COMPLIANCE REPORT

for

Online C Program Evaluation

Prepared by

Swarup Padhi (20CS30062)

Roopak Priydarshi(20CS30042)

Shah Dhruv Rajendrabhai (20CS10088)

Sloth Coders

March 30, 2022

Contents

1 Unit Testing
- 1.1 Developer
  - 1.1.1 submit_code()
- 1.2 Contestant
  - 1.2.1 signup()
  - 1.2.2 login()
  - 1.2.3 logout()
  - 1.2.4 submit_code()
  - 1.2.5 evaluate()
- 1.3 Judge
  - 1.3.1 signup()
  - 1.3.2 login()
  - 1.3.3 logout()
  - 1.3.4 create_problem()
- 1.4 Administrator
  - 1.4.1 signup()
  - 1.4.2 login()
  - 1.4.3 logout()
  - 1.4.4 submit_code
  - 1.4.5 create_problem()

2 Some Images of Software

1 Unit Testing

We followed the following steps for Unit Testing of the software:

- Examination of requirements and specification of the software.
- Equivalence classes are created by considering all possible valid and invalid inputs for various scenarios.
- Proper inputs from each possible equivalence class are designed.
- For each input, the expected output is determined.
- After the implementation would be completed, the expected output would be compared with the actual output.

We test functions of different users separately as shown below.

1.1 Developer

We tested the following functionalities for Developer:

1.1.1 submit_code()

Input	**Expected Output**	**Actual Output**
Submit a code and provide its input	Should compile the program,return the error if any else display the output of the program	Compiling the program,returning the error where there are errors and also displaying the output
Submit a code and do not provide its input even if it requires input	Should compile the program,return error if any, will return pass/ fail message(not output)	Compiling the file but then giving runtime error
Do not submit a code	Should give compilation error	Does not allow empty submission

Example Input/Output

General Input	General Output
• userId : int - unique • code : string - long • input : string - long	• output : string • errors : string

Input	Output
• userId : "Dhruv" • code : "#include < stdio.h > int main() { printf("hello"); }" • input : ""	• output : "hello" • errors : ""

1.2 Contestant

We tested the following functionalities for Contestant:

1.2.1 signup()

Input	Expected Output	Actual Output
Create a contestant with all details	Details get saved in corresponding fields	Details are stored in the database and the contestant is able to login successfully
Create contestant with missing detail(s)	Appropriate failure response	Unable to sign up
Create contestant with duplicate email	Appropriate failure response	Unable to sign up

General Input	General Output
• name : string , Ex=”Dhruv” • email : string - proper a@b.c format, Ex=”dhruv@gmail.com” • password : string - length > 8 characters, Ex=”dhruv@123”	• userId : int - unique, Ex=10088 • success/ error response : string, Ex=”User created” • call login()

1.2.2 login()

Input	Expected Output	Actual Output
Login as existing contestant	Should return success response and create/ update applicable SESSION variables locally	Allowing to log in
Login as a non-existent contestant	Appropriate failure response	Not allowing to log in the software and giving the error
Login with wrong password	Appropriate failure response	Not allowing to log in the software and giving the error

General Input	General Output
• userId : int - unique, Ex=10088 • name : string, Ex="Dhruv" • email : string - proper a@b.c format, Ex="Dhruv@gmail.com" • password : string - length > 8 characters, Ex="dhruv@123"	• success/ error response, Ex="Logged in" • set appropriate SESSION variables locally

1.2.3 logout()

Input	Expected Output	Actual Output
Logout	Should UNSET applicable SESSION variables	Succesfully logged out of the software

General Input	General Output
	• success/ error response,Ex="logged out" • unset appropriate SESSION variables locally

1.2.4 submit_code()

Input	Expected Output	Actual Output
Submit a code and provide its input	Should compile the program,return the error if any else display the output of the program	Compiling the program, showing the errors if there are any and showing the output on succcessful execution
Submit a code and do not provide its input even if it requires input	Should compile the program,return error if any, will return pass/ fail message(not output)	Giving runtime error
Do not submit a code	Should give compilation error	Submission nor allowed

General Input	General Output
• userId : int - unique, Ex=10088 • problemId : int - unique • code : string - long, Ex="#include <stdio.h>int main() {printf("hello");}" • input : string - long, Ex=""	• output : string, Ex="hello" • success/ errors : string, Ex=0

1.2.5 evaluate()

Input	Expected Output	Actual Output
Select a problem and submit a code for that	Call submit_code() to compile, check against expected output(s); displays the number of passed cases and score	Selecting the problem, providing the code and on clicking the submit button, it is giving the result according to the code provided :AC,RE,TLE, CE,etc

General Input	General Output
• userId : int - unique, Ex=10088 • contestId : int - unique, Ex=731 • problemId : int - unique, Ex=238 • code : string - long, Ex="include <stdio.h>int main(){printf("hello");}" • input : string - long, Ex=""	• call submit_code() with additional parameters • update score using userId

1.3 Judge

We test the following functionalities for Judge:

1.3.1 signup()

Input	Expected Output	Actual Output
Create Judge with all details in expected format	All the details (except password as it is encrypted) as used while creation should be same	Making an account and storing the details of the judge
Create Judge without a detail	Appropriate failure response	Showing the failure message
Create Judge with duplicate email	Appropriate failure response	Showing the failure message

General Input	General Output
• name : string • email : string - proper a@b.c format • password : string - length > 8 characters	• userId : int - unique • success/ error response : string • call login()

1.3.2 login()

Input	Expected Output	Actual Output
Login as existing judge	Should return success response and application request acceptance should work	Allowing to get logged in, allowing to use the features available for the judge only.
Login as non-existent judge	Appropriate failure response	Not allowing to login,showing error message
Login with wrong password	Appropriate failure response	Not allowing to login, showing error message

General Input	General Output
• userId : int - unique,Ex=10088 • name : string,Ex="Dhruv" • email : string - proper a@b.c format,Ex="dhruv@gmail.com" • password : string - length > 8 characters,Ex=dhruv@123	• success/ error response • set appropriate SESSION variables locally

1.3.3 logout()

