Cloud Computing

Cloud Based Code Manager

Project Report

Branch: - ICT

Group No: - 2

Mentor: - Dr. Sanjay Chaudhary

Team Members:

Shreyas Patel	1401025
Ashutosh Kakadiya	1401075
Mihir Gajjar	1401076
Harsh Mehta	1401086

Acknowledgement

We would like to thank our academic mentor Dr. Sanjay Chaudhary for their persistent support and pivotal guidance throughout the course of the project and for providing us an opportunity to overcome theoretical knowledge by using and applying it to create realistic application.

Contents

Abstract

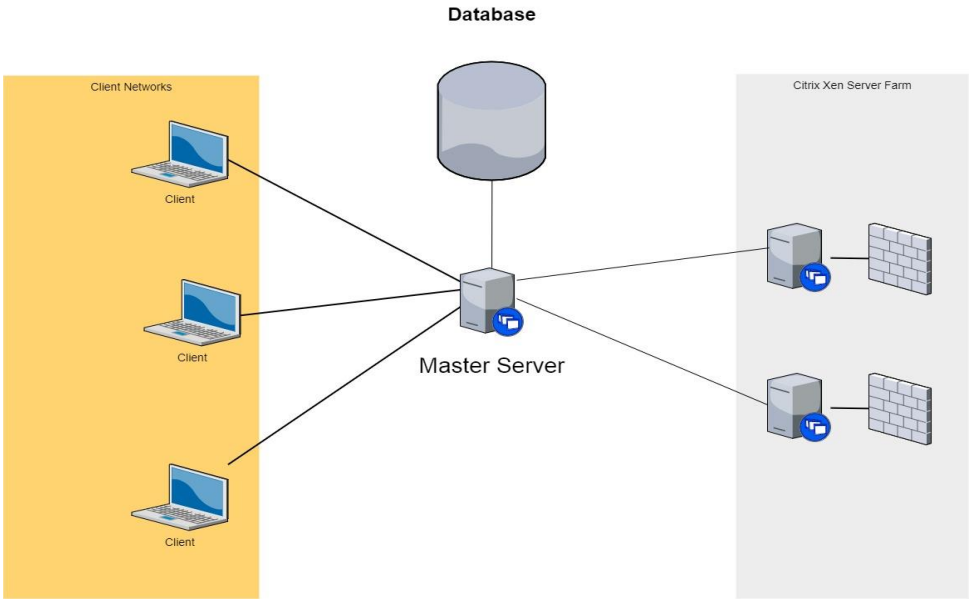
Build cloud based web application that provides developer an interface which helps in code compilation, execution, storage, analysis and also provides code optimisation related suggestion if required in code.

Project Description

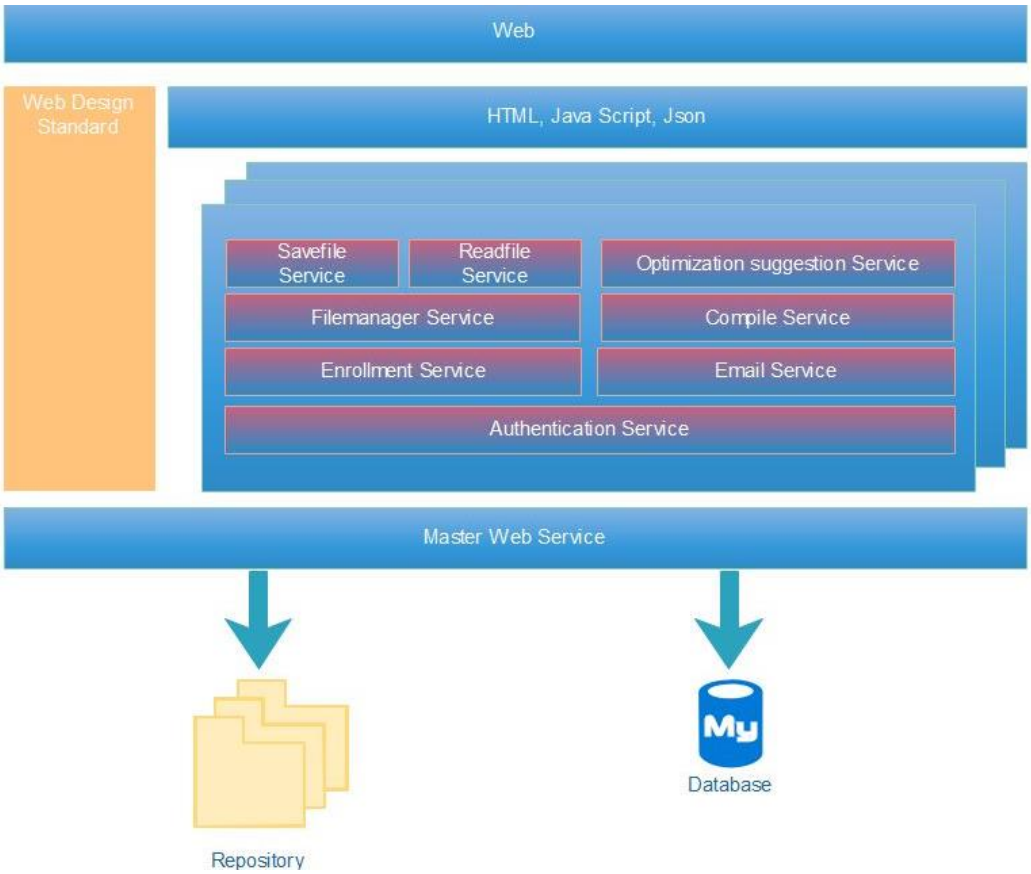
- Generally, for writing and practicing of code, programmer has to install different IDEs for different coding languages or has to set up coding environment on their PCs. Instead of that we are providing cloud based service oriented platform where, one can find different coding languages available online on cloud. One only needs a browser to access all services like authentication, registration, compilation, execution, analysis of code and get suggestions for code optimizations. User can also save their code for future reference and access.
- In code analysis, we provide the Memory and Time Usage of the code. In code optimization, we suggest the user that which lines in the code can be removed to improve the code performance.
- In future, we are planning to provide service which enables solution of error generated during code compilation.

