

DAILY ASSESSMENT FORMAT

Date:	06-07-2020	Name:	YAMUNASHREE N
Course:	Mat lab Onramp	USN:	4AL17EC097
Topic:	1. Course Overview 2. Commands 3. MATLAB Desktop and Editor 4. Vectors and Matrices	Semester & Section:	6th SEM & B section
Github Repository:	yamunashree-course		

FORENOON SESSION DETAILS

The screenshot shows a web browser window displaying the MATLAB Onramp course interface. The title bar reads "Verify Email Address - dhanyash... | https://in.mathworks.com/mwa... | MATLAB Onramp | Create MathWorks Account". The main content area is titled "MATLAB Onramp [2% complete]" and shows the "1.1 Course Overview" section. Below the title, there is a large image illustrating the MATLAB environment with three overlapping windows: "Online", "MATLAB Command Window", and "Desktop". The "Online" window shows a 3D surface plot. The "MATLAB Command Window" shows a command history with "clear" and "clc" commands. The "Desktop" window shows a file browser with files like "matrix.m" and "matrix.pdf". At the bottom of the browser window, there is a progress bar at 0.41 / 0.54, a "NEXT →" button, and other video controls.



Edit with WPS Office

This screenshot shows the MATLAB Onramp interface for task 2.1, titled "Entering Commands". The task involves understanding the assignment operator (=). It includes a text explanation, a task box with a hint, and a workspace window showing code execution.

Task 1
The equals sign (=) in MATLAB is the *assignment* operator, meaning that the expression on the right of the equals sign is assigned to the variable on the left.

When you enter `x = 3 + 4`, MATLAB first evaluates `3 + 4` and then assigns the result (7) to the variable `x`.

TASK
Enter the command `m = m + 1` to see what happens.

Hint | See Solution

Task 2
`>> 3*5`

`ans =`
15

Task 3
`>> m=3*5`

`m =`
15

Task 4
`>> m=m+1`

`m =`
16

Correct!
Space Continue Esc Try an alternative solution

WORKSPACE
Name
ans
m

This screenshot shows the MATLAB Onramp interface for task 2.3, titled "Saving and Loading Variables". The task involves saving and loading data files. It includes a text explanation, a task box with a hint, and a workspace window showing code execution and a data preview.

Task 1
`>> save datafile`

Task 2
`>> clear`

Task 3
`>> load datafile`

Task 4
`>> data`

Notice that the variable `data` is listed in the workspace. You can see contents of any variable by entering the name of the variable.
`>> myvar`

TASK
Display the contents of the variable `data`.

Hint | See Solution

Task 5

Further Practice

Correct!
Space Continue Esc Try an alternative solution

WORKSPACE
Name
data



Edit with WPS Office

The screenshot shows the MATLAB Onramp interface. The left sidebar lists 'Task 1', 'Task 2', and 'Task 3'. The main area displays the following code in the Command Window:

```
>> x=pi/2
x =
    1.5708
>> y=sin(x)
y =
    1
>> z=sqrt(-9)
z =
    0.0000 + 3.0000i
>> |
```

The workspace pane on the right shows variables `x`, `y`, and `z`. A note in the sidebar indicates that the solution contains the imaginary number `i`, which is a built-in constant in MATLAB.

The screenshot shows the MATLAB Onramp interface. The left sidebar lists 'Task 1' and 'Further Practice'. The main area displays the following code in the Live Editor:

```
1 r = 0.5
2 x = pi*r^2
```

The workspace pane on the right shows variables `r` and `x`. The live editor toolbar is visible at the top.



Edit with WPS Office

MATLAB Onramp (23% complete)

4.1 Manually Entering Arrays

Task 1

Background
A single number, called a *scalar*, is actually an array that contains 1 row and 1 column.

TASK
Create a variable named `x` with a value of 2.

Hint | See Solution | Reset

Task 2

Task 3

Task 4

Task 5

Task 6

Task 7

Further Practice

What's an Array?

All MATLAB variables are *arrays*. This means that each variable can contain multiple elements. You can use arrays to store related data in one variable.

Because you'll use arrays every time you program, it's important to get to know them and the terminology used to describe them.

The diagram shows a large circle labeled "ARRAY". Inside, a 2x2 matrix is labeled "matrix". To its right is a vertical column vector labeled "column vector" with elements 2, 3, 6, -9. Below the matrix is a scalar labeled "scalar" with value 2. To the right of the column vector is a horizontal row vector labeled "row vector" with elements 2, 3, 6, -9.

Task 4

MATLAB Onramp (23% complete)

4.1 Manually Entering Arrays

Task 1

Test Results: Correct!

