Topic	IMAGE TRACKING AR		
Class Description	**Students will learn about image tracking based augmented reality. Students will learn to create an image tracker for web AR to play video using the image marker.**		
Class	**C167**		
Class time	**45 mins**		
Goal	• Learn about image tracking augmented reality web apps. • Learn to create a basic web based AR app using Image trackers • Learn to play video as AR scenes.		

Resources Required	• Teacher Resources
	  ○ Visual Studio Code Editor
	  ○ laptop with internet connectivity
	  ○ smartphone
	  ○ earphones with mic
	  ○ notebook and pen
	• Student Resources
	  ○ Visual Studio Code Editor
	  ○ laptop with internet connectivity
	  ○ smartphone
	  ○ earphones with mic
	  ○ notebook and pen		

Class structure	**Warm-Up** **Teacher-led Activity** **Student-led Activity** **Wrap-Up**		**5 mins** **15 mins** **20 mins** **5 mins**
WARM-UP SESSION - 10 mins			
CONTEXT • **Web based A-Frame image tracking AR.**			

Teacher Starts Slideshow  **Slide 1 to 3** Refer to speaker notes and follow the instructions on each slide.	
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** • Greet the student. • Revision of previous class activities. • Quizzes.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session  **Slide 4 to 11**	
Following are the session deliverables: • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	

Class Steps	Teacher Action	Student Action
Step 1: Warm-Up (5 mins)	Today we will be learning about a different type of web based AR which is image tracking. As simple as it sounds, in this we can scan any image and render content over that. Not only images, but we can also pick any drawing or any picture and use it to show the content over that.	

	This is similar to the markers where we were showing the content over the marker. In this we will be learning how to tell the computer to identify the image Are you excited?	**ESR:** Yes.
	Let's get started then.	
Teacher Ends Slideshow 		
TEACHER-LED ACTIVITY - 15 mins		
Teacher Initiates Screen Share		
CHALLENGE • **Create A-Frame Web AR scene.** • **Play video using an image tracker.**		
Step 2: Teacher-led Activity (15 mins)	*<The teacher clones the code from the Teacher Activity 1.>* ***[Teacher Activity 1]*** We will be learning how to use an image as an image tracker and play the 3D video content over that image. To start with, let's quickly set up the meta information in the <head> tag. For 3D video content to be compatible with both Android and iOS devices, we	

	need to add the meta information to enable Apple mobile content.	

```
<!-- iOS has a lot of restrictions on playing videos in the browser.
To play an inline video texture, we must set the meta tag.
A-Frame will inject this if missing.-->
<meta name="apple-mobile-web-app-capable" content="yes" />
```

	We can add in the <body> to show the loading descriptor till the time video content is loaded. Let's add some CSS styling also, which have been studied in earlier classes, for the loading descriptor in the <head>.	

```
<!-- minimal loader shown until image descriptors are loaded.
  Loading may take a while according to the device computational power -->

<div class="arjs-loader">
  <div>Loading, please wait...</div>
</div>
```

```
<!-- style for the loader -->
<style>
  .arjs-loader {
    height: 100%;
    width: 100%;
    position: absolute;
    top: 0;
    left: 0;
    background-color: rgba(0, 0, 0, 0.8);
    z-index: 9999;
    display: flex;
    justify-content: center;
    align-items: center;
  }

  .arjs-loader div {
    text-align: center;
    font-size: 1.25em;
    color: white;
  }
</style>
```

We can now add the basic A-Frame scene and add components to **enable arjs** and **disable vr-mode-ui.**

We can also set the **renderer** component and its property **logarithmicDepthBuffer** as true to enhance the rendering of the 3D entities in the scene.

In simple terms, a **logarithmic depth buffer** provides more accuracy for objects near the camera.