Input	Expected Output	Actual Output
Logout	Should UNSET applicable SESSION variables	Successfully logged out of the software

General Input	General Output
	• success/ error response,Ex="logged out" • unset appropriate SESSION variables locally

1.3.4 create_problem()

Input	Expected Output	Actual Output Output
Provide problem statement and all test cases and corresponding expected outputs, time limit and memory limit	Should create problem with the provided details	Creating problems, and showing its details in the Practice part of the software for the contestants and are ready to be solved
Leave any of the above parameters uninitialised	Should display error "Problem cannot be created" and ask the user to enter details again	Showing error message

General Input	General Output
• problem : string - long • Ex: "Sorting Problem" • tests : dictionary of (string, string) - (input, output) • Ex: "test-1" • timeLimit : int • Ex: "1s" • memLimit : int • Ex: "128Kb" • score : int • Ex: "100"	• problemId - unique

1.4 Administrator

1.4.1 signup()

Input	Expected Output	Actual Output
Sign up with all the details and a license key	Succesful sign up and password set up if the license key is correct	Making the account successfully and storing the details in the database
Sign up with a wrong license key	Should give error	Showing error message
Sign up with less details	Should give error	Showing error message

General Input	General Output
• name : string Ex: "Dhruv" • email : string - "propera@b.c" format • password : string - length > 8 characters Ex: "Roopak"	• userId : int - unique Ex:"123" • success/ error response : string Ex :"Account Created Successfully" • call login()

1.4.2 login()

Input	Expected Output	Actual Output
Login as administrator that exists	Should return success response	Successfully able to login
Login as administrator that does not exist	Failure response	Showing error message
Login with wrong password	Failure response	Showing error message

General Input	General Output
• name : string Ex:"Dhruv" • email : string - "propera@b.c" format • password : string - length > 8 characters Ex:" Swarup"	• userId : int - unique Ex :10088 • success/ error response : string "Logged in Succesfully" • call login()

1.4.3 logout()

Input	Expected Output	Actual Output
Logout	Should UNSET applicable SESSION variables	Successfully logged out of the software

General Input	General Output
	• success/ error response,Ex="logged out" • unset appropriate SESSION variables locally

1.4.4 submit_code

Input	Expected Output	Actual Output
Submit a code and provide its input	Should compile the program,return the error if any else display the output of the program. This task is added to check the results of the software and see whether it is working fine or not.	Showing correct output for the corresponding code
Submit a code and do not provide its input even if it requires input	Should compile the program,return error if any;will return pass/ fail message(not output)	Showing compilation error
Do not submit a code	Should give compilation error	Showing compilation error

General Input	General Output
• userId : int - unique, Ex: "132" • problemId : int - unique, Ex: "156" • code : string - long, Ex:"# include <stdio.h> int main() { printf("Hello World"); return 0; }" • input : string - long, Ex:"5 9 7 8"	• output : string, Ex = "Hello World" • success/errors : string, Ex = 0

1.4.5 create_problem()

Input	Expected Output	Actual Output
Provide problem statement and all test cases and corresponding expected outputs, time limit and memory limit	Should create problem with the provided details	Problem is created and contestants are able to see it in the Practice tab
Leave any of the above parameters uninitialised	Should display error "Problem cannot be created" and ask the user to enter details again	Showing error message

General Input	General Output
• problem : string - long • Ex: "Sorting Problem" • tests : dictionary of (string, string) - (input, output) • Ex: "test-1" • timeLimit : int • Ex: "1s" • memLimit : int • Ex: "128Kb" • score : int • Ex: "100"	• problemId - unique Ex: "123"

