Dayananda Sagar University
School of Engineering
Bangalore – 560068, Karnataka, India

Report on

Value-added course: "Functional Programming 101 with Haskell"

Target audience 5th Semester students of Computer science and Technology

The Department of Computer Science & Technology has organized a 30 Hours Value-Added Course on "Functional Programming 101 with Haskell" from 7nth to 17nth August 2023 in Hybrid mode. Students successfully completed the value-added course.

Resource Person: Mr. Shivaraj, Product Engineer, Juspay.

Students are able to understand the

- ✓ Basic Concepts of Haskell
- ✓ Functional Programming with Haskell,
- ✓ Implement Small Scale Functional Programs in Elementary Haskell,
- ✓ Way of thinking about Programming
- ✓ Nomads, I/O
- ✓ Concepts like Data Types, Recursions and Lambda Functions.

A total of **49 students** registered for this course and completed the course successfully.

We have received a very good response from the students.

Prof. Baskar Venugopalan, Professor of Practice and Prof.M.Chithambarathanu, Assistant Professor CST have organized this course.

Brochure:

DAYANANDA SAGAR UNIVERSITY
School of Engineering
Department of Computer Science & Technology

Cordially invites you to

Value-Added Course On

"Functional Programming 101 with Haskell"

7-8-2023 To 17-8-2023

Target Audience: 5th Sem CST Students

Hybrid Mode

Objectives of the Course:

- Functional programming with Haskell
  (Leader in #GitHub repos on Functional Programming)
- Haskell basics
- Implement small-scale functional programs in elementary Haskell
- A new way of thinking about programming (Functional)
- Learn concepts like data types, Recursion, Lambda functions and more

Outcomes of the Course:

- *The basic concepts of functional programming, such as*
  - *recursion*
  - *higher-order functions*
  - *immutable data structures*
- *The student can skilled to write simple functional programs*

Resource Person
Mr. Shivaraj B H
Product Engineer at Juspay

Registration link : https://forms.gle/gvX7ZZYyENFpnrM78

Online Meet Link : https://meet.google.com/xgt-awzx-bkt

Organizer:
Prof. Baskar Venugopalan
Professor Of Practice, CST
Prof. Chithambarathanu
Assistant Professor, CST

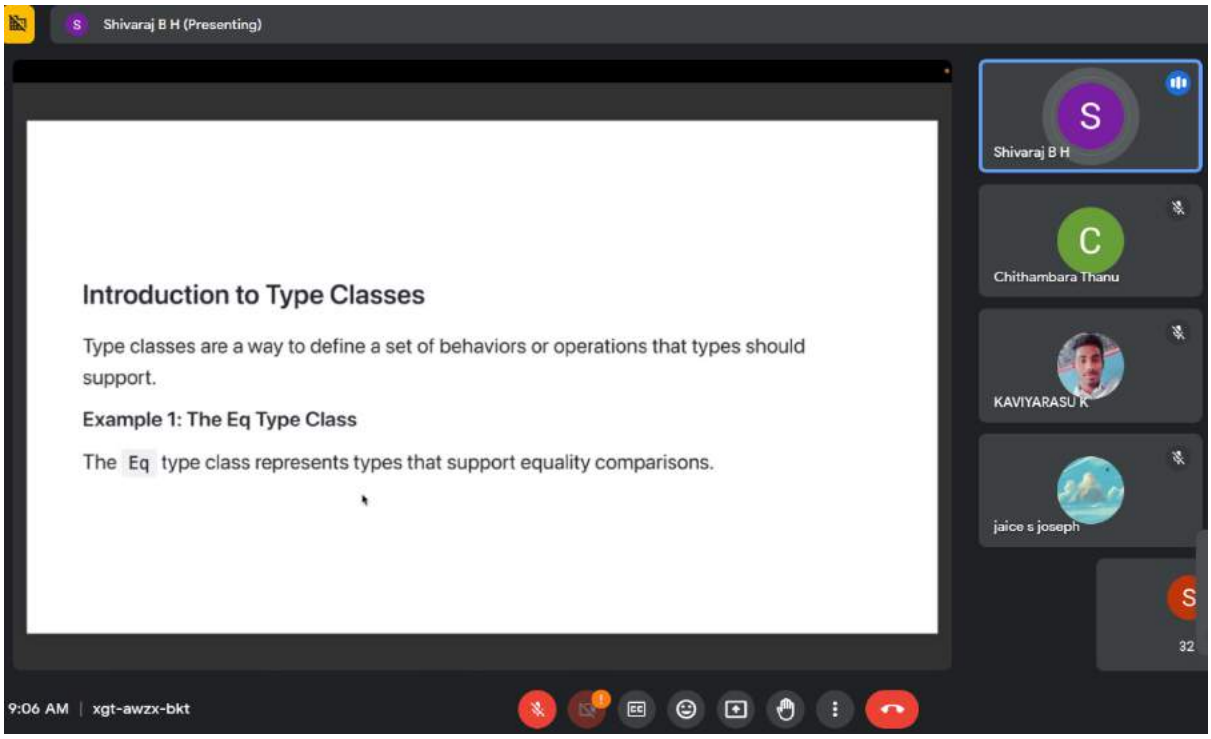
Chairperson:
Dr. M Shahina Parveen
Professor, CST