The teacher sets the <a-scene> element component:

- *vr-mode-ui="enabled: false;"*

- *renderer=”logarithmicDepthBuffer: true;”*
- *embedded*
- *arjs=”tackingMethod: best; sourceType: webcam; debugUIEnable: false;”*

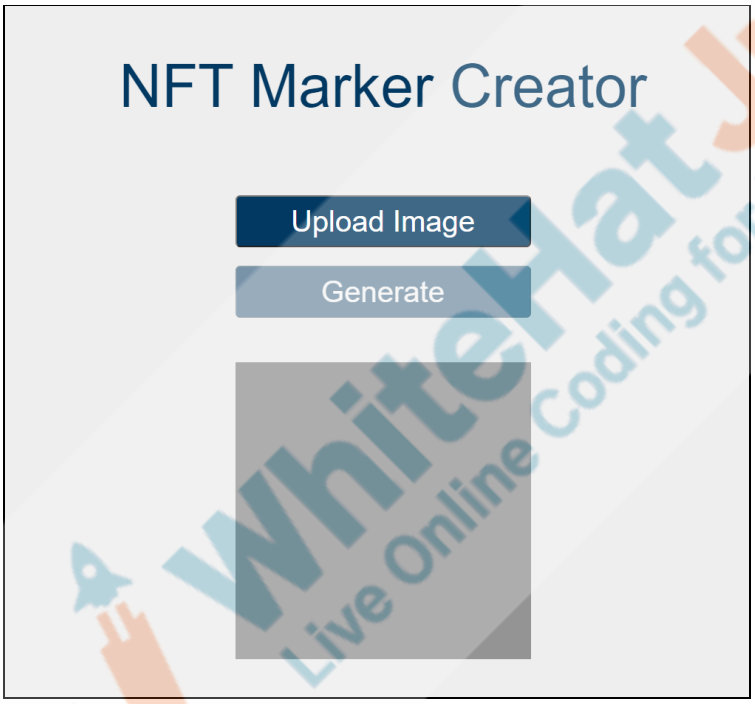
```
<!-- a-frame scene -->
<a-scene
  vr-mode-ui="enabled: false;"
  renderer="logarithmicDepthBuffer: true;"
  embedded
  arjs="trackingMethod: best; sourceType: webcam;debugUIEnabled: false;">

  <!-- static camera that moves according to the device movemenents -->
  <a-entity id="camera" camera position="0 0 10"></a-entity>




</a-scene>
```

Now once <head> and <a-scene> setup is done, to make an image tracking application the first thing that we need is an image.

You should try to pick a good quality image to have better tracking.

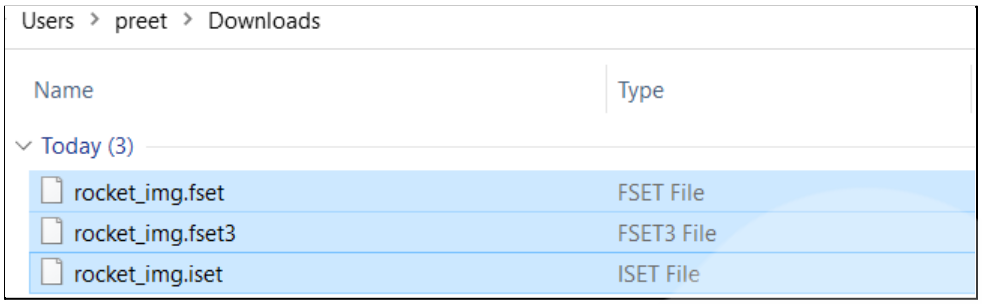
This image will be used as the image tracker.