- ✓ Does `x` have three elements?
- ✓ Is `x` a row vector?
- ✓ Does `x` have the correct values?

Task 2

Task 3

Task 4

Task 5

Creating Vectors

Instructions are in the task pane to the left. Complete and submit each task one at a time.

Task 1

Task 2

Task 3

COMMAND WINDOW



Edit with WPS Office

If you know the number of elements you want in a vector (instead of the spacing between each element), you could instead use the `linspace` function:

```
linspace(first, last, number_of_elements).
```

Note the use of commas (,) to separate inputs to the `linspace` function.

```
x = linspace(0, 1, 5)
x =
    0   0.250   0.500   0.750   1.000
```

TASK
Create a row vector named `x` that starts at `1`, ends at `10`, and contains `5` elements.

Test Results: Correct

- ✓ Does `x` have five elements?
- ✓ Does `x` have the correct values?

MAT Lab :

MATLAB is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

As of 2020, MATLAB has more than 4 million users worldwide. MATLAB users come from various backgrounds of engineering, science, and economics.

History

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s. He designed it to give his student's access to LINPACK and EISPACK without them having to learn Fortran. It soon spread to other universities and found a strong audience within the applied mathematics community. Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C and founded Math Works in 1984 to continue its development. These rewritten libraries were known as JACKPAC. In 2000,



Edit with WPS Office

MATLAB was rewritten to use a newer set of libraries for matrix manipulation, LAPACK.

MATLAB was first adopted by researchers and practitioners in control engineering, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra and numerical analysis, and is popular amongst scientists involved in image processing.

Syntax

The MATLAB application is built around the MATLAB programming language. Common usage of the MATLAB application involves using the "Command Window" as an interactive mathematical shell or executing text files containing MATLAB code.

Variables

Variables are defined using the assignment operator, =. MATLAB is a weakly typed programming language because types are implicitly converted. It is an inferred typed language because variables can be assigned without declaring their type, except if they are to be treated as symbolic objects, and that their type can change. Values can come from constants, from computation involving values of other variables, or from the output of a function. For example:

```
>> x = 17
```

```
x =
```

```
17
```

```
>> x = 'hat'
```

```
x =
```

```
hat
```

```
>> x = [3*4, pi/2]
```

```
x =
```

```
12.0000 1.5708
```

```
>> y = 3*sin(x)
```

```
y =
```

```
-1.6097 3.0000
```



Edit with WPS Office

Vectors and matrices

A simple array is defined using the colon syntax: *initial.increment.terminator*. For instance:

```
>> array = 1:2:9
```

```
array =
```

```
1 3 5 7 9
```

Defines a variable named array (or assigns a new value to an existing variable with the name array) which is an array consisting of the values 1, 3, 5, 7, and 9. That is, the array starts at 1 (the *initial* value), increments with each step from the previous value by 2 (the *increment* value), and stops once it reaches (or to avoid exceeding) 9 (the *terminator* value).

```
>> array = 1:3:9
```

```
array =
```

```
1 4 7
```

the *increment* value can actually be left out of this syntax (along with one of the colons), to use a default value of 1.

```
>> ari = 1:5
```

```
ari =
```

```
1 2 3 4 5
```

assigns to the variable named ari an array with the values 1, 2, 3, 4, and 5, since the default value of 1 is used as the increment.

Indexing is one-based, which is the usual convention for matrices in mathematics, unlike zero-based indexing commonly used in other programming languages such as C, C++, and Java.

Matrices can be defined by separating the elements of a row with blank space or comma and using a semicolon to terminate each row. The list of elements should be surrounded by square brackets []. Parentheses () are used to access elements and subarrays (they are also used to denote a function argument list).

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```



Edit with WPS Office

```
A =
```

```
16 3 2 13
```

```
5 10 11 8
```

```
9 6 7 12
```

```
4 15 14 1
```

```
>> A(2,3)
```

```
ans =
```

```
11
```

Sets of indices can be specified by expressions such as 2:4, which evaluates to [2, 3, 4]. For example, a submatrix taken from rows 2 through 4 and columns 3 through 4 can be written as:

```
>> A(2:4,3:4)
```

```
ans =
```

```
11 8
```

```
7 12
```

```
14 1
```

A square identity matrix of size n can be generated using the function eye, and matrices of any size with zeros or ones can be generated with the functions zeros and ones, respectively.

```
>> eye(3,3)
```

```
ans =
```

```
1 0 0
```

```
0 1 0
```

```
0 0 1
```

```
>> zeros(2,3)
```

```
ans =
```