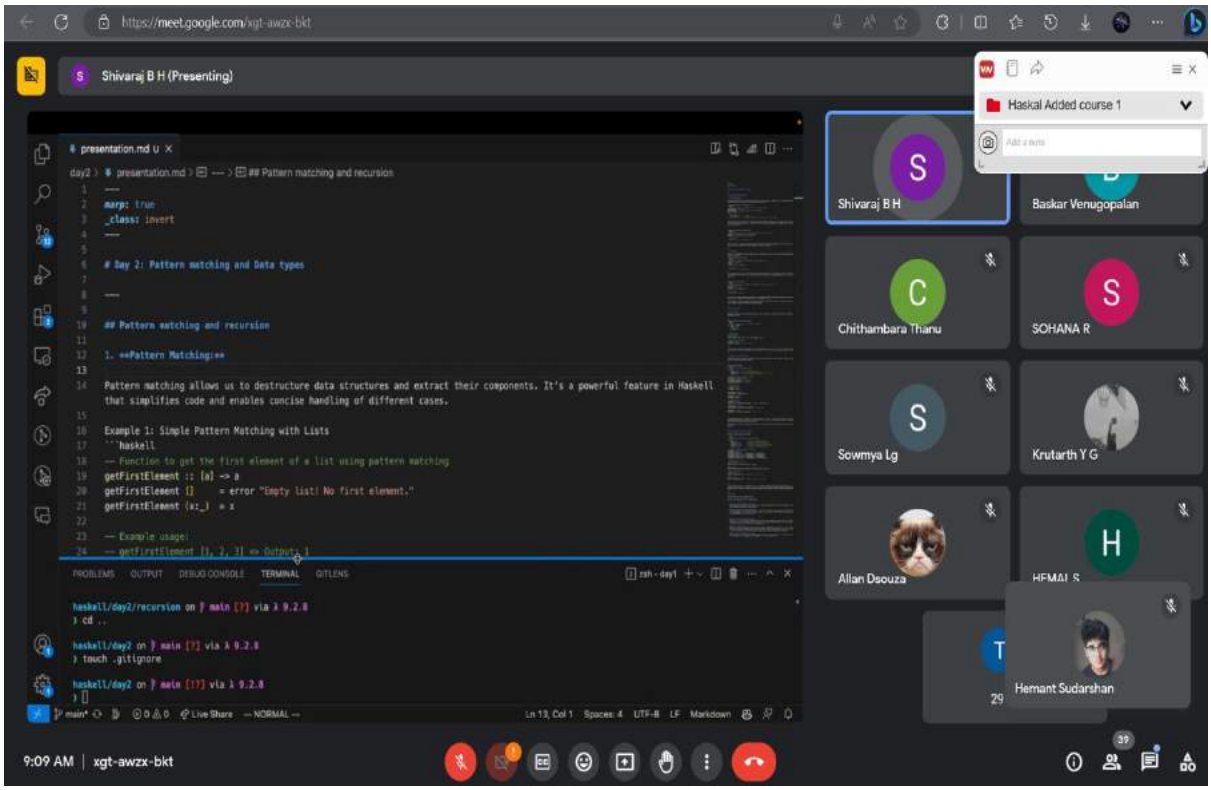
Convener:
Dr. Uday Kumar Reddy K R
Dean-School of Engineering
Dr.M Shahina Parveen
Professor, CST

