National College of Ireland

Computing Project (BSHCSD4)

Software Development

2023/2024

Fabian Felix Gal
x20434312

x20434312@student.ncirl.ie

VR-Adset

Technical Report

Contents

Executive Summary

This report outlines the development process of the VR-Adset application. This application aims to be a platform that will benefit both virtual reality (VR) developers and advertisers by offering easy-to-use 3D assets for in game advertisement. A 3D asset can be any object already existing in real life which is capable of displaying an image, such as a poster, a billboard, or a TV.

There will also be a website platform where advertisements can be uploaded and showcased within a virtual reality environment. In simpler words, this application aims to be an easily accessible website platform, where Advertisers can upload an ad campaign, such as an image, similar to one found on a billboard. After the ad is on the platform, it will be added to the 3D assets which are being used for the ad campaign. Those advertisements can now be viewed within a virtual reality environment by the end user.

This report will outline the development process starting with an introduction to the project, its aims, technologies used and the architecture. It will also detail the functional requirements which form the pillars of this application. It will cover the use of wide range of tools and technologies that made this project possible such as Ruby on Rails, SQLite, Unreal Engine, Blender, GitHub, Connecter, Oculus Quest, AWS and more. The report will then cover the technical, system and functional requirements that are necessary in delivering a functional application. Subsequently, this report will illustrate the application's potential and necessity in our ever-evolving, innovative world.

1. Introduction

1.1 Background

The inspiration for this project started with my personal experience with using a Meta Quest 2 VR headset. The lack of entertaining content highlighted a lack of development interest or resources available to incentivise developers to work or maintain existing projects. Some of the applications that I would be encountering within the available marketplace, would feel very lacking and that they could use additional resources and development to become a successful game or application. My initial thoughts were that if the developers had another source of income aside from the one-time purchase of the application or game, it could help motivate them expand and address the scarcity of resources they might be facing.

Researching the idea, I found out that there are few companies that are providing an easy-to-use platform for both advertisers and developers. In fact, most solutions would be agencies that would work with companies in developing one custom environment for their one purpose advertisement which becomes costly and time consuming for the advertiser. In addition, Google since 2017 has yet to release one easily available platform since their experiment with VR ad formats at Area 120 (Upadhyay Aayush, 2017). So far, only one company, Anzu, (Anzu, n.d.) has managed to implement VR advertising in a similar style, but their focus is limited to the Metaverse. In contrast, my approach supports new and emerging developers who are more likely to use Unreal Engine because of its rising popularity and range of features it offers to VR developers. (Unreal Engine, 2023)

1.2 Aims

The aim of this project is creating a platform where advertisers can upload their ads with confidence that they will be displayed in a VR environment, without having to hire developers to build custom worlds or assets. This will cut the costs in development allowing for a reallocation of funds on views and conversions. The advertisements should also be safe for kids and people of all ages before they can be displayed. For ethical reasons, the advertiser is required to comply with this requirement, or the advertisement will be discarded.

The web application will provide developers with a list of assets that they can download and use within their game or VR environment. These assets should require little to no setup and will then be automatically filled in with an ad upon runtime.

The aim for the end users is that they can view the ads placed around a VR environment by a developer.

The admin should be able to monitor the advertisements and delete them in case of any infringement on the rules such as inappropriate ads.

1.3 Technology

To achieve the goals stated above, the following technologies and applications will be required.

For the 3D development of assets, Blender will be utilized to create or refine the models. This is as Blender is one of the most popular and versatile open-source software that sometimes exceeds existing premium software used by large companies.

To build and test the addition of an asset within a VR environment, Unreal Engine 5.3.2 will be used. Unreal engine uses blueprint programming, and this method will be used for the development of assets. Blueprint programming is a graphical user interface that is underpinned by C++ code. This method makes use of nodes to represent actions, events, and logic similar to a flowchart. Connecting these nodes together allows for developing the logic of the assets.

Unreal Engine was chosen because of their wide range of tools available for beginner developers when creating VR environments. Its simplicity is likely to draw upcoming game developers, especially with the latest release of Unreal Engine 5. (Unreal Engine, 2023) Additionally, recent controversies have arisen from their main competitor, Unity, which introduced and then removed unprecedented fees due to public backlash. This has left many people within the community untrusting of their leadership and thus likely to hesitate when using them for future projects. (Totilo, 2023)

As Unreal Engine lacks a way of exporting an asset with the attached blueprint functionality in one simple file, the use of Connecter application is required. Currently, the only option available is to export the whole map or the entire folder structure and adding it to another project. This method is not very beginner friendly, and it becomes more of a struggle for the developer. Connecter, is a free asset management tool which is widely used within the 3D artist community. It is most widely used with apps such as 3DS MAX, Revit, Cinema 4D, Rhinoceros, Blender, Unreal Engine and Maya. (Connecter, 2023) With the use of this tool, the developer will have an easy access to importing downloaded assets straight into their project with only a few clicks. It is very intuitive and will require very little training even for people unfamiliar with 3D art.

To view the environment and assets which will be initially developed on Windows, into the Oculus Quest 2 headset, I will also be making use of Oculus App and the Air link feature that they have available. This will allow a direct and wireless connection to the computer running the application and simulate it in a VR environment.

Focusing on the development of the website platform, Ruby on Rails will be used for building the application. This approach will accelerate the development process by using scaffolding and methods of testing familiar from previous modules. Bootstrap will be employed to style the application and for version control, Git and GitHub will be utilized.

When choosing the database for the application, sticking with the default SQLite3 database that comes with Ruby on Rails seemed to be the more fitting approach due to its wide

adaptability in many existing projects making the development process easier and the troubleshooting process faster. The familiarity with SQL language further eases its usage.

The ad server and the Rails application will be hosted on an AWS EC2 instance, selected for its low cost, ease of use, and prior successful experiences with the service. This decision comes as an alternative to Oracle Cloud instances, which are no longer available in the desired region for setup.

For the application to be easily accessible for anyone having a domain name is crucial. People will generally not reach out to a website that they can not recognize or does not seem secure. To address this, Cloudflare DNS and its SSL certificate were used, to not only give a name to the website but also securing the connection from a user's browser to the website platform. This results in the following URL: https://vr-adset.com/

1.4 Structure

This document is structured in 4 main sections, the Introduction, System, Conclusions and Further Development or Research. Finally, there are 2 more sections which support and supplement the report, such as References and Appendices.

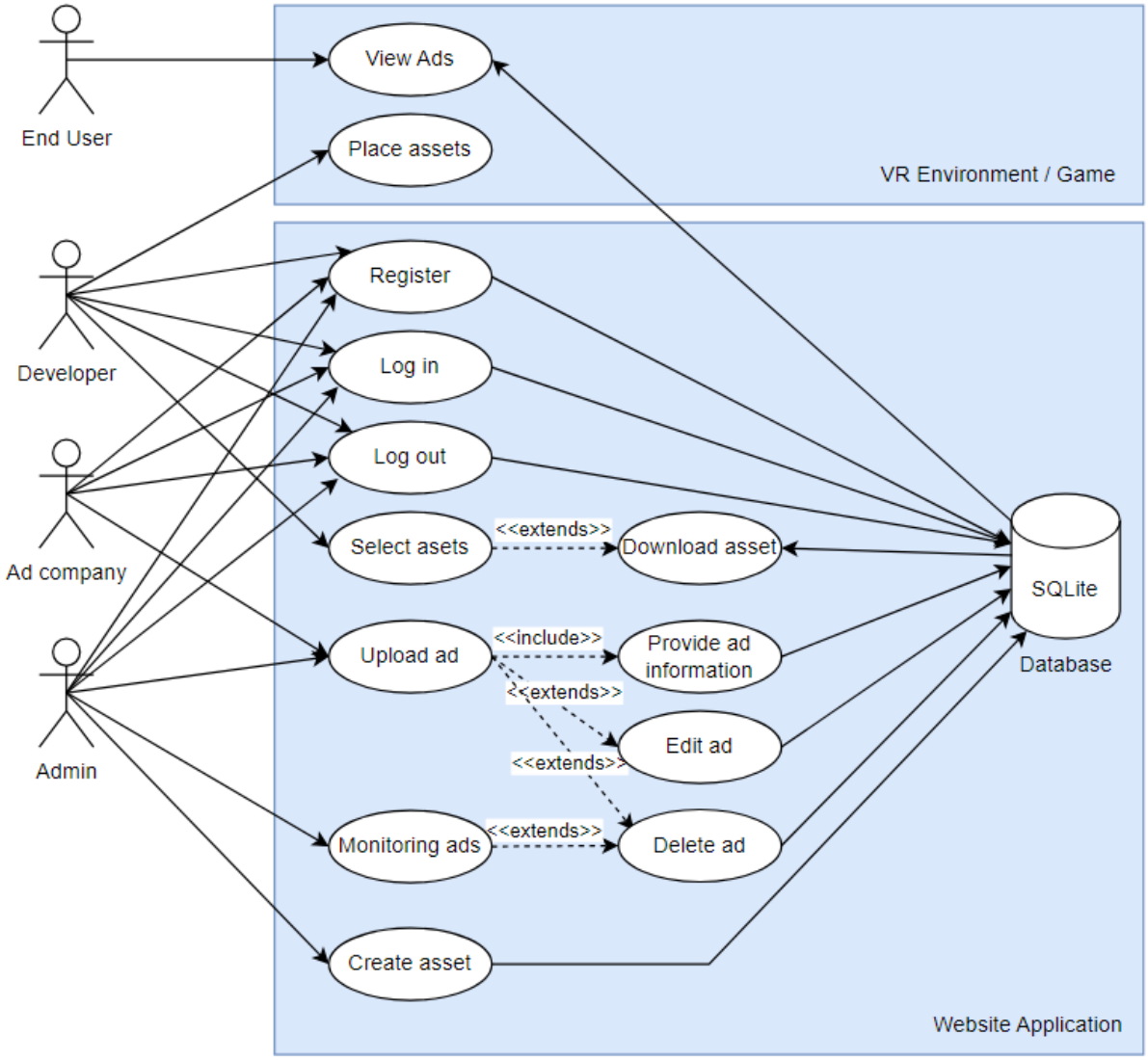
The main section relates to the System and the System requirements under section 2.1. These requirements provide an overview of my application's functional and non-functional, data, user, environmental, and usability requirements. These aim to provide an understanding of my application and its goals. The Priority of the following functional requirements are based on what would be necessary to achieve a full application.

2. System

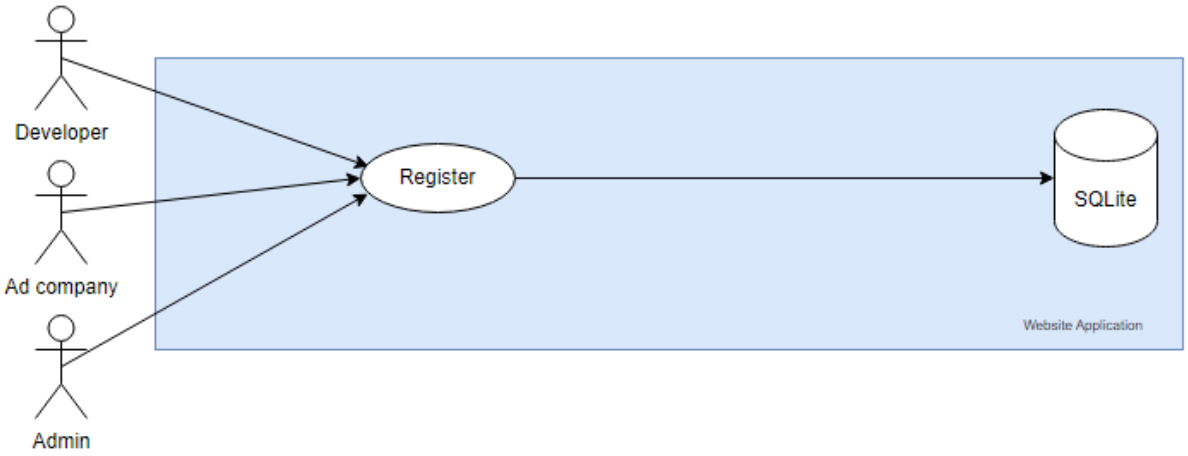
2.1 Requirements

2.1.1 Functional Requirements

2.1.1.1System Diagram



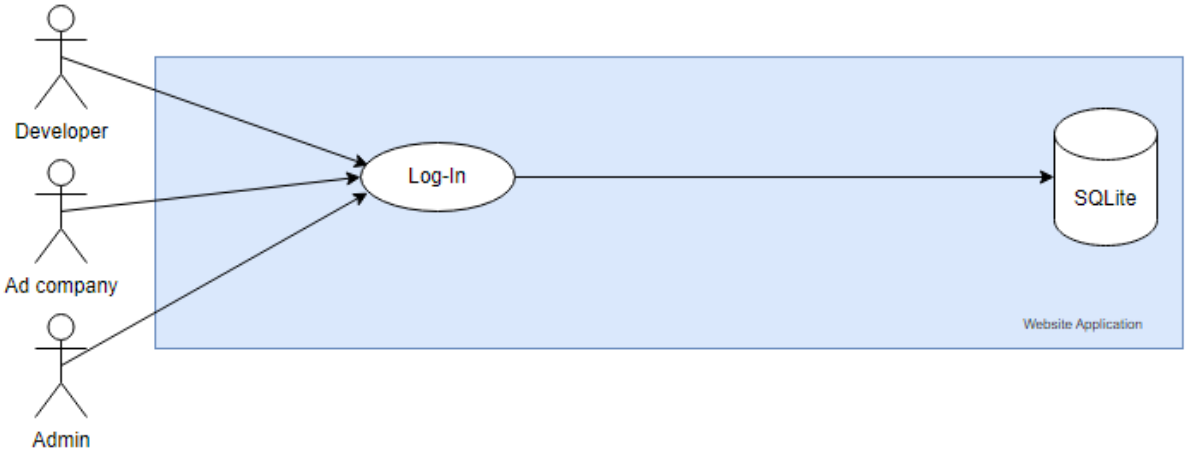
2.1.1.2Requirement 1: Registration



Use Case	Register	ID	UC-1	Priority	High
Description	The user should be able to register to make use of the website application such as download, upload, monitor and edit current ads or assets.				
Actors	Developer, Ad Company, Admin				
Precondition	A new user to the platform which has not yet registered.				
Activation	The user case starts when the actors chose to register for the application.				
Main flow	1. The actor accesses the site without being registered. 2. The actor selects to register from the navigation bar. 3. The system displays a list of required fields for the user to fill in. 4. The user fills in the required information. 5. The system asks which type of actor/role they will like to sign up under 6. The user submits their choice. 7. The system informs the user of the successful registration.				
Alternate flow	A1: Invalid information provided. 1. The user provides wrong information such as invalid email address, inputs an integer where a string is required or omits to fill in required information. 2. The system will inform the user of the missing or wrong information. A2: The new user registers as an admin 1. The user registers as an admin to the application. 2. They should not be able to perform any admin actions without permission from an already existent admin with all necessary permissions.				
Exceptional flow	E1: The application fails to register the user due to connection issues. 1. The user attempts to register. 2. The system is irresponsive due to a connection issue or instance being offline.				

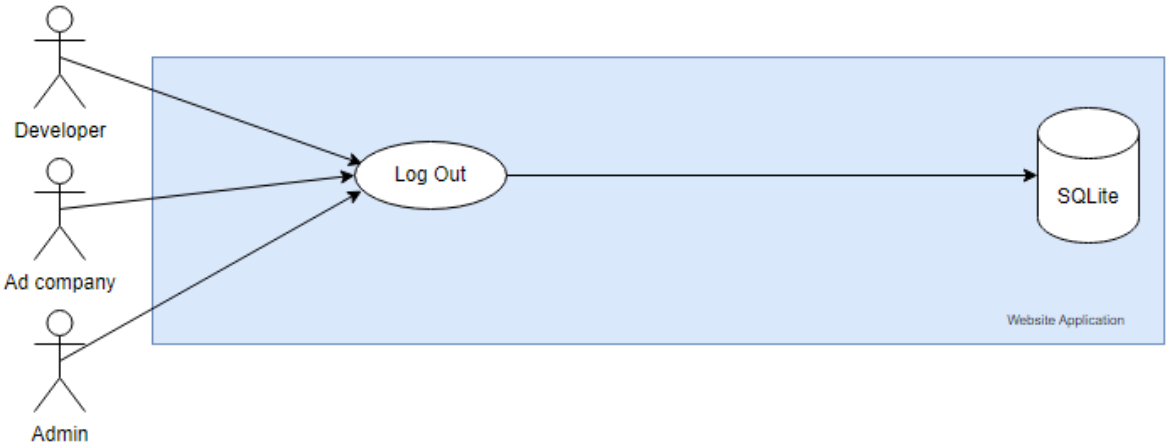
	3. The system is unable to respond or reply to the user and will be in a loading state before disconnecting the browser.
Termination	The system presents the user with the login screen.
Post Condition	A new account has been created for the user.

2.1.1.3 Requirement 2: Log in.



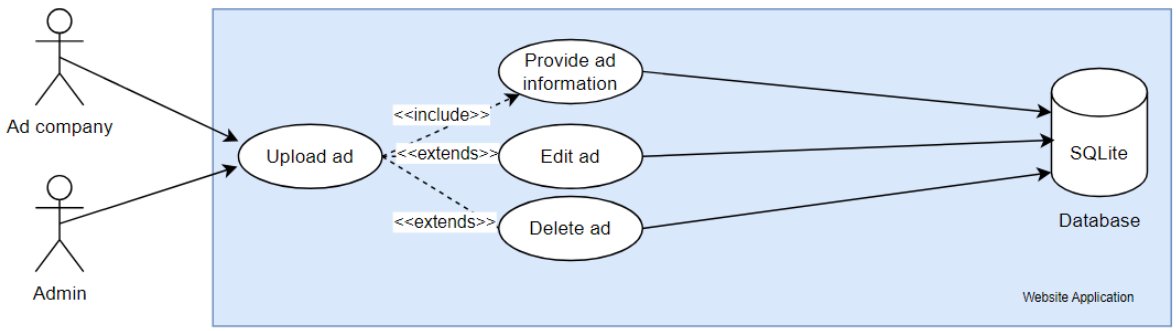
Use Case	Log in	ID	UC-2	Priority	High
Description	The user should be able to log in to make use of the website application such as download, upload, monitor and edit current ads or assets.				
Actors	Developer, Ad Company, Admin				
Precondition	A registered user to the application.				
Activation	The user case starts when the actors chose to log in to the application.				
Main flow	1. The actor selects the log in button. 2. The system prompts the user to enter their login credentials in a new page. 3. The user gets redirected to the home page.				
Alternate flow	A1: Invalid information provided. 3. The user provides wrong information such as unrecognized email address, or password. 4. The system will inform the user of the wrong information and to try again.				
Exceptional flow	E1: The application fails to log in the user due to timing out. 1. The connection times out due to connection issues 2. The user will be presented to a loading screen until the browser times out.				
Termination	The system presents the user with the main screen for each time of actor and now have the option to log out available				
Post Condition	The user is logged in to their account and able to use the application				

2.1.1.4 Requirement 3: Log out.



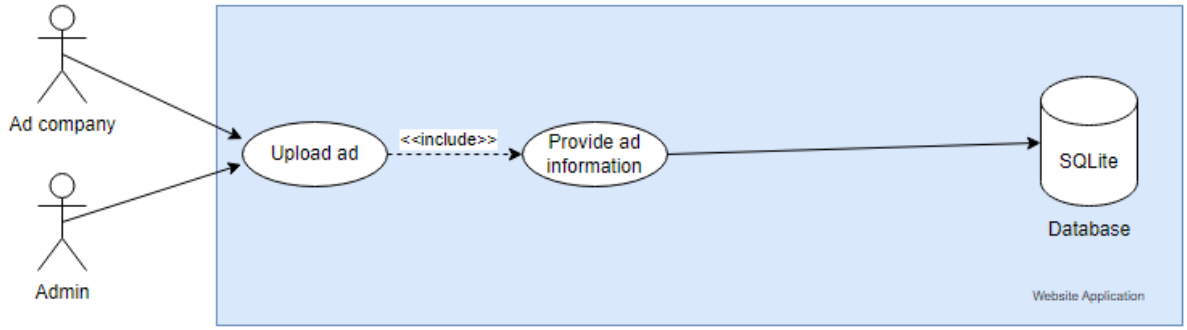
Use Case	Log out	ID	UC-3	Priority	High
Description	The user should be able to log out for security purposes and ensure no other party can use their account while they are away from the desk.				
Actors	Developer, Ad Company, Admin				
Precondition	A registered and logged in user.				
Activation	The user case starts when the actors chose to log out of the application.				
Main flow	1. The actor selects the log out button. 2. The system prompts the user to the main page just like any new and unregistered user will be prompted to.				
Alternate flow					
Exceptional flow	E1: The application fails to log out the user due to timing out. 1. The connection times out due to connection issues 2. The user will be presented to a loading screen until the browser times out. 3. User will still be logged in upon reconnection.				
Termination	The system is redirecting the user to the main page where they can log in again				
Post Condition	The user is no longer able to use the application.				

2.1.1.5 Requirement 4: Upload ad.



Use Case	Upload Ad	ID	UC-4	Priority	High
Description	The advertising company should be able to upload an advertisement in image format. This is important to the application for the add to be displayed in UC-11.				
Actors	Ad Company, Admin				
Precondition	A registered and logged in Ad Company or Admin user.				
Activation	The user case starts when the actors chose to upload a new advertising post.				
Main flow	1. The user selects the option to upload an Advertisement. 2. The system is then prompting the user to enter necessary information (UC-5) 3. The user is then able to view the ad where they can delete (UC-12), edit it (UC-13) or go back to view the main dashboard.				
Alternate flow					
Exceptional flow	E1: The application fails to upload the ad due to an error. 1. The user submits their ad information. 2. The system experiences and error either related to the connection or the file type. 3. The system informs the user of the error. 4. The user can try again.				
Termination	The system redirects the user to the main dashboard where they can view the analytics and the list of all ads they have added.				
Post Condition	The user can upload another application if they wish.				