2 Some Images of Software

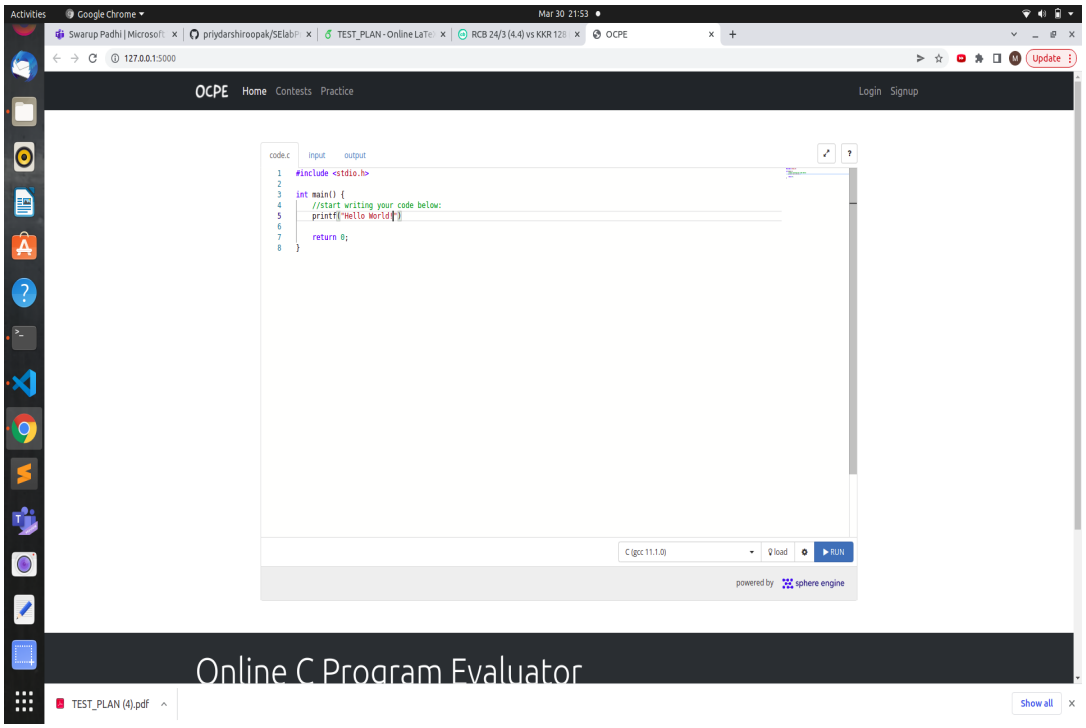

Figure 1 : Simple Code to compiler

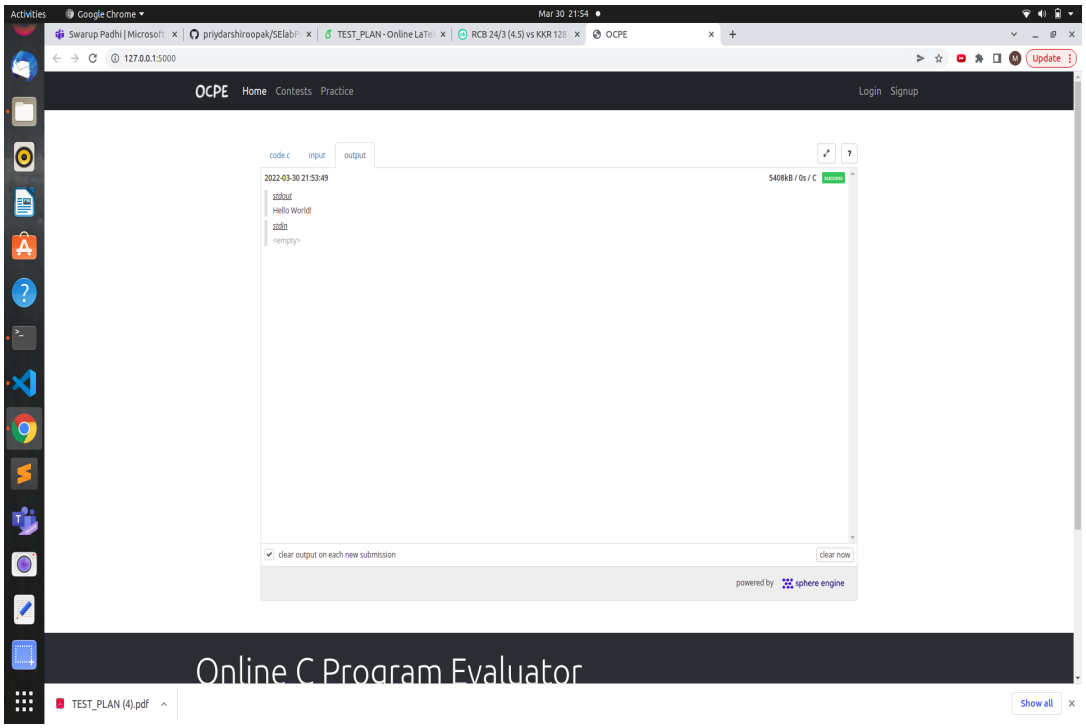

Figure 2 : Output of the above

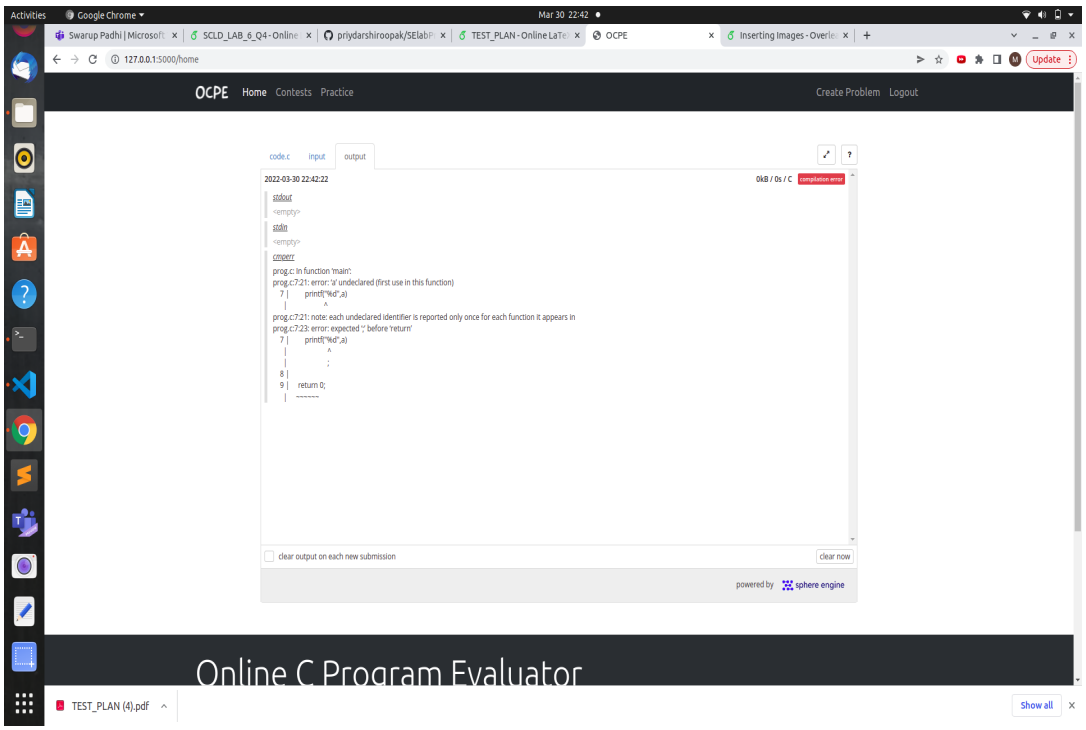