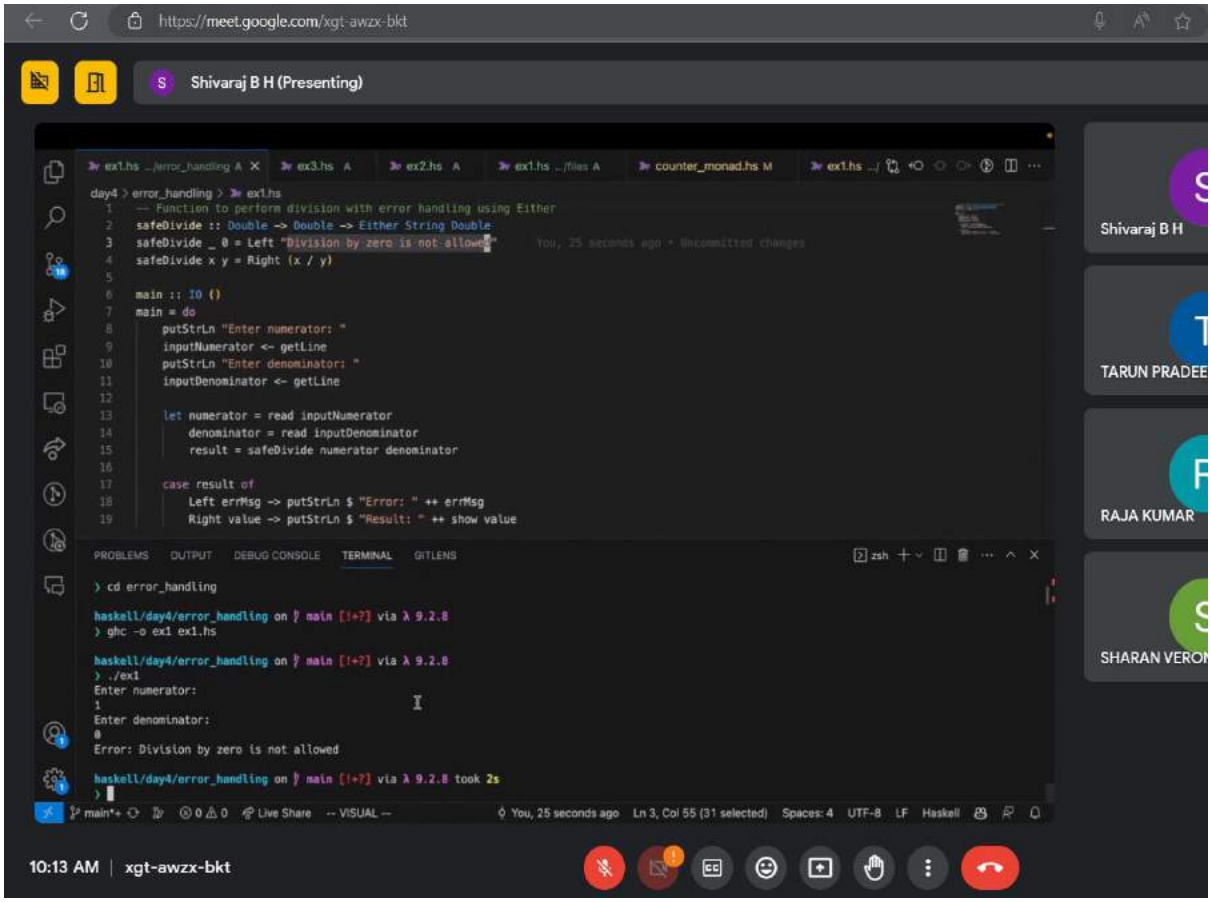
SCHEDULE FOR THE VALUE-ADDED COURSE

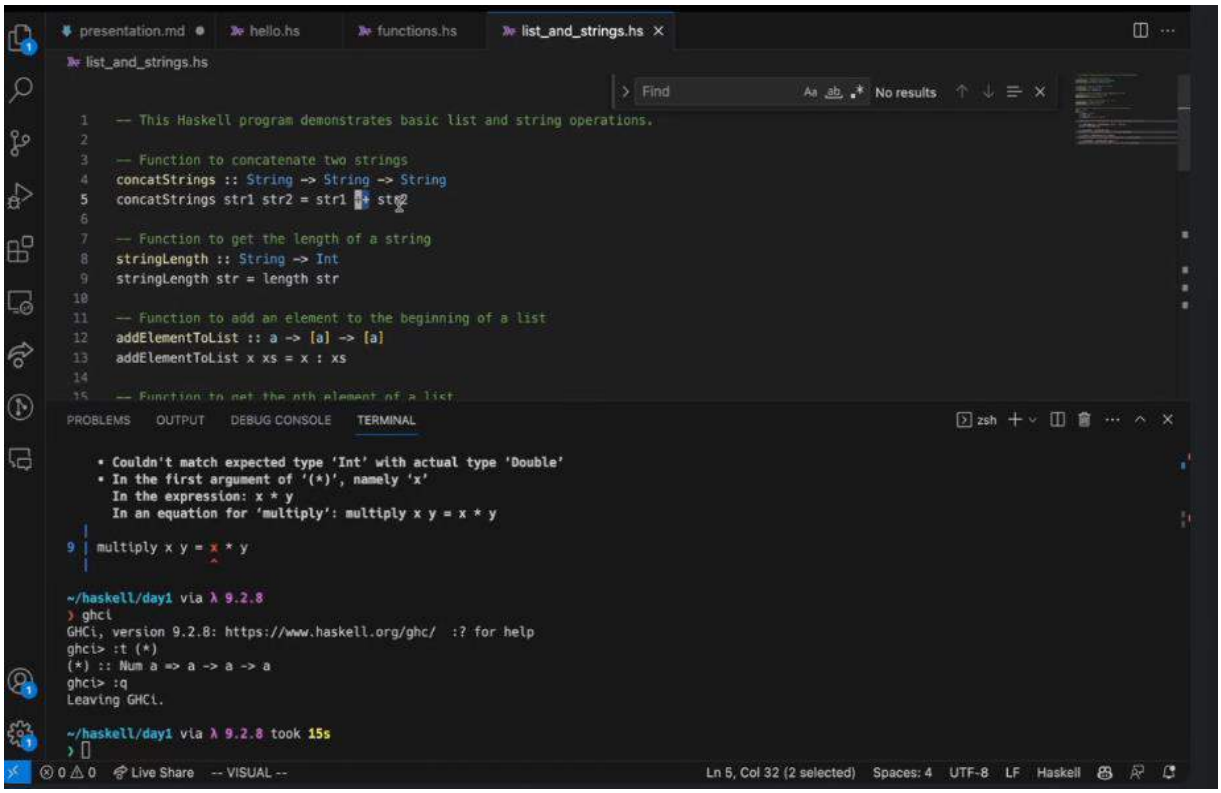
Sl No.	DATE	CONTENTS	ASSIGNMENT
1.	7/08/23	Introduction to Haskell: Key features and basic syntax. Installing Haskell: Setting up Haskell on your machine.	(EOD)
2.	8/08/23	Lists and Tuples: Understanding lists and tuples and their operations. Pattern Matching:Using pattern matching to deconstruct data structures.	(EOD)
3.	9/08/23	Recursion: Solving problems using recursive functions. Higher-Order Functions: Functions that take other functions as arguments or return functions.	(EOD)
4.	10/08/23	Type System: Understanding Haskell's type system and type declarations. Type Inference: How Haskell infers types without explicit annotations.	(EOD)
5.	11/08/23	IO and Side Effects: Introduction to IO monad and handling side effects in Haskell. File Handling: Reading and writing files in Haskell.	(EOD)
6.	12/08/23	Functors, Applicatives, and Monads: Understanding functor, applicative, and monadic operations.	(EOD)
7.	17/08/23	Final Project: Working on a small project or exercise that combines the concepts learned throughout the previous days.	(EOD)