Physical and Logical Design

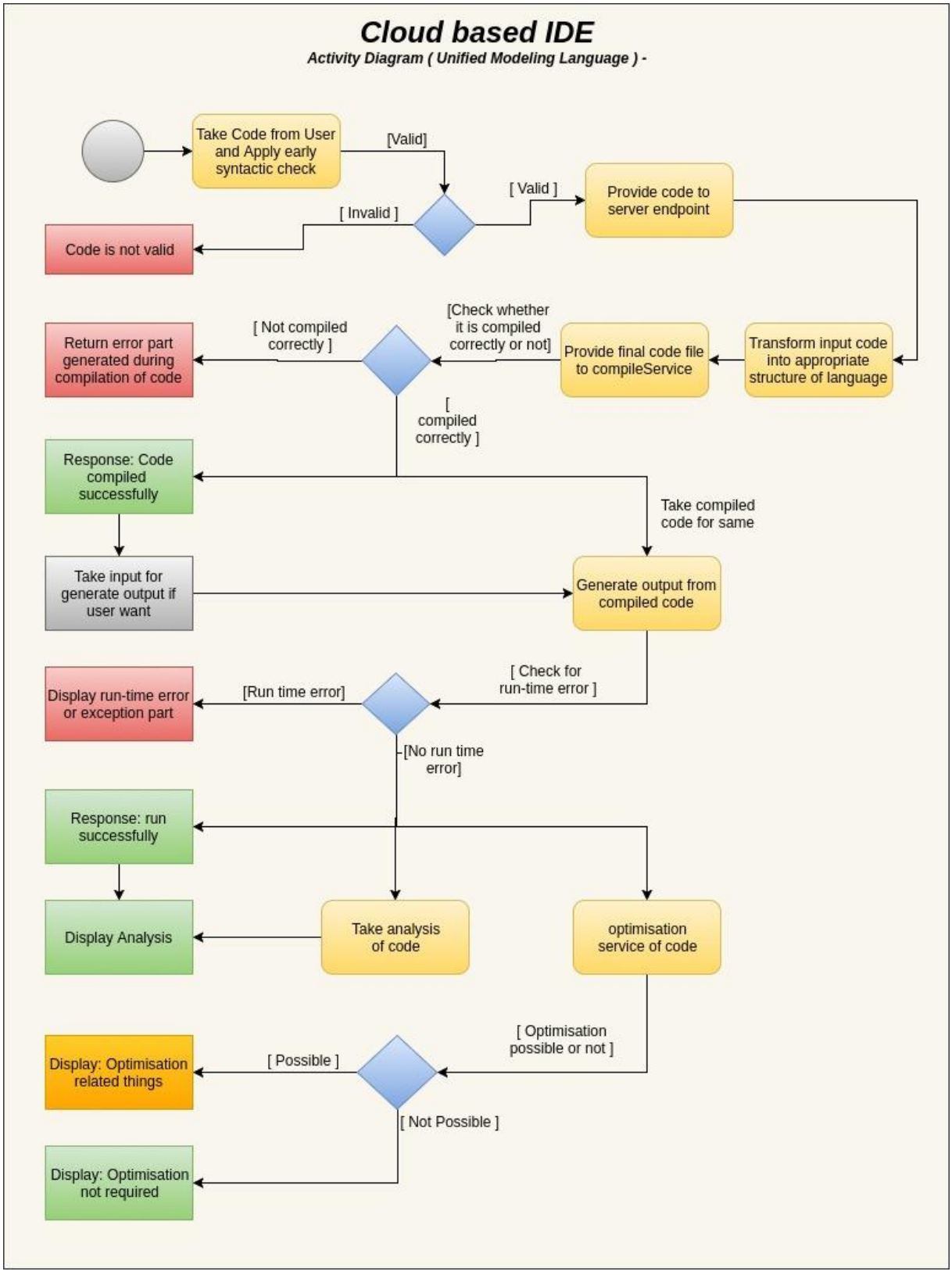
Cloud Architecture



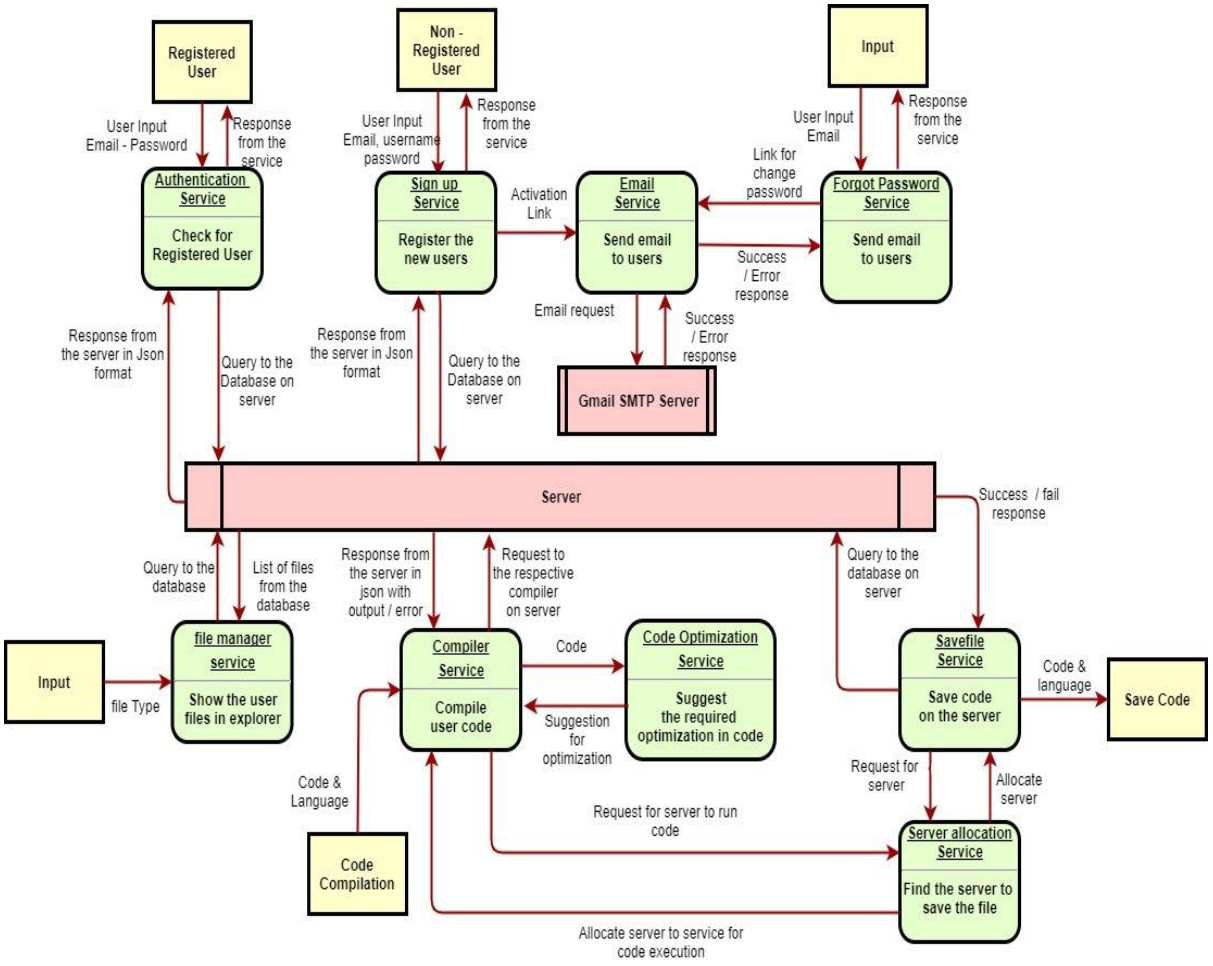
Application Architecture



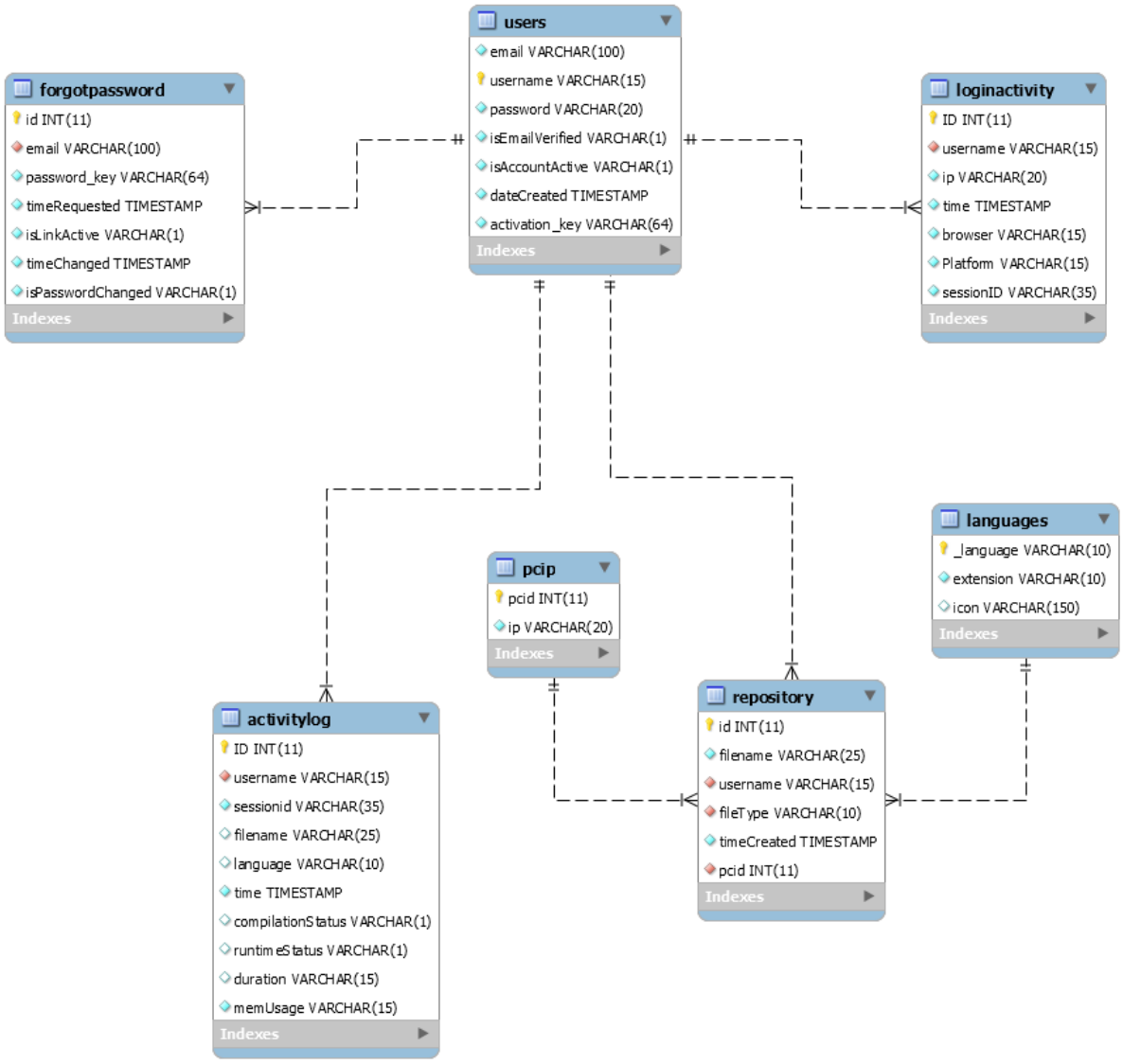
Logical Design



Dataflow Diagram



Database Schema



Services

Authentication

This service compares the credentials provided by the user with the database of authorized users' information. If the credentials match, the process is completed and the user is granted authorization for access. If the credentials do not match, the user is not granted access and is requested to enter valid credentials to gain access.

Sign-up

If the user is not registered, then the user can use this service and enter his details (sign - up) which will be added to the database of authorized users. This service provides the email service with the activation link that the user can use to complete the registration process.

Email

This service, using the **Gmail SMTP Server**, sends emails to the users containing the link for completing the registration process or for changing the password in case the user forgot the password.

Forgot password

In case the user has forgotten the password, this service can be used to change the password. It provides the email service with the link that the user can use to change the password.

File manager

This service shows the user all the files that the user has saved. In addition, it also provides the user, specific types of files which the user wishes to see.

Compilation

