Bilkent University

Department of Computer Engineering

Senior Design Project

craftual: a 3D model viewer and an asset management, presentation-based cloud platform

Low Level Design Report

Authors: Endri Suknaj, Çağrı Orhan, Deniz Doğanay, Doruk Altan, Sencer Umut Balkan

Supervisor:

- Prof. Ibrahim Korpeoglu

Innovation Expert:

- Yeliz Yigit

08 January 2020

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Content

1. Introduction **3**

1.1 Object design trade-offs **3**

1.1.1 Response Time vs. Space **3**

1.1.2 Usability vs. Functionality **4**

1.1.3 Security vs. Cost **4**

1.2 Interface documentation guidelines **4**

1.3 Engineering standards **4**

1.4 Definitions, acronyms, and abbreviations **4**

2. Packages **6**

2.1 Client **6**

2.1.1 View **6**

2.1.2 Redux **7**

2.2 Server **8**

2.3 Data **8**

3. Class Interfaces **9**

3.1 Client **9**

3.1.1 View **9**

3.1.2 Redux **15**

3.2 Server **19**

4. Glossary **21**

5. References **21**

1. Introduction

Presentations are an important part of work life that help people explain ideas, concepts and topics either for educational purposes or for other work related purposes. A survey conveyed with people working on different areas showed that 92% of the employees believed that presentation skills are critical for success at work. Yet 75% of adults are estimated to be affected by public speaking hindering their ability to be successful.[1] Aiming to help people affected by this condition we thought of ways to reduce their fear by boosting their confidence. The best tool to do this is to help people create a more unique and well-designed presentation. Research showed that 91% of people felt more confident during presentations when they were presenting with a well-designed slide deck.[2] The traditional presentation is made using Microsoft Powerpoint and consists of text and usually stock images. However, an estimated 79% of employees find these presentations boring and they claim that they often lose their concentration after the first 10 min. A novel approach that would increase the interest of people on these presentations is using 3D presentations in combination with AR technology. This technology could be used on the creative stage as well as the presentation of the final product. The interactive interface would allow the user to observe how small changes would affect the overall design of the product since all the components are in the appropriate ratios. Being able to easily see how minor or major changes influence the final product is a huge benefit regarding efficiency as it will prevent costly adjustments that were not foreseen. Easy remodelling and editing makes more accurate designs which means less cost and time invested. The 3D product presentation would offer a much more vivid depiction of the product that a 2D presentation could ever depict.

1.1 Object design trade-offs

1.1.1 Response Time vs. Space

The trade off between time and space is very common in systems that deal with high amounts of data. Craftual will store a lot of data including models, presentations, comments, personal information and more. The bulk of the data is stored on server time to provide efficient access even when there is high amounts of data stored. Furthermore, caching and database optimization techniques are applied to produce fast data inquiries as the most time consuming functionalities lie in the database operations.

1.1.2 Usability vs. Functionality

This trade off is approached in different ways regarding the two main functions of craftual which are model viewing and presentation creation. Aside from a couple of requests to the database, viewing a model requires very little functionality but an easy to understand interface, therefore the emphasis will be on usability.

On the other hand, when we consider making a presentation, the processes require more modification on data and more requests from the back-end. For this reason, functionality is prioritized over usability.

1.1.3 Security vs. Cost

Craftual stores sensitive data on every user such as personal information or intellectual property. One important concern of our project is to keep such information as safe as possible. Main precaution against any leak or theft of such data is using encrypted databases. As database gets bigger, the cost increases with the added labour and time requirements to deal with both the privacy and space management.

1.2 Interface documentation guidelines

All interface documentation is done according to UML design principles and the naming of every class, method and variable will be under camel case conventions. The description of classes will start with the class name, followed by the related attributes and methods. A sample outline is given below.

Class Name	The name of the class
Class Description	Concise description of class
Attributes	The set of attributes in the class
Methods	The set of methods that can be performed

1.3 Engineering standards

UML design principles are used for the visualization of use cases, interfaces, scenarios, class diagrams and subsystem decompositions [3]. This way, we were able to represent the system structure in an object oriented manner.

