Table of Contents

List of Table

List of Figures

Synopsis

1. **Title: -**

   WORDPLAY Game

2. **Statement about problem: -**

   A strong vocabulary, both written and spoken, requires more than a dictionary. In fact, it requires an educational commitment to overcoming four obstacles: the size of the task (the number of words students need to learn is exceedingly large), the differences between spoken and written English, the limitations of information sources including dictionaries, and the complexity of word knowledge (simple memorization is not enough). Learn more about these challenges to acquiring the 2,500 words a student needs to add each year to their reading vocabulary.

3. **Why this topic: -**
   - Promotes word recall and vocabulary building.
   - There is a healthy culture around the game with heavy emphasis on not spoiling the day's word for anyone else.
   - An active social media community exists because of the share function, once a word guessed correctly.
   - It forces users to think critically about all possibilities. Guessing a five-letter word is harder than it seems!

4. **Objective and Scope of the System: -**
   - Objective: -
     - To provide a game which is not addictive and can be played once a day.
     - The game should also be non-time consuming.
     - To provide a well-designed and eye please puzzle game.

5. **Methodology: -** Incremental Model

   Incremental model is the suitable methodology for this system.
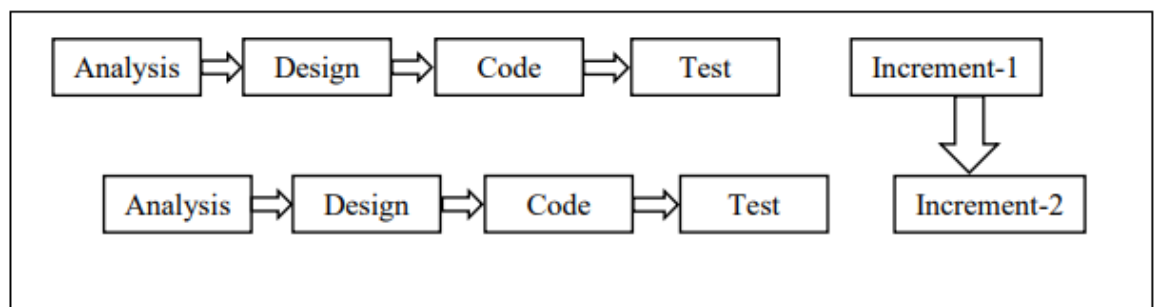
   Fig 1.1 Incremental Model Diagram

6. Proposal Architecture: -
The motive of the system to let user play the game.
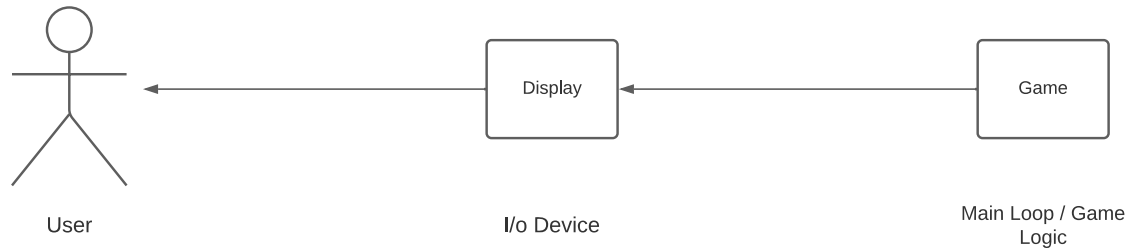
Fig 1.2 Proposed Architecture

7. Requirements: -

- **Software Requirements:** -
  - **Frontend:** - HTML, CSS, ReactJS
  - **Backend:** - NodeJS, ExpressJS
  - **Database:** - MongoDB
  - **Operating System:** - Windows, Linux, MacOS
- **Hardware Requirements:** -
  - **Processor:** - Intel Pentium or higher, AMD Ryzen
  - **RAM:** - 1GB or higher
  - **Graphics card:** - Inbuilt Graphics or dedicated graphics card

8. Platform: -
Visual Studio Code

9. Contribution towards Society: -

The main contribution of this game towards society is in order to keep your memory and your thinking sharp, the key is to really challenge and learning. Those are the one of the only ways that you're really actually exercising your brain, you're growing new neural pathways, so the key is you can't just busy.

CHAPTER 1:

Introduction

1.1 Background: -

   Wordplay is a game motivated by wordle game which is an online word-guessing game made up of five by six grid, with a new mystery word for players to guess each day. The five squares across represent a single letter in the five-letter word for the day and the six rows each represent one guess at what that day's secret word is.

1.2 Objectives: -

   To play six letter puzzle game.
   To compete with other players on the other devices.
   Allow to share players result on social media.

1.3 Purpose, Scope and Applicability: -

1.3.1 Purpose: -

   Playing games like Wordle allows us to use the brain for dynamic activities, rather than something passive. Your brain is working harder when you focus on something that forces your mind to be operating continuously, such as a board game or a book. This is especially true when compared to a more passive activity, such as watching a scripted TV show.

1.3.2 Scope: -

   Works on all type of devices.
   Requires internet connectivity.
   Will be available for all users.

1.3.3 Applicability: -

   Wordplay, the buzzy daily word game is minimal by design. The game is a website, not an app. It updates every day with new word puzzle, with only one available at a time. Players get six attempts to guess the correct word, and site makes it easy to share the results on social media.

CHAPTER 2: -
Survey of Technologies

1. **MongoDB:**

    MongoDB is a No SQL database and it is a cross-platform, open-source and document-oriented database written in C++. MongoDB is an open-source document database that provides high availability, automatic scaling and high performance. MongoDB is suitable for all the modern applications which requires big data, flexible deployment, big data and the older database systems which are not competent enough. Features of MongoDB are Indexing, Replication, Duplication of Data, Supports Map Reduce tools and it is Schema-less database language.

2. **ReactJS:**

    ReactJS is one of the most popular JavaScript front-end libraries which has a strong foundation and a large community. ReactJS is a flexible, declarative and efficient JavaScript library for building reusable User Interface (UI) components. ReactJS library is responsible only for the view layer of the application. The main objective of ReactJS is to develop User Interfaces (UI) that improves the speed of the apps. Virtual DOM (JavaScript object) helps ReactJS to improve the performance of the application. JavaScript virtual DOM is faster than the regular DOM. ReactJS can be used on both the client side and the server side and also with the other frameworks. It uses component and data patterns that improve readability and helps to maintain large apps.

3. **GitHub:**

    GitHub is an immense platform for code hosting which supports version controlling and collaboration. GitHub helps to host the source code of project in different programming languages and keeps a track of all the changes made by the programmers. It supports version controlling and collaboration and allow developer to work on project together. GitHub provide Distributed Version Control and Source Code Management functionality of Git. Bug Tracking, Feature Requests and Task Management collaboration features are provided for every project. GitHub make is easy to contribute to the open-source projects. Collaboration, Graphical Representation of Branches, Git Repositories Hosting, Project and Team Management, Code Hosting and Track and Assign Tasks are some significant features of GitHub.

4. **AngularJS:**

   AngularJS is an open-source JavaScript framework by Google to build web application. It can be used, changed and shared to anyone freely. It is an excellent framework for building single phase applications and line of business applications. AngularJS is designed in such a way that we test right from the start. Hence, it is easy to test any of the components through end-to-end testing and unit testing. It is perfect for Single Page Applications. It is continuously growing and expanding framework which provides better ways for developing web applications.

5. **Flutter:**

   Flutter is a User Interface toolkit for building fast, beautiful, natively compiled applications for desktop, web and mobile with one programming language and single codebase. Flutter is open-source and free. Flutter was developed from Google and it is now managed by an ECMA standard. Flutter applications uses Dart programming language for creating an application. Developing a mobile application is a very complex and challenging task and there are many frameworks which provide excellent features to develop mobile applications. Previously, to develop and application for both Android and iOS operating system, we needed to different languages and framework but now there are several frameworks which supports both OS along with desktop applications know and cross-platform. Hence, Flutter was a new framework which was introduced by Google in the cross-platform development family. Flutter uses its own high-performance rendering engine to draw widgets. It gives easy and simple methods to start building beautiful mobile and desktop apps with rich set of material design and widgets.

Why this language?

ReactJS:

React JS is basically a JavaScript library built and maintained by Facebook. According to the creator of React JS, Jordan Walke, react is an efficient, declarative, and flexible open-source JavaScript library for building simple, fast, and scalable frontends of web applications. Ever since its launch, it has taken the front-end development space by storm.

MongoDB:

MongoDB is built on a scale-out architecture that has become popular with developers of all kinds for developing scalable applications with evolving data schemas. As a document database, MongoDB makes it easy for developers to store structured or unstructured data. It uses a JSON-like format to store documents. This format directly maps to native objects in most modern programming languages, making it a natural choice for developers, as they don't need to think about normalizing data. MongoDB can also handle high volume and can scale both vertically or horizontally to accommodate large data loads.

CHAPTER 3:

Requirements and Analysis

3.1 Problem Definition: -

The goal of the project is to write a web application for playing the game of Wordplay, a popular internet game. The objective of the game is finding a secret word. Like Mastermind, the player makes guess and for each guess, receive some feedback from the program.

The program selected as to develop a system that will promotes your word recall and vocabulary your word recall and vocabulary building. We sometimes find struggling ourself to find a correct word for a specific period of a time. To counter this problem, we are developing this system.

- **Sub problem: -**
  - People face problem in vocabulary.
  - The players are not able to play together in multiplayer mode.

3.2 Requirement Specification: -
3.2.1 Requirement Gathering: -
3.2.1.1 Ways of Requirement Gathering: -

- **One-on-One Interviews: -**Introduce yourself and summarize the project including its scope and any timeline. Prepare some pre-determined questions to ask the interviewee.
- **Group Interviews:** - Group interviews are best for interviewees of same job position or level as they are expert in all those topics.
- **Brainstorming:** - Brainstorming helps gather many ideas as possible from as many people as possible to identify, categorize and assign tasks, opportunities and potential solutions quickly.
- **Survey:** - Survey is preparing a questionnaire which allows us to collect information from many people in a relatively short amount of time. Most of the questions are yes and no questions. Google Forms is one of the best-way to do a survey.
- **Prototyping:** - In prototyping, an early version is created to show to the client on the basis of surveys and brainstorming sessions.

3.2.1.2 Technique I used for Requirement Gathering: -

One-to-one Interview is the most widely used technique of requirement gathering. Theclients are interviewed in this technique. Some questions are asked to clients related to systemto get the need and expectation of client from the system. According to answers given by the client, analysis is done and system requirements are prepared.

