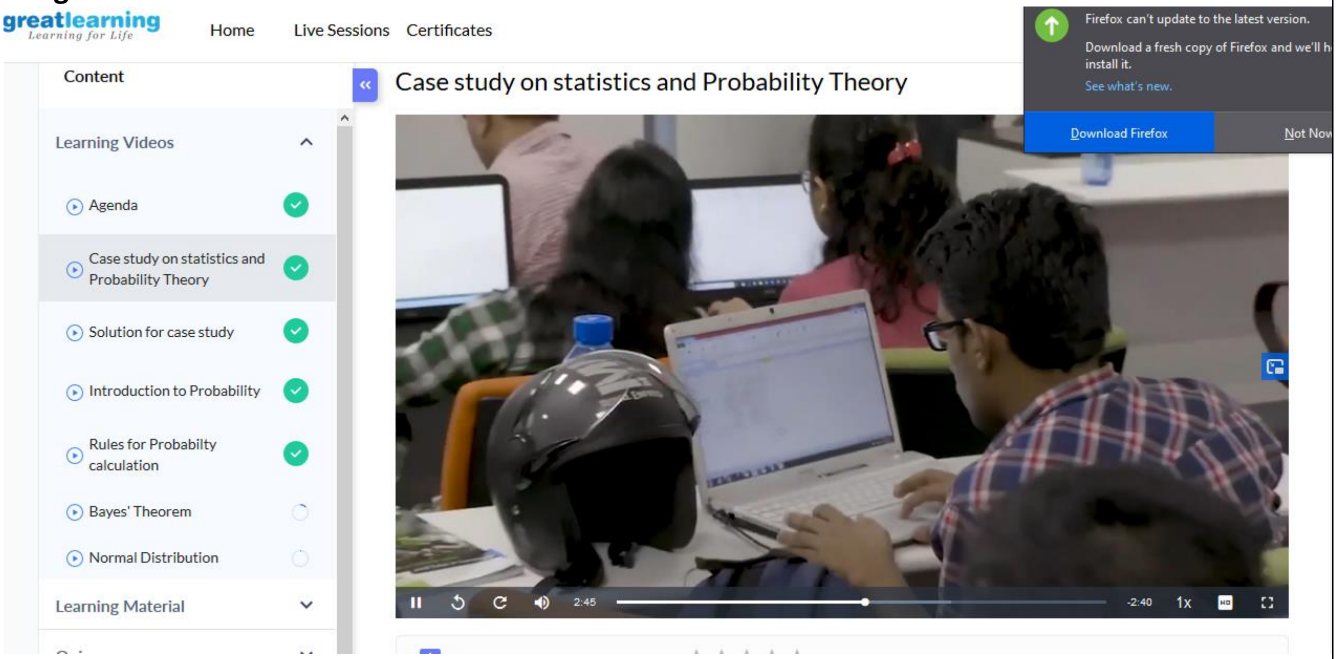
DAILY ASSESSMENT FORMAT

Date:	15/June/2020	Name:	nishanth
Course:	Statistical Learning	USN:	4al17ec063
Topic:	1.case study on statistics and probability theory 2.Solution for case study	Semester & Section:	6th b
GitHub Repository:	nishanthvr		

