Minor Project Report

On

BYGONE

(Retro Gaming and Video Streaming Website)

Submitted by:

Piyush Mali [19100BTIT06588]

Raghav Sood [19100BTIT06595]

Tanu Khatri [19100BTIT06627]

Under the guidance of

PROF. GAURAV VINCHURKAR

DEPARTMENT OF INFORMATION TECHNOLOGY

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY
SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDORE

JAN - JUNE 2022

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY, INDORE

DECLARATION

We here declare that work which is being presented in the project entitled **"BYGONE"** (Retro gaming and video streaming website) in partial fulfilment of degree of **Bachelor of Technology** is an authentic record of our work carried out under the supervision and guidance of **Mr. Gaurav Vinchurkar** Asst. Professor of Information Technology. The matter embodied in this project has not project has not been submitted for the award of any other degree.

Piyush Mali	Signature: _____________
Raghav Sood	Signature: _____________
Tanu Khatri	Signature: _____________

Date:

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY, INDORE

PROJECT APPROVAL SHEET

Following team has done the appropriate work related to the **“BYGONE”** (Retro gaming and video streaming website) in partial fulfilment for the award of **Bachelor of Technology** of “SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDORE” and is being submitted to SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY, INDORE.

Team:

1. **Piyush Mali**
2. **Raghav Sood**
3. **Tanu Khatri**

Internal Examiner **External Examiner**

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY, INDORE

CERTIFICATE

We are pleased to certify that the dissertation work submitted by Piyush Mali, Raghav Sood and Tanu Khatri entitled as **“BYGONE”** (Retro gaming and video streaming website) is partial fulfilment of degree in **Bachelor of Information Technology** awarded by **Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore** for academic year 2022.

Project Coordinator:	**Head of Department:**
Prof. Gaurav Vinchurkar	Dr. Jigyasu Dubey

ACKNOWLEDGEMENT

It has been great honor and privilege to complete our Minor project on Retro gaming and video streaming website (BYGONE) under the guidance of Prof. Gaurav Vinchurkar. We are very much thankful to Dr. Jigyasu Dubey (Head of the Department, IT) for providing all facilities and support to meet our project requirements. We would like to take this opportunity to express our humble gratitude toward under whom we executed this project. Their constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depth and helped us complete its assigned tasks. We are thankful to our project internal guide, whose invaluable guidance helped us understand this project better. Although there may be many who remain unacknowledged in this humble note of gratitude, there are none who remain unappreciated.

Piyush Mali

Raghav Sood

Tanu Khatri

TABLE OF CONTENT

Title	Page no.
0. ABSTRACT	**08**
1. INTRODUCTION	**09**
1.1 Problem Statement	
1.2 Need for the new System	
1.3 Objective	
1.4 Application	
2. LITRATURE SURVEY	**12**
2.1 Work done by other	
2.2 Benefits	
2.3 Proposed Solution	
2.4 Technology Used	
2.4.1 Software Requirement	
2.4.2 Hardware Requirement	
3. SOFTWARE ENGINEERING APPROACH	**15**
3.1 Software Engineering Paradigm Applied	
3.2.1 Description	
3.2.2 Advantage & Disadvantages	
3.2.3 Reasons for use	
3.2 Requirement Analysis	
3.2.1 Software Requirement Specification	
3.2.2 Glossary	
3.2.3 Supplementary Specifications	

3.2.4 Use Case Model

4. DESIGN **21**

4.1 Design Concept

4.2 Design Technique

4.3 Modeling

4.3.1 DFD Model Level-0 and 1

4.4 Activity Diagram

4.5 Class Diagram

4.6 Sequence Diagram

5. IMPLEMENTATION

5.1 Screenshots

5.2 Coding

6. TESTING OBJECTIVE **43**

7. LIMITATION OF PROJECT **46**

8. FUTURE ENHANCEMENT **47**

9. CONCLUSION **48**

10. REFERENCES **49**

Abstract

The goal of this project is primarily to establish a collection of Retro games we used to play and cartoon shows that we used to watching in our childhood. The other day we were having a throwback Thursday moment and remembered all of our favorite vintage games and cartoon. We were looking for these and it took us a while to find them, so we got the idea to write all of our favorite retro games, put them all in the same website and add to the website our findings regarding our favorite retro cartoons!

Not everyone has access to a local vintage gaming store when it comes to old games. Even if you know about an old-school games store in your area, it may not have the game you're seeking for. After all, certain games are really rare. Fortunately, there are several retro gaming stores on the internet. I'm not talking about Amazon or eBay merchants, because the quality of what you get isn't always guaranteed.

And, if we're talking about cartoon shows, what happened to the current generation? Their cartoons, on the other hand, are terrible. Back in the day, cartoons not only entertained us, but they also taught us. Pick any cartoon from the 1990s to the 2010s, and it will almost certainly give today's best animation a run for its money. As a result, the second section of our website features a variety of old cartoon shows that we used to watch as kids.

1. INTRODUCTION

When nostalgia hits you hard, all you want to do is curl up on the couch and wallow in it. Do you want to call over an old friend and go back to your childhood phase where you used to watch the old cartoons? This may get you thinking of how and where to watch old cartoon shows? YouTube and Vimeo are options, but you may not always get lucky with those. You may view classic cartoon shows and play popular retro video games for free on our website. Our website "Bygone" is a collection of old cartoon shows and retro games that we used to watch as kids. A place where we can hunt for old games from our childhood. Look for old shows and view them with a single click. These games reflect the pinnacle of video gaming for previous generations. Our goal is to revive games from the 1990s to 2010s that were made for consoles and platforms that are no longer available for purchase and are no longer maintained by their developers and manufacturers. It ties us to the good old days with today's technology because it is a webpage that can be accessed from anywhere.

It is simple to use. This is something you can utilize at any time and in any location. You can play games without having to download them; all you need is an internet connection. It also contains old cartoon shows on a single platform that take us back to the old days when we used to watch them on televisions, but this time it's on a single page with much more options and choices. Playing and scoring points is fun because we can share them on various social platforms and become popular among others.

1.1 Problem Statement

- The internet has become an integral part of human life in this age of information technology. The number of people using the internet is growing every day. Users like to learn new things and keep themselves engaged by playing online games, listening to music, and so on, but they can't find their childhood favourite games and cartoon shows on a single platform. Users were looking for these, and it took us a long time to find them, so we came up with the idea of writing down all of our favourite vintage games, putting them all on one website, and adding our findings on our favourite retro cartoons to the website.

- The installation of today's games is extremely complicated, requiring manual path setup and configuration. These games also need a very specific environment to run smoothly, like up-to-date software such as DirectX 11, Microsoft Visual C++, and the most recent version graphic drivers.

- Today's games are particularly fast-paced because they need 100% participation from the player, which can last for hours because most online games cannot be paused.

- All the retro animated shows that are all time hit are scattered over multiple platforms and it creates inconvenience for the user to switch between the platform. And in case the user wishes to play some games that becomes a new problem.

- Today's Games and Videos have extremely realistic graphics and necessitate large amount of texture data which consumes RAM and storage very quickly resulting in increase of hardware consumption and power consumption.

1.2 Need for the New System

- Going back to these older games helps you join a growing community of people who are going back. With today’s technology, you can join forums where you get to talk to other people who are playing the same games. Maybe you get to discover cheat codes you didn’t know about when you were younger, or you might make friends with people who share similar interests. Finding a community that you share a lot in common with is a good thing for anyone. It makes you feel like you belong, and that’s something all human beings need. Those who don’t want to use older consoles can play retro games directly on their computer. Some sites offer these games free of charge. You get to play while you chat with other players live. That’s something you could never do before. Of course, you can always host game days where you invite your friends to come and play with you for a bit.

- Good customer service- Paying Attention to Issues that May Arise from Excessive Gaming.

- User Friendly and Low Functionality- Easy to use, anyone can play games and watch cartoon shows.

- Single platform for Games and Shows-One stop solution for sourcing the Retro Games and Shows. You can play multiple games and also watch cartoon shows on the single platform.

- No need of any download- You can play game without downloading it and can also watch shows online. Internet is all that you need. No need of any downloads.

- Registration-100% free registration via Gmail or social media.

- Developer support of player feedback.

- Ease of player interaction.

1.3 Objective

The primary purpose of this project is to create a collection of retro games and cartoon shows that we used to watch as kids. Our goal is to resurrect games from the 1990s to 2010s that were created for systems and platforms that are no longer available for purchase and whose developers and makers have abandoned them. Because it is a webpage that can be visited from anywhere, it connects us to the good old days with today's technology.

1.4 Applications

- Device Compatibility:
  - Implement Responsive Design
  - Simplify Navigation
  - Get rid of Disruptions
  - Be uncompromising with Speed
  - Simplify Design
  - Keep the Search option front and center
  - Ensure cross-browser and cross-device compatibility

- Optimal Accessibility and Cost: The game should be easily accessible from any connected device, whether a stationary computer, a laptop, or a mobile device. Furthermore, it should not impose any additional costs on you. The benefit of online games is that they do not demand the high price ascribed to video and RPG games, and bring you a gaming experience without charges.

- High Quality Library: The cartoon shows should also be available in different pixel qualities, giving users the flexibility to change the quality as per their viewing preference and infrastructure.

- Wide variety of content: Discovering new and personalized content is the key feature that improves customer loyalty.

2. LITERATURE SURVEY

The number one cause of this assignment is to create a collection of retro video games and cool animated film indicates that we used to look at as kids. Our purpose is to resurrect video games from the 1990s to 2010s that had been created for structures and systems which might be not to be had for buy and whose builders and makers have deserted them. Because it's miles a website that may be visited from anywhere, it connects us to the best vintage days with ultra-modern technology. This is an internet-based web application which can be accessed from anytime anywhere basis.

Going again to those older video games allows you be a part of a developing network of those who are going again. With today's technology, you may be a part of boards in which you get to speak to different those who are gambling the identical video games. Maybe you get to find out cheat codes you didn't understand approximately while you have been younger, or you would possibly make buddies with those who percentage comparable interests. Finding a network which your percentage loads in not unusual place with is a great issue for anyone. It makes you experience such as you belong, and that's something all people need. Those who don't need to apply older consoles can play retro video games without delay on their computer. Some web sites provide those video games freed from charge. You get to play even as you chat with different gamers live. That's something you may by no means do before. Of course, you may continually host recreation days in which you invite your buddies to return back and play with you for a bit.

2.1 Work Done by other

Numerous programmers have attempted and advanced many retro gaming and cool animated film indicates internet site which are additionally blended each matter in addition to our task is likewise stimulated from the one's concepts. We have attempted to make our assignment tons steady in addition to higher that the rest, with the aid of using such as a sure component for authentication and verification it makes much higher and steady.

2.2 Benefits

- Registration-100% free registration through Gmail or social media.
- Ease of participant interaction.
- User Friendly and Low Functionality- Easy to use, everyone can play video games and watch cool animated film shows.
- Single platform for Games and Shows-One forestall answer for sourcing the Retro Games and Shows. You can play a couple of video games and additionally watch cool animated film suggests at the single platform.

- Good customer service- Paying Attention to Issues that May Arise from Excessive Gaming.

- No need of any download- You can play game without downloading it and also can watch shows online. Internet is all which you want. No need of any downloads.

2.3 Proposed Solution

The internet has end up a fundamental a part of human existence on this age of information technology. The wide variety of human beings the usage of the internet is developing each day. Users want to research new matters and preserve themselves engaged through gambling online video games, taking note of music, and so on, however they cannot discover their formative years favored video games and cool animated film indicates on a single platform. Users had been searching out these, and it took us a long term to discover them, so we got here up with the concept of writing down all of our favorite vintage video games, setting all of them on one website, and including our findings on our favorite retro cartoons to the website.

With today's technology, you can join a forum where you can talk to others playing the same game. You may discover cheat codes that you didn't know when you were young, or make friends with people with similar interests. Finding a community that has a lot in common is good for everyone. It makes you feel like you belong, and that's what everyone needs. If you don't want to use the old console, you can play retro games directly on your computer. Some websites offer these games for free. You can play while live chatting with other players. This has never been possible before. Of course, you can always hold a Game Day to invite your friends to play with.

It is straightforward to use. This is something you could make use of at any time and in any location. You can play video games while not having to download them; all you want is a web connection. It additionally consists of retro cool animated film indicates on a single platform that take us again to the retro days while we used to observe them on televisions, however this time it`s on a single web page with tons extra alternatives and choices. Playing and scoring factors is amusing due to the fact we are able to percentage them on diverse social systems and emerge as famous amongst others.

2.4 Technology Used:

2.4.1 Software Requirements:

- Html, CSS, JavaScript

- NodeJS
- Firebase

2.4.2 Hardware Requirements:

- System- Multimedia PC
- Processor- Intel Pentium 4\Core i3 or equivalent
- Memory- 1Gb RAM

3. SOFTWARE ENGINEERING APPROACH

3.1 Software Engineering paradigm Applied

We have used an Agile SDLC Model for building our project.

3.1.1 Description

Agile methodologies are product development approaches that are consistent with the Agile Manifesto's values and principles for software development. Agile techniques attempt to produce the proper product through small cross-functional self-organizing teams that supply small pieces of functionality on a regular basis, allowing for frequent customer input and course correction as needed.

Agile development model is also a type of Incremental model. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained.

In doing so, Agile tries to address the issues that traditional "waterfall" techniques of delivering huge goods over extended periods of time encounter, such as client requirements changing frequently and resulting in the delivery of incorrect products.

The agile software development emphasizes on four core values.

a. Individual and team interactions over processes and tools
b. Working software over comprehensive documentation
c. Customer collaboration over contract negotiation
d. Responding to change over following a plan

3.1.2 Advantages & Disadvantages

Advantages of Agile model:

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages of Agile model:

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

3.1.3 Reasons for use

- When new changes are needed to be implemented. The freedom agile gives to change are very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers, need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

3.2 Requirement Analysis

Requirement Analysis, also known as Requirement Engineering, is the process of defining user expectations for a new software being built or modified. In software engineering, it is sometimes referred to loosely by names such as requirements gathering or requirements capturing. Requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements.

Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design. Conceptually, requirements analysis includes four types of activity:

• Eliciting requirements

The task of communicating with customers and users to determine what their requirements are. This is sometimes also called requirements gathering.

• Analyzing requirements:

Determining whether the stated requirements are unclear, incomplete, ambiguous, or contradictory, and then resolving these issues.

• Requirements modeling

Requirements might be documented in various forms, such as natural-language documents, use cases, user stories, or process specifications.

• Review and retrospective

Team members reflect on what happened in the iteration and identifies actions for improvement going forward.

Requirements are generally split into two types: -

Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements. They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability

- Flexibility

3.2.1 Software Requirement Specification:

Software Requirements Specifications (SRS) are the requirements-gathering stage of the software development process (also called a requirements document). When all requirements have been obtained and reviewed, this report establishes the groundwork for software engineering tasks. SRS is a formal report that serves as a representation of software that allows customers to assess whether it (SRS) meets their needs. It also includes user needs for a system as well as precise system requirements specifications.

The SRS is a list of requirements for a certain software product, programmed, or set of apps that execute specific tasks in a specific environment. Depending on who is writing it, it fulfils a variety of purposes. The SRS could, for starters, be authored by a system's client. Second, the SRS might be written by a system developer. The two ways produce very different scenarios and serve entirely different objectives for the paper. The first case, SRS, is used to define the users' requirements and expectations. SRS, the second example, is written for a variety of objectives and acts as a contract between the customer and the developer.

3.2.2 Glossary:

The final output is the requirements specification document (SRS). For smaller problems or problems that can easily be comprehended; the specification activity might come after the entire analysis is complete. However, it is more likely that problem analysis and specification are done concurrently. All the information for specification activity as following the analysis activity. The transition from analysis to specification should also not be expected to be straightforward, even if some formal modelling is used during analysis. Essentially, what passes from requirements analysis activity to the specification activity is the knowledge acquired about the system. The modelling is essentially a tool to help obtain a thorough and complete knowledge about the proposed system.

Analysis of Factual Data:

Analysis of data is a process of inspecting, cleaning, transforming, and modelling data with the goal of highlighting useful information, suggesting conclusions. and supporting decision making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, in different business, science, and social science domains.

Data mining is a particular data analysis technique that focuses on modelling and knowledge discovery for predictive rather than purely descriptive purposes.

Identification of Essential requirement:

Identification of essential requirement is an important task in developing the project. In this system the essential requirements are identified through surveying. By surveying, the important needs of the user in our website are known. In the surveying, the different possibilities of tour information that have to be included in the website is given by questionnaire.

Questions included like:

- Need to enhanced the security and user privacy?
- Is it advantageous?

3.2.3 Supplementary Specifications:

The purpose of this document is to define requirements of online retro gaming and shows. This captures the system requirements that are not readily captured in the use cases of use case model.

