ONLINE GAMING WEBSITE

SOFTWARE ENGINEERING PROJECT REPORT

[SUBMITTED IN PARTIAL FULFILMENT]

As a part of the curriculum of

B.Sc. (H) COMPUTER SCIENCE

Submitted by:

ANJALI THAKUR (21075570012)

DIKSHA SHARMA (21075570025)

ISHIKA BHATT (21075570041)

KHUSHI SHARMA (21075570055)

SALONI KHANNA (21075570097)

B.Sc. (H) COMPUTER SCIENCE

Shyama Prasad Mukherji College for Women,

University of Delhi

Punjabi Bagh West, New Delhi-110002

ACKNOWLEDGEMENT

It gives me immense pleasure to present you to this Online Gaming Website. We were lucky to get enormous support from extremely talented people, who deserve our great gratitude.

Firstly, we would like to thank our teacher and guide, **Mr. Lavkush Gupta** who gave his valuable suggestions and ideas whenever we needed them. Also, he encouraged us to work on this project tirelessly by giving us numerous consultations.

We are also grateful to our parents for their constant support, guidance and providing us the necessary resources for the project.

Lastly, we would like to thank our classmates for their valuable suggestions for the betterment of this project and everybody who has helped us directly or indirectly in completion of this project.

We are immensely grateful to everyone involved in this project a without their inspiration and valuable suggestion it would not have been possible to develop the project within the prescribed time.

With sincere thanks,

ANJALI THAKUR (21075570012)

DIKSHA SHARMA (21075570025)

ISHIKA BHATT (21075570041)

KHUSHI SHARMA (21075570055)

SALONI KHANNA (21075570097)

CERTIFICATE

This is to certify that the project entitled, "**Online Gaming Website**", has been submitted by Anjali Thakur, Diksha Sharma, Ishika Bhatt, Khushi Sharma and Saloni Khanna in partial fulfilment of the requirements of Bachelor of Computer Science (Hons.) embodies the work done by them during, semester IV of their course under the supervision of **Mr. Lavkush Gupta, Assistant Professor,** Department of Computer Science, Shyama Prasad Mukherji College for Women, University of Delhi.

_____

Mr. Lavkush Gupta

(Project Guide)

TABLE OF CONTENTS

1. PROBLEM STATEMENT

Everyone loves to play games to pass time or just to have fun. There are multiple games available online to play on web and mobile. But it become troublesome when you have to use multiple websites or apps to play different games.

Therefore, to save the time and storage space our gaming website KIDS provide multiple games available all together at one place and anyone can play them for free.

Actors:

- Users
- Admin
- Developer

FEATURES:

This website provides following functionalities:

- Register: This functionality allows new users to first register in the website. Here users can create their profile by specifying the unique username and password.

- Login: This functionality allows the registered user to login into the website using username and password.

- Reset/Forgot Password: This feature is designed in order to reset the password in case the user forgets his/her password.

- Single and Multiplayer Selection: This functionality allows users to select from following modes:

- Single-player mode
- Multi-player mode

- Difficulty Level Selection: Using this feature users can select among various difficulty levels:
    - Easy
    - Intermediate
    - Difficult

- Round Tracking: This functionality helps to keep track of registered user's rounds.

- Scorekeeping: This functionality helps to save registered user's scores.

- Reset Functionality: This functionality allows users to reset the game.

- Quiz: This feature allows users to play quiz which is based on simple logics and facts.

- Chatbot: This feature help users to connect with a bot by written means for getting help and support or to solve queries regarding the website.

- Feedback: This functionality is used for taking feedback from the users to get review about our website and to get more ideas to make our software more efficient.

    - Rating
    - Review
    - Ways to improve the website

2. PROCESS MODEL

Model best suited for this project is **Incremental Process model.**

We have used the incremental model as it combines elements of linear and parallel process flows. It generates working software quickly and early during the software lifecycle. This model is more flexible and less costly to change scope and requirements. It is easier to test and debug during a smaller iteration. In this model, the customers can respond to each built. Also, functionality can be refined and expanded in the later stages in the later software releases. The user can visualize the software before the completion of the entire project in order to evaluate and provide feedback. We are using this model as requirements are completely understood, however, small changescan be incorporated.

ADVANTAGES OF INCREMENTAL MODEL

- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

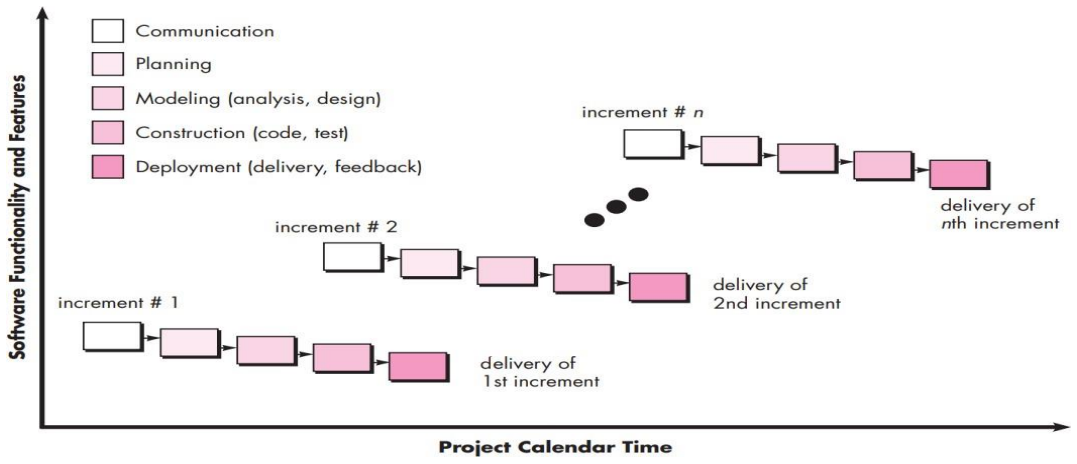


Fig 2.1 Incremental Model

3. REQUIREMENT ANALYSIS

3.1 DATA FLOW DIAGRAMS

3.1.1 CONTEXT LEVEL DATA FLOW DIAGRAM

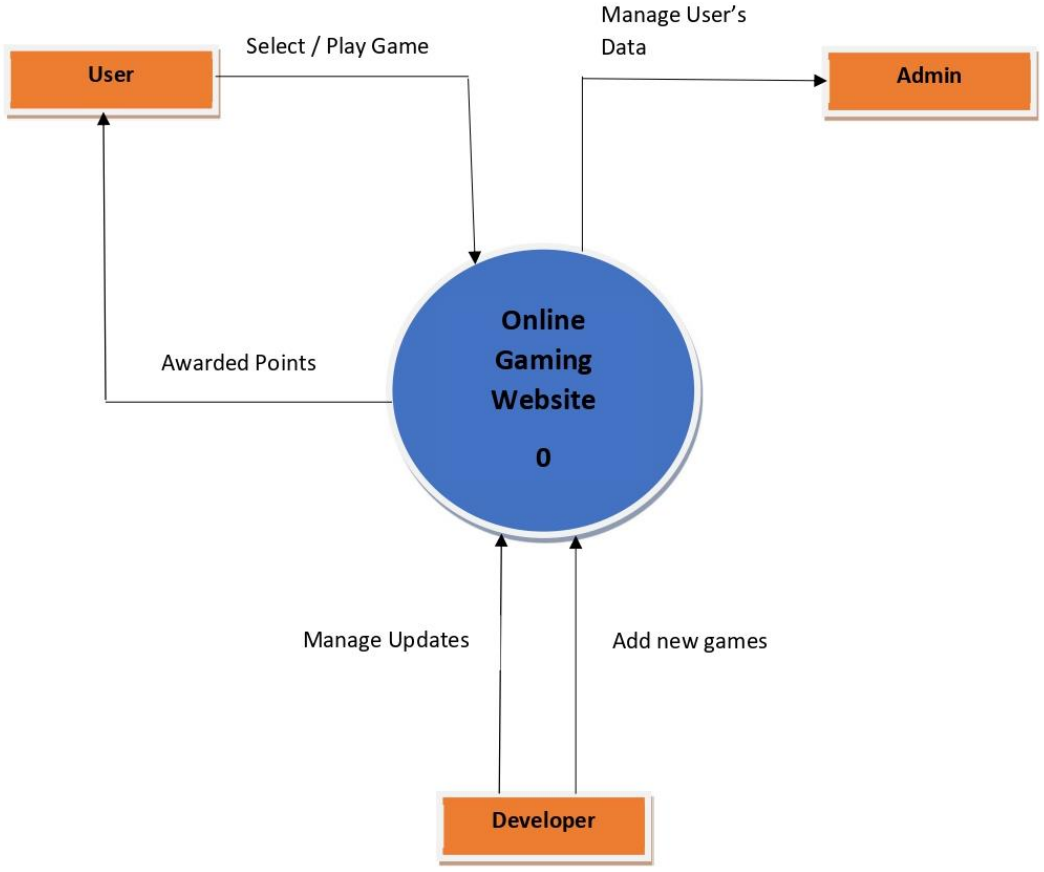


Fig 3.1 Context Level Diagram

3.1.2 LEVEL 1 DATA FLOW DIAGRAM

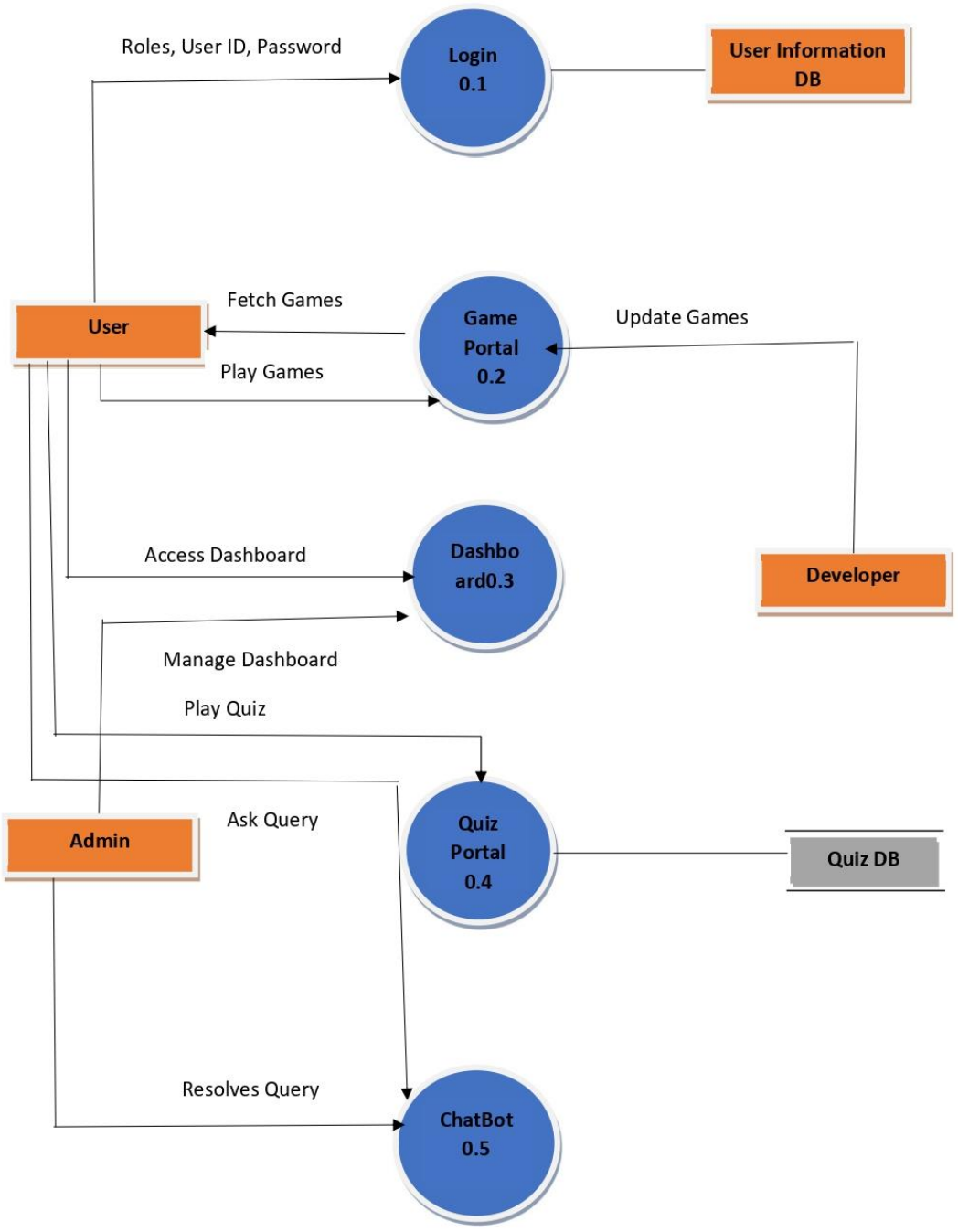


Fig 3.2 Level 1 DFD

3.1.3 LEVEL 2 DATA FLOW DIAGRAM

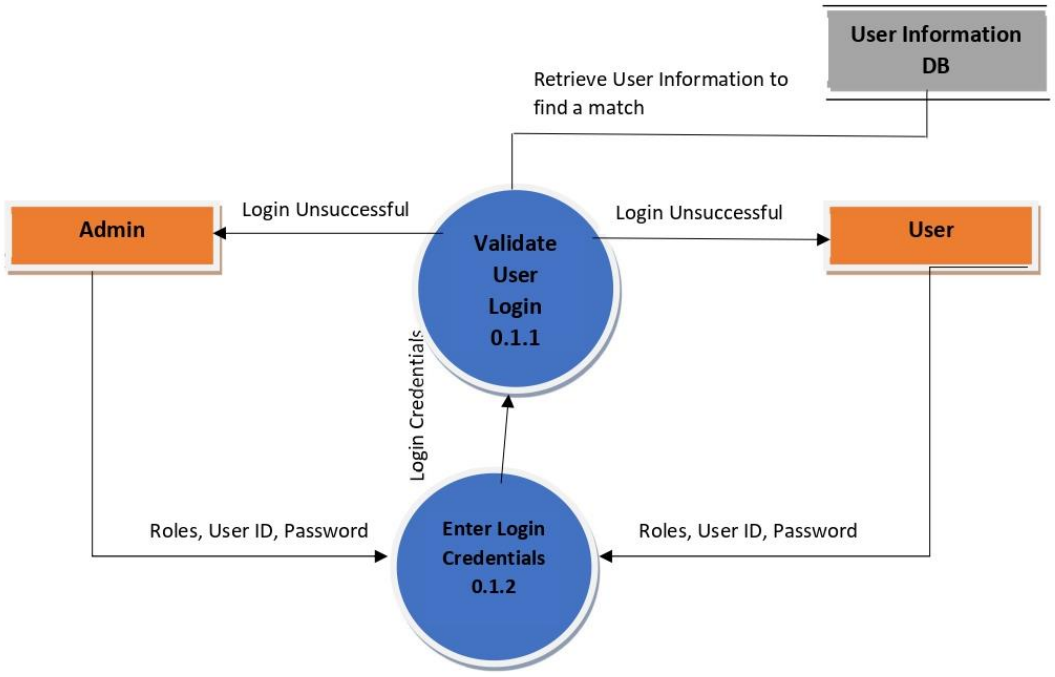


Fig 3.3 Level 2 DFD (Login)