This service performs the primary function i.e. it performs the compilation of the codes for the users. It requests the server allocation service to allocate a machine for the compilation of the code. The user provides the service as an input: the code and the language in which it has to be compiled. If the compilation of the code was successful, then this service calls the code optimization service which takes the code as an input and provides suggestions for optimization. In this case, the output of the code and the suggestions for optimization of the code are both returned to the user. If the compilation of the code was not successful, it returns the error which occurred during the compilation process.

At this stage, the compiler service compiles the code for C, C++, Java and Python.

Code Optimization

This service takes the file containing the code as an input, analyses the code and provides suggestions, which if implemented can optimize the performance in terms of the memory usage or the run-time of the code.

At this stage, this service optimize the code for Java and Python.

Savefile

This service is used to save the code entered by the user in the database so that it can be accessed in the future. It requests the server allocation service to allocate a machine in which the code can be stored.

Server allocation

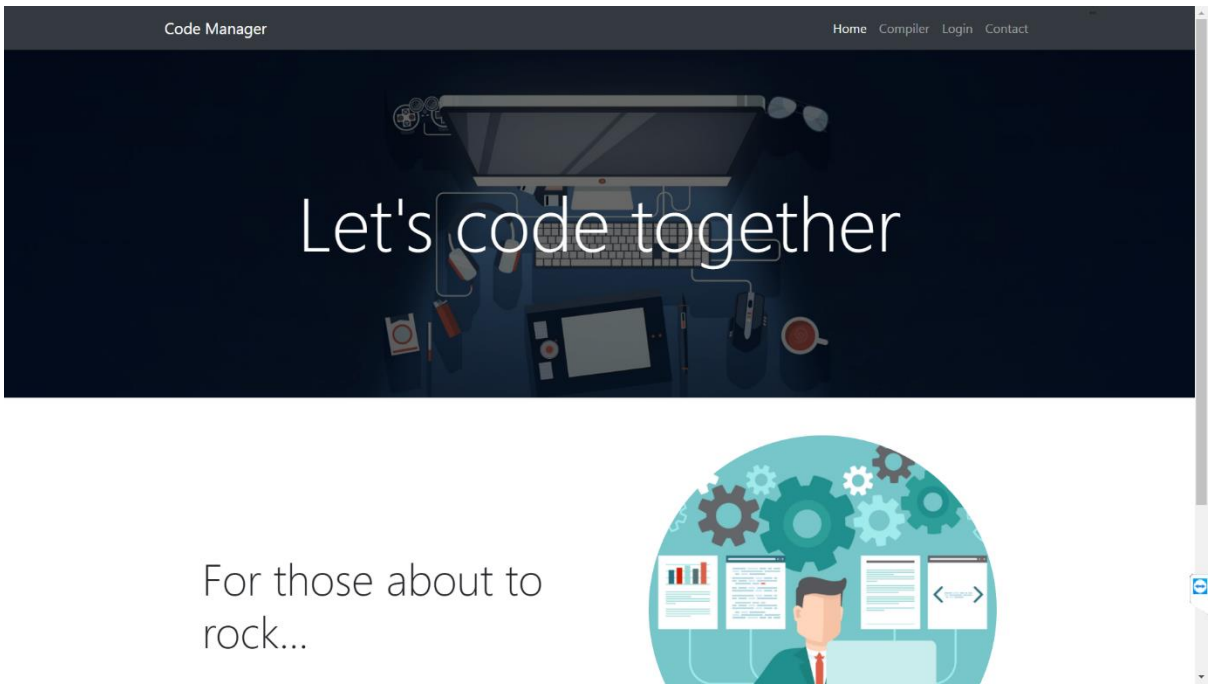
This service performs the function of allocating the machines and distributing the jobs for compilation and data storage. It checks the status for memory usage and storage availability in various machines that are available on the server side and allocates the machine which will be most suitable for compilation of the code or for storing the file containing the code.

