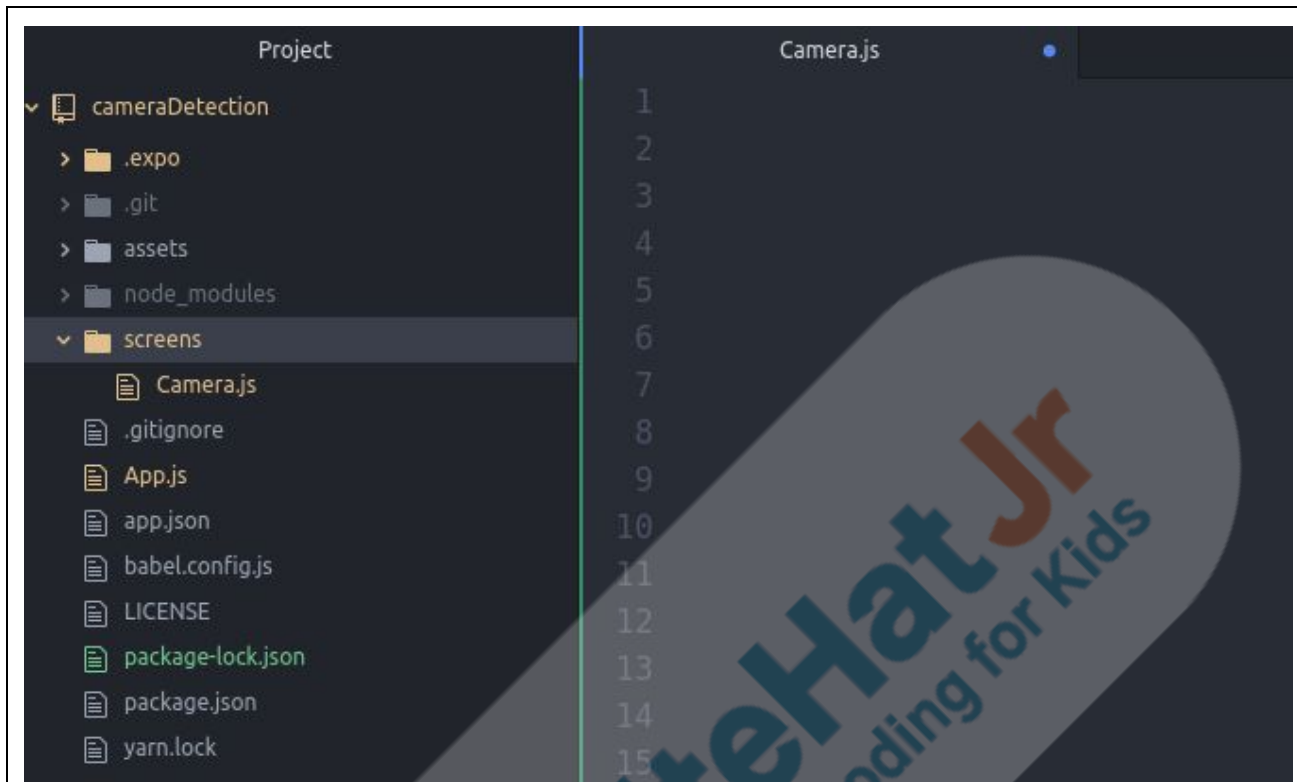


Topic	Live Image Prediction	
Class Description	Students learn to create a react native app and make POST requests on the API through the app.	
Class	C126	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> • Create react native app to pick Images • Host the API online using ngrok software. 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources <ul style="list-style-type: none"> ○ VS Code ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen • Student Resources <ul style="list-style-type: none"> ○ VS Code ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen 	
Class structure	Warm Up Teacher-led Activity Student-led Activity Wrap up	5 mins 15 min 15 min 5 min
<p style="text-align: center;"><u>CONTEXT</u></p> <ul style="list-style-type: none"> • Learn about creating an react native app and making POST request through the app. 		
Class Steps	Teacher Action	Student Action
Step 1: Warm Up (5 mins)	Hi <Student Name>. How are you doing today?	ESR: We created a classifier