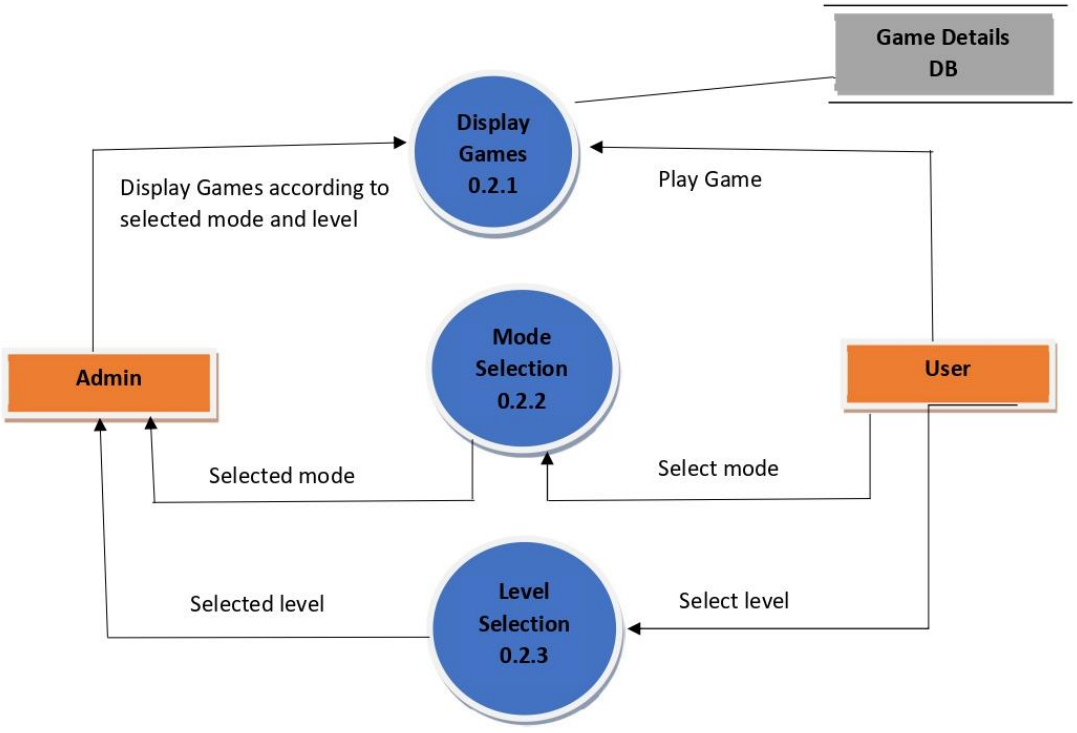


Fig 3.4 Level 2 DFD (Game Portal)

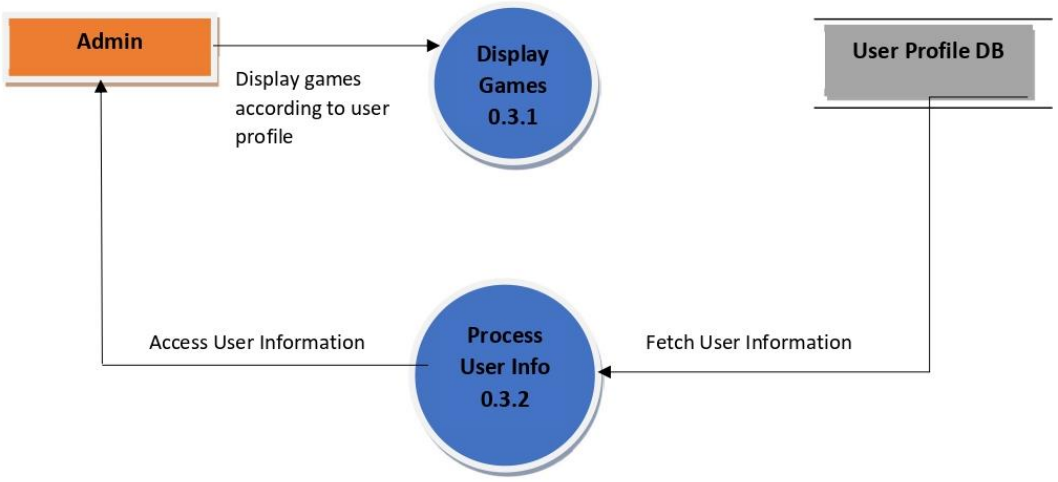


Fig 3.5 Level 2 DFD (Dashboard)

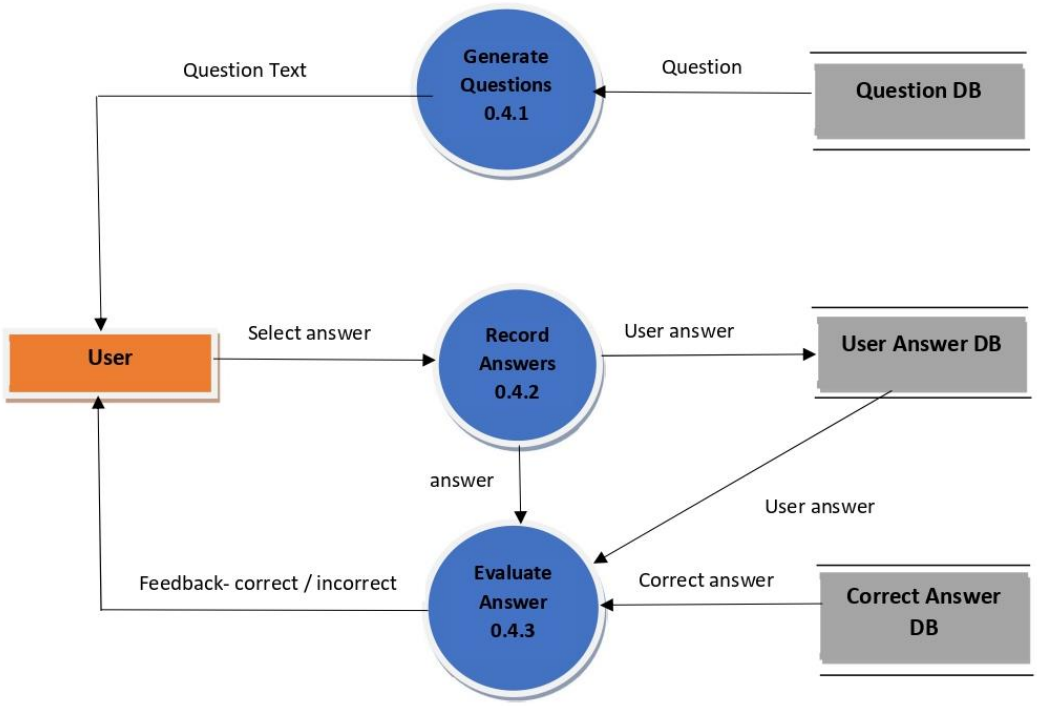


Fig 3.6 Level 2 DFD (Quiz Portal)

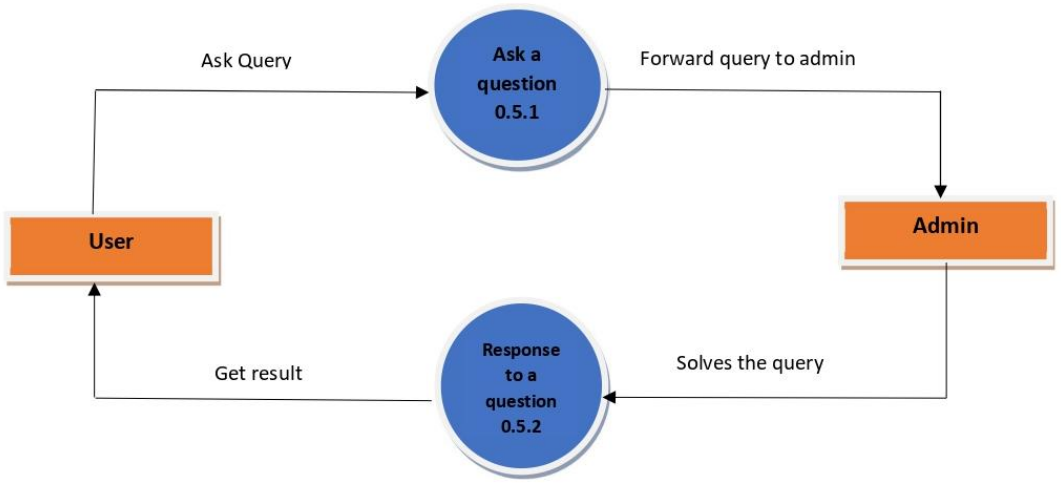


Fig 3.6 Level 2 DFD (ChatBot)

3.2 DATA DICTIONARY

Legal character: [a-z|A-Z]

Digit: [0-9]

Special character: [@|#|$|+|-|.]

1	User Name	{Legal character + Digit + Special Character} *
2	Password	{Legal character + Digit + Special Character} *
3	Name	{Legal character} *
4	Country	{Legal character} *
5	Role	{Legal character} *

Table 3.1 Data Dictionary

3.3 USE CASES

3.3.1 USE CASE DIAGRAM

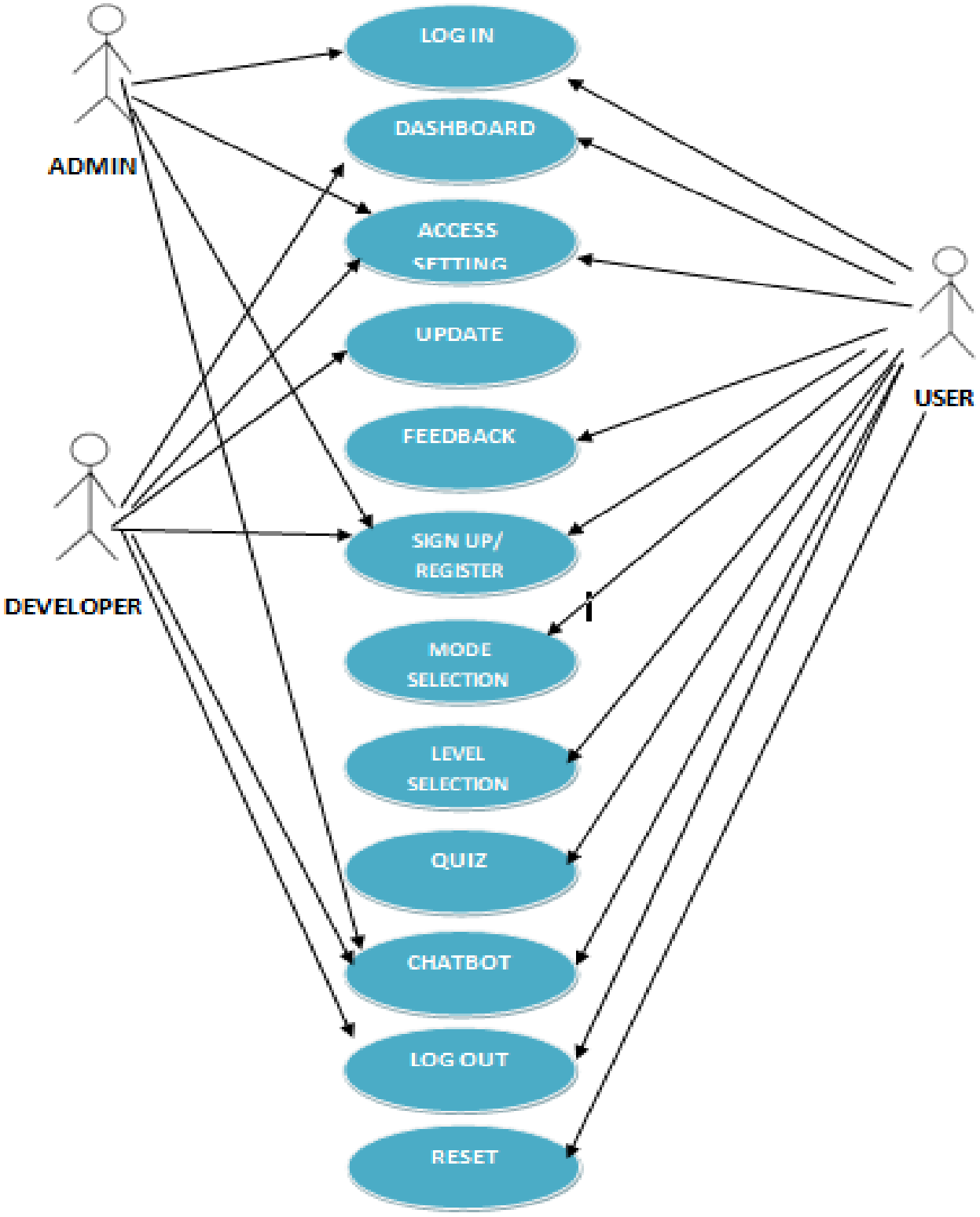


Fig 3.7 USE CASE DIAGRAM

3.3.2 USE CASE BRIEF AND DESCRIPTION

3.3.2.1 USE CASE (BRIEF)

- **LOGIN:**
  Login is a way to enter through user id and password. It also includes forgot and reset password functionality.

- **SIGN UP / REGISTER:**
  Register provides new user to first register in the software via mail or google account.

- **DASHBOARD:**
  It is a type of GUI which provides at a glance the view of all the games and performance of the user.

- **ACCESS SETTINGS:**
  Settings include sound control features and dark/light mode.

- **QUIZ:**
  It allows user to play mind provoking quizzes.

- **CHATBOT:**
  It is used to simulate a conversation between user and bot for seeking help regarding games.

- **UPDATE:**
  Admin can update any game or add new game. Also manages databases.

- **MODE SELECTION:**
  User can choose from single and multiplayer mode using this feature.

- **LEVEL SELECTION:**
  User can select difficulty level of games by using this feature.

- **FEEDBACK:**
  Rating / Review / Ways of improving website.

- **LOGOUT:**
  User can no longer access its stored data.

- **RESET:**
  User can delete all the previous records and start over again.

3.3.2.2 USE CASE (DESCRIPTION)

1. LOGIN SYSTEM:

1.1. Introduction: Describes how user can log into the gaming software.

1.2. Actors: User/ Admin/ Developer

1.3. Pre-Conditions: User must be registered in the software.

1.4. Post-Conditions: If the use case is successful, the actor is logged into the system. If not, the system state is unchanged.

1.5. Basic Flow:

i. System requests that the actor enter his/her user ID and password.
ii. The actor enters his/her user ID & password.
iii. System validates user ID & password, and if found correct allows the actor to log into the system.

1.6. Alternate Flows:

i. Invalid user ID, the system displays ERROR. The actor can choose to either return to the beginning of the basic flow or cancel the login, at that point, the use case ends.
ii. Invalid password, the system displays ERROR. The actor can choose to either return to the beginning of the basic flow or can choose the forgot password option to reset the password and then actor gets verification mail on their email id.
iii. Not registered, the system displays ERROR. The actor can choose to either return to the beginning of the basic flow or cancel the login, at that point, the use case ends.

2. REGISTER SYSTEM:

2.1. Introduction: Allows the new user to first register in the software via mail or google account. Here users can create their profile by specifying the unique user ID, email ID and password.

2.2. Actors: User/ Admin/ Developer

2.3. Pre-Conditions: None.

2.4. Post-Conditions: If the use case is successful, the actor can successfully login anytime to play games and save their progress.

2.5. Basic Flow:

i. System needs the actor to enter his/her user ID, email ID and password. The System verifies if the email ID is correct or not. After the verification is successful, the user ID and password is registered and now user can login.

2.6. Alternate Flows:

i. If the compulsory options left, the system displays ERROR. The actor then has to again fill that particular option.

ii. Invalid email ID, the system displays ERROR. The actor then has to again fill it again.

3. DASHBOARD:

3.1. Introduction: It is a type of GUI which provides at a glance the view of all the games and performance of the user.

3.2. Actors: User/ Admin/ Developer

3.3. Pre-Conditions: The user should be logged in.

3.4. Post-Conditions: If the user is logged in, the dashboard will display all the data about games and also user performance data.

3.5. Basic Flow:

i. The user is already logged in, and clicks on the dashboard all the brief data is displayed on screen for user to view and navigate through.

3.6. Alternate Flows:

i. If the user is not logged in, the display will show error and ask user to first login.

4. QUIZ:

4.1 Introduction: It allows user to play mind provoking quiz.

4.2. Actors: User/ Admin/ Developer

4.3. Pre-Conditions: User must be registered in the software.

4.4. Post-Conditions: If the use case is successful, the actor can play quiz. If not, the system state is unchanged.

4.5. Basic Flow:

i. The user will be asked questions.

ii. Then user enters his/her answers to the questions.

iii. System shows the result of each question whether it is correct or wrong with the updated score.

4.6. Alternate Flows:

i. Invalid input data, the system displays ERROR. The actor can choose to either return to the beginning of the basic flow or cancel the quiz, at that point, the use case ends.

ii. Problem in submitting quiz, the system displays ERROR. The actor can choose to either return to the beginning of the basic flow or can choose the different browser option to reset the quiz and then actor gets a chance to attempt the quiz again.

iii. Not registered, the system displays ERROR. The actor can choose to either return to the beginning of the basic flow or cancel the login, at that point, the use case ends.

5. CHATBOT:

5.1. Introduction: It is used to stimulate a conversation between user and bot for seeking help regarding games.

5.2. Actors: User/ Admin/ Developer

5.3. Pre-Conditions: None.

5.4. Post-Conditions: If the use case is successful, the actor can successfully ask questions anytime regarding games to play and save their progress.

5.5. Basic Flow:

i. System needs the actor to ask their queries regarding games. The System guides its users through a customer support journey. If the problem is not listed in concern, then the developer solves the customers queries and solve their problems regarding games.