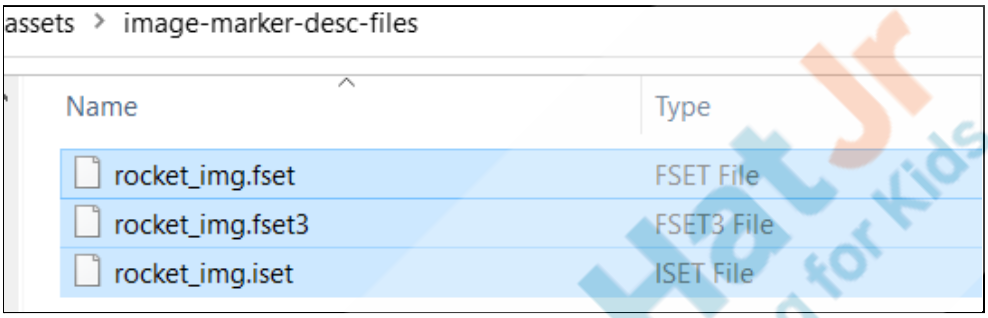
To convert this image into a tracker, we will be using an NFT converter.

	Link: https://carnaux.github.io/NFT-Marker-Creator/ NFT stands for **natural feature tracking**. This is a technology which helps to create image trackers.	
		
	We can upload the image and generate the NFT marker. ***Note**: Do not use images having very high pixel height and width value to generate the nft marker files.* Resize the image to reduce the time required to generate the nft marker files. *<The teacher opens the image from the system and uploads it in NFT.>*	

	The NFT marker creator will create 3 files with **.fset**, **.fset3** and **.iset** extension to be used as marker descriptor information. Allow the permission to download multiple files.	
		

	Check the downloaded files in the download folder. Copy and paste these 3 files into the working directory.	





	Now we are going to use the video asset to be over the image. Can you tell me how we add assets in A-Frame using an asset management system? Till now we have used 3D models, images and audio as assets in <a-assets>. Today we will be using <video> to add the video src files and set other properties to play the video. For <video> we can set: • **src**: the file path to video; • **preload**: whether to preload the video content before rendering the scene;	**ESR**: We use <a-assets> .

- **loop**: whether to play the video again and again;
- **playsinline** and **webkit-playsinline**: to play the video right where it is and avoid video to play in full screen mode; and
- **crossorigin**: sets the Cross-Origin Resource Sharing permission to share the information on the web browser. The crossorigin attribute is valid on the <audio>, <img>, <link>, <script>, and <video> elements.

```
<a-assets>

  <video id="video1"
    src="./assets/videos/Rocket_Launching_Animation.mp4"
    preload="auto"
    loop="true"
    playsinline
    webkit-playsinline
    autoplay
    crossorigin="anonymous">
  </video>

</a-assets>
```

Now we will play the video with the nft marker information.

For this we will need the **aframe-ar-nft.js** library.

	Link: https://raw.githack.com/AR-js-org/AR.js/master/aframe/build/aframe-ar-nft.js Then we will use the **<a-nft>** tag to add the nft marker files. For <a-nft> we can set: • **type**: nft • **url**: file path to nft image descriptor created before. ***Note**: While adding the nft file descriptor in the src path, the filename (excluding extension) is used only once for all 3 files.* Now let's understand how image information is stored for tracking. Images are stored as a set of pixel values in the form of rows and columns. This is known as the **image matrix**. We can have multiple matrices for better tracking of images. While using <a-nft> we can also set the tracking properties. In <a-nft> we can set: **smooth**: turns on/off camera smoothing, default: false	

	smoothCount: number of matrices for smooth tracking, default: 5 **smoothTolerance**: distance tolerance for smoothing, if smoothThreshold number of matrices are less than tolerance, tracking will stay still, default: 0.01 **smoothThreshold**: threshold for smoothing, will keep still unless enough matrices are more than tolerance, default: 2 Now to set the video entity, we will use **<a-video>** as the child of the <a-nft> and set the src id, height, width, position and rotation to set its orientation.	

```
<script src="https://raw.githack.com/AR-js-org/AR.js/master/aframe/build/aframe-ar-nft.js"></script>
```

```
<!-- a-nft is the anchor that defines an Image Tracking entity -->
<!-- on 'url' use the path to the Image Descriptors created before. -->
<!-- The file path should end with the name WITHOUT the extension & ONLY ONCE
  e.g. if file is rocket_img.fset' the path should end with rocket_img -->

<a-nft id="nft1"
  type="nft"
  url="./assets/image-marker-desc-files/rocket_img"
  smooth="true"
  smoothCount="10"
  smoothTolerance=".01"
  smoothThreshold="5">

  <!-- As a child of the a-nft entity, define the content to show. -->

  <a-video id="vid1"
    src="#video1"
    width="600"
    height="509"
    position="0 0 -30"
    rotation="-90 0 0"
    >
  </a-video>

</a-nft>
```