	Can you tell me what we did in last class?	model. ESR: We wrote an API which provided data to the classifier model to make predictions.
	Awesome. Today we are going to make use of what we learned earlier to provide data to the classifier model and that is we are going to create a react native app which would allow you to upload digits pictures from your phones directly to the classifier model. Does that sound exciting?	ESR: Yes!
	It will be easy as you have already worked on react native . Let's get started.	
Teacher Initiates Screen Share		
<u>CHALLENGE</u> <ul style="list-style-type: none"> • Create a react native app which will choose the images from the library 		
Step 2: Teacher-led Activity (15 min)	We have already created the classifier model, And our API is ready too. So what's remaining?	ESR: The react native app.
	Yes. So let's start with it. Do you remember the command to create a react native app?	ESR: expo init <app name>
	Good. So we are going to name this app as "ImageDetection". <i><Teacher codes to create a new app.></i>	<i>The student helps the teacher with the code.</i>

	<p>Code :</p> <p>expo init imageDetection</p> <p>and then select the blank template.</p>	
<pre>ashura@zeros:~\$ expo init imageDetection</pre> <div style="border: 1px solid green; padding: 10px; margin: 10px 0;"> <p>There is a new version of expo-cli available (3.27.4). You are currently using expo-cli 3.21.5 Install expo-cli globally using the package manager of your choice; for example: `npm install -g expo-cli` to get the latest version</p> </div> <pre>? Choose a template: (Use arrow keys) ----- Managed workflow ----- > blank a minimal app as clean as an empty canvas blank (TypeScript) same as blank but with TypeScript configuration tabs several example screens and tabs using react-navigation ----- Bare workflow ----- minimal bare and minimal, just the essentials to get you started minimal (TypeScript) same as minimal but with TypeScript configuration</pre>		
	<p>Our app will have just one screen with a button which will take us to our phone gallery and let us pick an image.</p> <p><Teacher codes to create a screens folder and inside the folder creates a file called "Camera.js".></p>	<p><i>The student helps the teacher with the code.</i></p>



We'll need some libraries to get the permission from the device to pick the image and one to pick the image from the device.

Can you tell me the libraries which will help us do that?

The libraries are "expo-permissions" to gain permission from the device and "expo-image-picker".

We can install them using commands "expo install expo-permissions" and "expo install expo-image-picker".

<Teacher codes to install the libraries.>

Code:

expo install expo-permissions
expo install expo-image-picker

ESR:

Varied!

The student helps the teacher to install the libraries.

```
$ expo install expo-permissions
```

```
$ expo install expo-image-picker
```

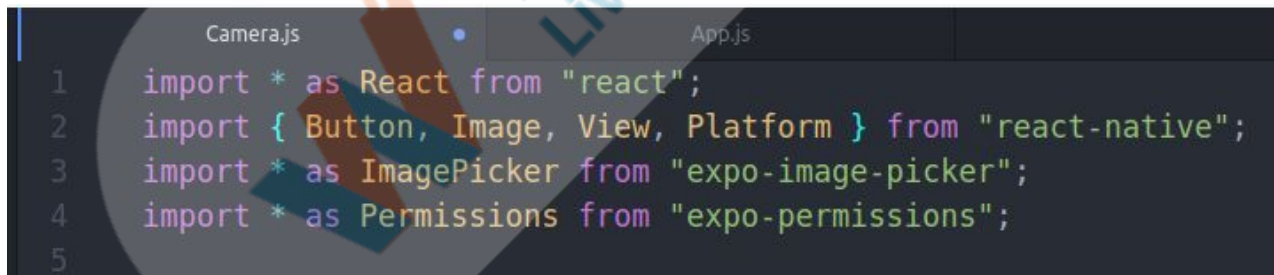
Now let's import them in the Camera.js file.
We'll also import Button, view and platform from react.