> **Question and Answers: -**

1. What is the need of this Wordplay game?
   > To improve your vocabulary.
2. What things are focused while developing this Game?
   > Playing games like Wordle allows us to use the brain for dynamic activities, rather than something passive.
3. In which manner do you think this game will help you to grow?
   > It's perfect for those lazy days when you want to relax on your couch and play a quick game or two. Main motive is to improve our word dictionary.
4. How many data space will required for such kind of systems?
   > Not much.
5. What kind of interface should be created for such system?
   > Clean and easily handy interface would be a great option for such kind of systems.

3.2.2 Requirement Analysis: -
3.2.2.1 Functional Requirements: -

o **Login with authentication: -** As a social media website the need for a proper login and logout methods with authentication is needed for safety of user and safer experience.
o **Knowledge/Information sharing:** - The functionalities of new information added, information changing, knowledge providing, knowledge searching is important for knowledge sharing.
o **The comment and share system:** - The users will need a way to interact with other users but still be able to maintain the privacy. The liking and comment system will enhance the experience of website using website.

3.2.2.2 Non-functional Requirements: -

o **Performance: -** Performance of a website should be fast and smooth. The website should be quick and not take much time to load after it's loaded once.
o Software system need developed easily to add new functionalities and delete unwanted capabilities.
o **Scalability: -** Website should work as expected even when it scales.
o **Usability:** - Website's interface should be clean, good looking and easy to access.

3.2.2.3 System Requirements: -

1. **Register: -**

   - **Function: -** To register
   - **Description:** - To register once to create your account and have necessary information save
   - **Inputs:** - ID, password, email, number

- **Output:** - User is registered
- **Destination:** - Database
- **Action:** - All the provided information is saved in database
- **Pre-condition:** - User needs and email or phone number

2. **Login: -**

- **Function: -** To log in to your registered account
- **Description:** - To log into your requirements your previous states and changes saved
- **Input:** - ID, password
- **Output:** - User logged in to the system
- **Destination:** - Database
- **Action:** - User being successfully logged in to system without any errors
- **Pre-condition:** - User needs to provide login ID and password

3. **Keyboard Permission: -**

- **Input: -** Dialogue box for asking access to keyboard
- **Output:** - Display keyboard
- **Source:** - The device currently in use
- **Destination:** - Database
- **Action:** - The current word typed by keyboard and it is sent to backend

4. **Result: -**

- **Input: -** To solve the puzzle
- **Output:** - To show result
- **Function:** - Sharing result option after solving the puzzle
- **Destination:** - Database
- **Action:** - User will get a pop-up window to show the result with other peoples
- **Pre-condition:** - User needs to solve the puzzle

3.3 Planning and Scheduling: -

3.3.1 Activity Table: -

Activity No.	Chapter No.	Chapter Name	Start Date	End Date
T1	0	Synopsis	15-06-2022	02-07-22
T2	1	Introduction	20-06-2022	02-07-22
	2	Survey of Technologies		
T3	3	Requirements and Analysis	27-06-2022	02-07-22
	3.1	Problem Definition		
T4	3.2	Requirement Specification	04-07-2022	11-07-22
	3.2.1	Requirement Gathering		
	3.2.2	Requirement Analysis		
T5	3.2.3	System Requirements	11-07-2022	18-07-22
T6	3.3	Planning and Scheduling	18-07-2022	25-07-22
	3.4	Software and Hardware Requirements		
	3.5	Conceptual Models		
T7	3.5.1	Data Model	25-07-2022	01-08-22
T8	3.5.2	Data Flow Diagram	15-08-2022	22-08-22
T9	3.5.3	Class Diagram	22-08-2022	29-08-22
T10	3.5.4	Use Case Diagram	29-08-2022	12-09-22
T11	3.5.5	Sequence Diagram	12-09-2022	14-09-22
T12	3.5.6	Activity Diagram	14-09-2022	19-09-22
	3.5.7	State Diagram		
T13	3.5.6	User Interface	19-09-2022	26-09-22
T14	3.5.5	Test Cases	19-09-2022	26-09-22

Table 3.1 Activity Table

3.3.2 Gantt Chart: -

Activity No.	June				July						August						September					
	15	20	25	30	05	10	15	20	25	30	05	10	15	20	25	30	05	10	15	20	25	30
T1	■																					
T2		■																				
T3			■																			
T4					■																	
T5						■																
T6							■	■														
T7								■	■	■	■	■	■									
T8														■								
T9																■	■					
T10																		■				
T11																			■			
T12																				■		
T13																					■	
T14																					■	■
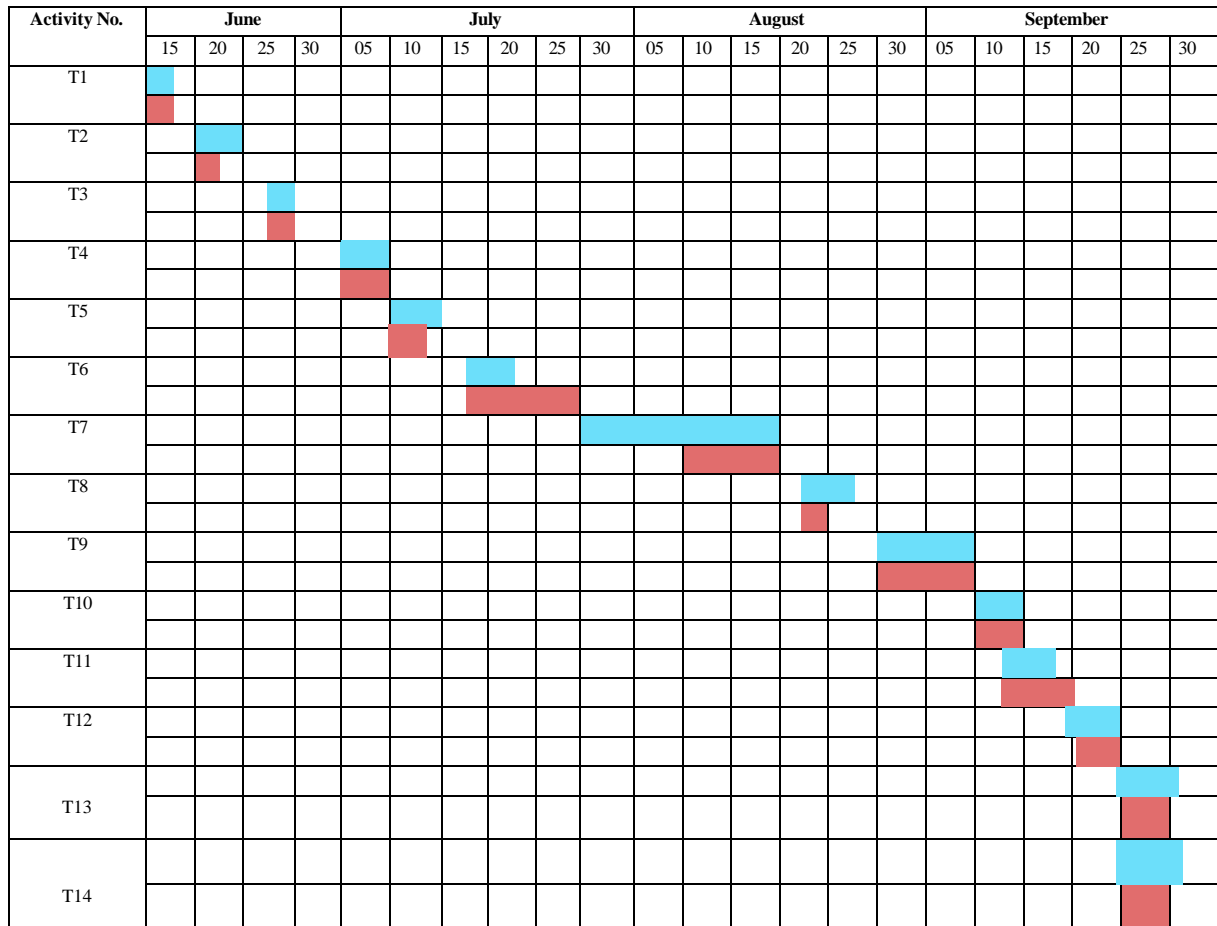
Fig 3.1 Gantt Chart

3.4 Software and Hardware Requirements: -
3.4.1 Software Requirements: -

- **Frontend: -** HTML, CSS, JavaScript
- **Backend: -** Nodejs, ExpresJS
- **Database:** - MongoDB

3.4.2 Hardware Requirements: -

- **Operating System: -** Windows
- **RAM:** - 2GB or more
- **ROM:** - 500MB of disk space available
- **Processor:** - Intel Duo+ or AMD Ryzen 3+