Definition of input requirements:

- **Registration and login-** For watching shows and playing retro games you need to create an account without it you cannot access it. This will enable to access website. You only need email and password for login and after login you can play games and watch cartoon shows. You can also see your profile. Registration process will also add greater security to the system, if user has logged in with his/her email and password then only he/she can see their profile.
- **Viewing Profile-** When user logged in then they can see their profile. If they want to change password they can change.
- **Update Details-** Admin is able to update retro game and cartoon shows and can add user and verify user. User is also able to update their profile. User can also their password. Users can not update game and carton shows.
- **Query** – If user has any query than they can send it. admin will response for his/her query.

Definition of Processing Requirement:

The user interface will have to clear and simple so anyone can access. It will easy to use and understand. Being an online it will viewable from any computer with an internet connection. It will not complex to use. You just need to click on login and mention details and account will create and then you can watch any show from it and can play games. You do not need to download anything you can use this with internet connection. There are many functions that it can perform. The system must display the information of users to users. The users only can see there information’s, whereas admin will see all users information and will verify user’s account.

3.2.4 Use Case Model:

The Use-case model is defined as a model which is used to show how users interact with the system in order to solve a problem. As such, the use case model defines the user's objective, the interactions between the system and the user, and the system's behavior required to meet these objectives.

Various model elements are contained in use-case model, such as actors, use cases, and the association between them.

We use a use-case diagram to graphically portray a subset of the model in order to make the communication simpler. There will regularly be a numerous use-case diagram which is related to the given model, each demonstrating a subset of the model components related to a specific purpose. A similar model component might be appearing on a few use-case diagrams; however, each use-case should be consistent. If, in order to handle the use-case model, tools are used then this consistency restriction is automated so that any variations to the component of the model (changing the name, for instance) will be reflected automatically on each use-case diagram, which shows that component.

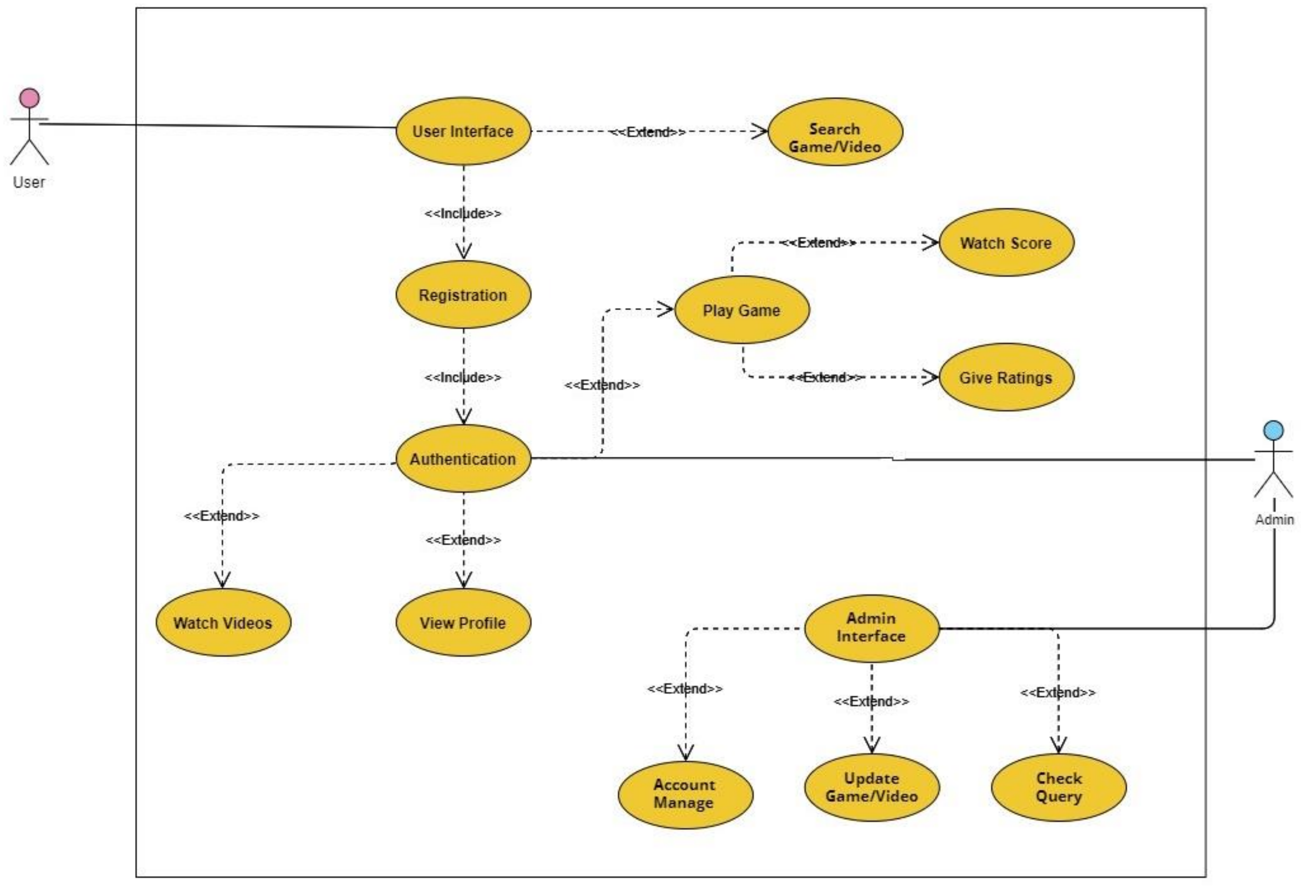


Fig. 3.2.4.1 Use Case of Retro Gaming and Video Streaming Website

4. Design

4.1 Design Concept:

The set of fundamental software design concepts are as follows:

1. Abstraction

- A solution is stated in large terms using the language of the problem environment at the highest-level abstraction.
- The lower level of abstraction provides a more detail description of the solution.
- A sequence of instruction that contain a specific and limited function refers in a procedural abstraction.
- A collection of data that describes a data object is a data abstraction.

2. Architecture

- The complete structure of the software is known as software architecture.
- Structure provides conceptual integrity for a system in a number of ways.
- The architecture is the structure of program modules where they interact with each other in a specialized way.
- The components use the structure of data.
- The aim of the software design is to obtain an architectural framework of a system.
- The more detailed design activities are conducted from the framework.

3. Patterns

A design pattern describes a design structure and that structure solves a particular design problem in a specified content.

4. Modularity

- A software is separately divided into name and addressable components. Sometime they are called as modules which integrate to satisfy the problem requirements.
- Modularity is the single attribute of a software that permits a program to be managed easily.

5. Information hiding

Modules must be specified and designed so that the information like algorithm and data presented in a module is not accessible for other modules not requiring that information.

6. Functional independence

- The functional independence is the concept of separation and related to the concept of modularity, abstraction and information hiding.

- The functional independence is accessed using two criteria i.e. Cohesion and coupling.

Cohesion

- Cohesion is an extension of the information hiding concept.
- A cohesive module performs a single task and it requires a small interaction with the other components in other parts of the program.

Coupling

Coupling is an indication of interconnection between modules in a structure of software.

7. Refinement

- Refinement is a top-down design approach.
- It is a process of elaboration.
- A program is established for refining levels of procedural details.
- A hierarchy is established by decomposing a statement of function in a stepwise manner till the programming language statement are reached.

8. Refactoring

- It is a reorganization technique which simplifies the design of components without changing its function behaviour.
- Refactoring is the process of changing the software system in a way that it does not change the external behaviour of the code still improves its internal structure.

9. Design classes

- The model of software is defined as a set of design classes.
- Every class describes the elements of problem domain and that focus on features of the problem which are user visible.

4.2 Design Technique

Software design is a process to conceptualize the software requirements into software implementation. Software design takes the user requirements as challenges and tries to find optimum solution. While the software is being conceptualized, a plan is chalked out to find the best possible design for implementing the intended solution.

There are multiple variants of software design:

Structured Design

Structured design is a conceptualization of problem into several well-organized elements of solution. It is basically concerned with the solution design. Benefit of structured design is, it gives better understanding of how the problem is being solved. Structured design also makes it simpler for designer to concentrate on the problem more accurately. Structured design is mostly based on ‘divide and conquer’ strategy where a problem is broken into several small problems and each small problem is individually solved until the whole problem is solved. The small pieces of problem are solved by means of solution modules. Structured design emphasis that these modules be well organized in order to achieve precise solution. These modules are arranged in hierarchy. They communicate with each other. A good structured design always follows some rules for communication among multiple modules, namely -

Cohesion - grouping of all functionally related elements.

Coupling - communication between different modules.

A good structured design has high cohesion and low coupling arrangements.

Function Oriented Design

In function-oriented design, the system is comprised of many smaller sub-systems known as functions. These functions are capable of performing significant task in the system. The system is considered as top view of all functions. Function oriented design inherits some properties of structured design where divide and conquer methodology is used.

This design mechanism divides the whole system into smaller functions, which provides means of abstraction by concealing the information and their operation.. These functional modules can share information among themselves by means of information passing and using information available globally.

Another characteristic of functions is that when a program calls a function, the function changes the state of the program, which sometimes is not acceptable by other modules. Function oriented design works well where the system state does not matter and program/functions work on input rather than on a state.

Design Process

- The whole system is seen as how data flows in the system by means of data flow diagram.
- DFD depicts how functions changes data and state of entire system.
- The entire system is logically broken down into smaller units known as functions on the basis of their operation in the system.
- Each function is then described at large.

Object Oriented Design

Object oriented design works around the entities and their characteristics instead of functions involved in the software system. This design strategies focuses on entities and its characteristics. The whole concept of software solution revolves around the engaged entities.

the important concepts of Object-Oriented Design:

- Objects - All entities involved in the solution design are known as objects. For example, person, banks, company and customers are treated as objects. Every entity has some attributes associated to it and has some methods to perform on the attributes.

- Classes - A class is a generalized description of an object. An object is an instance of a class. Class defines all the attributes, which an object can have and methods, which defines the functionality of the object.

  In the solution design, attributes are stored as variables and functionalities are defined by means of methods or procedures.

- Encapsulation - In OOD, the attributes (data variables) and methods (operation on the data) are bundled together is called encapsulation. Encapsulation not only bundles important information of an object together, but also restricts access of the data and methods from the outside world. This is called information hiding.
- Inheritance - OOD allows similar classes to stack up in hierarchical manner where the lower or sub-classes can import, implement and re-use allowed variables and methods from their immediate super classes. This property of OOD is known as inheritance. This makes it easier to define specific class and to create generalized classes from specific ones.
- Polymorphism - OOD languages provide a mechanism where methods performing similar tasks but vary in arguments, can be assigned same name. This is called polymorphism, which allows a single interface performing tasks for different types. Depending upon how the function is invoked, respective portion of the code gets executed.

Design Process

Software design process can be perceived as series of well-defined steps. Though it varies according to design approach (function oriented or object oriented, yet It may have the following steps involved:

- A solution design is created from requirement or previous used system and/or system sequence diagram.
- Objects are identified and grouped into classes on behalf of similarity in attribute characteristics.
- Class hierarchy and relation among them is defined.
- Application framework is defined.

Software Design Approaches

Here are two generic approaches for software designing:

Top-Down Design: We know that a system is composed of more than one sub-systems and it contains a number of components. Further, these sub-systems and components may have they're on set of sub-system and components and creates hierarchical structure in the system. Top-down design takes the whole software system as one entity and then decomposes it to achieve more than one sub-system or component based on some characteristics. Each sub-system or component is then treated as a system and decomposed further. This process keeps on running until the lowest level of system in the top-down hierarchy is achieved. Top-down design starts with a generalized model of system and keeps on defining the more specific part of it. When all components are composed the whole system comes into existence. Top-down design is more suitable when the software solution needs to be designed from scratch and specific details are unknown.

Bottom-up Design: The bottom-up design model starts with most specific and basic components. It proceeds with composing higher level of components by using basic or lower-level components. It keeps creating higher level components until the desired system is not evolved as one single component. With each higher level, the amount of abstraction is increased. Bottom-up strategy is more suitable when a system needs to be created from some existing system, where the basic primitives can be used in the newer system. Both, top-down and bottom-up approaches are not practical individually. Instead, a good combination of both is used.

4.3 Modeling:

4.3.1 DFD:

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

Levels in DFD:

- DFD Level 0 is also called a Context Diagram. It’s a basic overview of the whole system or process being analyzed or modeled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

- DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its subprocesses.

- DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system's functioning.

- Progression to Levels 3, 4 and beyond is possible, but going beyond Level 3 is uncommon. Doing so can create complexity that makes it difficult to communicate, compare or model effectively.

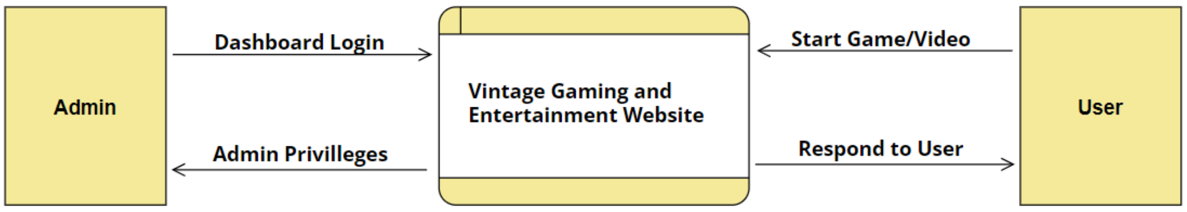


Fig 4.3.1.1 Level 0 DFD of Retro Gaming and Video Streaming Website

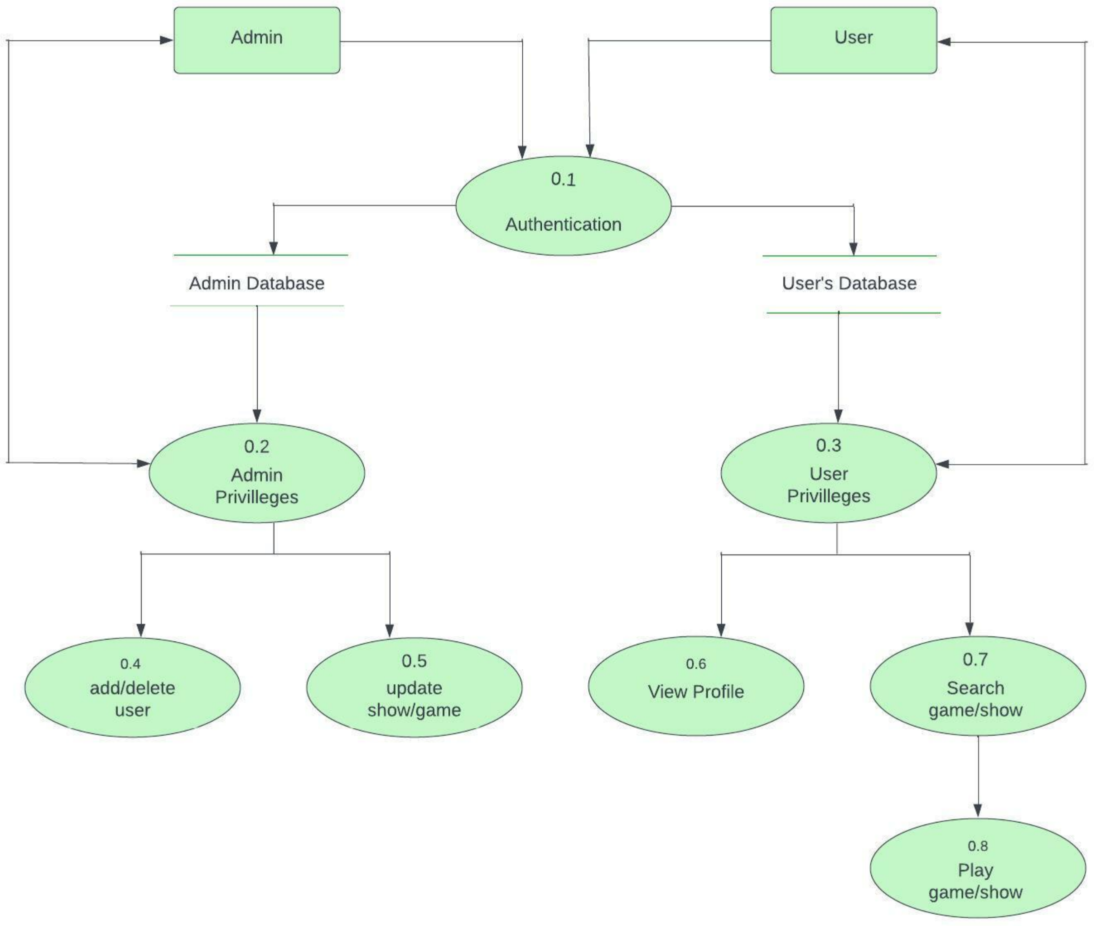


Fig 4.3.1.2 Level 1 DFD of Retro Gaming and Video Streaming Website

4.4 Activity Diagram:

Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity. Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modeling how a collection of use cases coordinates to represent business workflows

- Identify candidate use cases, through the examination of business workflows
- Identify pre- and post-conditions (the context) for use cases
- Model workflows between/within use cases
- Model complex workflows in operations on objects
- Model in detail complex activities in a high-level activity Diagram