<Teacher codes to import the libraries in the app.>

Code:

```
import * as React from "react";
import { Button, View, Platform }
from "react-native";
import * as ImagePicker from
"expo-image-picker";
import * as Permissions from
"expo-permissions";
```

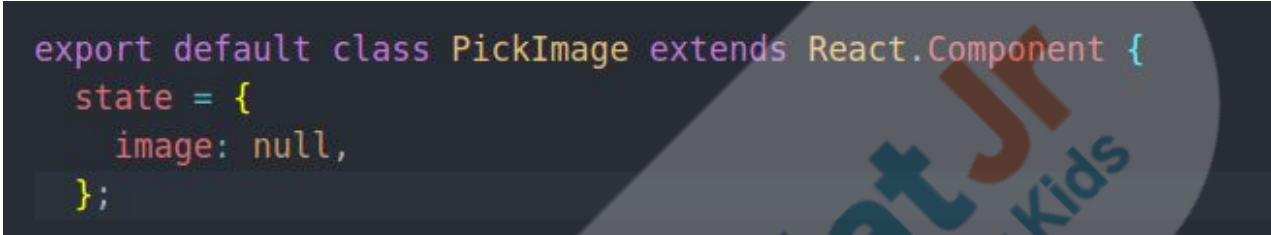
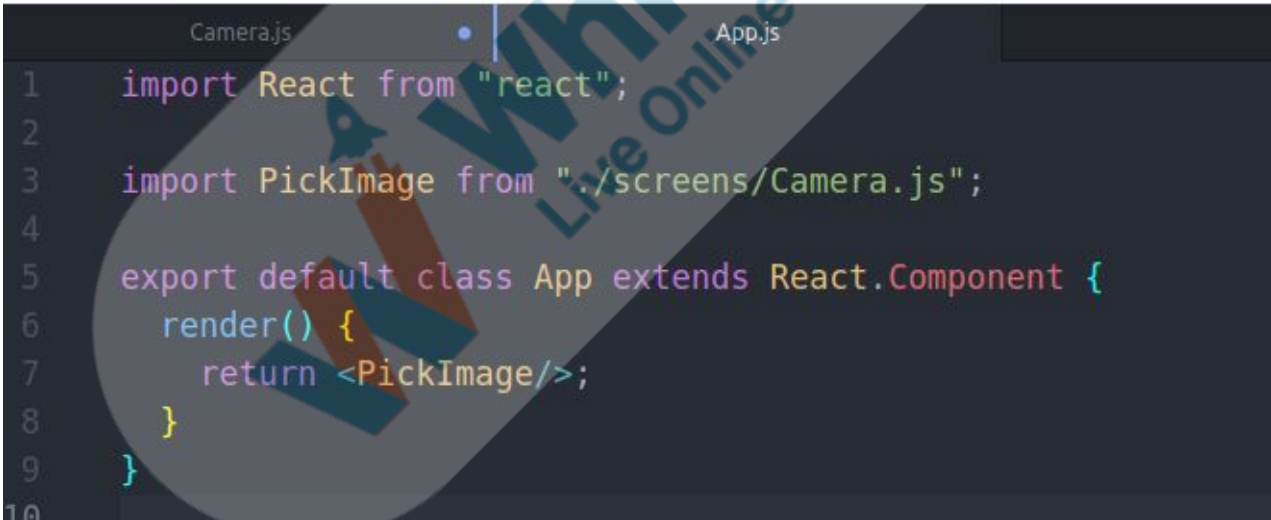
The student helps the teacher with the code.



Now we'll create a component called PickImage and inside it create a state for the image and set it to null. Here we'll store the uri of the image later.

<Teacher codes to create a PickImage and inside it creates a

The student helps the teacher with the code.

