectator, I want to be able to view, current games, upcoming games, completed games, and teams playing in the tournament

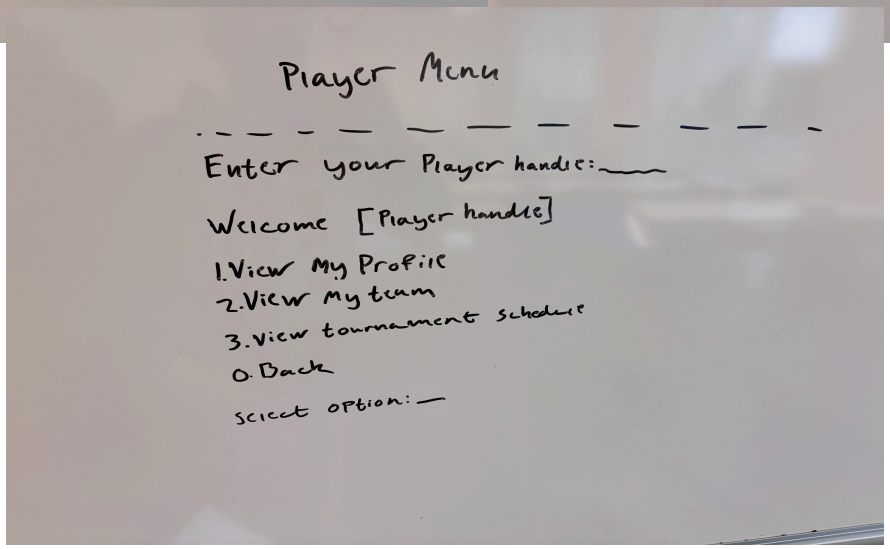
Menu :


Spectator Menu
1. View tournament schedule
2. View Teams
3. View Players
4. View Clubs
0. Back
select option:


RU E-Sport tournament menu
1. Organizer
2. Captain
3. Player
4. spectator
0. Exit
Select your type:


organizer menu
1. Create tournament
2. Generate tournament schedule
3. Enter match results
4. Change team captain
5. View tournament schedule
6. View teams
0. Back
select option:


Team Captain Menu
Before Continuing Enter your teams name:
Welcome Captain of [team name]
1. create Player
2. Edit Player information
3. View my team
4. View tournament schedule
0. Back
Select option:


Player Menu
Enter your Player handle:
Welcome [Player handle]
1. View My Profile
2. View My team
3. View tournament schedule
0. Back
select option:

Conclusions

This design report provides a complete technical and structural overview of the e-Match Booking Records System and serves as the foundation for the upcoming implementation phase. With the requirements, use cases, UI flow, class structure and testing methods now clearly defined, the development team has a unified blueprint to follow.

Next steps:

The next phase in the project is to translate this design into a functional, fully integrated 3-layer system. The team will begin by:

- Implementing the data layer (file handling, team/player storage, tournament persistence).
- Building the logic layer, including bracket generation, scheduling rules, validation and match-result processing.
- Connecting the UI layer to the logic through the text-based interface described in the wireframes.
- Conducting iterative testing, starting with functional tests for each requirement and then full end-to-end scenario testing.
- Updating this design report whenever implementation details evolve so the document stays aligned with the codebase.

Design decisions and their reasoning:

Several design choices were made intentionally to keep the project manageable, consistent and testable within the 3-week development period:

The system is text-based to fully comply with 3-tier requirements and allow predictable, deterministic testing.

A fixed 3-day tournament structure was chosen to simplify scheduling logic and avoid overly complex calendar handling.

Text-file storage (rather than databases) was selected to meet the course specification and reduce setup overhead.

Single elimination as the main tournament type, with double elimination and last-man-standing as B-requirements, ensures core functionality is stable before extending features.

Strict role separation (Organizer, Captain, Player, Spectator) makes the system easier to validate and reduces accidental data modification.

What is not included in the design but may be needed:

Although the core design is complete, several elements are intentionally left out of this document and must be addressed during coding:

Detailed error-handling flows for malformed files, corrupted data or missing resources.

Edge-case handling (e.g., teams with missing players, incomplete club data, unexpected match input formatting).

Persistence format specifications (exact file structure for teams, players, matches and results).

Future scalability considerations, such as supporting more than 16 teams, multiple tournaments, or custom tournament formats.

This design provides a strong, unified starting point. By following it closely—and keeping it updated as development progresses—the team can ensure the final implementation is consistent, maintainable and aligned with the project goals.