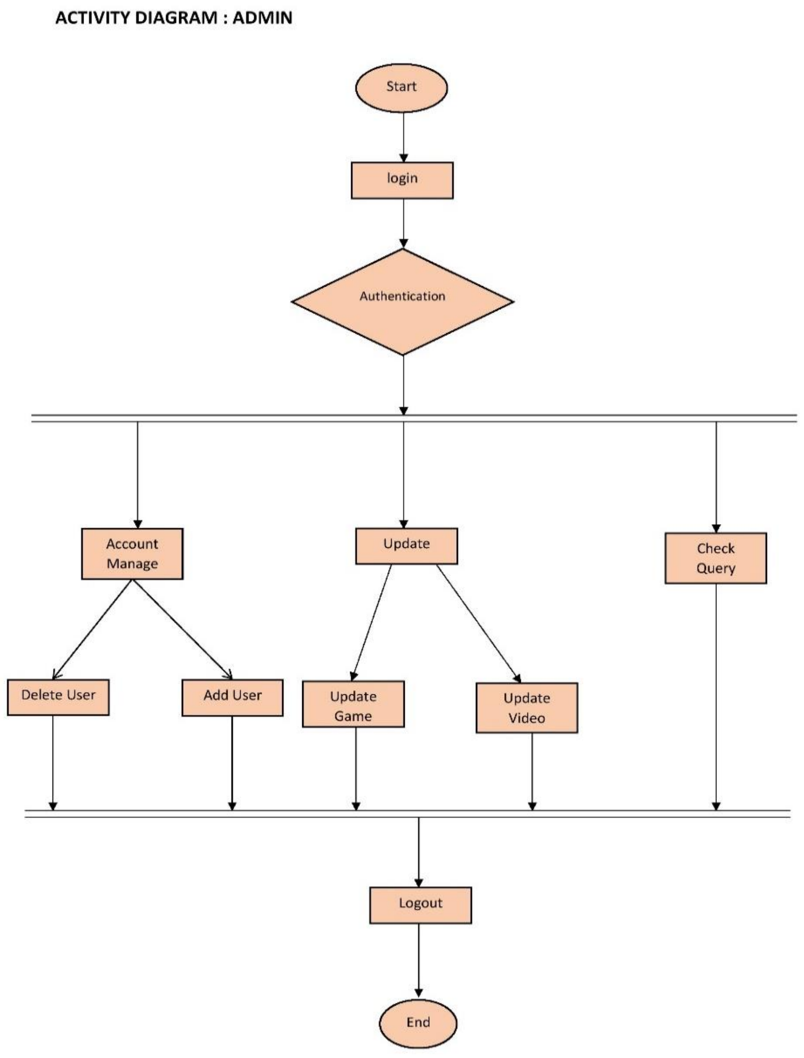


Fig 4.4.1 Activity Chart of Retro Gaming and Video Streaming Website

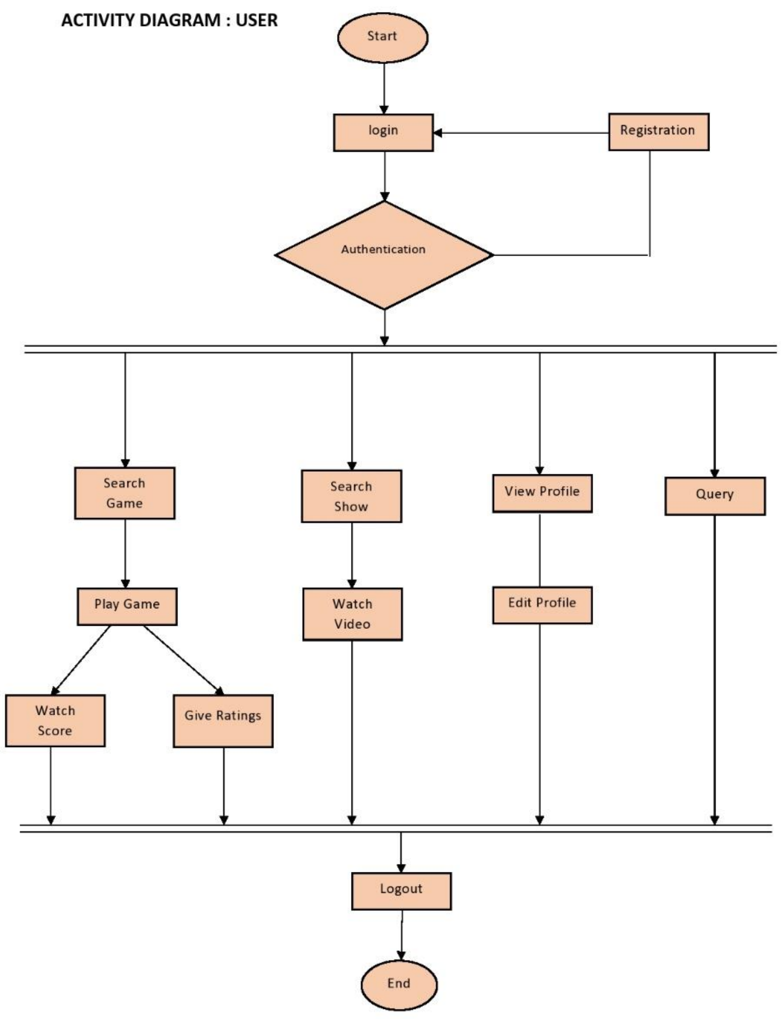


Fig 4.4.2 Activity Chart of Retro Gaming and Video Streaming Website

4.5 Class Diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Purpose of Class Diagrams:

- Shows static structure of classifiers in a system
- Diagram provides a basic notation for other structure diagrams prescribed by UML
- Helpful for developers and other team members too
- Business Analysts can use class diagrams to model systems from a business perspective

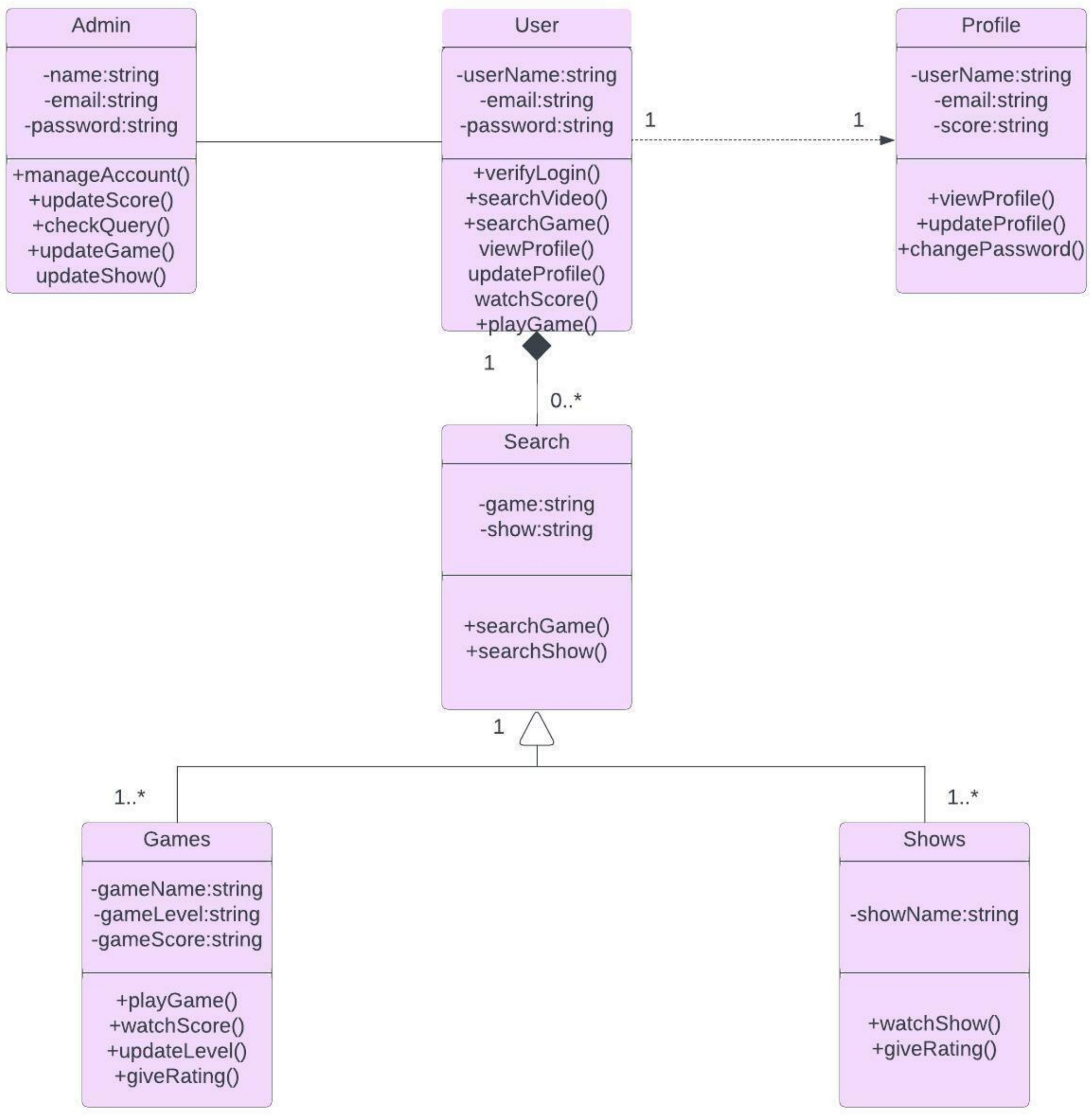


Fig 4.5.1 Class Diagram of Retro Gaming and Video Streaming Website

4.6 Sequence Diagram:

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

Sequence Diagrams captures:

- the interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)

- high-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams)

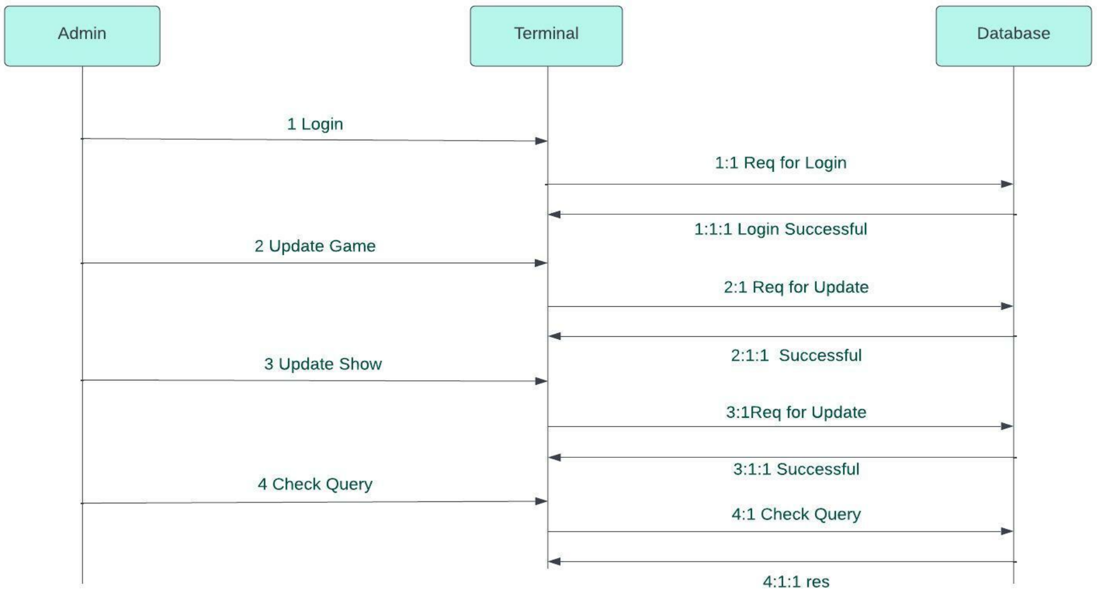


Fig 4.6.1 Admin: Sequence Diagram of Retro Gaming and Video Streaming Website

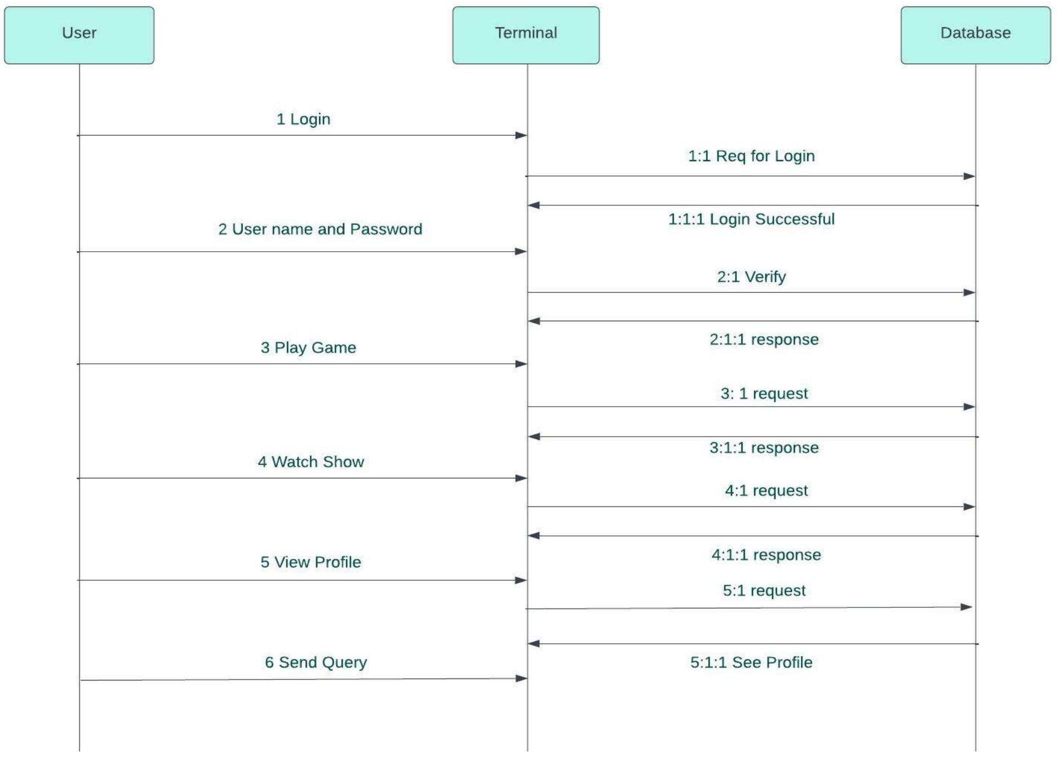
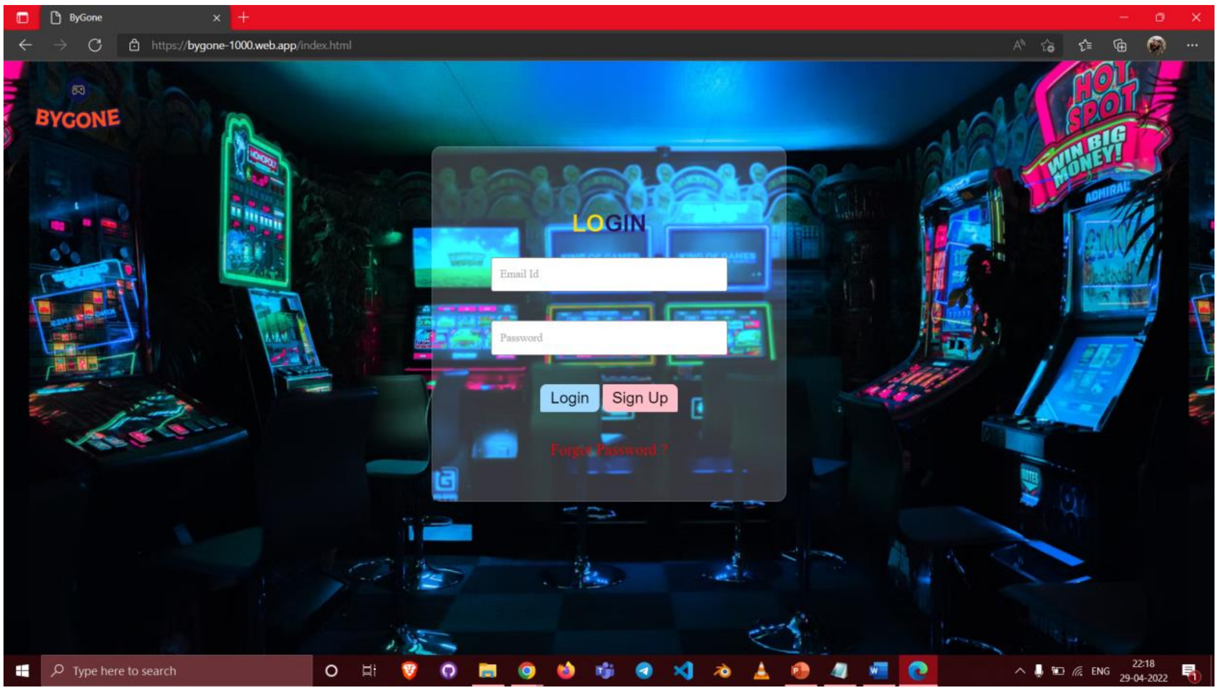