	<p><i>state for image and sets it to null.></i></p> <p>Code:</p> <pre>export default class PickImage extends React.Component { state = { image: null, };</pre>	
	 <pre>export default class PickImage extends React.Component { state = { image: null, }; };</pre>	
	<p>Now we have our component created so to render it we'll import it in the app.js file and pass it inside the render function of app.js.</p>	
	 <pre>Camera.js App.js 1 import React from "react"; 2 3 import PickImage from "../screens/Camera.js"; 4 5 export default class App extends React.Component { 6 render() { 7 return <PickImage/>; 8 } 9 } 10</pre>	
	<p>As we know that our component has 2 functions: render and return. Return function is used inside of the render function. Now inside the render function we'll</p>	<p><i>The student helps the teacher with the code.</i></p>

declare a variable image and set state value to it.
Then inside the return function we'll create a button.
Button has a title prop and onPress prop. On the on press prop we'll call a sudo _pickImage function. We'll write code for it later.

<Teacher codes to declare the image variable and set it's value to state. The codes to create a button.>

Code:

```
render() {
  let { image } = this.state;

  return (
    <View style={{ flex: 1,
alignItems: "center",
justifyContent: "center" }}>
      <Button
        title="Pick an image from
camera roll"
        onPress={this._pickImage}
      />
    </View>
  );
}
```



```
render() {
  let { image } = this.state;

  return (
    <View style={{ flex: 1, alignItems: "center", justifyContent: "center" }}>
      <Button
        title="Pick an image from camera roll"
        onPress={this._pickImage}
      />
    </View>
  );
}
```

To pick the images from the device we need to get the permissions from the device. So we'll write the `getPermissionAsync` function to gain the permission from the device. In this function we'll also show the message if the permission is not granted by the user.

<Teacher codes to write the `getPermissionAsync` function.>

Code:

```
getPermissionAsync = async () => {
  if (Platform.OS !== "web") {
    const { status } = await
    Permissions.askAsync(Permission
    s.CAMERA_ROLL);
    if (status !== "granted") {
      alert("Sorry, we need camera
      roll permissions to make this
      work!");
    }
  }
};
```

The student helps the teacher with the code.


```
getPermissionAsync = async () => {
  if (Platform.OS !== "web") {
    const { status } = await Permissions.askAsync(Permissions.CAMERA_ROLL);
    if (status !== "granted") {
      alert("Sorry, we need camera roll permissions to make this work!");
    }
  }
};
```

We want to call this function when the app starts so we'll have to call it inside the componentDidMount function. componentDidMount function is auto called when the app starts.

<Teacher codes to call the function inside componentDidMount function.>

Code:

```
componentDidMount() {
  this.getPermissionAsync();
}
```

The student helps the teacher with the code.

```
componentDidMount() {
  this.getPermissionAsync();
}
```

Now time to code for _pickImage function which we have already called on the button press. This function will help us to pick the image from the device. In this function we'll use the try catch block. We use this block to get the exceptions that code might throw. The try block will contain the code we want to execute and the catch block

The student helps the teacher with the code.

	<p>will contain the error or the exception which we'll denote by "e".</p> <p>Now inside the try block we'll use the <code>launchImageLibraryAsync()</code> function of the Image Picker library. And pass <code>mediaTypes</code>, <code>allowsEditing</code>, <code>aspect</code> and <code>quality</code> as its parameters and store it inside a variable called <code>result</code>.</p> <p>If the results are not cancelled then we'll set the data provided by the result to the image in the state. Here we'll also call an <code>uploadImage()</code> function and pass the <code>result.uri</code> to it. We'll code for the function later.</p> <p><i><Teacher codes for <code>_pickImage()</code> function.></i></p> <p>Code:</p> <pre><code>_pickImage = async () => { try { let result = await ImagePicker.launchImageLibraryA sync({ mediaTypes: ImagePicker.MediaTypeOptions.All }, { allowsEditing: true, aspect: [4, 3], quality: 1, }); if (!result.cancelled) { this.setState({ image: result.data }); console.log(result.uri) this.uploadImage(result.uri); } } }</code></pre>	
--	---	--