5.6. Alternate Flows:

i. If there are limited options, system not able to answer multi-part questions or questions that require decisions. The actor then has to again ask for that particular option.

ii. Invalid concern or queries asked by user, the system not able to answer it. The actor then has to ask it again.

6. SETTINGS:

6.1. Introduction: It is a type of GUI which provides at a glance the view of the environment where the action takes place in a game.

6.2. Actors: User/ Admin/ Developer

6.3. Pre-Conditions: The user should be logged in.

6.4. Post-Conditions: If the user is logged in, the settings will display all the changes about games and also user performance data.

6.5. Basic Flow:

i. The user is already logged in, and clicks on the setting, user can change the mode of a game, enables to change in dark/light mode and enables the user to use many features to play game in more appealing manner.

6.6. Alternate Flows:

i. If the user is not logged in, the display will show error and ask user to first login.

7. UPDATE:

7.1. Introduction: Allows the developer to update the settings in the software via google account.

7.2. Actors: Developer

7.3. Pre-Conditions: None.

7.4. Post-Conditions: If the use case is successful, the actor can successfully change the settings of the games in the software.

7.5. Basic Flow:

i. System needs the actor to enter his/her user ID, email ID and password. The System verifies if the email ID is correct or not. After the verification is successful, the developer can access and change the settings.

7.6. Alternate Flows:

i. If the compulsory options left, the system displays ERROR. The actor then has to again fill that particular option.

ii. Invalid email ID, the system displays ERROR. The actor then has to again fill it again.

8. MODE SELECTION:

8.1. Introduction: Allows the user to select the mode of players whether it be single or multiplayer in the games via mail or google account.

8.2. Actors: User

8.3. Pre-Conditions: User must be registered in the software.

8.4. Post-Conditions: If the use case is successful, the mode is saved into the system. If not, the system state is default.

8.5. Basic Flow:

i. The user will be given option.
ii. Then user enters his/her choice.
iii. System saves the mode into the system.

8.6 Alternate Flows:

i. Invalid input data, the system displays ERROR. The actor can choose to either return to the beginning of the basic flow or cancel the selection if chose wrong option.
ii. Not registered, the system displays ERROR. The actor can choose to either return to the beginning of the basic flow or cancel the login, at that point, the use case ends.

9. LEVEL SELECTION:

9.1. Introduction: Allows the user to select the mode of games whether it be easy or medium or hard via mail or google account.

9.2. Actors: User

9.3. Pre-Conditions: User must be registered in the software.

9.4. Post-Conditions: If the use case is successful, the level of the game is saved into the system. If not, the system state is default.

9.5. Basic Flow:

i. The user will be given option.
ii. Then user enters his/her choice.
iii. System saves the level into the system.

9.6. Alternate Flows:

i. Invalid input data, the system displays ERROR. The actor can choose to either return to the beginning of the basic flow or cancel the selection if chose wrong option.

ii. Not registered, the system displays ERROR. The actor can choose to either return to the beginning of the basic flow or cancel the login, at that point, the use case ends.

10. FEEDBACK:

10.1. Introduction: This functionality is used for taking feedback from the users to get reviews about our services provided to them and to get more ideas to make our software more approachable and efficient. It covers these points – Rating, Review, Ways of improving the website

10.2. Actors: Users/Volunteers

10.3. Pre-Conditions: Actors must be registered and logged in the software.

10.4. Post-Conditions: Review successfully posted and stored in database of admin.

10.5. Basic Flow:

i. Actor can visit feedback section to write reviews and rating for the gaming website by filling the form feedback form available on the website.

10.6. Alternate Flows: None.

11. LOGOUT:

11.1. Introduction: Describes how user can log out of the system.

11.2. Actors: User/ Developer

11.3. Pre-Conditions: User must be logged in into the software.

11.4. Post-Conditions: If the use case is successful, the actor is logged out of the system. If not, the system state is unchanged.

11.5. Basic Flow:

i. User logs out of the system and can no longer access its stored data.

11.6. Alternate Flows: None.

12. RESET:

12.1. Introduction: Allows the user to delete all the previous records and start over again.

12.2. Actors: User

12.3. Pre-Conditions: User must be logged in into the system.

12.4. Post-Conditions: If the use case is successful, all data of the user is deleted and he/she can start anew.

12.5. Basic Flow:

i. User selects the reset option. Then the system asks for user confirmation. After the confirmation is successful, all data and progress of the user from the database is deleted.

12.6. Alternate Flows:

i. If the confirmation is unsuccessful, the system remains unchanged.