Fig 4.6.2 User: Sequence Diagram of Retro Gaming and Video Streaming Website

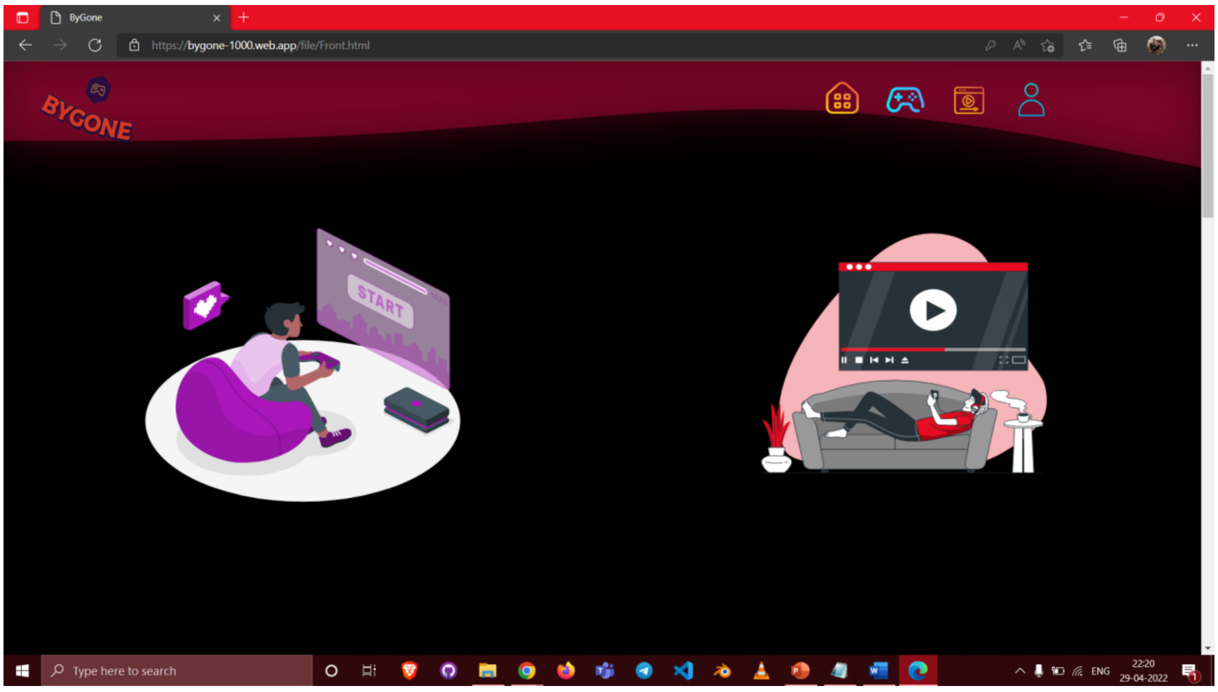
5. Implementation:

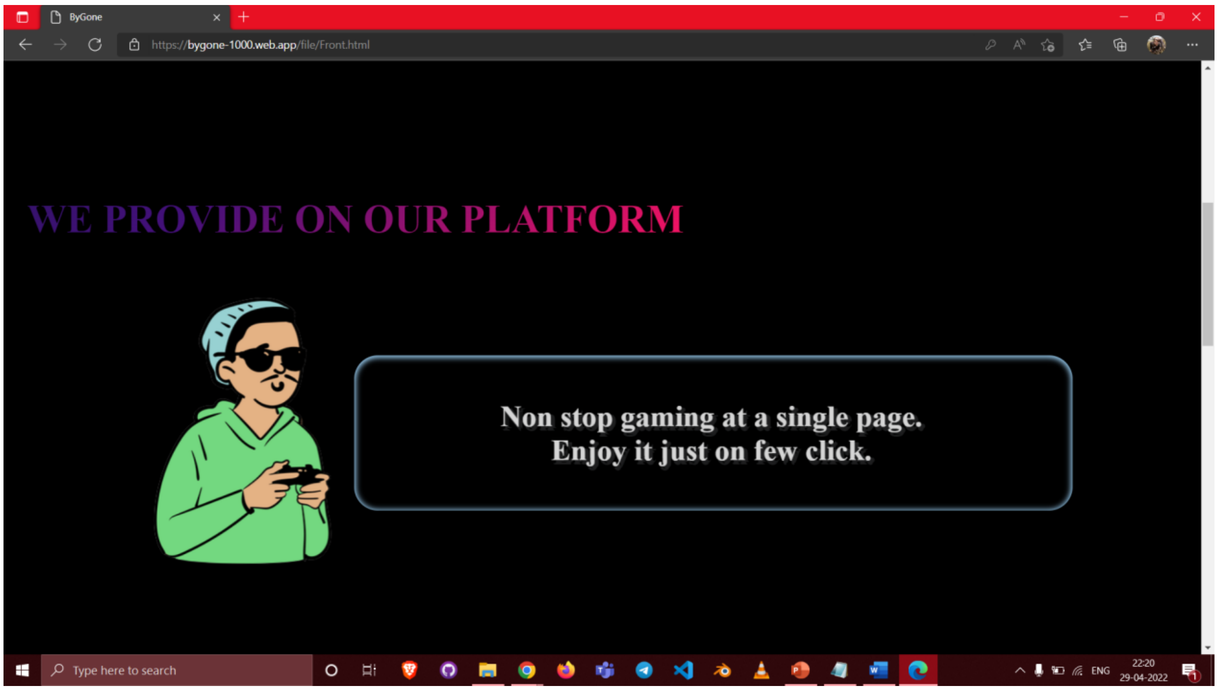
5.1 Screenshots:

Login-Page



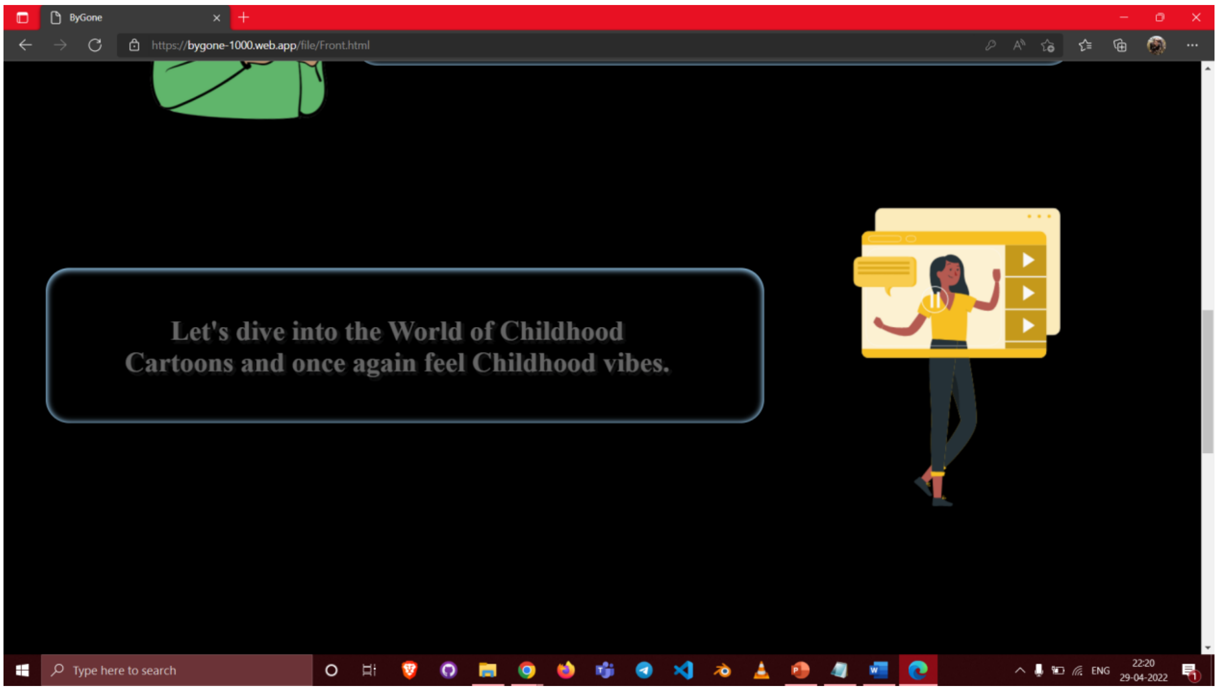
Home-Page



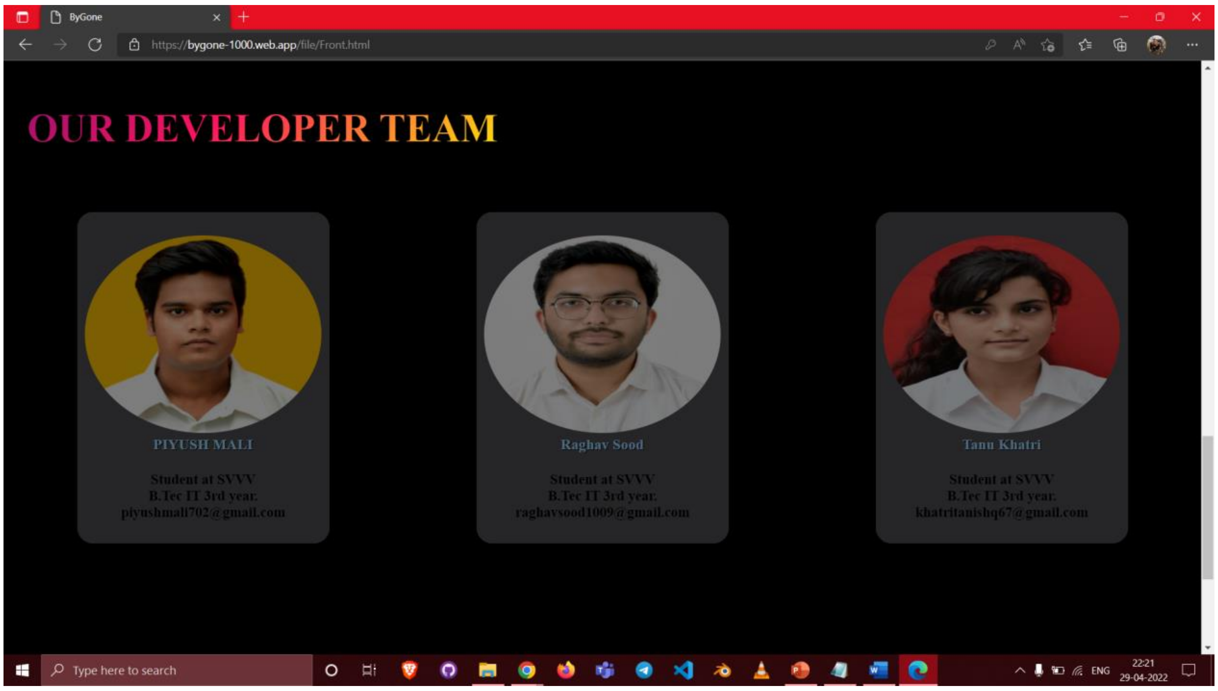


WE PROVIDE ON OUR PLATFORM

Non stop gaming at a single page.
Enjoy it just on few click.



Let's dive into the World of Childhood
Cartoons and once again feel Childhood vibes.



OUR DEVELOPER TEAM

PIYUSH MALI

Student at SVVV
B.Tec IT 3rd year.
piyushmali702@gmail.com

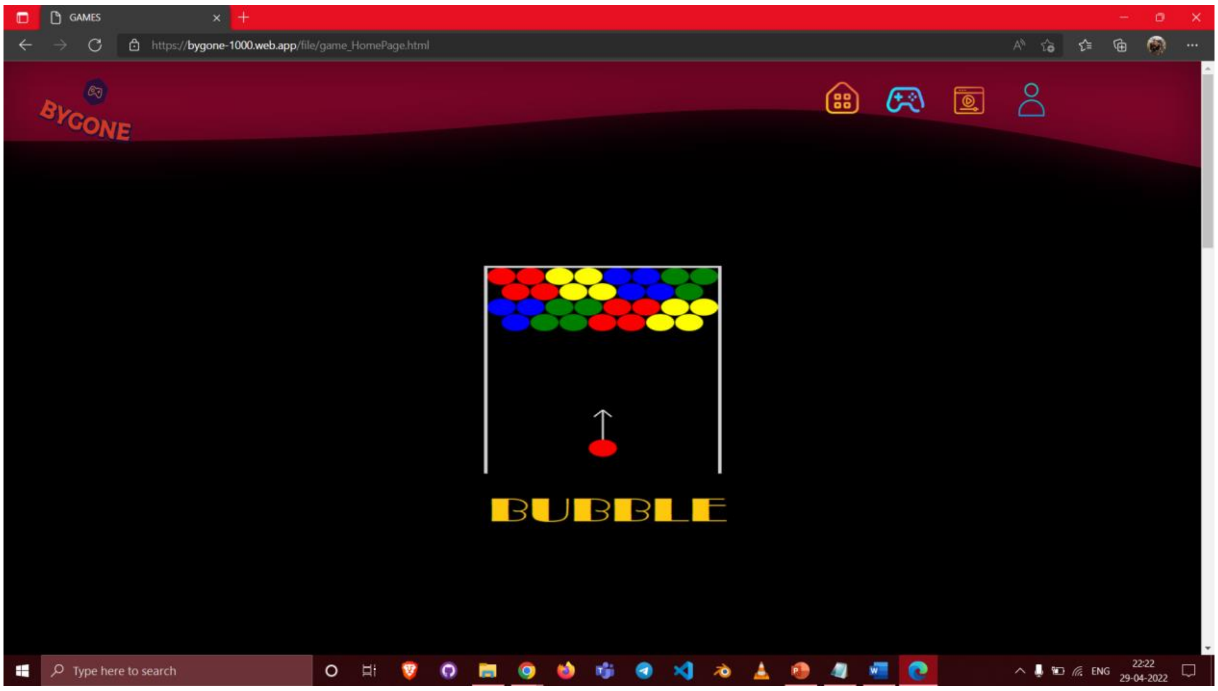
Raghav Sood

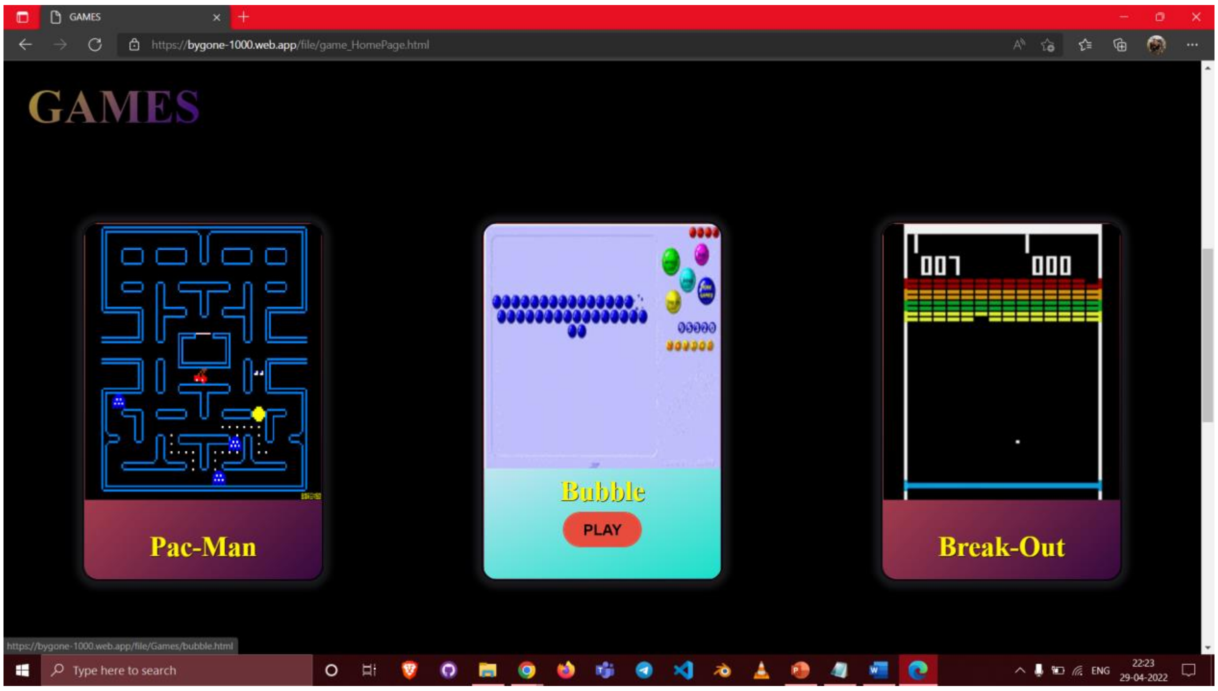
Student at SVVV
B.Tec IT 3rd year.
raghavsood1009@gmail.com