2.1.1.6 Requirement 5: Provide ad information.



Use Case	Provide ad information	ID	UC-5	Priority	Medium
Description	The ad Company must be able to fill in and provide the necessary ad information both when creating it and editing it. This is crucial as the image or video will be displayed in UC-11.				
Actors	Ad Company, Admin				
Precondition	A registered and logged in Ad Company user.				
Activation	The user case starts when the actors chose to upload a new advertising post or edit an existing one.				

Main flow	1. The system is prompting the user to enter necessary information such as the title, description, URL link to redirect the user to, and the file upload. 2. The user is required to confirm that the ad they are uploading is safe for kids and made for all ages. 3. The user submits the information.
Alternate flow	A1: The user fails to provide required information. 1. The user fails to fill in necessary fields. 2. The system will alert the actor of their mistake. 3. The system will not save the changes and allow the user to try again. A2: The user provides a wrong file format. 1. The user provides an unaccepted file format. 2. The system will not recognize the file as one of the necessary ones and display an error asking the user to try again with the correct file type.
Exceptional flow	E1: The application fails to upload the ad due to an error. 5. The user submits their ad information. 6. The system experiences and error either related to the connection or the file type. 7. The system informs the user of the error. 8. The user can try again.
Termination	The system redirects the user to the main dashboard where they can view the analytics and the list of all ads they have added.
Post Condition	The user can upload or edit another application if they wish.

1. Requirement 6: Create Asset.

Use Case	Create Asset	ID	UC-6	Priority	High
Description	The admin must be able to create an asset that the Developer actor can use when implementing it into their VR environment (UC-9).				
Actors	Admin				
Precondition	A registered and logged in Admin.				
Activation	The use case starts when the admin selects the option to create a new asset on the web application.				

Main flow	1. The admin selects the option to add a new asset. 2. The system prompts the admin to enter additional information such as the title, description, file and image. 3. The admin then inputs the necessary information. 4. The system redirects the user to view the new asset they have created and have the option to edit, delete or go back to the main dashboard where they can view previously created applications.
Alternate flow	A1: The user fails to provide required information. 1. The user fails to fill in necessary fields. 2. The system will alert the actor of their mistake. 3. The system will not save the changes and allow the user to try again.
Exceptional flow	E1: The application fails to upload the asset due to an error. 1. The user submits their ad information. 2. The system experiences and error either related to the connection or the file type. 3. The system informs the user of the error. 4. The user can try again.
Termination	The system redirects the admin to the main dashboard where they can view the previously created assets.
Post Condition	The admin can now create additional assets or use any other functionality of the application.

2. Requirement 7: Select asset.

Use Case	Select asset	ID	UC-7	Priority	Medium
Description	The developer must be able to select an asset from the list of created assets by the admin, so that they can download them and upload them in their VR environment.				
Actors	Developer				
Precondition	A registered and logged in developer.				
Activation	The use case starts when the developer selects the option to use an asset on the web application.				
Main flow	1. The Developer selects one of the available assets. 2. The system redirects the user to provide any additional information required such as what environment will the asset be				

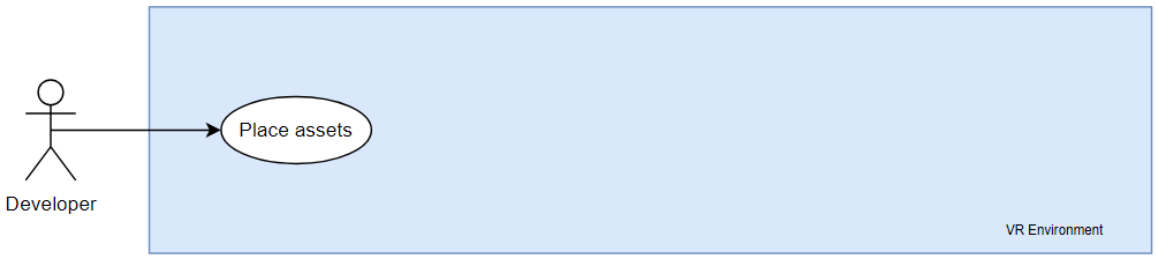
	used for and if they want to limit the asset to a specific category of ads such as clothing or food. 3. The system will also display information on how the asset can be placed and used. 4. The user can download the asset and place it in their environment in UC-10.
Alternate flow	A1: The user fails to provide additional information. 1. The asset will be set up with the default settings allowing every category of advertisements instead of limiting it to only the food or clothes industries as an example.
Exceptional flow	E1: The application fails to download the asset due to an error. 5. The user presses the file or download button. 6. The system experiences and error related to the connection. 7. The system will not be able to download the file. 8. The user can try again upon reconnection.
Termination	The system redirects the developer to the main page where they can select any other available assets
Post Condition	The developer can now use the asset and place it in their environment.

3. Requirement 8: Download asset.

Use Case	Download asset	ID	UC-8	Priority	High
Description	The developer must be able to download the set-up asset to be used in UC-10.				
Actors	Developer				
Precondition	A registered and logged in developer and a set up asset.				
Activation	The use case starts when the developer selects the option to download an asset.				
Main flow	1. The user has selected and finished setting up the asset and presses the download button. 2. The system will then put everything together and provide the user with a downloadable file.				
Alternate flow					
Exceptional flow	E1: The application fails to download the asset due to an error. 1. The user presses the file or download button. 2. The system experiences and error related to the connection. 3. The system will not be able to download the file. 4. The user can try again upon reconnection.				

Termination	The system redirects the developer to the main page where they can select any other available assets.
Post Condition	The developer can now use the asset and place it in their environment.

4. Requirement 9: Place asset.

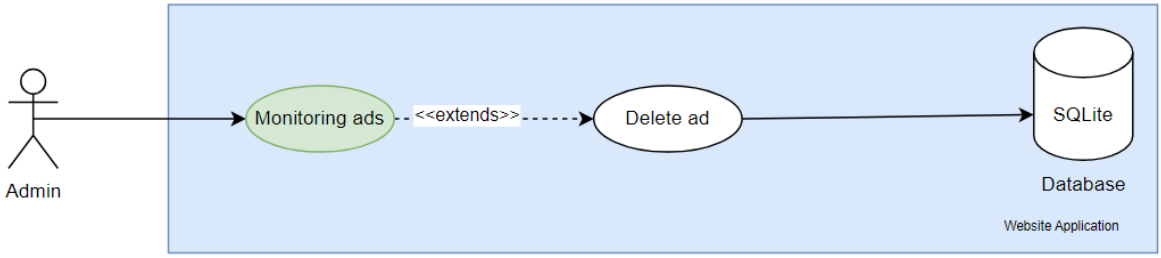


Use Case	Place asset	ID	UC-9	Priority	High
Description	The developer must be able to place the asset into their development engine so that they can be displayed with ads in UC-11.				
Actors	Developer				
Precondition	The developer must have an asset that was set up on the web application.				
Activation	The use case starts when the developer attempts to upload their asset in their development engine and VR environment.				
Main flow	1. The user uploads their file into their development engine using Connecter software. 2. They import their file into their connecter asset management. 3. The user then opens their development engine such as Unreal engine and select their project. 4. The user then right licks on the asset and selects “load in open UE project”. 5. The asset should then be loaded in their project directory. 6. The user can then drag and drop in the blueprint file into the world which will then be automatically filled in with an ad.				
Alternate flow	A1: The user is does not have the Connecter software. 1. The website application would inform the user in UC-8 about the required software and the user should be able to set it up within 10 minutes of learning how to use the software.				
Exceptional flow	E1: The application fails to load an add due to an error within the blueprint file. 1. The engine is unable to load the file as either the website has gone offline thus causing a connection issue, or the website was not able to load an add due to an error. 2. The asset will then display the default material applied to it such as the logo of the website.				
Termination	The asset successfully loads an ad to upon runtime				
Post Condition	The developer can now continue developing his environment or game or application without worrying about monetizing their game.				

5. Requirement 10: View ad

Use Case	View ad	ID	UC-10	Priority	High
Description	During the testing process of the developed application, the developer should be able to view an add upon runtime while simulating the user's experience.				
Actors	User				
Precondition	The user must be within the VR environment set up by the developer.				
Activation	This use case starts when the user runs the application and is in proximity with one of the custom assets placed by the developer with the ads.				
Main flow	1. The user is using the application developed by the developer. 2. The user comes across of one of the assets made by the admin and placed by the developer. 3. An advertisement is being placed in from the website into the asset within the VR environment based on a bidding system.				
Alternate flow	A1: The asset does not load an ad. 1. The asset is not ad for the user to view. 2. The asset will have a default logo of the VR-Adset application that will be loading instead.				
Exceptional flow	E1: The application fails to load an add due to an error within the blueprint file. 3. The engine is unable to load the file as either the website has gone offline thus causing a connection issue, or the website was not able to load an add due to an error. 4. The asset will then display the default material applied to it such as the logo of the website.				
Termination	The asset successfully loads an ad to upon runtime.				
Post Condition	The user can now continue to enjoy the content.				

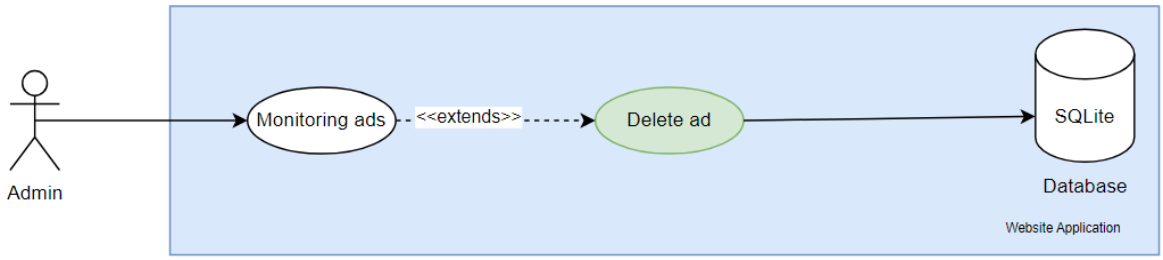
6. Requirement 11: Monitoring ads



Use Case	Monitoring ads	ID	UC-11	Priority	Medium

Description	The website should allow the admin to monitor the ads that are being uploaded and approve them or discard them in the event they are not appropriate. This is essential in the event of any complaint.
Actors	Admin
Precondition	The admin should have an account with full privileges.
Activation	This case starts when the admin selects an ad that they want to review.
Main flow	1. The admin opens an advertisement video or image. 2. Inspects the content. 3. if deemed to not be appropriate the admin can remove the advertisement from the database. 4. The ad company gets informed that their ad has been removed.
Alternate flow	
Exceptional flow	E1 The website does not load. 1. The website or page fails to load likely due to a connection issue. 2. The application will be in a loading state until reconnected with no data loss.
Termination	The admin can monitor and remove ads
Post Condition	The application can be used as usual by all parties.

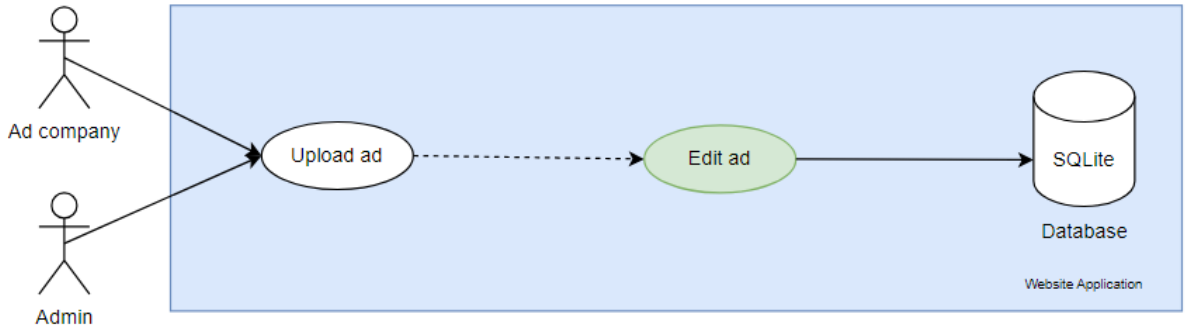
7. Requirement 12 Delete ad.



Use Case	Delete ad	ID	UC-12	Priority	Medium
Description	The website should provide the ability to delete an ad to both the admin and the ad company.				
Actors	Ad Company, Admin				
Precondition	Both the admin and the add company should have a registered account and an ad must be already created to be deleted.				
Activation	This case starts when either the advertiser or the admin presses the delete button on the ad.				
Main flow	1. The advertiser or the admin press the delete button. 2. The ad gets deleted. 3. The system will redirect the user to the advertisements view where they can choose any other ad to delete or view or edit.				
Alternate flow					
Exceptional flow	E1: The website fails to delete due to a connection issue. 1. The website fails to process the delete. 2. If the delete request has been sent before the connection got timed out the application will still remove the ad as intended.				

Termination	The ad will no longer be in the ads view as it has been removed.
Post Condition	The ad company and admin can continue to use the site as usual.

8. Requirement 13: Edit ad.



Use Case	Edit ad	ID	UC-13	Priority	Medium
Description	The website should provide the ability to edit an ad by the ad company.				
Actors	Ad Company, Admin				
Precondition	The add company should have a registered account and an ad must be already created.				
Activation	This case starts when either the advertiser views the add and selects the edit button.				
Main flow	1. The advertiser views an already existent ad or newly created one. 2. The advertiser opens the edit menu. 3. The advertiser edits the required field such as the description. 4. The newly updated field should be displayed in the ad view within the next screen as the system saves the changes.				
Alternate flow	A1: The updated fields have the wrong inputs provided. 1. The advertiser provides the wrong file type or does not meet the length requirement for a title. 2. An information popup will be displayed advising to change or update the fields as required. A2: The user leaves fields as blank 1. If the advertiser attempts to update a field by leaving it blank the system will inform the user of the error. 2. The system will not save any of the progress.				
Exceptional flow	E1: The advertiser closes the page while editing. 1. The advertising company is updating the field of one of their ads and accidentally close the tab. 2. Any change being made will not be saved and the ad will revert to the previous changes.				
Termination	The ad now has the updated fields				
Post Condition	The ad company can continue to use the site as usual.				

ii. Data Requirements

As part of the VR-Adset application the following information is required.

1. User data

For all users signing up to the website portal their email password and role will be required. In this case, the role is a multiple-choice dropdown where the user can select to either be an admin, advertiser, or developer. A user can only select to be an admin if they are the first user to sign up on the portal.

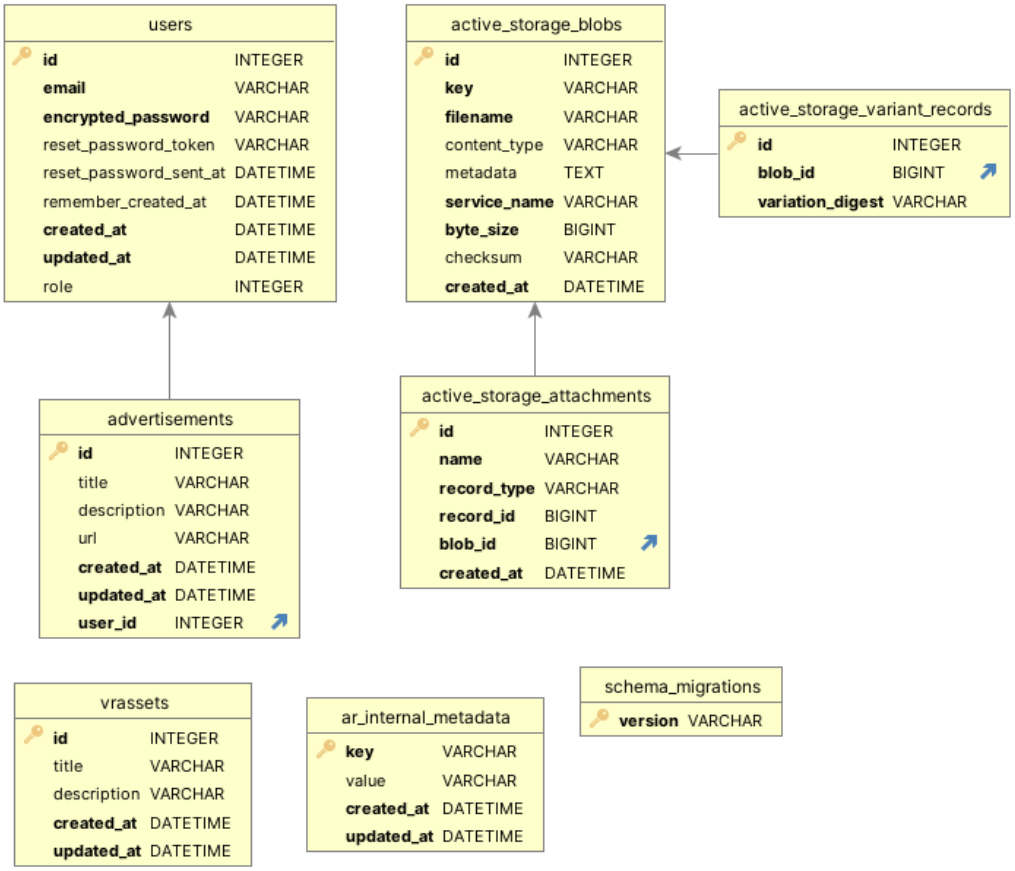
2. Advertisement information

When setting up an advertisement, the following information will be required; Title, description, URL to which the advertisement should redirect the end user to, and a file which in this case is an image.

3. Asset information

Creating assets will require a title, a description, a file upload for the 3D asset, and another file upload for the image of the asset.

Lastly, the data will be structured as seen in the Entity Relationship Diagram below.



Active Storage is the preferred method of managing and storing data within Rails applications. It works by saving the raw data into the table “active_storage_blobs” and linking it to the advertisement or VR asset in the table “active_storage_attachments”. The last table that Active Storage creates is “active_storage_variant_records” which saves variants of the files. Typically, an image can be formatted in a few different ways, which this table is responsible for saving that new format.

iii. User Requirements

Each user plays a different but crucial role in this application.

As part of my applications there are 4 types of users:

1. End user

The End user is the person who will see the advertisements within the VR environment. This user only needs to be able to run the environment or game that the developer has been setting up with the assets. The end user does not have to concern himself with signing up to the website platform or know how it works and how to use it.

2. Developer

Any Developer looking to use the application, will be required to first sign up to the website platform. Upon signing up, they will need to be logged in and download the assets provided within the “VR Assets” web page.

To be able to use the assets, the developers will require knowledge in using Connecter, which is an asset management tool. If they do not know how to use it, a 15 to 30-minute training session will be beneficial. Thanks to the YouTube channel VizualFlow, who put out a video titled “Unreal Engine Asset Management Tool”, anyone, with very little experience with Unreal Engine, can learn how to use it in approximately 6 minutes. (Vizualflow, 2023)

Upon downloading the assets and loading them into Connecter, the developer can easily import them into their Unreal Engine project and place the blueprint asset within their environment. Next, upon starting the application, the asset will automatically be filled in with an advertisement. This finalizes the developer’s required actions.

3. Advertiser

Advertisers will be required to sign up to the platform to upload advertisements and make use of the application. Upon signing up and logging in, the advertiser will be able to upload an advertisement providing the advertisement information such as the title, description, URL, and image. However, for the advertisement to be uploaded, the advertiser is required to comply with the terms and conditions by checking a box. This will ensure that the advertisement is safe for all ages. If all the above requirements are met, the advertisement will be uploaded, finalizing the advertiser’s required actions.

4. Admin

The admin, just like all other users, will be required to sign up and sign in. Their main priority is to ensure that the VR assets have been set up and uploaded properly to the platform. This will ensure that the developers are able to download the assets. Lastly, the admin is required to ensure that the advertisements are complying with the terms and conditions and deleting them otherwise.

iv. Environmental Requirements

This application will mainly be used on the cloud as a web platform. Consequently, a key requirement is maintaining continuous uptime, such as through an AWS EC2 cloud instance running the Ubuntu operating system. Additionally, the EC2 instance is also qualifies for the always free plan which provides 1 year of continuous cloud compute, which exceeds the duration of this project.

To interact and view the advertisements, the application will require the end user and developer to have access to a VR headset.

To run the VR environment at first the application will require a computer that is able to run Unreal Engine with consideration to their high hardware requirements. Additionally, the computer and headset must be connected to the same local network as the VR or linked via a cable to the computer. This is necessary for the application to be tested.

Due to the development of the VR environment being done in Unreal Engine, the application will not be limited only to the Oculus Quest 2 headset that I own and use for testing. The application will be able to run on any headset supported by Unreal Engine such as HTC Vive, Oculus Rift, Valve index and more. (Unreal Engine, n.d.)

Battery life of the Oculus quest 2 is not something that can easily be measured. The assets themselves in application do not use much memory or functionality. Compared to the full scale of a VR application, the usage that it has on the system is very much negligeable. The usage of the battery is simply determined by the type of application the developer is working on.

When it comes to security, it is primarily handled by Ruby on Rails which is known to be a very secure framework which implements a lot of security procedures by default. Additional gems will contribute to the login process such as Devise. Limiting the pages that the users can access is also a very important step in the security of the app. Therefore, users will have to be signed in to display certain information or functionality. Due to the very nature of the app where no sensitive information is aimed to be stored, there is little to no risk of any data leaks or identifiable information.

v. Usability Requirements

The application aims to be easy to use for all users involved. As such, there are two main web pages. One page for advertisements and one for assets where both advertisers and developers can easily access them.