Features

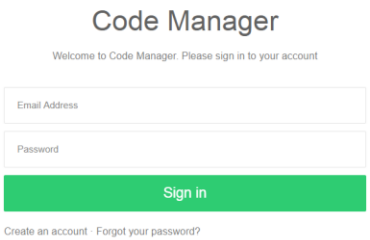
- Multitenancy — Developed
- Customizable GUI — Working
- Customizable business logic — Working
- Subscription — Developed
- Monitoring and billing — Developed [Monitoring]
- Security and privacy — Developed [privacy], Working on Security
- Scalability, high availability and reliability — Developed
- Managing and administration for separate tenant — Developed
- Runtime per tenant customization — Working

Result

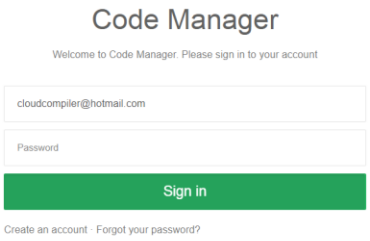
- Home Screen



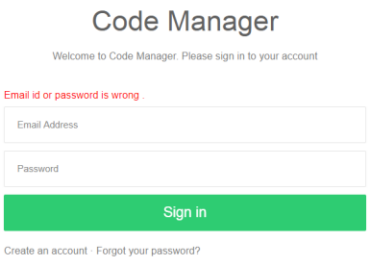
- Login Screen



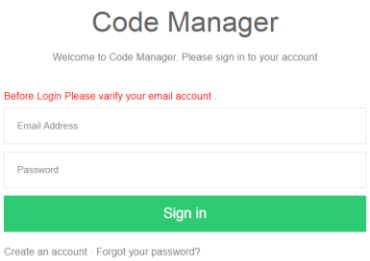
- Error based on user side input validation



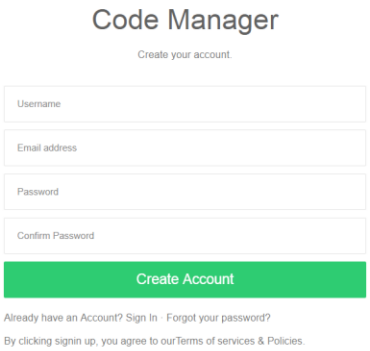
- Error based on server side validation



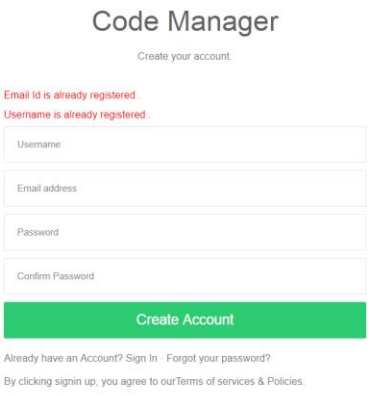
- User need to verify the account before login by clicking the link sent to user's mail account



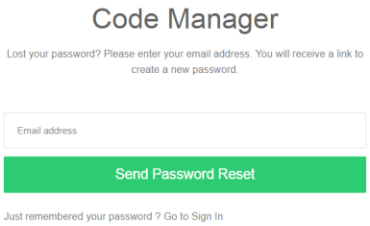
- Sign up screen



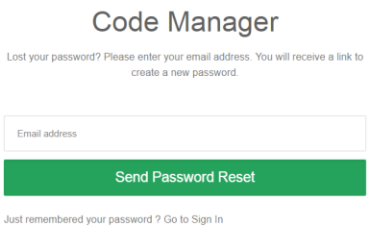
- Server side validation for sign up



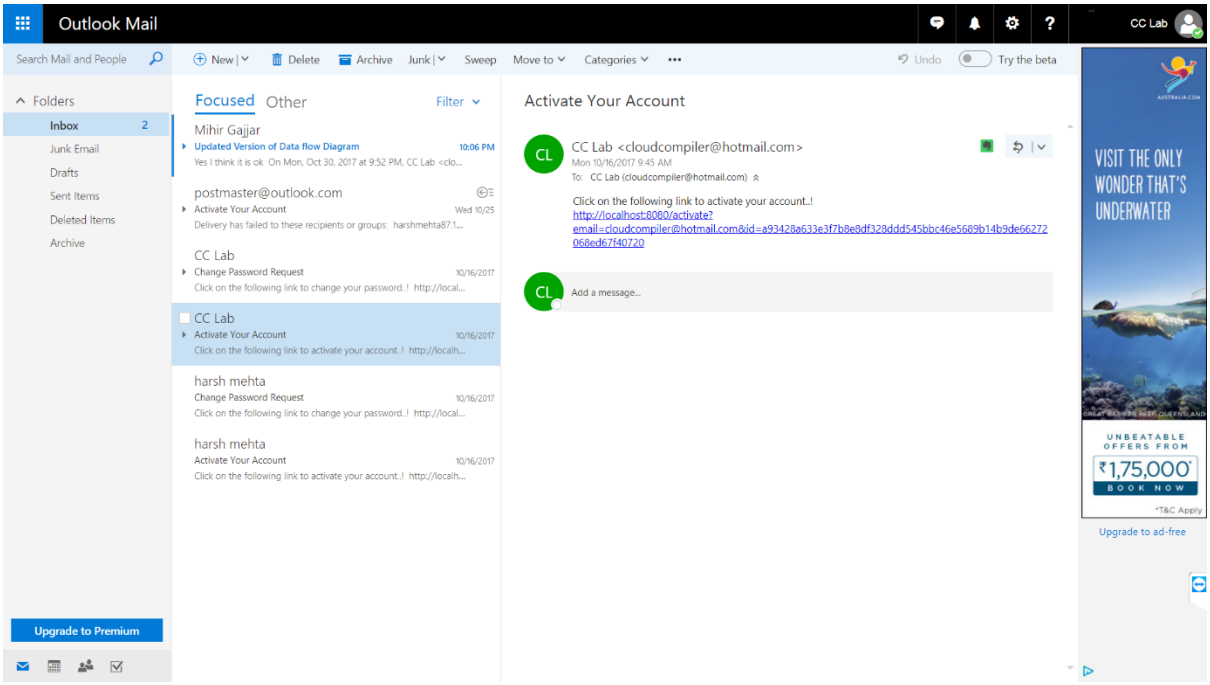
- Forgot password Screen



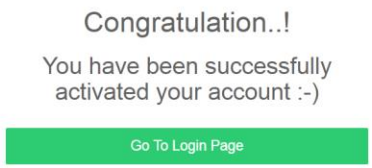
- User side validation for forgotpassword