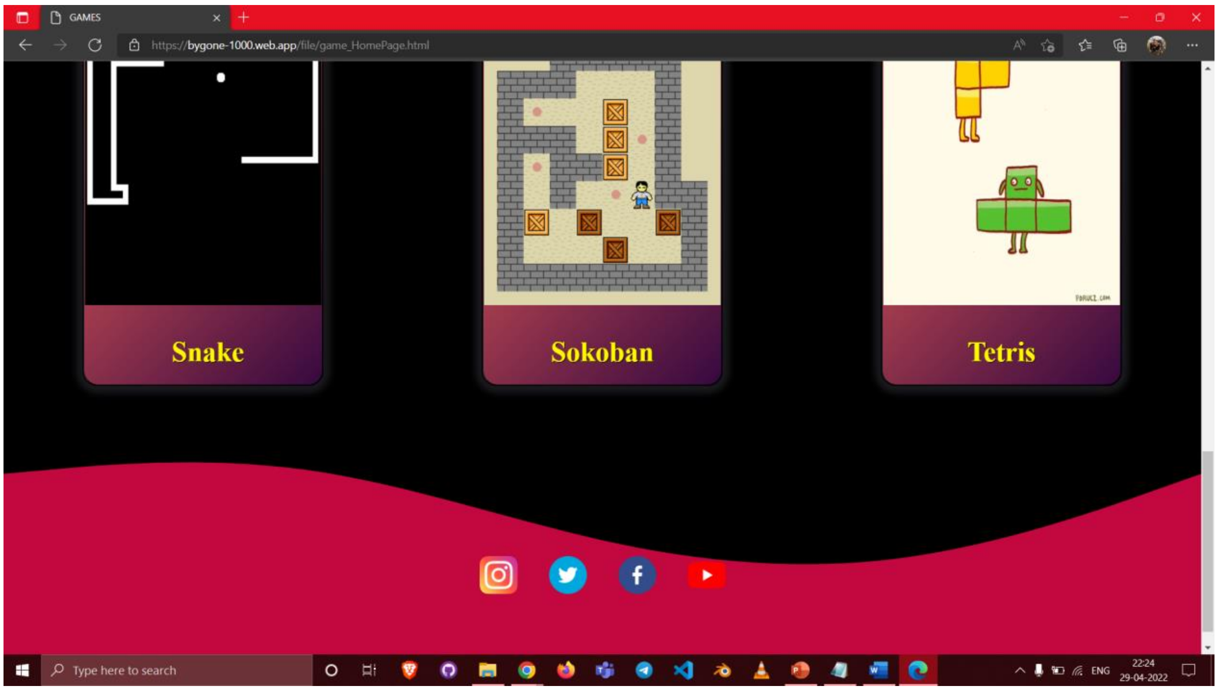
Tanu Khatri

Student at SVVV
B.Tec IT 3rd year.
khatritanishq67@gmail.com

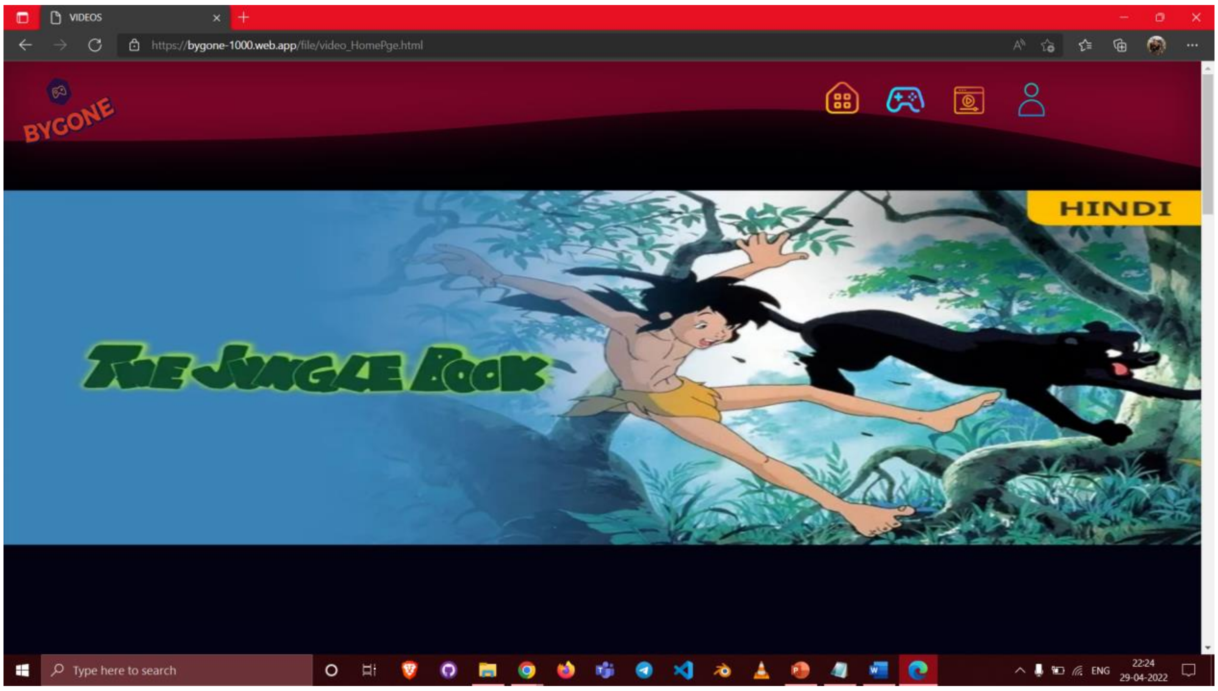
Game-Page

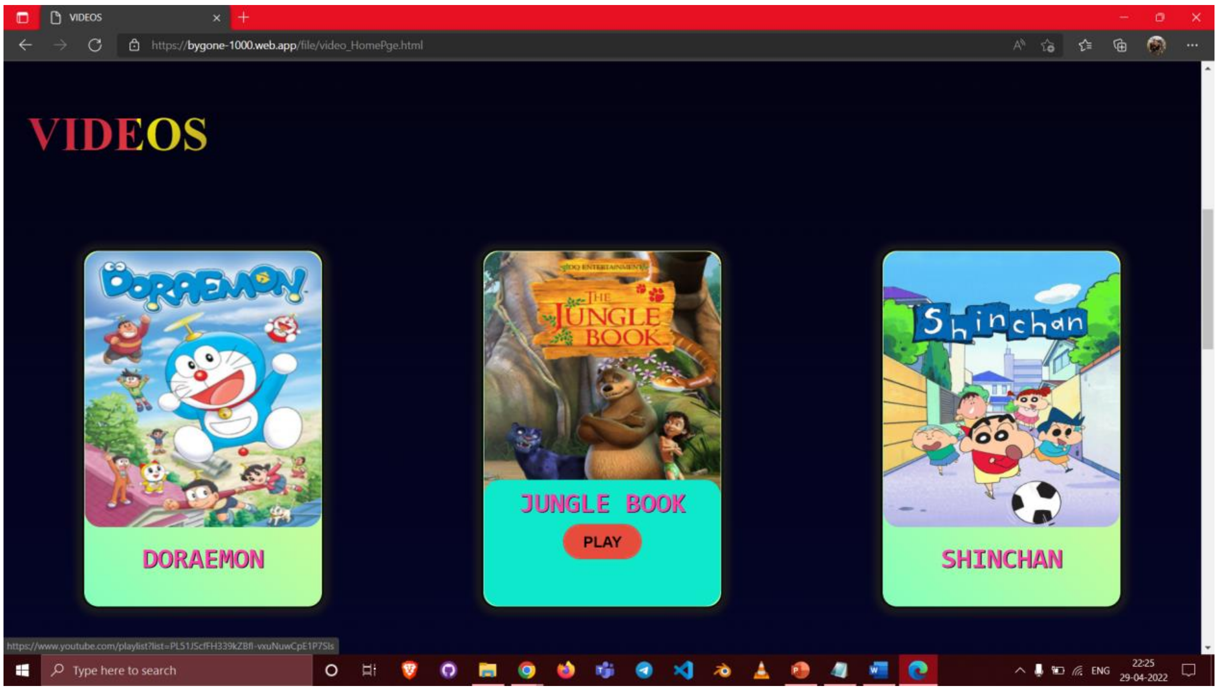


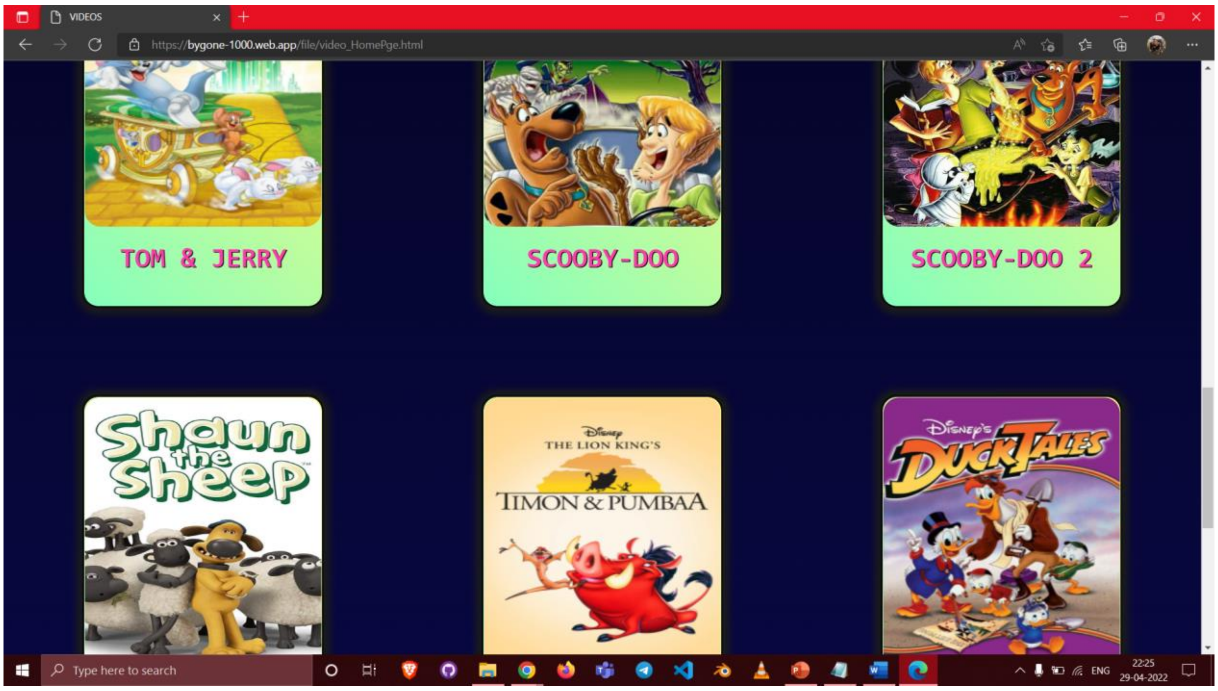

GAMES
https://bygone-1000.web.app/file/game_HomePage.html
GAMES
Pac-Man
Bubble
PLAY
Break-Out
https://bygone-1000.web.app/file/Games/bubble.html
Type here to search
22:23
29-04-2022

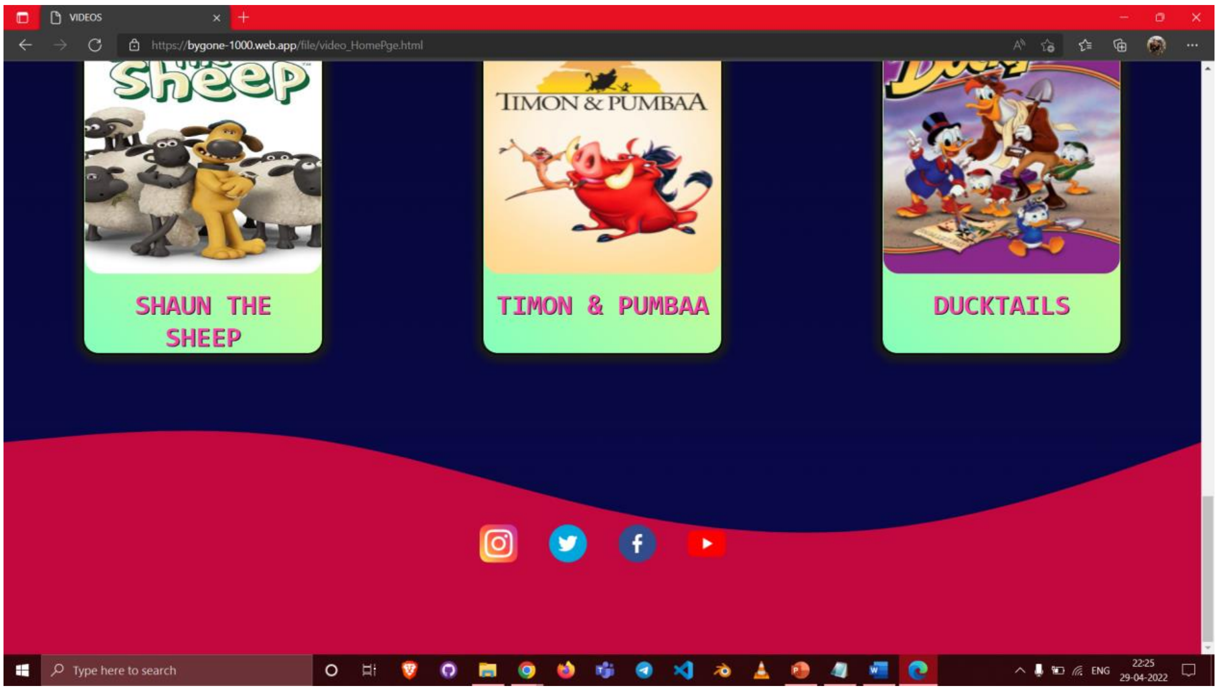

GAMES
https://bygone-1000.web.app/file/game_HomePage.html
Snake
Sokoban
Tetris
Type here to search
22:24
29-04-2022

Video-Page

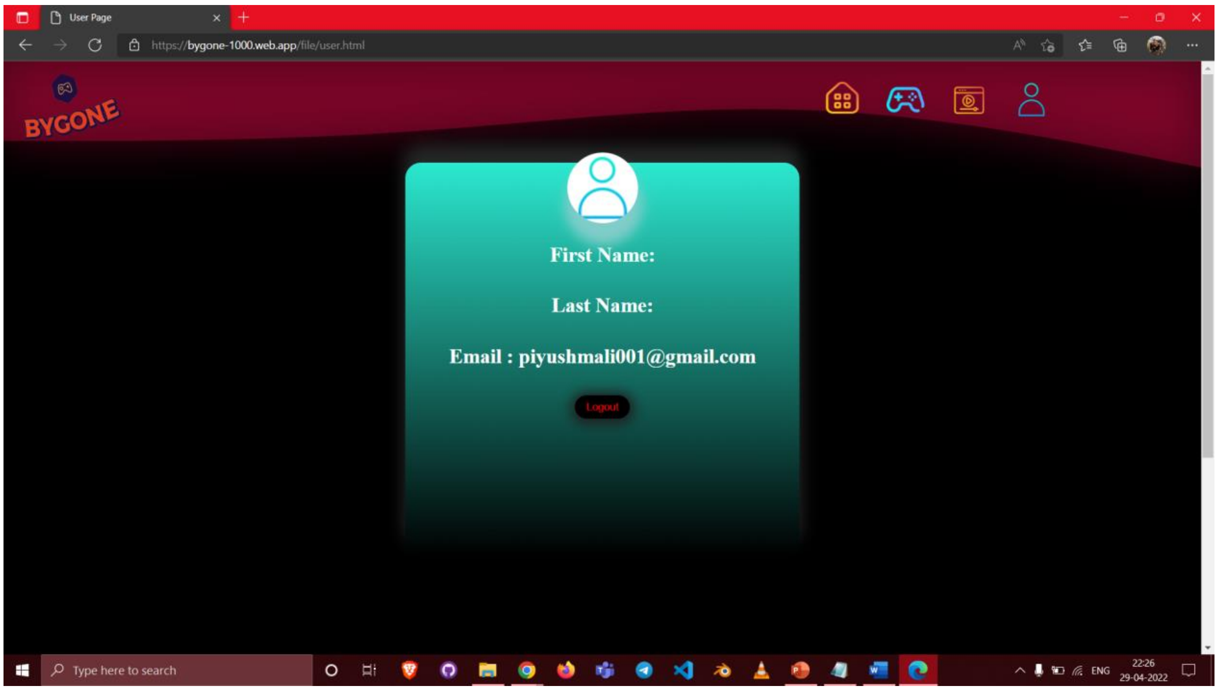

VIDEOS
https://bygone-1000.web.app/file/video_HomePge.html
BYGONE
HINDI
THE JUNGLE BOOK
Type here to search
ENG
22:24
29-04-2022


VIDEOS
https://bygone-1000.web.app/file/video_HomePge.html
VIDEOS
DORAEMON
JUNGLE BOOK
PLAY
SHINCHAN
https://www.youtube.com/playlist?list=PL51JScfFH339kZ8fl-vxuNuwCpE1P7SIs
Type here to search
ENG
22:25
29-04-2022