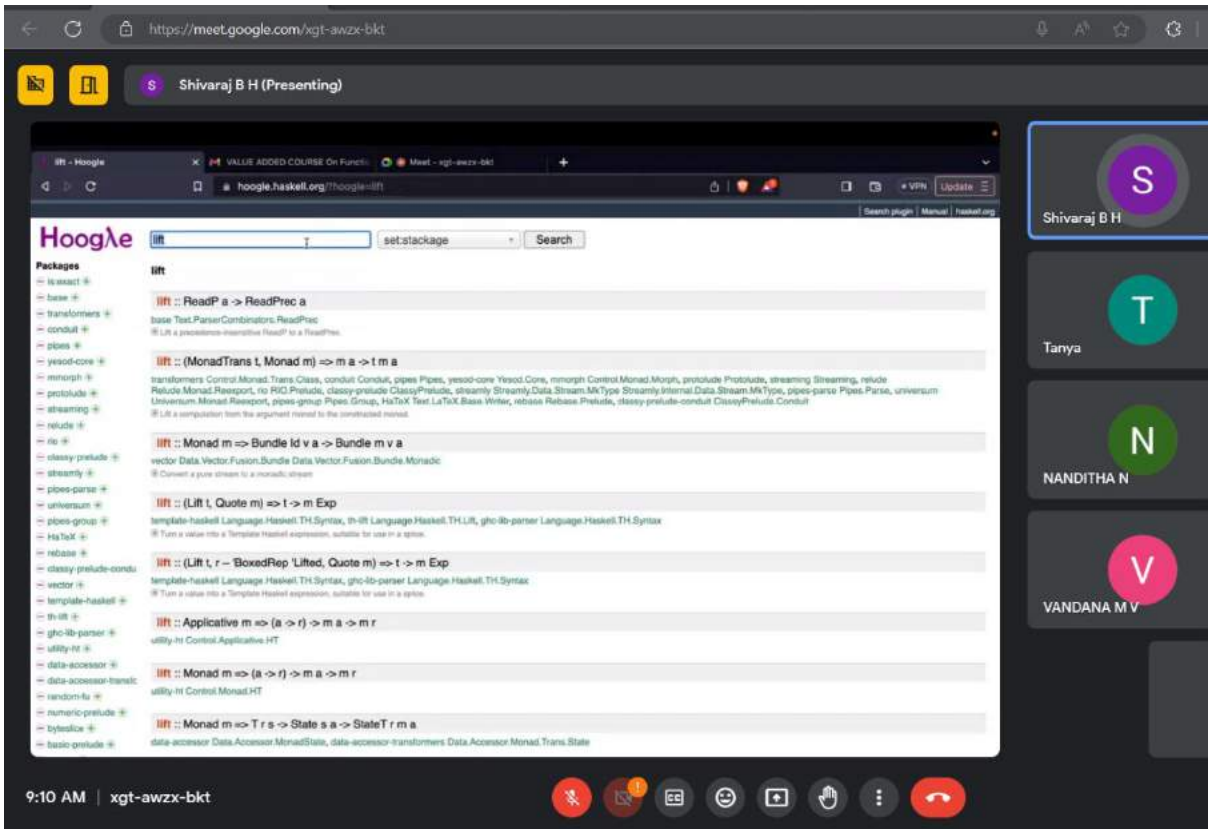
Screenshot of online Value-added class presentation:





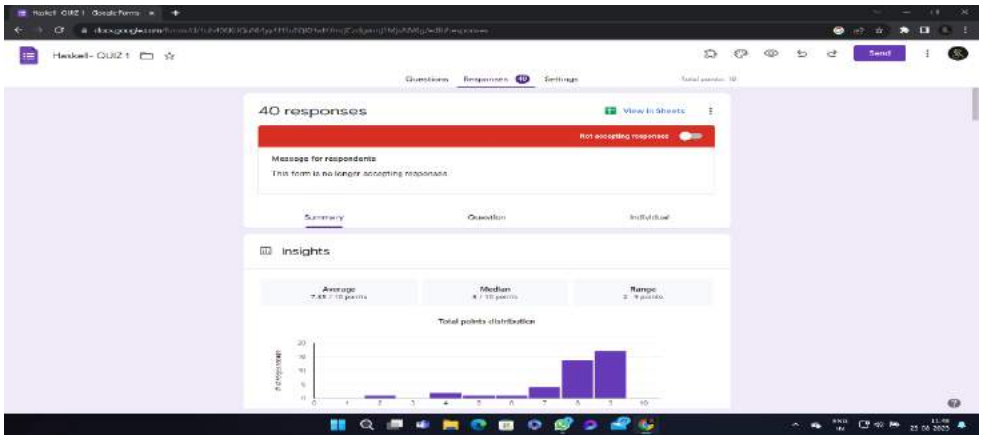

https://meet.google.com/xgt-awzx-bkt
Shivaraj B H (Presenting)
ex1.hs .../error_handling A
ex3.hs A
ex2.hs A
ex1.hs .../files A
counter_monad.hs M
ex1.hs
day4 > error_handling > ex1.hs
-- Function to perform division with error handling using Either
safeDivide :: Double -> Double -> Either String Double
safeDivide _ 0 = Left "Division by zero is not allowed"
You, 25 seconds ago • Uncommitted changes
safeDivide x y = Right (x / y)
main :: IO ()
main = do
putStrLn "Enter numerator: "
inputNumerator <- getLine
putStrLn "Enter denominator: "
inputDenominator <- getLine
let numerator = read inputNumerator
denominator = read inputDenominator
result = safeDivide numerator denominator
case result of
Left errMsg -> putStrLn $ "Error: " ++ errMsg
Right value -> putStrLn $ "Result: " ++ show value
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
GITLENS
zsh
❯ cd error_handling
haskell/day4/error_handling on ᚠ main [!+?] via λ 9.2.8
❯ ghc -o ex1 ex1.hs
haskell/day4/error_handling on ᚠ main [!+?] via λ 9.2.8
❯ ./ex1
Enter numerator:
1
Enter denominator:
0
Error: Division by zero is not allowed
haskell/day4/error_handling on ᚠ main [!+?] via λ 9.2.8 took 2s
❯
main*+
0 0
Live Share
-- VISUAL --
You, 25 seconds ago
Ln 3, Col 55 (31 selected)
Spaces: 4
UTF-8
LF
Haskell
Shivaraj B H
TARUN PRADEE
RAJA KUMAR
SHARAN VERO
10:13 AM | xgt-awzx-bkt