Figure 3 : Compilation Error

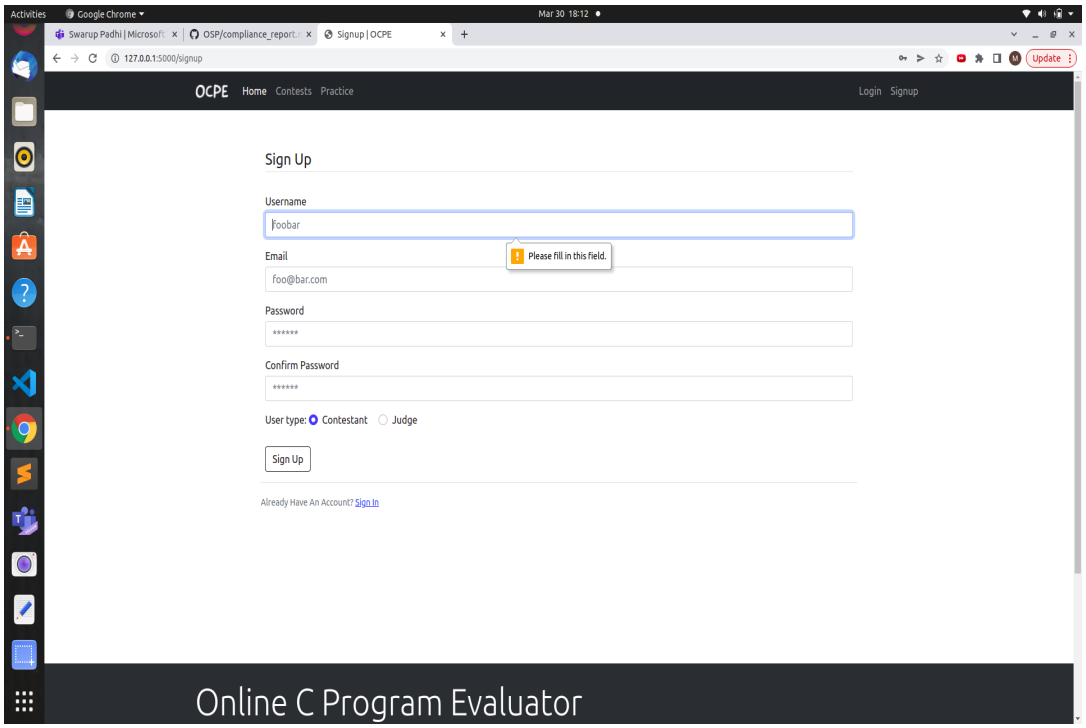

Figure 4 : Empty field error in sign up page

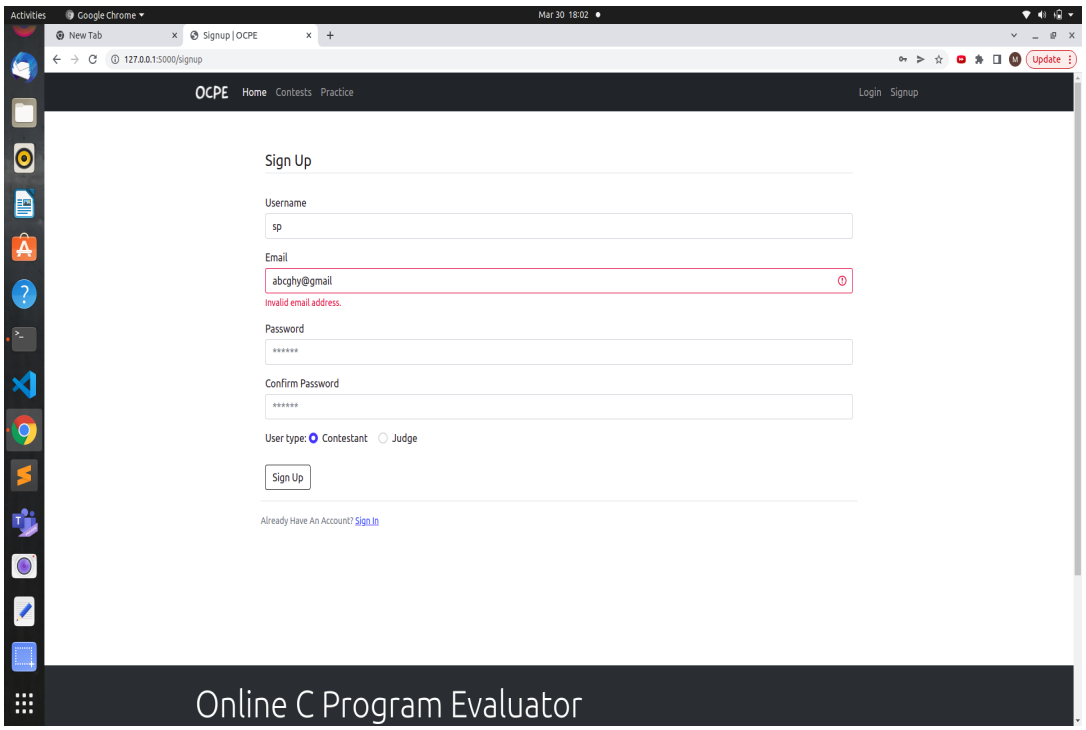

Figure 5 : Invalid email address

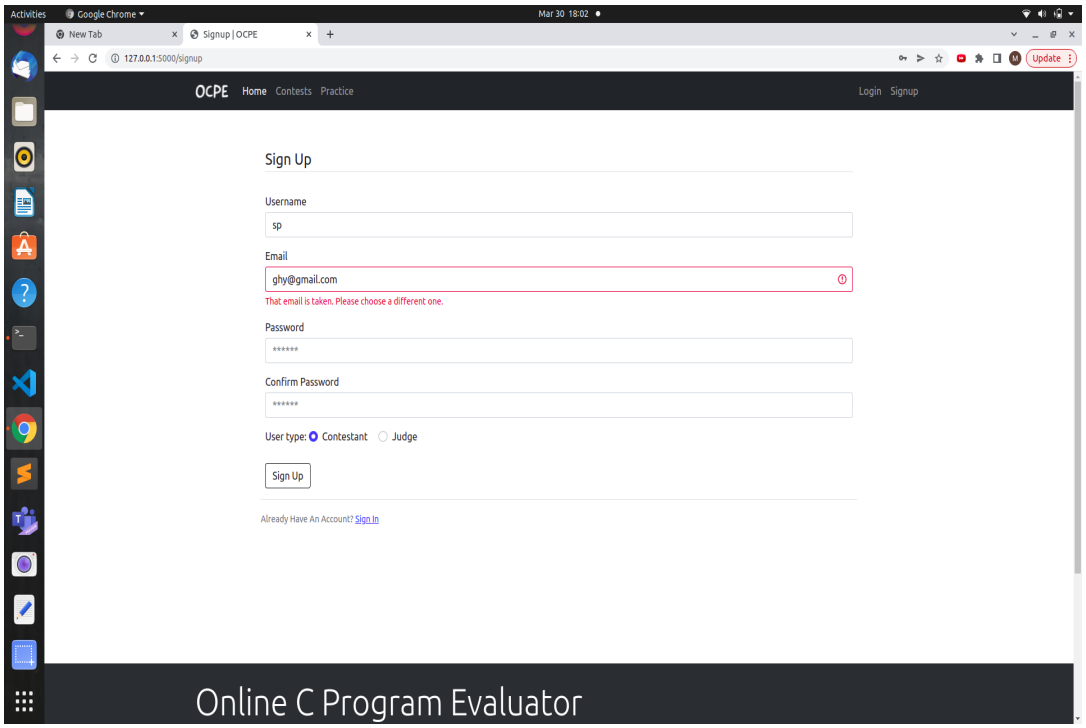