VIDEOS
https://bygone-1000.web.app/file/video_HomePge.html
TOM & JERRY
SCOOBY-DOO
SCOOBY-DOO 2
Type here to search


VIDEOS
https://bygone-1000.web.app/file/video_HomePge.html
SHAUN THE SHEEP
TIMON & PUMBAA
DUCKTAILS
Type here to search

User-Page

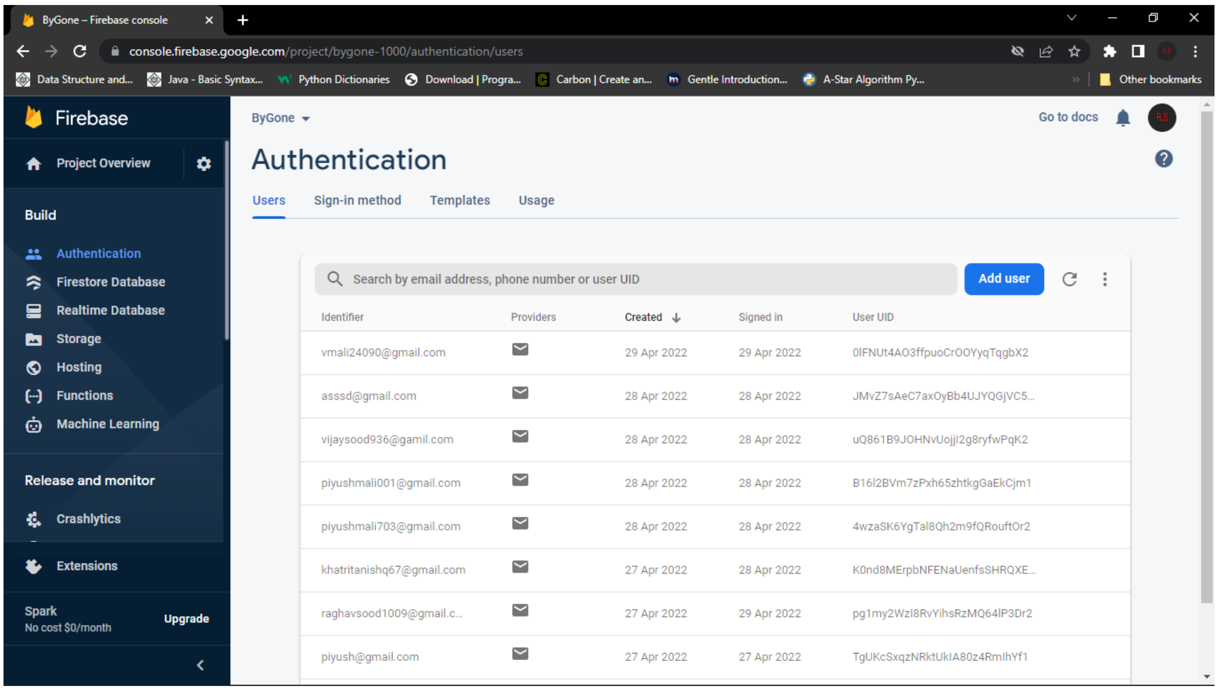


First Name:

Last Name:

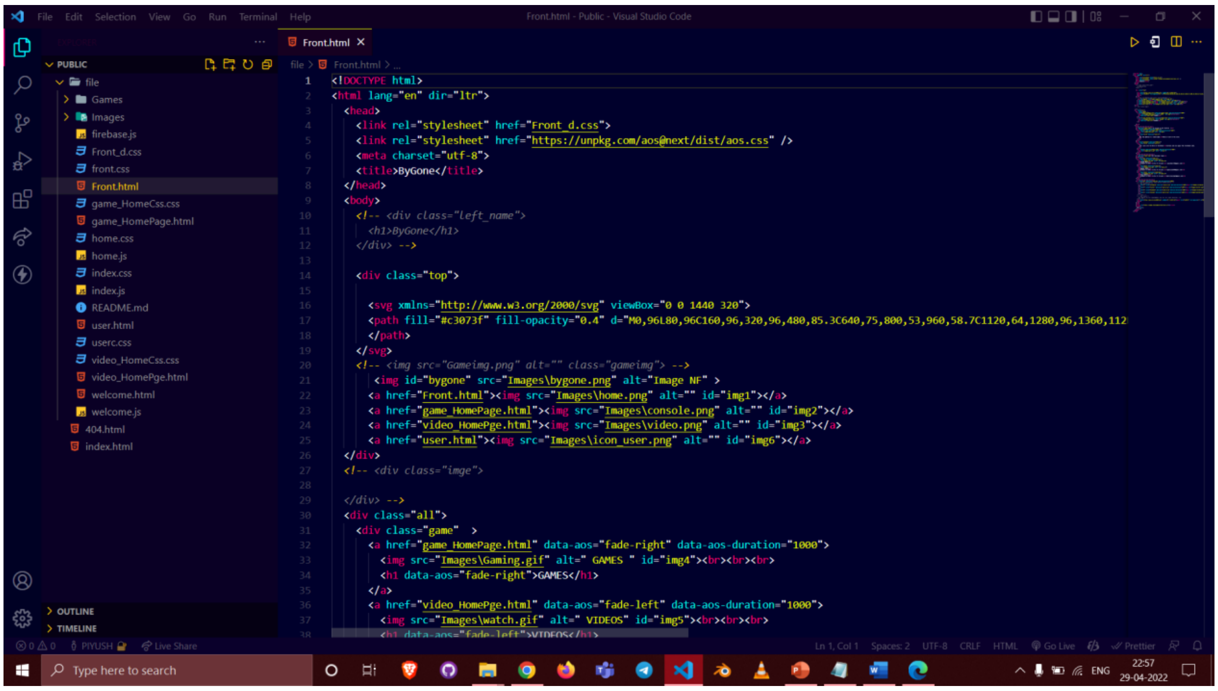
Email : piyushmali001@gmail.com

Firebase

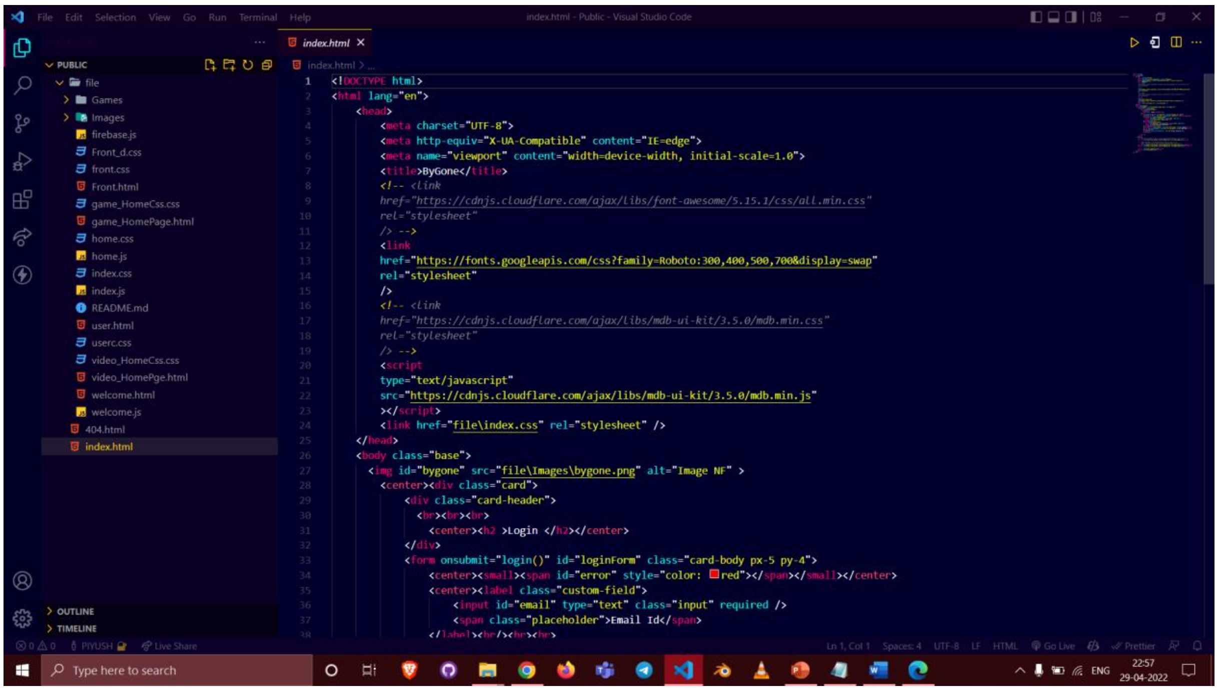


Identifier	Providers	Created	Signed in	User UID
vmali24090@gmail.com	✉	29 Apr 2022	29 Apr 2022	0lFNUt4AO3ffpuoCrOOYyqTqgbX2
asssd@gmail.com	✉	28 Apr 2022	28 Apr 2022	JMvZ7sAeC7axOyBb4UJYQGjVC5...
vijaysood936@gamil.com	✉	28 Apr 2022	28 Apr 2022	uQ861B9JOHNvUojjl2g8ryfwPqK2
piyushmali001@gmail.com	✉	28 Apr 2022	28 Apr 2022	B16l2BVm7zPxh65zhtkgGaEkCjm1
piyushmali703@gmail.com	✉	28 Apr 2022	28 Apr 2022	4wzaSK6YgTal8Qh2m9fQRouftOr2
khatritanishq67@gmail.com	✉	27 Apr 2022	28 Apr 2022	K0nd8MErpbNFENaUenfsSHRQXE...
raghavsood1009@gmail.c...	✉	27 Apr 2022	29 Apr 2022	pg1my2WzI8RvYihsRzMQ64lP3Dr2
piyush@gmail.com	✉	27 Apr 2022	27 Apr 2022	TgUKcSxqzNRktUkIA80z4RmlhYf1

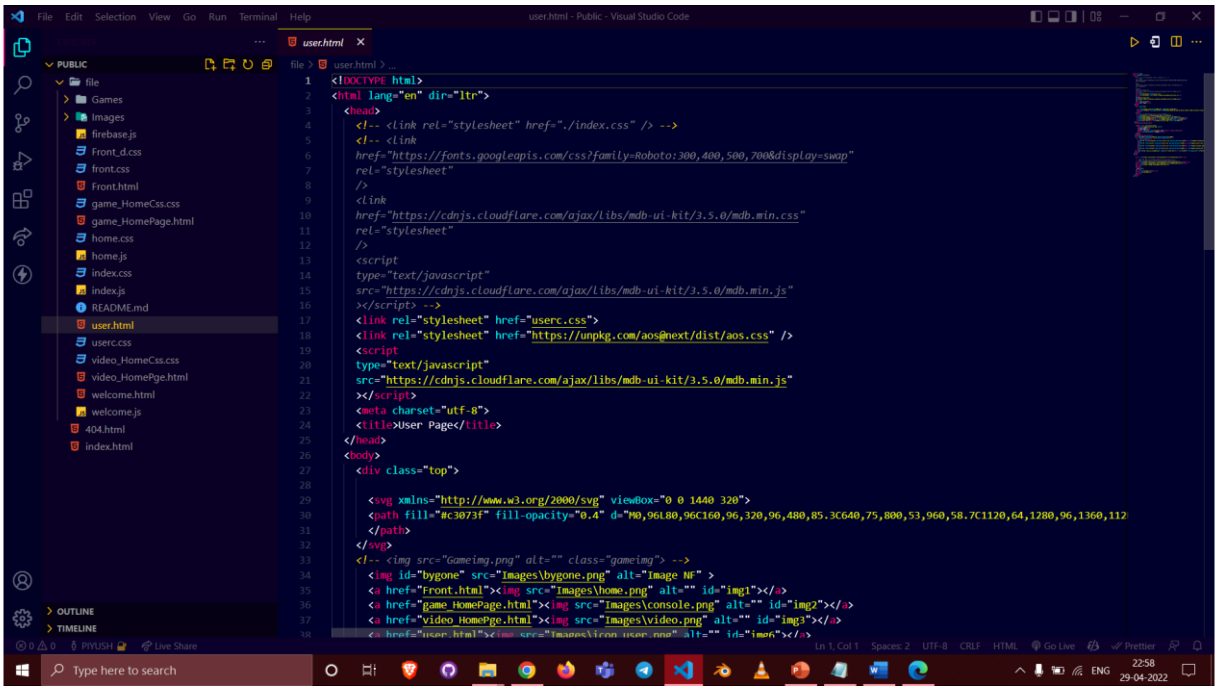
5.2 Coding:



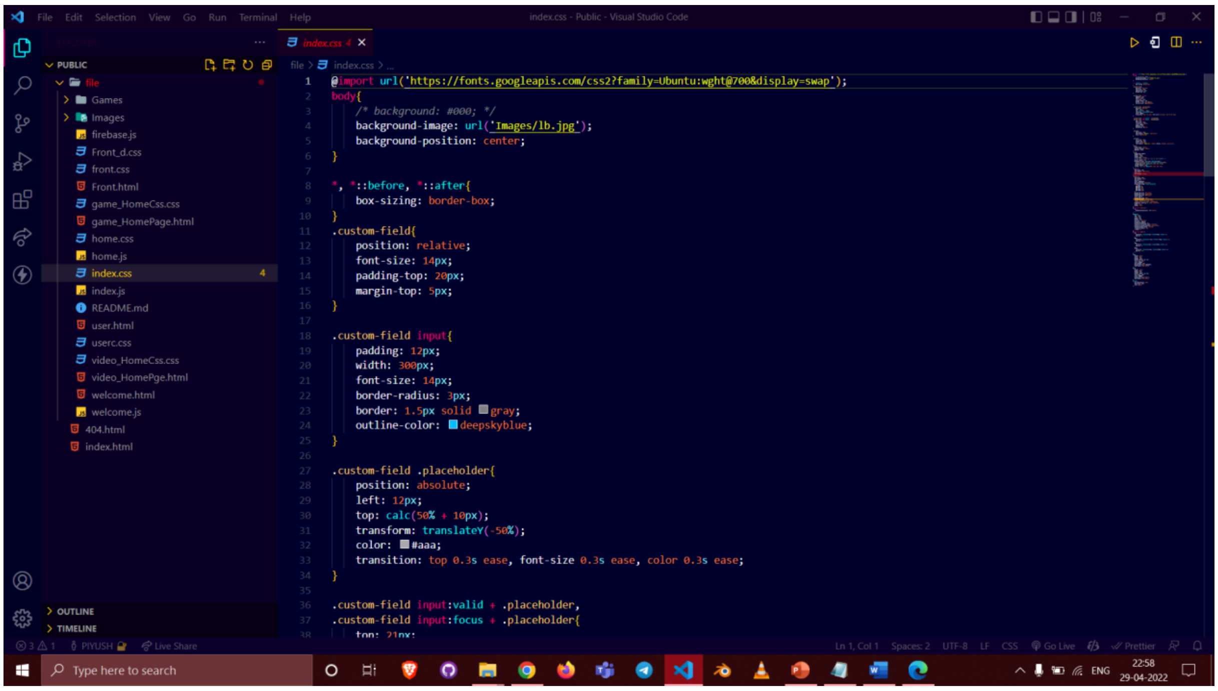
```html
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <link rel="stylesheet" href="Front_d.css">
    <link rel="stylesheet" href="https://unpkg.com/aos@next/dist/aos.css" />
    <meta charset="utf-8">
    <title>ByGone</title>
  </head>
  <body>
    <!-- <div class="left_name">
      <h1>ByGone</h1>
    </div> -->

    <div class="top">

      <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1440 320">
      <path fill="#c3073f" fill-opacity="0.4" d="M0,96L80,96C160,96,320,96,480,85.3C640,75,800,53,960,58.7C1120,64,1280,96,1360,112
      </path>
    </svg>
    <!-- <img src="Gameimg.png" alt="" class="gameimg"> -->
       <img id="bygone" src="Images\bygone.png" alt="Image NF" >
      <a href="Front.html"><img src="Images\home.png" alt="" id="img1"></a>
      <a href="game_HomePage.html"><img src="Images\console.png" alt="" id="img2"></a>
      <a href="video_HomePge.html"><img src="Images\video.png" alt="" id="img3"></a>
      <a href="user.html"><img src="Images\icon_user.png" alt="" id="img6"></a>
  </div>
  <!-- <div class="imge">

  </div> -->
  <div class="all">
    <div class="game"  >
      <a href="game_HomePage.html" data-aos="fade-right" data-aos-duration="1000">
        <img src="Images\Gaming.gif" alt=" GAMES " id="img4"><br><br><br>
        <h1 data-aos="fade-right">GAMES</h1>
      </a>
      <a href="video_HomePge.html" data-aos="fade-left" data-aos-duration="1000">
        <img src="Images\watch.gif" alt=" VIDEOS" id="img5"><br><br><br>
        <h1 data-aos="fade-left">VIDEOS</h1>
```