From the end user's perspective, they shouldn't even be aware of what goes on behind the scenes, or how the advertisement gets displayed in VR. They should be able to enjoy the content designed by the developers whilst seeing the ads being placed around the environment.

For the developer, the application should be easy to understand and should have clear calls to action when they visit the “VR assets” page, such as a download button. The developer should also have access to a tutorial page where they can learn how to set up the asset in their VR environment. Next steps should be rather easy for the developer as they can add the asset within Connecter and easily import them into the currently running project with just a few clicks.

Advertisers should have a clear call to action on their “Advertisements” web page, to create a new advertisement. Following the instructions and adding all the required data for the advertisement, the webpage should update with the advertisement. This will then result in the ad being displayed in one of the assets placed within a VR environment.

b. Design & Architecture

VR-Adset brings advertisements within a virtual reality environment. This will be done using the website platform where advertisers upload their desired ad campaigns, and developers download the easy-to-use assets, uploading them to their desired environment.

To bridge the gap between the website and VR environments, the app needs to make use of routes that can be accessed using Hypertext Transfer Protocol (HTTP), in a format similar to an application programming interface (API) request. Consequently, the application is structured using Ruby on Rails framework.

Ruby on Rails adheres to the conventional Model-View-Controller (MVC) architecture. This separates the application into three components, which provide structure and help organizing the codebase, making it easy to maintain. In the case of VR-Adset application, there will be 2 main components which will be making use of the MVC architecture. These are “Advertisements” and “Vrassets”, which represent the advertisements web page and VR assets web page and their respective functionality.

The Model within the MVC architecture is responsible with managing the data that is to be stored in the SQLite3 database. For example, advertisements make use of the model to state that they will be storing a file, and that each advertisement created belongs to the advertiser which created them. Similarly, VR assets is also using the model to state that they will have a file attached.

The view is responsible with displaying information on to the web page and allowing any user to interact with the application through the use of buttons and text fields. For example, the view for advertisements will allow advertisers to interact with the website platform by pressing on buttons to create a new advertisement, and text fields to input information.

Controllers handle the functionality of the application. When a user interacts with any web page the controller will intercept the call from the view, handle the logic and functionality in the controller, and then calls upon the model to store or update the information. One simple example for this is when an admin ads a new asset to the website platform. The admin will fill in the information available from the view and press the submit button. The controller will verify that all fields have been filled in, followed by sending the information to the model which then stores it to the database.

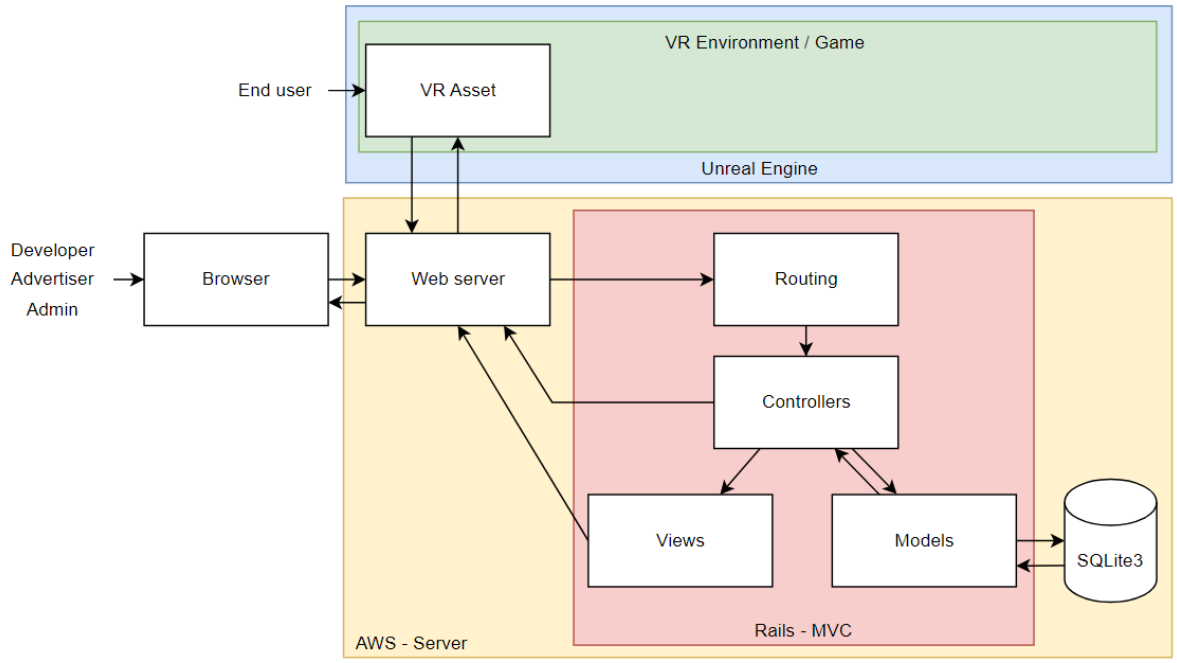
There are additional models, views and controllers required for the application. These are a general application controller and model which acts as a base controller where all controllers can share common functionality and help manage user authentication. Additionally, a user model is being used to define the role of users the application will accept, such as admin, advertiser, and developer. This

will also come with a custom view for Devise which is used to edit the sign-up form to allow the user to select their desired role. This essentially acts as a RESTful API where the specific GET request is designed to retrieve an image.

Devise is a gem available for Rails applications which provides a flexible authentication solution to manage the user sign-ups and log ins. Essentially, it provides functionality with safety features that allowed me to save time on development and focus on the safety features that matter.

For the VR asset to communicate with the website platform, a GET HTTP route would need to be established. This involves creating a URL, such as "https://vr-adset/get_image", that when accessed, would return an image path from any advertisement, which can be downloaded.

The process of selecting an image from the pool of advertisement images available, will make use of a randomised queue database. Where an algorithm would randomize all images and place them into a queue dataset. From here, each time the route get_image is being called, the first image in the queue will be removed. This process will repeat until the queue finishes where the queue will get filled again randomly, starting from the beginning. This is done to mitigate the issue where the same advertisement will be displayed when having multiple assets side by side.



The architecture diagram above shows how the application works when accessed form a browser.

1. The Developers, Advertisers and the Admin can access the web platform from any browser, such as Google Chrome. The browser then sends a GET request to the web server, e.g. "https://vr-adset/get_image".

2. The web server is responsible for handling HTTP requests and passing them to the Rails application.

3. Within the Rails application, the router is responsible for deciding which controller and action is being called upon examining the URL path. After the examination, the Router calls the appropriate action within the corresponding controller.

4. The Controller handles the actions of the method called upon by the router. It executes any functionality required and requests the data necessary from the model.

5. The Model then gets the data from the database and returns it back to the controller.

6. The controller will then call upon the view which will render the page that the user requested, e.g. the advertisements page. And send the HTML code over to the web server.

7. Finally, the web server will forward the rendered page onto the browser where it will be displayed to the users.

However, the process in which the End user interacts with the application is slightly different. Similarly, to accessing the application from the browser, the VR Asset sends a GET request to the web server, e.g. “https://vr-adset/get_image”. The same process repeats itself, except for rendering of a view. In this case, the controller only needs to send back an image from any advertisement. From here, the web server provides the VR Asset with the image requested, that then gets downloaded and display to the end user.

c. Implementation

The following section will cover the steps taken in developing the VR-Adset application with screenshots and explanations of the code and thinking process. Throughout the explanation, you are encouraged to follow along with the GitHub repository. Some of the screenshots cut through some comments or even other code to highlight the functionality being explained, and to make the text easily readable for this document.

2.3.1 Registration and security

One important aspect of the application is the registration and login functionality. This ensures that the users are given access only to the resources they need access to, fulfilling the security principle of least privilege. This is also done to aid the development of the application where the role of a user becomes crucial in developing a feature. An example of this is having an admin that can monitor the ads are complying with all the Standards for Advertising and Marketing Communications. (ASAI, 2024)

To implement the registration, log in, sign out, and password protection functionality, the Devise gem was used. However, as the application required the registration of 3 different types of users or, in my case roles, I had to create a dropdown in the sign-up page.

```
<%#  Devise - Dropdown for selecting the Role %>
<div class="field mb-3">
  <!-- Form Label -->
  <%= f.label :role, class: "form-label" %><br />

  <!-- If there are no users in the database, show "Admin" as an option in the dropdown -->
  <% if User.count == 0 %>
    <!-- Form select element within the dropdown named Role. -->
    <!-- User.roles.keys.map creates a hash map where the first value visible e.g. "Admmin", and second is value that gets submitted with the form e.g. "admin" -->
    <!-- https://stackoverflow.com/questions/24193814/select-enum-from-form-to-set-role -->
    <!-- https://api.rubyonrails.org/classes/Hash.html#method-i-assert_valid_keys -->
    <%= f.select :role, User.roles.slice(:admin).keys.map { |role| [role.titleize, role] }, { required: true } %>
  <% else %>
    <!-- Remove the "Admin" from the dropdown -->
    <!-- https://stackoverflow.com/questions/6227600/how-to-remove-a-key-from-hash-and-get-the-remaining-hash-in-ruby-rails -->
    <%= f.select :role, User.roles.except(:admin).keys.map { |role| [role.titleize, role] }, { required: true }, class:"form-select" %>
  <% end %>
</div>
```

The above snippet of code comes from the directory app\views\devise\registrations\new.html.erb. The code above lists a HTML form, written in embedded ruby, where a label for the “Role” is being rendered to specify what the dropdown is for. Following that, the dropdown is rendered however there can be observed a conditional statement checking the number of users in the database. If the user is the first user to sign up, then only display the option to be an admin. Otherwise, if there are other users in the database then only display the options of advertisers and developers. This ensures that only one admin can be created and that no other admins can be created by other users.

In addition to the view if statement, acting as a client-side check, within the user model there is another check which ensures the sign-up form was not modified by an attacker. In this case, the model acts as a server-side check, preventing anyone form signing up if there is a user in the database. This will also verify that the first user can only be an admin, as seen in the below snippets of code from the app\models\user.rb.

```
6
7     # Check that first_user_must_be_admin method before saving the user in the database
8     validate :first_user_must_be_admin, on: :create
9
```

```
14  # Mehtod that validates on the server side that the first user signs up as an admin.
15  def first_user_must_be_admin
16    # The first registered user has to be an admin.
17    # If there are no users in the database AND the user is trying to sign up as anything other than admin, an error will be displayed.
18    if User.count == 0 && !admin?
19      errors.add(:role, 'must be admin for the first user.')
20
21    # Only the first user can be an admin.
22    # If there are users in the database AND the user tries to sign up as an admin, an error will be displayed.
23    elsif User.count > 0 && admin?
24      errors.add(:role, 'cannot be admin. Contact the curent admin for assistance.')
25    end
26  end
27
28  has_many :advertisements
29 end
30
```

Distinguishing what role a user has, it is a functionality that had to be developed. Firstly, the database needed to store said role that the user is to have. To do so, I have created a simple migration, seen below, where I am adding a column to the users table with the role and value of integer.

```
db > migrate > 20240325131015_add_role_to_users.rb > ruby-symbols > AddR
1   class AddRoleToUsers < ActiveRecord::Migration[7.1]
2     def change
3       add_column :users, :role, :integer
4     end
5   end
6
```

The integer value was then mapped to a role. This was done in the users model, app\models\user.rb, which is the file responsible for the user authentication and registration. This means if the user is an admin, they will be assigned the integer 0, 1 if the user is an advertiser, and 2 if the user is a developer.

```
10    # Enum used to save the string into an integer in the database
11    enum role: { admin: 0, advertiser: 1, developer: 2 }
```

Lastly, for the roles to be fully integrated into the application, the controller app\controllers\application_controller.rb seen in the screenshot below, had to be modified. The

controller will make use of the Devise parameters to allow new attributes within the sign-up process. In this case, Devise will allow the role parameter selected from the dropdown mentioned earlier, to reach the model which will then be saved in the database.

app > controllers > application_controller.rb > ruby-symbols > ApplicationController < ActionController::Base

```ruby
class ApplicationController < ActionController::Base
  before_action :configure_permitted_parameters, if: :devise_controller?

  protected

  # Allow the use of role variable in as per devise documentation
  def configure_permitted_parameters
    devise_parameter_sanitizer.permit(:sign_up, keys: [:role])
  end
end

```

To further commit to fulfilling the security principle of least privilege, the following was done to limit what a role can have access to. Within the advertisement controller, the “before_action” filter is being applied which will run a method such as the “authenticate_user!”, before executing any controller actions. In the code snippet found below, the filter will check if the user is not logged in, which will limit their access to everything except the index route and the get_image route.

In this case, anyone should be able to access the index route which displays all advertisements, and the get_image route used by the VR assets to request an image from the database. As the asset is not a user, it does not have the ability to log in. Additionally, as the goal of an advertisement is to be seen by as many people as possible, having an attacker intercept this route and only be able to see an image, still achieves the end goal of the advertiser whilst keeping any other data private.

```ruby
class AdvertisementsController < ApplicationController
  include ApplicationHelper # Calls upone the application_helper.rb file where simple methods
  before_action :authenticate_user!, except: [:index, :get_image] #only display and show adve
  before_action :set_advertisement, only: %i[ show edit update destroy ]
  before_action :verify_permission, only: [:show, :new, :edit, :create, :update, :destroy]
```

In the method set_advertisement, the @advertisement variable is being set with an advertisement from the database, that matches the id provided in the request made to the server. E.g. https://vr-adset/1 for showing the advertisement and https://vr-adset/1/edit for editing.

```ruby
# GET /advertisements/1 or /advertisements/1.json
def show
  # limit so only the owner of the user can view the advertisement when searching for /advertisements/1
  @advertisement = Advertisement.find(params[:id])
  if user_advertiser? && current_user != @advertisement.user
    redirect_to advertisements_path, alert: 'You are not authorized to view this advertisement.'
  end
end
```

One other functionality that can be observed in the code snippet above, is the if statement where an advertisement needs to belong to a user that is currently signed in. This will prevent advertiser 1 from accessing the advertisements of advertiser 2.

Lastly the controller checks to see if the user has permission to perform the actions listed in the square brackets. A user that is not an advertiser or an admin will not be able to access the show, new, edit, create, update and destroy routes. This check can be observed in the code provided below.

```
103
104     # Check if the user is an advertiser or admin
105     def verify_permission
106       unless user_advertiser_or_admin?
107         redirect_to advertisements_path, alert: "You are not authorized to perform this action."
108       end
109     end
```

A similar approach was done for the VR assets where only the admin should be able to create, edit and delete them, as seen below.

```
class VrassetsController < ApplicationController
  before_action :authenticate_user!, except: [ :index ] # Display VR assets to users who are not logged in.
  before_action :set_vrasset, only: %i[ show edit update destroy ] # Method used for the CRUD function of the VR asset.
  before_action :verify_user_is_admin, only: [:show, :new, :edit, :create, :update, :destroy] # Only admins should be able
```

```
# Verify that the user is admin
def verify_user_is_admin
  unless user_admin?
    redirect_to vrassets_path, alert: "You are not authorized to perform this action."
  end
end
```

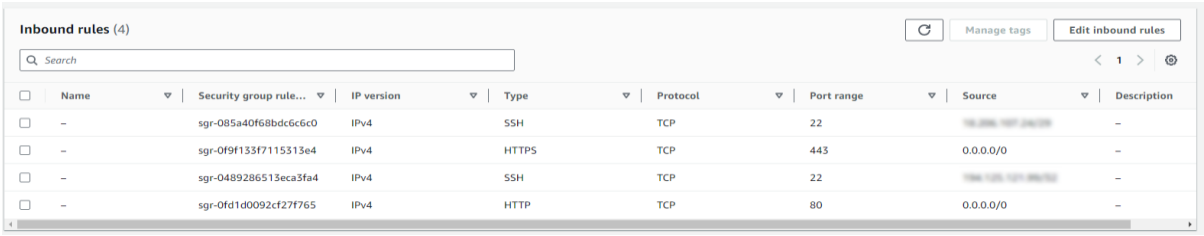
It can be observed that the method names used “user_admin?” or “user_advertiser_or_admin” are all custom-made methods from the app\helpers\application_helper.rb. These methods are only written to simplify the readability of the code and shorten the checks.

app > helpers > application_helper.rb > ...

```
1   module ApplicationHelper
2     # Helper used to shorten the IF function within the views and some controllers.
3
4     #ADMIN
5     def user_admin?
6       user_signed_in? && current_user.admin?
7     end
8
9     # ADVERTISER
10    def user_advertiser?
11      user_signed_in? && current_user.advertiser?
12    end
13
14    def user_advertiser_or_admin?
15      user_signed_in? && (current_user.advertiser? || current_user.admin?)
16    end
17
18    # DEVELOPER
19    def user_developer?
20      user_signed_in? && current_user.developer?
21    end
22
23    def user_developer_or_admin?
24      user_signed_in? && (current_user.developer? || current_user.admin?)
25    end
26
27  end
28
```

Ultimately, what all the above changes do is limit users to the following. A developer has access to create an asset and advertisement, but they can download an asset. An advertiser role limits them from being able to create or download an asset, however they can create, edit, view and delete their own advertisements.

An admin can create, edit, view, and delete both and advertisement and an asset, thus having access to every functionality that the website platform provides. However, an admin can only be created only when the application is first run or with the manual manipulation of the database. This not only requires access to my personal AWS account but also have to fulfil the following security groups set within the launch wizard. The screenshot below shows how I have set up 2 security rules which limit the connection via SSH only to my personal IP and the general area allowing me to connect using the browser and the Visual Studio Code terminal.



2.3.2 Data and Active Storage

Both advertisements, and VR assets require a file upload. Advertisements require the upload of one image file, whilst VR assets require both an image and a file upload, used for the asset. To implement the upload functionality, I made use of the Active Storage framework within Ruby on Rails.

As Active Storage is the preferred method of managing storage within Rails applications, setting it up was a process that took a few iterations to get right. First, the Active Storage gem was added to the gem file, installed, and running the “rails db:migrate” command. With the addition of the lines below into my models, the database is now ready to save and link files with the advertisements and VR assets.

```
app > models > vrasset.rb > ruby-symbols > Vrasset < ApplicationRecord
1   class Vrasset < ApplicationRecord
2       has_one_attached :file # https://edgeguides.rubyonrails.or
3       has_one_attached :image # This is the image of the asset
```

```
1  class Advertisement < ApplicationRecord
2      has_one_attached :file # Active storage - Declares that there is a file to be attached
3      belongs to :user # This line declares that an advertisement belongs to one user
4      validates :title, :description, :url, :file, presence: true # Ensures that the following
```

In addition to the file attachments, it can be observed above that in the case of advertisements, they are also linked to a user. To implement such functionality, it required the creation of a migration so that the user id who created an ad, can be attached to the advertisement. This can be seen in the next screenshot.

```
db > migrate > 20240424125148_add_user_id_to_advertisements.rb > ...
1   class AddUserIdToAdvertisements < ActiveRecord::Migration[7.1]
2     def change
3       add_reference :advertisements, :user, null: false, foreign_key: true
4     end
5   end
```

To allow an advertiser to upload an image, the advertisements view needed to be modified to display a file input box which can be seen below. This code snippet comes from app\views\advertisements_form.html.erb.

```
<div class="mb-3">
  <%= form.label :file, class: "form-label" %>
  <%= form.file_field :file, class:"form-control", accept:'image/jpeg,image/png'%>
</div>
```

This restriction was introduced as a result of usability testing, where one of the users that I have introduced to the program broke some functionality by uploading a file format that wasn't expected. More information on this can be found in the Testing section under Usability tests.

To fully ensure there is no data being inserted into the application using other methods but the browser access, I have also added the following lines into the advertisement model, app\models\advertisement.rb.