Now let's add one A-Frame component, "play-on-click", which can help to play and pause the video on click.

In the **schema** of the component we can take **isPlaying** boolean variable with default value as false, as the data for the component.

<The teacher adds the src file in index.html.>

<The teacher registers "play-on-click" components and adds the schema & .init(), play() and onClick() functions.>

```
<script src="./play-on-click.js"></script>
```

```
AFRAME.registerComponent("play-on-click", {
  schema: {
    isPlaying: { type: "boolean", default: false }
  },
  init: function() {

  },
  play: function() {
    window.addEventListener("click", this.onClick);
  },
  onClick: function(evt) {

  }
});
```

Now we can take the **videoEl** variable and select the video src to be played using onClick() and .init() methods.

In **onClick()** function:

- Select the **isPlaying** attribute.
- Use if/else condition to check the value of the isPlaying variable.
- Set the isPlaying value inside if/else condition and use .play() method to play the video src.

Then call the onClick() function inside .init() method and attach the component to the <a-video> entity.

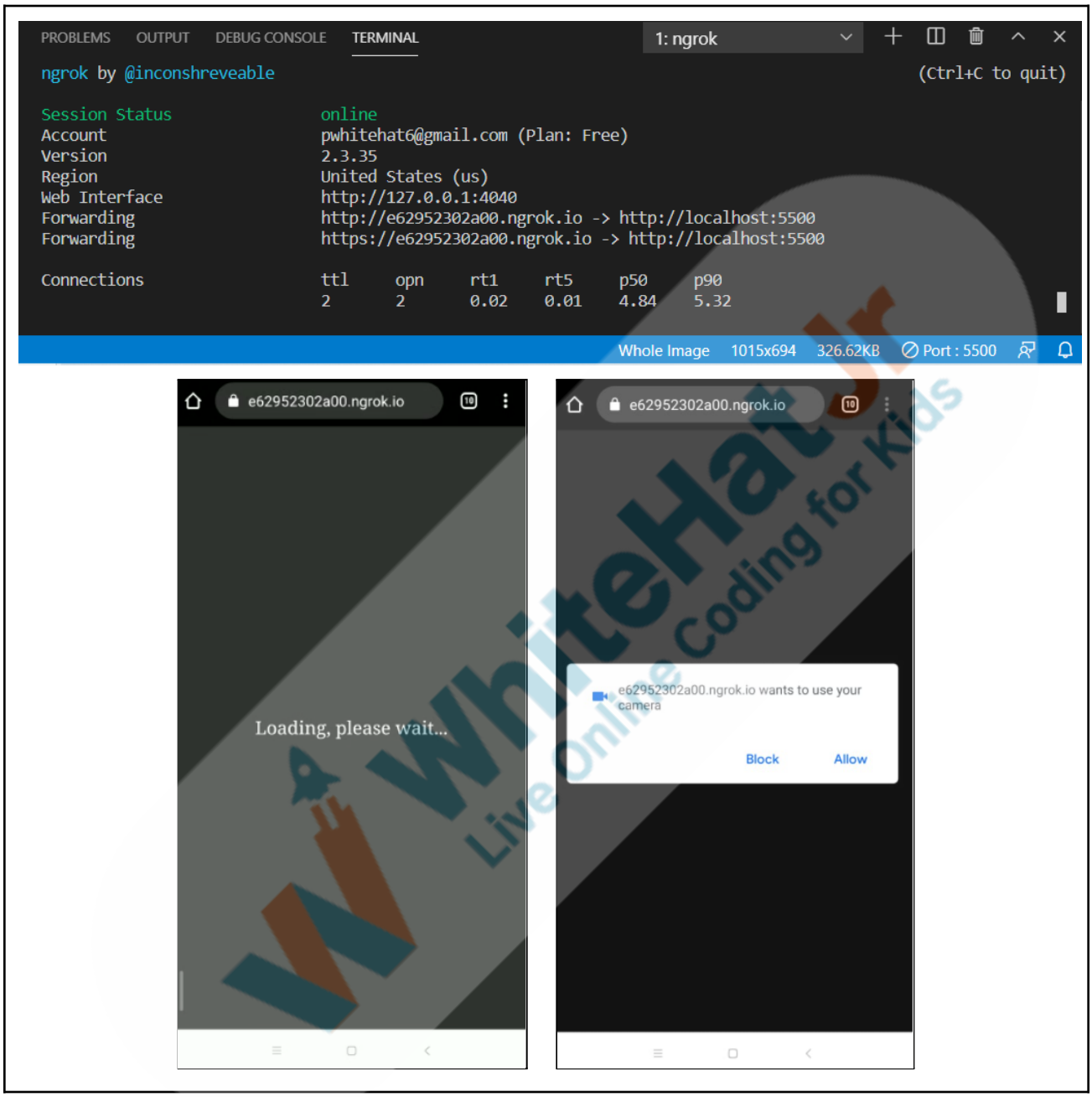
```
init: function() {
  this.videoEl = this.el.getAttribute("material").src;
},
```