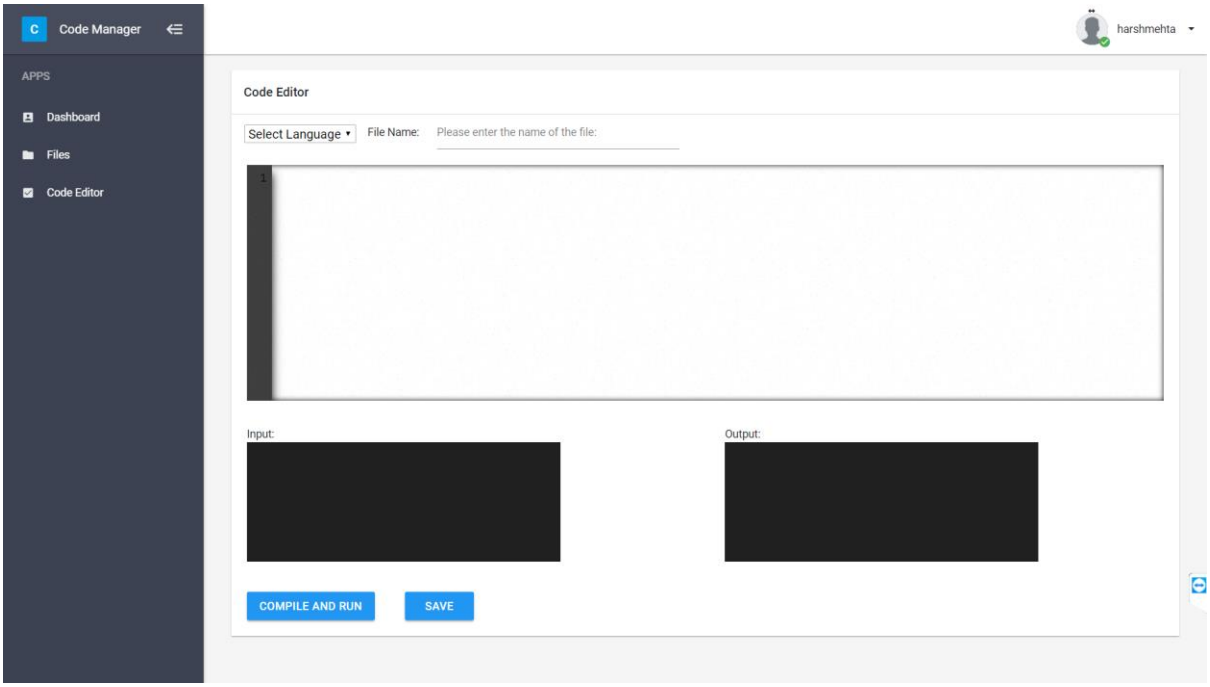
- Link sent to user to activate account



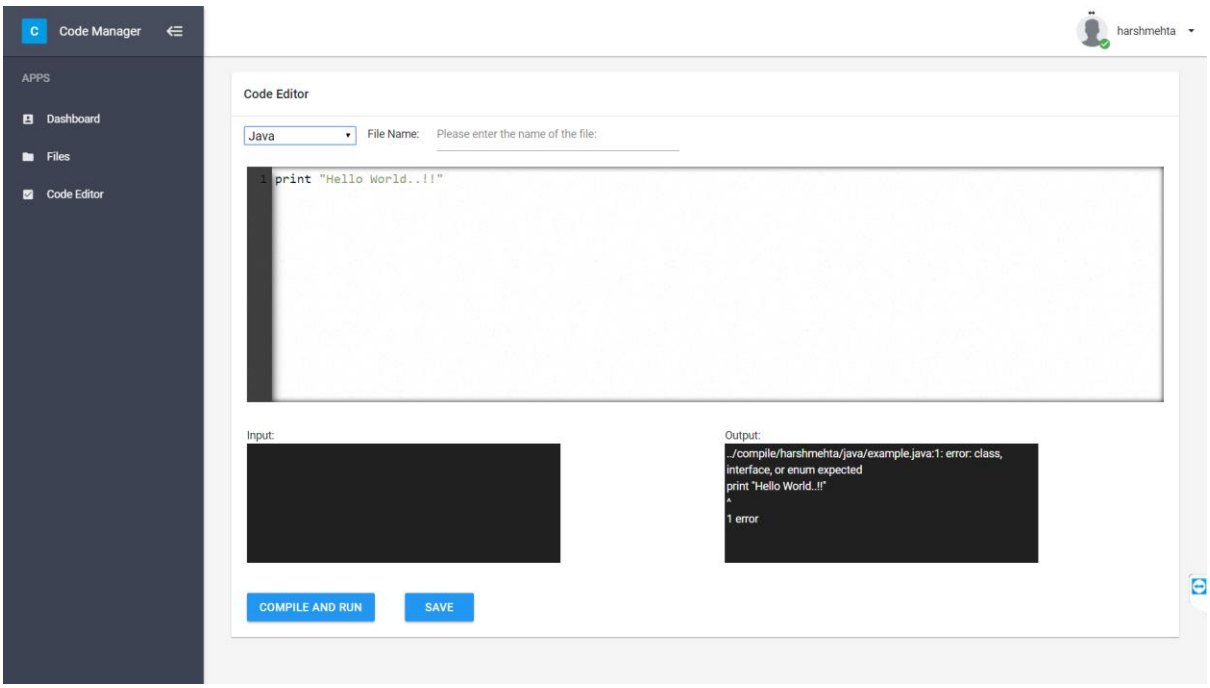
- When user go to the link for activation of an account