3.5 Conceptual Models
3.5.1 Data Model: -

The key challenge in data modelling is balancing the needs of the application, the performance characteristics of the database engine, and the data retrieval patterns. When designing data models, always consider the application usage of the data (i.e., queries, updates, and processing of the data) as well as the inherent structure of the data itself.

```
User
{
   _id: <ObjectID>,
   username: <string>,
   password: <string>,
   name: <string>,
   email: <string>,
   phone_no: <string>,
   dob: <ISODate>
}
```

```
Leadeboard
{
   _id: <ObjectID>,
   user_id: <ObjectID>,
   username: <string>,
   user_rank: <int>,
   description: <string>,
   score: <int>
}
```

```
Score
{
   _id: <ObjectID>,
   user_id: <ObjectID>,
   username: <string>,
   score: <int>
}
```

```
Comments
{
   _id: <ObjectID>,
   user_id: <ObjectID>,
   comment: <string>,
   username: <string>,
   createdAt : <ISODate>,
   upadtedAt: <ISODate>
}
```

```
Platform
{
   _id: <ObjectID>,
   name: <string>
}
```
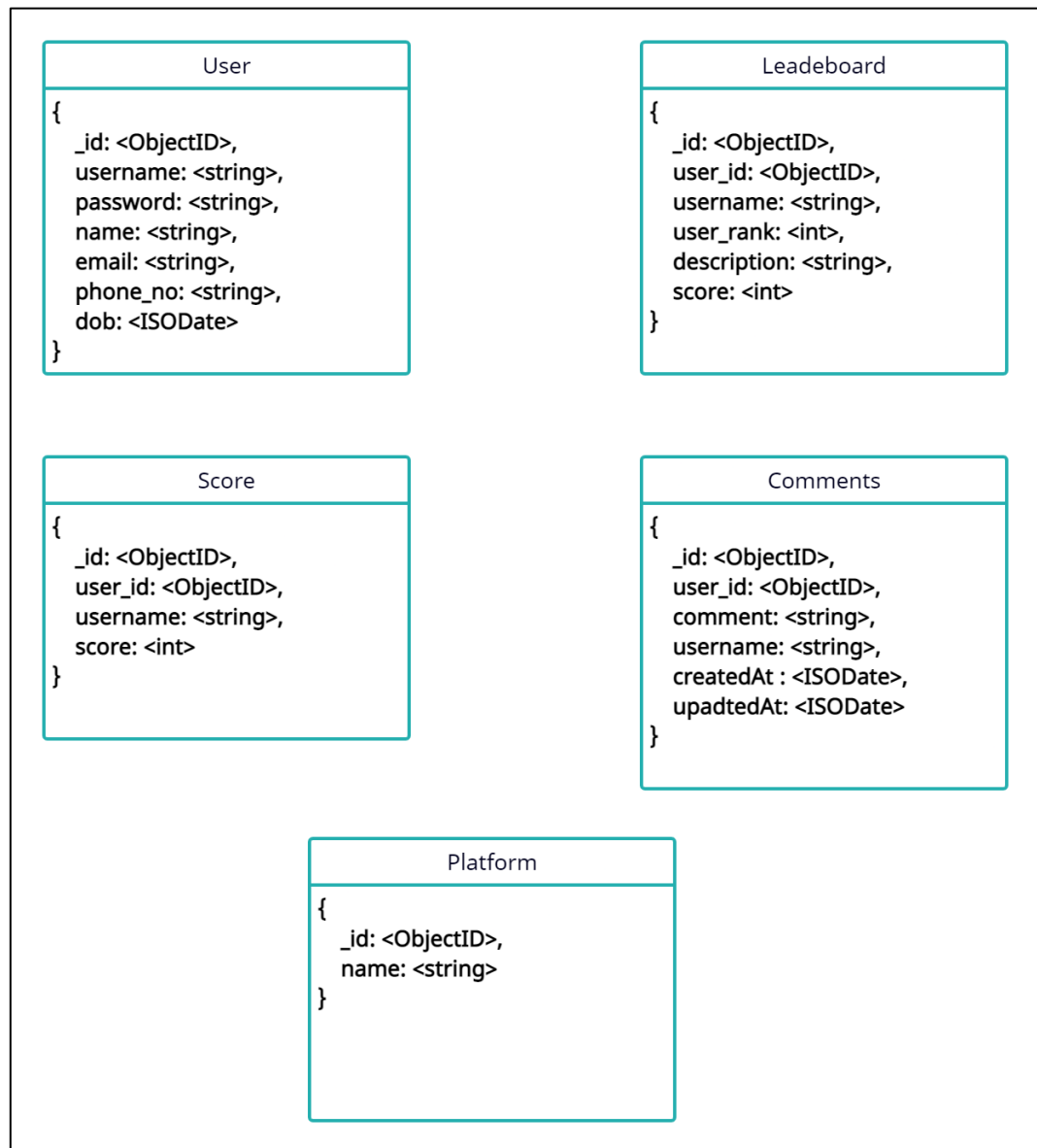
Fig 3.2 Data Model

3.5.2 Data Flow Diagram: -

A Data Flow Diagram maps out the flow of information for any system or process. It uses symbols like circles, rectangles, arrows with short text label, storage points and the routesbetween each destination. Data Flow Diagrams range from simple i.e., hand-drawn process overviews to in-depth multi-level Data Flow Diagrams that dig progressively deeper into howthe data is handled. They can be used to analyses an existing system or model a new one.

Data Flow Diagram works properly for the real-time or data-oriented software and systems. The symbols used to draw a Data Flow Diagram are:

Name	Symbol	Description
Rectangle		A recta1gle is used to show an external entity
Rounded Corner Rectangle		A rounded corner rectangle is used to define process of data flow
Data Store		This symbol is used to define the data entity storage
Line		The line symbol describes flow of data

Table 3.2 Data Flow Diagram Symbols

Diagram: -

1. **Context Level: -**
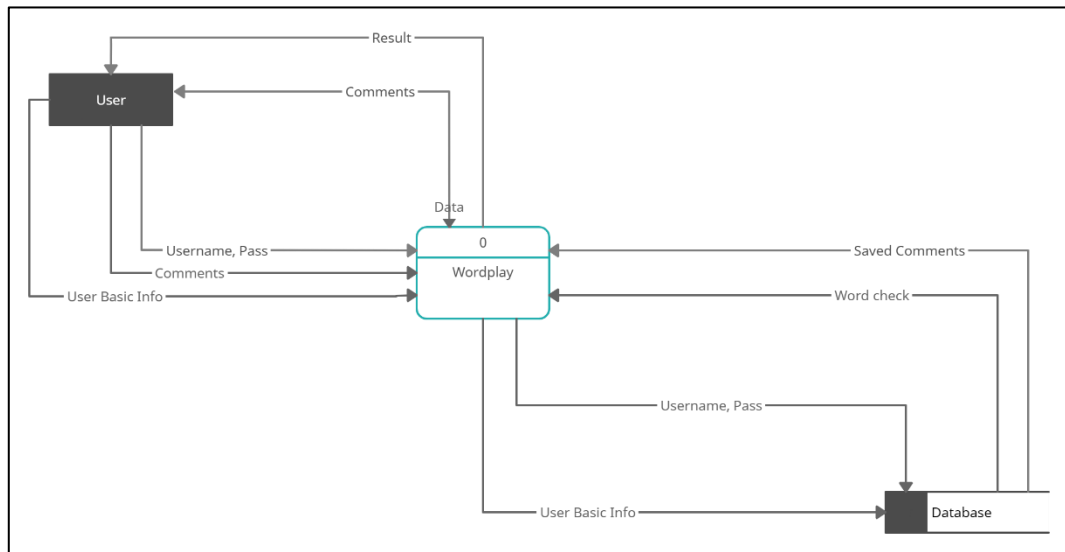
Fig 3.3 Context level Diagram

2. **Level 1 Diagram: -**
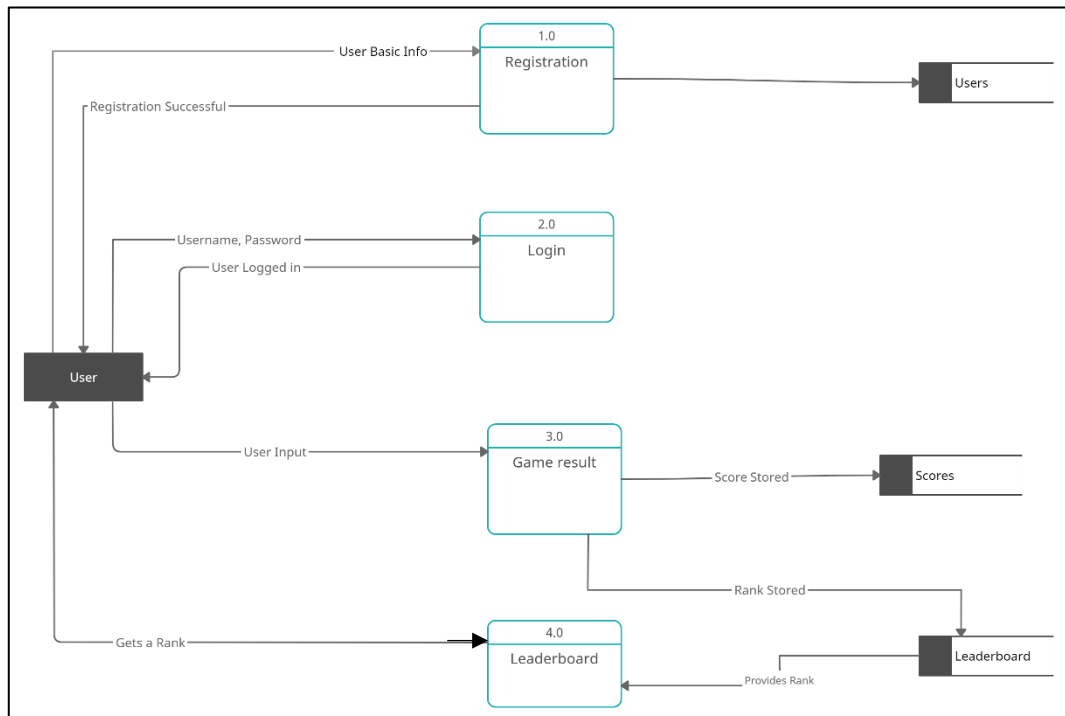
Fig 3.4 DFD level 1 Diagram

3.5.3 Class Diagram: -

Class diagrams are used in developing an object-oriented system model to show the classes in a system and the associations between these classes. An object class is a general definition of one kind of system object. An association is a link between classes that indicatesthat there is a relationship between these classes. Each class may need to have some knowledgeof its associated class. The symbols used in class are as follow:

Name	Symbol	Description
Class	**Name** + attribute1: type - attribute2: type# atrribute3: type / attribute4: type +operation1()	This box is used to define a class and the attributes and operation are listed in it along with access modifiers
Public	+	The plus sign states that attribute can be accessed in any other class
Private	-	The dash sign state that attribute cannot be accessed by any other class
Composition	◆———	A composition states the one class is totally depended on the other class
Aggregation	◇———	An aggregation states that one class is partially associated with the other class
One	——— 1	This describes only one entity participation in the association
Many	——— 1..*	This describes one or more entity participation in the association