Figure 6 : Using Already used email address

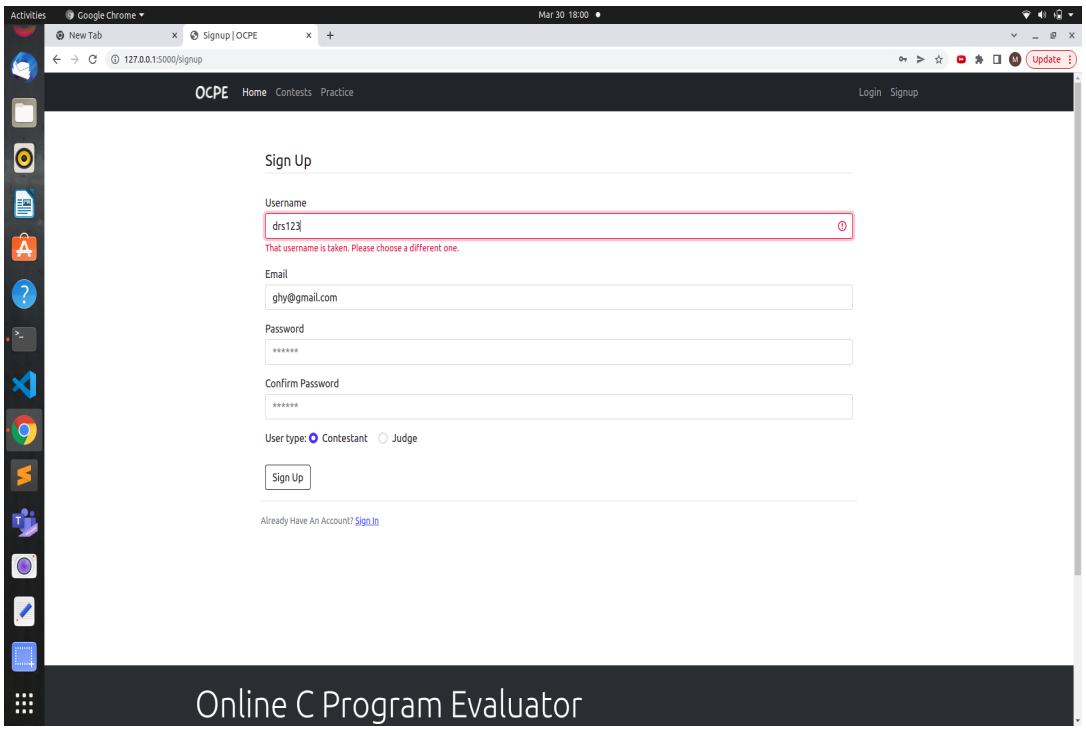

Figure 7 : Using Already used username

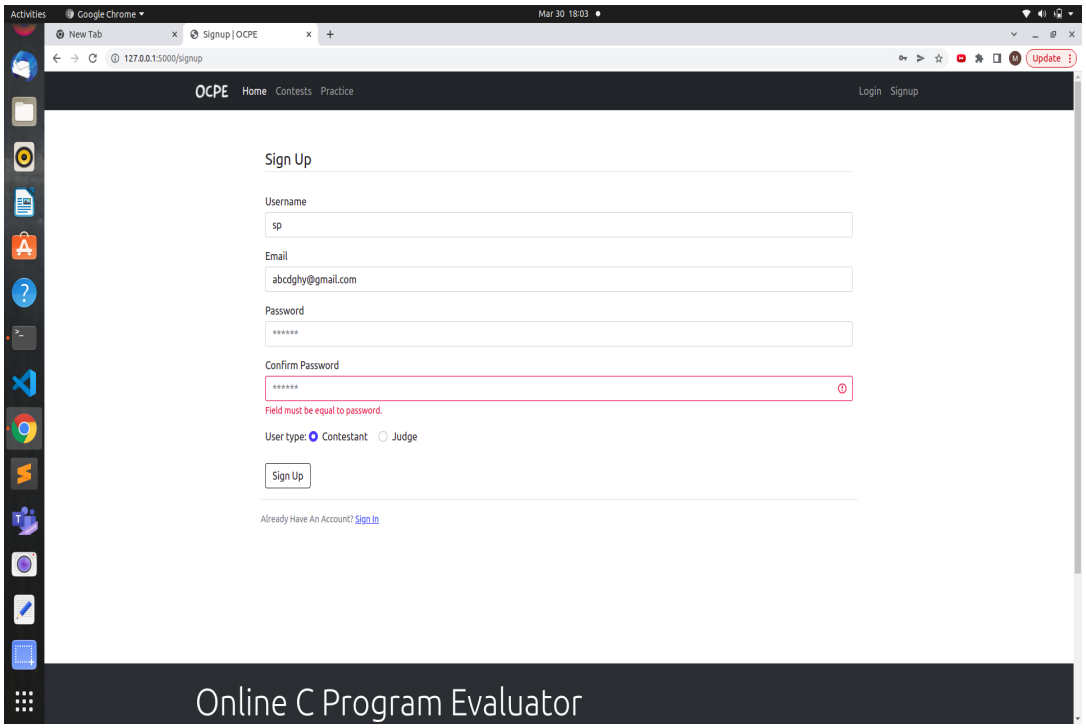

Figure 8 : Not matching confirm password