Edit with WPS Office

```
0 0 0  
0 0 0  
  
>> ones(2,3)  
ans =  
1 1 1  
1 1 1
```

Transposing a vector or a matrix is done either by the function transpose or by adding dot-prime after the matrix (without the dot, prime will perform conjugate transpose for complex arrays):

```
>> A = [1 ; 2], B = A.', C = transpose(A)
```

```
A =
```

```
1
```

```
2
```

```
B =
```

```
1 2
```

```
C =
```

```
1 2
```

```
>> D = [0 3 ; 1 5], D.'
```

```
D =
```

```
0 3
```

```
1 5
```

```
ans =
```

```
0 1
```

```
3 5
```

Most functions accept arrays as input and operate element-wise on each element. For example, `mod(2*J,n)` will multiply every element in J by 2, and then reduce each element



Edit with WPS Office

modulo n . MATLAB does include standard for and while loops, but (as in other similar applications such as R), using the vectorized notation is encouraged and is often faster to execute. The following code, excerpted from the function *magic.m*, creates a magic square M for odd values of n (MATLAB function *meshgrid* is used here to generate square matrices I and J containing $1:n$).

```
[J,I] = meshgrid(1:n);
A = mod(I + J - (n + 3) / 2, n);
B = mod(I + 2 * J - 2, n);
M = n * A + B + 1;
```

Structures

MATLAB supports structure data types. Since all variables in MATLAB are arrays, a more adequate name is "structure array", where each element of the array has the same field names. In addition, MATLAB supports dynamic field names (field look-ups by name, field manipulations, etc.).

Functions

When creating a MATLAB function, the name of the file should match the name of the first function in the file. Valid function names begin with an alphabetic character, and can contain letters, numbers, or underscores. Variables and functions are case sensitive.

Function handles

MATLAB supports elements of lambda calculus by introducing function handles, or function references, which are implemented either in .m files or anonymous/nested functions.

Classes and object-oriented programming

MATLAB supports object-oriented programming including classes, inheritance, virtual dispatch, packages, pass-by-value semantics, and pass-by-reference semantics. However, the syntax and calling conventions are significantly different from other languages. MATLAB has value classes and reference classes, depending on whether the class has *handle* as a super-class (for reference classes) or not (for value classes).

Method call behavior is different between value and reference classes. For example, a call to a method

```
object.method();
```

can alter any member of *object* only if *object* is an instance of a reference class, otherwise value class methods must return a new instance if it needs to modify the



Edit with WPS Office

object.

An example of a simple class is provided below.

```
classdef Hello
```

```
methods
```

```
    function greet(obj)
```

```
        disp('Hello!')
```

```
    end
```

```
end
```

```
end
```

When put into a file named hello.m, this can be executed with the following commands:

```
>> x = Hello();
```