Table 3.3 Class Diagram Symbols

References: https://www.edrawsoft.com/article/class-diagram-relationships
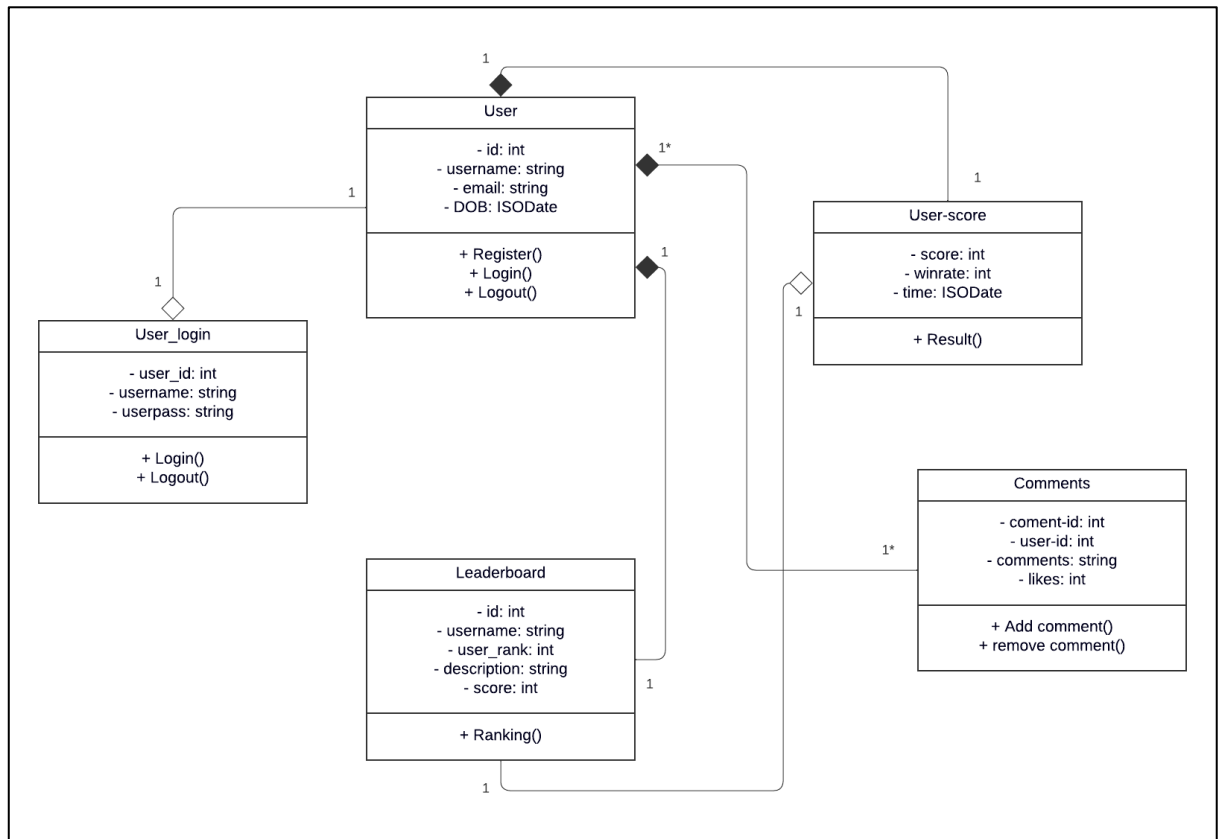
Diagram: -

Fig 3.5 Class Diagram

3.5.4 Use Case Diagram: -

Use Case Diagram is used to show interaction between the system and its environment. A use case can be taken as a simple scenario that describes what a user expects from a system. Each use case represents a discrete task that involves external interaction with the system. A use case is shown in ellipse with the actors involved in the use case. Use Case Diagram uses line without arrow and line with arrow indicates the direction of flow of messages. The symbols used to create a Use Case Diagram are as follow:
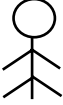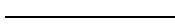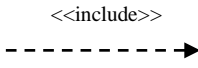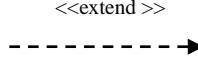
Name	Symbol	Description
Actor		The actors are used to define the environment interacting with the system
Use Case		A Use Case describes the functionality of the system
Association		An association is used to show interaction of actors with use cases
Include	<<include>>	This association states that the base use case is executed with the help of include use case
Extend	<<extend >>	The extend states that the extend use case will be executed after the execution of base use case but it will not always execute

Table 3.4 Use Case Diagram Symbols

Reference: Software Engineering 9th Edition

https://www.lucidchart.com/pages/uml-use-case-diagram
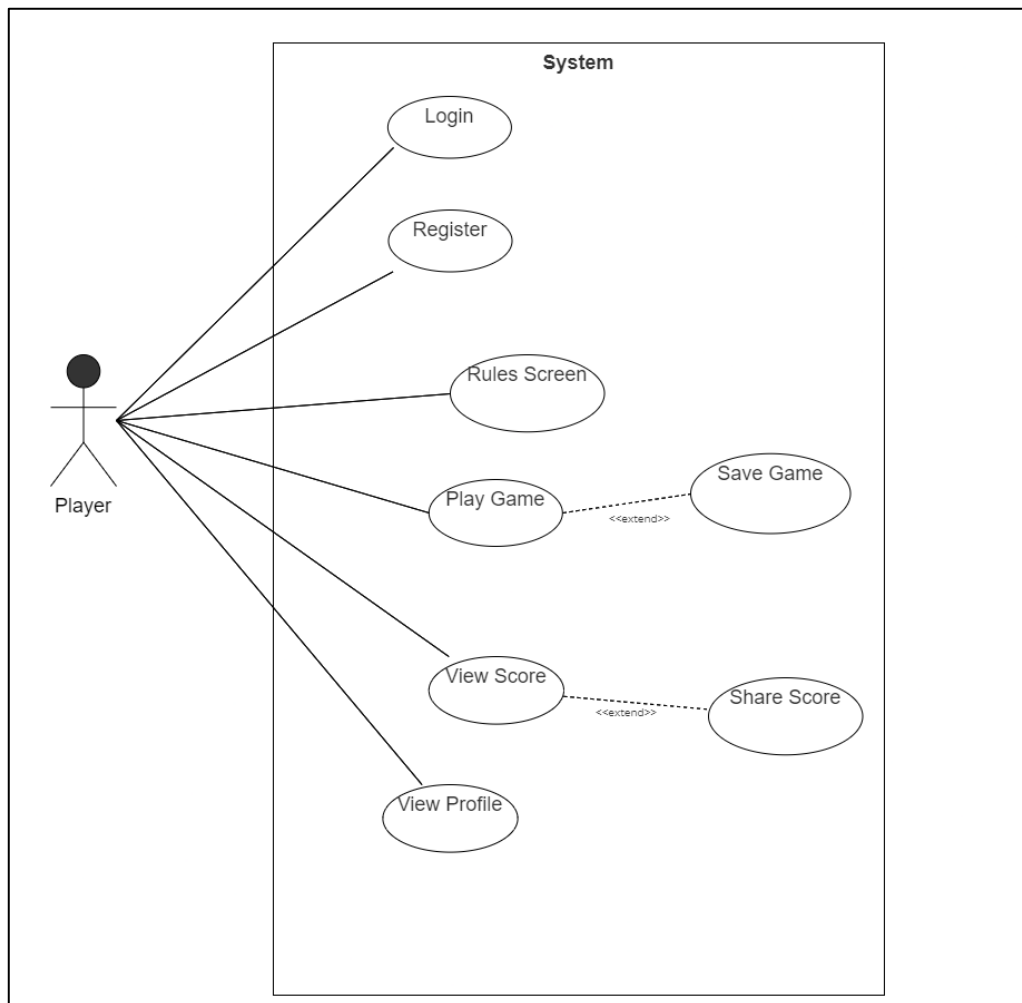
Diagram: -

Figure 3.6 Use Case Diagram

Description for Use Cases: -

1. **Use Case: - Register**
    - **Description –**
      > User needs to register their account, after which they get the username and password to log in to their account. To register, users will be given a form in which they have to fill in their personal details.
    - **Actors –** User
    - **Pre-condition –**
      > User needs to fill all the details asked in the form in the correct format.
    - **Exception –**
      > If the format of any detail is incorrect or any required field is kept empty, registration will not be successful.
    - **Post-condition –**
      > Registered Successfully. Username Password provided successfully

2. **Use Case: - Login**
   - **Description –**
     This use case describes the login process. The user needs to login to the system after creating an account, with username and password.
   - **Actors –** User
   - **Pre-condition –**
     The user/admin need to provide correct login and password to login to the system.
   - **Exception –**
     If any detail is not correct, the user/admin can't login to the system and has to try again.
   - **Post-condition –**
     Provided all the details correctly, the user will get logged in to the system.

3. **Use Case: - How to play**
   - **Description: -**
     User will get a screen how to play the given game. To understand the rules this is important.
   - **Actors: -** User
   - **Pre-condition: -**
     User should play the game.
   - **Post-condition: -**
     The how to play (rules) game screen will be displayed.

4. **Use Case: - Play Game**
   - **Description: -**
     User can play the game after knowing the rules.
   - **Actors: -** User
   - **Pre-condition: -**
     User must be logged in
   - **Post-condition: -**
     Game window provided to the user

5. **Use Case: - View Score**
   - **Description: -**
     If user wants to see the score of the games, then user should be logged in. If user is not logged in then the score will not be shown.
   - **Actors: -** User
   - **Pre-condition: -**
     User must be logged in
   - **Post-condition: -**
     If user is logged in score will be shown successfully.

3.5.5 Sequence Diagram: -

Sequence Diagrams in the UML are used to model the interactions between the actors and the object in system and the interaction between the objects themselves. The UML has a rich syntax for sequence diagrams, which allows many kinds of interaction to be modelled.

A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance. The objects and actors involved are listed along the top of the diagram with a dotted line drawn vertically from these. Interactions between objects areindicated by annotated arrows. The rectangle of the dotted lines indicates the lifeline of the object concerned i.e., the time that object instance is involved in the computation.

The sequence diagram is read from top to bottom. The annotations on the arrows indicate the calls to the object, their parameters, and the return values. The about symbols are used to draw a sequence diagram.
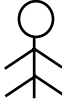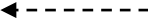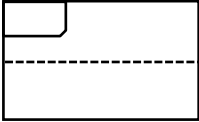
Name	Symbols	Description
Objects		This symbol is used to define the data objects
Actor		The actors are used to define the environment interacting with the system
Lifeline of the Object		The amount of time for which the object instance is involved in the computation.
Message		It is used to show the interactions between actor and objects.
Return Message		It returns a message for a request
Alternative Frame		This frame is used to show two different conditions and their working.