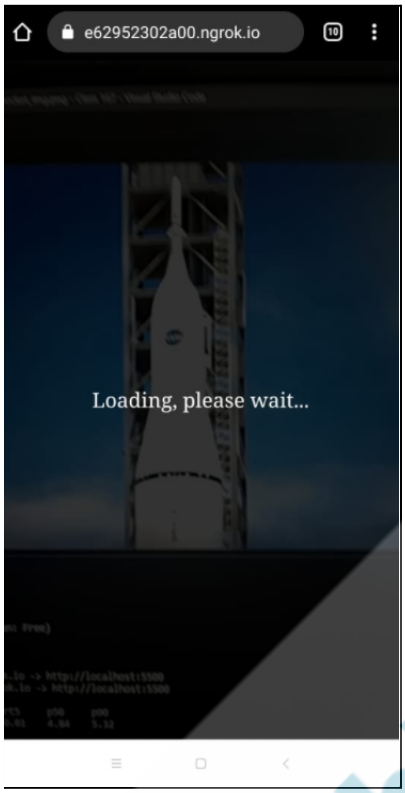
```
onClick: function(evt) {
  if (!this.videoEl) {
    return;
  }

  var isPlaying = this.el.getAttribute("play-on-click").isPlaying;

  this.el.object3D.visible = true;

  if (!isPlaying) {
    this.el.setAttribute("play-on-click", {
      isPlaying: true
    });
    this.videoEl.play();
  } else {
    this.el.setAttribute("play-on-click", {
      isPlaying: false
    });
    this.videoEl.pause();
  }
}
```