	<pre> } } catch (E) { console.log(E); } }; </pre>	
<pre> _pickImage = async () => { try { let result = await ImagePicker.launchImageLibraryAsync({ mediaTypes: ImagePicker.MediaTypeOptions.All, allowsEditing: true, aspect: [4, 3], quality: 1, }); if (!result.cancelled) { this.setState({ image: result.data }); console.log(result.uri); this.uploadImage(result.uri); } } catch (E) { console.log(E); } }; </pre>		
	<p>Let's code for the uploadImage function. This function takes the uri of the image as a parameter.</p> <p>As we have seen in previous class that the data to be sent on an API should be a form data as we are sending image files. So we'll create a variable called data and set new form data as its value.</p> <p>From the uri we also want the name of the image and its type. We'll get that by splitting the uri using split()</p>	<p><i>The student helps the teacher with the code.</i></p>

	<p>function.</p> <p>Then we'll create a fileToUpload object which will have the uri, image, type as key and the image name and type which we got before as its values. We need this data to send the image using an API.</p> <p>We'll then upload the fileToUpload object to the form data.</p> <p>Then using fetch we'll make a post request to the provided API. The method will be POST, body will contain the form data, header will contain content-type as multipart form data.</p> <p>We'll also have to resolve a promise which we will be doing using .then and getting the response in json form. Using catch we'll catch any exceptions or errors that might occur and console.log them.</p> <p><i><Teacher codes to write the upload Image function.></i></p> <p>Code:</p> <pre>uploadImage = async (uri) => { const data = new FormData(); let filename = uri.split("/")[uri.split("/").length - 1] let type = `image/\${uri.split('.')[uri.split('.').length - 1]}`</pre>	
--	--	--

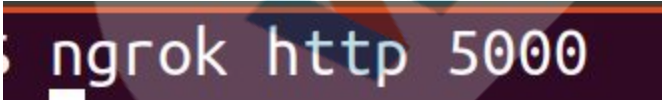
```
const fileToUpload = {
  uri: uri,
  name: filename,
  type: type,
};
data.append("digit",
fileToUpload);

fetch("https://07afd951a187.ngrok.io/predict-digit", {
  method: "POST",
  body: data,
  headers: {
    "content-type":
"multipart/form-data",
  },
})
.then((response) =>
response.json())
.then((result) => {
  console.log("Success:",
result);
})
.catch((error) => {
  console.error("Error:", error);
});
};
```

```
uploadImage = async (uri) => {
  const data = new FormData();
  let filename = uri.split("/")[uri.split("/").length - 1]
  let type = `image/${uri.split('.')[uri.split('.').length - 1]}`
  const fileToUpload = {
    uri: uri,
    name: filename,
    type: type,
  };
  data.append("digit", fileToUpload);
  fetch("https://07afd951a187.ngrok.io/predict-digit", {
    method: "POST",
    body: data,
    headers: {
      "content-type": "multipart/form-data",
    },
  })
  .then((response) => response.json())
  .then((result) => {
    console.log("Success:", result);
  })
  .catch((error) => {
    console.error("Error:", error);
  });
};
```

	Awesome so our app is ready. Now all that is left is to host the API online and then make a POST request to check the prediction model. Can you try to do that?	ESR: Yes!
	Awesome! Let's get you started.	-
Teacher Stops Screen Share		
	Now it's your turn. Please share your	

	screen with me.	
<ul style="list-style-type: none"> • Ask Student to press ESC key to come back to panel • Guide Student to start Screen Share • Teacher gets into Fullscreen 		
<p style="text-align: center;"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Hosting the API online using ngrok • Test the prediction model 		
Step 3: Student-Led Activity (15 min)	Teacher helps student code to create a full App.	Student Codes to create a full App.
	<p>To make the post request we'll need to host our API online so that we won't get unexpected errors.</p> <p>To do so we'll use an online hosting software called "ngrok". Let's install it in our system.</p> <p>For windows:</p> <ol style="list-style-type: none"> 1. Just download the zip and then double click on it to open. <p>For linux and mac:</p> <ol style="list-style-type: none"> 1. Download the zip and extract it. 2. Copy the extracted file and paste it in the bin folder. <p><Teacher helps student download zip from Student Activity 1.></p>	<p>Student downloads the zip from the Student Activity 1.</p> <p>Student codes to run the API server and host it on the ngrok.</p>