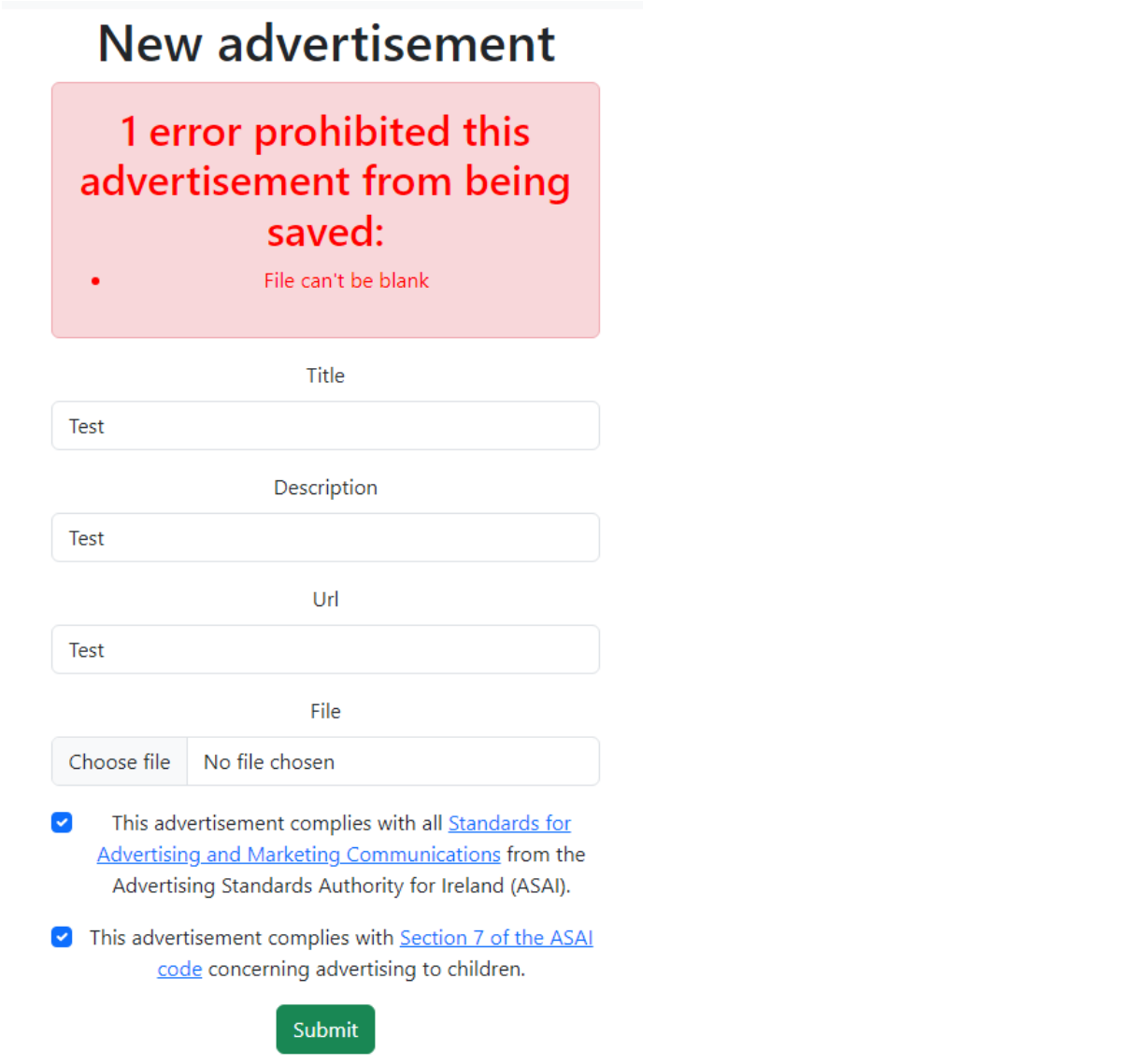
```
14      # Check if the image uploaded is either a JPEG or PNG format.
15      def correct_image_type
16          if file.attached? && !file.content_type.in?(%w(image/jpeg image/png))
17              errors.add(:file, 'must be a JPEG, OR PNG')
18          end
19      end
20  end
```

In addition to the file limitations, both the model and the view will check to see if all the fields such as title, description, URL, and file are present when submitting the form. Effectively creating both a client-side check and a server-side check.

```
4       validates :title, :description, :url, :file, presence: true #
```

The following snippet of code comes from app\views\advertisements_form.html.erb. Here it can be observed how every text field was required to be filled in except for the input check box. This is intentional, as it allows users to edit an advertisement without being required to resubmit an image on the client side. However, they are still required to at least have an image in the database as per the advertisement model. This essentially creates an easier process for the advertiser who no longer has to re-upload and find their original image, just because they wanted to change the title of the advertisement. However, if the advertiser does not upload an image when creating and advertisement, then they will be prompted with a server-side error message.

```
16      <div class="mb-3">
17        <%= form.label :title, class: "form-label" %>
18        <%= form.text_field :title, class:"form-control", required: true %>
19      </div>
20
21      <div class="mb-3">
22        <%= form.label :description, class: "form-label" %>
23        <%= form.text_field :description, class:"form-control", required: true %>
24      </div>
25
26      <div class="mb-3">
27        <%= form.label :url, class: "form-label" %>
28        <%= form.text_field :url, class:"form-control", required: true %>
29      </div>
30
31      <div class="mb-3">
32        <%= form.label :file, class: "form-label" %>
33        <%= form.file_field :file, class:"form-control", accept:'image/jpeg,image/png'%>
34      </div>
```



2.3.3 Virtual Reality implementation

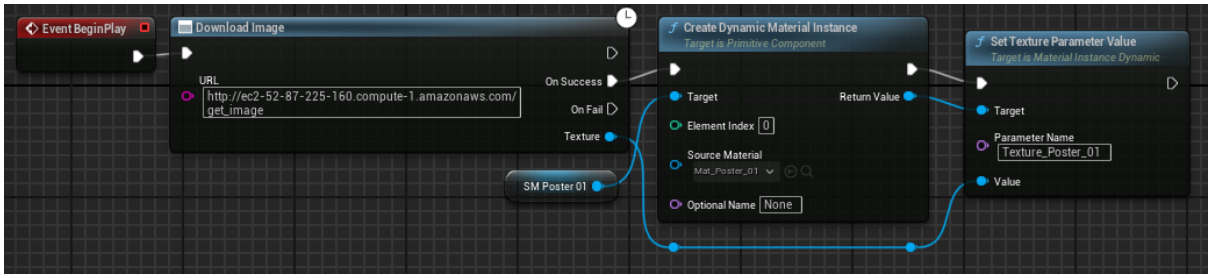
For the advertisements to appear in the VR environment, first a GET route needed to be open and ready to receive requests. Secondly the route should be able to redirect the user to the location of one of the images, where it can be downloaded by the asset. The snippet of code below demonstrates how the application selects and responds back with an image.

```
6
7   # Class variable array of adcertisements with attached files(images) from the database that are in a random order.
8   # A class variable is used to keep the state upon each request. Otherwise on each request the variable will reset.
9   @@image_queue = Advertisement.with_attached_file.order('RANDOM()').to_a
```

```
21
22  # GET /get_image - Used to request an image from Unreal Engine
23  def get_image
24    # If the image queue is empty then refill the queue
25    @@image_queue = Advertisement.with_attached_file.order('RANDOM()').to_a if @@image_queue.empty?
26
27    # Removes the first image from the queue whilst holding the first in the advertisement variable
28    # https://www.geeksforgeeks.org/ruby-queue-shift-function/
29    advertisement = @@image_queue.shift
30
31    # if there is a file attached, redirect to the blob with said file.
32    # https://stackoverflow.com/questions/50424251/how-can-i-get-url-of-my-attachment-stored-in-acti
33    if advertisement&.file&.attached?
34      redirect_to rails_blob_url(advertisement.file, disposition: "attachment", only_path: true)
35    else
36      head :not_found # else display not found message
37    end
38  end
```

First, an array of advertisements with attached files is being pulled from the database and sorted in a random order. Then when the GET request is called from the “get_image” route, the queue is checked to be refilled in the event many requests have been made, removing all the advertisements. Following this, an advertisement from the image queue is being pulled out and saved in the “advertisement” variable. This will finally be used to redirect the user to the location of the attached image. This process was done to prevent the chance of providing the same image to the assets which when placed side by side in an environment, could display the same image twice or even 3 times in a row.

For the assets to display any image, they need to be set up with the correct materials first, and be a actor which allows codding of functionality in a blueprint format. The following screenshot displays how the image is being placed to the material of an object. In this instance, a poster is used to demonstrate how it works.

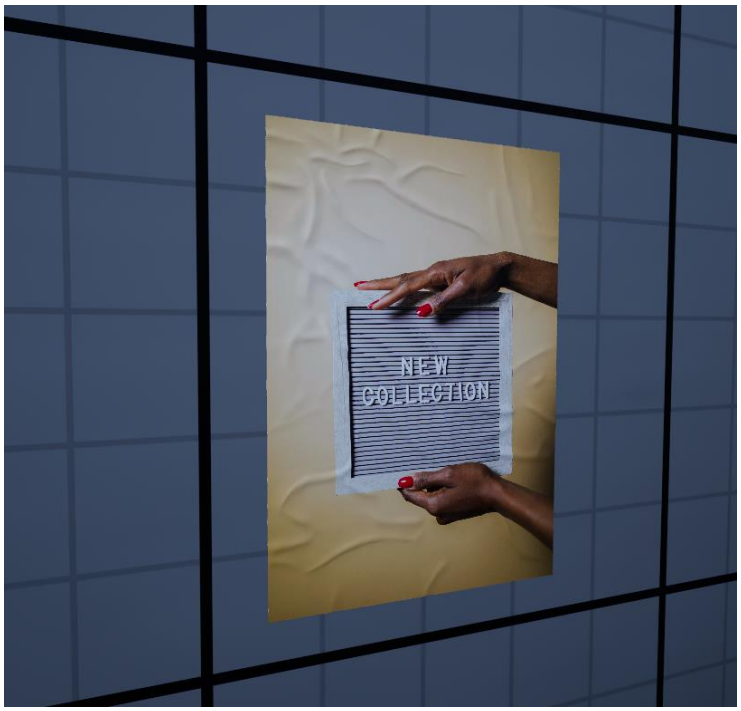


First, there is an event “Begin Play” which waits for the application to start in order to execute the following methods to the right. Once the application started, the “Download Image” method makes a request to the website by using the “get_image” route. Then it will attempt to download the image

given by the application. If successful, it will redirect the “Create Dynamic Material Instance” method, followed by the “Set Texture Parameter Value”.

The way these two methods work is similar to replacing a painting on a wall with a modern slim TV, with the same purpose of displaying a painting. However, you now have the option to easily change the paintings to your liking, making it dynamic. That is what the “Create Dynamic Material Instance” method does.

The “Set Texture Parameter Value” is similar to the remote you would use to replace the image displayed on the TV. In this case however, the method brings the image downloaded from the website, and sets the default material of the poster to change with the new image.



2.4 Graphical User Interface (GUI)

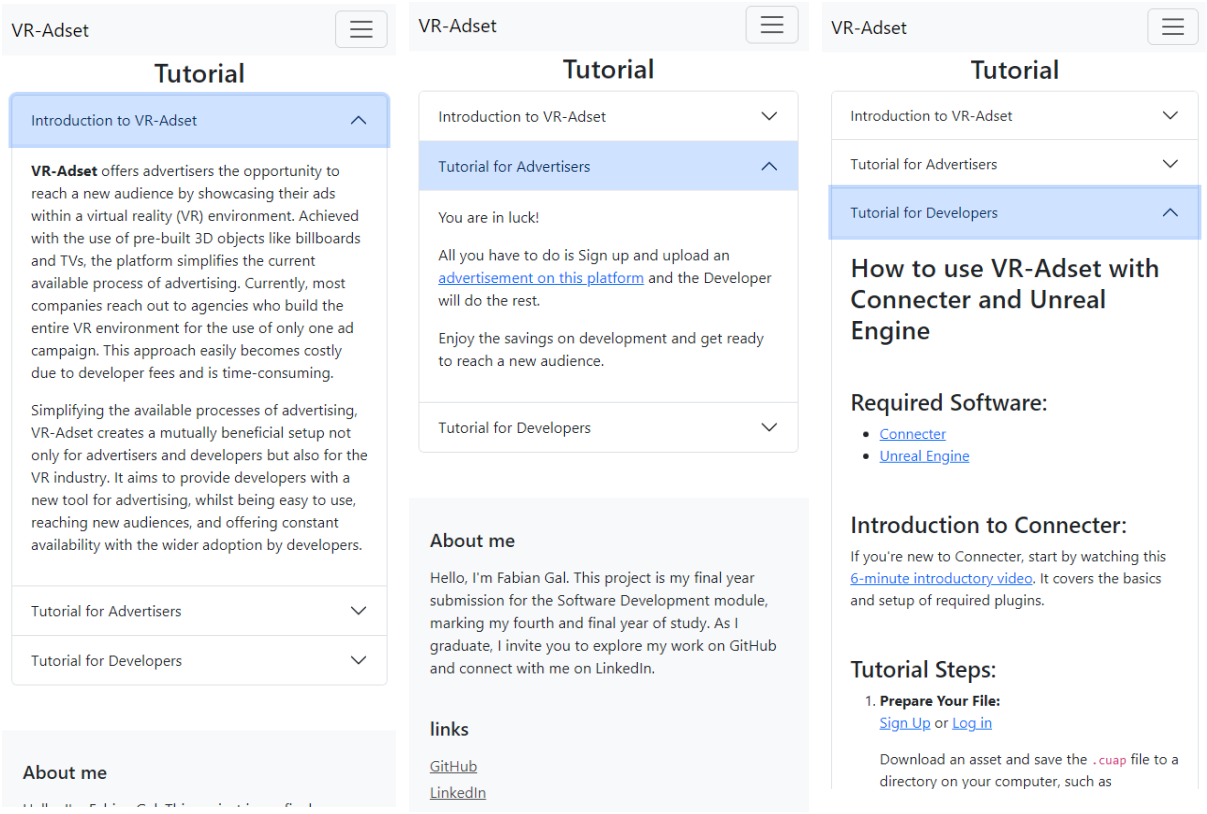
This section will be covering all the GUI views that a user can interact with.

It is worth mentioning that throughout the designing process of the user interface, I took a mobile first approach. This means that this application is very much suited to be used on mobile phones as well as on any other device due to its responsiveness to any device size. In order to optimise the space within this document, I will be showing the mobile format primarily, with some screenshots for the desktop sizes.

It is also worth noting that all pictures used for the advertisements, are taken from Pexels which are completely free to use and edit. (Pexels, 2024)

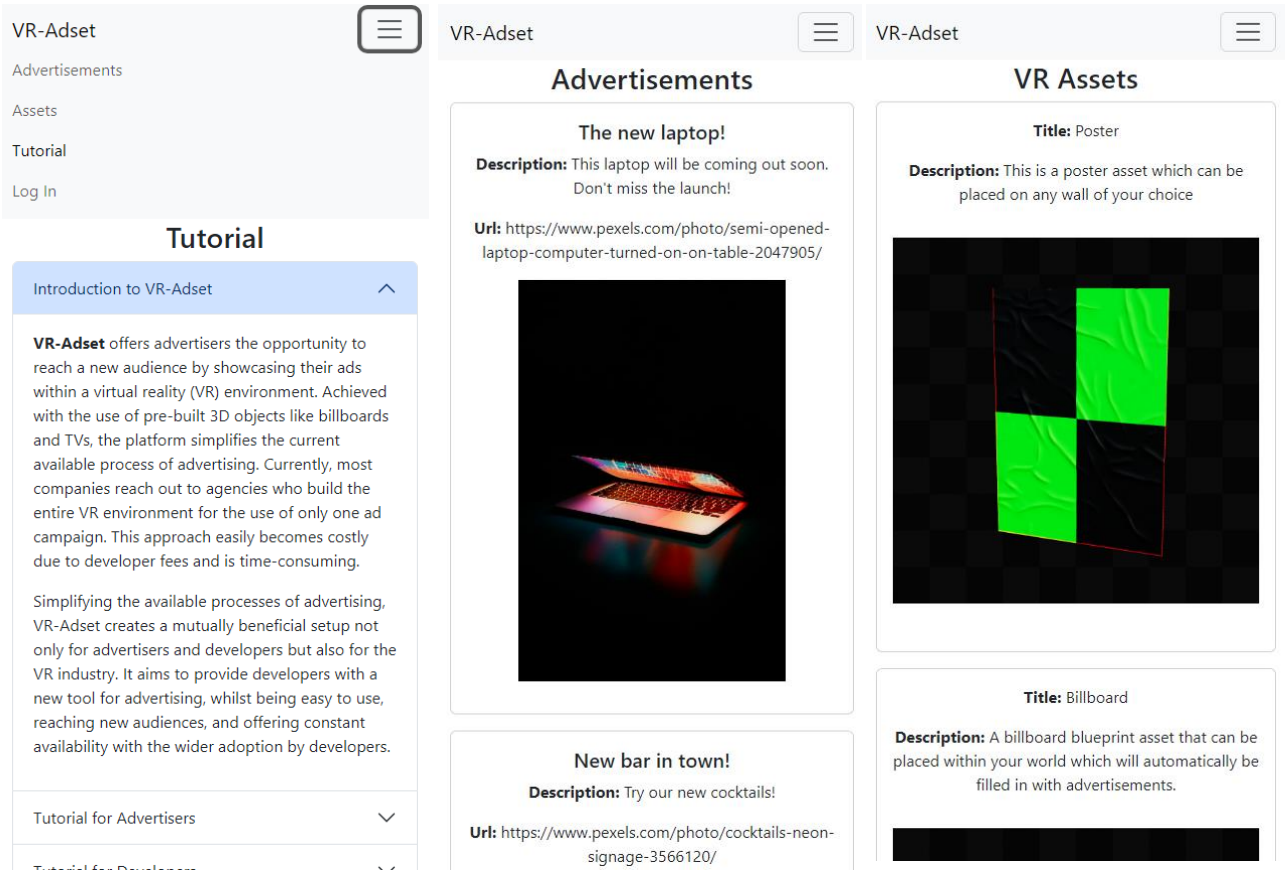
2.4.1 Signed out user

When a user first visits the app they will be shown a general introduction in addition to two tutorials. These tutorials are in an accordion component so that the text is easy to read and only to the interest of the user. A developer for example might only be interested in how to get use VR-Adset within their project so they might skip right away to that section.

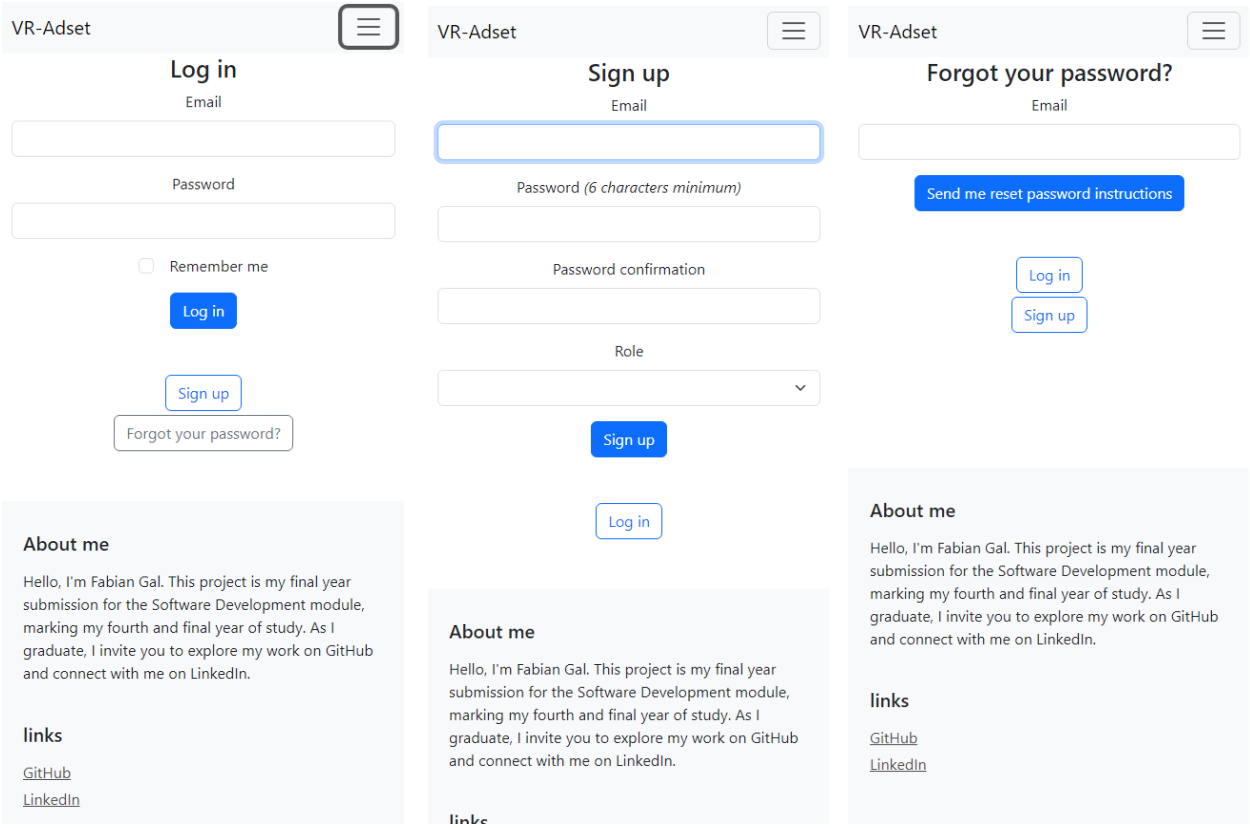


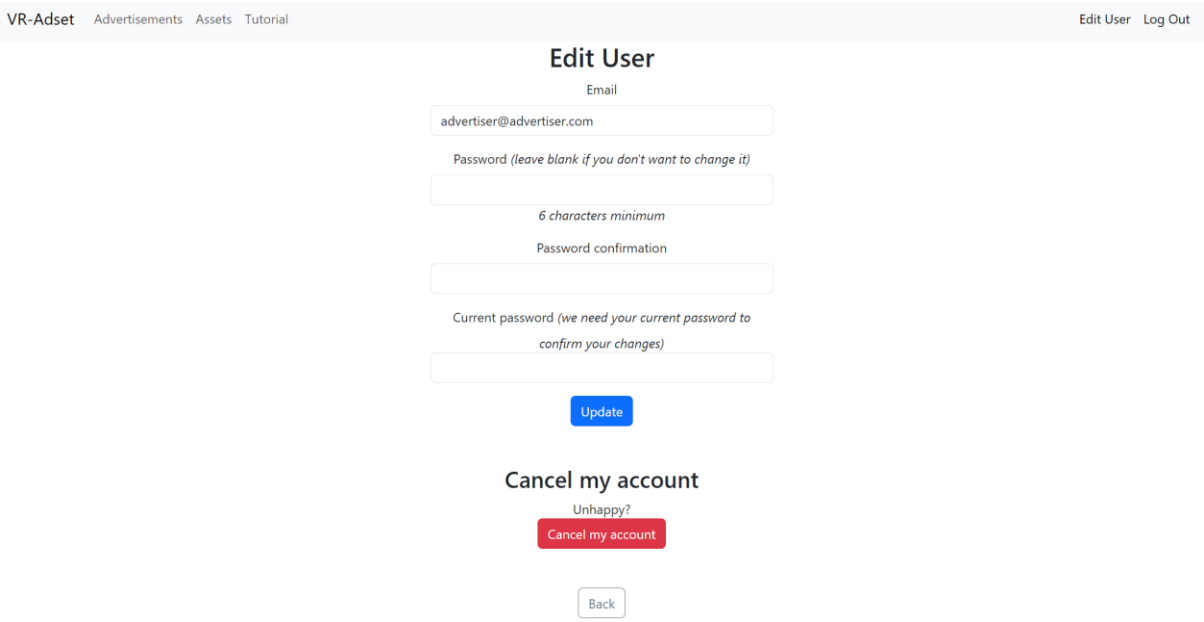
A signed-out user also has the option to navigate to the advertisements, assets, and log-in page.