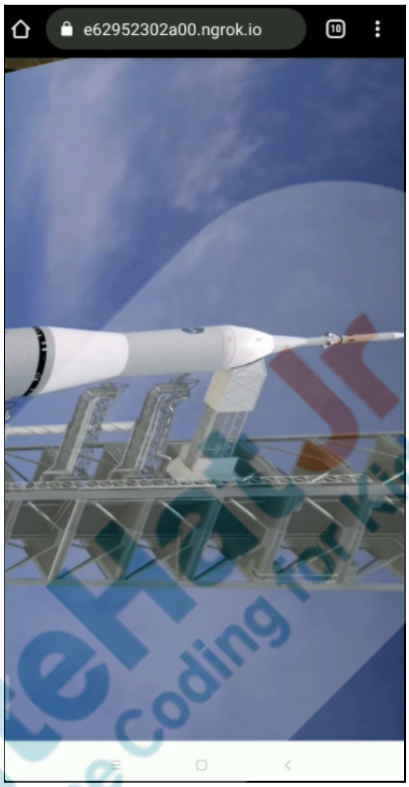
```
init: function() {
  this.videoEl = this.el.getAttribute("material").src;
  this.onClick = this.onClick.bind(this);
},
```

```
<!-- As a child of the a-nft entity, define the content to show. -->

<a-video id="vid1"
  src="#video1"
  width="600"
  height="509"
  position="0 0 -30"
  rotation="-90 0 0"
  play-on-click>
</a-video>
```

	We can now test the output using ngrok. To see the output: • Use **ngrok** to run the application. • Open **HTTPS URL** in your smartphone/laptop and give permission to use the camera. • Open the **original image** that was used to create the nft image marker and point the camera towards it. The **window screen can be clicked** or the **phone screen can be tapped** to **play and pause the video.** **Output Reference** ***Note 1**: The output video can be played and paused multiple times on touch.* ***Note 2**: Switch on rotation mode and use the phone in the landscape mode to better cover the video content.*	



<table>
<tr><td colspan="3"> 


</td></tr>
<tr><td></td><td>That was very interesting!

Now you will also create the nft marker and play and pause the video on click (windows) or touch (smartphone).

Are you excited?</td><td>ESR: Yes!</td></tr>
<tr><td colspan="3">Teacher Stops Screen Share</td></tr>
<tr><td></td><td>Now it's your turn. Please share your screen with me.</td><td></td></tr>
<tr><td colspan="3">Teacher Starts Slideshow
Slide 12 to 13
Refer to speaker notes and follow the instructions on each slide.</td></tr>
</table>

We have one more class challenge for you. Can you solve it? Let's try. I will guide you through it.		
Teacher Ends Slideshow 		
STUDENT-LED ACTIVITY - 20 mins		
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start screen share.** • **Teacher gets into fullscreen.**		
ACTIVITY • **Create an image tracking based A-Frame Web AR scene.** • **Play 3D video in the AR scene using NFT image markers.**		
Step 3: Student-Led Activity (20 mins)	*The teacher guides the student to clone the code from Student Activity 1.* *[Student Activity 1]* ***Note**: The student will repeat the activity performed by the teacher.*	
	What should we do to start making the AR scene? *Guide the student to add the meta info and aframe-ar-nft.js src files in the <head>.* *Guide the student to add CSS styles and element for loading the descriptor.*	**ESR**: Add the supported library in the A-Frame scene.

```
<meta name="apple-mobile-web-app-capable" content="yes" />

<!-- import aframe and then ar.js with image tracking / location based features -->
<script src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>

<script src="https://raw.githack.com/AR-js-org/AR.js/master/aframe/build/aframe-ar-nft.js"></script>

<script src="./play-on-click.js"></script>

<!-- style for the loader -->
<style>
  .arjs-loader {
    height: 100%;
    width: 100%;
    position: absolute;
    top: 0;
    left: 0;
    background-color: rgba(0, 0, 0, 0.8);
    z-index: 9999;
    display: flex;
    justify-content: center;
    align-items: center;
  }

  .arjs-loader div {
    text-align: center;
    font-size: 1.25em;
    color: white;
  }
</style>
```

```
<!-- minimal loader shown until image descriptors are loaded.
  Loading may take a while according to the device computational power -->

<div class="arjs-loader">
  <div>Loading, please wait...</div>
</div>
```