3.4 SEQUENCE DIAGRAMS

1. REGISTER

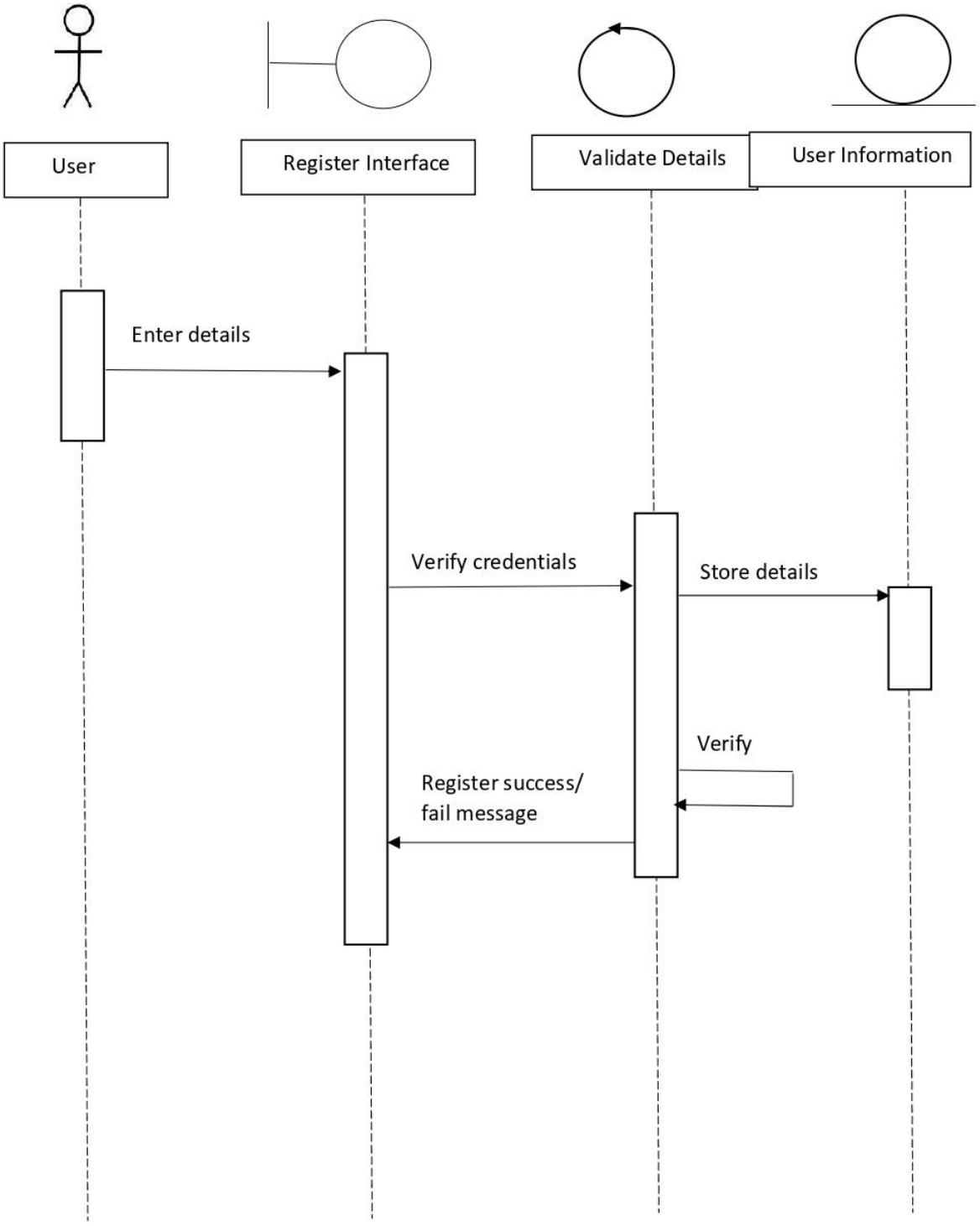


Fig 3.8 REGISTER

2. LOGIN

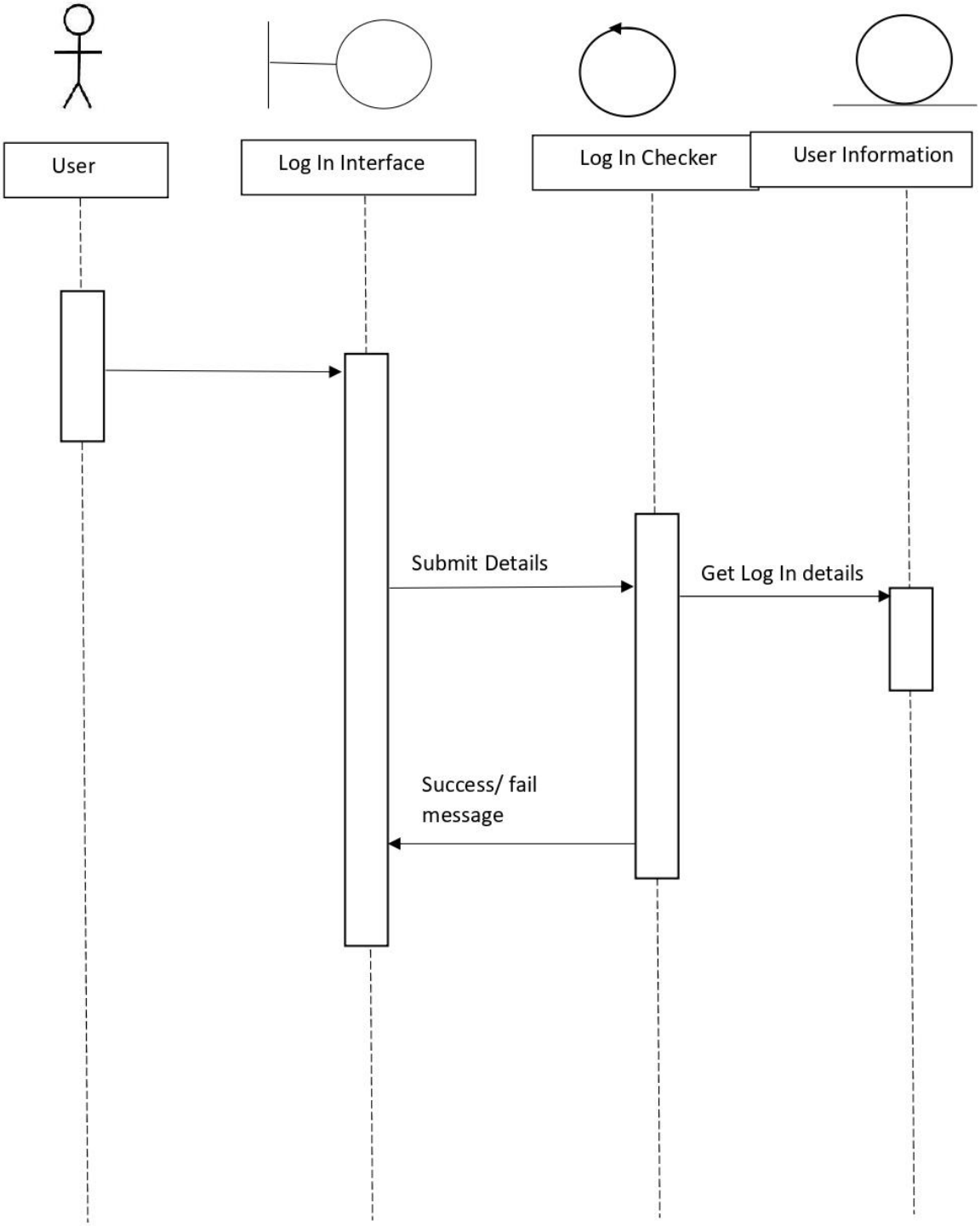


Fig 3.9 LOGIN

3. GAME

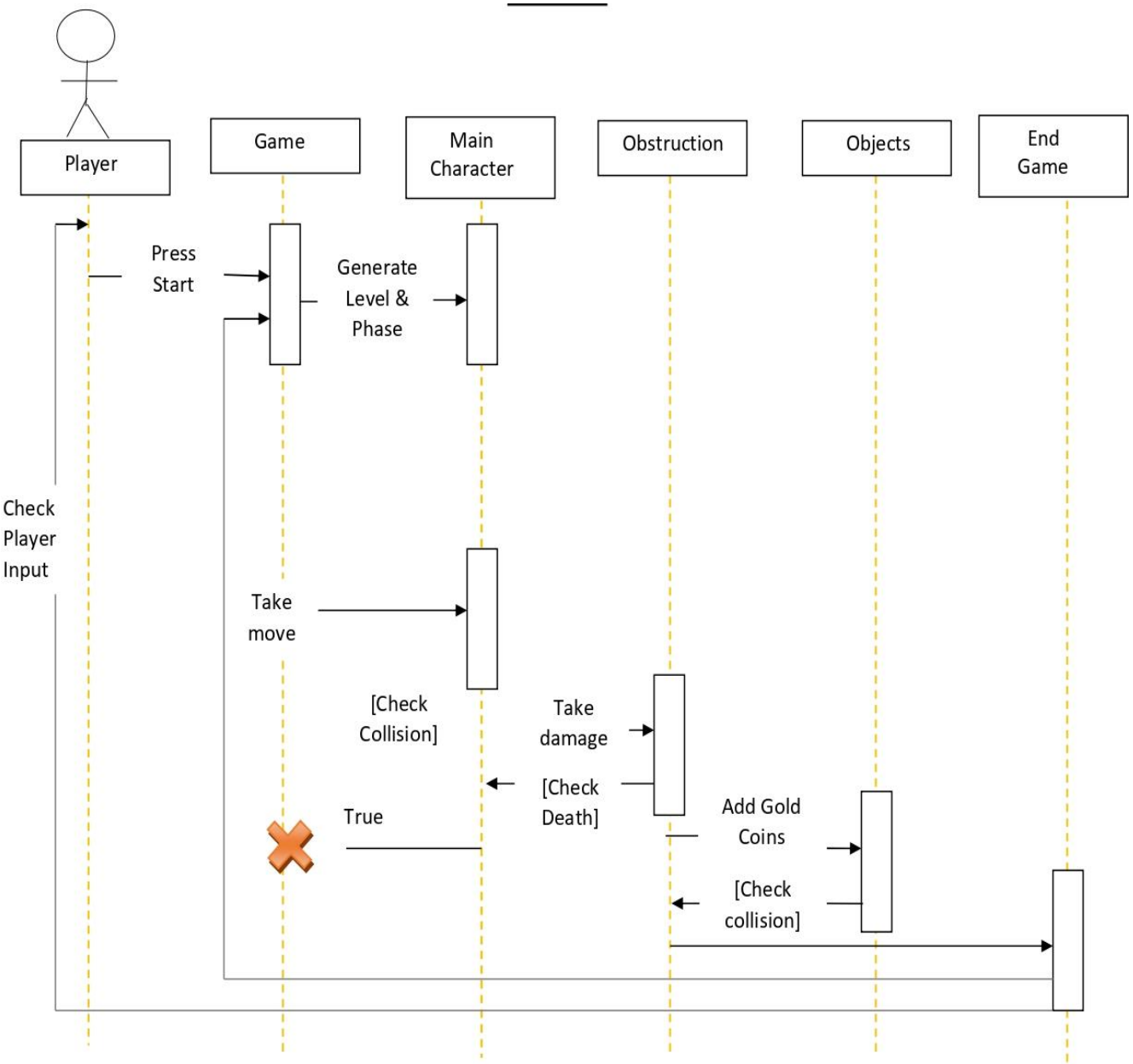


Fig 3.10 GAME

4. QUIZ

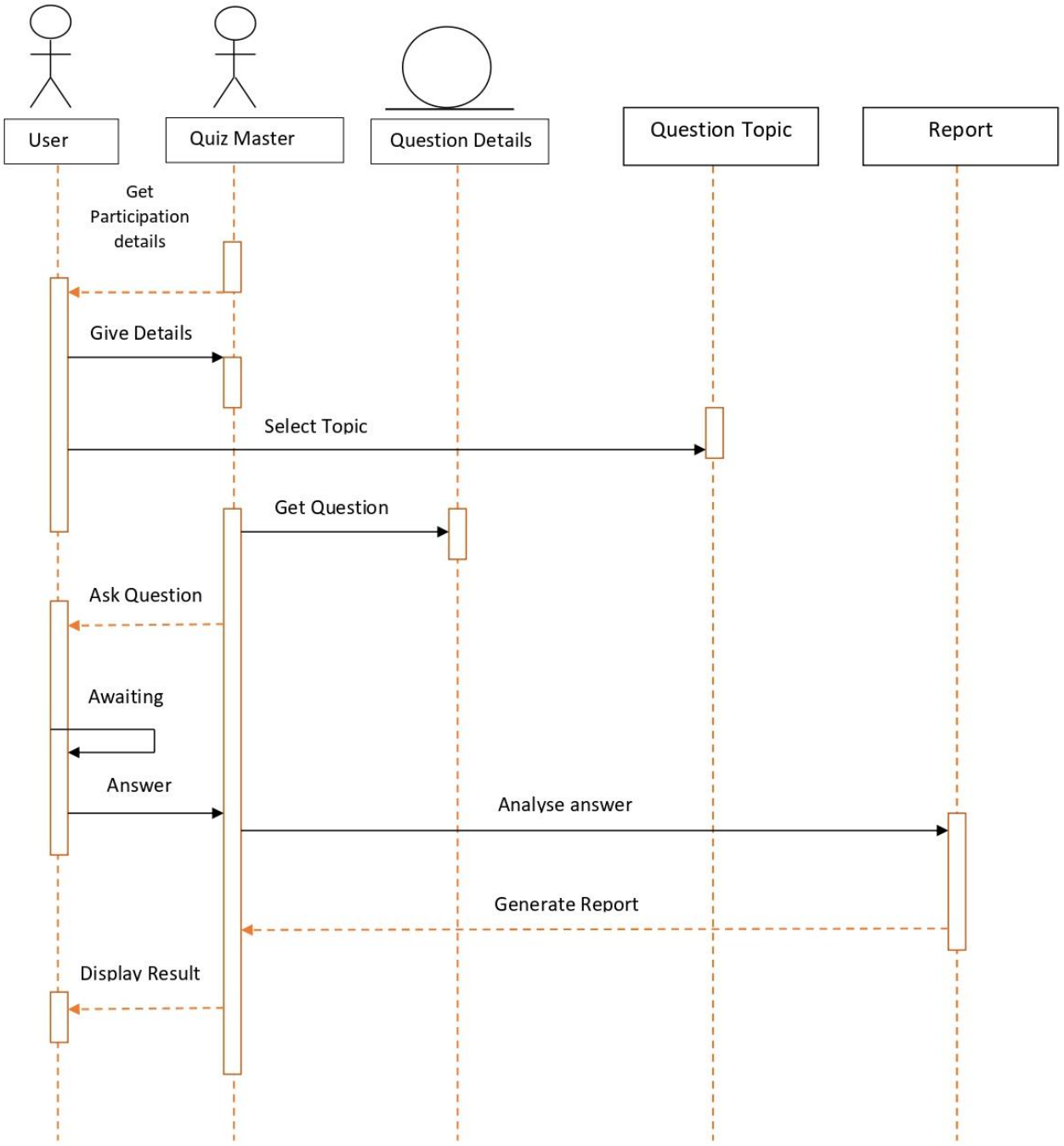


Fig 3.11 QUIZ

5. FEEDBACK

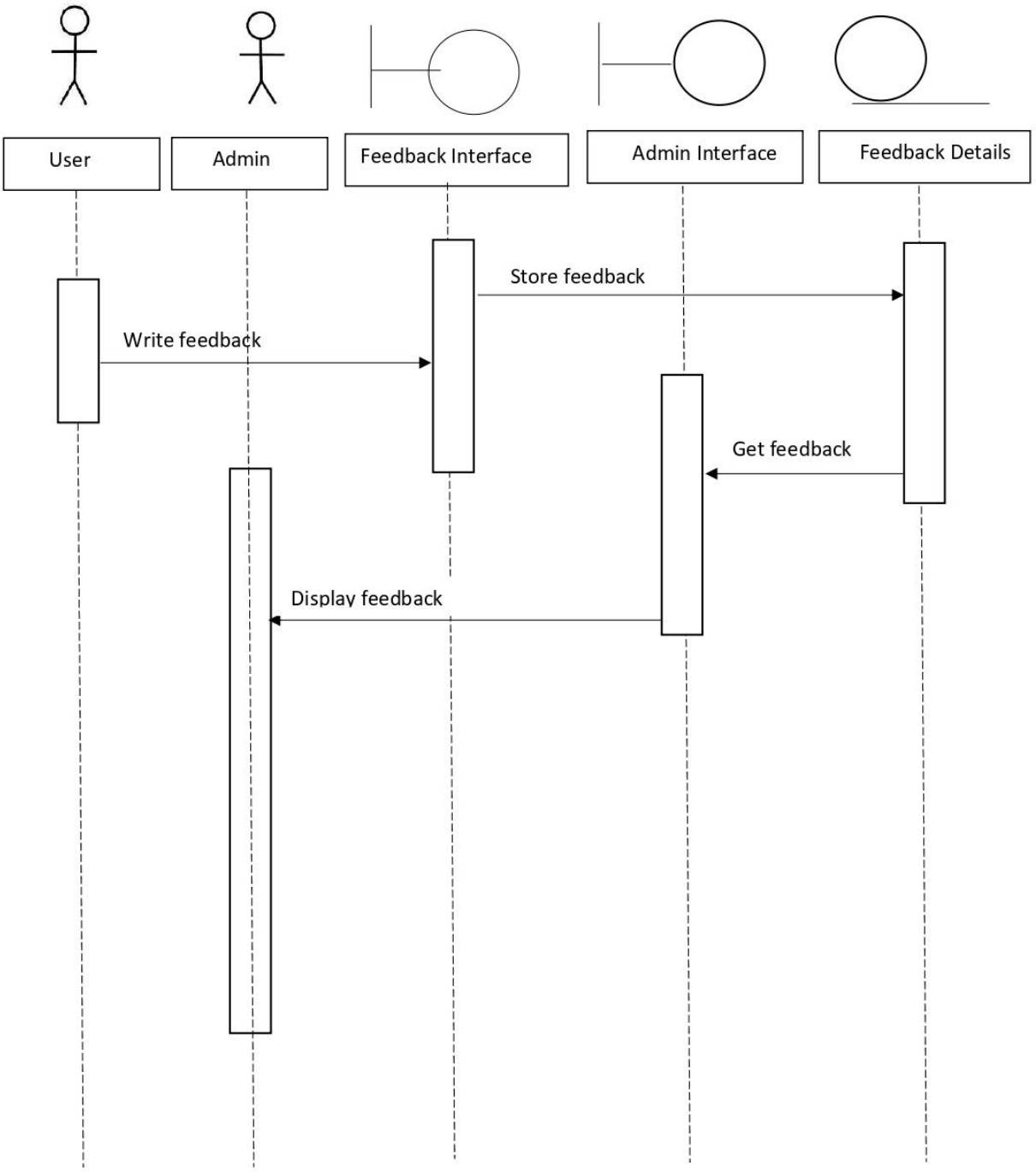


Fig 3.12 FEEDBACK

4. SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

1. INTRODUCTION

Software Requirements Specification (SRS) is used to describe the behavior of the software system. It lays out the functional and non-functional requirements of the software system.

1.1 CATEGORY

Web Application.

1.2 PURPOSE

The purpose of this document is to provide a detailed description of the requirements of "KIDS" software. The primary aim of the project is to develop a system that provides a platform where users can find and play games to pass time or just to have fun. There are multiple games available online to play on web and mobile. But it become troublesome when you have to use multiple websites or apps to play different games.

Therefore, to save the time and storage space our gaming website KIDS provide multiple games available all together at one place and anyone can play them for free.

1.3 SCOPE

The "KIDS" software is an online gaming web application which provides the users an ease in playing multiple games at single platform.

The users will register and login into the application to play games. The users can choose their games based on mode (no of players) and difficulty levels. Their progress and score will be recorded into the system and displayed on their user profile. The history is saved so the user can come back and resume from where they left. They can also enjoy the feature of playing mind provoking quizzes and challenge their knowledge. They can reset the data stored at any time.

1.4DEFINITION, ACRONYMS AND ABBREVIATIONS

1.4.1 DEFINITIONS:

User: The person/persons who interact directly with the work product. The players of the games.

Developer/Admin: The person who manages the database, deletes and updates the content.

1.4.2 ACRONYMS:

SRS: Software Requirements Specification

GUI: Graphical User Interface

1.4.3 ABBREVIATION

i.e.: that is

etc: etcetera

1.5 OVERVIEW

The rest of the document deals with all the main functionalities and features of the software in detail and describes various system requirements, interfaces.

2. THE OVERALL DESCRIPTION:

This section will give an overview of the whole system. The system will be explained in the context to show how the system interacts with other systems and introduce the basic functionality of the system. It will also describe what type of stakeholders will use the system and what functionality is available for them. Constraints also presented in the end of this section.

2.1 PRODUCT PERSPECTIVE:

The "KIDS" is an independent system. The system involves 2 end users

1. Users/Players
2. Admin.

The database is also provided to get easy and fast access to the data.

2.1.1 SYSTEM INTERFACES:

The application runs over the latest version of Chrome or Firefox browser on Windows, Linux and Mac.

2.1.2 USER INTERFACES:

Graphical User Interface (GUI) is provided to the users of the software. There is no command line interface. User interface is implemented using a HTML, CSS.

2.1.3 HARDWARE INTERFACES AND MEMORY CONSTRAINTS:

- Cores: 4
- RAM: 8GB
- Require Internet Connection as the application will run over the Internet.
- Screen resolution of at least 1080 x 2030 required for proper and complete viewing of screens. Higher resolution would not be a problem.

2.1.4 SOFTWARE INTERFACES:

- Supported by any web browsers
- The software is on server so it requires the support of scripting languages like PHP, Java and JavaScript.
- Also, to handle a large number of users and user's related databases the software requires the help of database systems like MySQL.
- The version of system's software shall be compatible to the application to make it accessible.

2.1.5 COMMUNICATION INTERFACE

- The system shall use the HTTP protocol for communication over the internet and TCP/IP protocol suite shall be used for intranet communication.

2.2 PRODUCT FUNCTIONS

The user must enter a valid username and password to login into the application. If the user is new to the application (i.e., first time users), then the user has to create a profile or register in order to store their data. If users forget the password they will be directed to a page where they can set a new password and then enter the system. Without valid username and password, they can't use the services of the system and an error message will be displayed if the user enters the wrong credentials or if they are not registered.

2.3 ACTORS CHARACTERISTICS:

There are 3 types of actors in this software:

- User/Player
- Developer
- Admin

Users can access the website and play the games they like and also record their progress and scores.

Admin and Developers can manage, delete, update, review the settings. Manage the database.

2.4 CONSTRAINTS:

- The interface will be English only.
- This supports only web
- Internet is essential

2.5 ASSUMPTIONS AND DEPENDENCIES:

The product requires back-end database server for storing the username and password and valid email-id for different types of the users of the system.

2.5.1 ASSUMPTION:

- Users must have basic knowledge of computers functionality.
- Users must have basic knowledge of English.

3. SPECIFIC REQUIREMENTS

This section contains all the software requirements at a level of detail sufficient to enable users to play games or system to satisfy the user's requirements, and testers to test that the system satisfies those requirements.

3.1 EXTERNAL INTERFACES:

There are many types of interfaces which are supported by the system.

- User interface
- Hardware interface
- Software interface
- Communication interface

3.2 FUNCTIONS

- Login: - This functionality allows the registered user to login into the software using their username and password.

- Register: - This functionality allows the new user to first register in the software via email. Here users can create their profile by specifying the unique username and password.

- Reset/Forgot Password: - This functionality is designed in order to reset the password in case the user forgets his/her password.

- Feedback: - This functionality is used for taking feedback from the users to get reviews about our services provided to them and to get more ideas to make our software more approachable and efficient.

    - Rating
    - Review
    - Ways of improving the website

- Settings: It allows user to control sound features and dark/light mode.

- Quiz: It allows user to play mind provoking quizzes.

- Chatbot: It is used to simulate a conversation between user and bot for seeking help regarding games.

- Update: Admin can update any game or add new game. Also manages databases.

- Mode Selection: User can choose from single and multiplayer mode using this feature.

- Level Selection: User can select difficulty level of games by using this feature.

- Logout: User can no longer access its stored data.

- Reset: User can delete all the previous records and start over again.

3.3 PERFORMANCE REQUIREMENTS

- Server will be working the whole 24 x 7 times.
- No limitation on the number of users entering in the system.

3.4 LOGICAL DATABASE REQUIREMENTS

The database used by the software system is only one and includes the following tables -

- User Information Table: stores all the information entered by the user during registration.

- User Profile Table: stores the information displayed to the user regarding their progress of games, points earned, etc.

- Game Details Table: stores the info about each and every game played by the user.

- Quiz History Table: stores the data of the quizzes attempted by the user.

- Feedback Table: stores the feedback information given by the user.

3.5DESIGN CONSTRAINTS

- Software Language Used – HTML, CSS, JavaScript, PHP

- Database Design - In our database design, we used MySQL.

3.5.1 STANDARDS COMPLIANCE

- Report format - All the reports produced for this project are in compliance with the standard templates in accordance with the standard guidelines and policy.

- Naming Conventions - All the documents are named using the standard naming conventions

3.6 SOFTWARE SYSTEM ATTRIBUTES (NON-FUNCTIONAL)

3.6.1 Security: The application will be secured by a username and password.

3.6.2 Reliability: The application is reliable as it has many rare chances of failures and if any failure occurs it can be repaired in 2-3 days.

3.6.3 Portability: The application is portable.

3.6.4 Availability: This application server is working 24x7.

3.6.5 Maintainability: Any updates or detect fixes shall be made on the server-side computers with updates done by Developer.