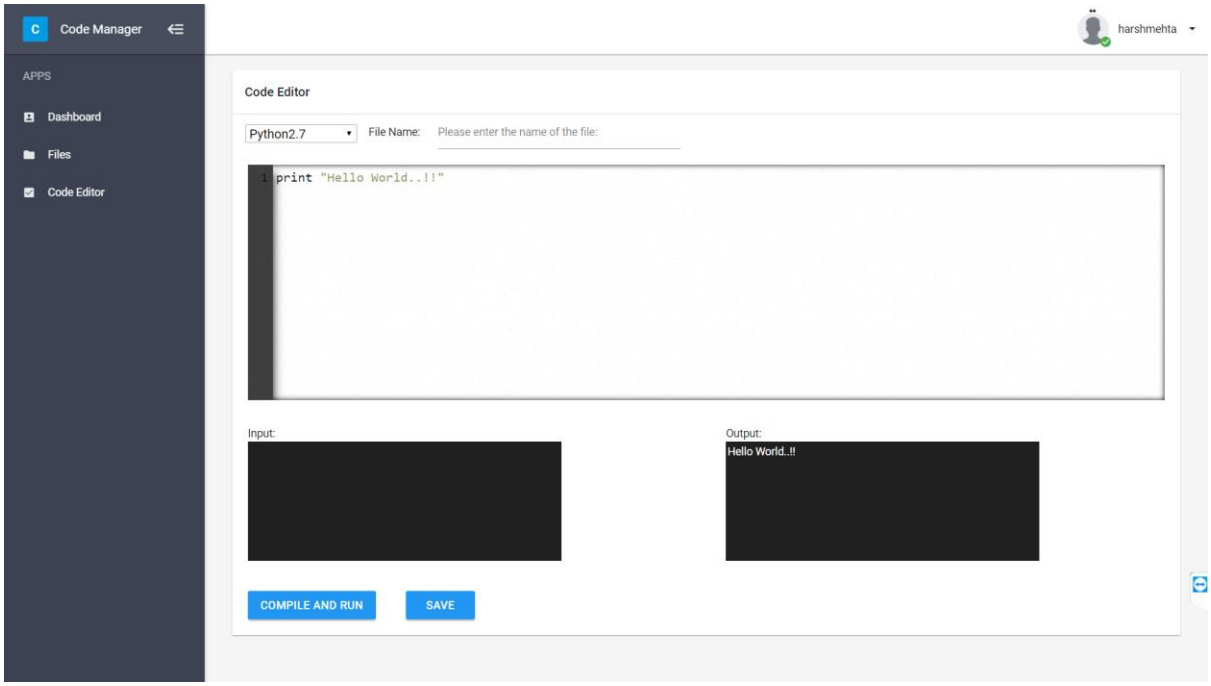
- Code Editor screen



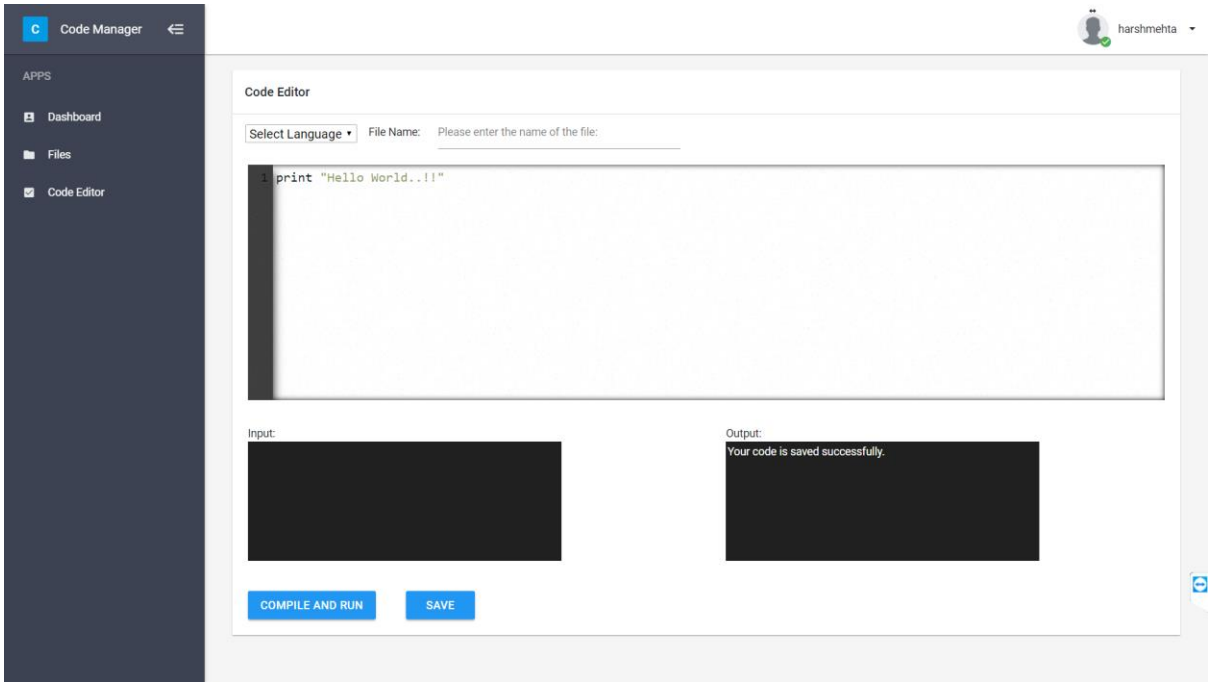
- Compilation error is showing in output



