EDUCATION APP
WITH AUGMENTED REALITY
(EduWAR)

The Minor Project report submitted to

Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal

towards partial fulfillment of the Degree of

Bachelor of Engineering

in

Computer Science

Guided by
Prof. Gaurav Bairagi

Submitted by
Mr. Ronit Gurjar

Department of Computer Science and Engineering
Mahakal Institute of Technology & Management, Ujjain (M.P.)
2022-23

MAHAKAL INSTITUTE OF TECHNOLOGY & MANAGEMENT, UJJAIN

RECOMMENDATION

This is to certify that Mr. Ronit Gurjar student of third year B.tech (CSE) in the year 2022-23 of Department of Computer Science and Engineering of this institute has completed their work on "***Education app With Augmented Reality***" for Minor project based on syllabus and has submitted a satisfactory account of their work in this report which is recommended for the partial fulfillment of the degree of Bachelor of Engineering in Computer Science.

Project Guide
CSE Dept.
M.I.T.M. Ujjain

HOD
CSE Dept.
M.I.T.M. Ujjain

Director
M.I.T.M. Ujjain

MAHAKAL INSTITUTE OF TECHNOLOGY & MANAGEMENT, UJJAIN

CERTIFICATE

This is to certify that the Minor Project report entitled "***Education app With Augmented Reality***" submitted by Mr. Ronit Gurjar, student of B.tech V semester, Department of Computer Science and Engineering in the year 2022-23, is a satisfactory account of their work based on syllabus which is accepted in partial fulfillment of degree of Bachelor of Engineering in Computer Science .

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Date **Date**

ACKNOWLEDGEMENT

It gives me immense pleasure to express my deepest sense of gratitude & sincere thanks to my respected guide **Prof. Gaurav Bairagi** (Asst. Professor, CSE) for his valuable guidance and help for completing this work. I would also like to show my sincere thanks to **Prof. Aashish Anjana**, and **Prof. Vaishali Pathak** for their constant support. Their useful suggestions for this work and cooperation are sincerely acknowledged.

I would like to express my sincere thanks to **Dr. J N Vyas**, Director, Mahakal Institute of Technology and Management, Ujjain for giving me this opportunity to undertake this project.

I also wish to express my gratitude to **Prof. Deepali Y Kelkar**, HOD, CSE for her kind hearted support.

I also wish to express my indebtedness to my parents as well as my family member whose blessings and support always helped me to face the challenges ahead.

At the end, I would like to express my sincere thanks to all my friends & others who helped me directly or indirectly during this project work.

Mr. Ronit Gurjar
[0714CS201066]

ABSTRACT

The purpose of this project is to develop an AR-based app to teach engineering students Fundamentals and structure in Computer Architecture and analyze the educational effect. EduWAR is an application which will help in teaching subjects such as Computer Architecture and Organization to the students with the help of Augmented Reality consisting of 3D models and Labels in form of a Puzzle game.

Students will be provided with special cards with image targets printed over them and students have to arrange cards in proper way to complete the structure of computer system. Any random placements would not work as the Aim of the application is to teach students how particular components are assembled in a computer system.

EduWAR will be build using UNITY game engine and Vuforia package which consist of Built-in libraries and classes which helps in creating AR environment for a system.

TABLE OF CONTENTS

FIGURE INDEX

Chapter 1

INTRODUCTION

1.1 Overview

Augmented reality (AR) is the integration of digital information with the user's environment in real time. Unlike virtual reality (VR), which creates a totally artificial environment, AR users experience a real-world environment with generated perceptual information overlaid on top of it.

Augmented reality is used to either visually change natural environments in some way or to provide additional information to users. The primary benefit of AR is that it manages to blend digital and three-dimensional (3D) components with an individual's perception of the real world. AR has a variety of uses, from helping in decision-making to entertainment.

Boeing Computer Services Research employee Thomas Caudell coined the term augmented reality in 1990 to describe how the head-mounted displays that electricians use when assembling complicated wiring harnesses worked. One of the first commercial applications of augmented reality technology was the yellow first down marker that began appearing in televised football games sometime in 1998. Today, Google Glass, smartphone games and heads-up displays (HUDs) in car windshields are the most well-known consumer AR products.

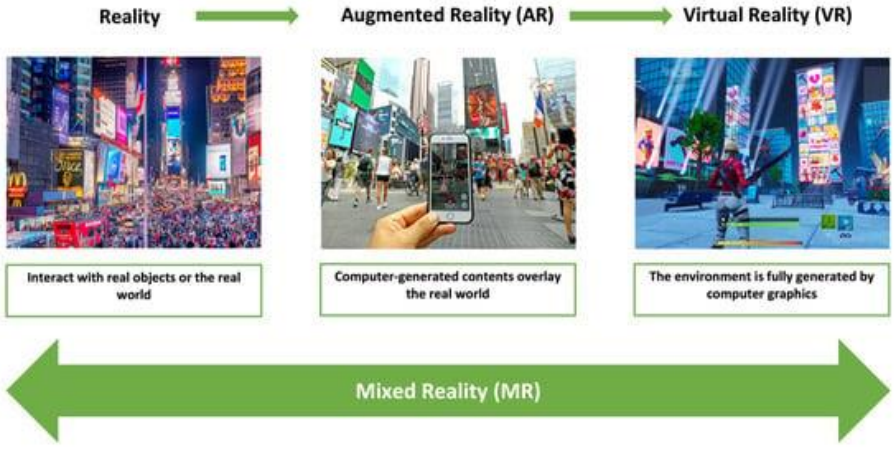


Fig. 1.1: Mixed Reality

1.2 Current System

In higher education, augmented reality is used for a wide range of applications. Faculty use AR platforms to incorporate gamification into curricula and create educational

material. Through AR technology, teachers can materialize abstract concepts to help students visualize and understand challenging subjects. Consider these examples of how universities use augmented reality in higher education.

1) Theatre

With theatres embracing technologies such as the ARShow platform, which allows producers to add AR elements into live performances, university drama departments are incorporating augmented reality into their curricula. For example, one college developed an AR app to visualize stage design and allow virtual walk-throughs before set construction.

2) Science, technology, engineering, and math

AR in higher education is gaining traction in science, technology, engineering, and math departments across the U.S. For example, a hands-on, collaborative lab enables students to use AR technology to operate a chemical plant and experiment with different chemical reactions.

3) Medicine

AR is transforming medical training. It can provide medical students with opportunities to watch live surgeries taking place in real time. AR applications can also help medical students learn about the human anatomy through simulations and models. One innovative app allows surgeons to take a walk-through of a patient's organs before performing a procedure.