5. PROJECT PLANNING

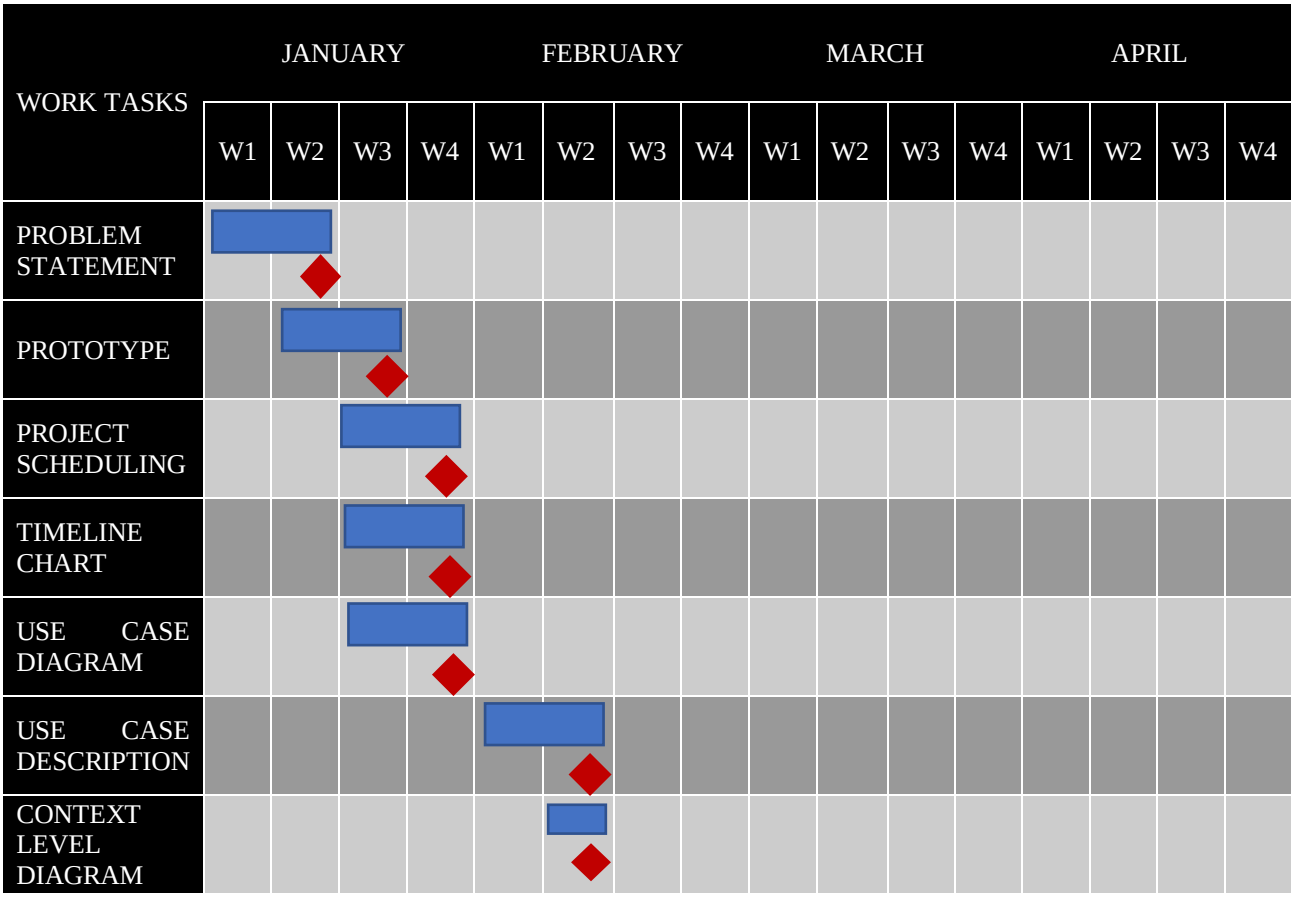
5.1 PROJECT SCHEDULING

WORK TASKS	PLANNED START	ACTUAL START	PLANNED COMPLETE	ACTUAL COMPLETE	ASSIGNED PERSON(s)	EFFORT ALLOCATED
PROBLEM STATEMENT	Jan w1	Jan w1	Jan w2	Jan w2	Khushi Sharma	1 person per week
PROTOTYPE	Jan w2	Jan w2	Jan w3	Jan w3	Saloni Khanna, Anjali Thakur, Khushi Sharma	3 persons per week
PROJECT SCHEDULING	Jan w3	Jan w3	Jan w4	Jan w4	Khushi Sharma	1 person per week
TIMELINE CHART	Jan w3	Jan w3	Jan w4	Jan w4	Khushi Sharma	1 person per week
USE CASE DIAGRAM	Jan w3	Jan w3	Jan w4	Jan w4	Saloni Khanna	1 person per week
USE CASE DESCRIPTION	Feb w1	Feb w1	Feb w2	Feb w2	Anjali Thakur, Diksha Sharma, Ishika Bhatt	3 persons per week
CONTEXT LEVEL DIAGRAM	Feb w2	Feb w2	Feb w3	Feb w3	Saloni Khanna	1 person per week
DFD LEVEL-1	Feb w2	Feb w2	Feb w3	Feb w4	Saloni Khanna	1 person per week
DFD LEVEL-2	Feb w3	Feb w4	Feb w4	Feb w4	Khushi Sharma	1 person per week
DATA DESIGN	Feb w4	Feb w4	Mar w1	Feb w4	Diksha Sharma, Ishika Bhatt, Anjali Thakur	3 persons per week
DATA DICTIONARY	Feb w4	Feb w4	Mar w1	Feb w4	Khushi Sharma	1 person per week
SRS	Mar w1	Mar w1	Mar w2	Mar w2	Anjali Thakur	1 person per week
SEQUENCE DIAGRAM	Mar w2	Mar w2	Mar w3	Mar w3	Saloni Khanna	1 person per week

DESIGN	Mar w2	Mar w2	Mar w4	Mar w4	Khushi Sharma, Anjali Thakur, Ishika Bhatt, Saloni Khanna	4 persons per week
PROJECT METRICS	Apr w1	Apr w1	Apr w2	Apr w2	Anjali Thakur	1 person per week
EFFORT ESTIMATION (COCOMO)	Apr w2	Apr w2	Apr w3	Apr w3	Anjali Thakur	2 persons per week
RISK ANALYSIS	Apr w3	Apr w3	Apr w4	Apr w4	Diksha Sharma, Ishika Bhatt	2 persons per week
TESTING	Apr w3	Apr w3	Apr w4	Apr w4	Diksha Sharma	1 person per week

TABLE 5.1 PROJECT SCHEDULLING

5.2 PROJECT TIMELINE CHART



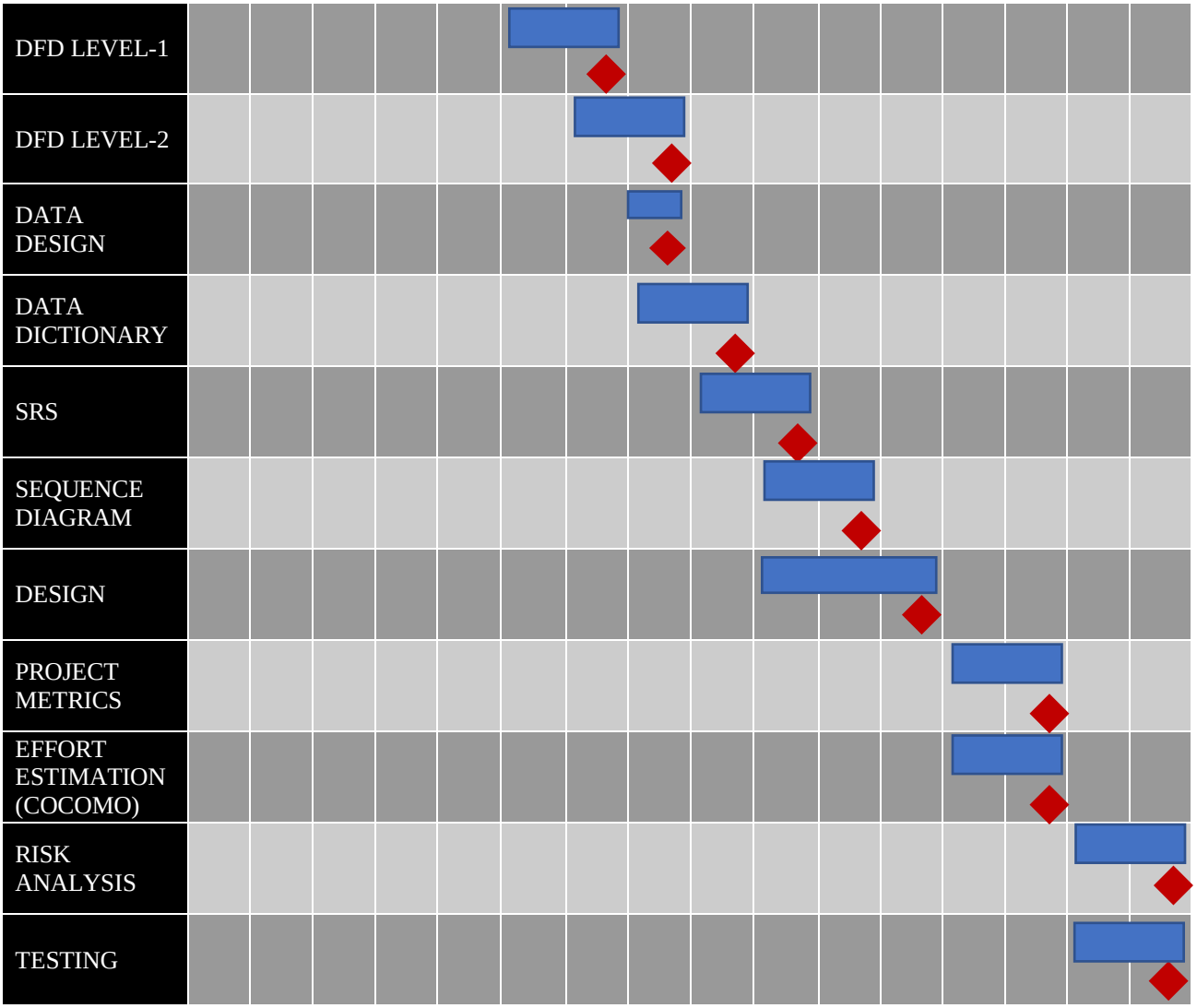


TABLE 5.2 PROJECT TIMELINE CHART

5.3 EFFORT ESTIMATION & FP- BASED COMPUTING

The function point (FP) metric can be used effectively as a means for measuring the functionality delivered by a system. Using historical data, the FP metric can be used to:

1. Estimate the cost or effort required to design, code, and test the software.
2. Predict the number of errors that will be encountered during testing.
3. Forecast the number of components and/or the number of projected source lines in the implemented system.

Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and qualitative assessments of software complexity.

Information domain values are defined as:

1. **Number of external inputs (EIs):** Each external input originates from a user or is transmitted from another application and provides distinct application-oriented data or control information. Inputs are often used to update internal logical files (ILFs).

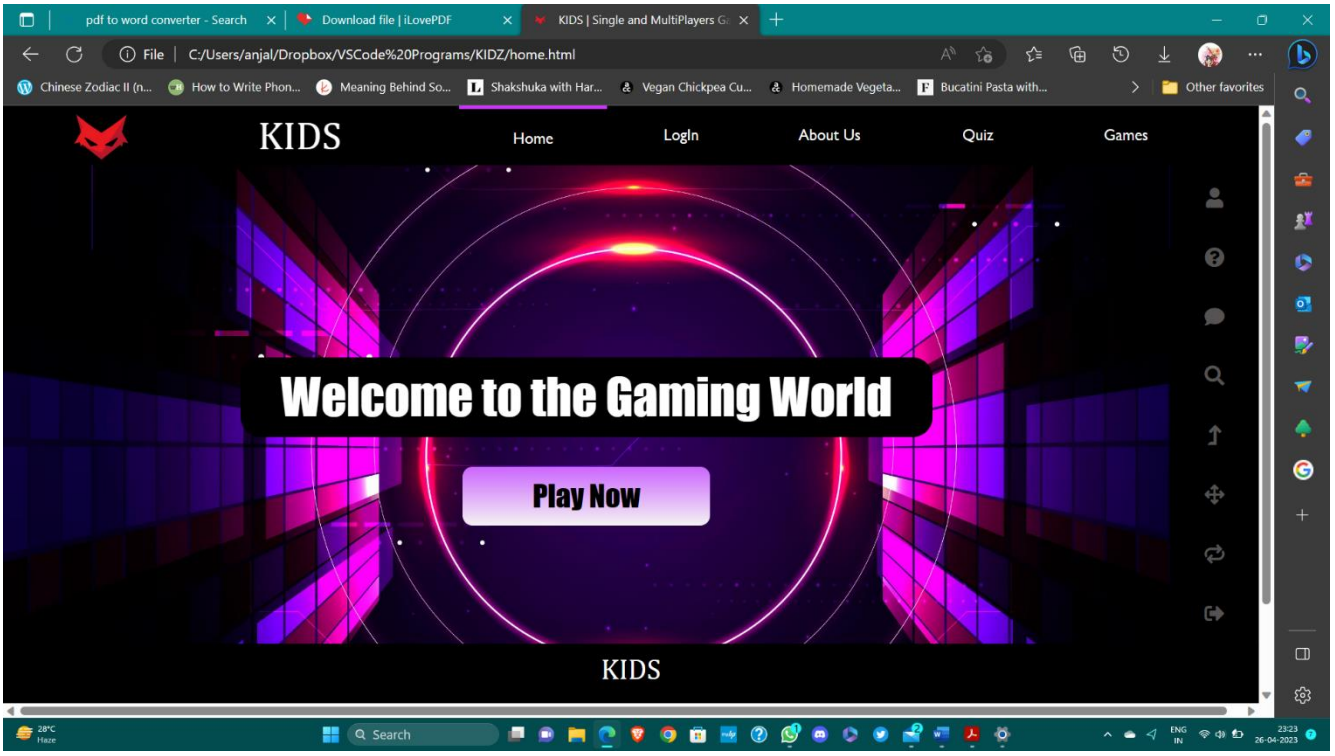
2. **Number of external outputs (EOs):** Each external output is derived data within the application that provides information to the user. In this context external output refers to reports, screens, error messages, and the like.

3. **Number of external inquiries (EQs):** An external inquiry is defined as an online input that results in the generation of some immediate software response in the form of an online output (often retrieved from an Internal Logical File).

4. **Number of internal logical files (ILFs):** Each internal logical file is a logical grouping of data that resides within the application's boundary and is maintained via external inputs

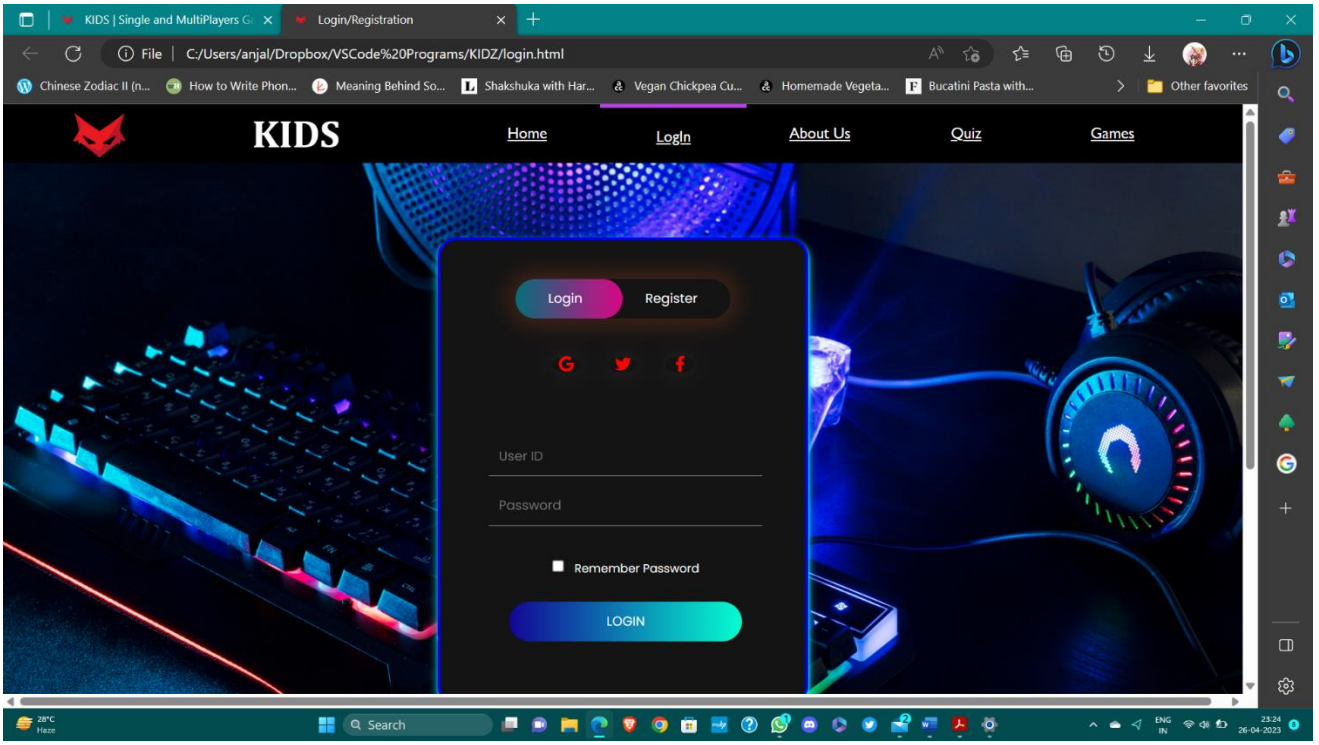
5. **Number of external interface files (EIFs):** Each external interface file is a logical grouping of data that resides external to the application but provides data that may be of use to the application.