1.4 Definitions, acronyms, and abbreviations

AR Augmented Reality

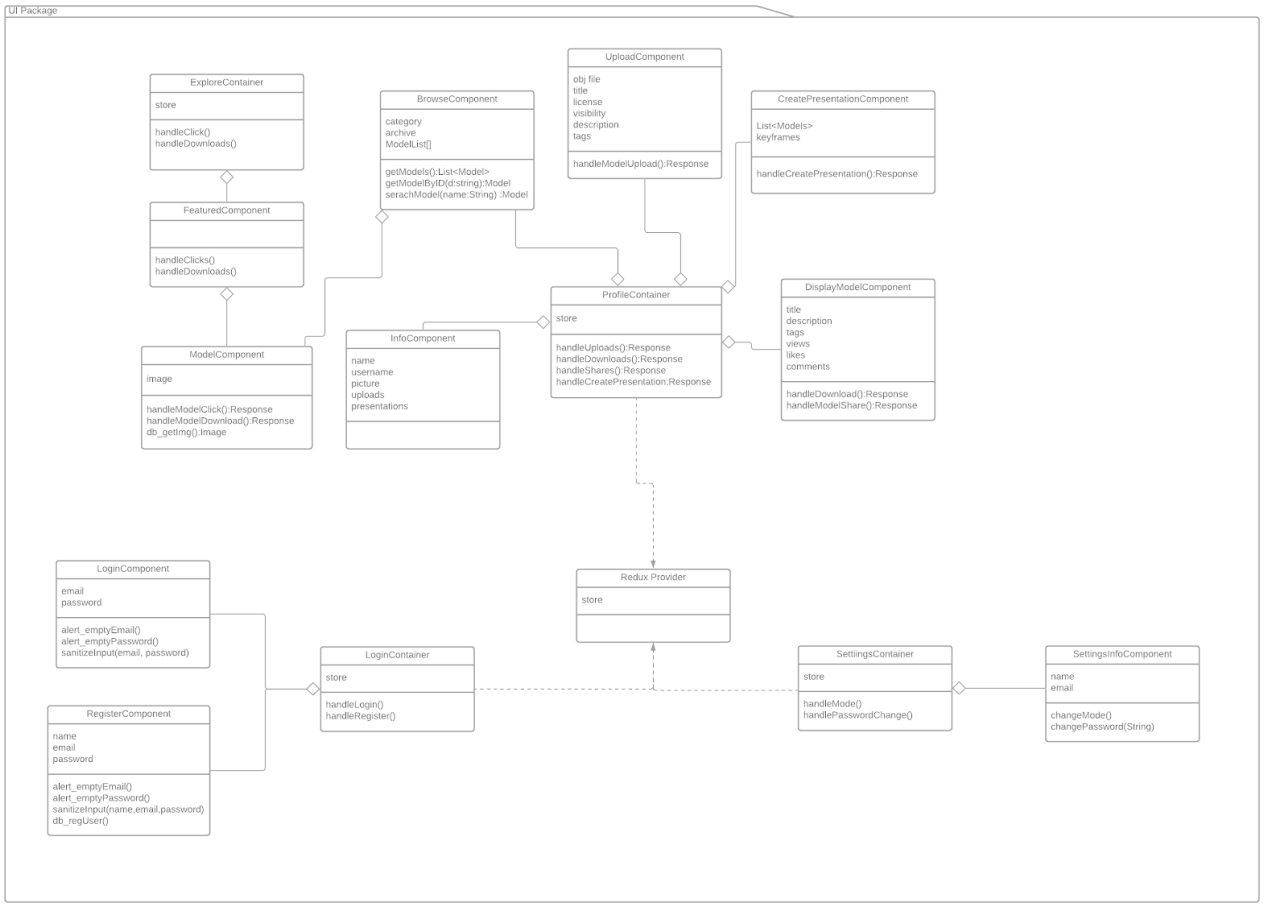
VR Virtual Reality

GUI	Graphical User Interface
UML	Unified Modelling Language
XSS	Cross Site Scripting
DBM	Database Management
API	Application Programming Interface
I/O	Input/Output
HTTP	Hypertext Transfer Protocol
DAC	Discretionary Access Control
MAC	Mandatory Access Control