	*Guide the student to add the **aframe-ar.js** library and attach the **embedded arjs** component to the scene element.*	

```
<!-- a-frame scene -->
<a-scene
  vr-mode-ui="enabled: false;"
  renderer="logarithmicDepthBuffer: true;"
  embedded
  arjs="trackingMethod: best; sourceType: webcam;debugUIEnabled: false;">

  <!-- static camera that moves according to the device movemenents -->
  <a-entity id="camera" camera position="0 0 10"></a-entity>

  <a-assets>

    <video id="video1"
      src="./assets/videos/Rocket_Launching_Animation.mp4"
      preload="auto"
      loop="true"
      playsinline
      webkit-playsinline
      autoplay
      crossorigin="anonymous">
    </video>

  </a-assets>
```

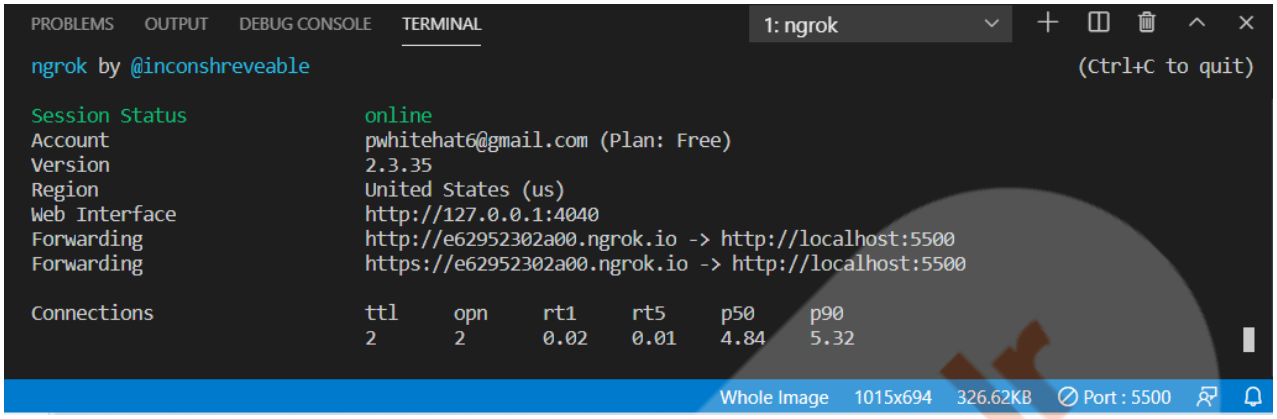
	What should we do now? Yes. *Guide the student to generate the nft marker using the image and add the files in the scene using <a-nft>.*	**ESR**: Create the NFT marker.

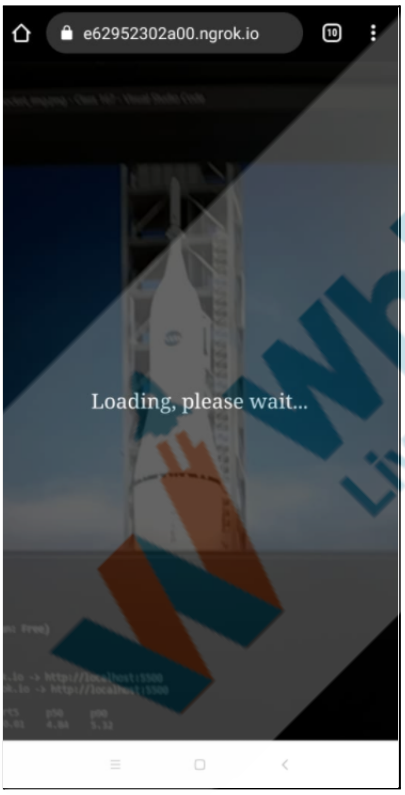

```
<!-- a-nft is the anchor that defines an Image Tracking entity -->
<!-- on 'url' use the path to the Image Descriptors created before. -->
<!-- The file path should end with the name WITHOUT the extension & ONLY ONCE
  e.g. if file is rocket_img.fset' the path should end with rocket_img -->

<a-nft id="nft1"
  type="nft"
  url="./assets/image-marker-desc-files/rocket_img"
  smooth="true"
  smoothCount="10"
  smoothTolerance=".01"
  smoothThreshold="5">

  <!-- As a child of the a-nft entity, define the content to show. -->

  <a-video id="vid1"
    src="#video1"
    width="600"
    height="509"
    position="0 0 -30"
    rotation="-90 0 0"
    play-on-click>
  </a-video>

</a-nft>
```