HOMEPAGE



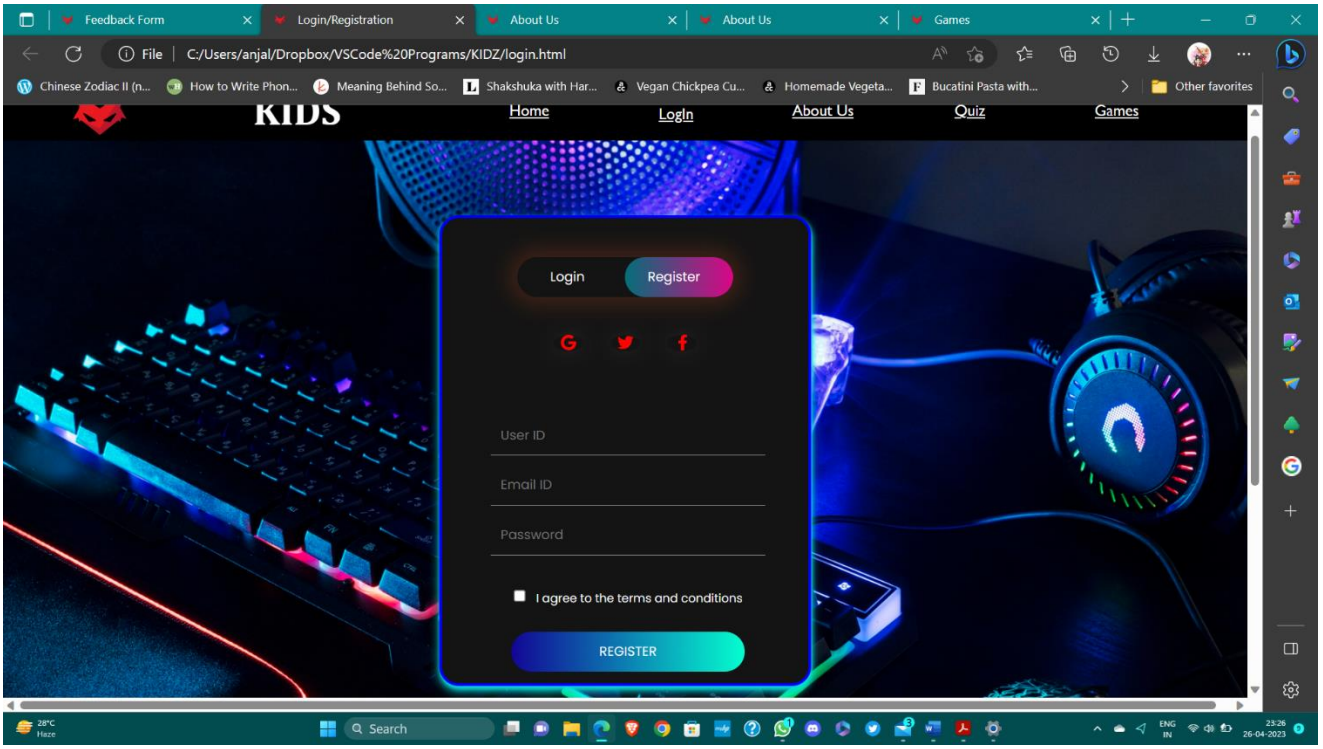
Number of External Inputs: 0
Number of External Outputs: 1
Number of External Inquiries: 0
Number of Internal Logical File: 0
Number of External Interfaces: 0

LOGIN



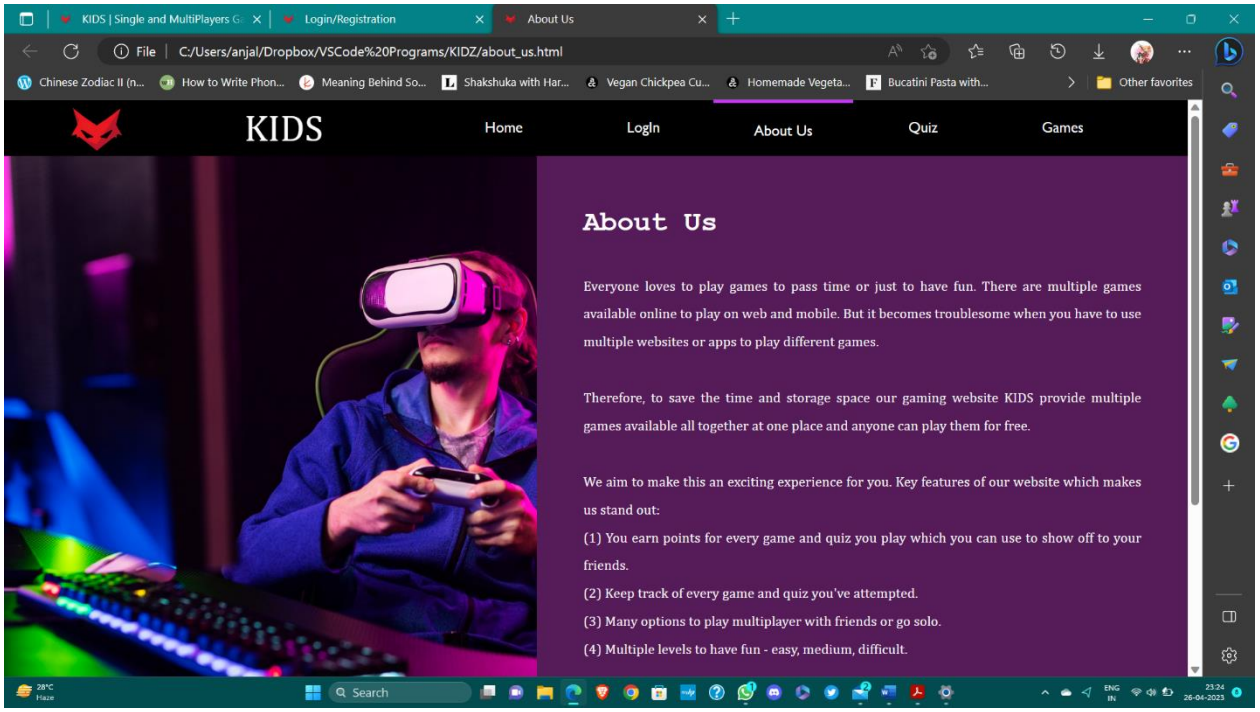
Number of External Inputs: 3
Number of External Outputs: 1
Number of External Inquiries: 0
Number of Internal Logical File: 0
Number of External Interfaces: 0

REGISTER



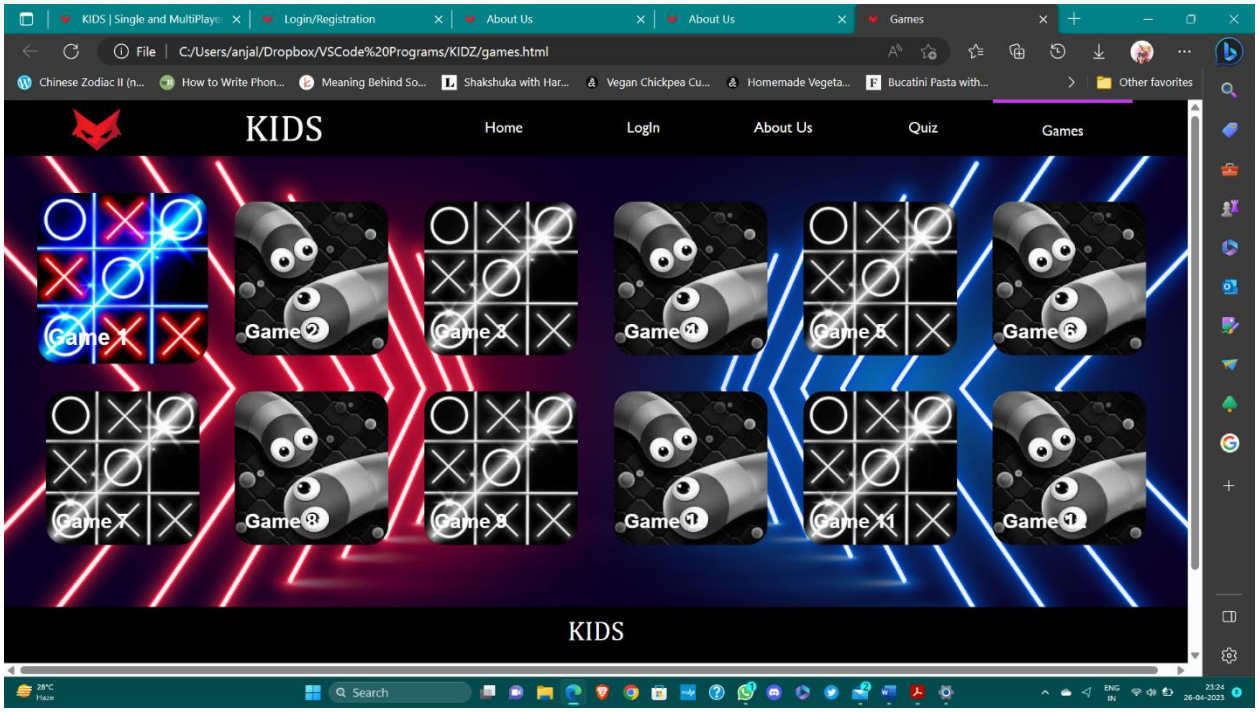
Number of External Inputs: 4
Number of External Outputs: 1
Number of External Inquiries: 0
Number of Internal Logical File: 1
Number of External Interfaces: 0

ABOUT US



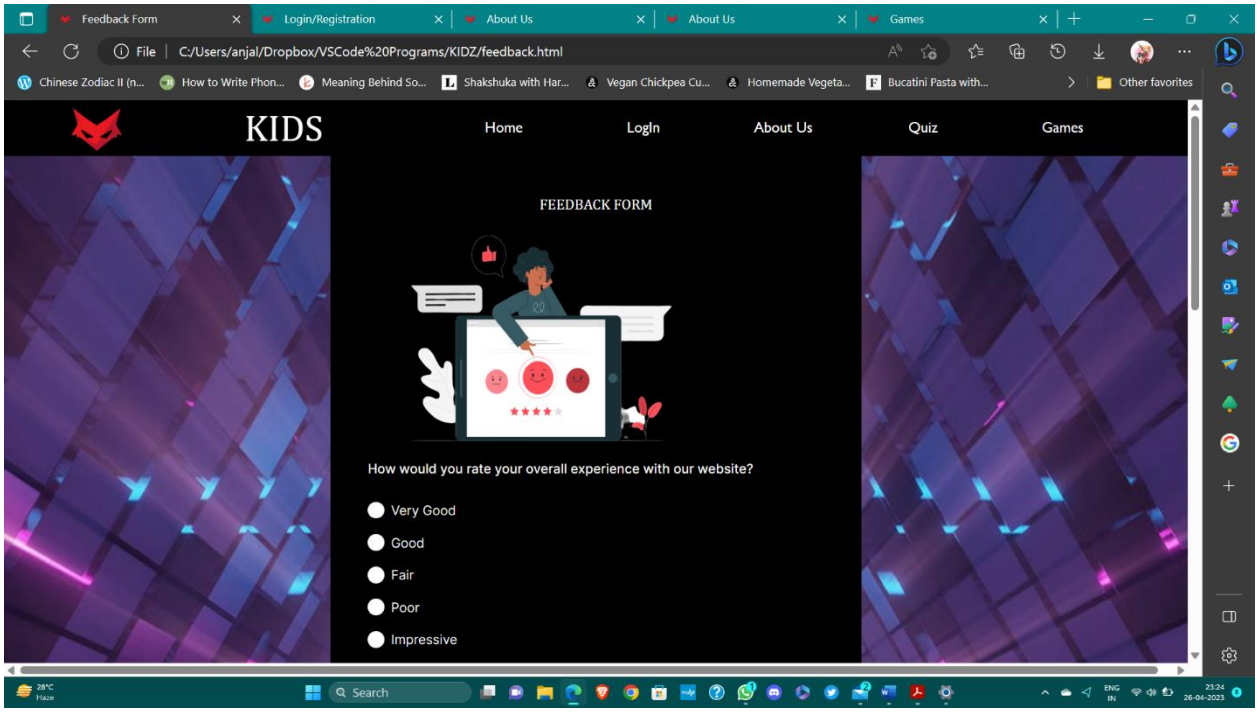
Number of External Inputs: 0
Number of External Outputs: 1
Number of External Inquiries: 0
Number of Internal Logical File: 0
Number of External Interfaces: 0

GAMES PAGE



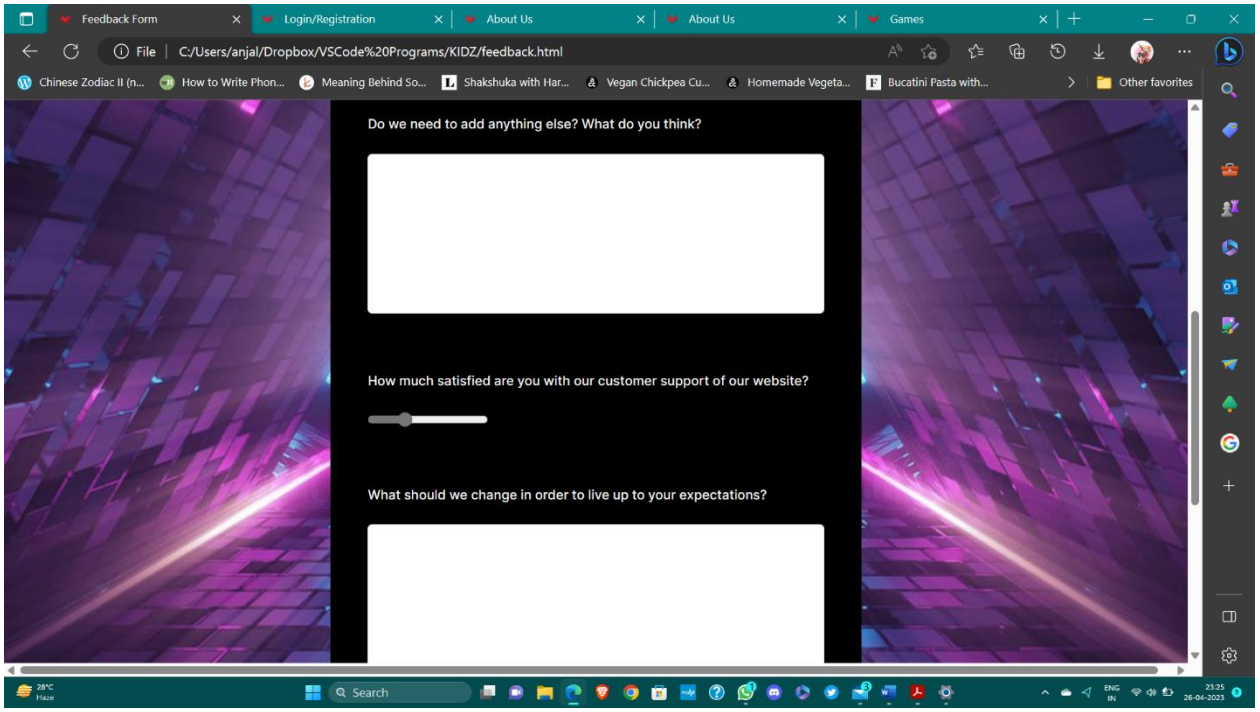
Number of External Inputs: 0
Number of External Outputs: 1
Number of External Inquiries: 0
Number of Internal Logical File: 0
Number of External Interfaces: 0

FEEDBACK FORM 1



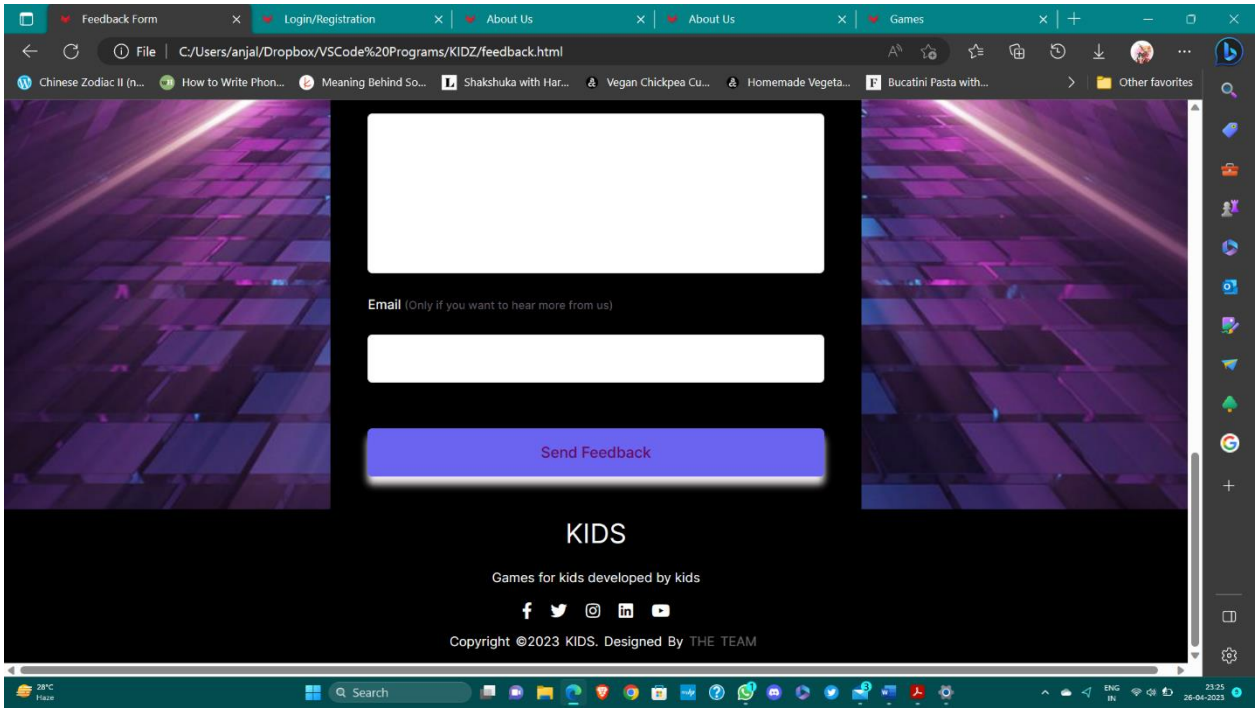
Number of External Inputs: 1
Number of External Outputs: 1
Number of External Inquiries: 0
Number of Internal Logical File: 1
Number of External Interfaces: 0

FEEDBACK FORM 2



Number of External Inputs: 3
Number of External Outputs: 0
Number of External Inquiries: 0
Number of Internal Logical File: 1
Number of External Interfaces: 0

FEEDBACK FORM 3



Number of External Inputs: 1
Number of External Outputs: 1
Number of External Inquiries: 0
Number of Internal Logical File: 1
Number of External Interfaces: 0

The Fi (i = 1 to 14) are value adjustment factors (VAF) based on responses to the following questions:

1. Does the system require reliable backup and recovery?
   3

2. Are specialized data communications required to transfer information to or from the application?
   3

3. Are there distributed processing functions?
   3

4. Is performance critical?
   3

5. Will the system run in an existing, heavily utilized operational environment?
   3

6. Does the system require online data entry?
   5

7. Does the online data entry require the input transaction to be built over multiple screens or operations?
   3

8. Are the ILFs updated online?
   5

9. Are the inputs, outputs, files, or inquiries complex?
   2

10. Is the internal processing complex?
    3

11. Is the code designed to be reusable?
    3

12. Are conversion and installation included in the design?
    3

13. Is the system designed for multiple installations in different organizations?
    1
14. Is the application designed to facilitate change and ease of use by the user?
    3

TOTAL = 43

Once these data have been collected, a complexity value is associated with each count. Organizations that use function point methods develop criteria for determining whether a particular entry is simple, average, or complex.