	<p>Now time to run the API server and host it on ngrok.</p> <p><i><Teacher asks student to open the app.js (which contains the routes) file from previous class ></i></p> <p>To do that first we'll run the file from the previous class where we created the API.</p> <p><i><Teacher asks student to open a different terminal and helps student to run the API server.></i></p> <p>Code: python3.8 app.py</p> <p>Now let's host on ngrok.</p> <p><i><Teacher asks student to open another terminal and helps student to host the API on ngrok.></i></p> <p>Code: ngrok http <port></p> <p>Note: Check on which port is your API server running and host that port on ngrok.</p>	
		

```
ashura@zeros:~/Desktop/API$ python3.8 app.py
/home/ashura/.local/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
warnings.warn("The max_iter was reached which means ")
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
/home/ashura/.local/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
warnings.warn("The max_iter was reached which means ")
* Debugger is active!
* Debugger PIN: 328-239-193
```

Now we need to copy the ngrok API and pass it in the fetch.

<Teacher helps the student to copy the ngrok API and pass it in the fetch function. This will help us to make the POST request.>

The student copies the ngrok API and passes it in the fetch function.

```
data.append("digit", fileToUpload);
fetch("https://07afd951a187.ngrok.io/predict-digit", {
  method: "POST",
  body: data,
  headers: {
    "content-type": "multipart/form-data",
  },
```

Now all we have to do is to test the code.
For that let's start our expo server using **expo start -c** and run the app on our phone.

You'll see the blue button on the App. After pressing it the app will ask for permissions. Allow the permissions and select an image with a digit in it.

Student codes to run the App and test the prediction model.

After that you can see the prediction output on the Terminal.

```
$ expo start -c
```

PICK AN IMAGE FROM CAMERA ROLL

	We can see that the model is making a prediction.	
<pre>Running application on vivo Y51L. file:///data/data/host.exp.exponent/cache/ExperienceData/%2540anonymous%252FcameraDetection-57504155-ddae-49fb-8b3a-3acf11c629ec/ImagePicker/87d79f86-f61e-4634-89a8-7af7b719862d.jpg Success: Object { "prediction": "2", }</pre>		
Teacher Guides Student to Stop Screen Share		
<p style="text-align: center;"><u>FEEDBACK</u></p> <ul style="list-style-type: none"> • Appreciate the student for their efforts • Identify 2 strengths and 1 area of progress for the student 		
Step 4: Wrap-Up (5 min)	So how was your experience in today's class?	<i>Student talks about the experience in the class.</i>
	I hope you had fun today. And we have seen how to create an API and also how to use it with a react native app. There are endless possibilities how	-

	you can use it. You can try using your imagination. See you in the next class.	
Project Overview	So Today we revised the concepts of react native and created an app which would make a post request on the API. Story:-	
<div> <div>Teacher Clicks</div> <div>✕ End Class</div> </div>		
Additional Activities	<i>Encourage the student to write reflection notes in their reflection journal using markdown.</i> Use these as guiding questions: <ul style="list-style-type: none"> • What happened today? <ul style="list-style-type: none"> - Describe what happened - 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? 	<i>The student uses the markdown editor to write her/his reflection in a reflection journal.</i>

Activity	Activity Name	Links
Teacher Activity 1	Reference code	https://github.com/whitehatjr/Realtim

		e-Image-detection
Student Activity 1	Ngrok	https://ngrok.com/download