	Guide the student to write the "play-on-click" component.	

```
AFRAME.registerComponent("play-on-click", {
  schema: {
    isPlaying: { type: "boolean", default: false }
  },

  init: function() {
    this.videoEl = this.el.getAttribute("material").src;
    this.onClick = this.onClick.bind(this);
  },

  play: function() {
    window.addEventListener("click", this.onClick);
  },
  onClick: function(evt) {
    if (!this.videoEl) {
      return;
    }

    var isPlaying = this.el.getAttribute("play-on-click").isPlaying;

    this.el.object3D.visible = true;

    if (!isPlaying) {
      this.el.setAttribute("play-on-click", {
        isPlaying: true
      });
      this.videoEl.play();
    } else {
      this.el.setAttribute("play-on-click", {
        isPlaying: false
      });
      this.videoEl.pause();
    }
  }
});
```

	Guide the student to run and test the application using the https ngrok URL.	





Teacher Guides Student to Stop Screen Share

WRAP UP SESSION - 5 mins

Teacher Starts Slideshow **Slide 14 to 19** 		
Activity details **Following are the WRAP-UP session deliverables:** ● Appreciate the student. ● Revise the current class activities. ● Discuss the quizzes.		
WRAP-UP QUIZ Click on In-Class Quiz		
Continue WRAP-UP Session **Slide 20 to 25** 		
Activity Details **Following are the session deliverables:** ● Explain the facts and trivia ● Next class challenge ● Project for the day ● Additional Activity (Optional)		
FEEDBACK ● **Compliment the student for her/his effort in the class.** ● **Encourage the student to think and come up with their own solutions.**		
	You get a “hats-off”. Alright. See you in the next class.	*Make sure you have given at least 2 Hats Off during the class for:*  

		
PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections		
Teacher Clicks 		

Additional Activities	*Encourage the student to write reflection notes in their reflection journal using markdown.*
	
	Use these as guiding questions:
	• What happened today?
	  ○ Describe what happened.
	  ○ The code I wrote.
	• How did I feel after the class?
	• What have I learned about programming and developing games?
	• What aspects of the class helped me? What did I find difficult?
		The student uses the markdown editor to write their reflections in a reflection journal.

Activity	Activity Name	Links
Teacher Activity 1	Boilerplate Code	https://github.com/whitehatjr/PRO-C167-Boilerplate
Teacher Activity 2	Teacher Reference Code	https://github.com/whitehatjr/PRO-C167-Teacher-Ref
Teacher Activity 3	Output Reference	https://curriculum.whitehatjr.com/PRO+Asset/PRO+167+Output+Ref+(1).mp4
Teacher Activity 4	A-Frame NFT AR.js Link	https://raw.githack.com/AR-js-org/AR.js/master/aframe/build/aframe-ar-nft.js

Student Activity 1	Boilerplate Code	https://github.com/whitehatjr/PRO-C167-Boilerplate
Teacher Reference 1	Ngrok Updates	https://docs.google.com/document/d/1dlMry188lIEJl6rHEc3AkBashQSOwGQ40HQft29S8vQ/edit?usp=sharing
Teacher Reference 2	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/4ab0b6c9-ff7c-4413-be92-4ddfe5a57b60.pdf
Teacher Reference 3	Project Solution	https://github.com/whitehatjr/PRO-C167-Project-Solution
Teacher Reference 4	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/fb1a1d97-dfd1-4ac9-82a7-27b734024480.html
Teacher Reference 5	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/0a85ec69-dbbb-4a1e-bd21-a7f5fed0420a.pdf