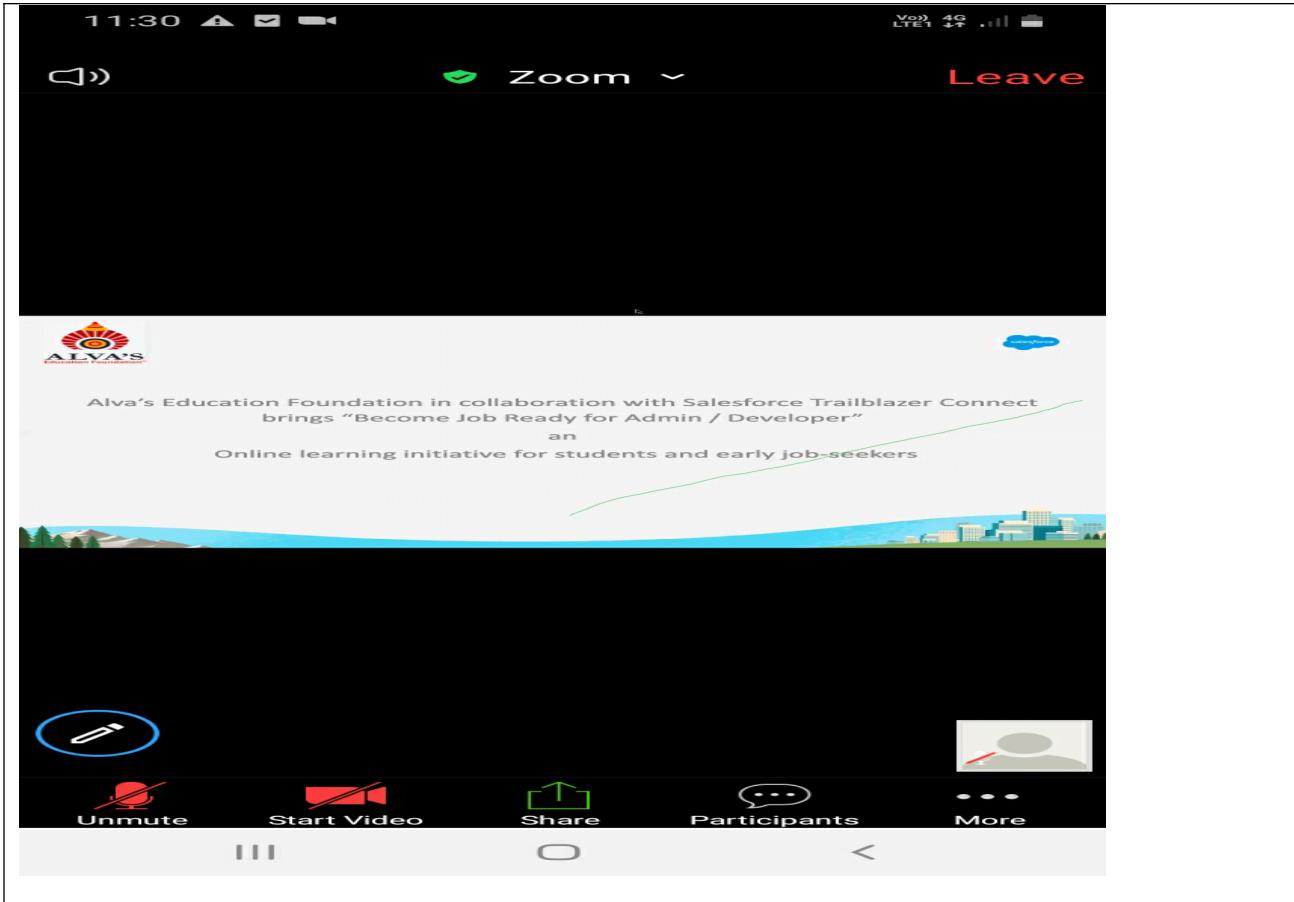
```
>> x.greet();
```

```
Hello!
```

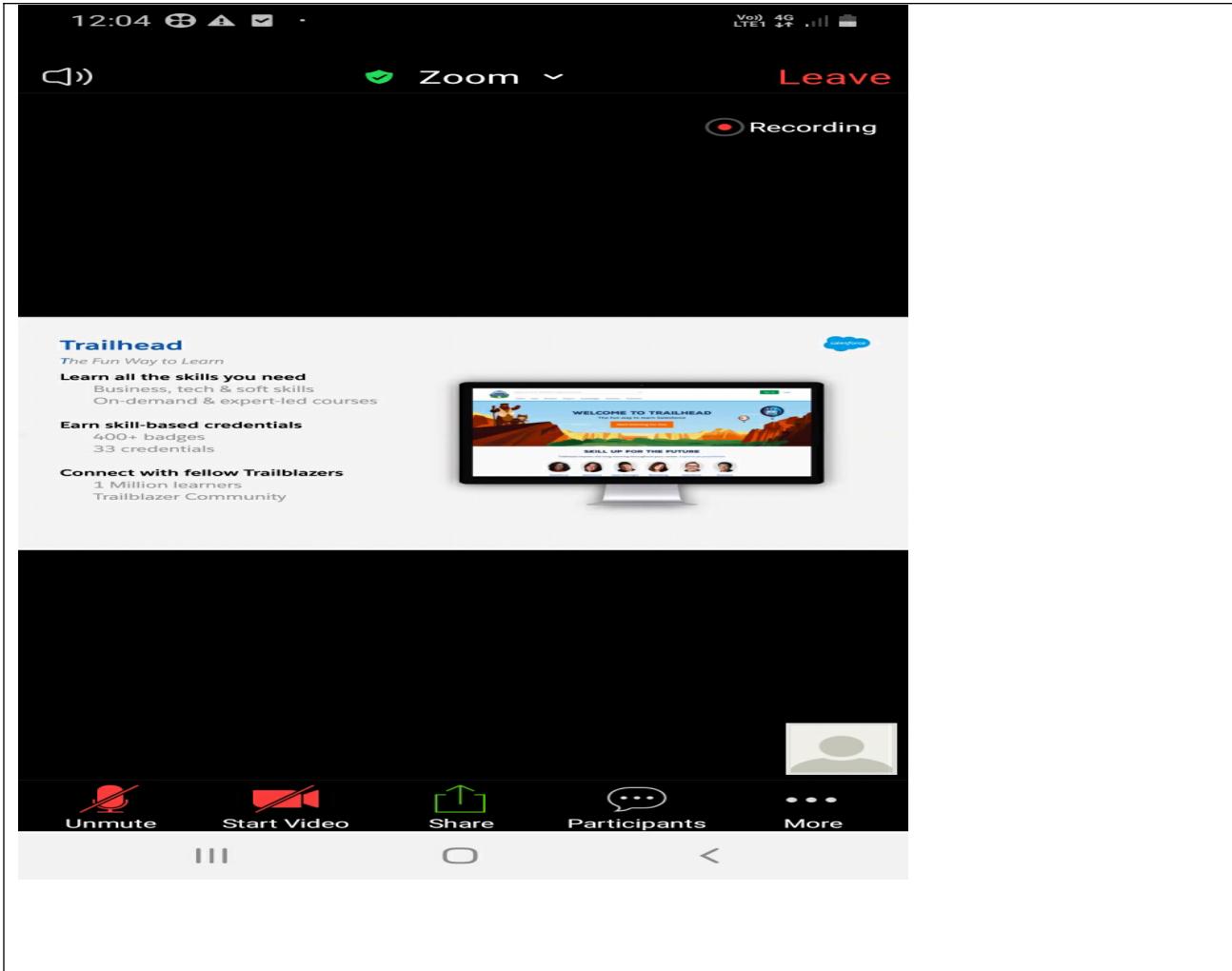
WEBINAR ON SALESFORCE – JOB READY PROGRAM READ ABT COMPANY ON MONDAY , 06 JULY 2020