2. Packages

2.1 Client

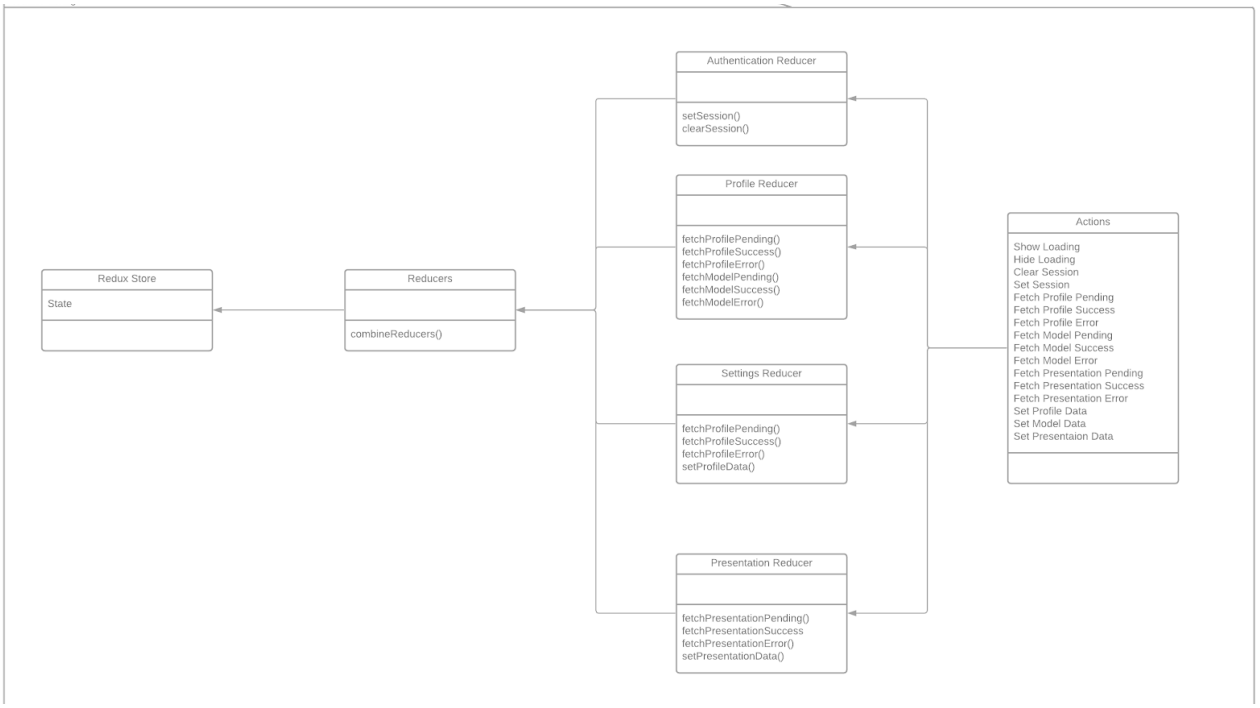
2.1.1 View



- **Redux Provider:** This component will encapsulate the all the presentation layer components and containers and will be used to make the redux store available the entire app.
- **Login Container:** This is the component that will control the view of the Login Page. This component will have two child components: the login component and the register component.
- **Login Component:** This component will control the view of the login component. After the user enter the credentials, they will be dispatched to the redux reducer to manage the login operation.
- **Register Component:** This component will control the view of the register component. A new user will be prompted to fill a form and that data will be dispatched to the redux reducer to proceed with the registration of the user.
- **Settings Container:** This component will control the view of the settings page. This component will have a child component named Settings Component.
- **Settings Component:** This component will display and allow the user to change some of his profile data such as his username, email, password, image etc.

- **Profile Container:** This component will control the view of the profile page. This component will have various child components such as Info Component, Display Model Component, Browse Component, Upload Component, Create Presentation Component.
- **Info Component:** This component will be able to display some general information about the user on the Profile Page.
- **Display Model Component:** This component will be able to display a model and its information when a user clicks on a model.
- **Browse Component:** This component will display publicly available models and presentations.
- **Upload Component:** This component will display the form that is filled when a user is uploading a model to their account.
- **Create Presentation Component:** This component will display the screen and necessary tool required to make a presentation from a basic model.
- **Explore Container:** This component will display the screen that shows all the publicly available models and presentations. This component will have a child component named Featured Component.
- **Featured Component: T**his component will display the featured or the most viewed and liked models and presentations.