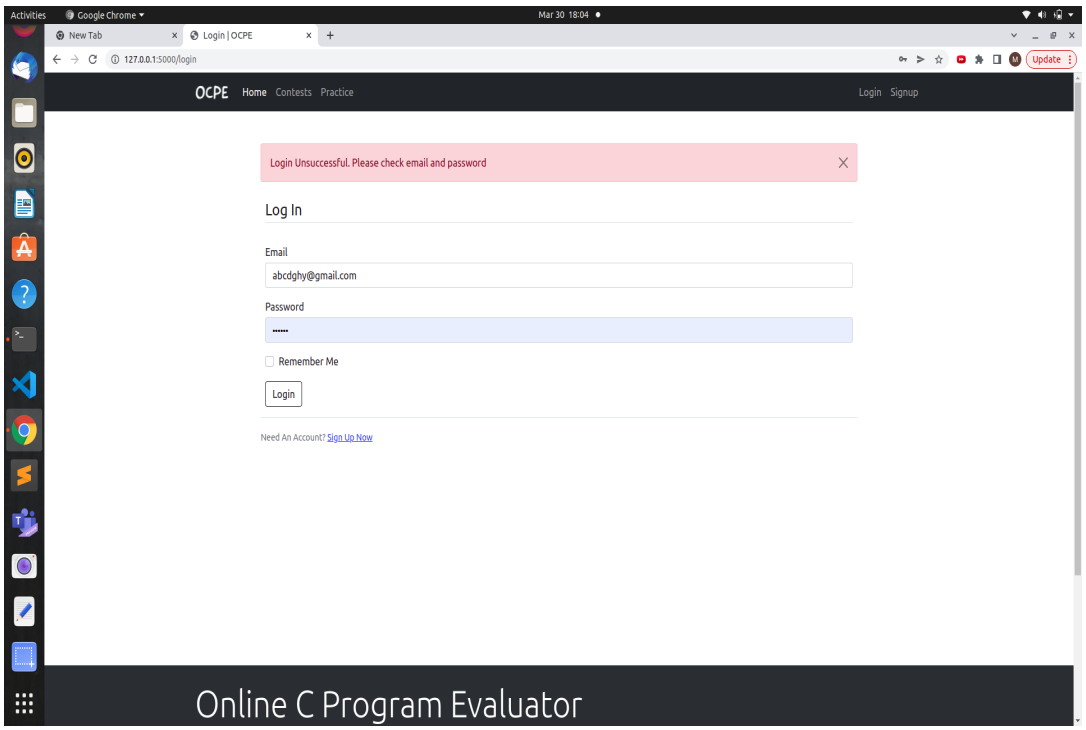

Figure 9 : Invalid login credentials

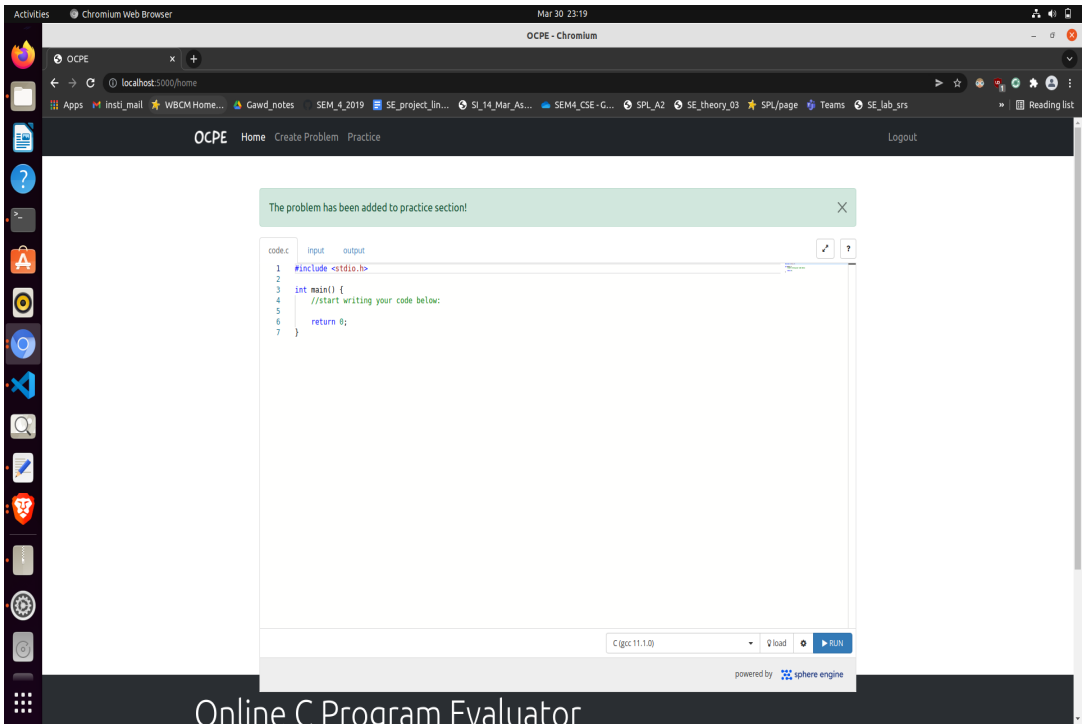

Figure 10 : Sucessful creation of problem by Judge

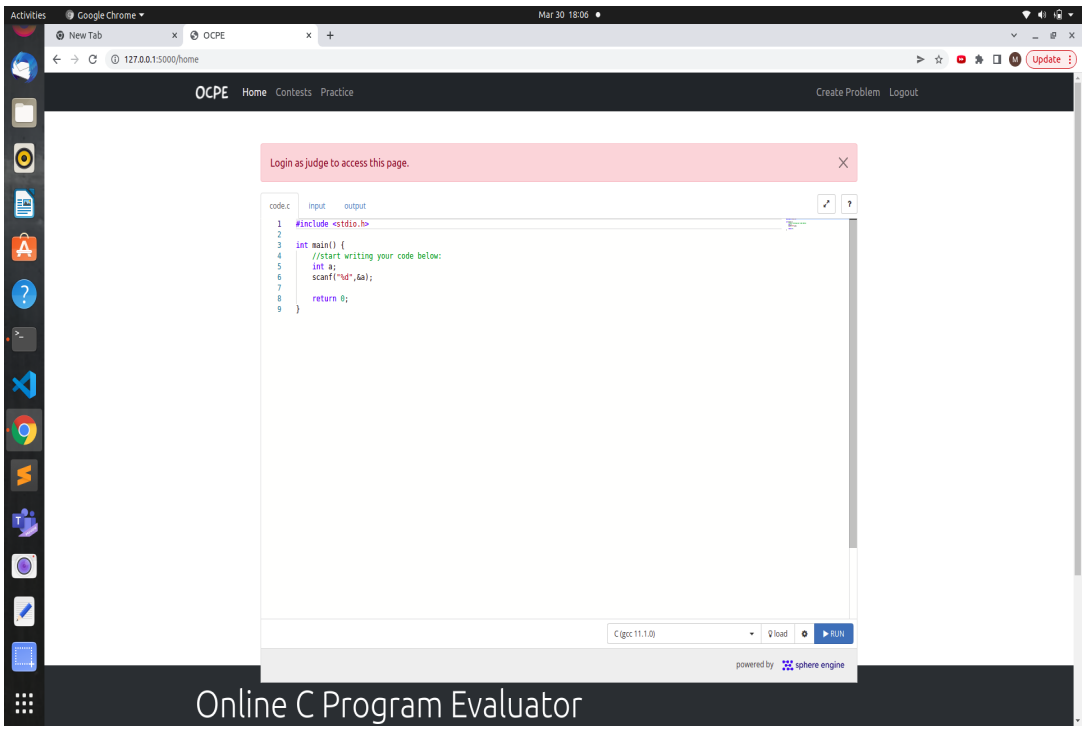