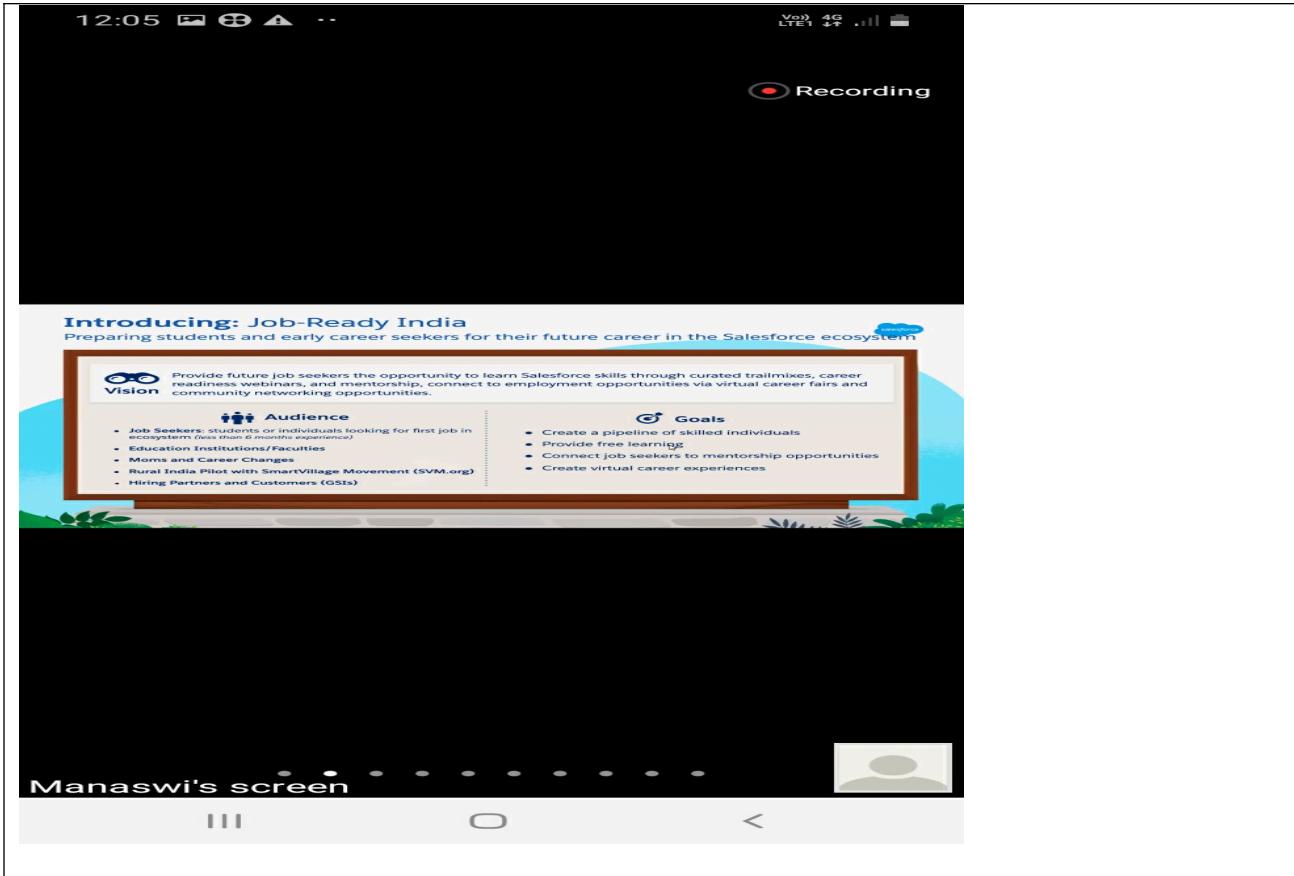
Edit with WPS Office



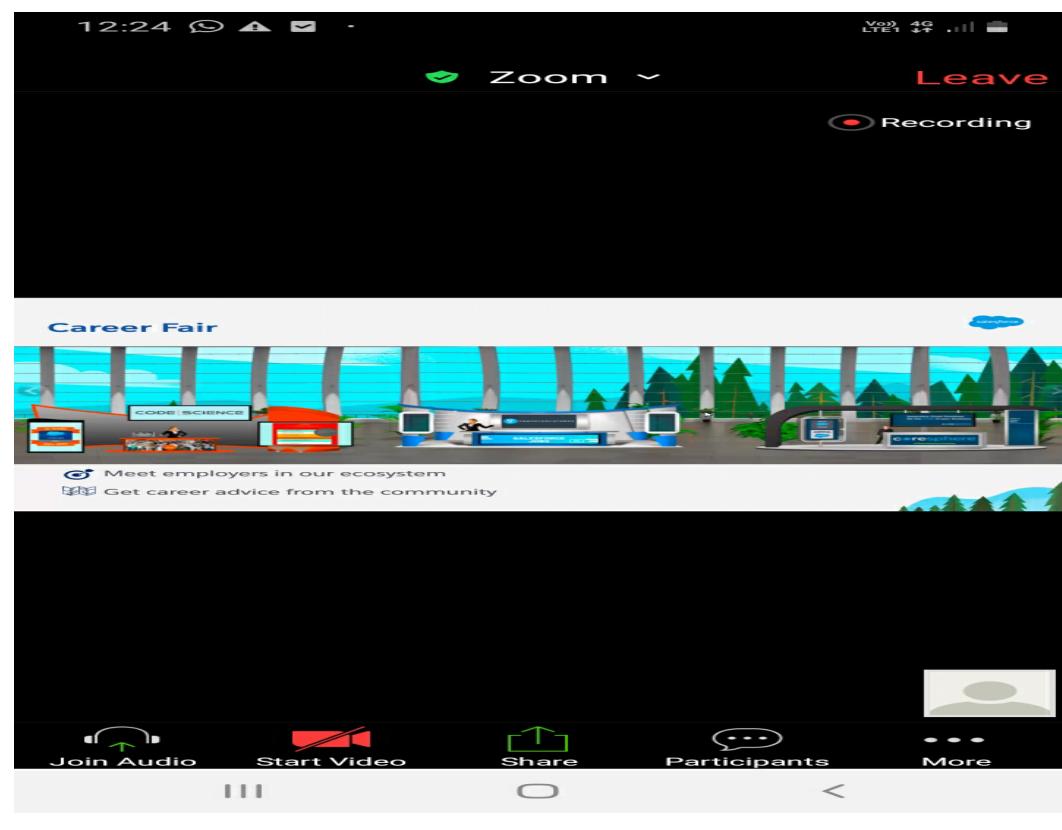