To compute function points (FP), the following relationship is used:

FP = count total × [0.65 + (0.01 × Σ(Fi))]

where count total is the sum of all FP entries obtained from the table given below.

Information Domain Value	**Count**		**Weighting Factor**				
			Simple	**Average**	**Complex**		
External Inputs	12	**x**	3	4	6	=	36
External Outputs	7	**x**	4	5	7	=	28
External Inquiries	0	**x**	3	4	6	=	0
Internal Logical Files	4	**x**	7	10	15	=	28
External Interface Files	0	**x**	5	7	10	=	0
Count Total						=	**92**

Function Point Table

In our project, we have the **weighting factor as simple** for all the five information domain values.

W(FP) = count total × [0.65 + (0.01 × ∑(Fi))]

= 92 × [0.65 + (0.01 × 43)]

= 92 x 1.08

W(FP) = 99.36 FP

Let

Average Productivity = X = 4 FP/pm
Labour rate = Y = Rs 8,000 person per month

So,

Cost per FP = Z
= Y/X
= Rs 8,000/4
Cost per FP = Rs 2,000 (approx.)

Total estimated project cost = W*Z
= Rs 1,98,720 (approx.)
Estimated effort = W/X
= 25 person-month (approx.)

5.4 COST ESTIMATION USING COCOMO II MODEL

The Constructive Cost Model (COCOMO II) is a more comprehensive estimation model. It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity. COCOMO II is actually a hierarchy of estimation models that address different "stages" of the software process.

1. **Stage-I:**
   It supports estimation of prototyping. For this it uses **Application Composition Estimation Model.** This model is used for the prototyping stage of application generator and system integration.

2. **Stage-II:**
   It supports estimation in the early design stage of the project, when we less know about it. For this it uses **Early Design Estimation Model.** This model is used in early design stage of application generators, infrastructure, and system integration.

3. **Stage-III:**
   It supports estimation in the post architecture stage of a project. For this it uses **Post Architecture Estimation Model.** This model is used after the completion of the detailed architecture of application generator, infrastructure, and system integration.

The COCOMO II models require sizing information. Three different sizing options are available as part of the model hierarchy:

1. Object points
2. Function points
3. Lines of source code.

The object point is an indirect software measure that is computed using counts of the number of –

i. screens (at the user interface)
ii. Reports
iii. Components likely to be required to build the application

Each object instance (e.g., a screen or report) is classified into one of three complexity levels (i.e., simple, medium, or difficult). Once complexity is determined, the number of screens, reports, and components are weighted according to the table.

OBJECT TYPE	COMPLEXITY WEIGHT		
	SIMPLE	MEDIUM	DIFFICULT
SCREEN	1	2	3
REPORT	2	5	8
3GL COMPONENT			10

Complexity Weighing for Object Points

The object point count is then determined by multiplying the original number of objects instances by the weighting factor in the figure and summing to obtain a total object point count.

When component-based development or general software reuse is to be applied, the percent of reuse (%reuse) is estimated and the object point count is adjusted:

NOP = (object points) x [(100 - %reuse)/100]

where NOP is defined as new object points.

To derive an estimate of effort based on the computed NOP value, a "productivity rate" must be derived: -

$$\text{PROD} = \frac{\text{NOP}}{\text{person-month}}$$

Productivity rate for different levels of developer experience and development environment maturity: -

Developer's experience	Very low	Low	Nominal	High	Very high
Environment maturity	Very low	Low	Nominal	High	Very high
PROD	4	7	13	25	50

Once the productivity rate has been determined, an estimate of project effort is computed using

$$\text{Estimated effort} = \frac{\text{NOP}}{\text{PROD}}$$

COCOMO ESTIMATION FOR OUR PROJECT

Number of screens = 5
Number of reports = 1
Number of 3GL components used = 0

In our project there are simple screens and reports.

So, object points = 5 x 1 + 1 x 2
= 7

Since we are not reusing any of the components in our project, the % reuse is zero here.

NOP = 7 x [(100-0)/100] = 7

PROD = 4

Estimated effort = $\frac{\text{NOP}}{\text{PROD}}$

$= \frac{7}{4}$

= 1.75 person-month (2 person-month)

5.5 RISK ANALYSIS

Risk analysis is the process of identifying the risks or uncertainties in the applications orsoftware.

Steps involved in Risk analysis are: -

- Risk identification i.e., identify anything that may cause harm. Decide who may be harmed and how.
- Assess the risks and make record of the findings Development of plan to manage these risks.

Risk Identification

Risk identification is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.).

There are two distinct types of risks: -

Generic risks are a potential threat to every software project.

Product-specific risks are the risks that are specific for that project.

The checklist can be used for risk identification in the following generic subcategories:

• **Product size (PS)—**Risks associated with the overall size of the software to be built or modified.

• **Business impact (BU**)—Risks associated with constraints imposed by management or the marketplace.

• **Stakeholder characteristics (CU)—**Risks associated with the sophistication of the stakeholders and the developer's ability to communicate with stakeholders in a timely manner.

• **Process definition (PD)—**Risks associated with the degree to which the software process has been defined and is followed by the development organization.

• **Development environment (DE)—**Risks associated with the availability and quality of the tools to be used to build the product.

• **Technology to be built (TE)—**Risks associated with the complexity of the system to be built and the "newness" of the technology that is packaged by the system.

• **Staff size and experience (ST)—**Risks associated with the overall technical and project experience of the software engineers who will do the work.

Assessing Overall Project Risk

The following questions have derived from risk, data obtained by surveying experiencedsoftware project managers in different parts of world.

1. Have top software and customer managers formally committed to support the project?
   Yes

2. Are end users enthusiastically committed to the project and the system/ product to bebuilt?

   Yes

3. Are requirements fully understood by the software engineering team and its customers?

   Yes

4. Have customers been involved fully in the definition of requirements?
   Yes

5. Do end users have realistic expectations?
   Yes

6. Is the project scope stable?
   Yes

7. Does the software engineering team have the right mix of skills?
   No

8. Are project requirements stable?
   Yes

9. Does the project team have experience with the technology to be implemented?
   No

10. Is the number of people on the project team adequate to do the job?
    No

11. Do all customer/user constituencies agree on the importance of the project and on therequirements for the system/product to be built?

Yes

Risk Projection

Risk projection, also called risk estimation, attempts to rate each risk in two ways—

(1) The likelihood or probability that the risk is real and will occur and
(2) The consequences of the problems associated with the risk, should it occur.

Four risk projection steps are:

1. Establish a scale that reflects the perceived likelihood of a risk.
2. Delineate the consequences of the risk.
3. Estimate the impact of the risk on the project and the product.
4. Assess the overall accuracy of the risk projection so that there will be no misunderstandings.

RISK TABLE FOR THE PROJECT

RISKS	CATEGORY	PROBABILITY	IMPACT
Size estimate may be significantlylow	PS	25%	2
Large no. of user than planned	PS	20%	2
Less reuse than planned	PS	35%	3
End user may resist the system	BU	30%	2
Delivery decline will be tightened	BU	40%	2
Funding will be lost	CU	30%	1
Customer will change requirements	PS	25%	3
Technology will not meet expectation	TE	20%	2

Lack of training on tools	DE	40%	2
Self in-experienced	ST	40%	3
Staff turnover will be high	ST	5%	2

Figure 7: Risk Table

Impact Values:

1- Catastrophic
2- Critical
3- Marginal
4- Negligible

6.DESIGN

6.1 DATA DESIGN

USER INFORMATION TABLE

Field Name	Type	Specifications	Constraint	Unique	Description
Name	String	30 alphanumeric Characters	Not Null	Yes	Name of the user
Username	Alphanumeric	6 alphanumeric Characters	Not Null, Primary Key	Yes	User Id of User
Password	Alphanumeric	10 alphanumeric Characters	Not Null	Yes	User's password must contain atleast 8 alphanumeric characters

USER PROFILE TABLE

Field Name	Type	Specifications	Constraint	Unique	Description
Username	Alphanumeric	6 alphanumeric characters	Not Null, Primary Key	Yes	Name of the player
Games Played	Integer	3 digits	Null	No	Number of gamesplayed
County	Varchar	30 characters	Not Null	No	Name of the country
Account Since	Date Time	DD-MM-YY TT:TT:TT	Not null	No	When the playercreated his/her account
Total points	Integer	5 digits	Null	No	Total points scored bythe player

GAME DETAILS TABLE

Field Name	Type	Specifications	Constraint	Unique	Description
Name of the Game	Varchar	50 alphanumeric characters	Not Null	Yes	Name of game
Levels	Integer	2 digits	Not Null	No	Easy/Medium/ Hard
Last Played On	Date Time	YY-MM-DD TT:TT:TT	Null	No	Date of game last played
Points	Integer	10 digits	Null	No	Point scored by the player

QUIZ HISTORY TABLE

Field Name	Type	Specifications	Constraint	Unique	Description
Name of the Quiz	String	30 alphabetic characters	Not Null	Yes	Name of the quiz
Points Scored	Integer	10 digits	Null	No	This provides the points when player played the quiz.
Times Played	Integer	3 digits	Null	No	Number of times thequiz is played after evaluation.

FEEDBACK TABLE

Field Name	Type	Specifications	Constraint	Unique	Description
Name of the User	String	30 alphabetic characters	Not Null	Yes	Name of the student
Rating/Customer Satisfaction	Integer	2 digits	Null	No	This provides the data when user submitted the Feedback.

6.2 COMPONENT LEVEL DESIGN

6.2.1 CODE FOR KIDS (ONLINE GAMING WEBSITE)

home.html

```
<!DOCTYPE html>
<html>
   <head>
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css">
      <title>KIDS | Single and MultiPlayers Games |</title>
      <link rel="icon" href="image/logo.png">
      <link rel="stylesheet" type="text/css" href="home.css">
      <script src="https://cdn.tailwindcss.com"></script>
      <meta charset="UTF-8">
   </head>
   <body>
      <header>
         <img src="image/logo.png" alt="" width=80px>
         <h1>KIDS</h1>
            <ul>
               <li>
                  <a href="home.html" target="_blank">Home</a>
                </li>
               <li>
                 <a href="login.php" target="_blank">LogIn</a>
               </li>
               <li>
                 <a href="about_us.html" target="_blank">About Us</a>
               </li>
               <li>
                 <a href="" target="_blank">Quiz</a>
               </li>
               <li>
                  <a href="games.html" target="_blank">Games</a>
               </li>
            </ul>
      </header>
      <nav><ul>
         <li><a href="">
```

```
            <i class="fas fa-user"></i>
            <span class="nav-item">Dashboard</span>
        </a></li>
        <li><a href="">
            <i class="fas fa-question-circle"></i>
            <span class="nav-item">ChatBot</span>
        </a></li>
        <li><a href="feedback.html">
            <i class="fas fa-comment"></i>
            <span class="nav-item">Feedback</span>
        </a></li>
        <li><a href="">
            <i class="fas fa-search"></i>
            <span class="nav-item">Search</span>
        </a></li>
        <li><a href="">
            <i class="fas fa-level-up"></i>
            <span class="nav-item">Level</span>
        </a></li>
        <li><a href="">
            <i class="fas fa-arrows"></i>
            <span class="nav-item">Mode</span>
        </a></li>
        <li><a href="">
            <i class="fas fa-cog"></i>
            <span class="nav-item">Settings</span>
        </a></li>
        <li><a href="">
            <i class="fas fa-repeat"></i>
            <span class="nav-item">Reset</span>
        </a></li>
        <li><a href="logout.php">
            <i class="fas fa-sign-out-alt"></i>
            <span class="nav-item">Logout</span>
        </a></li>
    </ul>
</nav>
    <main>
        <p>Welcome to the Gaming World</p>
        <a href="games.html"><button>Play Now</button></a>
    </main>
    <footer>
```

```
        <div class="footer-content">
            <h3>KIDS</h3>
            <p>Games for kids developed by kids</p>
            <ul class="socials">
                <li><a href="#"><i class="fab fa-facebook-f"></i></a></li>
                <li><a href="#"><i class="fab fa-twitter"></i></a></li>
                <li><a href="#"><i class="fab fa-instagram"></i></a></li>
                <li><a href="#"><i class="fab fa-linkedin"></i></a></li>
                <li><a href="#"><i class="fab fa-youtube"></i></a></li>
            </ul>
        </div>
        <div class="footer-bottom">
            <p>copyright &copy;2023 KIDS. designed by <span>the
team</span></p>
        </div>
    </footer>
  </body>
</html>
```

home.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;700&displ
ay=swap');
*{
  margin: 0;
  padding: 0;
  font-family: sans-serif;
  box-sizing: border-box;
}

body{
  background: url(image/wallpaper10.jpg);
  background-size: 100%;
  background-repeat: no-repeat;
  background-position: center;
  padding-bottom: 58px;
  min-height: 100vh;
}

header{
  height: 60px;
  display: flex;
```

```
    align-items: center;
    text-align: center;
    justify-content: space-around;
    color: white;
    background-color: #000;
}

nav{
    position: relative;
    float: right;
    top: 0;
    bottom: 0;
    height: 100%;
    left: 0;
    background: #000;
    width: 90px;
    overflow: hidden;
    transition: width 0.2s linear;
    box-shadow: 0 20px 35px rgba(0,0,0,alpha);
}

.fas{
    position: relative;
    width: 70px;
    height: 40px;
    top: 14px;
    font-size: 20px;
    text-align: center;
}

.nav-item{
    position: relative;
    top: 12px;
    margin-left: 10px;
}

nav ul li a{
    position: relative;
    color: rgb(85,83,83);
    font-size: 14px;
    display: table;
    width: 300px;
```

```
    padding: 10px;
}

nav ul li a:hover{
    background: #eee;
}

nav:hover{
    width: 280px;
    transition: all 0.5s ease;
}

.logout{
    position: absolute;
    bottom: 0;
}

header ul{
    list-style: none;
    display: flex;
}

header h1{
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
    font-size: 40px;
    font: bolder;
}

header ul li{
    width: 150px;
    height: 60px;
    Line-height: 60px;
}

header ul li a{
    font: size 20px;
    color:white;
    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
}

header ul li:hover{
    background: linear-gradient(#cc66ff,#f4f4f4);
```

```
    color: #000;
}

header ul li:nth-child(1){
    border-top: 4px solid #cc33ff;
}

main{
    height: calc(100vh - 60px);
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
}

main p{
    width: 700px;
    height: 80px;
    Line-height: 80px;
    text-align: center;
    font-size: 50px;
    background-color: #000;
    color: white;
    border-radius: 20px;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
}

main button{
    width: 250px;
    height: 60px;
    border: none;
    outline: none;
    border-radius: 10px;
    margin-top: 30px;
    font-size: 30px;
    background: linear-gradient(#cc66ff,#f4f4f4);
    cursor: pointer;
    font-family: fantasy;
}

main button:hover{
    transform: translateY(5px);
```

```
    transition: 0.2s;
}

footer{
    position:absolute;
    left: 0;
    right: 0;
    height:10%;
    width:100%;
    padding-top: 2px;
    bottom: 0;
    background: black;
    color: white;
}