2.1.2 Redux



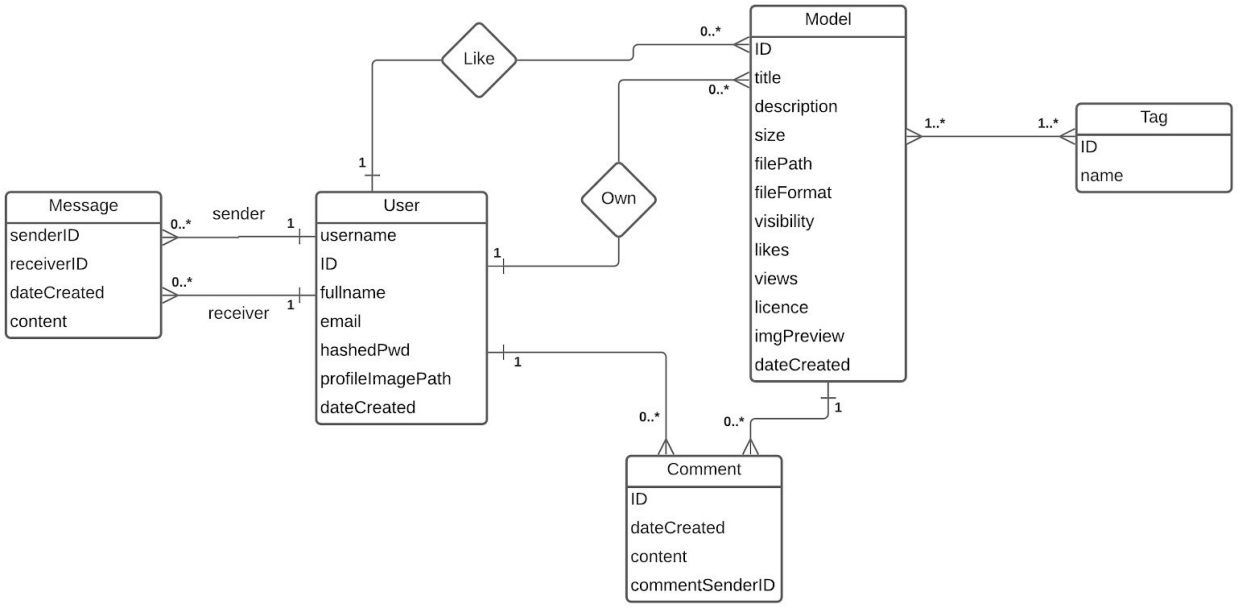
- **Redux Store:** This component will contain the state of the app and will be updated using the various reducers.
- **Reducers:** This reducer will be used to combine all reducers into one, so that we can generate a single store.
- **Authentication Reducer:** This reducer will be used to authenticate or logg out the user based on the actions that it receives.
- **Profile Reducer:** This reducer will control and update the state with regard to the profile component, based on the actions that are dispatched from this component.
- **Settings Reducer:** This reducers will control an update the state with regard to the settings component, based on the actions that are dispatched from that component.

- **Presentation Reducer:** This reducer will control and update the state with regard to the presentation component, based on the actions that are dispatched from that component.
- **Actions:** This component will define all the actions that enable the reducers to change the state of the application.

2.2 Server

- **server:** Server layer that manages application logic and wrapper on persistent data.
    - **server.item:** Model feed provider.
    - **server.cache:** Manages in-memory caching of persistent data.
    - **server.facebook:** Manages Facebook authentication and publishing.
    - **server.models:** Data transport objects, summarized models for “model” package.
    - **server.oauth2:** OAuth2 open authentication server provider implementation
    - **server.resizer:** Image resizer service.
    - **server.storage:** Cloud storage uploader and wrapper for picture files.
    - **server.twitter:** Twitter authentication and publishing functionality
    - **server.modelviewer:** 3D asset viewer service.

2.3 Data



3. Class Interfaces

3.1 Client

3.1.1 View

Redux Provider	
Attributes	
store	This attribute contains the state of the entire app. This attribute is accessible over the entire app.
Methods	

LoginContainer	
Attributes	
store	The store contains the state of the entire app and is accessed through the Redux Provider.
Methods	
handleLogin()	This method is called after the user enters their credentials to login in. This method dispatches a Set Session action if the credentials are valid.
handleRegister()	This method is called when the user enters the required information for a registration and dispatches a Register User action.