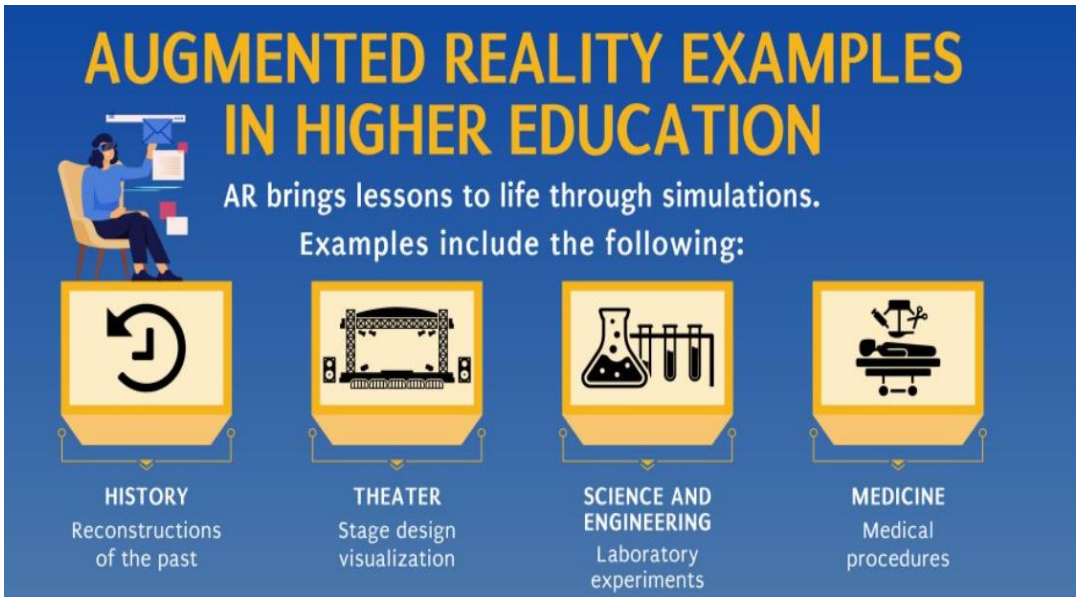


Fig. 1.2: Augmented Reality Example

Below is a list of augmented reality apps for education:

- Human Anatomy Atlas 2021 — 3D models and simulations of male and female anatomy help students and healthcare professionals understand how the human body works. Users can perform virtual dissections, view animations, explore muscle action, and more.

- GeoGebra Augmented Reality — From geometry and algebra to statistics and calculus, this interactive tool supports science, technology, engineering, and mathematics (STEM) education through AR features that allow students to explore shapes and 3D functions, use critical thinking skills, and more.
- Exoplanet — This app, developed by a professional astronomer, provides an interactive catalogue of known planets orbiting stars in the Milky Way.

1.3 Need for a Proposed System

The education system all over the world was only following traditional instructor-led teaching approach. Integrating Augmented Reality in education can enrich and bring in benefits for educational institutions, educators as well as learners enriching the entire learning experience.

Using AR in the classroom can turn an ordinary class into an engaging experience. AR technology provides virtual examples and adds gaming elements to support textbook materials. As a result, classes become more interactive. AR helps students better remember the information they've just learned.

Students with weak imagination sometimes stays behind as they could not understand how a “specific thing” is required for “specific process” resulting in poor grades. The **EduWAR** app focuses mainly on helping the student understand those “things” with the use of well-rendered and well-defined 3D models.

The app functions as a puzzle game in which the students are given some special cards which when seen from the camera shows a Augmented 3D model for a specific card. Each card is labelled based on particular components of computer architecture and same goes for their respective 3D models.

The cards are to be placed in a specific manner to complete the computer system, in case of wrong placements or combinations of cards the system will throw errors with explanation so that the student can understand the next steps, hence learning a Complex subject in a fun way.

The App consists of a ‘Help’ option which consists of a link for the target images for the cards and also instructions on how to utilize the app efficiently.

Chapter 2

SOFTWARE DEVELOPMENT LIFE CYCLE

2.1 Software Development Life Cycle

From quality, accuracy and precision, the Software Development Life cycle acts as a methodical, systematic process for building software or a mobile application.

Regarded as an industry benchmark, SDLC basically ensures the quality part of work and emphasizes smooth flow and correctness of the end product. The end results down the SDLC process is high-quality software as expected by customers.

Abiding by the SDLC rules while shaping the product also affects or determines the time and cost taken to complete the project. Due to detailed plan and road map suggested by SDLC, one can easily look ahead and plan in advance and build software of their vision.

Each phase of SDLC cycle has its own value and deliverables, which eventually impacts the coming phases.

The basic importance of SDLC: -

- Forms the foundation for project planning and scheduling
- Helps estimate cost and time
- Includes the project activities and deliverables of each phase
- Boosts the transparency of the entire project and the development process
- Enhance the speed and accuracy of development
- Minimizes the risk potential and maintenance during any given project
- Its defined standard improves client relations

2.2 SDLC Stages

Every software development company goes through an array of stages as they embark on systematic process of development. From planning to design and development, here is a brief glance at the six essential stages of SDLC required to create flawless software:

- Planning
- Analysis
- Design
- Development
- Testing
- Maintenance

1)Planning

Fig. 2.1: SDLC Planning

Without a clear, visionary plan in place, it is difficult to align everything with your project goals and judge all its strengths, scope and challenges involved.

The planning is to ensure the development goes easy and smooth, meets its purpose and achieve its desired progress within given time limit.

2)Analysis

Fig. 2.2: SDLC Analysis

Analysing the requirements and performance of the software through its multiples stages is key to deriving process efficiency.

Analysis always helps be in the know of where you exactly stand in the process and where you need to be and what it takes to pass through the next step down the path.

3)Design

Fig. 2.3: SDLC Design

After the analytical part is complete, design is the next step that needs to look forward to. The basic aim in this phase is to create a strong, viable architecture of the software process.

As it works by a standard adherence, it helps eliminate any flaws or errors that may possibly hinder the operation.

4)Development

Fig. 2.4: SDLC Development

Once the design is ready, the development takes over along with the efficient data management and recording. This is a complicated phase where clarity and focus are of great significance.

Post development, implementation comes in picture to check whether or not the product functions as expected.

5) Testing

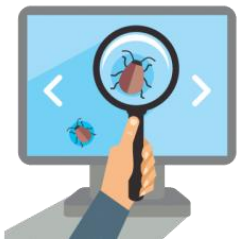


Fig. 2.5: SDLC Testing

The testing phase that comes now is inevitable as it studies and examines the software for any errors and bugs that may cause trouble.