One of the intended behaviours of the application was to have all the user be able to see the advertisements available which benefits the advertisers. However, they are not able to edit the advertisement in any way without signing in.



2.4.2 Log-in, Sign-up, Forgot Password, Edit user

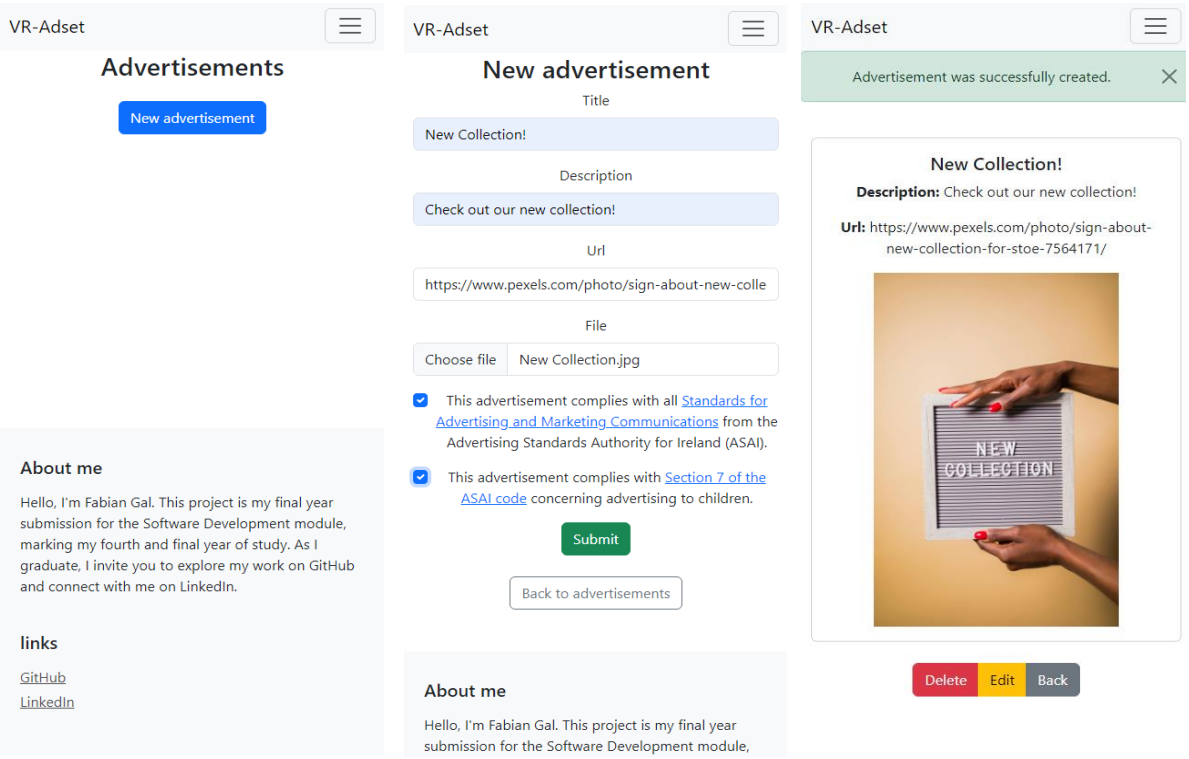




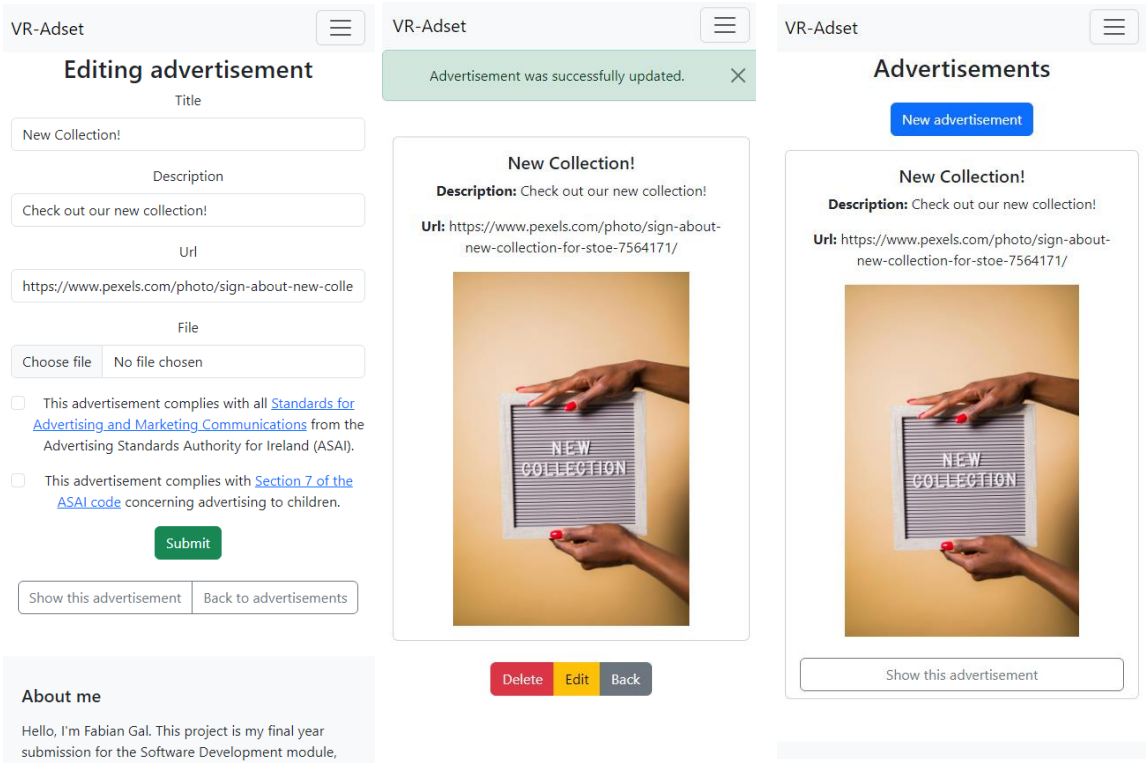
After a user sign up to the platform, they will unlock access to functionality related to their roles. E.g. Advertisers can create advertisements and Developers can download assets.

2.4.3 Advertisers

Upon a successful registration an advertiser has access to add new advertisements.

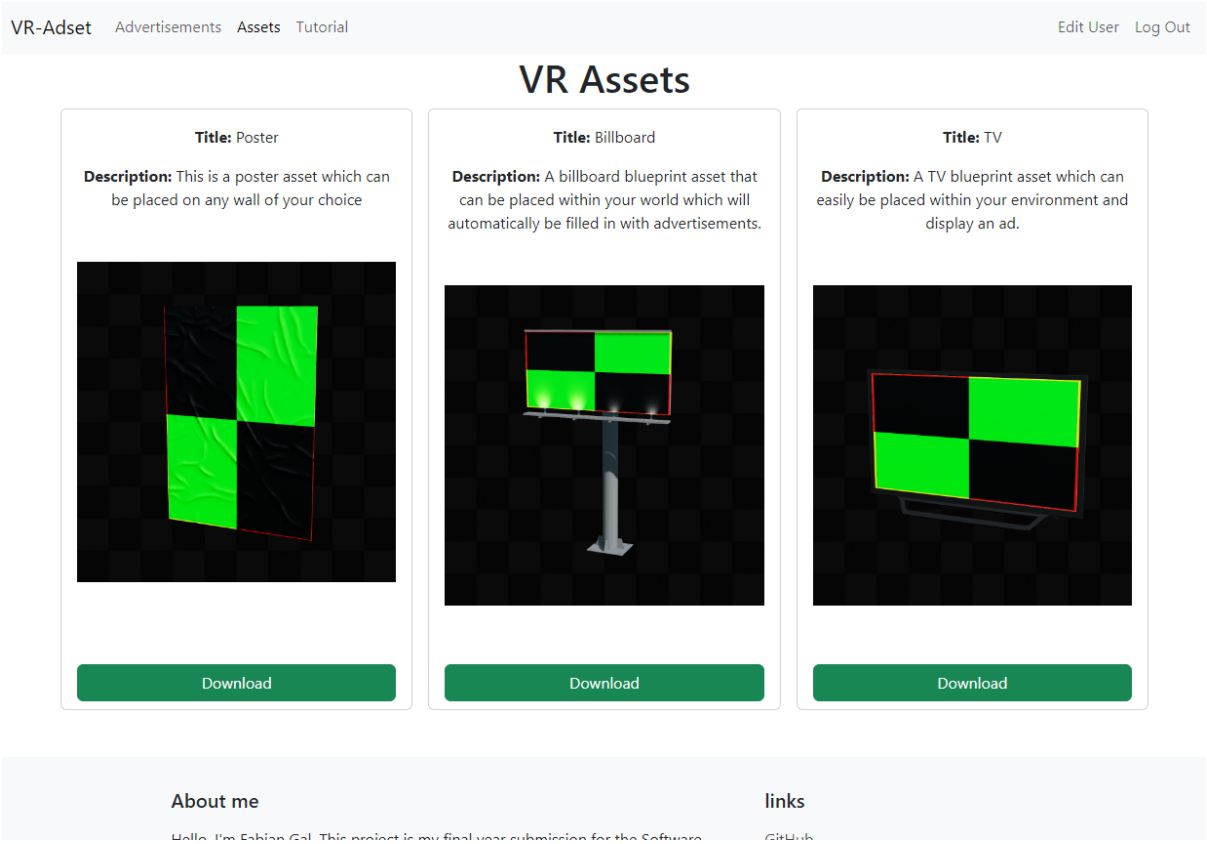


The advertiser will then get redirected to the view page of the advertisement they just created. From here the advertiser can edit the advertisement or go back to the advertisements.



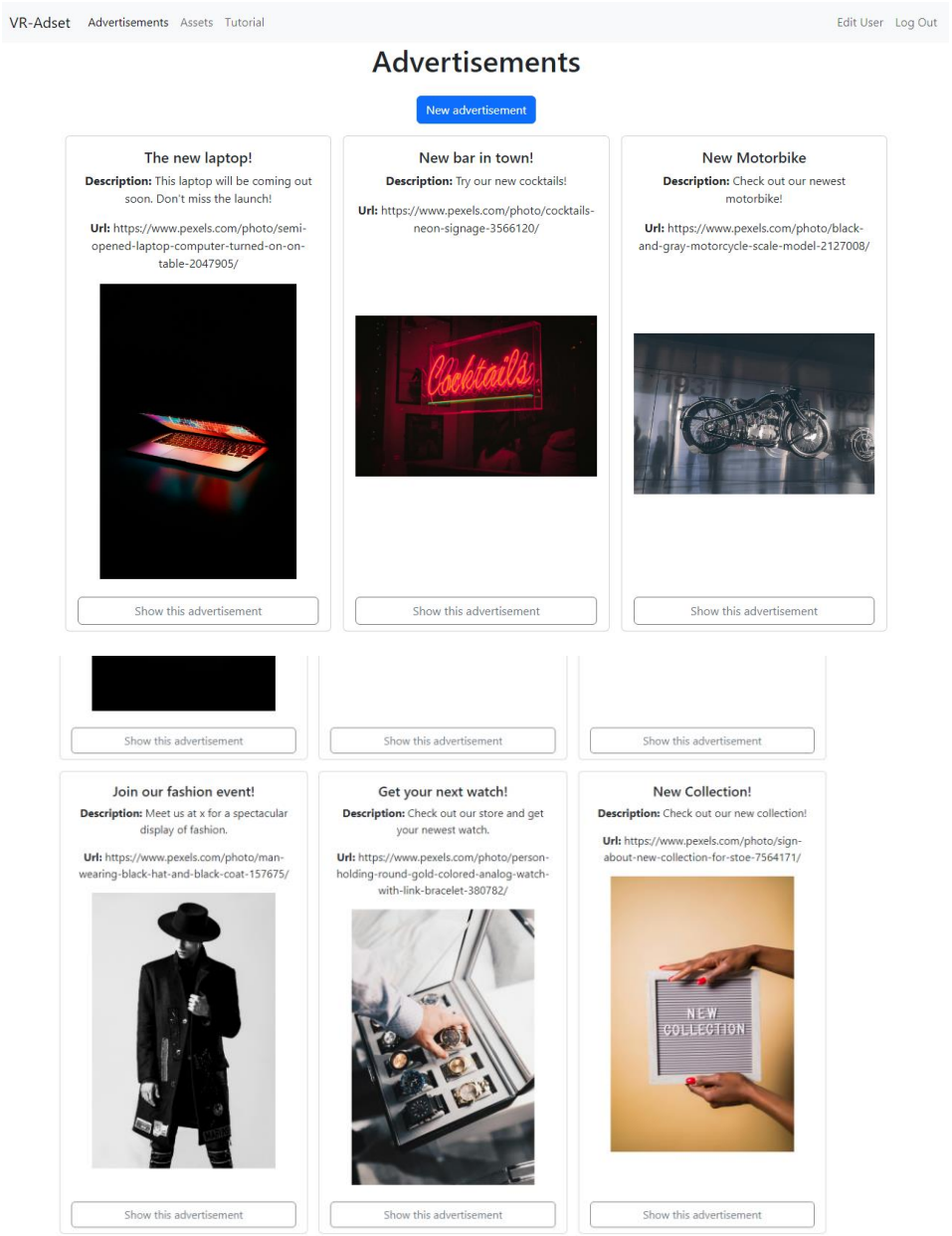
2.4.4 Developers

Upon successfully signing up the developers unlock the functionality to download any of the available assets.

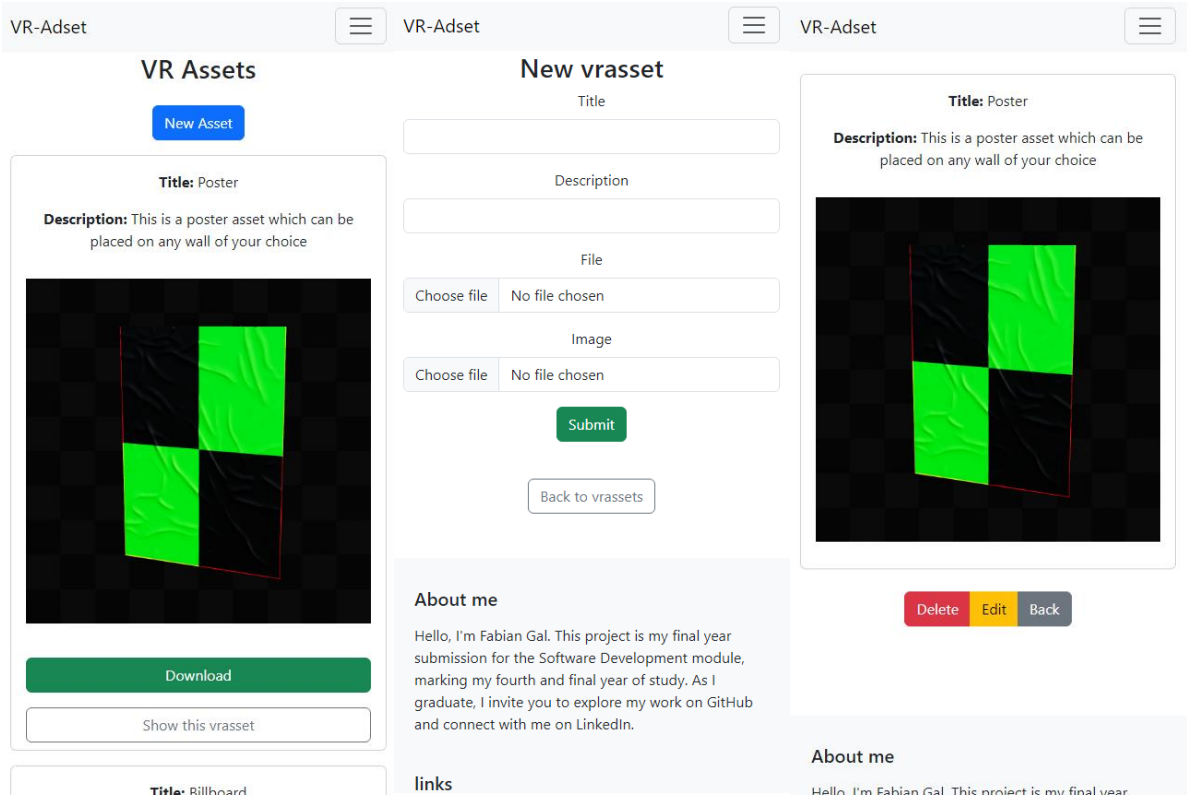


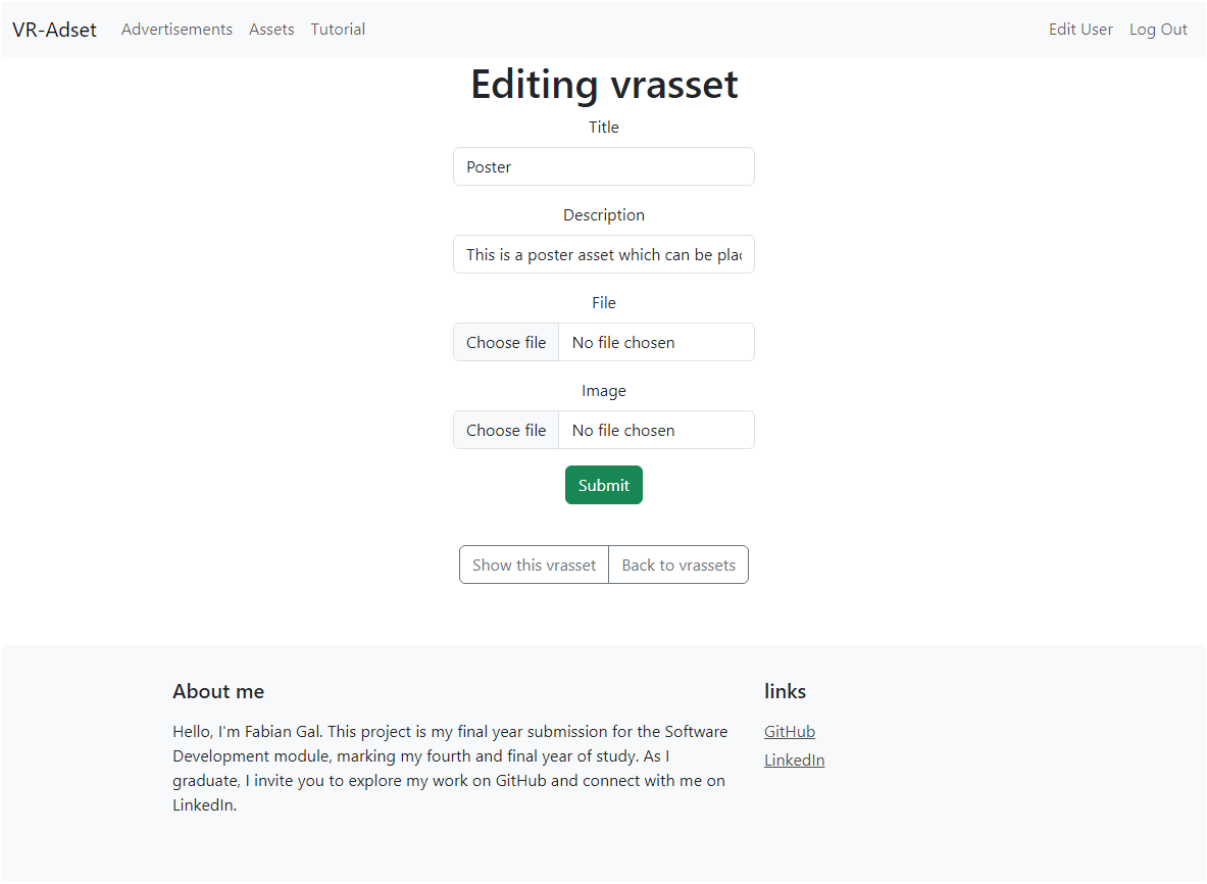
2.4.5 Admin

The admin has access to all the functionality an advertiser does. This includes being able to view all advertisements created and deleting any advertisements that do not follow the ASAI standards.



Additionally, the admin is responsible for creating any asset and uploading them. Therefore, they have access to the following GUIs.


VR-Adset
VR Assets
New Asset
Title: Poster
Description: This is a poster asset which can be placed on any wall of your choice
Download
Show this vrasset
Title: Billboard
VR-Adset
New vrasset
Title
Description
File
Choose file
No file chosen
Image
Choose file
No file chosen
Submit
Back to vrassets
About me
Hello, I'm Fabian Gal. This project is my final year submission for the Software Development module, marking my fourth and final year of study. As I graduate, I invite you to explore my work on GitHub and connect with me on LinkedIn.
links
VR-Adset
Title: Poster
Description: This is a poster asset which can be placed on any wall of your choice
Delete
Edit
Back
About me
Hello, I'm Fabian Gal. This project is my final year


VR-Adset
Advertisements
Assets
Tutorial
Edit User
Log Out
Editing vrasset
Title
Poster
Description
This is a poster asset which can be pla
File
Choose file
No file chosen
Image
Choose file
No file chosen
Submit
Show this vrasset
Back to vrassets
About me
Hello, I'm Fabian Gal. This project is my final year submission for the Software Development module, marking my fourth and final year of study. As I graduate, I invite you to explore my work on GitHub and connect with me on LinkedIn.
links
GitHub
LinkedIn

2.5 Testing

To test the VR-Adset application, the following techniques have been employed. Manual testing, unit, integration, system, functional, usability, and security testing.