LoginComponent	
Attributes	
email	The email entered by the user.
password	The password entered by the user.
Methods	
alert_emptyEmail()	This method displays an alert if the user clicks

	Login without entering the email information.
alert_emptyPassword()	This method displays an alert if the user clicks Login without entering the password.
sanitizeInput(email, password)	This method is used to check whether the credentials entered by the user are correct. If they are then they call the handleLogin() methods which is passed down by its parent component, LoginContainer.

RegisterComponent	
Attributes	
name	The name of the user entered in the Registration Form.
email	The email of the new user entered in the Registration Form.
password	The password of the new user entered in the Registration Form.
Methods	
alert_emptyEmail()	This method displays an alert if the user clicks Register without entering the email information.
alert_emptyPassword()	This method displays an alert if the user clicks Register without entering the email information.
sanitizeInput(email, password)	
db_regUser()	This method is called when the user presses on the button Register. This method is taken as a prop from its parent component, LoginContainer.

SettingsContainer	
Attributes	
store	The store contains the state of the entire app and is accessed through the Redux Provider.
Methods	
handleMode()	This method is used to update the settings of the app.
handlePasswordChange()	This method is used to change password of the user. When the user enters a new password, the Set Profile Data action is

	dispatched with the new data.

SettingsInfoComponent	
Attributes	
name	The name of the user that is logged in.
email	The password of the user that is logged in.
Methods	
changeMode()	This method call the handleMode() method that is passed down from its parent component, SettingsContainer
changePassword(String)	This method calls the handlePasswordChange() method that is passed down from its parent component, SettingsContainer.

ProfileContainer	
Attributes	
store	The store contains the state of the entire app and is accessed through the Redux Provider.
Methods	
handleUpload()	This method is called when the user clicks on the Upload button to upload a model or presentation to their account. This method dispatches a Set Model Data or Set Presentation Data action based on the type of the object that is being uploaded.
handleDownloads()	This method is called when the user clicks on the Download button and downloads the selected models/presentations to the users device.
handleShares()	This method is called when the user wants to make one of their model publicly available. This method dispatches a Set Model Data or Set Presentation Data action, based on the type of the object that is being shared.
handleCreatePresentation()	This method is called when the user clicks on Save button after they have been working on a presentation. This method dispatches Set

	Presentation Data action.

InfoComponent	
Attributes	
name	The name of the user that is logged in.
username	The username of the user that is logged in.
picture	The profile picture of the user that is logged in.
uploads	The Models uploaded by the user.
presentation	The presentations available on the users account.
Methods	

BrowseComponent	
Attributes	
category	The user can choose a category to filter the available models.
archive	This attribute holds the models archived by the user.
ModelList[]	This attribute contains a list of the available Models.
Methods	
getModels()	This method return all the publicly available models.
getModelById()	This method returns a single model selected by the user.
searchModel()	This method filters the displayed models by the selected category.

UploadComponent
Attributes

Obj file	The type of the file that is being uploaded.
title	The title of the object being uploaded.
license	The type of the license of the object being uploaded.
visibility	Option to make the object publicly available or private.
description	Description of the object being uploaded.
tags	The tags of the object being uploaded.
Methods	
handleModelUpload()	This method is called when the user clicks on the Upload button. This method calls the handleUpload() methods that is passed down from its parent component, profileContainer.

CreatePresentation	
Attributes	
List<Models>	This attribute is a list of models that are being used in the presentation.
List<keyframes>	This attribute is a list of keyframes of the models in different positions.
Methods	
handleCreatePresentation()	This method is called when the user clicks on Create Presentation button. This method calls handleCreatePresentation() method that is passed down by its parent component, ProfileContainer.

DisplayModelComponent	
Attributes	
title	The title of the Model that is being displayed.
description	The description of the Model that is being displayed.
tags	The Tags of the Model that is being displayed.

views	The number of view of the Model that is being displayed.
likes	The number of likes of the Model that is being displayed.
comments	The comments of the Model that is being displayed.
Methods	
handleDownload()	This method is called when the user clicks on the Download button. This method calls the handleDownloads() method that is passed down from its parent component, ProfileContainer.
handleShare()	This method is called when the user clicks on Share button. This method calls the handleShares() method that is passed down by its parent component, ProfileContainer.