Table 3.5 Sequence Diagram Symbols

Diagram: -

1. **Register:**
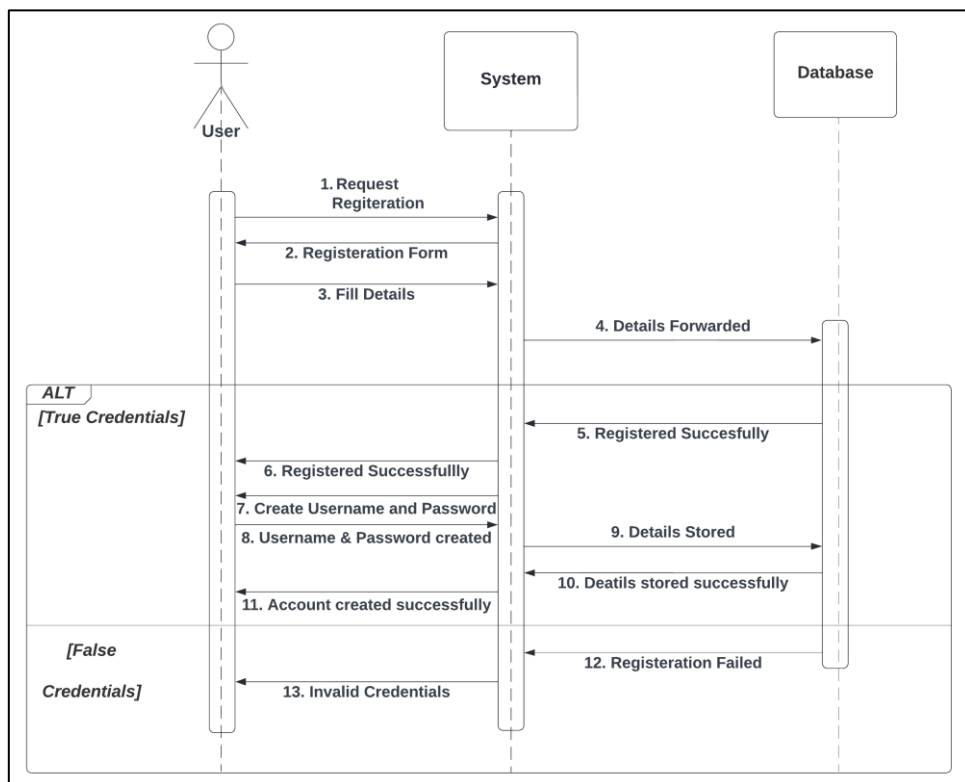
Fig 3.7 Sequence Diagram for Registration

2. **Login:**
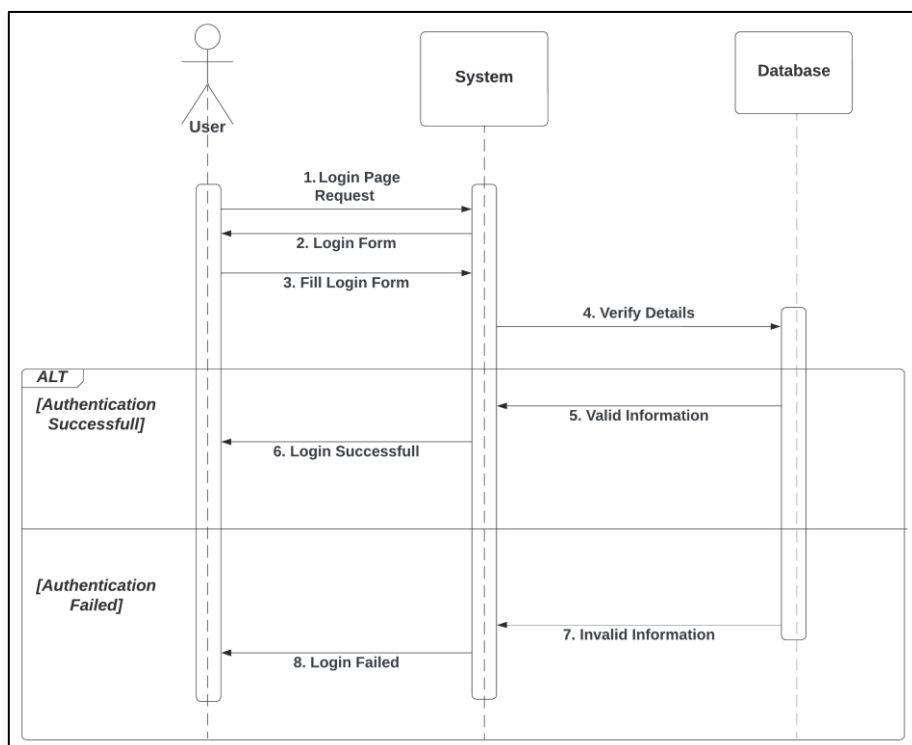
Fig 3.8 Sequence Diagram for Login

3. How to play (Rules):
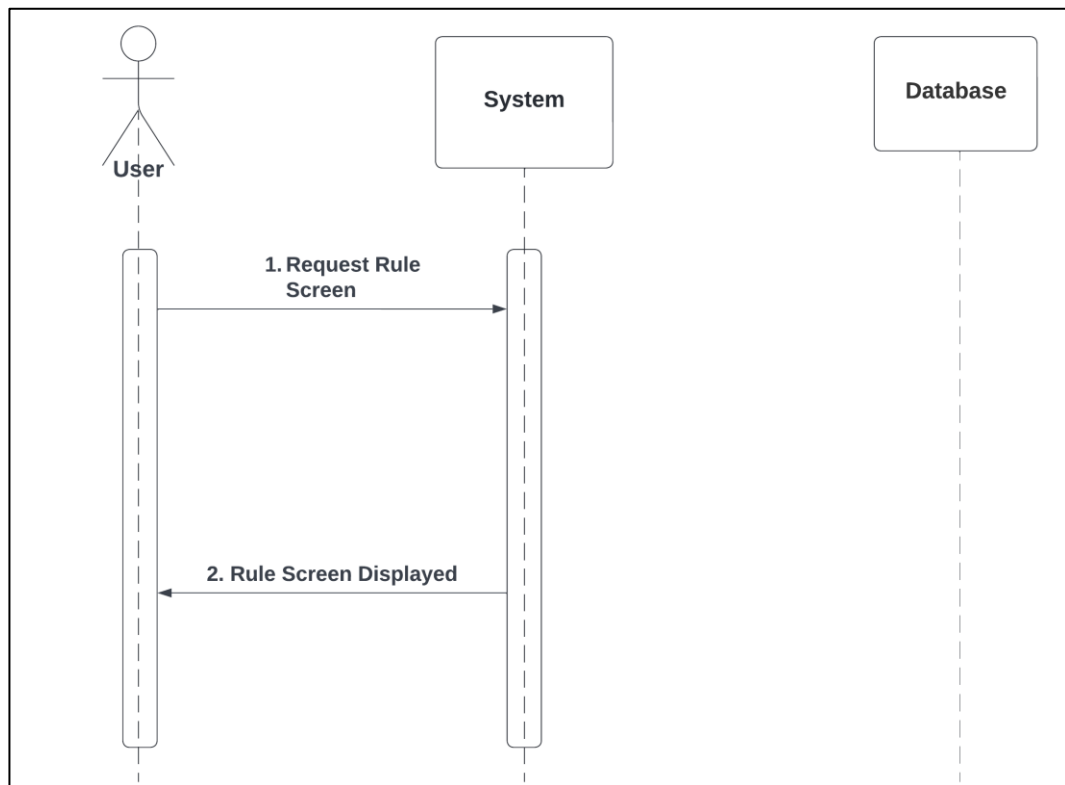
Fig 3.9 Sequence Diagram for Rules
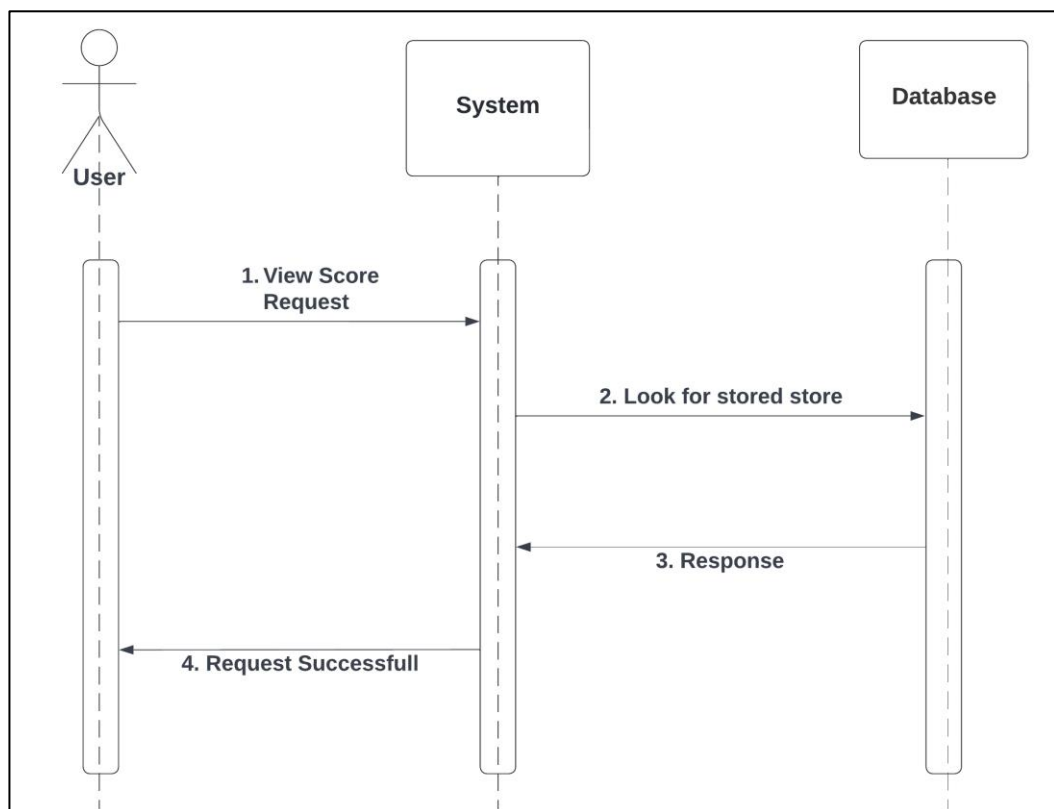
4. Score:
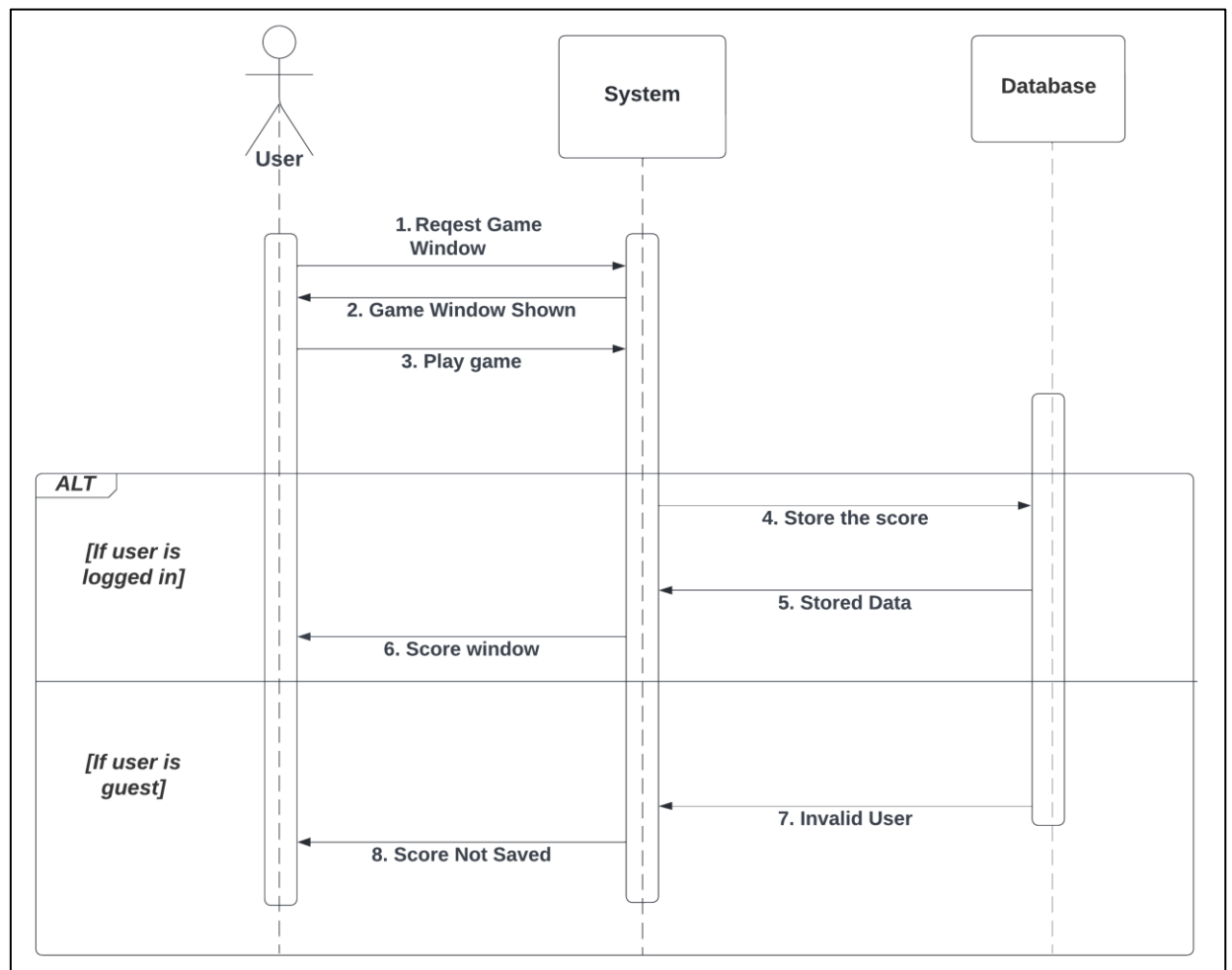