FORENOON SESSION DETAILS

Image of session



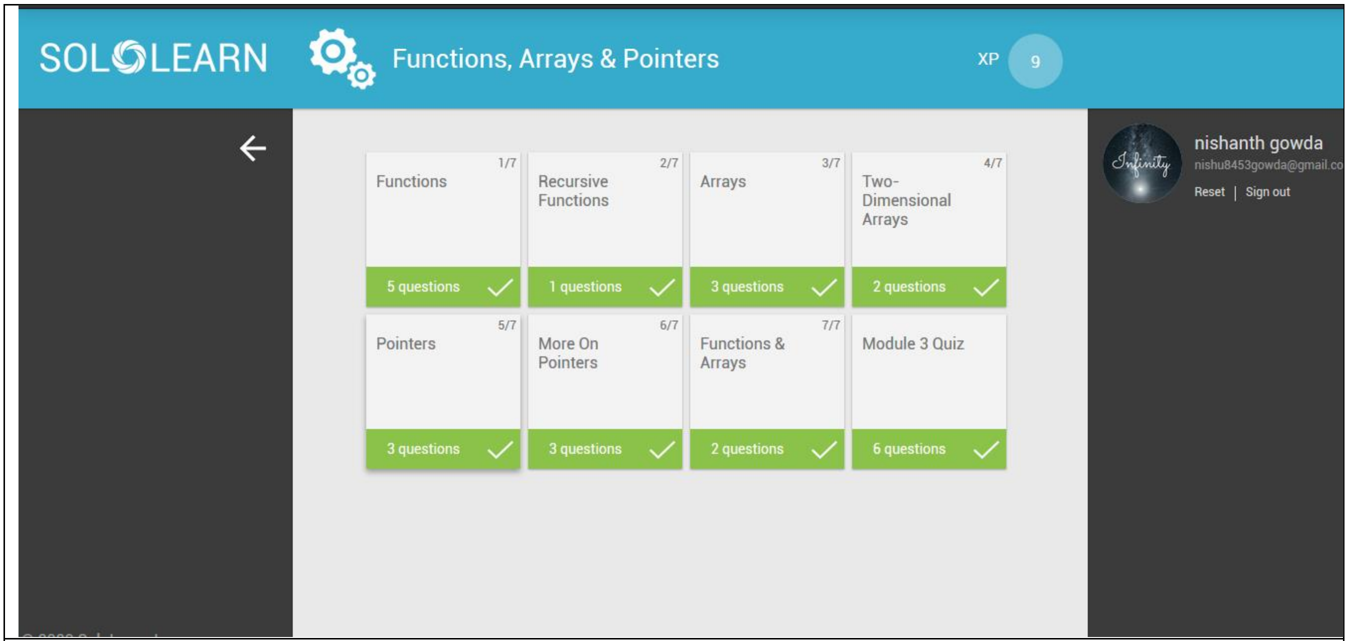
Case study on statistics and Probability Theory

Firefox can't update to the latest version.

Download a fresh copy of Firefox and we'll h install it.

Date:	15/June/2020	Name:	nishanth
Course:	C programming	USN:	4al17ec063
Topic:	1.basic concepts 2.function,pointer,string	Semester&Section :	6th b
Git hub repository	nishanthvr		

AFTERNOON SESSION DETAILS


SOLOLEARN
Functions, Arrays & Pointers
XP 9
nishanth gowda
nishu8453gowda@gmail.co
Reset | Sign out
Functions 1/7 5 questions
Recursive Functions 2/7 1 questions
Arrays 3/7 3 questions
Two-Dimensional Arrays 4/7 2 questions
Pointers 5/7 3 questions
More On Pointers 6/7 3 questions
Functions & Arrays 7/7 2 questions
Module 3 Quiz 6 questions

v **The switch Statement:** The switch statement branches program control by matching the result of an expression with a constant case value.

The switch statement often provides a more elegant solution to if-else if and nested if statements. The switch takes the form: switch (expression) { case val1: statements break; case val2: statements break; default: statements } For example, the following program outputs "Three": The while Loop The while statement is called a loop structure because it executes statements repeatedly while an expression is true, looping over and over again. It takes the form: while (expression) { statements } The expression evaluates to either true or false, and statements can be a single statement or, more commonly, a code block enclosed by curly braces { }. For example:

```
#include <stdio.h>

int main() { int count = 1;

while (count < 8) { printf("Count = %d\n", count); count++; }

return 0; }Try It Yourself
```

The code above will output the count variable 7 times.

Functions in C

Functions are central to C programming and are used to accomplish a program solution as a series of subtasks. By now you know that every C program contains a main() function. And you're familiar with the printf() function.

You can also create your own functions. A function: • is a block of code that performs a specific task • is reusable • makes a program easier to test • can be modified without changing the calling program

Even a simple program is easier to understand when main() is broken down into subtasks that are implemented with functions. For example, it's clear that the goal of this program is to calculate the square of a number: int main() { int x, result;

x = 5; result = square(x); printf("%d squared is %d\n", x, result);

return 0; } In order to use the square function, we need to declare it. Declarations usually appear above the main() function and take the form: return_type