6)Maintenance

Fig. 2.6: SDLC Maintenance

If the software has performed well through all the previous five steps, it comes to this final stage called maintenance. The product here is properly maintained and upgraded as and when needed to make it more adaptive to the target market.

2.3 Process Model Used

Agile SDLC methodology is an approach allowing to organize a project by dividing it into multiple stages. It implies continuous partnership with client and regular enhancements at every phase. Currently, it is one of the most widely-used software development models.

In 2001 the Agile software development cycle and its methodologies were invented and started to be applied. Seventeen software creators released the Agile Manifesto that described the principles and values of Agile most successful practices, which will be covered in this article a little bit later.

Ensured flexibility is the primary benefit of the Agile development life cycle due to its main concept-product creation accomplished by small, short stages. The model is created with continuous iterations of solutions that enable teams to make edits and upgrades regularly. Every upgraded version in the Agile development cycle is a fundament for the following one

2.3.1 Benefits

- Flexibility to comply with adaptive market and user requirements
- The space for a creative problem-solving approach
- Optimized resource allocation
- Regular updates and higher client satisfaction
- Rigid cadence, deadline adaptiveness

2.3.2 Values

- People and interaction are more significant than processes and instruments.
- Operating software is more important than thorough documentation.
- Partnership with clients is more prioritized than agreeing on the contract terms.
- Positive perception of changes is more significant than complying with the initial plan.

2.3.3 Phases

Let’s observe the Agile software development phases, as they also have certain peculiarities:

Fig. 2.7 Agile Phases

1. Concept: - During this software development cycle Agile stage, a client provides the details concerning his project to define the project scale. In case the aim is to create several projects simultaneously, they should be put in priority. A future product owner should also prepare fundamental requirements comprising the functionality, features, and gather all the essential documentation for proper structuring.

   The Agile life cycle model begins with minimal requirements needed as there's an opportunity to widen them despite the current phase. In this initial step, if the team is already gathered, developers are also able to conduct approximate calculations concerning the final costs, set deadlines, assess risks, and gather all these details in an Agile SDLC process document.

2. Inception: - After defining the essential product requirements and preparing Agile SDLC documentation, a project owner should choose suitable specialists with relevant expertise.

   According to SDLC and Agile methodology, the chosen team can proceed to the design creation. Specialists will build a UI mock-up which is a layout and placement of all the design components of the solution. During the inception phase in SDLC and Agile development, the client are usually engaged to define and provide more details on the requirements and clarify the functionality of the future software. Frequent reports will show that all the needs and requirements are followed during the design process.

3. Iteration: -The following Agile cycle model phase is iteration, called construction as well. Commonly, this stage takes the most time in comparison to the other Agile model phases, due to the fact that the major scope of work is fulfilled here. The programmers cooperate with UX designers to comply with all the software requirements and customer needs, converting the design into the script. The developers create essential solution functionality within the first iteration or sprint.

   Due to Agile development SDLC, extra features and tweaks may be implemented in the following iterations. The phase allows tech specialists to create operating digital products rapidly and refine them to deliver an enhanced user experience.

4. Release: - The software is almost ready for deployment. However, prior to launching, the SDLC life cycle in Agile requires quality assurance engineers to conduct certain tests to verify the software operation and make sure there are no bugs, flaws, or failures. In case any issues are detected, the solution has to be fixed by developers. If the script is clean, it's time to complete the release stage of the Agile project life cycle model. User training is also an indispensable process of this stage.

5. Maintenance: - Here, the product is deployed and may be tested by the first users. In the SDLC using Agile methodology, this stage refers to the software support and enhancement from the developers' side to keep it working seamlessly and flawlessly. In case there are any product running problems, they are fixed in this Agile software development lifecycle phase. Furthermore, the team can help with the user onboarding process to acquaint them with all the opportunities provided by the software. Then, new iterations can take place due to the Agile software cycle to upgrade the functionality of the existing product.

6. Retirement: - It is one more stage among the SDLC Agile model phases, which is conditioned by the two main reasons:

   - The solution is substituted by the new product
   - The software has become outdated

   The software development team will inform the users that the product is obsolete and has stopped operating. In case there's a brand-new solution available, clients will be asked to move to the new one. Lastly, specialists fulfil further end-of-life tasks to stop the support of the older software and finish all SDLC Agile phases.

The Iterative approach underlies the whole Agile software development life cycle. The result of each iteration is the next piece of software and may be working software, some element, feature, documentation, etc. Iteration is repeated until the final product is completed and delivered to customers.

Iterative development can also be divided into the following phases:

Fig. 2.8 Iterative Agile Workflow

- Requirements – development team defines the requirements for current iteration based on feedback from customers, business owners, and the whole project plan
- Development – the identified requirements are used to design and develop software

- Testing – the software is tested and documentation is created
- Delivery – software is released
- Feedback – the development team gathers feedback from users (both internal and external) and uses it for further requirement definition

The whole iterative process looks like a loop, where all steps are repeated, until all requirements in the product backlog are met. Repetitiveness defines the Agile SDLC.

Chapter 3

ANALYSIS

3.1 Requirement Analysis

It is related to getting the requirements and various specifications required for developing particular software. The basic idea behind the requirement analysis is to know about what the customer wants. This phase is of utmost importance as it contains huge amount of important data, which will be used in developing software. If this phase is not carried out properly, then certain bugs or shortcomings will be there in the software, which is going to be developed by the programmer.

3.1.1 Hardware Requirement

Hardware requirements for the system to work are:

1) CPU: Any Processor (above 500MHz) with supported Graphic card

2) Hard disk: 256 GB

3) Memory: 4 GB ram

4) Camera: Any standard camera (720p or above webcam)

3.1.2 Software Requirement

Software requirements for the system to work are as follows:

1) The software will work on any of the Microsoft Windows OS.

2) IDE-Visual Studio.

3) Unity game engine + Vuforia package

4) Blender 3D

3.2 REQUIREMENT SPECIFICATIONS

3.2.1 Functional Requirement

It deals with the functionalities required from the system which are as follows:

- Proper arrangement of Action buttons on menu.
- The User can perform any of the Action displayed on the menu

3.2.2 Non-functional Requirement

- They are the quality requirements that stipulate how well a software does what it has to do.
- Questions such as How fast the application loads or how does the 3D elements of app work when user uses the camera in certain angle, are answered.

3.3USECASE ANALYSIS

3.3.1 Usecase Diagram

Usecase diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally. Following are some use-case diagram symbols: -

Introduction of UML diagrams:

- Unified Modelling Language (UML) is a general-purpose visual modelling language.
  - Can support all existing lifecycles.
  - Intended to be supported by CASE tools.
- Unifies past modelling techniques and experience.
- Incorporates current best practice in software engineering.
- UML is not a methodology!
- UML is a visual language.