Fig 3.10 Sequence Diagram for Score

5. Game:

Fig 3.11 Sequence Diagram for Playing Game

3.5.6 Activity Diagram: -

An Activity Diagram show the activities involved in a process or in data processing. Activity Diagrams are intended to show the activities that make up a system process and the flow of control from one activity to another. The start of a process is indicated by a filled circleand the end by a filled circle inside another circle. Rectangles with round corners represent activities i.e., the specific sub-processes that must be carried out. An arrow is used to representthe flow of work form one activity to another.

A solid bar is used to indicate activity coordination. When the flow from more than oneactivity leads to a solid bar then all of these activities must be complete before progress is possible. When the flow from solid var leads to several activities, these may be executed in parallel. The symbols used in an Activity Diagram are:
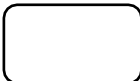
Name	Symbol	Description
Start	⭕	This symbol indicates start of an activity
End	◉	This symbol indicates end of an activity
Activity	▢	This symbol indicated the activities of a particular system.
Flow of Work	→	This symbol is used to show the flow of work from activity to activity
Decision	◇	This symbol represents the decision parameter and used where a decision is need to be taken

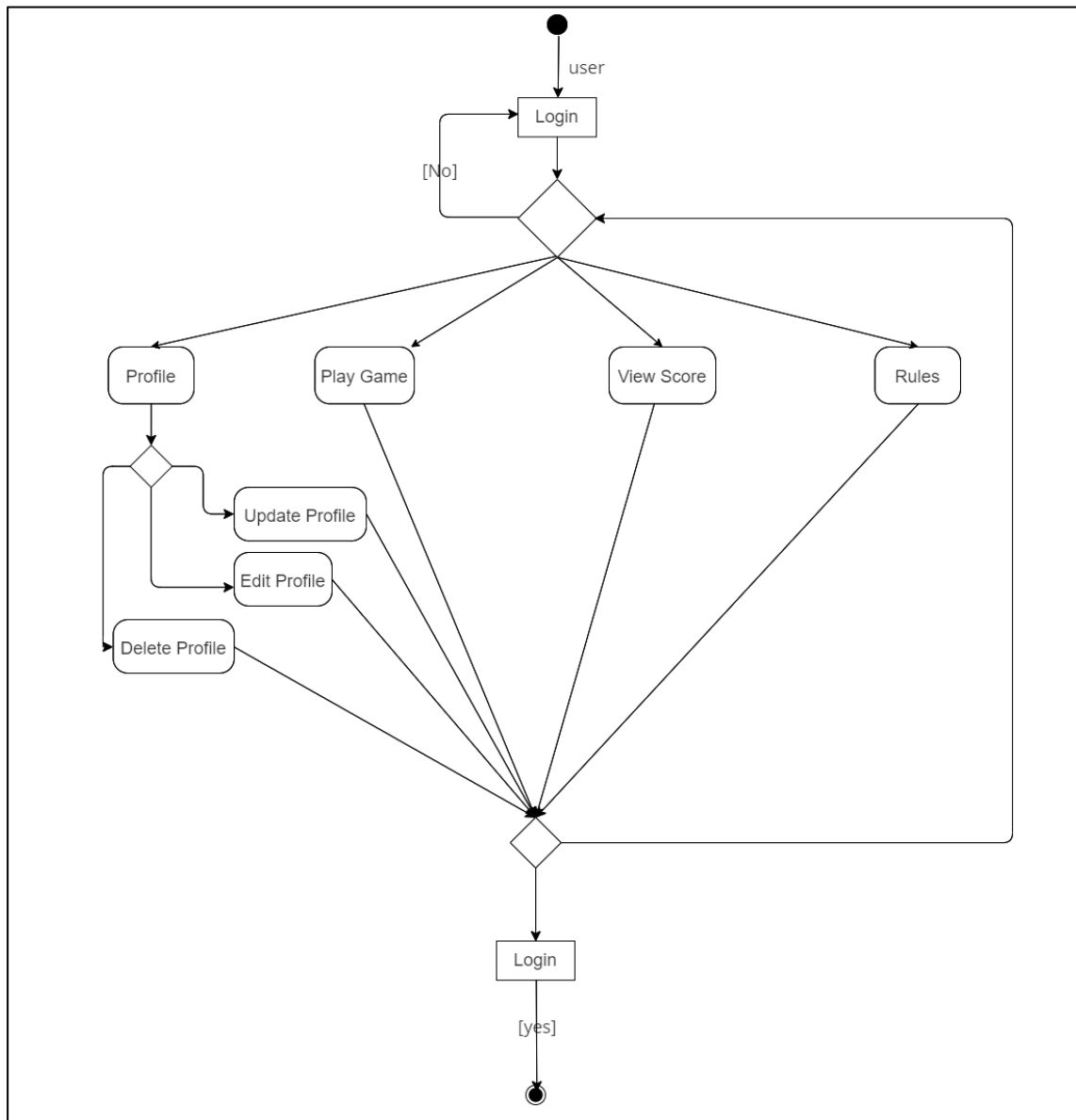***Table 3.6 Activity Diagram Symbols***

Diagram:

Fig 3.12 Activity Diagram

3.5.7 State Chart Diagram: -

A State-Chart Diagram show system states and events that cause transitions from one state to another. They do not show the flow of data within the system but may include additionalinformation on the computation carried out in each state. In State Chart Diagrams, rounder rectangles represent system states. They may include brief description of the actions taken in that state by using do keyword. The labelled arrow represents stimuli that force a transition from one state to another. It shows the states of a system and the events that trigger a transitionfrom one state to another. The symbols used to draw a State-Chart Diagrams are:
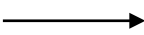
Name	Symbol	Description
Start	○	This symbol indicates start of an activity
End	◉	This symbol indicates end of an activity
Activity	▢	This symbol indicated the activities of a particular system.
Flow of Work	→	This symbol is used to show the flow of work from activity to activity