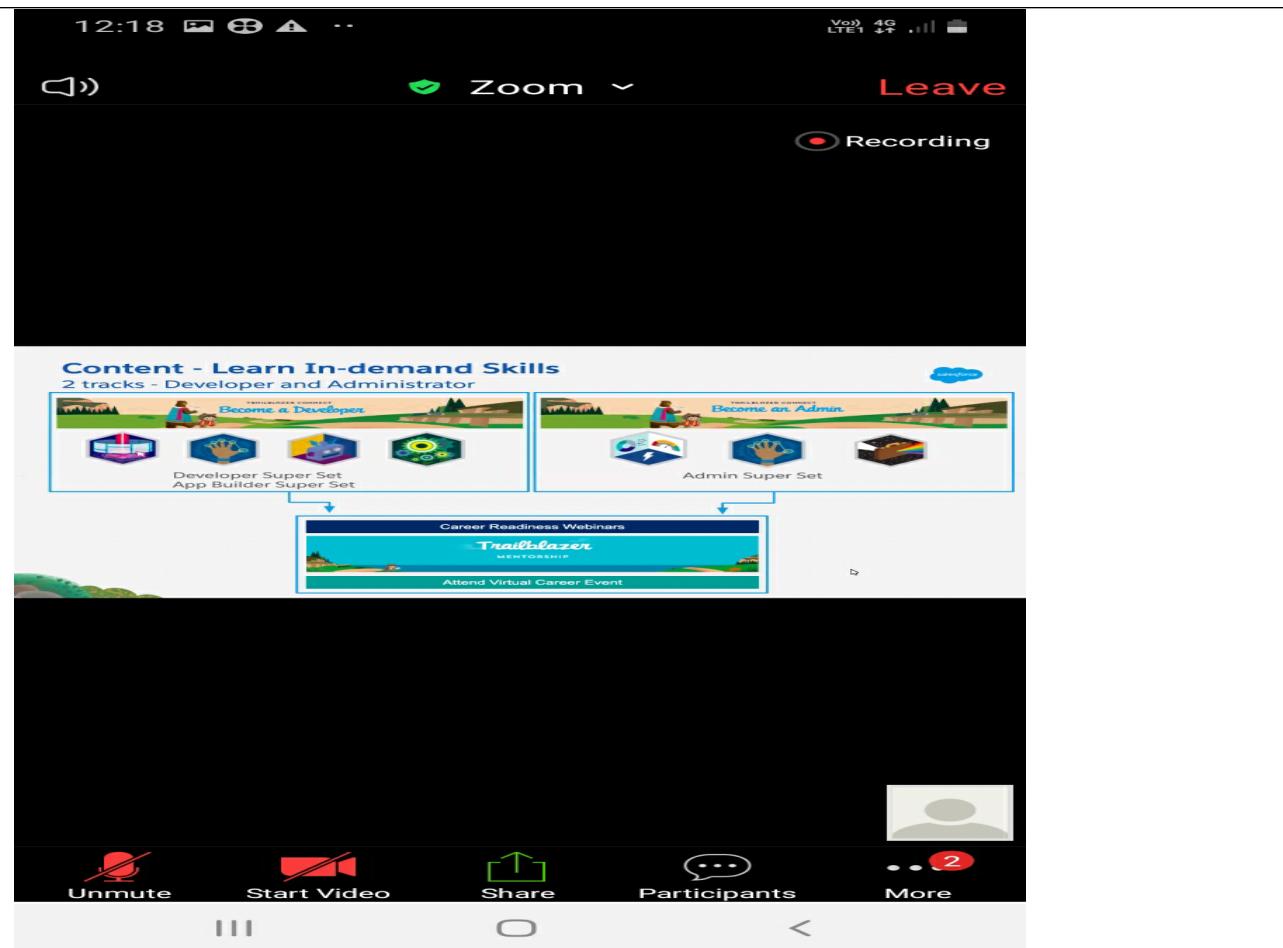
Edit with WPS Office