3.3.2 USECASE DISCRIPTION

A use case is a written description of how users will perform tasks on your website. It outlines, from a user's point of view, a system's behaviour as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled.

Elements of a Use-Case:

Depending on how in depth and complex you want or need to get, use case describe a combination of the following elements:

- Actor – anyone or anything that performs a behaviour (who is using the system). In this case, User is the actor.

• Primary Actor – stakeholder who initiates an interaction with the system to achieve a goal. In this case, User is also the Primary actor.

• Preconditions – The Preconditions for this usecase to run is that the user must have the application installed properly.

• Triggers – when the user opens the application, the usecase is initiated.

• Main success scenarios [Basic Flow] – The Actor or the User must first download the target images and then use the AR Camera mode to properly utilize the application.

• Alternative paths [Alternative Flow] – The Actor might try to open the camera without downloading the target images, In this case the Camera will open successfully, but there will be no 3D models in it.

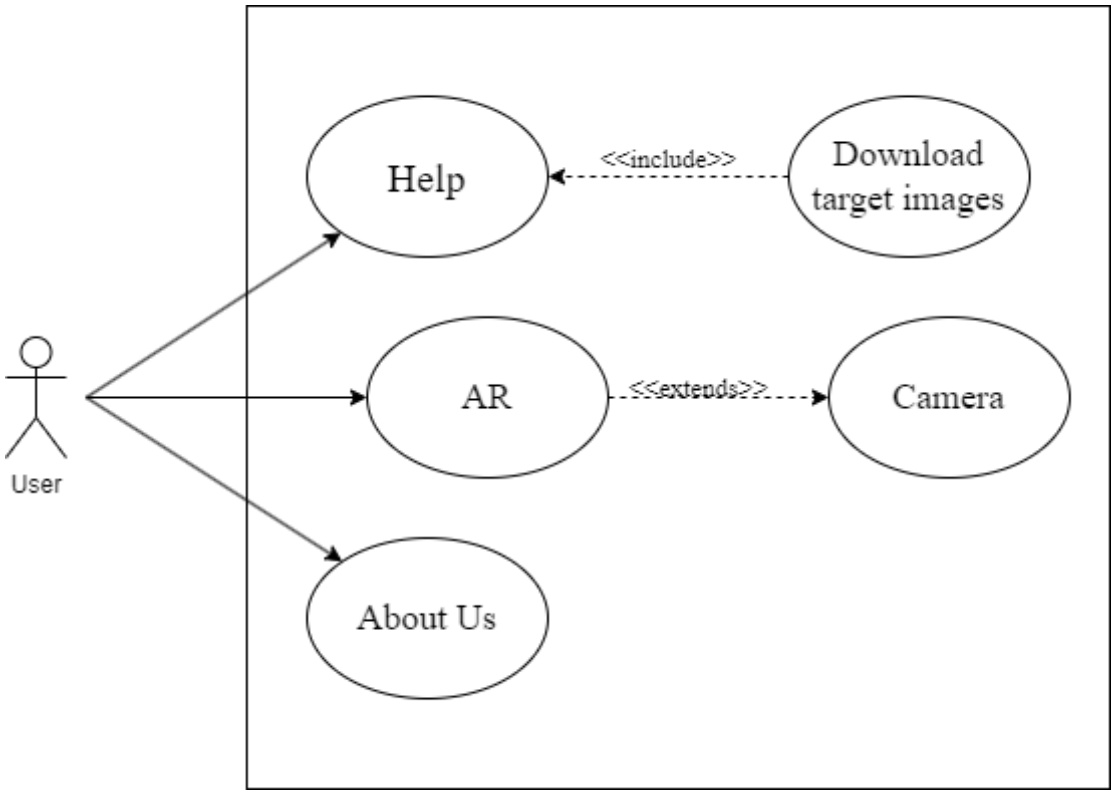


Fig. 3.1: Use-case diagram

- The use-case for the EduWAR application is displayed above.
- In this, User can select any activity given such as Help, AR, or About Us.
- In Help, user can read the instructions for the app and also download the target images. In AR, User can open camera which will show the 3D models on the display.

- In About Us, User can know the information about the Developers.

3.3.3 Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

In UML, an activity diagram provides a view of the behaviour of a system by describing the sequence of actions in a process.

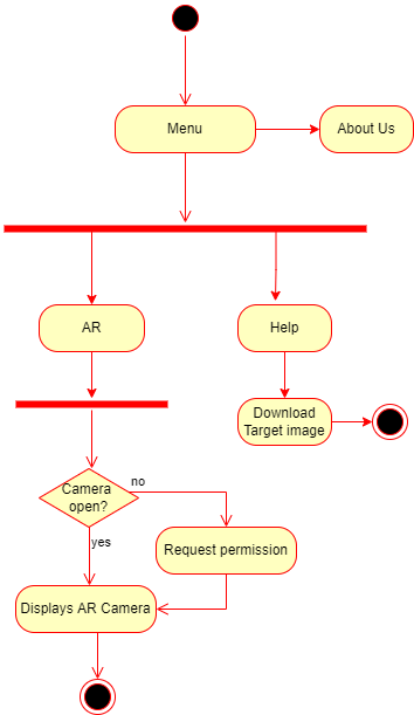


Fig. 3.2: Activity Diagram

- The Activity diagram for the EduWAR application is displayed above, We start with the initial node.
- The User selects Menu activity from which user can select other activities.
- The Menu Activity forks for splitting in 2 activities, AR and Help.
- In Help activity, user can perform the Download Target image activity which will download the Target images.
- In AR activity, user can Open camera for which the system will ask for permission. If permission allowed then 3D model in displayed.

Chapter 4

DESIGN

4.1 System Flow Diagram

The system flow diagram is one of the graphical representations of the flow of data in a system in software engineering. The diagram consists of several steps that identify where the input is coming to the system and output going out of the system. With the help of the diagram, it is possible to control the event decisions of the system and how data is flowing to the system. Therefore, the system flow diagram is basically a visual representation of data flow, excluding the minor parts and including the major parts of the system in a sequential manner.