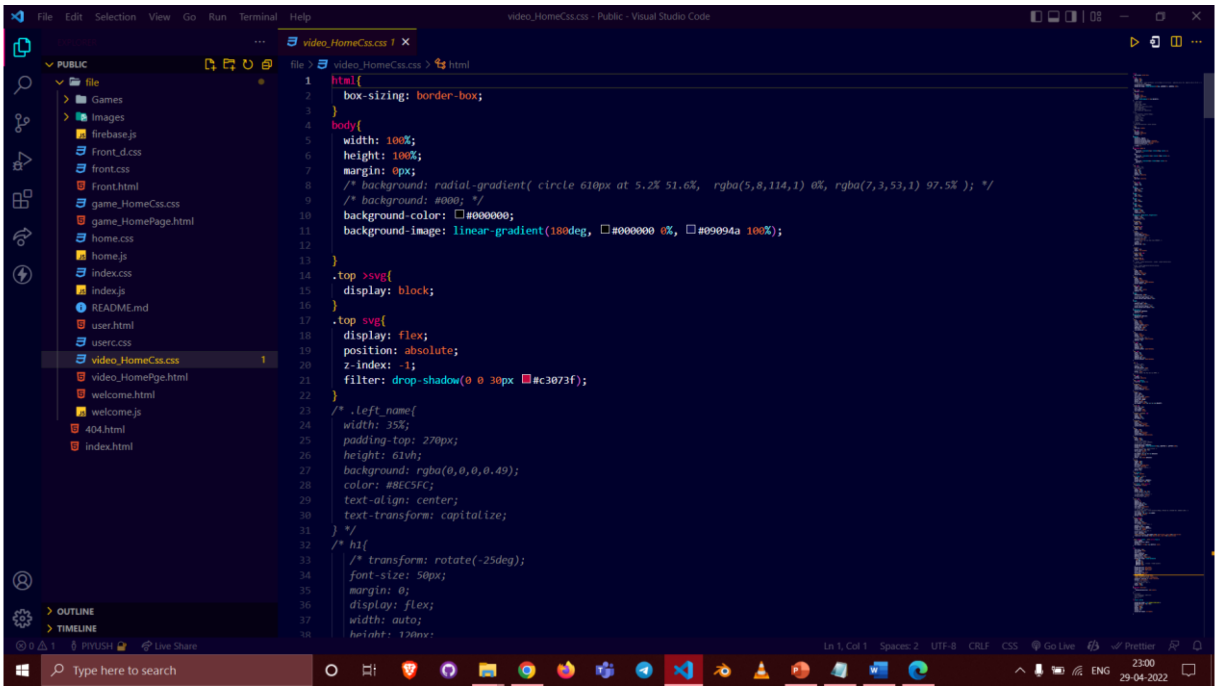
```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>ByGone</title>
        <!-- <link
        href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.1/css/all.min.css"
        rel="stylesheet"
        /> -->
        <link
        href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap"
        rel="stylesheet"
        />
        <!-- <link
        href="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.5.0/mdb.min.css"
        rel="stylesheet"
        /> -->
        <script
        type="text/javascript"
        src="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.5.0/mdb.min.js"
        ></script>
        <link href="file\index.css" rel="stylesheet" />
    </head>
    <body class="base">
      <img id="bygone" src="file\Images\bygone.png" alt="Image NF" >
        <center><div class="card">
            <div class="card-header">
              <br><br><br>
                <center><h2 >Login </h2></center>
            </div>
            <form onsubmit="login()" id="loginForm" class="card-body px-5 py-4">
                <center><small><span id="error" style="color: 🟥red"></span></small></center>
                <center><label class="custom-field">
                    <input id="email" type="text" class="input" required />
                    <span class="placeholder">Email Id</span>
                </label><br/><br><br>
```



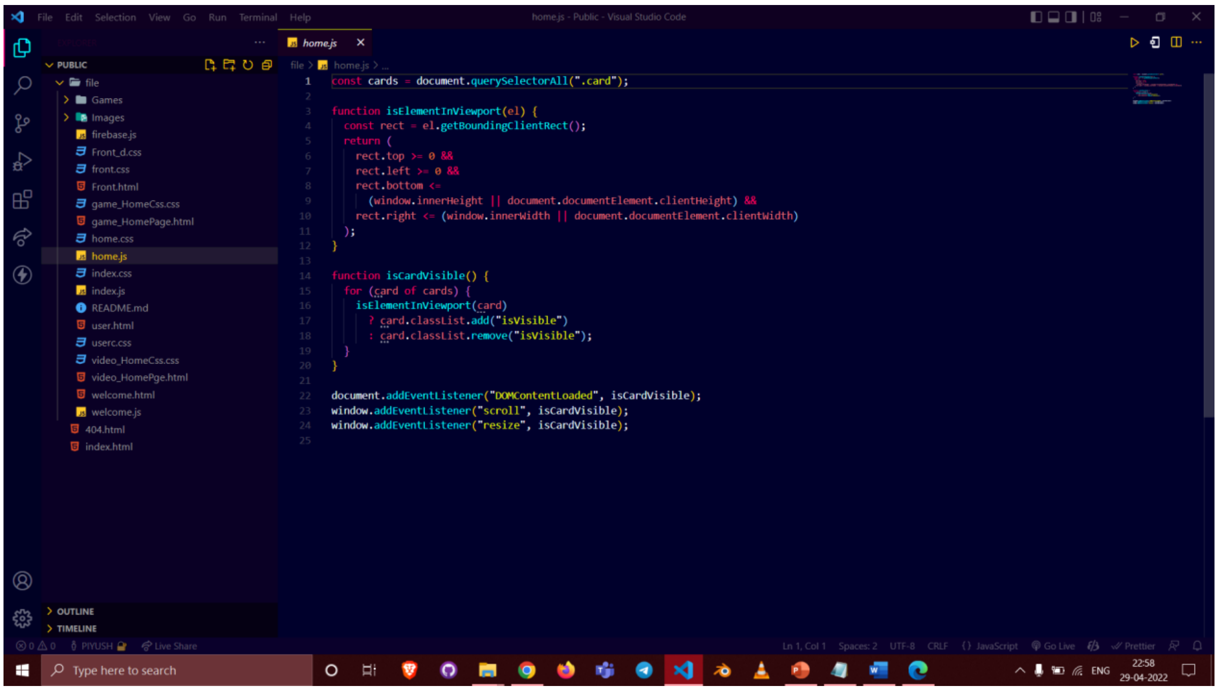
```html
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <!-- <link rel="stylesheet" href="./index.css" /> -->
    <!-- <link
    href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap"
    rel="stylesheet"
    />
    <link
    href="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.5.0/mdb.min.css"
    rel="stylesheet"
    />
    <script
    type="text/javascript"
    src="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.5.0/mdb.min.js"
    ></script> -->
    <link rel="stylesheet" href="userc.css">
    <link rel="stylesheet" href="https://unpkg.com/aos@next/dist/aos.css" />
    <script
    type="text/javascript"
    src="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.5.0/mdb.min.js"
    ></script>
    <meta charset="utf-8">
    <title>User Page</title>
  </head>
  <body>
    <div class="top">

      <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1440 320">
      <path fill="#c3073f" fill-opacity="0.4" d="M0,96L80,96C160,96,320,96,480,85.3C640,75,800,53,960,58.7C1120,64,1280,96,1360,112
      </path>
    </svg>
    <!-- <img src="Gameimg.png" alt="" class="gameimg"> -->
      <img id="bygone" src="Images\bygone.png" alt="Image NF" >
      <a href="Front.html"><img src="Images\home.png" alt="" id="img1"></a>
      <a href="game_HomePage.html"><img src="Images\console.png" alt="" id="img2"></a>
      <a href="video_HomePge.html"><img src="Images\video.png" alt="" id="img3"></a>
      <a href="user.html"><img src="Images\icon_user.png" alt="" id="img6"></a>
```



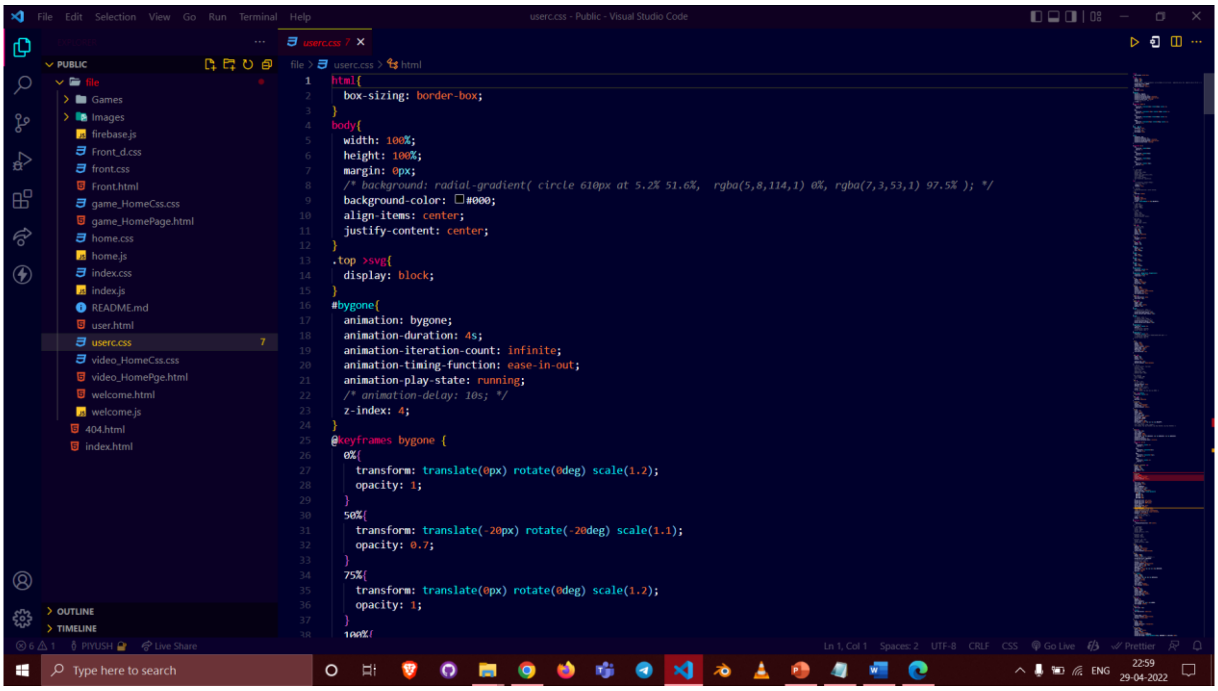
```css
@import url('https://fonts.googleapis.com/css2?family=Ubuntu:wght@700&display=swap');
body{
    /* background: #000; */
    background-image: url('Images/lb.jpg');
    background-position: center;
}

*, *::before, *::after{
    box-sizing: border-box;
}
.custom-field{
    position: relative;
    font-size: 14px;
    padding-top: 20px;
    margin-top: 5px;
}

.custom-field input{
    padding: 12px;
    width: 300px;
    font-size: 14px;
    border-radius: 3px;
    border: 1.5px solid gray;
    outline-color: deepskyblue;
}

.custom-field .placeholder{
    position: absolute;
    left: 12px;
    top: calc(50% + 10px);
    transform: translateY(-50%);
    color: #aaa;
    transition: top 0.3s ease, font-size 0.3s ease, color 0.3s ease;
}

.custom-field input:valid + .placeholder,
.custom-field input:focus + .placeholder{
    top: 21px;
```



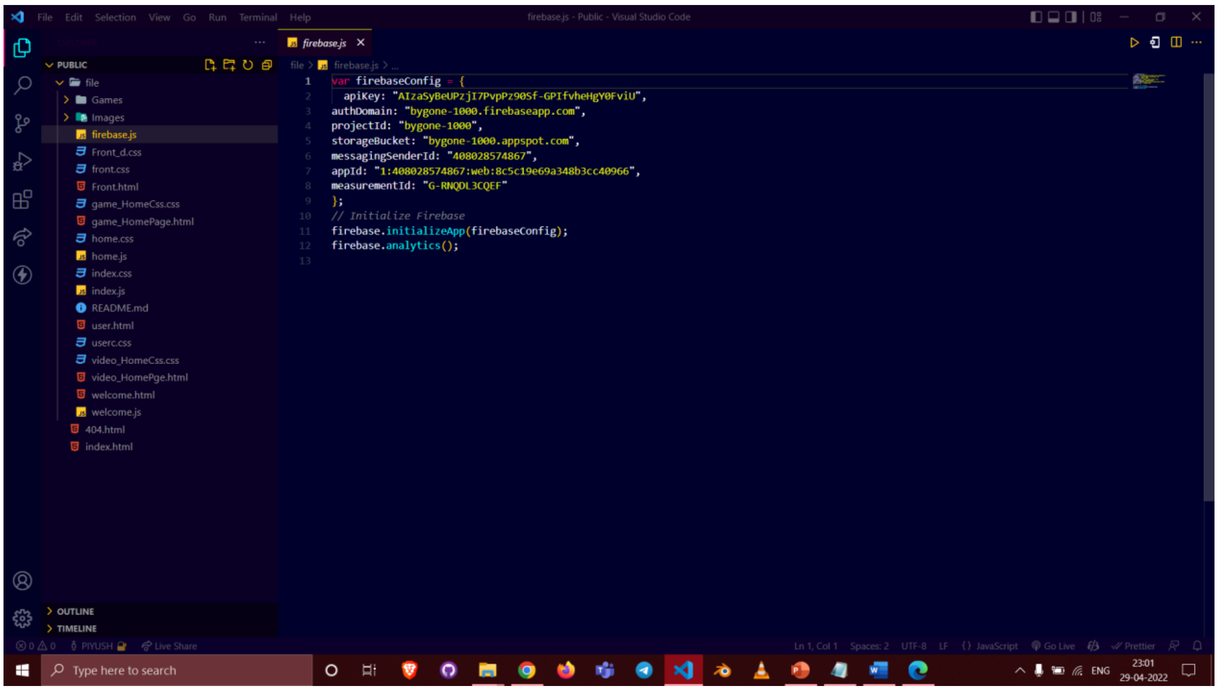
```css
html{
  box-sizing: border-box;
}
body{
  width: 100%;
  height: 100%;
  margin: 0px;
  /* background: radial-gradient( circle 610px at 5.2% 51.6%,  rgba(5,8,114,1) 0%, rgba(7,3,53,1) 97.5% ); */
  /* background: #000; */
  background-color: #000000;
  background-image: linear-gradient(180deg, #000000 0%, #09094a 100%);

}
.top >svg{
  display: block;
}
.top svg{
  display: flex;
  position: absolute;
  z-index: -1;
  filter: drop-shadow(0 0 30px #c3073f);
}
/* .left_name{
  width: 35%;
  padding-top: 270px;
  height: 61vh;
  background: rgba(0,0,0,0.49);
  color: #8EC5FC;
  text-align: center;
  text-transform: capitalize;
} */
/* h1{
   /* transform: rotate(-25deg);
   font-size: 50px;
   margin: 0;
   display: flex;
   width: auto;
   height: 120px;
```



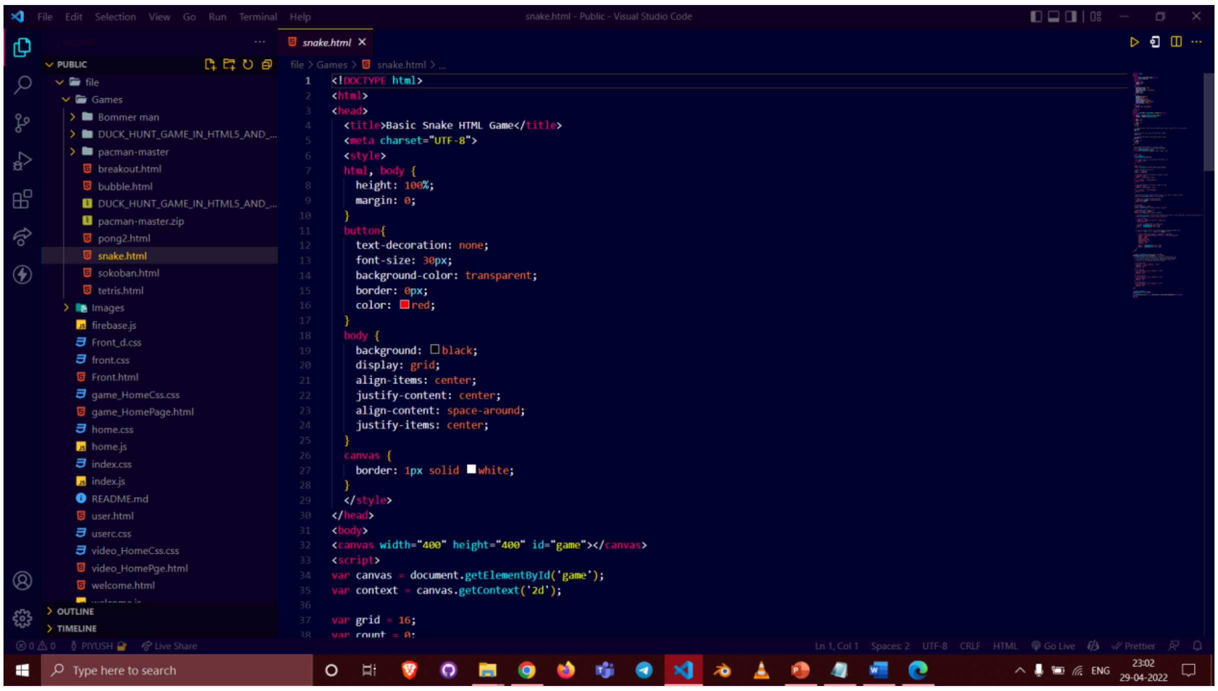
```javascript
const cards = document.querySelectorAll(".card");

function isElementInViewport(el) {
  const rect = el.getBoundingClientRect();
  return (
    rect.top >= 0 &&
    rect.left >= 0 &&
    rect.bottom <=
      (window.innerHeight || document.documentElement.clientHeight) &&
    rect.right <= (window.innerWidth || document.documentElement.clientWidth)
  );
}

function isCardVisible() {
  for (card of cards) {
    isElementInViewport(card)
      ? card.classList.add("isVisible")
      : card.classList.remove("isVisible");
  }
}

document.addEventListener("DOMContentLoaded", isCardVisible);
window.addEventListener("scroll", isCardVisible);
window.addEventListener("resize", isCardVisible);
```