Edit with WPS Office



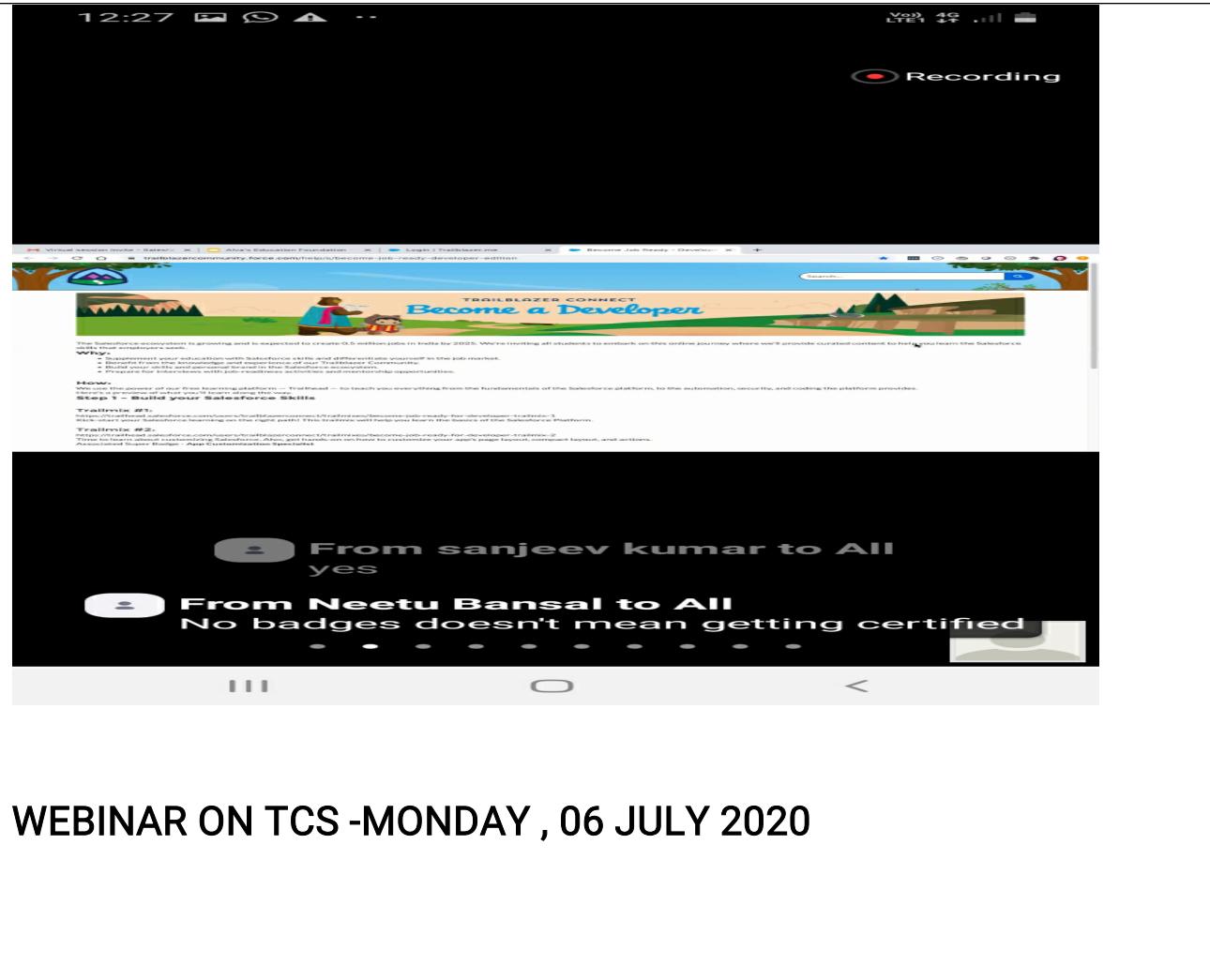
Edit with WPS Office



Edit with WPS Office



Edit with WPS Office



WEBINAR ON TCS -MONDAY , 06 JULY 2020



Edit with WPS Office



Need

1. To provide industry exposure to Students
2. To Bridge the gap between Industry – Student – Academia
3. To enable application of concepts in industrial scenario

Internship

Internship +

1. To ensure learning continuity during crisis time
2. To provide multiple options to the students to select the preferred Industry
3. To Provide exposure to latest trending topics
4. To enable industry exposure independent of location
5. To enable uniform process for award of credits (1-3-5 credits)
6. To enable program execution compliant to AICIE Guidelines

Remote Internship Opportunity

TCS iON



Edit with WPS Office



RIO - Introduction

TCS will collaborate with **Industry Leaders**, come out with trending topics for Internship

TCS will bring the **Technology levers** to enable students to complete the internship remotely

TCS will e2e program manage the RIO

Institutions to acknowledge, include RIO as part of their curriculum and **award credit points** to students on successful completion

Institutions to identify & **nominate guides** for the students

Students will **carry out the internship program of their choice**

Offer project **Internships** to Students

Industry will **identify and nominate mentors** to guide the students

Industry will issue a **certificate** to all the students who have successfully completed the Internship

TCS

Institution

Industry

TCS iON



Edit with WPS Office

2:57

VoIP 4G
LTE1

RIO - Streams / Domains

<ul style="list-style-type: none">- Electrical- Electronics- Instrumentation- Nanoelectronics <p>Electronics</p>	<ul style="list-style-type: none">- Computer Science- Information technology- Information and communication technology <p>Information</p>	<ul style="list-style-type: none">- Civil- Transportation- Architecture <p>Civil</p>
<ul style="list-style-type: none">- Mechanical- Mechatronics- Aerospace- Aeronautical- Production Engineering <p>Mechanical</p>	<ul style="list-style-type: none">- Chemical- Pharmaceutical <p>Chemical</p>	<ul style="list-style-type: none">- Biomedical- Bioinformatics- Biotechnology- Medical Nanotechnology <p>Biological</p>
<p>Environmental Public Health Affordable Technologies Rehabilitation Geriatric & Pediatric care Agriculture Education Disaster Management</p>		



TCS iON



Edit with WPS Office



Edit with WPS Office