.footer-content{
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    text-align: center;
    background-color: #000;
}

.footer-content h3{
    font-size: 1.8rem;
    font-weight: 400;
    text-transform: capitalize;
    line-height: 3rem;
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
}

.footer-content p{
    max-width: 500px;
    margin: 10px auto;
    line-height: 28px;
    font-size: 14px;
    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
}

.socials{
    list-style: none;
```

```
    display: flex;
    align-items: center;
    justify-content: center;
}

.socials li{
    margin: 0 10px;
}

.socials a{
    text-decoration: none;
    color: #fff;
}

.socials a i{
    font-size: 1.1rem;
    transition: color .4s ease;
}

.socials a:hover i{
    color: aqua;
}

.footer-bottom{
    background: #000;
    width: 100vw;
    padding: 10px 0;
    text-align: center;
}

.footer-bottom p{
    font-size: 14px;
    word-spacing: 2px;
    text-transform: capitalize;
    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
}

.footer-bottom span{
    text-transform: uppercase;
    opacity: .4;
    font-weight: 200;
}
```

6.2.2 PHP CONNECTION

db.php

```
<?php
   // Enter your host name, database username, password, and database name.
   // If you have not set database password on localhost then set empty.
   $con = mysqli_connect("127.0.0.1:3307","root","","LoginSystem");
   // Check connection
   if (mysqli_connect_errno()){
      echo "Failed to connect to MySQL: " . mysqli_connect_error();
   }
?>
```

auth_session.php

```
<?php
   session_start();
   if(!isset($_SESSION["username"])) {
      header("Location: login.php");
      exit();
   }
?>
```

login.php

```
<!DOCTYPE  html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css">
      <link rel="stylesheet" href="login.css">
      <link rel="icon" href="image/logo.png">
      <title>Login/Registration</title>
   </head>
<body>
   <?php
   require('db.php');
   session_start();
```

```
   // When form submitted, check and create user session.
   if (isset($_POST['username'])) {
      $username = stripslashes($_REQUEST['username']);    // removes
backslashes
      $username = mysqli_real_escape_string($con, $username);
      $password = stripslashes($_REQUEST['password']);
      $password = mysqli_real_escape_string($con, $password);
      // Check user is exist in the database
      $query    = "SELECT * FROM `users` WHERE username='$username'
                AND password='" . md5($password) . "'";
      $result = mysqli_query($con, $query) or die(mysql_error());
      $rows = mysqli_num_rows($result);
      if ($rows == 1) {
         $_SESSION['username'] = $username;
         echo "<div class='form'>
               <h3>You are logged in successfully.</h3><br/>
               <p class='link'>Click here to go to <a
href='home.html'>Home</a></p>
               <p class='link'>Click here to go to <a href='games.html'>Games
Page</a></p>
               </div>";
      } else {
         echo "<div class='form'>
               <h3>Incorrect Username/password.</h3><br/>
               <p class='link'>Click here to <a href='login.php'>Login</a>
again.</p>
               </div>";
      }
   } else if (isset($_REQUEST['username'])) {
      // removes backslashes
      $username = stripslashes($_REQUEST['username']);
      //escapes special characters in a string
      $username = mysqli_real_escape_string($con, $username);
      $email    = stripslashes($_REQUEST['email']);
      $email    = mysqli_real_escape_string($con, $email);
      $password = stripslashes($_REQUEST['password']);
      $password = mysqli_real_escape_string($con, $password);
      $create_datetime = date("Y-m-d H:i:s");
      $query    = "INSERT into `users` (username, password, email,
create_datetime)
                VALUES ('$username', '" . md5($password) . "', '$email',
'$create_datetime')";
```

```
        $result   = mysqli_query($con, $query);
        if ($result) {
           echo "<div class='form'>
                <h3>You are registered successfully.</h3><br/>
                <p class='link'>Click here to <a href='login.php'>Login</a></p>
                </div>";
        } else {
           echo "<div class='form'>
                <h3>Required fields are missing.</h3><br/>
                <p class='link'>Click here to <a href='login.php'>registration</a>
again.</p>
                </div>";
        }
    } else {
?>
    <div class="hero">
        <header>
           <img src="image/logo.png" alt="" width=80px>
           <h1>KIDS</h1>
              <ul>
                 <li>
                    <a href="home.html" target="_blank">Home</a>
                  </li>
                 <li>
                  <a href="login.php" target="_blank">LogIn</a>
                 </li>
                 <li>
                  <a href="about_us.html" target="_blank">About Us</a>
                 </li>
                 <li>
                  <a href="" target="_blank">Quiz</a>
                 </li>
                 <li>
                    <a href="games.html" target="_blank">Games</a>
                 </li>
              </ul>
        </header>
        <div class="form-box">
           <div class="button-box">
              <div id="btn"></div>
              <button type="button" class="toggle-btn"
onclick="login()">Login</button>
```

```
            <button type="button" class="toggle-btn"
onclick="register()">Register</button>
        </div>
        <div class="social">
            <a href=""><i class="fab fa-google"></i></a>
            <a href=""><i class="fab fa-twitter"></i></a>
            <a href=""><i class="fab fa-facebook-f"></i></a>
        </div>
        <form id="login" method="post" class="input-group">
            <input type="text" class="input-field" name="username"
placeholder="User ID" required>
            <input type="password" class="input-field" name="password"
placeholder="Password" required>
            <input type="checkbox" class="check-box"><span>Remember
Password</span>
            <button type="submit" class="submit-btn">LOGIN</button>
        </form>
        <form id="register" method="request" class="input-group">
            <input type="text" class="input-field" name="username"
placeholder="User ID" required>
            <input type="email" class="input-field" name="email"
placeholder="Email ID" required>
            <input type="password" class="input-field" name="password"
placeholder="Password" required>
            <input type="checkbox" class="check-box"><span>I agree to the
terms and conditions</span>
            <button type="submit" name="submit" value="Register"
class="submit-btn">REGISTER</button>
        </form>
    </div>
    <footer>
        <div class="footer-content">
            <h3>KIDS</h3>
            <p>Games for kids developed by kids</p>
            <ul class="socials">
                <li><a href="#"><i class="fab fa-facebook-f"></i></a></li>
                <li><a href="#"><i class="fab fa-twitter"></i></a></li>
                <li><a href="#"><i class="fab fa-instagram"></i></a></li>
                <li><a href="#"><i class="fab fa-linkedin"></i></a></li>
                <li><a href="#"><i class="fab fa-youtube"></i></a></li>
            </ul>
        </div>
```

```
            <div class="footer-bottom">
                <p>copyright &copy;2023 KIDS. designed by <span>the
team</span></p>
            </div>
        </footer>
    </div>
    <script>
        var x=document.getElementById("login");
        var y=document.getElementById("register");
        var z=document.getElementById("btn");

        function register(){
            x.style.left="-400px";
            y.style.left="50px";
            z.style.left="110px";
        }
        function login(){
            x.style.left="50px";
            y.style.left="450px";
            z.style.left="0";
        }
    </script>
    <?php
}
?>
</body>
</html>
```

7. TESTING

7.1 WHITE BOX TESTING FOR LOGIN MODULE

1. Begin
2. Open the login page
3. If user already have an account
4. Login
5. Enter e-mail and pswd
6. If email and pswd valid
7. Logged in to the system
8. Else
9. Login again
10. Else
11. Sign up
12. Fill out the info
13. Submit and Login
14. End

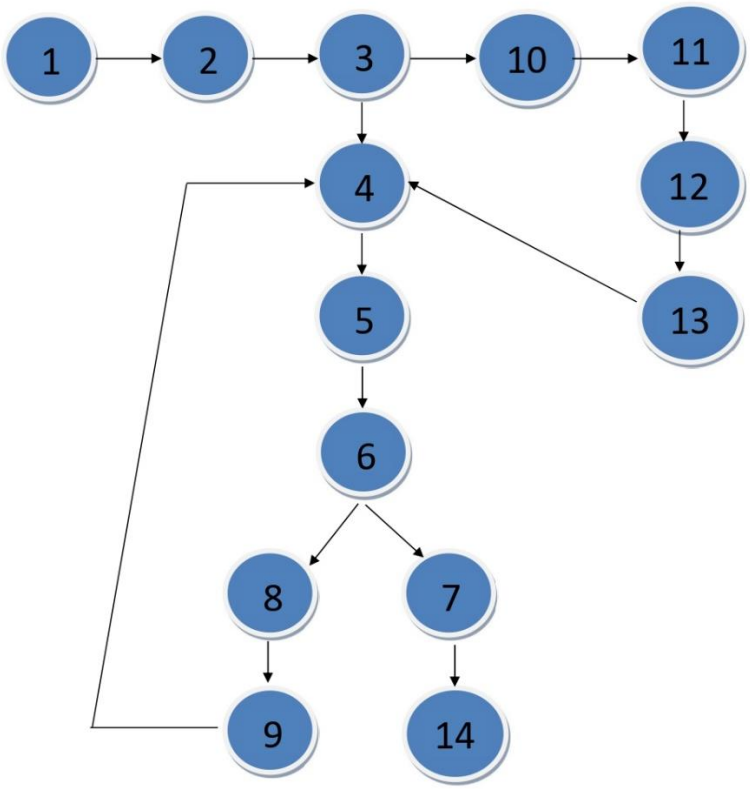


Fig 7.1 CONTROL FLOW DIAGRAM FOR LOGIN

CYCLOMATIC COMPLEXITY OF RESULTANT GRAPH

CC = Total number of regions

= 3

CC = Number of edges – Number of nodes + 2p

= 15 – 14 + 2x1

= 17 – 14

= 3

CC = Number of predicate nodes + 1

= 2 + 1

= 3

INDEPENDENT PATHS FROM THE GRAPH

Path 1: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 14

Path 2: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 8 -> 9 -> 4 -> 5 -> 6 -> 7 -> 14

Path 3: 1 -> 2 -> 3 -> 10 -> 11 -> 12 -> 13 -> 4 -> 5 -> 6 -> 7 -> 14

GENERATE TEST CASES FROM BASIS PATH SET

1. User successfully gets logged in to the system
2. User is asked to enter login credentials again
3. User successfully registers into the system

7.2 WHITE BOX TESTING FOR FEEDBACK MODULE

1. User login to the website
2. If details are correct then
3. Login approved
4. User can give the feedback
5. Notification retrieved
6. Else
7. Incorrect credentials
8. End if

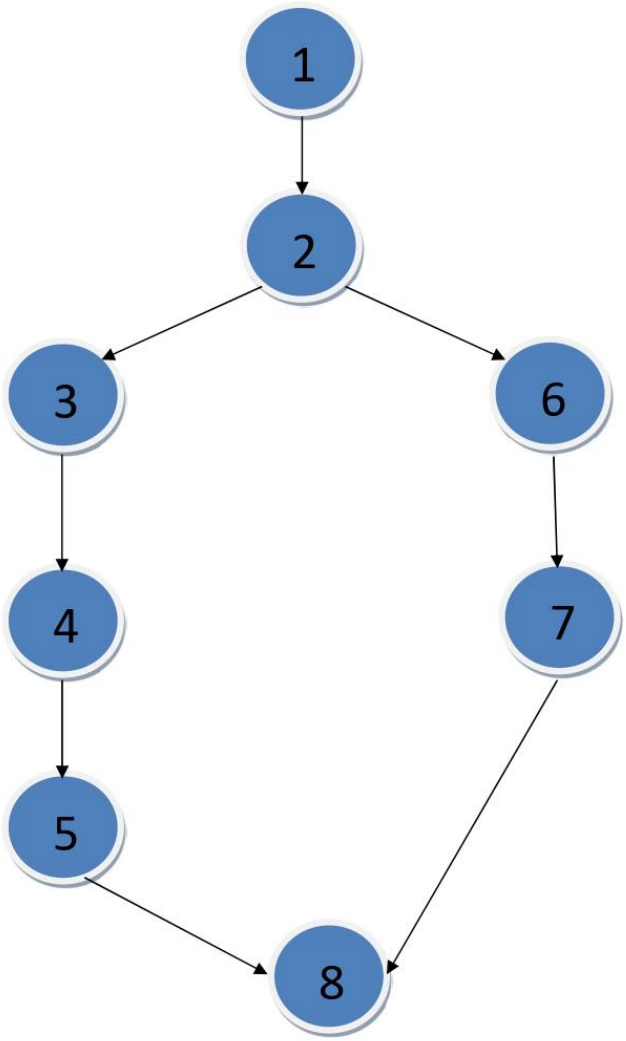


Fig 7.2 CONTROL FLOW DIAGRAM FOR FEEDBACK

CYCLOMATIC COMPLEXITY OF RESULTANT GRAPH

CC = Total number of regions

= 2

CC = Number of edges – Number of nodes + 2p

= 8 – 8 + 2x1

= 10 – 8

= 2

CC = Number of predicate nodes + 1

= 1 + 1

= 2

INDEPENDENT PATHS FROM THE GRAPH

Path 1: 1 -> 2 -> 3 -> 4 -> 5 -> 8

Path 2: 1 -> 2 -> 6 -> 7 -> 8

GENERATE TEST CASES FROM BASIS PATH SET

1. User successfully gives the feedback
2. User isn't able to give feedback

7.3 WHITE BOX TESTING FOR GAMES PAGE MODULE

1. Open the homepage
2. Login
3. While user wants to play game
4. Pick game
5. Wait for game to end
6. Evaluate scores
7. Update user's scores
8. End while

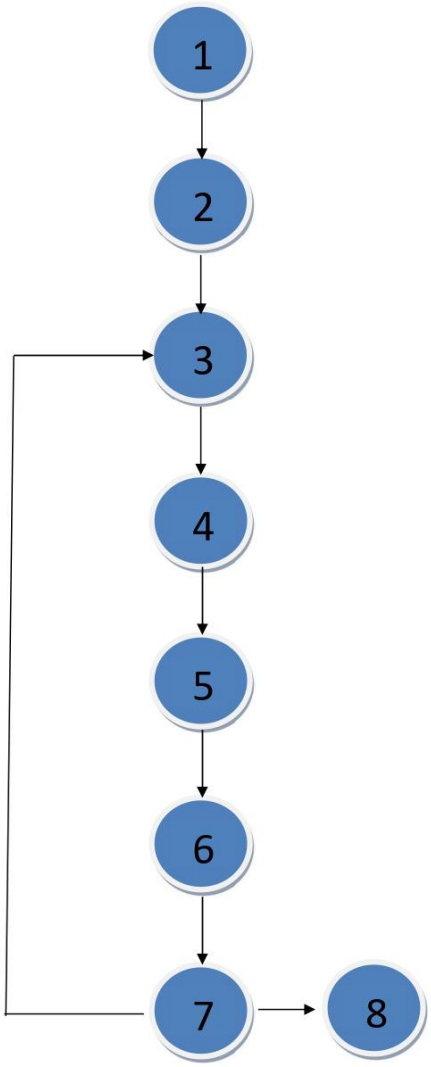


Fig 7.3 CONTROL FLOW DIAGRAM FOR GAMES PAGE

CYCLOMATIC COMPLEXITY OF RESULTANT GRAPH

CC = Total number of regions

= 2

CC = Number of edges – Number of nodes + 2p

= 8 – 8 + 2x1

= 10 – 8

= 2

CC = Number of predicate nodes + 1

= 1 + 1

= 2

INDEPENDENT PATHS FROM THE GRAPH

Path 1: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8

Path 2: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8

GENERATE TEST CASES FROM BASIS PATH SET

1. User plays the selected game
2. User ends the game

8. REFERENCES

1. **Pressman, R. S., & Maxim, B. R., "Software Engineering: A Practitioner's Approach", 8th edition, (2015). McGraw-Hill.**

2. **Aggarwal, K. K., & Singh, Y., "Software Engineering", 3rd edition, (2007), New Age International Publishers.**

9. ANNEXURES

Code for our website -> anjalit03/kids (github.com)

Website link -> KIDS | Single and MultiPlayers Games | (kidzzz.netlify.app)

(Deployment done partially)