presentation.md
hello.hs
functions.hs
list_and_strings.hs
list_and_strings.hs
Find
No results
-- This Haskell program demonstrates basic list and string operations.
-- Function to concatenate two strings
concatStrings :: String -> String -> String
concatStrings str1 str2 = str1 ++ str2
-- Function to get the length of a string
stringLength :: String -> Int
stringLength str = length str
-- Function to add an element to the beginning of a list
addElementToList :: a -> [a] -> [a]
addElementToList x xs = x : xs
-- Function to get the nth element of a list
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
zsh
• Couldn't match expected type 'Int' with actual type 'Double'
• In the first argument of '(*)', namely 'x'
In the expression: x * y
In an equation for 'multiply': multiply x y = x * y
9 | multiply x y = x * y
~/haskell/day1 via λ 9.2.8
❯ ghci
GHCi, version 9.2.8: https://www.haskell.org/ghc/ :? for help
ghci> :t (*)
(*) :: Num a => a -> a -> a
ghci> :q
Leaving GHCi.
~/haskell/day1 via λ 9.2.8 took 15s
❯
0 0
Live Share
-- VISUAL --
Ln 5, Col 32 (2 selected)
Spaces: 4
UTF-8
LF
Haskell

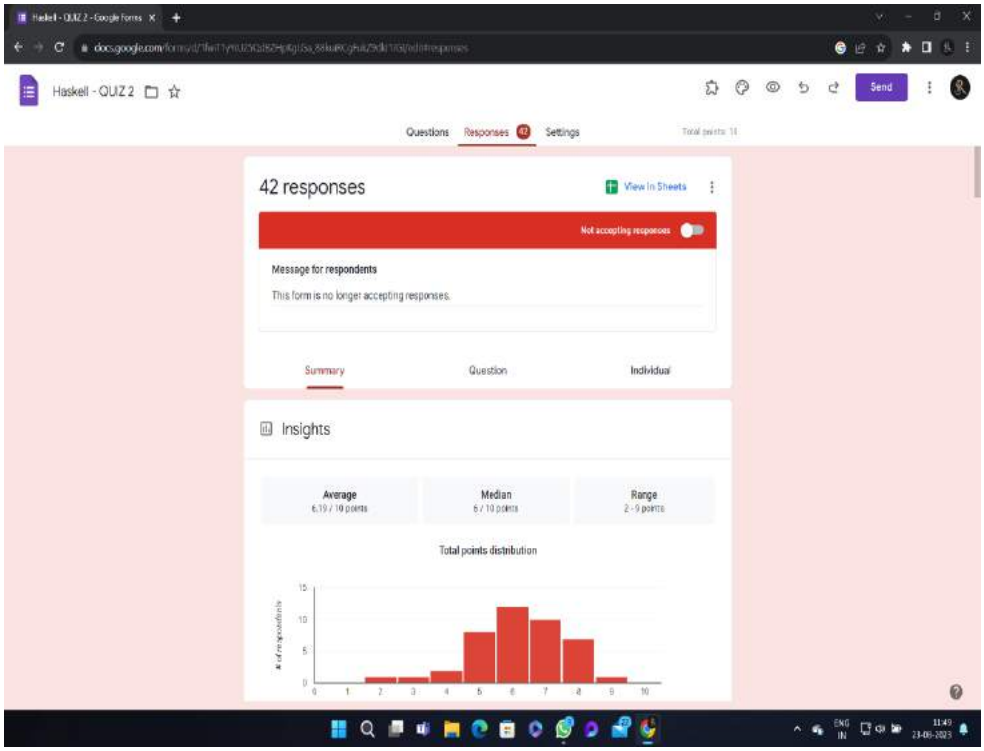


Assessment:

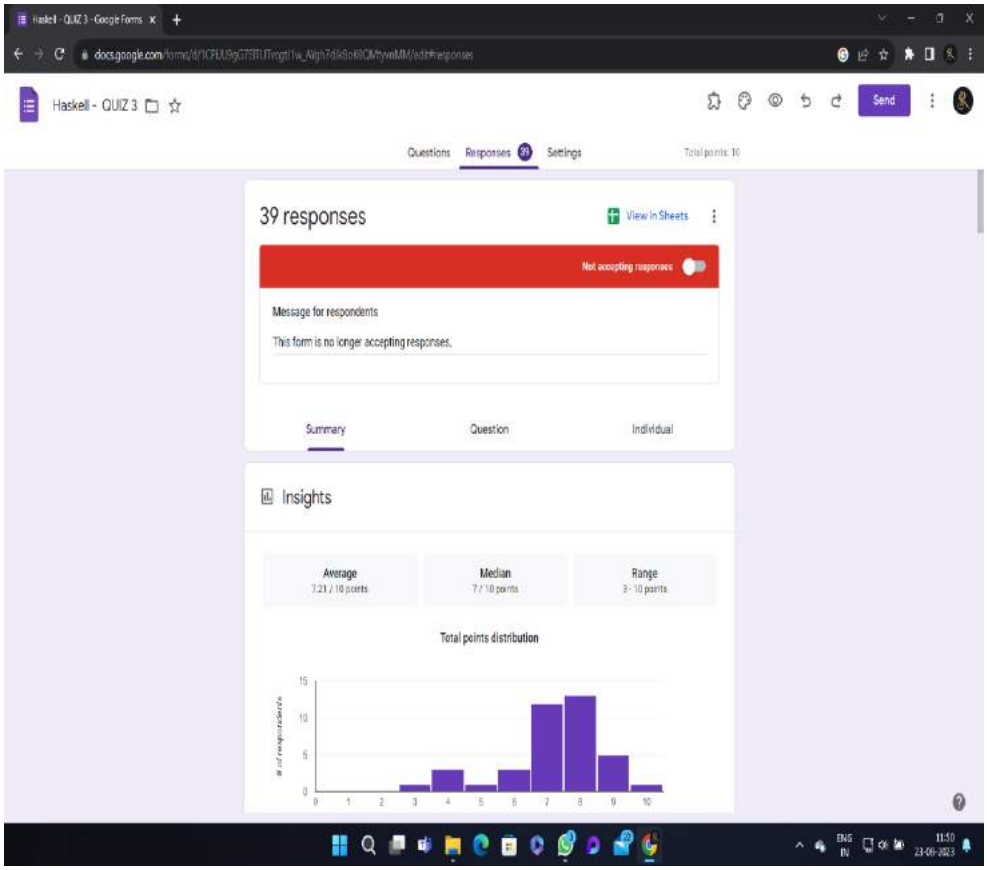
The assessment for the value-added course on "**Functional Programming 101 with Haskell**" was conducted through daily quizzes using Google forms at the end of each day. These quizzes were designed to gauge the students' comprehension of the material covered. The responses received from the quizzes indicated a satisfactory level of performance among the students.