Common Flowchart Symbols

- Rectangle Shape – Represents a process (Target placement, Model display)
- Oval or Pill Shape – Represents the start or end (Start, End)
- Diamond Shape – Represents a decision (target detected condition)

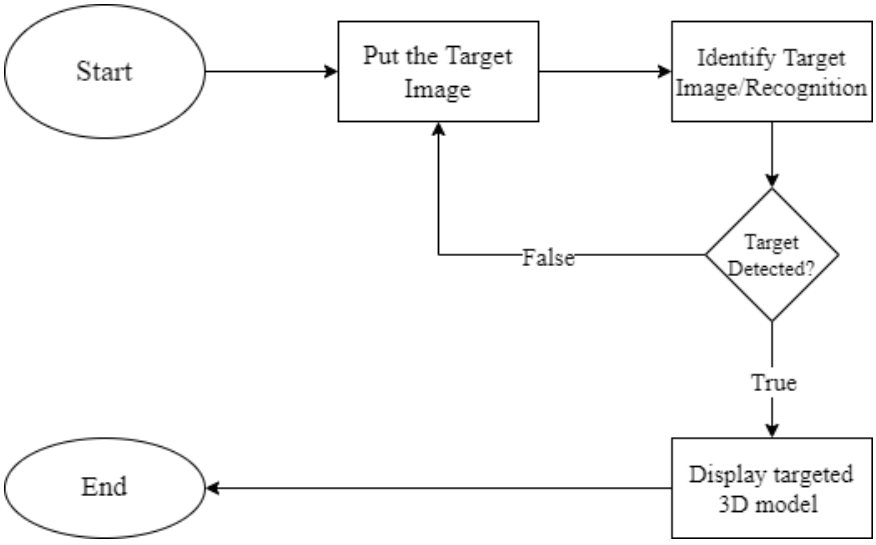


Fig. 4.1: System Flow Diagram

- The System flow diagram for the EduWAR application is displayed above. We start with the Start state(oval).
- we put an image Infront of the camera.
- The image is then processed, if the image is a valid target image(True), then a 3D model is shown, and if not detected(False), we can either change the camera angles or try different images.
- After a 3D model is shown the flow ends with End state(oval).

4.2 Data Flow Diagram

A data flow diagram shows how data is processed within a system based on inputs and outputs. Visual symbols are used to represent the flow of information, data sources and destinations, and where data is stored. Data flow diagrams are often used as a first step toward redesigning a system. They provide a graphical representation of a system at any level of detail, creating an easy-to-understand picture of what the system does.

Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

External Entity – Also known as actors, sources or sinks, and terminators, external entities produce and consume data that flows between the entity and the system being diagrammed. They can represent another system or indicate a subsystem. In this project, user acts as an external entity.

Process – An activity that changes or transforms data flows. Since they transform incoming data to outgoing data, all processes must have inputs and outputs on a DFD. In this project We have 3 processes namely, Smart-Phone, AR, ArObject.

Data Store – A data store does not generate any operations but simply holds data for later access. Data stores could consist of files held long term or a batch of documents stored briefly while they wait to be processed. In this project Models collection is a data store which consists of all the 3D models.

Data Flow – Movement of data between external entities, processes and data stores is represented with an arrow symbol, which indicates the direction of flow.

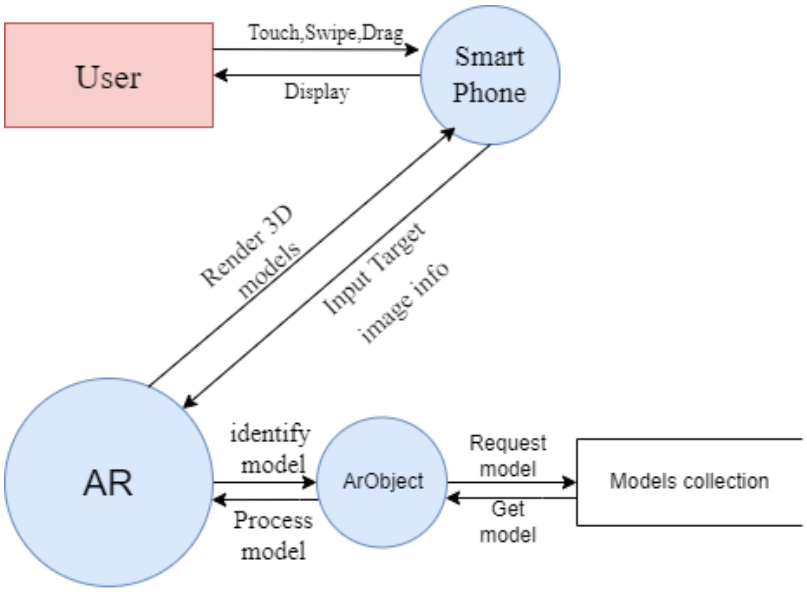


Fig. 4.2: Data flow diagram

- The Data flow diagram of level 1 for the EduWAR application is displayed above. Any interaction with the smart phone(touch, swipe, drag) made by the user is taken as input.
- The Smart phone camera captures features of the target image and gives this information to AR which identifies the model and request it from the Model collection through ArObject and renders it as a 3D model on smart phone.

4.3 Modules Identified

Menu:- This module consists the main page canvas of the application. It consists of buttons that navigate to the other pages, the Logo of the application, The copyright text and a button to close the application.

AR:- This module consists of the canvas for the camera page of the application. It also contains the 3D models including their labels, which activates whenever an image is detected and a button that returns to menu page.

About_Us:- This module consists of the About us page canvas. It also contains buttons to Socials and a button that returns to menu page.

Help:- This module consists of the help page canvas. It consists of the instruction for using the application, and also a button to dounload image targets.

4.4 Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.

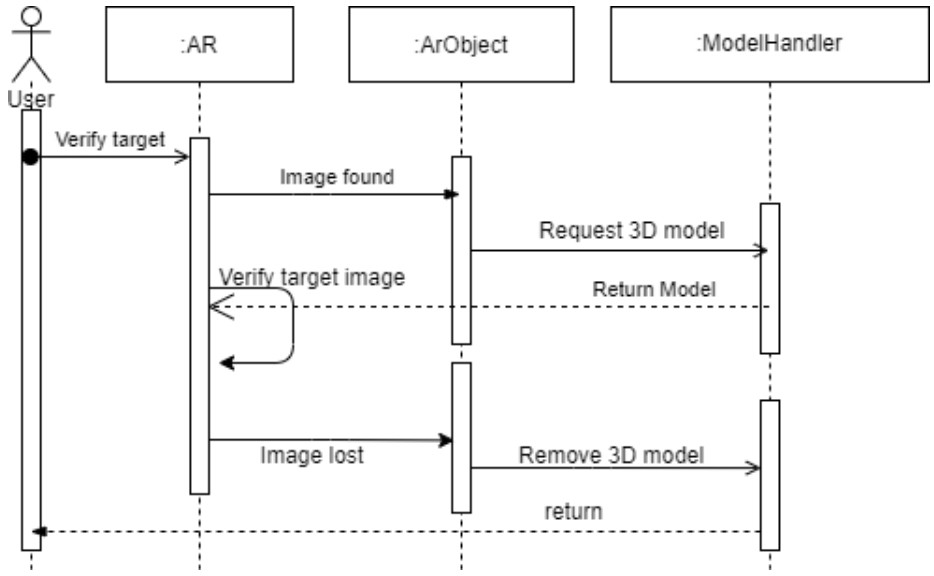


Fig. 4.3: Sequence Diagram (AR)

- The Sequence diagram for the EduWAR application is displayed above. This diagram represents the AR class of the application.
- When User opens the camera, The system looks for a target image (marker).
- If the target is found then the AR-Object will load model from Model Handler which will be displayed to the User, and if the image is removed or even partially hidden, the model disappears.