Table 3.7 State Chart Diagram Symbols

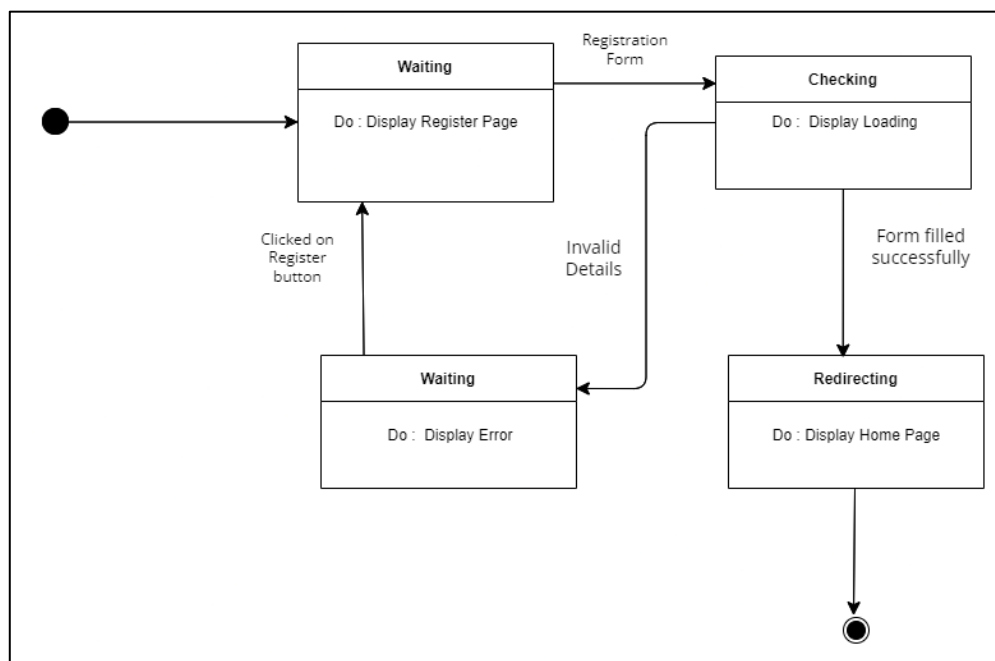**Diagram:**
1. **Register:**

Fig 3.13 State-Chart Diagram for Register
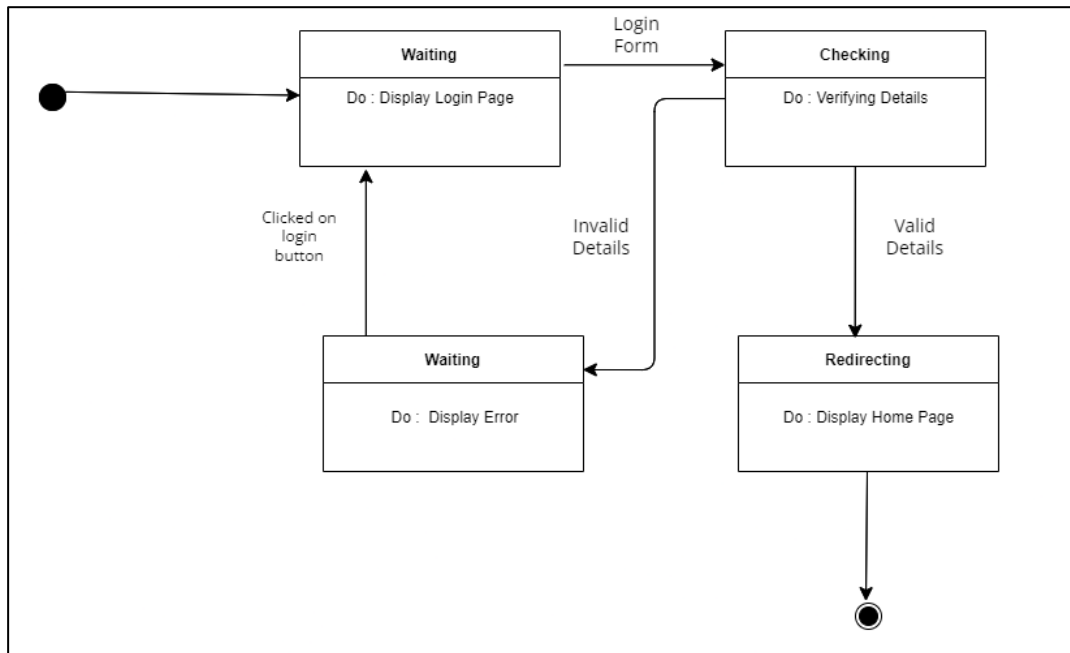
2. Login:

Fig 3.14 State-Chart Diagram for login
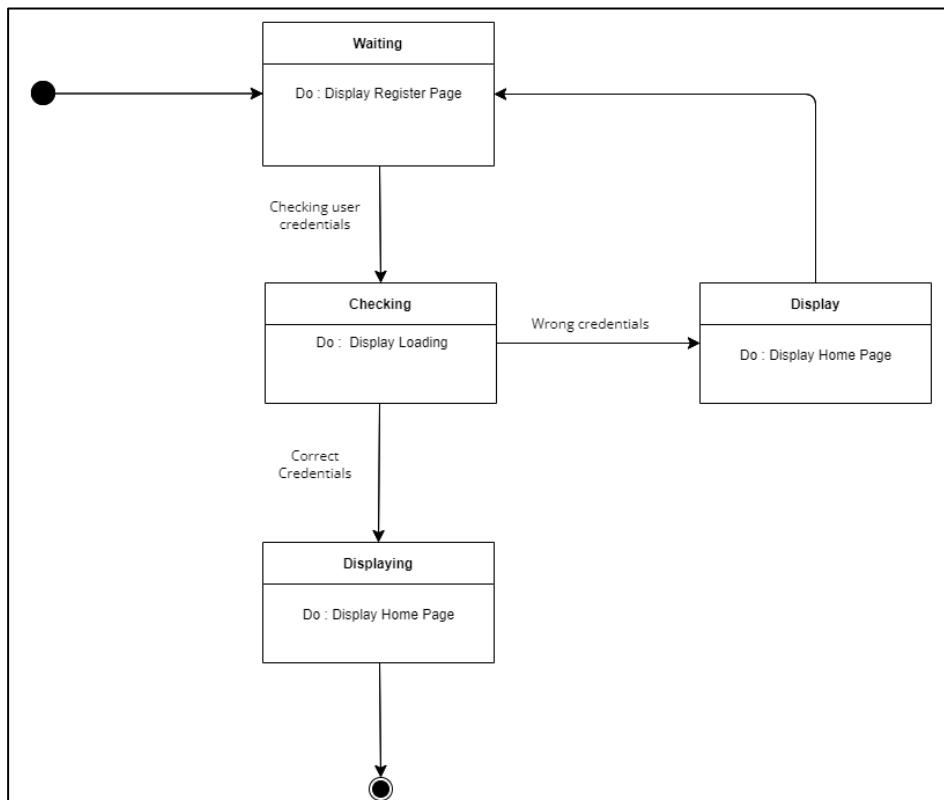
3. Score:

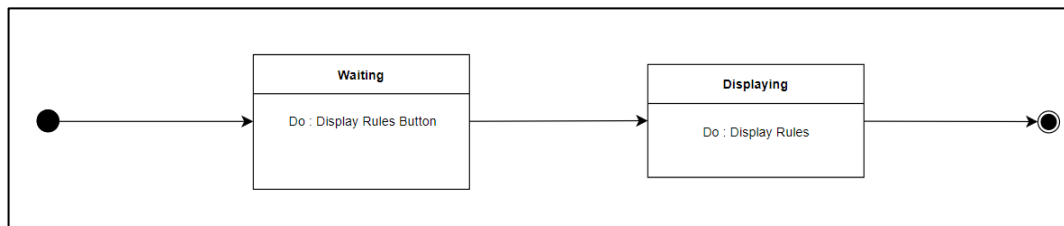
Fig 3.15 State-Chart Diagram for Score

4. Rules:

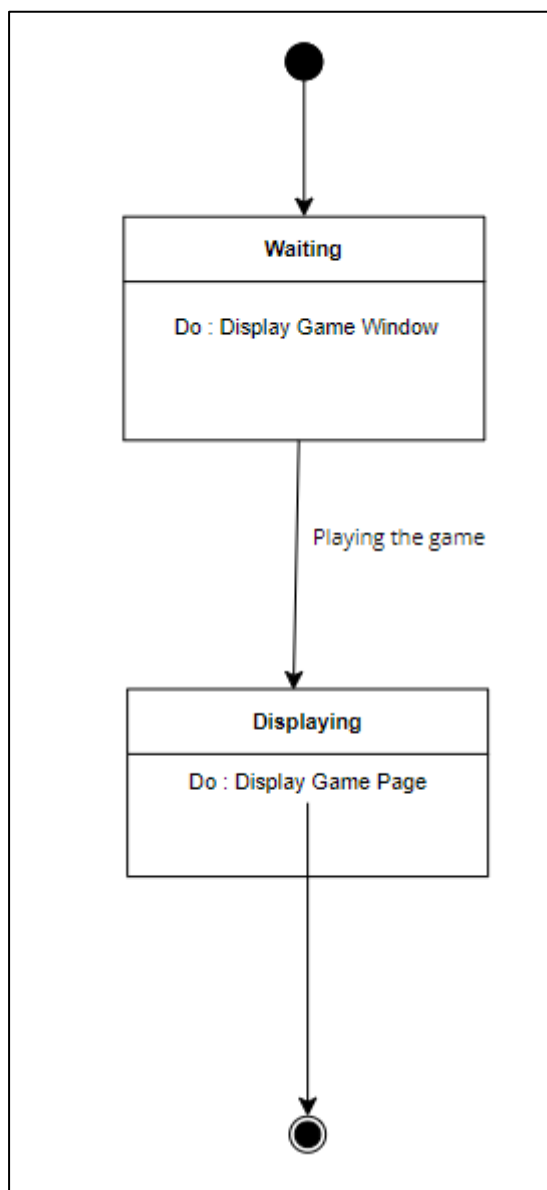*Fig 3.16 State-Chart Diagram for Rules*

5. Game:

Fig 3.17 State-Chart Diagram for Playing the game
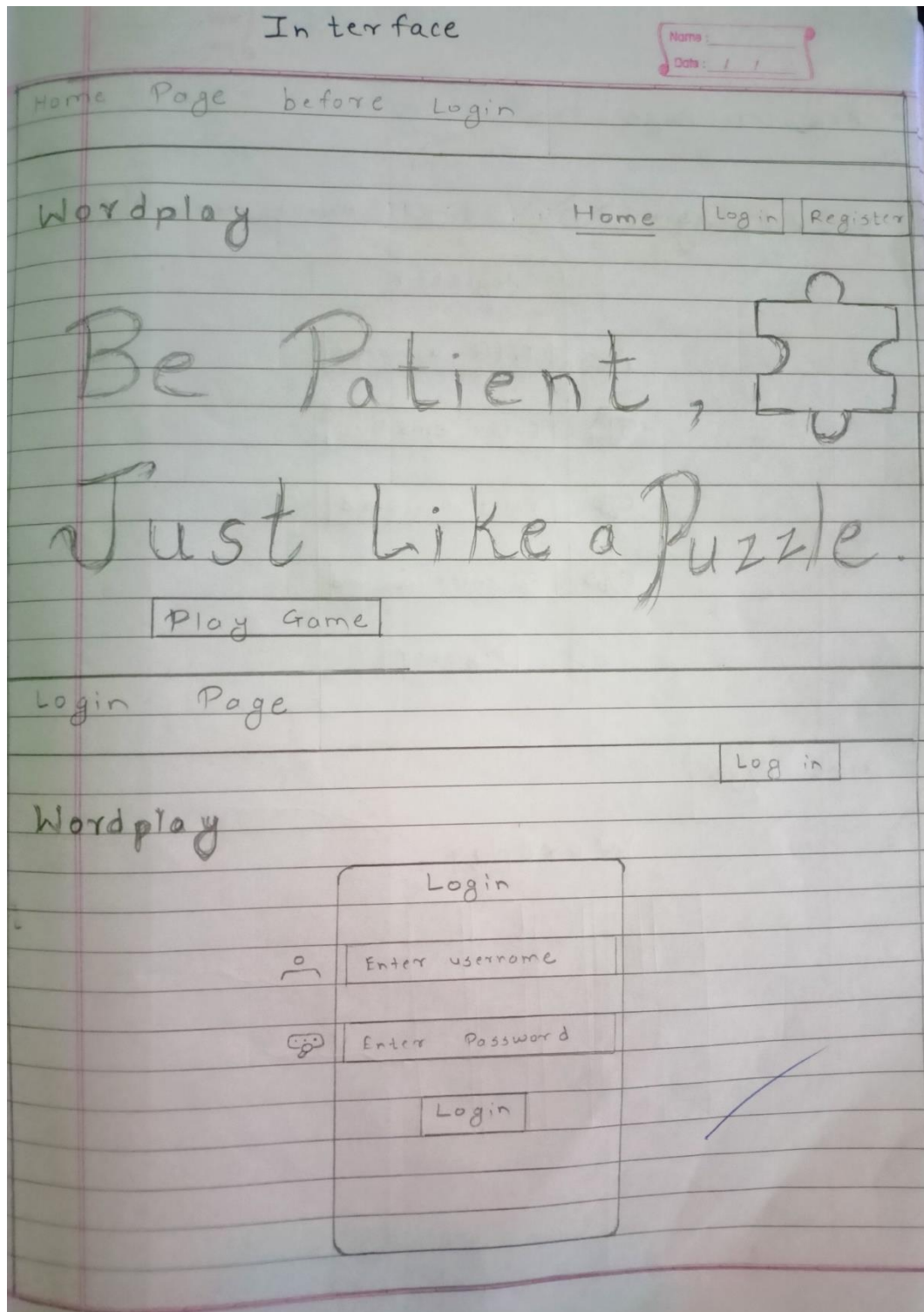
3.5.8 User Interface: -
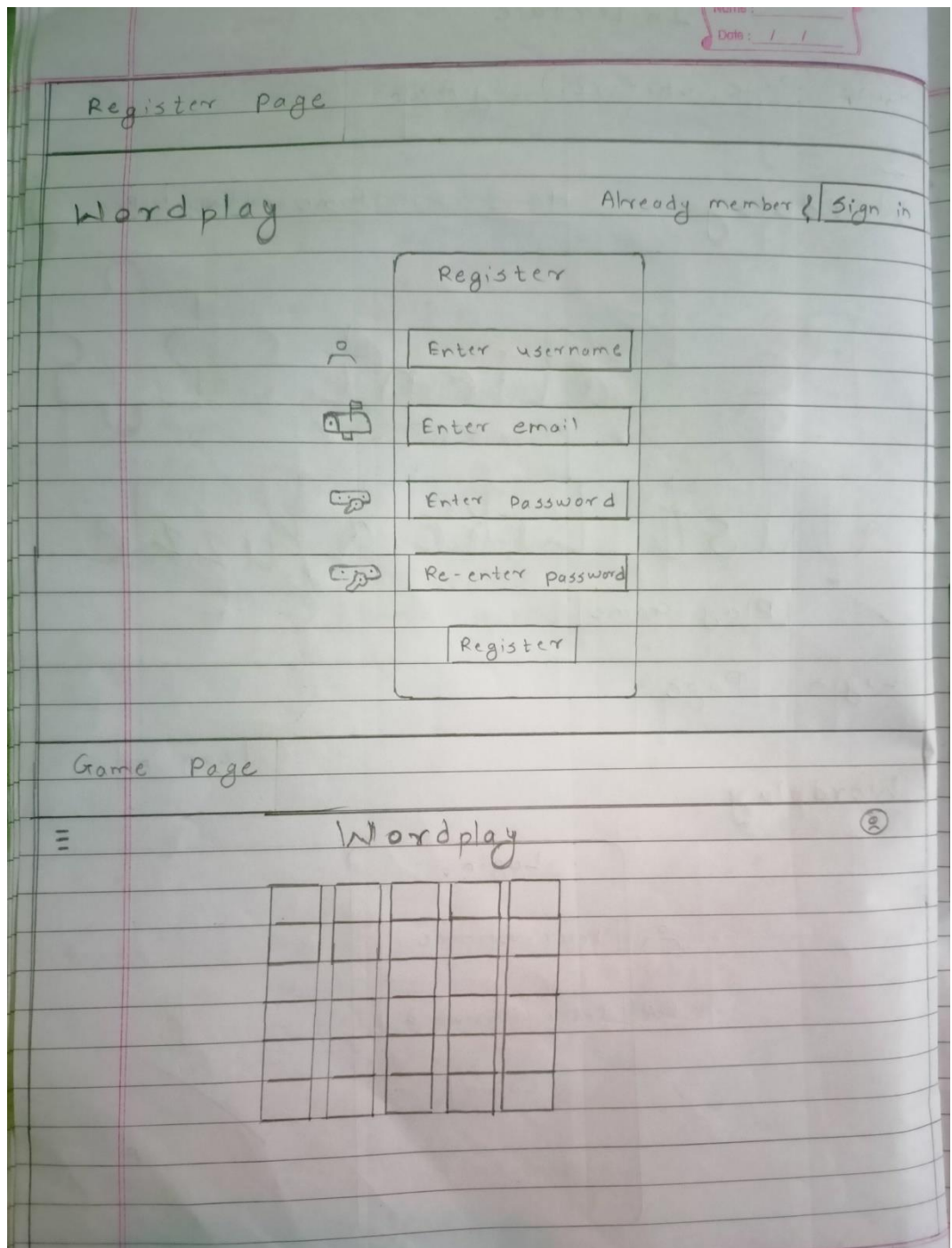

Fig 3.18 User interface of Home and Login Page
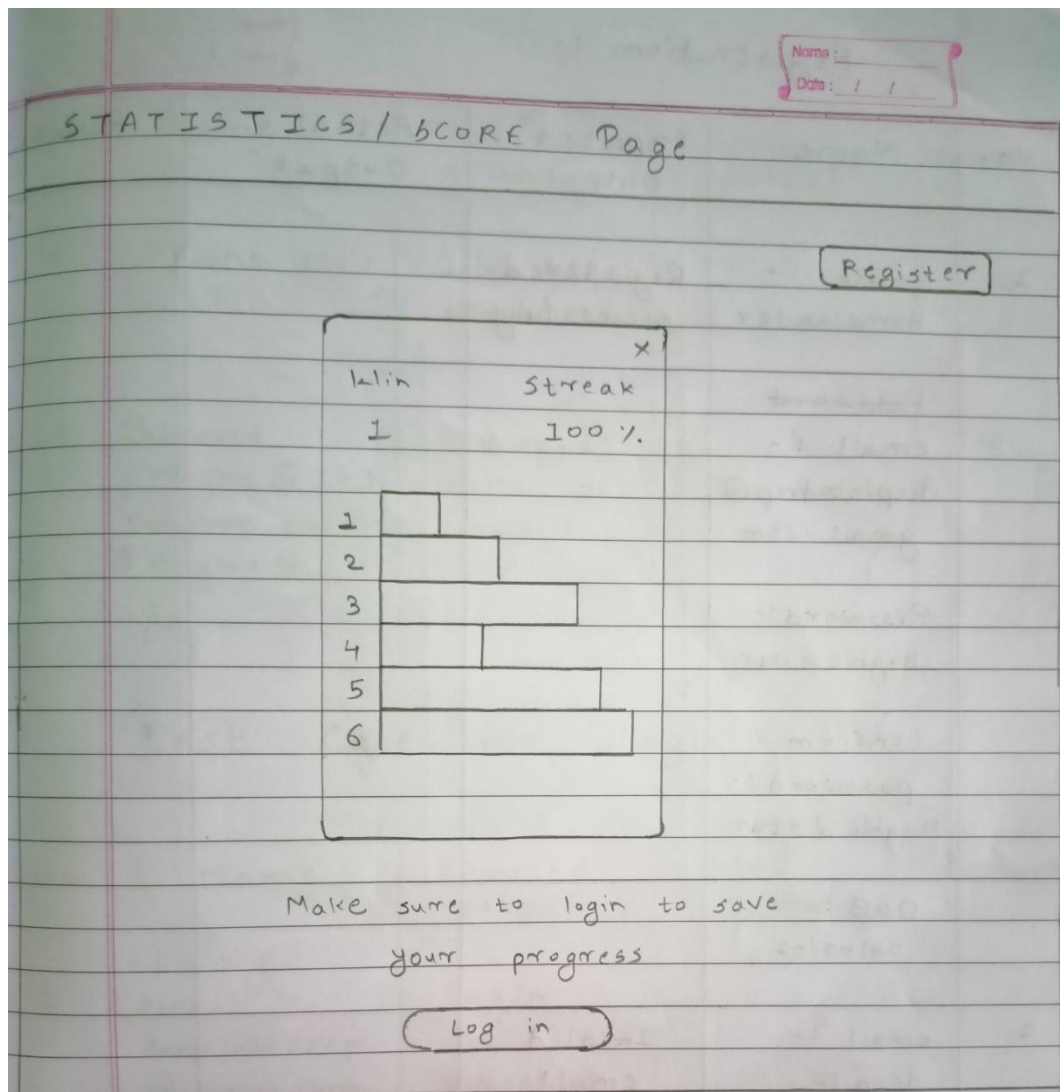
Fig 3.19 User interface of Register and Game Page

Fig 3.20 User interface of Score Page

3.5.9 Test Cases: -
3.5.9.1 Registration: -

SN NO	NAME	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARK
1	Email: pop@gmail.com Username: Poke038 Phone No: 91XXXX Password:12@Ph34 C-Password :1234 Address: B-8,Mulund.	Register Successfully.		
2	Email: popgmailcom	Invalid Email.		
3	Email: pop@gmail.com Password:1234	Password must contain Special character.		
4	Password:12@pH34 C-Password: 12@Ph34	Password Doesn't match.		
5	Email: pop@gmail.com Username: Poke038	User Already Exist.		
6	NULL	Please Enter Details.		

Table 3.8 Test Case for Registration

3.5.9.2 Login: -

SN NO	NAME	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARK
1	Username: Poke038 Password:12@Ph34	Login Successfully.		
2	Username: Poke038 Password:12@Ph35	Password Invalid		
3	Username: Poke038	Invalid Details		
4	NULL	Please fill all details.		

Table 3.9 Test Case for Login

3.5.9.3 Game: -