2.5.1 Manual testing

Manual testing was performed after each new feature to see the impact of the feature on the program, especially in the beginning where the features were limited. Most of the security flaws were detected using manual testing. Additionally, a majority of the testing done within Unreal Engine was manual testing with every change made to the VR assets or the routes within Rails.

As the scope of the project and features increased, automated techniques had to be employed.

2.5.2 Unit testing

Currently the VR-Adset uses Unit testing to test the “correct_image_type” method used in the app\models\advertisement.rb file. This would check to ensure that the image uploaded when creating an advertisement is a PNG or a JPEG format.

The test can be found in test\models\advertisement_test.rb, as seen below in the attached screenshot.

This test makes use of two files from the fixtures folder at test\fixtures\files, where one file is a PNG, and the other is CUAP, which is an asset file. This test would prepare two advertisements with their respective values also coming from the fixture folder however, this time from test\fixtures\advertisements.yml. These two advertisements will then each be assigned a file within the setup phase of the test. The first advertisement will be provided with a PNG, whilst the second advertisement will be provided with the CUAP file.

The first test “should check image is png” is checking for any errors within the file of the advertisement and if there are none, then it will pass the test. Otherwise, if there are errors the message “There should be no errors on the file for PNG image” will be displayed.

The second test, however, checks to see that there are errors when uploading a CUAP file, which is not a supported format. This test should also pass considering the file is not permitted and the test checks for any error which there is.

Running this test with the command “rails test test\models\advertisement_test.rb” generates the following results.

```
# Running:

..

Finished in 0.609150s, 3.2833 runs/s, 6.5665 assertions/s.
2 runs, 4 assertions, 0 failures, 0 errors, 0 skips
```

This dictates that the test successfully passed.

2.5.3 Integration testing

In contrast to unit testing, integration testing, has been implemented from the very start of the application. They test the controllers and models and how it retrieves data and primarily focus on how the routes.

The setup for this test uses all 3 roles, admin, advertiser and developer to test the functionality of the application. Then as a default, signs in the user admin to test functionality such as being able to get the index page, access the creation, and all other functionalities that an admin can access, for both new advertisement and VR assets.

As these tests span over 200 lines of code, the below attached screenshots will only be snippets and would encourage the reader to follow along with the GitHub repository.

The following test snippet comes from test\controllers\advertisements_controller_test.rb.
The fixtures mentioned in the below code snippet, can all be found in the test\fixtures folder.

```ruby
class AdvertisementsControllerTest < ActionDispatch::IntegrationTest
  setup do
    # Graps the users from the users fixtures
    @admin = users(:admin)
    @advertiser = users(:advertiser)
    @developer = users(:developer)

    # Signs in the admin to allow for general tests to pass. Where user needs specified sign_in method will be used.
    sign_in @admin #This makes use of the Devise helper to sign in the user.

    # Prepares the advertisemetns with the values from the fixtures
    @advertisement = advertisements(:one)
    @advertisementTwo = advertisements(:two)

    # As a file cannot be added within the fixtures due to the yml format, I am adding it here.
    @file = fixture_file_upload("Advertisement_test.png", "image/png")
    @advertisement.file.attach(@file)
    @advertisementTwo.file.attach(@file)
  end
```

```
23    test "user is signed in" do
24      sign_in @admin
25      get root_url
26      assert_response :success
27    end
28
29    test "should get index" do
30      get advertisements_url
31      assert_response :success
32    end
33
34    test "should get new" do
35      get new_advertisement_url
36      assert_response :success
37    end
38
39    test "should create advertisement" do
40      # The advertisement count should increase
41      assert_difference("Advertisement.count") do
42        post advertisements_url, params: { advertisement: {
43            description: @advertisement.description,
44            title: @advertisement.title,
45            url: @advertisement.url,
46            file: @file, # This requires the fixture_file_upload method to pass
47            check_asai_all: '1',
48            check_asai_children: '1'
49          }}
50      end
51
52      # When an advertisement gets created it will redirect to it's show page.
53      assert_redirected_to advertisement_url(Advertisement.last)
54    end
```

These integration tests are also quite extensive in testing the access of each user, for each page and action. The ":authenticate_user!" method, is provided by Devise and can be found in the app\controllers\advertisements_controller.rb, near the top of the file.

```ruby
# This checks :authenticate_user!
# When the user is not signed in they should not be able to access the following
# with exception to index and get_image
test "should deny access to show advertisements" do
  sign_out @admin
  get advertisement_url(@advertisement)
  assert_redirected_to new_user_session_path
end

test "should deny access to new advertisements" do
  sign_out @admin
  get new_advertisement_url
  assert_redirected_to new_user_session_path
end

test "should deny access to edit advertisements" do
  sign_out @admin
  get edit_advertisement_url(@advertisement)
  assert_redirected_to new_user_session_path
end

test "should deny access to create advertisements" do
  sign_out @admin
  post advertisements_url, params: { advertisement: {
    description: @advertisement.description,
    title: @advertisement.title,
    url: @advertisement.url,
    file: @file, # This requires the fixture_file_upload method to pass.
    check_asai_all: '1',
    check_asai_children: '1'
  }}
  assert_redirected_to new_user_session_path
end
```

```ruby
# This checks :verify_permission
# When the user is a developer
# they should not be able to access the following.
test "should deny developer access to show advertisements" do
  sign_in @developer
  get advertisement_url(@advertisement)
  assert_redirected_to advertisements_path
  assert_equal "You are not authorized to perform this action.", flash[:alert]
end

test "should deny developer access to new advertisements" do
  sign_in @developer
  get new_advertisement_url
  assert_redirected_to advertisements_path
  assert_equal "You are not authorized to perform this action.", flash[:alert]
end

test "should deny developer access to edit advertisements" do
  sign_in @developer
  get edit_advertisement_url(@advertisement)
  assert_redirected_to advertisements_path
  assert_equal "You are not authorized to perform this action.", flash[:alert]
end

test "should deny developer access to create advertisements" do
  sign_in @developer
```

Lastly the same process is repeated in the test\controllers\vrassets_controller_test.rb file, and a very basic index test for the test\controllers\tutorial_controller_test.rb file.

test > controllers > vrassets_controller_test.rb > Ruby Solargraph > VrassetsControllerTest

```ruby
require "test_helper"

class VrassetsControllerTest < ActionDispatch::IntegrationTest
  setup do
    # Graps the users from the users fixtures
    @admin = users(:admin)
    @advertiser = users(:advertiser)
    @developer = users(:developer)

    # Signs in the admin to allow for general tests to pass. Where user needs specified sign_in method will be used.
    sign_in @admin #This makes use of the Devise helper to sign in the user.

    # Variable vor VR asset object that uses the values from the fixtures.
    @vrasset = vrassets(:one)
  end
```

```ruby
  test "user is signed in" do
    sign_in @admin
    get root_url
    assert_response :success
  end

  test "should get index" do
    get vrassets_url
    assert_response :success
  end

  test "should get new" do
    get new_vrasset_url
    assert_response :success
  end

  test "should create vrasset" do
    assert_difference("Vrasset.count") do
      post vrassets_url, params: { vrasset: {
        description: @vrasset.description,
        title: @vrasset.title,
        file: fixture_file_upload("BP_Poster_01.cuap", ".cuap"),
        image: fixture_file_upload("BP_Poster_01_Preview.png", "image/png")
      } }
    end

    assert_redirected_to vrasset_url(Vrasset.last)
  end

  test "should show vrasset" do
    get vrasset_url(@vrasset)
    assert_response :success
  end
```

```ruby
# This checks :authenticate_user!
# When the user is not signed in they should not be able to access the following
# with exception to index and get_image
test "should deny access to show VR assets" do
  sign_out @admin
  get vrasset_url(@vrasset)
  assert_redirected_to new_user_session_path
end

test "should deny access to new VR assets" do
  sign_out @admin
  get new_vrasset_url
  assert_redirected_to new_user_session_path
end

test "should deny access to edit VR assets" do
  sign_out @admin
  get edit_vrasset_url(@vrasset)
  assert_redirected_to new_user_session_path
end
```

```ruby
# This checks :verify_permission
# When the user is an advertiser and sometimes developer,
# they should not be able to access the following.
test "should deny advertiser & developer access to show VR assets" do
  sign_in @advertiser
  get vrasset_url(@vrasset)
  assert_response :found
  assert_equal "You are not authorized to perform this action.", flash[:alert]

  sign_in @developer
  get vrasset_url(@vrasset)
  assert_response :found
  assert_equal "You are not authorized to perform this action.", flash[:alert]
end

test "should deny advertiser & developer access to new VR assets" do
  sign_in @advertiser
  get new_vrasset_url
  assert_redirected_to vrassets_path
  assert_equal "You are not authorized to perform this action.", flash[:alert]

  sign_in @developer
  get new_vrasset_url
  assert_redirected_to vrassets_path
  assert_equal "You are not authorized to perform this action.", flash[:alert]
end
```

```
test > controllers > tutorial_controller_test.rb > ...
1   require "test_helper"
2
3   class TutorialControllerTest < ActionDispatch::IntegrationTest
4     # test "the truth" do
5     #   assert true
6     # end
7
8     test "should get index" do
9       get tutorial_url
10      assert_response :success
11    end
12  end
13
```

Finally, running these integration tests provides me with the confidence to that every user has access exactly to what they require. These tests have been extremely useful when adding or editing new features or even small edits. They provide the confidence that everything within the website platform should behave as expected.

To run these tests, simply using the command “rails test” would check for all test files including the unit, and integration tests, running all the tests within them. This provides an output of 44 separate executed tests, with 107 assertions, 0 failures, and 0 errors.

```
# Running:

............................................

Finished in 2.317232s, 18.9882 runs/s, 46.1758 assertions/s.
44 runs, 107 assertions, 0 failures, 0 errors, 0 skips
```

2.5.4 System testing

Unlike unit and integration testing, system testing consists of testing the whole website platform. When run, it opens a Chrome browser and starts clicking and acting with the GUI just like a normal user would.

System tests are being run from the following 3 files: test\system\advertisements_test.rb, test\system\tutorial_test.rb, and test\system\vrassets_test.rb. Whilst these tests are not as extensive as the integration tests, being conscious of the page count, the following screenshots are only snippets of the code. Their full implementation can be found on GitHub.

test > system > advertisements_test.rb > Ruby LSP > AdvertisementsTest

```ruby
require "application_system_test_case"
# Documentation: https://rubydoc.info/github/jnicklas/capybara/Capybara/Node/Actions
class AdvertisementsTest < ApplicationSystemTestCase
  setup do
    @user = users(:admin) #grabs the admin from the users fixtures
    sign_in @user #This makes use of the Devise helper to sign in the user.
    @advertisement = advertisements(:one)
  end

  test "visiting the index" do
    visit advertisements_url
    assert_selector "h1", text: "Advertisements"
  end

  test "should create advertisement" do
    visit advertisements_url
    click_on "New advertisement"

    fill_in "Description", with: @advertisement.description
    fill_in "Title", with: @advertisement.title
    fill_in "Url", with: @advertisement.url
    page.attach_file('advertisement_file', 'test\fixtures\files\Advertisement_test.png')
    check('advertisement_check_asai_all')
    check('advertisement_check_asai_children')
    click_on "Submit"

    assert_text "Advertisement was successfully created"
    click_on "Back"
  end
```

test > system > tutorial_test.rb

```ruby
require "application_system_test_case"
# Documentation: https://rubydoc.info/github/jnicklas/capybara/Capyba
class TutorialTest < ApplicationSystemTestCase

  test "visiting the index" do
    visit tutorial_url
    assert_selector "h1", text: "Tutorial"
  end

  test "introduction to VR-Adset is showing then collapse" do
    visit tutorial_url
    click_on "Introduction to VR-Adset"
    assert_selector "strong", text: 'VR-Adset', visible: true

    # Collapse
    click_on "Introduction to VR-Adset"
    assert_selector "strong", text: 'VR-Adset', visible: false
  end
```

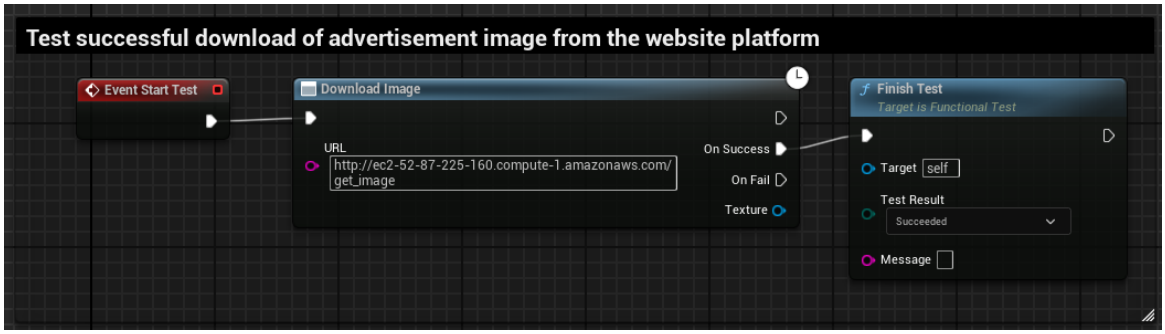
```
test > system > vrassets_test.rb
1   require "application_system_test_case"
2
3   class VrassetsTest < ApplicationSystemTestCase
4     setup do
5       @user = users(:admin) #grabs the admin from the users fixtures
6       sign_in @user #This makes use of the Devise helper to sign in the user.
7       @vrasset = vrassets(:one)
8     end
9
10    test "visiting the index" do
11      visit vrassets_url
12      assert_selector "h1", text: "VR Assets"
13    end
14
15    test "should create vrasset" do
16      visit vrassets_url
17      click_on "New Asset"
18
19      fill_in "Description", with: @vrasset.description
20      fill_in "Title", with: @vrasset.title
21      page.attach_file('vrasset_file', 'test\fixtures\files\BP_Poster_01.cuap')
22      page.attach_file('vrasset_image', 'test\fixtures\files\BP_Poster_01_Preview.png')
23      click_on "Submit"
24
25      assert_text "Vrasset was successfully created"
26      click_on "Back"
27    end
```

Running the system tests, using “rails test:system”, would provide the following results.

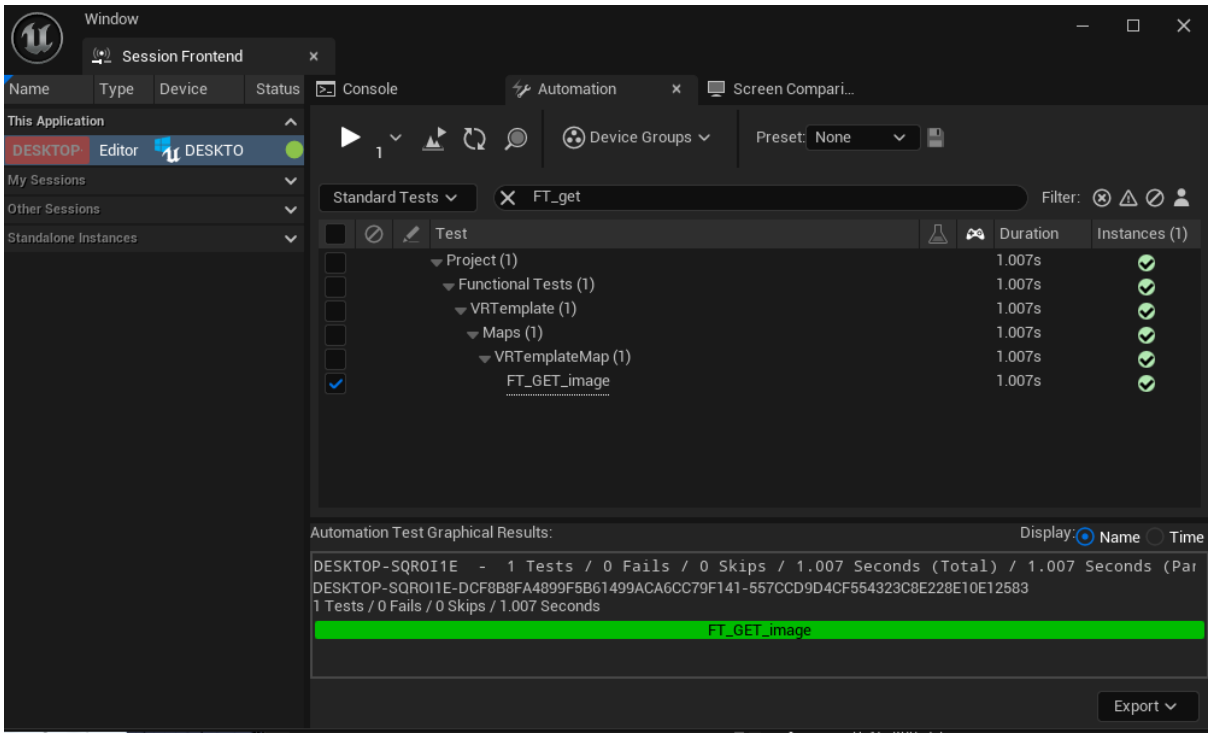
```
DevTools listening on ws://127.0.0.1:64110/devtools/browser/014abef0-e19f-4c04-b528-105f936b38db
Rack::Handler is deprecated and replaced by Rackup::Handler
Capybara starting Puma...
* Version 6.4.0 , codename: The Eagle of Durango
* Min threads: 0, max threads: 4
* Listening on http://127.0.0.1:64122
............

Finished in 15.662348s, 0.7662 runs/s, 0.9577 assertions/s.
12 runs, 15 assertions, 0 failures, 0 errors, 0 skips
```

2.5.5 Functional testing

This method of testing was used in Unreal Engine to test the functionality of the VR assets. In particular, the automated test will ensure that an image from an advertisement can be successfully retrieved from the website platform.



To run this automated test, select tools from the very top menu and then selecting test automation will bring up a page allowing you to search for the created test. From here running the tests, will provide the following results.



2.5.6 Usability testing

This test was carried out by reaching out to friends that were available to test the program. The testers were only informed that this application has the capability to upload advertisements within a virtual reality environment. They were told that upon successful sign up as an advertiser they should be able to upload an advertisement, without any other additional information guiding or directing them. They were provided with a screenshare of my VR environment to demonstrate the application is working and ads can be displayed.

One of the involved testers identified one critical problem that prevented them from being able to upload an advertisement. Whilst the blame was initially attributed to the browser, being that they used Opera, after further investigation, turned out to be the adblocker that they were using. Apparently, adblockers are filtering the input box for the title as a potential advertisement and automatically removes it.

Upon informing the user having an adblocker when attempting to create an advertisement is not a wise choice, the test resumed. Quickly thereafter, the tester uploaded an unsupported file format which was unable to load within Unreal Engine.

This testing has proven very successful in highlighting the two issues mentioned above. This allowed me to fix the errors and update the integration testing which resulted in other bugs being found, which have been promptly addressed.

2.5.7 Automated Security testing

For the automated security testing, I have made use of ZAP security scanner application used for finding vulnerabilities in web applications. Upon running numerous tests with the most successful one being for about 30 minutes and making over 5000 requests, the final reports found no High security vulnerabilities.