4.5 Class Diagram

A class diagrams model the static structure of a system. They show relationships between classes, objects, attributes, and operations. The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagram is widely used in the modelling of object-oriented systems because they are the only UML diagrams which can be mapped directly with object-oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram. Some of the points should be remembered while drawing a class diagram:

- The name of the class diagram should be meaningful to describe the aspects of the system.
- Each element and their relationships should be identified in advance.
- Responsibility of each class should be clearly identified.
- For each class minimum number of properties should be specified. Because unnecessary properties make the diagram complicated.

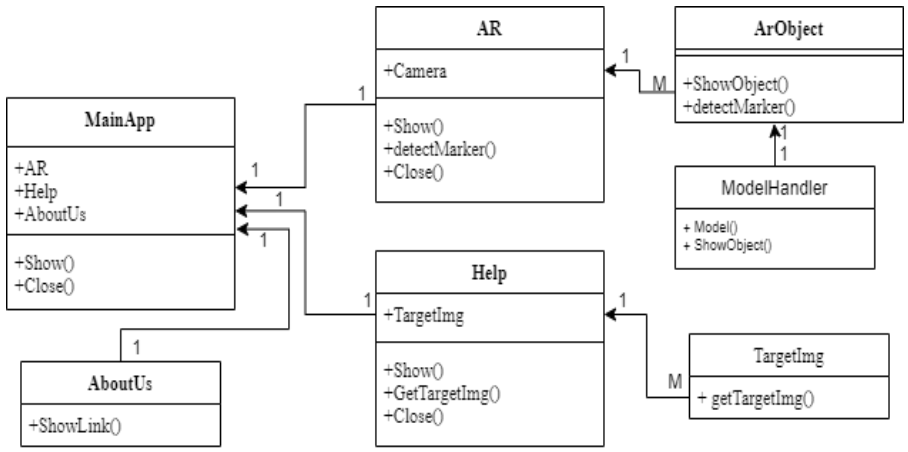


Fig. 4.4: Class Diagram

- The Class diagram for the EduWAR application is displayed above. Here, we have 7 classes.
- The MainApp class inherits Methods from other classes. A common Operation in this diagram is Show() and Close().
- The AR class inherits the Methods from ArObject class. One Camera can show multiple models at a time. The ArObject inherits from ModelHandler.
- The Help class inherits multiple Target images from TargetImg class. About us class is used for displaying the application developer's information.

Chapter 5

IMPLEMENTATION & TESTING

5.1 Implementation

The implementation of an AR application as a minor project can be broken down into several components:

Planning: Define the scope of the project, including the target audience and other key features.

Design: Develop the Application layout and user interface, considering usability, accessibility, and branding. Use wireframes and mock-ups to visualize the website before building it.

Development: Build the Application using Unity game engine using Vuforia Package which consists of Addon scripts that helps in development of Augmented reality applications and scripting language C#.

Testing: Test the Application thoroughly before launching it to the public, including functional testing, usability testing, and security testing.

Launch: Once the website is ready, launch it to the public and promote it through marketing channels like social media , YouTube, and email marketing.

5.2 Platform used:

5.2.1 HARDWARE PLATFORM:

Workplace Application - Laptop

Processor – AMD Ryzen 5 5000 series

Monitor- 14'' colour monitor.

RAM – 8 GB

5.2.2 SOFTWARE PLATFORM

Operating System - Windows 11

Language – C#

Tool – Unity Engine, Unity Hub, Vuforia

5.3 Testing Strategy Adapted

Software testing is the process of evaluating a software item to detect differences between given input and expected output. Also, to assess the feature of A software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words, software testing is a verification and validation process.

There are various types of Testing Techniques:

Functional Testing types include:

- Unit Testing
- Integration Testing
- System Testing
- Interface Testing
- Beta/Acceptance Testing

Non-functional Testing types include:

- Performance Testing
- Load Testing
- Stress Testing

5.4 System Testing

System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of software to perform various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different types of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements.

5.4.1 Test Cases

A test case has an input, an action, and an expected result. In using test cases, the tester is trying to break the application. The whole point of using test cases is to find defects. Hopefully, serious defects that crash the system are found before your application is released to the customer. These show stoppers, or defects that may delay the release of the application, must be found and fixed to save time and money and prevent customer dissatisfaction. Test cases must exercise every feature of the application to prevent defects from being released. Each test case needs to contain a set of test steps for a feature or function. At the end of the test, the expected results are compared to the actual results to determine if the application is working as it should.

5.5 Piece of Code that highlight some of our best work

➔ **Event System(modelHandler) for Augmented reality:-**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Socials : MonoBehaviour
{
    public void GitHubLink(){
        Application.OpenURL("https://github.com/Ronit-gurjar");
    }
    public void InstaLink(){
        Application.OpenURL("https://www.instagram.com/ronitgurjar__");
    }
}
using System;
using System.Collections.Generic;
using System.Text;
using UnityEngine;
using UnityEngine.Serialization;

namespace UnityEngine.EventSystems
{
    [AddComponentMenu("Event/Event System")]
    public class EventSystem : UIBehaviour
    {
        private List<BaseInputModule> m_SystemInputModules = new
List<BaseInputModule>();

        private BaseInputModule m_CurrentInputModule;

        private  static List<EventSystem> m_EventSystems = new
List<EventSystem>();

        /// <summary>
        /// Return the current EventSystem.
        /// </summary>
        public static EventSystem current
        {
            get { return m_EventSystems.Count > 0 ? m_EventSystems[0] : null;
}
            set
            {
                int index = m_EventSystems.IndexOf(value);

                if (index >= 0)
```

```
                {
                    m_EventSystems.RemoveAt(index);
                    m_EventSystems.Insert(0, value);
                }
            }
        }
        private GameObject m_CurrentSelected;
        public BaseInputModule currentInputModule
        {
            get { return m_CurrentInputModule; }
        }
        public GameObject firstSelectedGameObject
        {
            get { return m_FirstSelected; }
            set { m_FirstSelected = value; }
        }
        public GameObject currentSelectedGameObject
        {
            get { return m_CurrentSelected; }
        }