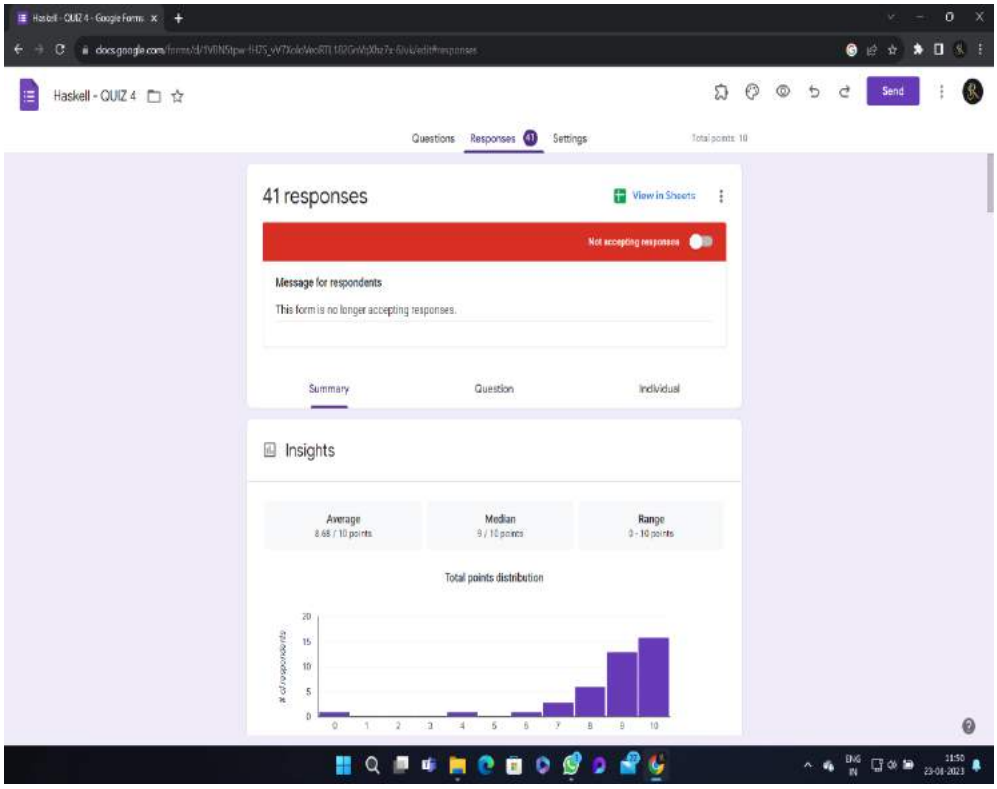
Day 1 – Quiz & Assignment



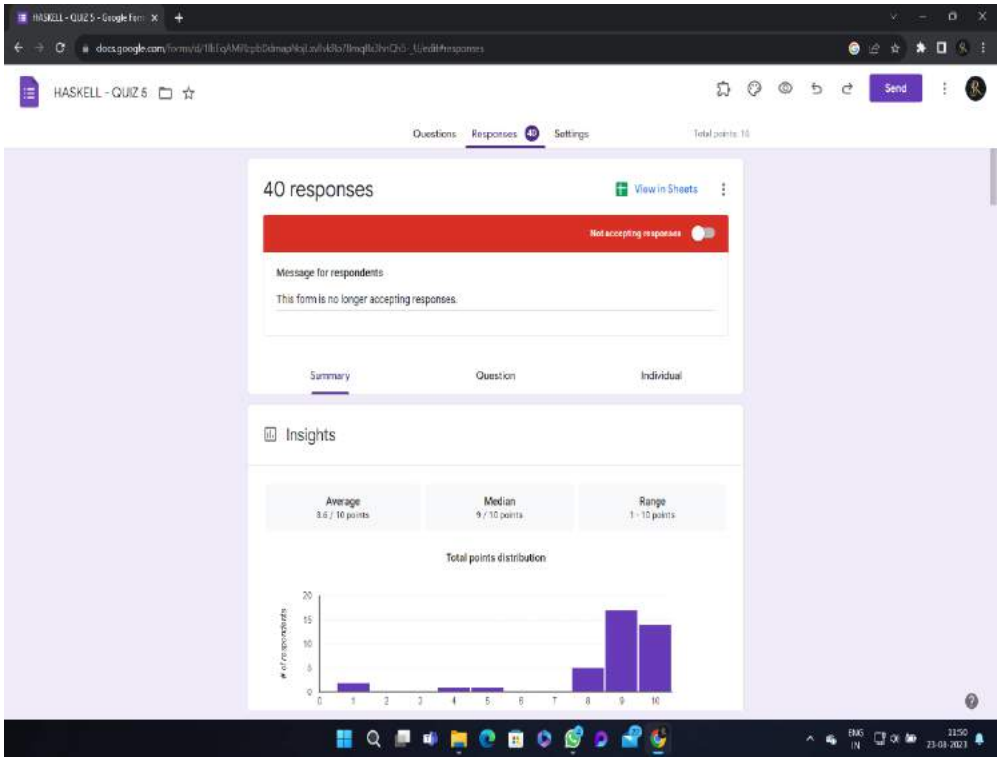
Day 2 - Quiz & Assignment



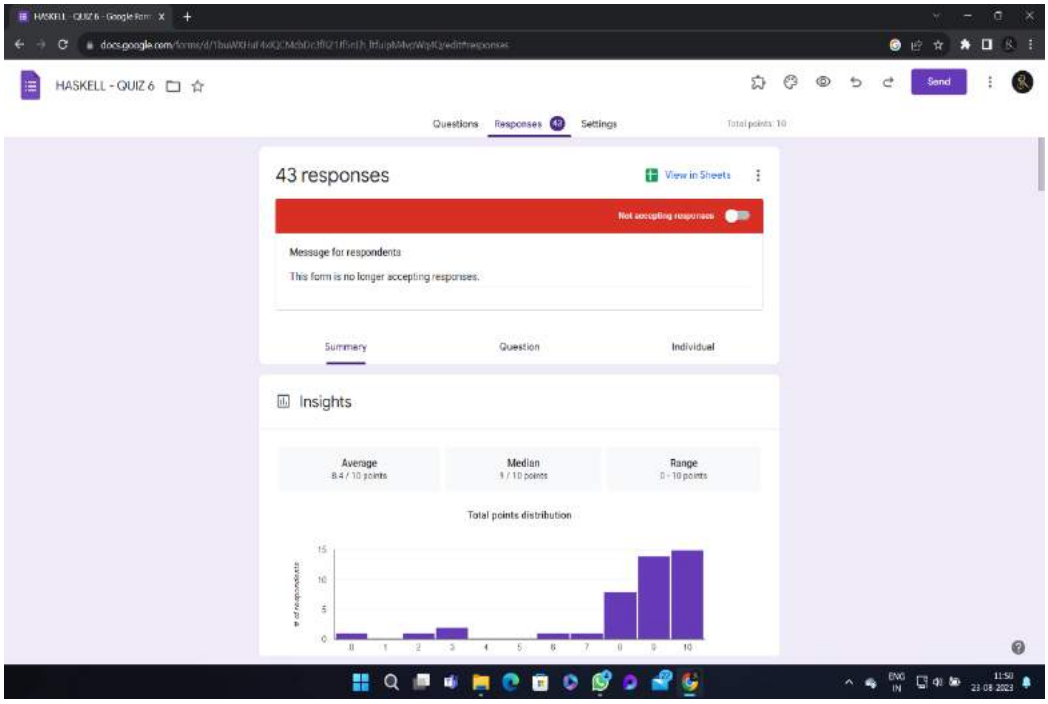
Day 3 - Quiz & Assignment



Day 4 - Quiz & Assignment



Day 5 - Quiz & Assignment

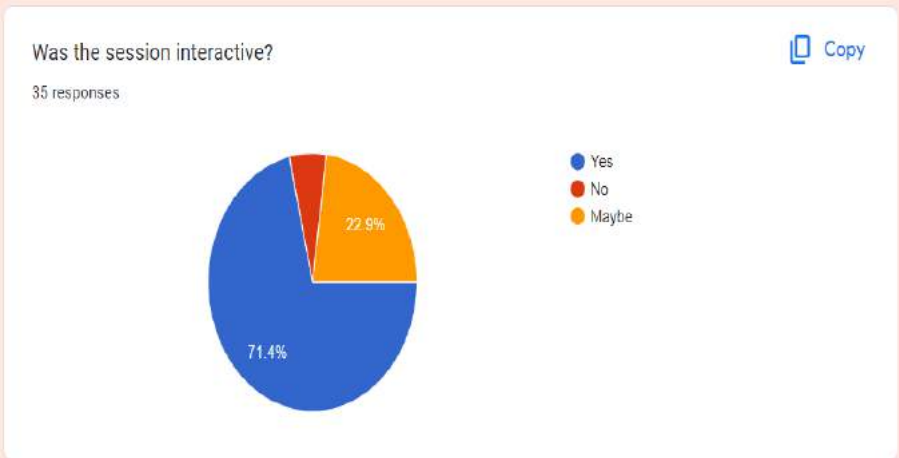


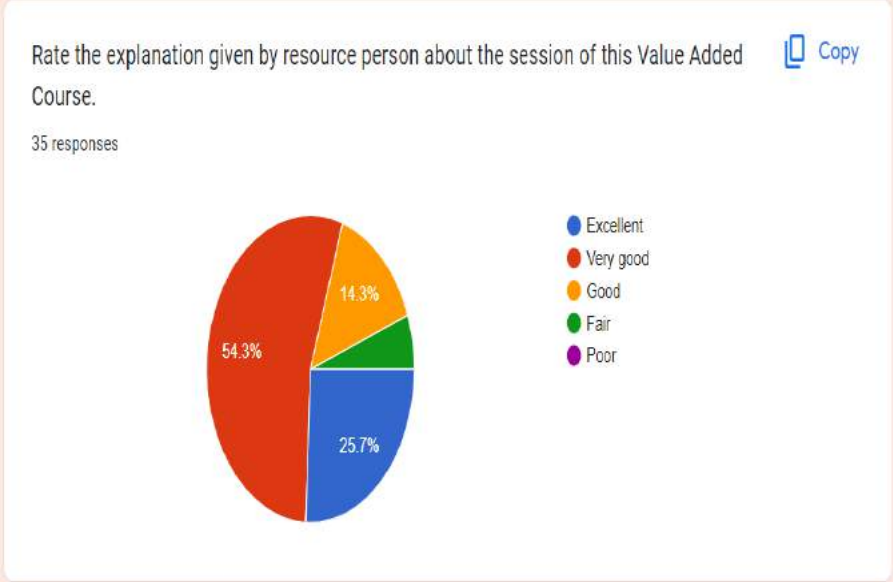
Day 6 - Quiz & Assignment

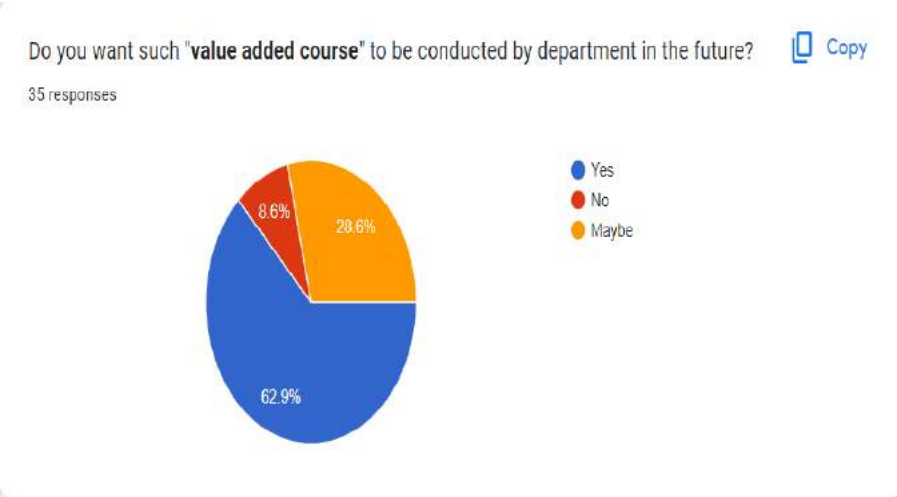
On 17nth August 2023, every participants have worked on mini project leveraging the knowledge gained during this value added course and submitted the same in the GitHub repo assigned. The same used to access and decide on next course of action around functional programming.

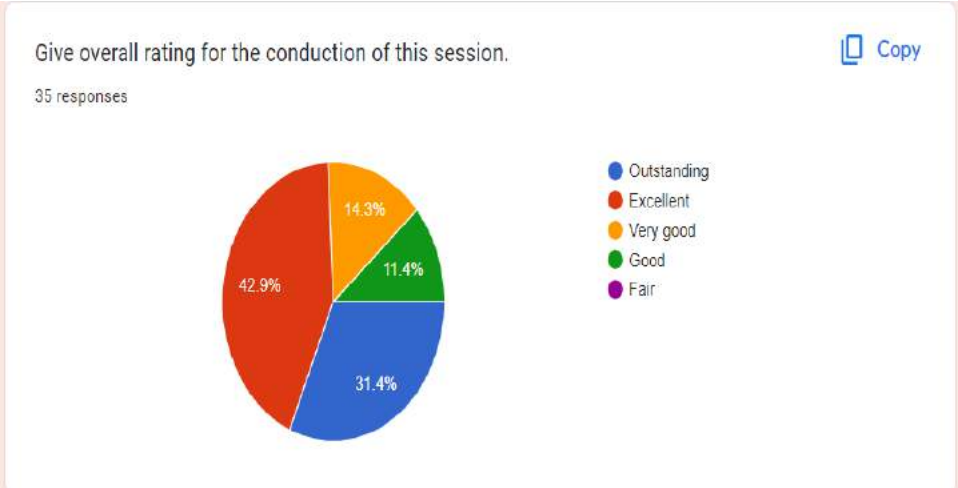
Feedback:

Feedback for the value-added course on "**Functional Programming 101 with Haskell**" was gathered from students using Google Forms. Students were encouraged to provide their insights on various aspects of the course, including content, overall satisfaction, effectiveness of the sessions in achieving its stated objectives and presentation by resource person. Overall, the feedback reflects that the objectives of the course were met, and participants gained valuable insights into the functional programming of haskell.









Conclusion:

In summary, the value-added course on "**Functional Programming 101 with Haskell**" provided students with an opportunity to learn the fundamental programming such as Recursions, Higher order functions and immutable data structures. The students can skill to write simple functional programs. Students who attended this value-added course have been tasked to complete a mini project (working POC) and upload the same into the GitHub repo for evaluation in deciding the next course of action around functional programming skills. This skill building exercise will sure help our students for their career betterment.