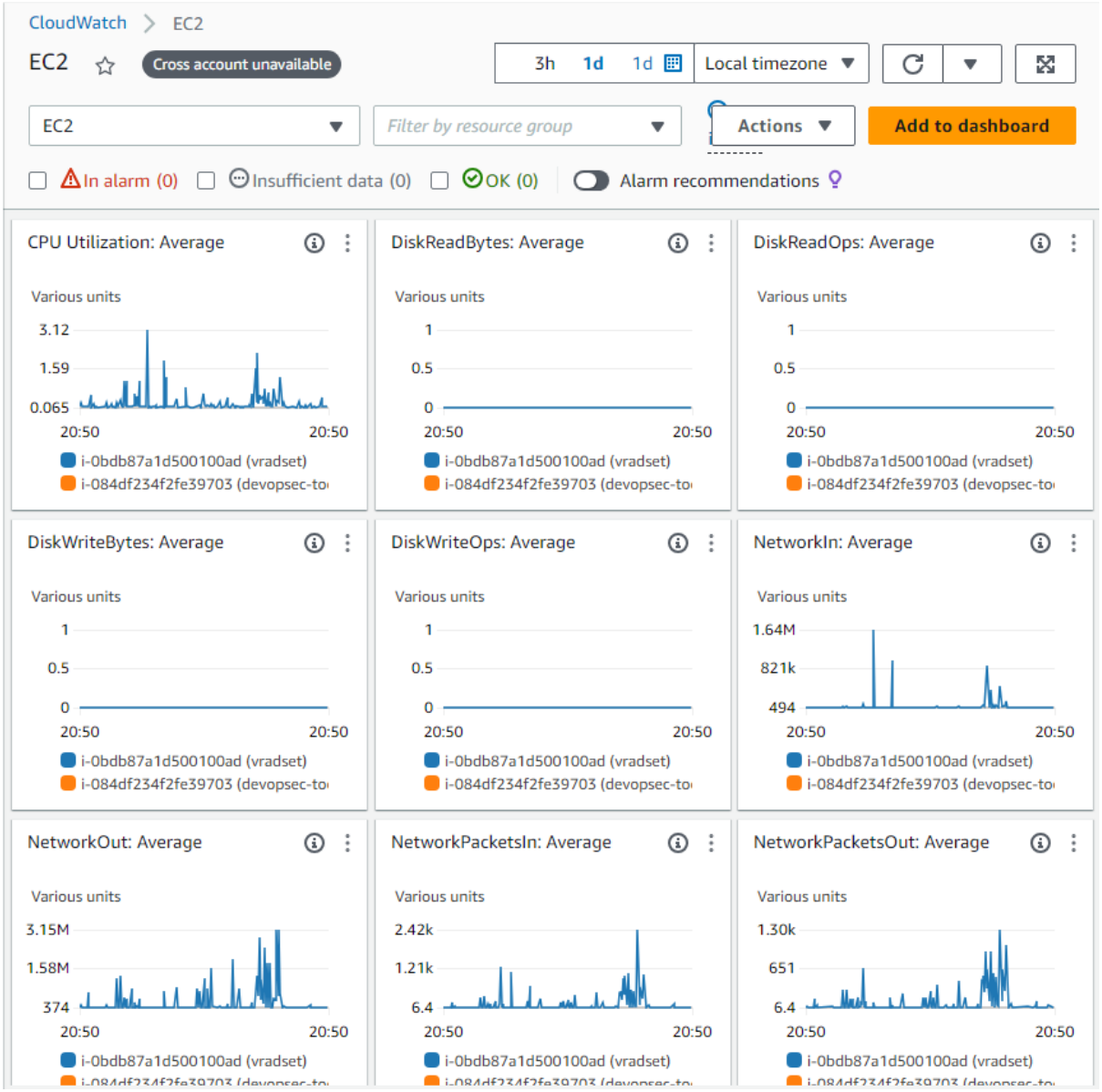
		Confidence				
		User Confirmed	High	Medium	Low	Total
Risk	High	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
	Medium	0 (0.0%)	1 (16.7%)	0 (0.0%)	0 (0.0%)	1 (16.7%)
	Low	0 (0.0%)	0 (0.0%)	1 (16.7%)	0 (0.0%)	1 (16.7%)
	Informational	0 (0.0%)	0 (0.0%)	2 (33.3%)	2 (33.3%)	4 (66.7%)
	Total	0 (0.0%)	1 (16.7%)	3 (50.0%)	2 (33.3%)	6 (100%)

The full report can be found in the appendix.

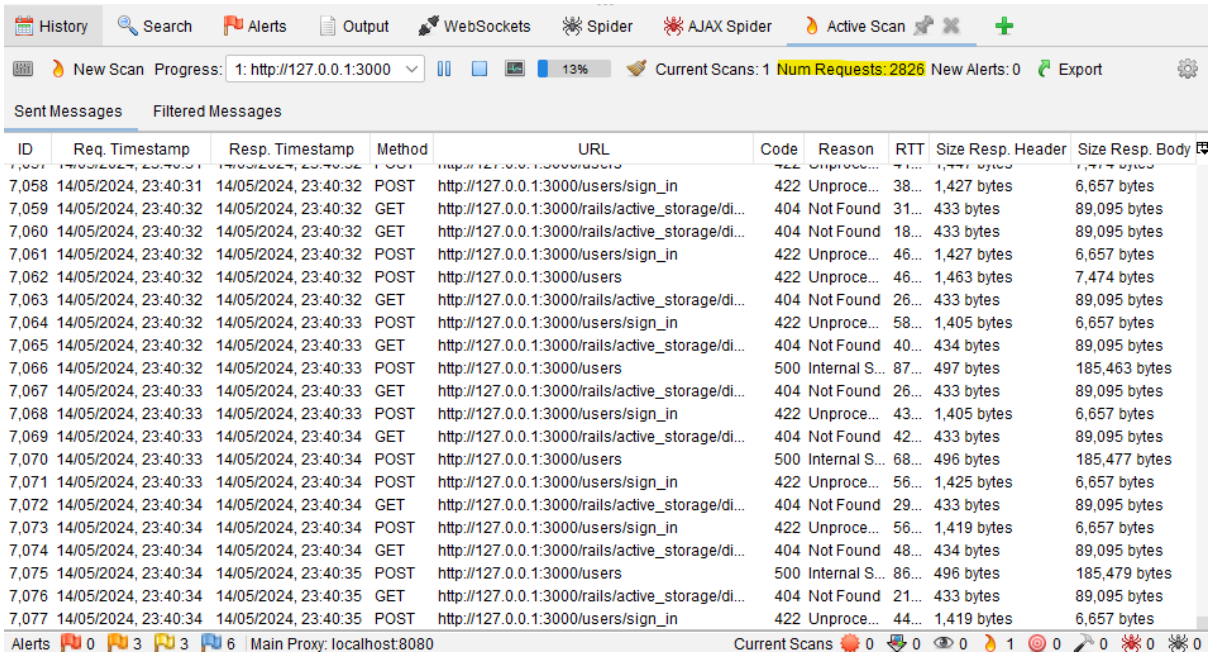
2.6 Evaluation

VR-Adset is a challenging application to evaluate due to its intended purpose is to be used by many developers in many applications at any time. That being said, using AWS as a service provides the comfort of knowing that if the application requires more resources, it can be easily accommodated. Updating or migrating the resource to any other service with larger storage, higher core performance and more RAM, makes it very scalable.

Currently the application is using very little resources nearing an average of 15% regarding the CPU usage. As it can also be observed in the screenshot attached below from the AWS CloudWatch the application is mostly idle when it comes to Network bytes.



Having a strong security foundation on the web application, was highly important to avoid unnecessary payments and ensure the application cannot be abused. Knowing that the application was withstanding about 2,826 requests in 2 minutes of the automated security testing, VR-Adset is a great success.



4. Conclusions

In conclusion, with the level of development that the application has to date, VR-Adset is ready to make a serious impression into the advertising market. It is one of the first applications to not be limited by one technology, such as Anzu (Anzu, n.d.) with Metaverse, but allow the developers the freedom of choosing what they build, using Unreal Engine. For that reason, VR-Adset is the first of its kind to be ready to take in advertisements. It provides advertisers with great value as they can start uploading their campaign at any time starting now and have it displayed in any VR environment.

Using VR-Adset as a developer provides them with the easiest experience of placing an advertisement. No setup, no hassle, just use a common asset library that you probably already own, two clicks, one drag-and-drop and it is done. You now have a reliable way of placing advertisements within VR.

The development of this web platform didn't come without challenges. many hours were spent in trial and error. This is especially true with how complex rails can get when simply noting the many versions available and how many solutions are being implemented in older versions. Ultimately a lot of techniques were learned when using the MVC controller with a web application, which is something I am proud of.

With further development and founding, this application has the potential to be the best advertising alternative available. It can be the solution to many advertisers looking to reach a new audience of technically inclined individuals who are likely to use such technology.

5. Further Development or Research

Looking forward towards the feature of this application, it becomes clear there is a lot of potential for improvement and scaling. I may argue it can even compete with Google's yet unreleased method of advertising within VR. (Upadhyay Aayush, 2017) With a larger library of assets and additional functionalities the potential for VR-Adset is truly exceptional.

One of the main features that is worth developing is having a compensation system for all the developers. This may also involve developing a whole bidding and making use of payment software which is capable of not only transactions but also storing their balance per individual advertisement.

These two features are something that are likely to be a tremendous amount of work. This is not only due to their complexity, but it is extremely important than when working with money, things are working well and not only so but they are thoroughly tested. This does mean that developing such features will require some sort of capital investment.

Two other features that work well together that could be implemented in feature development are: redirecting the user straight to a browser when interacting with an advertisement and displaying metric data to advertisers. Simply thinking about implementing the first feature introduces a lot of questions. How should the user interact? How does the user get informed that they can interact? What would the call to action look like based on the type of asset?

I also do recognize that for scenarios involving large number of advertisements are being uploaded and accessed globally, a database like Apache Cassandra might be more beneficial in the future. This preference is due to its clustering capabilities and NoSQL database, which is designed to handle large amounts of data across many servers. These features will prove essential when the application has developed to a world-wide advertisement platform.

Implementation of these features will elevate the application to well beyond market level but a strong competitor with the possibility of reaching one of the Fortune 500 companies. This is especially true as we know that a large amount of money is being invested in this technology by Meta alone and the technology can only get better from here.

6. References

Anzu, n.d.. *METAVERSE CREATORS.* [Online]
Available at: https://www.anzu.io/metaverse-creators
[Accessed 30 April 2024].

ASAI, 2024. *Children.* [Online]
Available at: https://adstandards.ie/code/children/
[Accessed 14 May 2024].

ASAI, 2024. *Code of Standards for Advertising.* [Online]
Available at: https://adstandards.ie/asa-code/
[Accessed 12 May 2024].

Connecter, 2023. *Free and visual creative assets management for 3D.* [Online]
Available at: https://www.designconnected.com/connecter
[Accessed 21 December 2023].

Meta for Developers, 2023. *Graph API Overview.* [Online]
Available at: https://developers.facebook.com/docs/graph-api/overview
[Accessed 21 December 2023].

Meta Quest, 2023. *Meta Quest Developer Center.* [Online]
Available at: https://developer.oculus.com/
[Accessed 21 December 2023].

Meta, 2023. *Meta for Developers.* [Online]
Available at: https://developers.facebook.com/
[Accessed 21 December 2023].

Pexels, 2024. *Legal Simplicity.* [Online]
Available at: https://www.pexels.com/license/
[Accessed 14 May 2024].

Totilo, S., 2023. *Unity apologizes, makes controversial new game development fees optional.* [Online]
Available at: https://www.axios.com/2023/09/22/unity-apologizes-runtime-fees
[Accessed 01 May 2024].

Unreal Engine, 2023. *Unreal Engine 5.* [Online]
Available at: https://www.unrealengine.com/en-US/unreal-engine-5
[Accessed 1 May 2024].

Unreal Engine, n.d.. *Supported XR Devices.* [Online]
Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/supported-xr-devices-in-unreal-engine?application_version=5.0
[Accessed 4 May 2024].

Upadhyay Aayush, N. R., 2017. *Experimenting with VR Ad formats at Area 120.* [Online]
Available at: https://developers.googleblog.com/2017/06/experimenting-with-vr-ad-formats-at.html
[Accessed 07 November 2023].

Vizualflow, 2023. *Unreal Engine Asset Management Tool.* [Online]
Available at: https://www.youtube.com/watch?v=FsYbiQ-xsUU&t=178s
[Accessed 4 May 2024].

7. Appendices

3.1 Project Proposal

National College of Ireland

Project Proposal

VR Advertising platform and assets

11/11/2023

Computing Project (BSHCSD4)

Software Development

2023/2024

Fabian Felix Gal

x20434312

x20434312@student.ncirl.ie

Contents

4. Objectives

My project will involve crating assets that VR Map builders can place in their worlds and get automatically filled in with ads on runtime. Similar to a virtual billboard that you can place in a world that will automatically display an advertisement based on the type of environment, game or even customer.

The VR advertising platform aims to provide VR map builders with easy-to-use assets that automatically fill with ads at runtime. This offers developers an alternative tool with a very easy setup.

The advertisements will be provided by companies on a web platform. The goals here is to provide a seamless way for advertisers to reach other demographics and mediums of advertising such as VR without the need of hiring their own developers. Upon completion, it will allow developers to easily advertise within their environment by uploading an asset from my gallery.

5. Background

This idea came to mind when thinking of emerging technologies such as VR and how Meta is investing in research and developers but also predicts it to be the future of technology. As I have recently purchased a Quest headset, I was able to see potential. I was also aware that the future is uncertain, and that many of the currently available applications are invested in gaming, storytelling and experiences. I have also noticed that most applications are only available by purchasing them. The idea come to mind when thinking that VR headsets will become more affordable with time and that people will likely be disappointed to see they have to pay for most of the apps they would like to experience, and thus a making use of advertising can be a perfect balance not only for the user, and developer but also companies.

The first step in fulfilling the objectives stated above is to create a web app where developers and companies can sign up. The developers will be able to quickly download assets and receive compensation, and the advertisers will have the ability to upload and label their ads for their desired target audience.

6. State of the Art

During my research, as of now companies that chose to advertise in VR would be employing their own developers to create a virtual store and products. This can be time consuming and an expensive option that only a few companies can afford.

Additionally, Google have been experiencing with advertising in VR which to this day it is still a concept and not integrated. (Upadhyay Aayush, 2017) The way they have approached this is by creating a cube which can be placed in an environment where a user can interact with. Upon interacting with the cube, by either tapping it or gazing it for a few seconds, a video will be played. When the video finishes, as per the demo presentation, it appears to be sending a notification to the user, presumably with the additional information of the ad.

My project will be using a different approach, where the ad can be displayed on a virtual poster, newspaper, billboard, and TV for the video ads, where the map builder can decide their location within their environment.

In my personal experience advertising in niche new markets provide the best return on investment. Additionally, this will also be targeting a more tech literate audience with a certain skill set that can be very useful to certain companies.

7. Technical Approach

When approaching this project, I aim to use an Agile development style. This will allow me to focus on the core features and get started with the most viable product. From there on I can add on the features that matter the most and iterate through them until the project is fully developed. This approach should allow me to identify any challenges. If at any point the project gets halted due to an unforeseen complexity on one task, I will have peace of mind that the core features are working and can iterate and adapt as required.

When identifying tasks, I will be first focusing on the needs of each member, creating user stories. Having the user stories outlining every need that a user, developer or company might need will allow me to break the project into tasks and priorities.

The needs and wants of each party involved in my project can be outlined in a web tool that I am familiar with and has helped me with previous projects, Trello. I will be using Trello to develop cards with each of the user's story and within it develop a checklist with every feature that the user needs. I will also aim to follow a Scrum like framework approach. The main columns I will be having are: "User stories", "Backlog", "To-Do", "Working on", and "Done".

Following this format will allow me to run Sprints that can be situated based on the importance under the "To-Do" column. The selection of tasks will be based on how critical the task is to the project. In this case, one critical aspect of the assignment is the database and web platform where both the developers and the company can sign up to upload their adds. Following that is having a test environment with at least one asset that the developers can make use of.

Following the mentioned approach will help me focus on the most important aspects of the project and come up with a prototype or MVP (Minimum Viable Product) and iterate as required until the project becomes as close as possible to being commercially viable.

8. Technical Details

As this project will require me to build assets for a VR environment, I will be making use of the Unreal Engine. This choice was made based on its recent popularity, it's growth of the engine, as well as the recent controversy created by their main competitor Unity. My main concern with Unity were the fact that while I might not be charged today for using and releasing the engine, I might be studying an engine that could potentially be declining in popularity with other companies resulting in me having to learn another engine in the future.

With Unreal Engine I will have to edit the materials or program via their GUI or using C++.

For building the web application, I will be making use of Ruby on Rails, where I would be migrating the database to Firebase if required for easy implementation. Additionally, I plan to make use Bootstrap and experiment with other approaches or programming languages during the development process.

When it comes to building the assets, I will be making use of Blender which is one of the most popular open-source 3D building software.

I will also investigate the use of advertisement servers and how they operate to either build my own iteration of one or incorporate them as part of the product.

9. Special Resources Required

When building the test environment where the product will be showcased, I will likely be opting for premade assets. This will help in building the room or environment where the users can experience and view the product working.

As all the data will be provided and collected from the users, developers and companies, I will not be requiring any other additional resources.

10. Project Plan

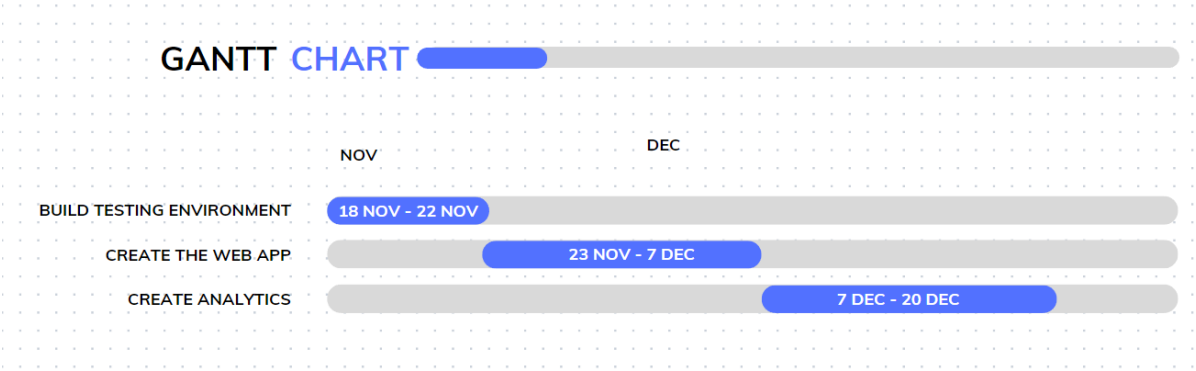
To start working on the project I will be first setting up a plan on Trello as mentioned earlier. The first and most important step in my development plan is to build a Minimum Viable Product. In order to make sure that I focus on the most important features, I will first create the user stories which will help me in creating new cards that can be placed in the backlog for the MVP.

For example, a developer will usually be creating his own environment where the assets can be placed in. This will require me to build my own environment for the assets to be tested. Following, are the assets themselves.

A company and developer will both require a portal or an app where they can download the assets or upload their adds to. This will create my next requirement which is to develop the web application.

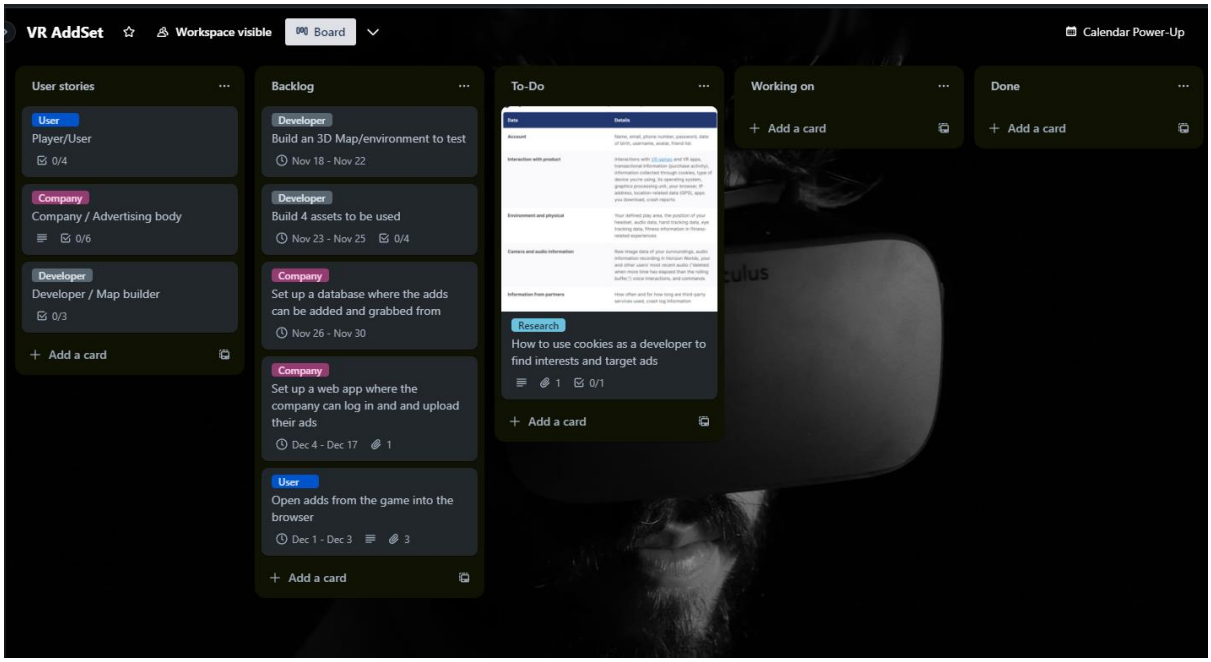
And a third requirement will be that the user is able to view the data, along with send a minimum level of analytics back to the server or application.

Focusing on these three requirements and keeping in mind that I will require a prototype by December 20th, I will be allocating a specific time for each of these main tasks.



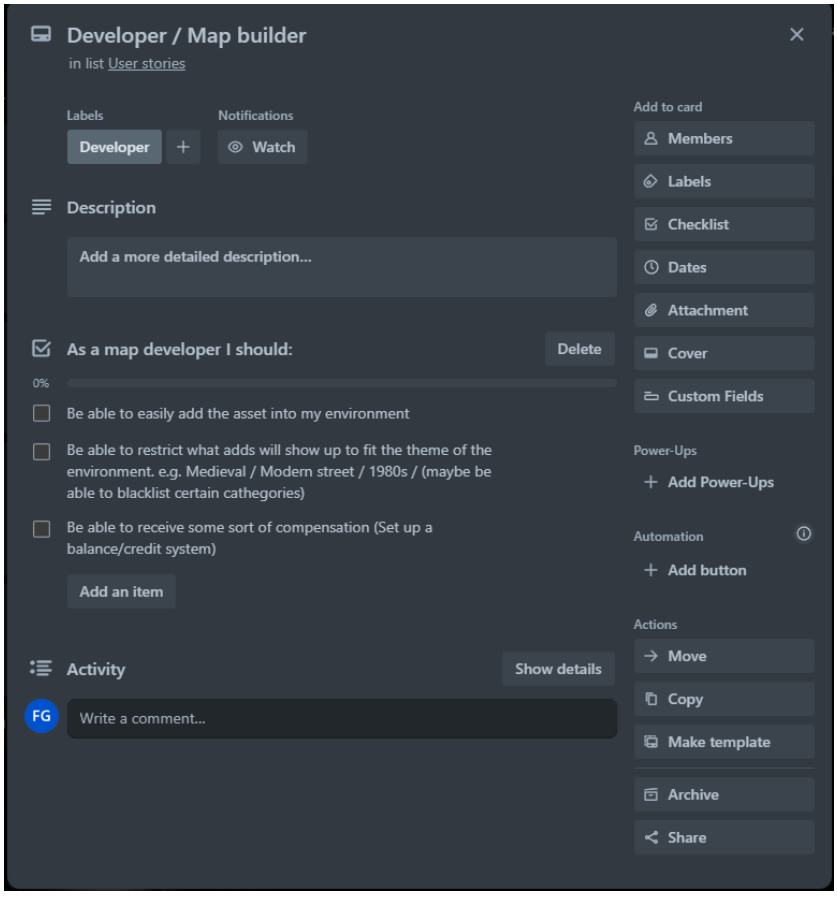
These tasks can then be broken down further into other cards or checklists. Some examples include “Build a room where the assets can be placed in”, “Create the 3 basic assets”, “build the web app”, “Set up the database” and so on.