        [Obsolete("lastSelectedGameObject is no longer supported")]
        public GameObject lastSelectedGameObject
        {
            get { return null; }
        }

        private bool m_HasFocus = true;
        public bool isFocused
        {
            get { return m_HasFocus; }
        }

        protected EventSystem()
        {}
        public void UpdateModules()
        {
            GetComponents(m_SystemInputModules);
            for (int i = m_SystemInputModules.Count - 1; i >= 0; i--)
            {
                if (m_SystemInputModules[i] &&
m_SystemInputModules[i].IsActive())
                    continue;

                m_SystemInputModules.RemoveAt(i);
            }
        }
```

```
        private BaseEventData m_DummyData;
        private BaseEventData baseEventDataCache
        {
            get
            {
                if (m_DummyData == null)
                    m_DummyData = new BaseEventData(this);

                return m_DummyData;
            }
        }
        public void SetSelectedGameObject(GameObject selected)
        {
            SetSelectedGameObject(selected, baseEventDataCache);
        }

        private static int RaycastComparer(RaycastResult lhs, RaycastResult
rhs)
        {
            if (lhs.module != rhs.module)
            {
                var lhsEventCamera = lhs.module.eventCamera;
                var rhsEventCamera = rhs.module.eventCamera;
                if (lhsEventCamera != null && rhsEventCamera != null &&
lhsEventCamera.depth != rhsEventCamera.depth)
                {
                    // need to reverse the standard compareTo
                    if (lhsEventCamera.depth < rhsEventCamera.depth)
                        return 1;
                    if (lhsEventCamera.depth == rhsEventCamera.depth)
                        return 0;

                    return -1;
                }

                if (lhs.module.sortOrderPriority !=
rhs.module.sortOrderPriority)
                    return
rhs.module.sortOrderPriority.CompareTo(lhs.module.sortOrderPriority);

                if (lhs.module.renderOrderPriority !=
rhs.module.renderOrderPriority)
                    return
rhs.module.renderOrderPriority.CompareTo(lhs.module.renderOrderPriority);
            }

            if (lhs.sortingLayer != rhs.sortingLayer)
```

```
            {
                // Uses the layer value to properly compare the relative order
of the layers.
                var rid = SortingLayer.GetLayerValueFromID(rhs.sortingLayer);
                var lid = SortingLayer.GetLayerValueFromID(lhs.sortingLayer);
                return rid.CompareTo(lid);
            }

            return lhs.index.CompareTo(rhs.index);
        }
        public bool IsPointerOverGameObject()
        {
            return IsPointerOverGameObject(PointerInputModule.kMouseLeftId);
        }
        public bool IsPointerOverGameObject(int pointerId)
        {
            if (m_CurrentInputModule == null)
                return false;

            return m_CurrentInputModule.IsPointerOverGameObject(pointerId);
        }

        protected override void OnEnable()
        {
            base.OnEnable();
            m_EventSystems.Add(this);
        }

        protected override void OnDisable()
        {
            if (m_CurrentInputModule != null)
            {
                m_CurrentInputModule.DeactivateModule();
                m_CurrentInputModule = null;
            }

            m_EventSystems.Remove(this);

            base.OnDisable();
        }

        protected virtual void OnApplicationFocus(bool hasFocus)
        {
            m_HasFocus = hasFocus;
        }

        protected virtual void Update()
```

```
        {
            if (current != this)
                return;
            TickModules();

            bool changedModule = false;
            for (var i = 0; i < m_SystemInputModules.Count; i++)
            {
                var module = m_SystemInputModules[i];
                if (module.IsModuleSupported() &&
module.ShouldActivateModule())
                {
                    if (m_CurrentInputModule != module)
                    {
                        ChangeEventModule(module);
                        changedModule = true;
                    }
                    break;
                }
            }
            if (m_CurrentInputModule == null)
            {
                for (var i = 0; i < m_SystemInputModules.Count; i++)
                {
                    var module = m_SystemInputModules[i];
                    if (module.IsModuleSupported())
                    {
                        ChangeEventModule(module);
                        changedModule = true;
                        break;
                    }
                }
            }

            if (!changedModule && m_CurrentInputModule != null)
                m_CurrentInputModule.Process();
        }

        private void ChangeEventModule(BaseInputModule module)
        {
            if (m_CurrentInputModule == module)
                return;

            if (m_CurrentInputModule != null)
                m_CurrentInputModule.DeactivateModule();

            if (module != null)
```

```
                module.ActivateModule();
            m_CurrentInputModule = module;
        }

        public override string ToString()
        {
            var sb = new StringBuilder();
            sb.AppendLine("<b>Selected:</b>" + currentSelectedGameObject);
            sb.AppendLine();
            sb.AppendLine();
            sb.AppendLine(m_CurrentInputModule != null ?
m_CurrentInputModule.ToString() : "No module");
            return sb.ToString();
        }
    }
}
```

➔ **Script for Menu window buttons:-**

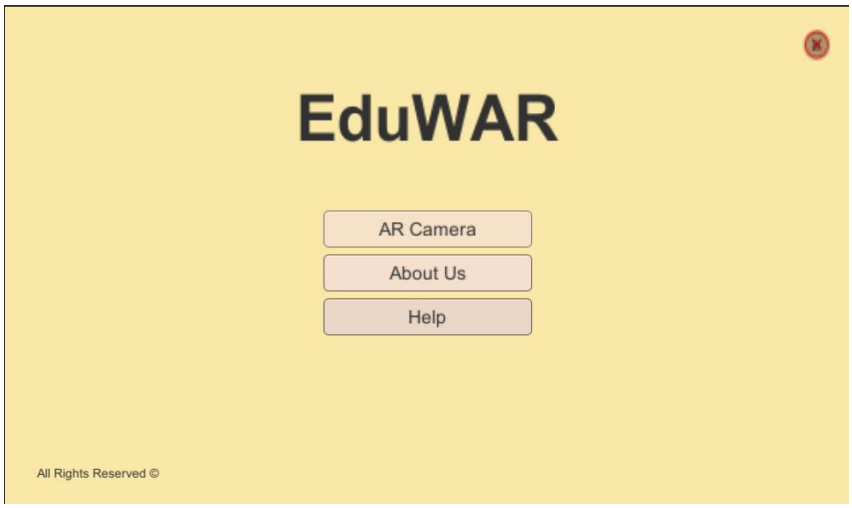
```
using UnityEngine;
using UnityEngine.SceneManagement;
public class MainMenu : MonoBehaviour
{
    public void GoToScene(string sceneName){
        SceneManager.LoadScene(sceneName);
    }