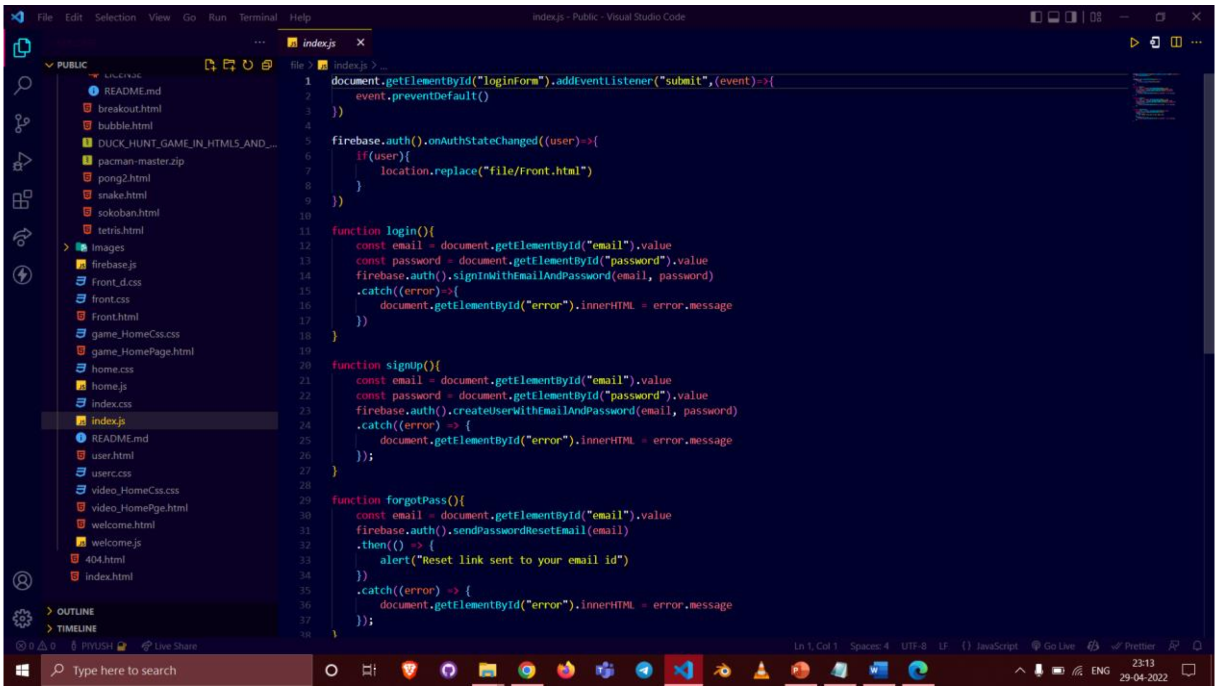
```css
html{
  box-sizing: border-box;
}
body{
  width: 100%;
  height: 100%;
  margin: 0px;
  /* background: radial-gradient( circle 610px at 5.2% 51.6%,  rgba(5,8,114,1) 0%, rgba(7,3,53,1) 97.5% ); */
  background-color: #000;
  align-items: center;
  justify-content: center;
}
.top >svg{
  display: block;
}
#bygone{
  animation: bygone;
  animation-duration: 4s;
  animation-iteration-count: infinite;
  animation-timing-function: ease-in-out;
  animation-play-state: running;
  /* animation-delay: 10s; */
  z-index: 4;
}
@keyframes bygone {
  0%{
    transform: translate(0px) rotate(0deg) scale(1.2);
    opacity: 1;
  }
  50%{
    transform: translate(-20px) rotate(-20deg) scale(1.1);
    opacity: 0.7;
  }
  75%{
    transform: translate(0px) rotate(0deg) scale(1.2);
    opacity: 1;
  }
  100%{
```



```javascript
var firebaseConfig = {
  apiKey: "AIzaSyBeUPzjI7PvpPz90Sf-GPIfvheHgY0FviU",
authDomain: "bygone-1000.firebaseapp.com",
projectId: "bygone-1000",
storageBucket: "bygone-1000.appspot.com",
messagingSenderId: "408028574867",
appId: "1:408028574867:web:8c5c19e69a348b3cc40966",
measurementId: "G-RNQDL3CQEF"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);
firebase.analytics();
```


snake.html - Public - Visual Studio Code
file > Games > snake.html > ...

```html
<!DOCTYPE html>
<html>
<head>
  <title>Basic Snake HTML Game</title>
  <meta charset="UTF-8">
  <style>
  html, body {
    height: 100%;
    margin: 0;
  }
  button{
    text-decoration: none;
    font-size: 30px;
    background-color: transparent;
    border: 0px;
    color: red;
  }
  body {
    background: black;
    display: grid;
    align-items: center;
    justify-content: center;
    align-content: space-around;
    justify-items: center;
  }
  canvas {
    border: 1px solid white;
  }
  </style>
</head>
<body>
<canvas width="400" height="400" id="game"></canvas>
<script>
var canvas = document.getElementById('game');
var context = canvas.getContext('2d');

var grid = 16;
var count = 0;
```

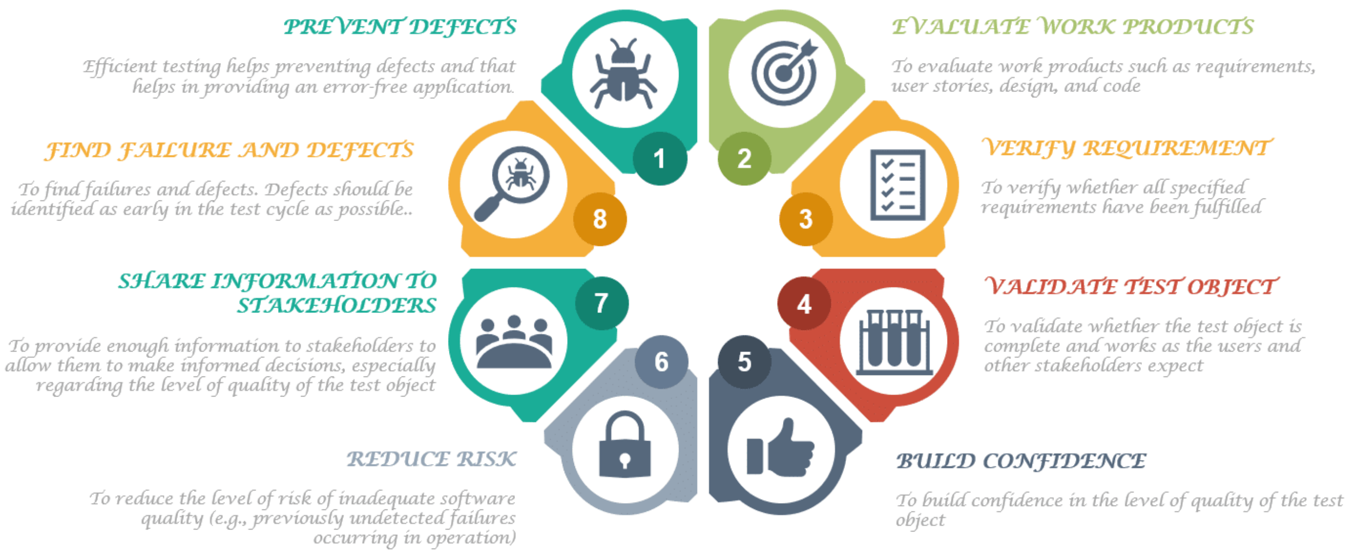


```javascript
document.getElementById("loginForm").addEventListener("submit",(event)=>{
    event.preventDefault()
})

firebase.auth().onAuthStateChanged((user)=>{
    if(user){
        location.replace("file/Front.html")
    }
})

function login(){
    const email = document.getElementById("email").value
    const password = document.getElementById("password").value
    firebase.auth().signInWithEmailAndPassword(email, password)
    .catch((error)=>{
        document.getElementById("error").innerHTML = error.message
    })
}

function signUp(){
    const email = document.getElementById("email").value
    const password = document.getElementById("password").value
    firebase.auth().createUserWithEmailAndPassword(email, password)
    .catch((error) => {
        document.getElementById("error").innerHTML = error.message
    });
}

function forgotPass(){
    const email = document.getElementById("email").value
    firebase.auth().sendPasswordResetEmail(email)
    .then(() => {
        alert("Reset link sent to your email id")
    })
    .catch((error) => {
        document.getElementById("error").innerHTML = error.message
    });
}
```

6. Testing Objective:

Software testing is an activity which aims at evaluating the quality of a software product and also to improve it by identifying defects. Software testing strives to achieve its objectives but has certain limitations. However, adherence to the established objectives ensures effective testing.

Some of the significant objectives of software testing are as follows:


PREVENT DEFECTS
Efficient testing helps preventing defects and that helps in providing an error-free application.
EVALUATE WORK PRODUCTS
To evaluate work products such as requirements, user stories, design, and code
FIND FAILURE AND DEFECTS
To find failures and defects. Defects should be identified as early in the test cycle as possible..
VERIFY REQUIREMENT
To verify whether all specified requirements have been fulfilled
SHARE INFORMATION TO STAKEHOLDERS
To provide enough information to stakeholders to allow them to make informed decisions, especially regarding the level of quality of the test object
VALIDATE TEST OBJECT
To validate whether the test object is complete and works as the users and other stakeholders expect
REDUCE RISK
To reduce the level of risk of inadequate software quality (e.g., previously undetected failures occurring in operation)
BUILD CONFIDENCE
To build confidence in the level of quality of the test object
1
2
3
4
5
6
7
8

To evaluate the work products such as requirements, design, user stories, and code:
The work products such as Requirement document, Design, and User Stories should be verified before the developer picks it up for development. Identifying any ambiguity or contradicting requirements at this stage saves considerable development and test time. The static analysis of the code (reviews, walk-thru, inspection, etc.) happens before the code integrates/is ready for testing. This idea of testing is known as Verification. It is the process of evaluating the product during the development phase of each work product.

To verify the fulfillment of all specified requirements:
This objective reveals the fact that the essential elements of testing should be to fulfill the customer's needs. Testers test the product and ensure the implementation of all the specified requirements have. Developing all the test cases, regardless of the testing technique ensures verification of the functionality for every executed test case. The Tester should also create a requirement traceability matrix (RTM), which will ensure the mapping of all the test cases to requirements. RTM is an effective way to ensure that test cases have got the right requirement coverage.

To validate if the test object is complete and works as per the expectation of the users and the stakeholders:
Testing ensures the implementation of requirements along with the assurance that they work as per the expectation of users. This idea of testing is called Validation. It is the process of

checking the product after development. Validation can be a manual or automation. It usually employs various types of testing techniques, i.e., Black Box, White Box, etc. Generally, testers perform validation, whereas customers can also validate the product as part of User acceptance testing. Every business considers the customer as the king. Thus the customer's satisfaction is a predominant need for any business.

For example, customer satisfaction and loyalty in online shopping and e-commerce environments is a useful indicator for long-term business success.

To build confidence in the quality level of the test object:
One of the critical objectives of software testing is to improve software quality. High-Quality software means a lesser number of defects. In other words, the more efficient the testing process is, the fewer errors you will get in the end product. Which, in turn, will increase the overall quality of the test object. Excellent quality contributes to a significant increase in customer satisfaction as well as lower maintenance costs.

To prevent defects in the software product:
One of the objectives of software testing is to avoid the mistakes in the early stage of the development. Early detection of errors significantly reduces the cost and effort. The prevention of defects involves doing a root cause analysis of the defects found previously and after that, taking specific measures to prevent the occurrence of those types of errors in the future. Efficient testing helps in providing an error-free application. If you prevent defects, it will result in reducing the overall defect count in the product, which further ensures a high-quality product to the customer.

To find defects in the software product:
Another essential objective of software testing is to identify all defects in a product. The main motto of testing is to find maximum defects in a software product while validating whether the program is working as per the user requirements or not. Defects should be identified as early in the test cycle as possible.

E.g., a defect found in the UAT phase will be much costlier to fix than the same defect found in the Sprint testing phase.

To provide sufficient information to stakeholders to allow them to make informed decisions, especially regarding the level of quality of the test object:
The purpose of testing is to provide complete information to the stakeholders about technical or other restrictions, risk factors, ambiguous requirements, etc. It can be in the form of test coverage, testing reports covering details like what is missing, what went wrong. The aim is to be transparent and make stakeholders fully understand the issues affecting quality.

To reduce the level of risk of insufficient software quality:
The possibility of loss is also known as risk. The objective of software testing is to reduce the occurrence of the risk. Each software project is unique and contains a significant number of uncertainties from different perspectives, such as market launch time, budget, the

technology chosen, implementation, or product maintenance. If we do not control these uncertainties, it will impose potential risks not only during the development phases but also during the whole life cycle of the product. So, the primary objective of software testing is to integrate the Risk management process to identify any risk as soon as possible in the development process.

To comply with contractual, legal, or regulatory requirements or standards, and to verify the test object's compliance with such requirements or standards:
This objective ensures that software developed for a specific region must follow the legal rules and regulations of that region. Moreover, the software product must be compatible with the national and international standards of testing. We have ISO/IEC/IEEE 29119 standards that deal with the software testing concept.

E.g., each country has laws specific to accessibility requirements which must be fulfilled to avoid legal implications. The European Union has strict rules on how the Personal Identifiable Information (PII) like social security number etc. should be handled. Failure to adhere to such requirements will lead to failure of the product, no matter how defect-free it has been working!

7. Limitations of Project:

- **Cost-** The major cost of online gaming comes from the charges of internet connection. The internet Service Provider (ISP) may impose large data charges depending on the time the user spends for gaming.
- **Security-** When playing online games, there is always a risk of hacking. Players enter their personal information in online gaming for various reasons. If a hacker manages to hijack this information, they could use it for illegal activities. As a result, the reputation of the player could be destroyed within seconds.
- **Multiple players cannot play-** Multiple players cannot play together in a single game.
- You cannot watch shows while playing the game.
- If you want to play games and watch shows you need to create an account.
- Not all shows and retro games are available.

8. Future Enhancement:

- Adding feature to connect with friends and able to play multi-player-games- In future we will add some features to able you to connect with friends so you can play with them in a game and multiple players can play the game in a single game.
- Chat for in-game- You can chat or communicate while playing the game.
- Taking user feedback and work on it- For more ideas we will take user's feedback and work accordingly. Since this website is for users so, on user's feedback we will work.
- Add More Retro Cartoon Shows in future- In future we will add more retro cartoon shows and games. Most of the retro cartoon shows are available on different platform, some of are not known so we will add more retro cartoon shows in single platform so everyone can watch it. And games also so you can play those games.

9. Conclusion:

The concept of “retro” has a lot to do with the place of birth. It refers to the games and shows of your childhood, and there are some offerings that will awaken a sense of nostalgia in you. Thinking of the game and cartoon shows as a part of a bigger educational process is really in the core mind-set that this project wants to promote. Games can do many things very well, but they certainly cannot do everything at once. Especially not without solid supporting structures around them. Throughout the project and the case studies we built this was true. Although there are many cons, we have concluded as a group that the pros heavily outweigh them, online old gaming helps the development of children, provides a fun and exciting way to pass time and develops teamwork skills, they enhance skills such as technology skills that are required in today’s life which is constantly changing and growing. But we do agree that online gaming does have a downside like everything in life, and that use should be limited and online gamers should be aware of the things that can possibly happen if their time gaming is abused.

10. References:

- https://www.retrogames.cz/?language=EN
- https://www.playretrogames.com/
- https://www.recalbox.com/
- https://disneynow.com/all-shows
- https://www.yidio.com/channel/cartoon-network
- https://techstory.in/the-future-of-retro-gaming/
- https://oldgameshelf.com/
- https://classicreload.com/
- https://playclassic.games/
- https://www.boomerang.com/usonly/
- https://www.wbkidsgo.com/
- https://www.toonamiaftermath.com/