ExploreContainer	
Attributes	
store	The store contains the state of the entire app and is accessed through the Redux Provider.
Methods	
handleClick()	This method is called when the user clicks on a model being displayed on the screen. This method redirect to a new screen where the DisplayModelComponent displays the clicked model.
handleDownloads()	This method is called when the user clicks on the Download button and downloads the selected models/presentations to the users device.

Featured	
Attributes	
store	The store contains the state of the entire app and is accessed through the Redux Provider.
Methods	

handleClicks()	This method is called when the user clicks on one of the Models that are being featured. This method calls the handleClick() method that is passed down by its parent component, ExploreComponent.
handleDownload()	This method is called when the user clicks on the Download button on one of the Models that are being displayed on the featured section. This method calls the handleDownloads() method passed down by its parent component, the ExploreComponent.

ModelComponent	
Attributes	
image	Image of the model.
title	The title of the image.
Methods	
handleModelClick()	This method is called when the user clicks on the Model. This method calls the method handleClick() that is passed down by its parent components, ExploreContainer or FeaturedComponent.
handleModelDownload()	This method is called when the user clicks on the Download button on the Model. This method calls the method handleDownload() that is passed down by its parent components, ExploreContainer or FeaturedComponent.

3.1.2 Redux

Actions	
Attributes	
Show Loading	This type of action is used to show a loading screen while the app is fetching data from the database.
Hide Loading	This type of action is used to stop displaying the loading screen.
Clear Session	This type of action is used to end the authenticated session and log the user out.
Set Session	This type of action is used to start or restart a

	session with a new token.
Fetch Profile Pending	This type of action is used while we are waiting for the data for be fetched from the database.
Fetch Profile Success	This type of action is used when we have successfully received the Profile data from the database.
Fetch Profile Error	This type of action is used when the app fails to retrieve Profile data from the database.
Fetch Model Pending	This type of action is used while we are waiting for the Model to load from the database.
Fetch Model Success	This type of action is used when we have successfully retrieved the Model from the database.
Fetch Model Error	This type of action is used when we have failed to retrieve the Model from the database.
Set Profile Data	This type of action is used to update the profile data on the redux store and the database.
Set Model Data	This type of action is used to update the Model data in the database.
Set Presentation Data	This type of action is used to set the presentation data on the database.
Methods	

AuthenticationReducer	
Attributes	
Methods	
setSession()	This method is used to start a session for an authenticated user. This method is called when the SetSession action is dispatched.
clearSession()	This method is used to end the authentication session and log the user used. This method is called when the ClearSession action is dispatched.

ProfileReducer

Attributes	
Methods	
fetchProfilePending()	This method is called when the Fetch Profile Pending action is dispatched. This method updates the store with the information that the app is pending to fetch the profile data.
fetchProfileSuccess()	This method is called when the Fetch Profile Success action is dispatched. This method updates the store with the retrieved profile data.
fetchProfileError()	This method is called when the Fetch Profile Error action is dispatched. This method updates the store with the information that loading the profile data has failed.
fetchModelPending()	This method is called when the Fetch Model Pending action is dispatched. This method updates the store with the information that we are pending to load the model data.
fetchModelSuccess()	This method is called when the Fetch Model Success action is dispatched. This method updates the store with the retrieved model data.
fetchModelError()	This method is called when the Fetch Model Error action is dispatched. This method updated the store with the information that the loading Model has failed.

SettingsReducer	
Attributes	
Methods	
fetchProfilePending()	This method is called when the Fetch Profile Pending action is dispatched. This method updates the store with the information that the app is pending to fetch the profile data.
fetchProfileSuccess()	This method is called when the Fetch Profile Success action is dispatched. This method updates the store with the retrieved profile data.
fetchProfileError()	This method is called when the Fetch Profile Error action is dispatched. This method updates the store with the information that loading the profile data has failed.

setProfileData()	This method is called when the Set Profile Data action is dispatched. This method updates the store with the new profile data.