Other cards can also be created for research and notes can be added to each card with every link found that helped with the research which will help keeping track of tasks if at any point they require to be built at the same time to test the compatibility.



I will also be making use of labels as seen in the above image to easily show which user story is it representing. Additionally, as the project gets more advanced, I can make use of priority labels such as “P1”, “P2”, “P3” to easily distinguish between the tasks as I become constrained by time.

Below is an example of a user story and what the interests of a Developer might be.



11. Testing

As the web application will be built on Ruby on Rails, most of the initial unit testing will be done using the built-in framework called Minitest. When testing the model, I could also make use of the RSpec which can be installed as a bundle.

Running this test will help me ensure that adding more features beyond the MVP will not break currently running features.

However, unit testing can be done mainly on the web application. Trying to implement testing using the Unreal Engine, whilst possible, it will only allow simulating the user's movements. This can allow me to move the player into the direction of each advertisement asset and generate analytics related to the amount of time a user has had the advertisement in their direct line of sight. However, in order to determine the test successful, I will also need to have another test within the web application to confirm the user has seen an advertisement and for how long. This is way more complex and as of now I am not aware of any framework or application that can do this automatically. All this complexity will only test the user, and not the developer experience and how easily they can place the assets. My best alternative is to run the tests manually for the VR environment.

Additionally, finding candidates that know how to build 3D environments for VR will be challenging. An alternative option will be to create a testing environment within the Unreal Engine where I can guide developers through the process of placing an asset and launching the environment where I can ask for feedback. I also have a prebuilt application or game that has been approved and uploaded to Oculus Quest store as a proof of concept. Users or other parties should be able to walk around and view ads every time the game or map is run.

References

Anzu, n.d.. *METAVERSE CREATORS.* [Online]
Available at: https://www.anzu.io/metaverse-creators
[Accessed 30 April 2024].

ASAI, 2024. *Children.* [Online]
Available at: https://adstandards.ie/code/children/
[Accessed 14 May 2024].

ASAI, 2024. *Code of Standards for Advertising.* [Online]
Available at: https://adstandards.ie/asa-code/
[Accessed 12 May 2024].

Connecter, 2023. *Free and visual creative assets management for 3D.* [Online]
Available at: https://www.designconnected.com/connecter
[Accessed 21 December 2023].

Meta for Developers, 2023. *Graph API Overview.* [Online]
Available at: https://developers.facebook.com/docs/graph-api/overview
[Accessed 21 December 2023].

Meta Quest, 2023. *Meta Quest Developer Center.* [Online]
Available at: https://developer.oculus.com/
[Accessed 21 December 2023].

Meta, 2023. *Meta for Developers.* [Online]
Available at: https://developers.facebook.com/
[Accessed 21 December 2023].

Pexels, 2024. *Legal Simplicity.* [Online]
Available at: https://www.pexels.com/license/
[Accessed 14 May 2024].

Totilo, S., 2023. *Unity apologizes, makes controversial new game development fees optional.* [Online]
Available at: https://www.axios.com/2023/09/22/unity-apologizes-runtime-fees
[Accessed 01 May 2024].

Unreal Engine, 2023. *Unreal Engine 5.* [Online]
Available at: https://www.unrealengine.com/en-US/unreal-engine-5
[Accessed 1 May 2024].

Unreal Engine, n.d.. *Supported XR Devices.* [Online]
Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/supported-xr-devices-in-unreal-engine?application_version=5.0
[Accessed 4 May 2024].

Upadhyay Aayush, N. R., 2017. *Experimenting with VR Ad formats at Area 120.* [Online]
Available at: https://developers.googleblog.com/2017/06/experimenting-with-vr-ad-formats-at.html
[Accessed 07 November 2023].

Vizualflow, 2023. *Unreal Engine Asset Management Tool.* [Online]
Available at: https://www.youtube.com/watch?v=FsYbiQ-xsUU&t=178s
[Accessed 4 May 2024].

6.2 Poster

SOFTWARE DEVELOPMENT

National College of Ireland

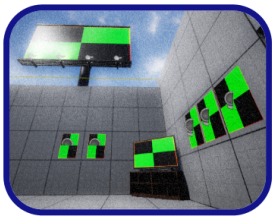
VR-Adset

Fabian Felix Gal, BSHCSD4

Overview

Virtual Reality for Advertisers & Developers.

Introducing advertisements into VR by providing a platform for advertisers and easy-to-use assets for developers.



Motivation

Many VR projects are being left behind due to lack of funding. This is especially true for new developers. As a VR user searching through the store, high prices, abandoned projects, and apps with great potential are a normal experience. VR-Adset aims to support the VR industry by providing developers with an alternative tool for advertising, and advertisers with the opportunity to reach a new audience.

Scan me!

https://vr-adset.com/

How it Works

This project uses an AWS EC2 instance to host the Ruby on Rails application with its MVC architecture. In combination with Unreal Engine blueprint assets, and Connecter asset management tool, placing and retrieving an ad from the website becomes a drag-and-drop process.

Technologies

aws GitHub RAILS UNREAL ENGINE oculus CLOUDFLARE Connecter

6.3 Reflective Journals

Supervision & Reflection

Student Name	Fabian Felix Gal
Student Number	x20434312
Course	Computing Project (BSHCSD4)
Supervisor	Emer Thornbury

Month: October

What?

During this month I have been looking at potential ideas for a project as well as settling on one.

At first, I have tried to look into ideas into my initial environment where I can see what not only, I struggle with that can be automated or aided by software, but something that can scale to other people, families, or communities. Thus, my initial idea was linked to a family planning app that will not be saving private information and that will aid in spreading of chores by allowing a family member to start a meeting on a TV. The idea was to develop a cloud and android app that can sync and be used by the children that wouldn't specifically have access to any other device but their TV. Another core function of the app was to gamify the experience with goals and rewards based on the number of tasks completed.

The other Idea I had was influenced by the increasing popularity of Virtual Reality. I have noticed how since Facebook has changed their name and tried to push this new technology into commercial practices and businesses that there is a lot of value in familiarizing with the technology. As such I have figured that the more mainstream the use of VR becomes, the more likely is that companies would like to advertise in these new mediums. I have taken inspiration form AdSense and how they are now advertising not only on YouTube but also on other sites with a simple API that can be added into a web page. As such I decided to try and create a similar asset where game developers can just place it in link it with my API and do not worry about what add will be displayed unless they decide on a specific theme. I was initially looking to make something like a billboard asset, a newspaper asset, a poster asset that can automatically get filled in with a list of adds that companies will upload on my website based on the user and environment.

So What?

Realizing that making a decision might not be easy and that we will need a supervisor I have reached out Emer via email ahead of time and she agreed. At the time of writing the email I was aware that we will soon have a class where we will have the opportunity to present our ideas and get some feedback from the program coordinator.

Upon presenting my ideas to the program coordinator, I have been informed that the idea with the most potential would be the VR advertising asset API.

I have then continued to record a video for my project pitch and uploaded it. Confident in the potential I have then waited for a verdict which came close to Friday the 27th close to the project proposal's deadline.

I have also started researching and experimenting with a personal VR headset, Oculus quest 2. I have investigated the software and applications that I can use to try and create a virtual environment myself. I have installed the Oculus software required for Air link and other connectivity with my personal computer. I have also investigated some game engines that map creators can use to create and environment and was even successful in running a personal demo VR environment from Unreal Engine 5.3.

Unfortunately, as my idea has been rejected and was provided with feedback informing me that there is potential, however I will need to schedule a time with my supervisor to rethink the process. I initially was anxious to find out of my idea being rejected but luckily that meant that I would have gotten an extension.

Now What?

As the supervisor list has also been released that day, I was happy to confirm that Emer was indeed my supervisor and immediately emailed her looking to schedule a 1 to 1 meeting to address the rejection of the idea. This will also allow me to refine and better plan the project.

This is the stage I am currently in. Due to bank holiday and the week off currently all I can do is wait for a reply on my email and attend, assess and replan my idea before the 12th of November.

Student Signature	

Supervision & Reflection

Student Name	Fabian Felix Gal
Student Number	x20434312
Course	Computing Project (BSHCSD4)
Supervisor	Emer Thornbury

Month: November

What?

In this month I have had a deeper look at the ethics implications of my project.

Starting this month, I was able to have 2 meetings with my supervisor where I did my best to cover the ethical implications of the data that I plan to be storing and how to avoid any serious implications to any of the involved members of the application.

I have also been encouraged to first come up with a use case diagram to make it easier to explain the contents and focus on the deliverables.

Ultimately, this month has been passing quickly due to other assignments needing to be covered thus leaving me with little time for this project.

So What?

As a result of the meetings I have focused on the it governance and ethics assignment thinking it will allow me to learn and address some of the ethical worries I have had. At first I was worried that the data being collected might need to be problematic however as I focused on the assignment more and had researched the issue a little more, I have realized that the data collected under the forms of cookies or other user information will not be provided and there will be no identifiable information as the user will be assigned an ID. Due to my project focusing on Meta's infrastructure and signing up all the data that will be available to me will be data that the user has agreed to within the terms and conditions and is data I will be able to collect. Even if that is not the case where identifiable information will be available, I do not intend nor plan on using any information that could identify a user of the application.

I have also had the opportunity to look into how the AdTech market works and the architecture behind displaying an ad while researching for my assignment. This has allowed me to better understand that I should consider having a server that can handle the advertisements and act as an exchange and bidding system.

While working on this assignment I have also completed the use case diagram which was followed by the second meeting with the supervisor. During this meeting, we went over any loose ends that there might be in the ethics implications and were able to come with strategies that will avoid any ethical issues.

Now What?

Upon finishing the ethics assignment, I have realized that there might be a need for a filtering system later in the development cycle as the application is to be released. This is to filter through any new advertisement before allowing them to be placed in the list of viewable ads.

Now that I am getting closer with finishing with other assignments I will be dedicating days to have a prototype done in time.

Student Signature	

Supervision & Reflection

Student Name	Fabian Felix Gal
Student Number	x20434312
Course	Computing Project (BSHCSD4)
Supervisor	Emer Thornbury

Month: December

What?

In this month I have focused on the development of the prototype of the application, technical report, and video presentation along with the slides.

This month has been a very busy month considering the deadlines of other subjects however I managed to end up with a prototype that I am satisfied with where the application has a base web app for the developers, and advertisers, and the blueprint asset is able to display an ad from the website.

So What?

Being limited on the time I have had available for the project; I have had to focus on quickly finishing any other subject so that I can afford most of the time and attention to this module. Upon doing so I have immediately focused on creating the bare bones asset and the blueprint of the application. This was a rather easy task to complete due to keeping it simple and focusing on the functionality first rather than the design. As such I was able to create a simple plane within the Unreal Engine and attach a blueprint functionality where it will be able to display an image from any web address that the blueprint is set to. This process took about a day to research, test multiple variations, and about 2 hours to implement.

Upon having a functioning blueprint and asset I have had to focus on how to export said asset as a one downloadable file that can be easily imported into the project. This turned out to be a bit more complex within Unreal Engine as they only allow the ability to merge assets from one project into another. This led me to research popular asset management tools that allow such functionality of easily implementing one downloadable file, such as Connecter. This application is very well known within the 3D building community where it is used for a very wide array of applications spanning from Blender to Revit, Maya and Unreal Engine.

Next, I have focused on the website part of my application. I have made use of Ruby on Rails to quickly put together a website with two pages one for advertisements and one for assets. Then I have followed up with making sure that the CRUD application also allows the storage of files and is working appropriately. Upon testing this functionality, I have pushed the application on one AWS EC2 instance. This in turn allowed me to use the URL of the uploaded advertisement and display it within a VR environment.

Lastly, I have recorded a short presentation of my application which was very hard to keep under 10 minutes and took me multiple attempts to get it right or in a state I was somewhat satisfied with.

Now What?

Looking forward I have realized more tasks that need to be completed and currently the most important one is to make sure that users are being able to authenticate. This will make it so that I can restrict access to what a user can do and see when logged in. I aim to have this functionality done by the end of January.

Following the with February I aim to fucus on the asset design and functionality where the asset will indeed be loaded based on an advertisement and developer. After having these two functionalities covered, I will be focusing on the tracking and any other aspect of the application that needs be refined in March if everything goes to plan. However, I do expect some of the functionalities to take longer than anticipated as such for now I only aim to get the login, restrictions, and asset management and functionality done as soon as possible.

Student Signature	

Supervision & Reflection

Student Name	Fabian Felix Gal
Student Number	x20434312
Course	Computing Project (BSHCSD4)
Supervisor	Emer Thornbury

Month: January

What? This month has had the smallest progress of all the months. This was due to holiday plans with family and friends followed by studying and preparation for our first in person exam, and a chain of health issues forcing me to an involuntary resting period and feeling unmotivated.
So What? As of this month there has been no progress in the project past the prototype delivered in December.
Now What? I hope to follow up with my supervisor and catch up on the project so I have a better understanding of what has gone right in my mid-point submission and what I need to work on.

Student Signature	

Supervision & Reflection

Student Name	Fabian Felix Gal
Student Number	x20434312
Course	Computing Project (BSHCSD4)
Supervisor	Emer Thornbury

Month: February

What?

This month had little progress compared to January. During this month I was able to receive feedback on my mid-point submission and get back into the programming. I have been held back by one of the error messages related to the application.js file and had an attempt at implementing the login into the website while having meetings with my supervisor receiving helpful advice.

So What?

I have had multiple fast attempts at solving the problem on my local repository where I haven't had much luck. Thus, I decided to focus back on to the AWS EC2 instance and try and fix the issue there. With some luck the application worked after transferring a copy of the CSS file that was missing form the EC2 instance.

I have at a later stage attempted to fix the error that I was having locally "Sprockets::DoubleLinkError in Vrassets#index". Following this I have also researched the Devise gem necessary for the addition of the login functionality.

Now What?

Currently I am still focusing on the login I feel there is a lot more that needs to be addressed in order to get it fully working.

Student Signature	

Supervision & Reflection

Student Name	Fabian Felix Gal
Student Number	x20434312
Course	Computing Project (BSHCSD4)
Supervisor	Emer Thornbury

Month: March

What? This month has been filled with many errors whilst trying to set up personalised access based on the role a user has on the website. I have had a few struggles setting up the roles and displaying them on to the main website. However, by the end I was able to successfully have the roles be selected upon user registration.
So What? Now that I have set up the roles, I can quickly integrate the per role permissions and only limit the access of the user based on their roles and the things they need based on the use case diagram.
Now What? Following this change I can focus on the finer details such as the assets and how the data will be displayed on the dashboard when viewed. I am still confident this can be done within time I was just side tracked with the security assignments that we have going which also thought me more about how hard it can be to build a rails app insecurely.

Student Signature	

Supervision & Reflection

Student Name	Fabian Felix Gal
Student Number	x20434312
Course	Computing Project (BSHCSD4)
Supervisor	Emer Thornbury

Month: April

What?

Within the month of April, a lot of progress has been done to the web application and the assets within Unreal engine. I have been able to limit the content that is displayed based on the roles of the user. For example, a Developer should not be able to see or edit the advertisements of any advertiser.

In addition, I have added and fixed some of the ruby tests that broke due to adding the login functionality and limitations. Following this, I have merged the main branch with the Sign-Up branch within GitHub.

And lastly, I have managed to limit the ads to only be displayed based on the user that set them up and only see their created ads. This then allowed me to create an API call or a route that the VR assets can be requesting which will result back with any random image added to the advertisements. This also required me to update the assets that I have and create one poster, one billboard and a TV which can be loaded with any one ad upon runtime which was successful.

While the work done in the previous months has helped me in adding these features a lot easier and laid a good foundation to work from, a lot of progress has also been lost due to running into another health issue that has had me in the hospital for a week and will require me to attend again within the next 2 to 3 weeks.

So What?

Despite me feeling confident I have a Minimum Viable Product ready; I still find there are some features that would be ideal to have such as Displaying the advertising metrics such as if a user has clicked on any one ad or the number of times it has been displayed. Other additions such as a checkbox for when uploading advertisements to make sure they comply with the terms and conditions, or making sure that the same ad is not being displayed twice in the same room. Currently when having multiple posters in one room there is a chance that the same add can be displayed one after the other which ruins the experience.

As for the health condition I have applied for an extension where I am yet to hear back from the admin staff.	
Now What? Going forward I will be focusing on the features mentioned earlier however after updating the Technical report some more. This was brought up to me by my supervisor advising that I start updating the report as soon as possible before working on the features or whilst working on them. Currently I am working on the report and would hope to have all features done by the 09/05/2024 so that I can allow the rest of the time finalizing the project and any other small features.	
Student Signature	

6.4 Automated Security Report

ZAP Scanning Report

Generated with ZAP on Tue 14 May 2024, at 23:48:57

ZAP Version: 2.14.0

ZAP is supported by the Crash Override Open Source Fellowship

Contents

About this report

Report description

Report on VR-Adset

Report parameters

Contexts

The following contexts were selected to be included:

- Default Context

Sites

The following sites were included:

- https://optimizationguide-
- pa.googleapis.com https://content-
- autofill.googleapis.com
- https://cdn.jsdelivr.net
- http://127.0.0.1:3000

https://accounts.google.com

(If no sites were selected, all sites were included by default.)

An included site must also be within one of the included contexts for its data to be included in the report.

Risk levels

Included: High, Medium, Low, Informational

Excluded: None

Confidence levels

Included: User Confirmed, High, Medium, Low

Excluded: User Confirmed, High, Medium, Low, False Positive

Summaries

Alert counts by risk and confidence

This table shows the number of alerts for each level of risk and confidence included in the report.

(The percentages in brackets represent the count as a percentage of the total number of alerts included in the report, rounded to one decimal place.)

Confidence

	User Confirmed	High	Medium	Low	Total
High	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Medium	0 (0.0%)	1 (16.7%)	0 (0.0%)	0 (0.0%)	1 (16.7%)
Low	0 (0.0%)	0 (0.0%)	1 (16.7%)	0 (0.0%)	1 (16.7%)
Informational	0 (0.0%)	0 (0.0%)	2 (33.3%)	2 (33.3%)	4 (66.7%)
Total	0 (0.0%)	1 (16.7%)	3 (50.0%)	2 (33.3%)	6 (100%)

Risk

Alert counts by site and risk

This table shows, for each site for which one or more alerts were raised, the number of alerts raised at each risk level.

Alerts with a confidence level of "False Positive" have been excluded from these counts.

(The numbers in brackets are the number of alerts raised for the site at or above that risk level.)

Risk

				Informational
	High (= High)	Medium (>= Medium)	Low (>= Low)	Informational (>= Informational)

Alert counts by alert type

This table shows the number of alerts of each alert type, together with the alert type's risk level.

(The percentages in brackets represent each count as a percentage, rounded to one decimal place, of the total number of alerts included in this report.)

Alert type	Risk	Count
Content Security Policy (CSP) Header Not Set	Medium	15 (250.0%)
X-Content-Type-Options Header Missing	Low	11 (183.3%)
Authentication Request Identified	Informational	5 (83.3%)
Information Disclosure - Suspicious Comments	Informational	15 (250.0%)
Modern Web Application	Informational	4 (66.7%)
Session Management Response	Informational	557

Identified	(9,283.3%)
Total	6

Alerts

Risk=Medium, Confidence=High (1)

Risk=Low, Confidence=Medium (1)

Risk=Informational, Confidence=Medium (2)

Risk=Informational, Confidence=Low (2)

Appendix

Alert types

This section contains additional information on the types of alerts in the report.

Content Security Policy (CSP) Header Not Set

Source	raised by a passive scanner (Content Security Policy (CSP) Header Not Set)
CWE ID	693
WASC ID	15
Reference	▪ https://developer.mozilla.org/en-

US/docs/Web/Security/CSP/Introducing_Content_Security_Policy

- https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
- https://www.w3.org/TR/CSP/
- https://w3c.github.io/webappsec-csp/
- https://web.dev/articles/csp
- https://caniuse.com/#feat=contentsecuritypoli cy
- https://content-security-policy.com/

X-Content-Type-Options Header Missing

Source	raised by a passive scanner (X-Content-Type-Options Header Missing)
CWE ID	693
WASC ID	15
Reference	- https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/iedeveloper/compatibility/gg622941(v=vs.85) - https://owasp.org/wwwcommunity/Security_Headers

Authentication Request Identified

Source raised by a passive scanner (Authentication Request Identified)

Reference
- https://www.zaproxy.org/docs/desktop/addons/authentication-helper/auth-req-id/

Information Disclosure - Suspicious Comments

Source raised by a passive scanner (Information Disclosure - Suspicious Comments)

CWE ID 200

WASC ID 13

Modern Web Application

Source raised by a passive scanner (Modern Web Application)

Session Management Response Identified

Source raised by a passive scanner (Session Management Response Identified)

Reference
- https://www.zaproxy.org/docs/desktop/addons/authentication-helper/session-mgmt-id