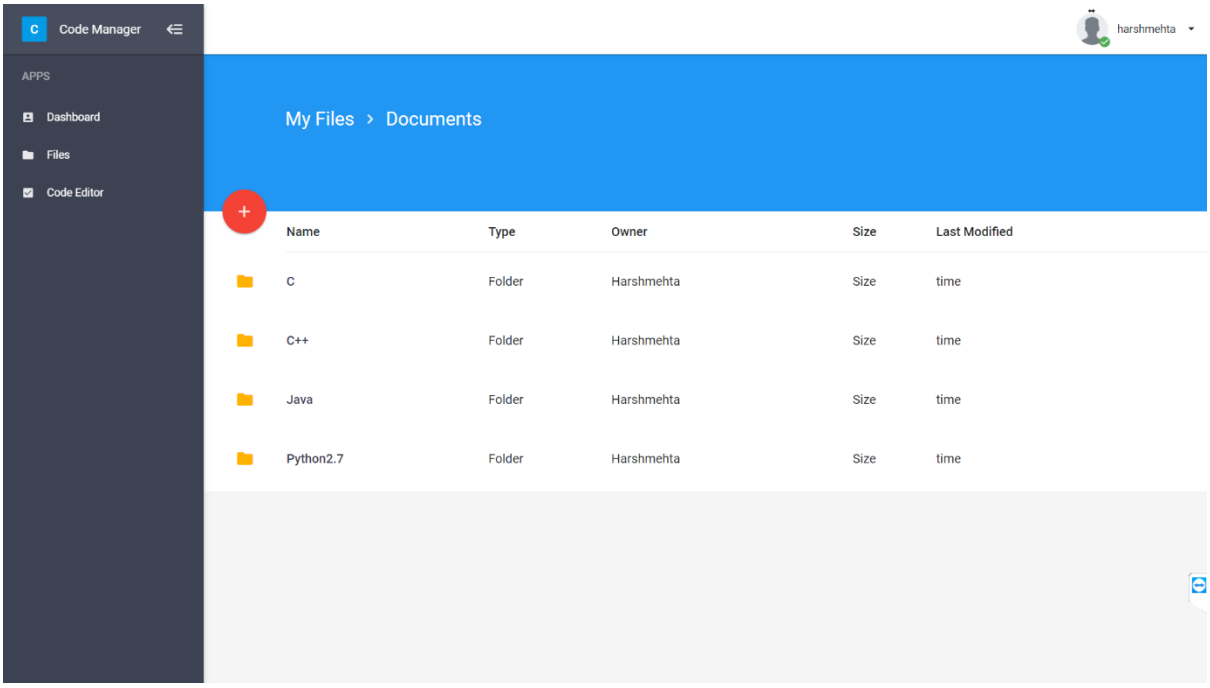
- Successful compilation

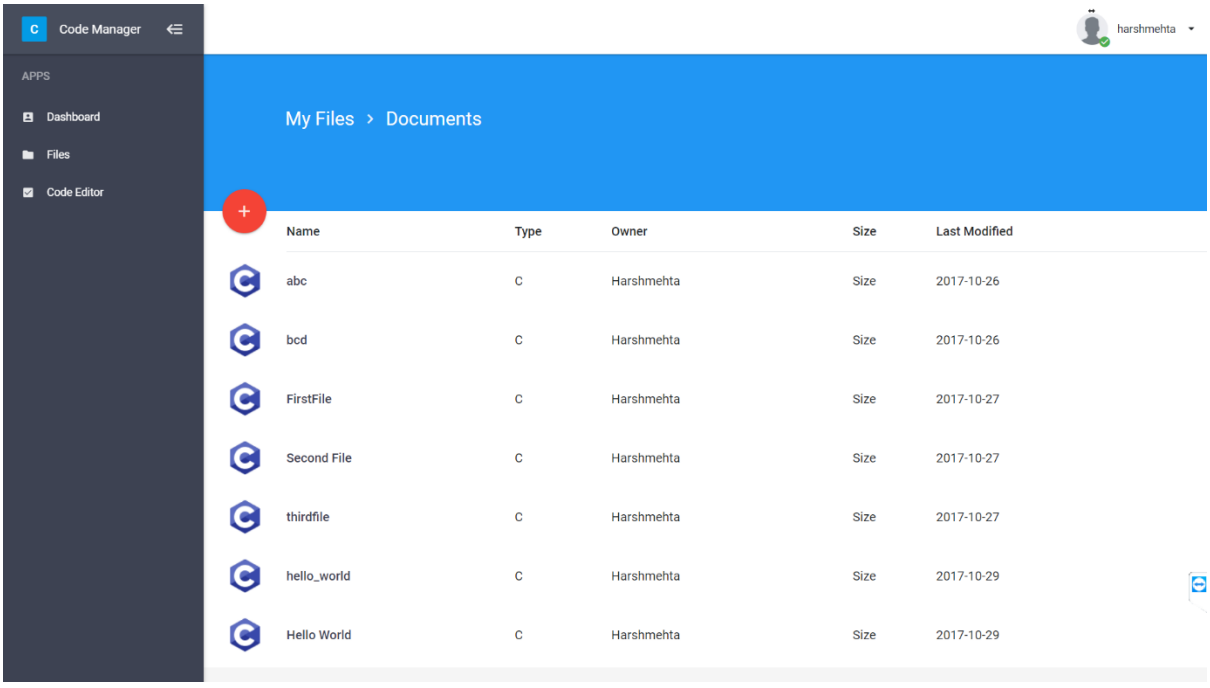


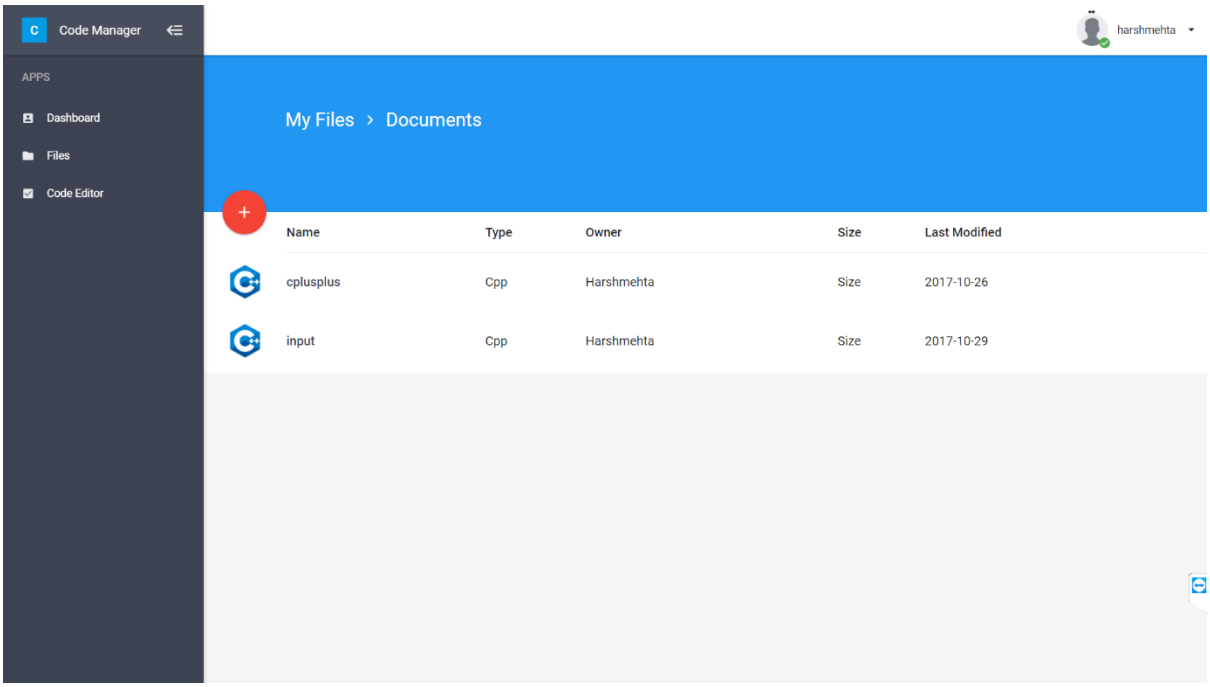
- Save file

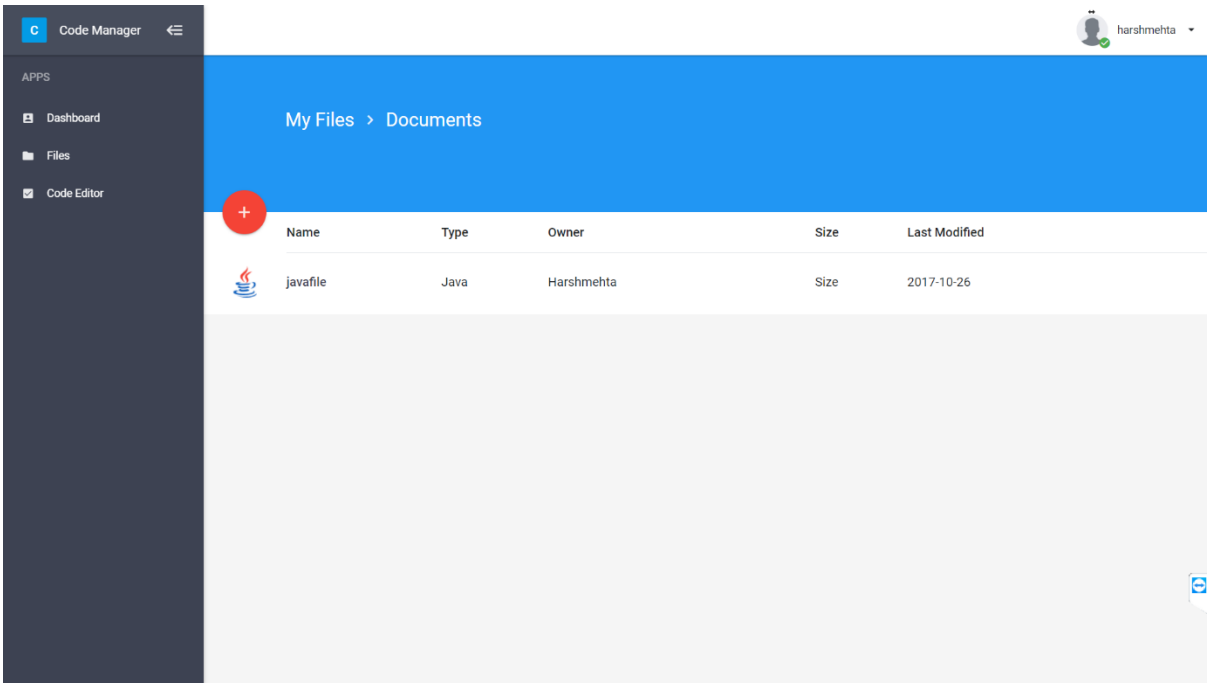


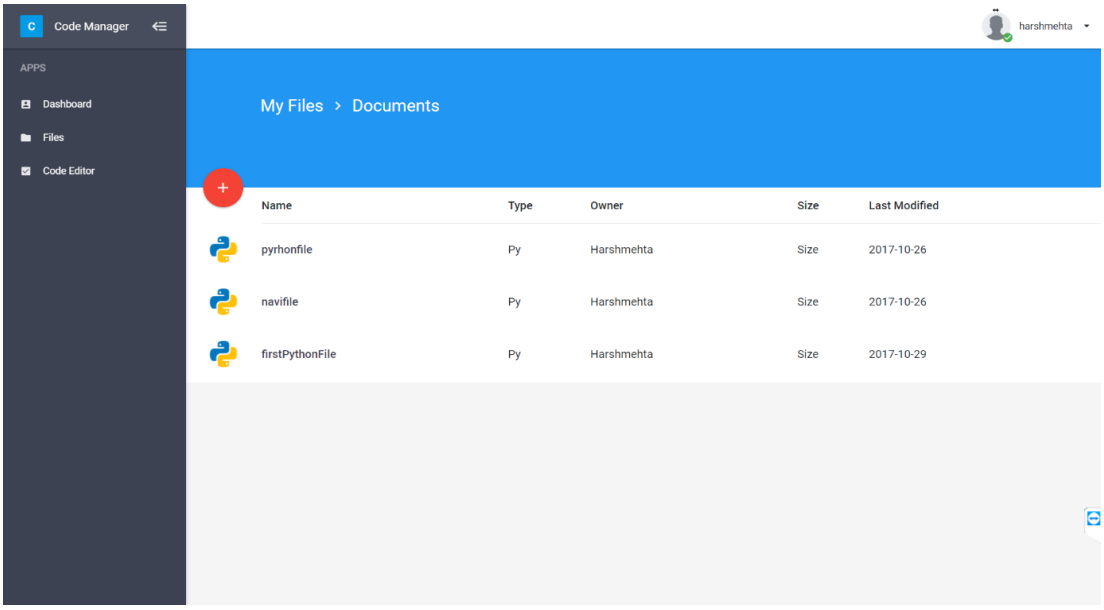
- File manager screen






Code Manager
APPS
Dashboard
Files
Code Editor
harshmehta
My Files › Documents
Name
Type
Owner
Size
Last Modified
cplusplus
Cpp
Harshmehta
Size
2017-10-26
input
Cpp
Harshmehta
Size
2017-10-29


Code Manager
APPS
Dashboard
Files
Code Editor
harshmehta
My Files › Documents
Name
Type
Owner
Size
Last Modified
javafile
Java
Harshmehta
Size
2017-10-26



Conclusion

Successfully implemented cloud based service oriented architecture which provides user a web based interface for compilation, execution, code suggestion and storage of code.