Figure 11 : Invalid access of creating questions by Contestants

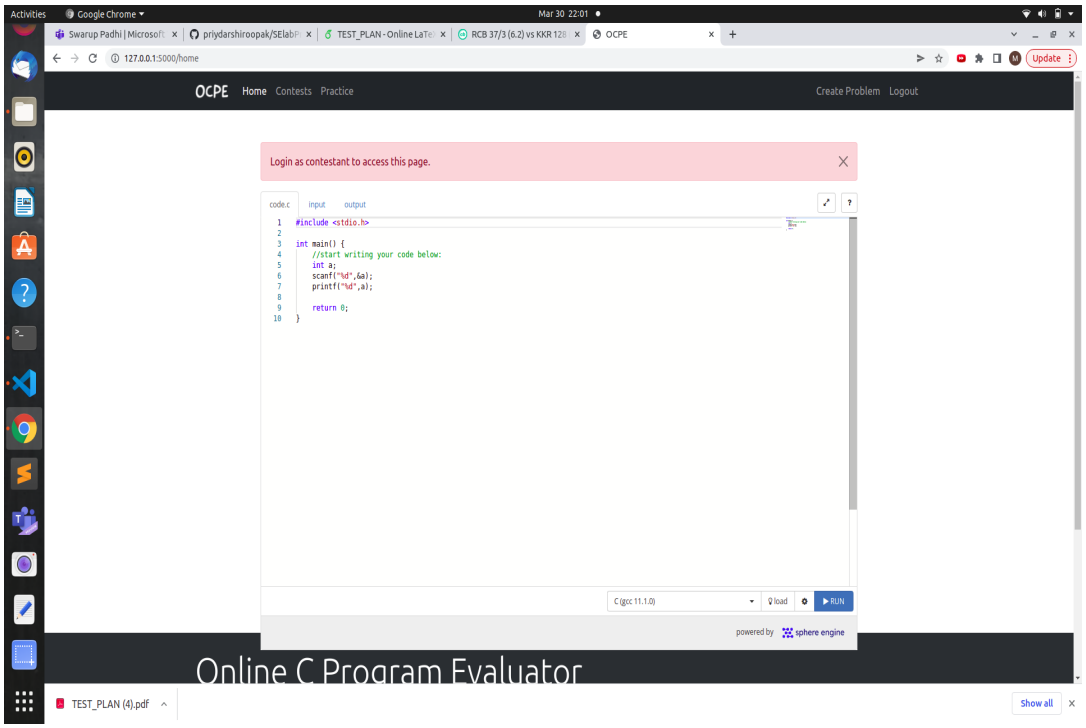

Figure 12 : Invalid access of Practice problems part by Judges

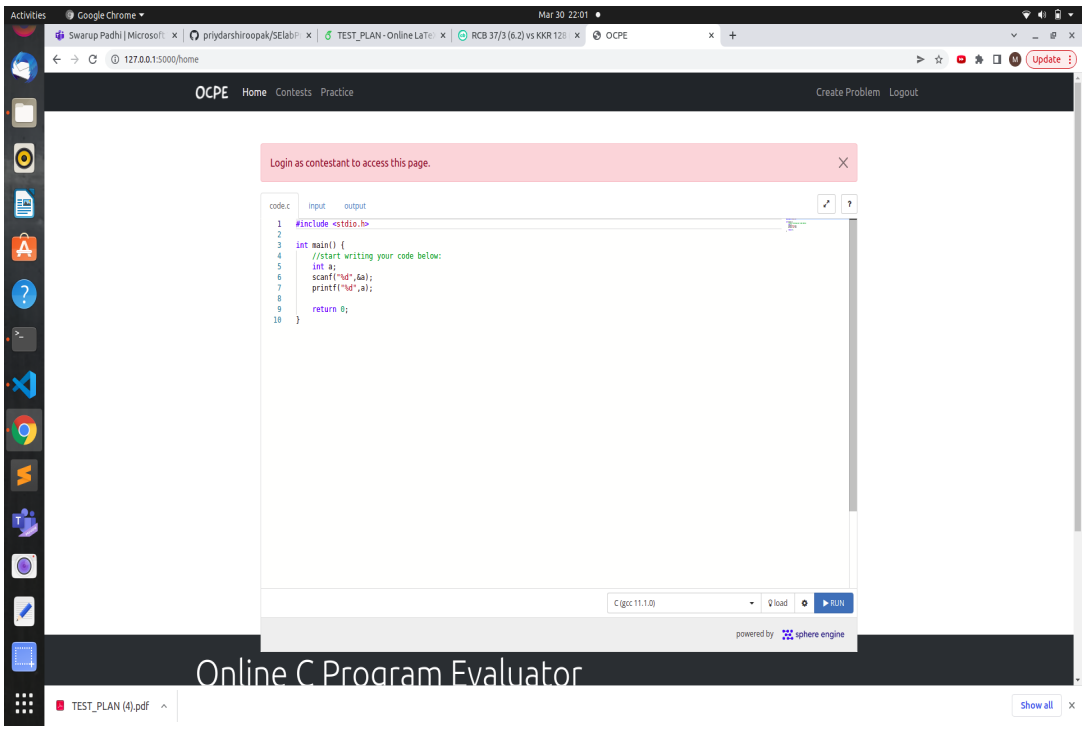