    public void QuitApp(){
        Application.Quit();
        Debug.Log("App quited successfully");
    }
}
```

➔ **Script for Help window “Download target images” button**

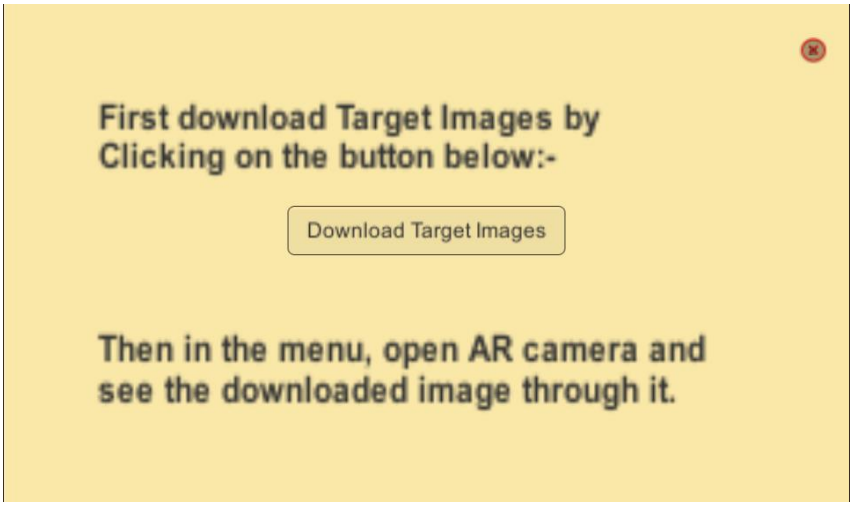
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Download_Link : MonoBehaviour
{
    public void OpenLink(){
        Application.OpenURL("https://github.com/Ronit-gurjar/Download-Images-
minor");
    }
}
```

5.6 SCREENSHOTS OF EduWAR:-

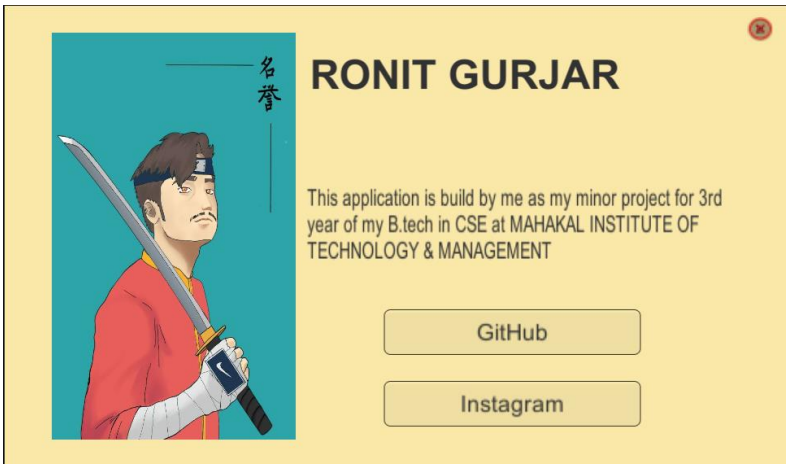
Menu page(home screen)



Help page



About Us



AR Camera

Below images show how, when a target image is presented infront of the AR camera, a 3D model is augmented over its respective image.

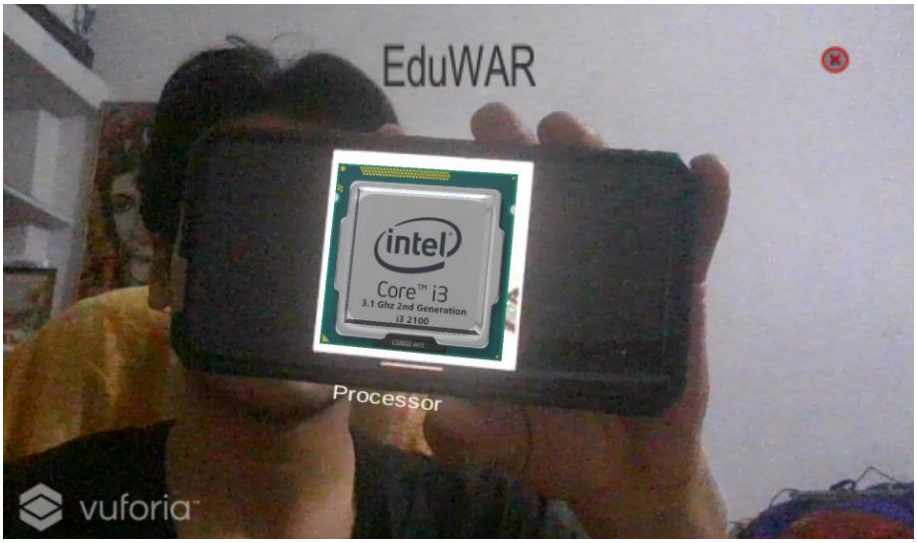


Image for "processor"

Image for "Output Device"

Chapter 6

CONCLUSION

6.1 Important Features

As a student of Computer Science & Engineering department, sometimes students face the difficulty in understanding certain topics in subjects like Computer Architecture. This project will help students in getting grasp of the difficult topics using 3D models with which they can interact and combine components to build a computer system from scratch.

The 3D models used in this project are based on real-life hardware components So that students know what a particular component looks like and have a basic understanding of how might the component is used in a computer system

6.2 Limitations

- The application is limited only to certain topics.
- The application can only be used on Mobile Phones that supports Android version 9.0.0 or higher

6.3 Future Scope

In future version of **EduWAR**, more subjects and topics will be added so that Student can become more imaginative and better thinker. This project will help advance the Education field using EdTech methodology. AR will also introduce students to the field of Extended Reality which will help them understand how the future technologies such as **Metaverse** Works.

REFERENCES: -

- https://docs.unity3d.com/Manual/AROverview.html
- http://library.vuforia.com/getting-started/getting-started-vuforia-engine-unity
- http://library.vuforia.com/objects/image-targets
- https://www.youtube.com/watch?v=Hf2esGA7vCc&list=PLjEaoINr3zgEL9UjPTLWQhLFAK7wVaRMR
- https://youtube.com/playlist?list=PLX2vGYjWbI0Thl0pOCbKWrbbiw7RWiRG7
- https://youtu.be/Qi3h18wJJiI
- https://www.youtube.com/@RenderIsland/featured