Future Work

- Provide code optimization for c, c++, python
- Provide same set of service for other frequent set of programming languages
- Provide solution to error generated during compilation

References

- Ansari, Aamir Nizam, Siddharth Patil, Arundhati Navada, Aditya Peshave, and Venkatesh Borole. "Online C/C++ compiler using cloud computing." 2011 International Conference on Multimedia Technology, 2011. doi:10.1109/icmt.2011.6002124.
- Ghaleb, Taher Ahmed. "Toward open-source compilers in a cloud-based environment: the need and current challenges." 2015 International Conference on Open Source Software Computing (OSSCOM), 2015. doi:10.1109/osscom.2015.7372684.
- Datta, Arjun, and Arnab Kumar Paul. "Online compiler as a cloud service." 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, 2014. doi:10.1109/icaccct.2014.7019416.
- Xu X., Hu H., Hu N., Ying W. (2012) Cloud Task and Virtual Machine Allocation Strategy in Cloud Computing Environment. In: Lei J., Wang F.L., Li M., Luo Y. Network Computing and Information Security. Communications in Computer and Information Science, vol 345. Springer, Berlin, Heidelberg

Annexure

Find following code which get input code and return status, compilation outputs, execution output and code optimization suggestions.

```
from multiprocessing import Process, Queue
from suggestion import suggestJava
import os, signal, sys
import subprocess
import time
import threading
import psutil

def memory_usage(z, thread_q):
        """Memory usage of the current process in kilobytes."""
        status = None
        memory = 0
        result = {'peak': 0, 'rss': 0,'data': 0, 'stk': 0}
        try:
                # This will only work on systems with a /proc file system
                # (like Linux).
                status = open("/proc/"+str(z)+"/status")
                for line in status:
                        parts = line.split()
                        key = parts[0][2:-1].lower()
                        if key in result:
                                result[key] = int(parts[1])
        finally:
                if status is not None:
                        status.close()
        memory = result['data']+result['stk']
        thread_q.put(memory)

def getFileDetails(filename):
        filename = filename.rsplit('.',1)
        exe = filename[1]
        filename = filename[0].rsplit('/',1)
        file = filename[1]
        path = filename[0]
        return path, file, exe

def getCompilationStatus(compilation_args):
    compilation_result = {}
    popen = subprocess.Popen(compilation_args, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
    compilation_result['stdout'] = popen.stdout.read()
    compilation_result['stderr'] = popen.stderr.read()
    return compilation_result

def getOutputStatus(execution_args, q):
    thread_queue = Queue()
    execution_result = {}
    popen = subprocess.Popen(execution_args, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)

    thr = threading.Thread(target=memory_usage, args=[popen.pid, thread_queue])
    thr.start()
    ru = os.wait4(popen.pid, 0)[2]
    thr.join()

    execution_result['stdout'] = popen.stdout.read()
    execution_result['stderr'] = popen.stderr.read()
    execution_result['mem_usage'] = str(thread_queue.get()*0.001)+' MB'
```

```
    q.put(execution_result)


def timer(timeout):
        time.sleep(timeout)

def compile(path, fname, exe, lan, timeout):
        compilation_status = {'stdout':"",'stderr':"",'exe_time':""}
        final_status = {}

        execution_command = ""
        if exe=="c":
                args = ("gcc -o "+path+"/"+fname+" "+path+"/"+fname+"."+exe)
                compilation_status = getCompilationStatus(args)
                execution_command = (path+"/"+fname+" < "+path+"/"+fname+"_input.txt")
        elif exe=="java":
                args = ("javac "+path+"/"+fname+"."+exe)
                compilation_status = getCompilationStatus(args)
                execution_command = ("java -cp "+path+" "+fname+" < "+path+"/"+fname+"_input.txt")
        elif exe=="cpp":
                args = ("g++ -o "+path+"/"+fname+" "+path+"/"+fname+"."+exe)
                compilation_status = getCompilationStatus(args)
                execution_command = (path+"/"+fname+" < "+path+"/"+fname+"_input.txt")
        elif exe=="py" and lan=="py":
                execution_command = ("python "+path+"/"+fname+"."+exe+" < "+path+"/"+fname+"_input.txt")
        elif exe=="py" and lan=="py3":
                execution_command = ("python3 "+path+"/"+fname+"."+exe+" < "+path+"/"+fname+"_input.txt")
        else:
                final_status['valid_selection']="False"
                return final_status

        final_status['valid_selection']="True"
        if compilation_status['stderr']=="":
                final_status['compilation_status']="True"
                final_status['compilation_error']=""
                q = Queue()
                start = time.time()
                p1 = Process(target=getOutputStatus, args=(execution_command,q,))
                p2 = Process(target=timer, args=(timeout,))
                p1.start()
                p2.start()

                while p1.is_alive() and p2.is_alive():
                        continue

                stop = time.time()
                if p1.is_alive():
                        final_status['execution_status'] = "False"
                        final_status['execution_error'] = "Terminated due to timeout"
                        final_status['execution_output'] = ""
                        final_status['execution_time'] = "-"
                        os.kill(p1.pid,signal.SIGKILL)
                else:
                        checkoutput = q.get()
                        os.kill(p2.pid,signal.SIGKILL)

                        if checkoutput['stderr']=="":
                                final_status['execution_status'] = "True"
                                final_status['execution_error'] = ""
                                final_status['execution_output'] = checkoutput['stdout']
                                final_status['execution_time'] = stop - start
                                final_status['memory_usage'] = checkoutput['mem_usage']
                                final_status['suggestion'] = ""
                                if exe=="java":
                                        final_status['suggestion'] = suggestJava(path, fname, 'cc2')
                        else:
```

```
                                final_status['execution_status'] = "False"
                                final_status['execution_error'] = checkoutput['stderr']
        else:
                final_status['compilation_status']="False"
                final_status['compilation_error']=compilation_status['stderr']
        return final_status

def getCompile(filename, lan, timeout):
        path,file,exe = getFileDetails(filename)
        return compile(path, file, exe, lan, timeout)

if __name__=="__main__":
        filename="../compile/123/java/example.java"
        print getCompile(filename, 'java', 10)
```

Following code is service code return using flask App which act as a socket between application logic and input user

```
import shutil
from flask import Flask, request
from saveFile import saveFILE
from compileCode import getCompile

app = Flask(__name__)

@app.route('/compile', methods=["POST"])
def compile():
    timeout = 10
    jsonRes = request.get_json()
    code = jsonRes['getCode']
    lan = jsonRes['type']
    userID = jsonRes['userid']
    filename = jsonRes['filename']
    inp = jsonRes['input']

    code = str(code.decode('utf-8'))
    lan = str(lan.decode('utf-8'))
    userID = str(userID.decode('utf-8'))
    filename = str(filename.decode('utf-8'))
    inp = str(inp.decode('utf-8'))

    getpath = saveFILE(code, lan, userID, filename, inp)
    compilation_status = getCompile(getpath,lan, timeout)
    #shutil.rmtree('../compile/'+str(userID))
    return str(compilation_status)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5004, debug=True, threaded=True)
```

* We cannot copy paste all the codes related system considering length of document.