Figure 12 : Invalid access of Practice problems part by Judges

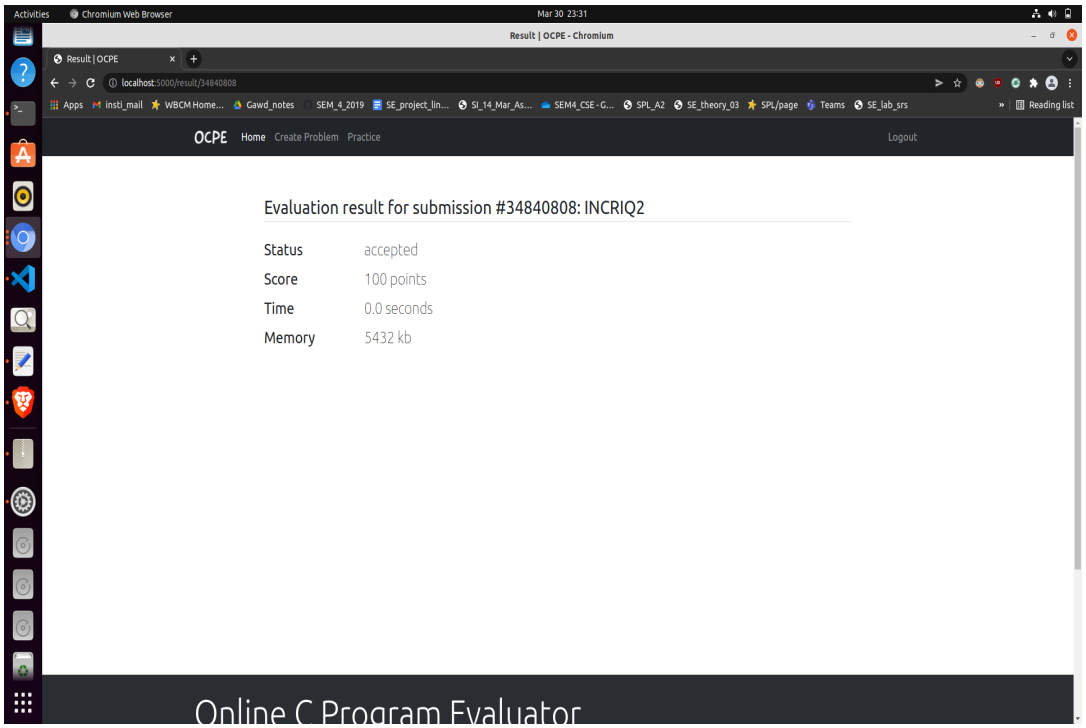

Figure 13 : Submission result of correct code

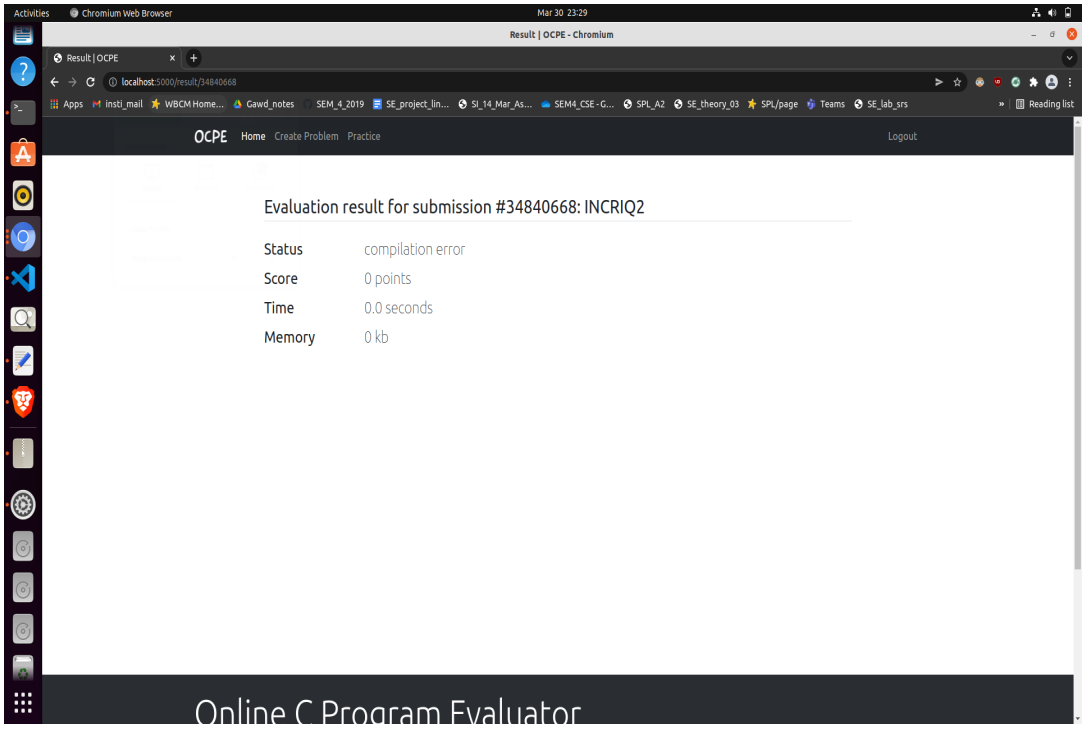

Figure 14 : Submission result of wrong code