PresentationReducer	
Attributes	
Methods	
fetchPresentationPending()	This method is called when the Fetch Presentation Pending action is dispatched. This method updates the store with the information that the app is pending to fetch the presentation data.
fetchPresentationSuccess()	This method is called when the Fetch Presentation Success action is dispatched. This method updates the store with the retrieved presentation data.
fetchPresentationError()	This method is called when the Fetch Presentation Error action is dispatched. This method updates the store with the information that loading the presentation data has failed.
setPresentationData()	This method is called when the Set Presentation Data action is dispatched. This method updates the store with the new presentation data.

Reducers	
Attributes	
Methods	
combineReducers()	This method is used to combine all the reducers into a single reducer.

Redux Store	
Attributes	
state	This attribute contains the state of the entire app.

Methods

3.2 Server

Type	Name	Description
pkg `server`	-	-
class	ApiClientService	Provides access layer for REST API Clients (Apps).
class	LocationService	Manages persistent service layer operations and caching implementation for Model objects.
class	PostService	Manages all service layer operations for Model Feed
class	PostVerbalizer	Converts a model post into text using it's text features
class	TimeSys	Time period verbalization, time-zone computations for timestamps
class	UserAgent	Manages all server layer computations for User
class	FeedAgent	Provides model sharing functionality, model feed.
pkg `server.item`	-	-
class	ItemAgent	Provides model information, search queries.
class	ItemProvider	Holds templates for different categories of models.
pkg `server.cache`	-	-
abstract	AbstractCache	A generic interface for PSR-16 compliance.
class	CacheAgent	Provides caching service for common accesses
class	UserCache	-
class	ItemCache	-
class	PostCache	-

pkg `server.facebook`	-	-
class	FacebookAuthAgent	Provides authentication to Facebook social network
class	FaceboookPostPublisher	Publishes a post to facebook when the user authenticates.
pkg `server.models`	-	-
class	ApiClientDataTransfer	Data Transfer Object (blob) for 3d assets
class	PostDataTransfer	Data Transfer Object (blob) for Posts
enum	UserProfilePrivacy	Describes privacy settings of the users profile.
enum	UserPostPrivacy	Describes privacy settings of the posts
class	Post	Entity class for model post metadata
class	Comment	Entity class for comment on a model post.
class	Like	Entity class for ‘like’ reaction on a model post.
class	Views	Entity class for ‘view’ count on a model post.
enum	SharingSettings	Sharing options for a model post.
enum	DownloadSettings	Download options for a model post.
enum	GeometryDefFormat	Describes 3D data interchange file formats.
class	AssetFormatConverter	Converts a 3D data interchange format to another.
pkg `server.oauth2`	-	-
class	OAuth2Agent	OAuth2 authentication implementation
class	InvalidOAuthCliException	-
class	InvalidOAuthSrvException	-
pkg `server.resizer`	-	-
class	GnrImageRes	A PHP image resizer that

		resizes images upon upload requests
class	ModelSnapSRes	Resizes the snapshot of the model placeholder image
class	ProfilePicRes	Resizes user profile pictures uploaded to server by 128x128 px.
pkg `server.twitter`	-	-
class	TwitterAuthenticationAgent	Connects Twitter accounts to the user account to be granted read, write access.
class	TweetPublisherAgent	Makes users to post a Tweet after publishing a model.
pkg `server.modelviewer`	-	-
class	ModelViewerAgent	WebGL 1.0 interactive model viewer
class	ModelRenderer	Renderer for the scene
class	Camera	-
class	Lights	-
class	Animate	-
class	Controls	Specifies zoom, pan, scroll speed for user inputs

4. Glossary

5. References

[1] Visual Learning Center by Visme. 2020. *24 Presentation Statistics You Should Know In 2020*. [Online] Available at: <https://visme.co/blog/presentation-statistics/> [Accessed 27 December 2020].

[2] Presentationpanda.com. 2020. *Presentation Statistics (Based On Real-World Survey Data!)*. [Online] Available at: <https://presentationpanda.com/blog/new-presentation-statistics/> [Accessed 27 December 2020].

[3] What is UML. 2020

[Online] Available at: <https://www.uml.org/what-is-uml.htm> [Accessed 8 February 2021].