Cloud based Unified IDE for.NET Languages

By

NC Nida Khan
NC Haziqa Sajid
NC Izza Rashid Dar

Submitted to the Faculty of Computer Software Engineering,
National University of Sciences and Technology, Islamabad in partial fulfillment for the requirements of a B.E. Degree in Computer Software Engineering
MAY 2014

CERTIFICATE OF CORRECTNESS AND APPROVAL

Certified that work contained in this thesis “Cloud based Unified IDE for .NET Languages” carried out by Nida Khan, Haziqa Sajid and Izza Rashid Dar, under the supervision of Dr. Sarmad Sadik for partial fulfillment of Degree of Bachelor of Software Engineering is correct and approved.

Approved by

_____

(Supervisor Name)

_____ Department

MCS

Dated: _____

ABSTRACT

Cloud based Unified IDE for .NET Languages

Cloud based Unified IDE for .NET Languages is an embedded web application that provides an environment for online compilation and execution of codes. It supports multi-languages which include C# and Visual Basic. Users will perform remote online compilation and execution of their codes without requiring intensive CPU and memory usage as Unified IDE is provided on the Cloud. By combining Cloud Computing technology, our web application will facilitate the remote users and remove the requirement for powerful systems and provide portability through provision of SaaS.

The objective is to provide remote code compilation and execution facility especially for the academic community in Pakistan. The students who don’t have Visual studio.NET, VB compilers on their personal computers, will open web browser and access the unified framework web service. They will write their C# and VB codes, compile and execute it online. The computation will be done at cloud servers and interface will also be available on iOS based mobile devices. The system shall fulfill the legal requirement by working under multi-user license from relevant vendors.

In the second phase, the prime objective is to create a middleware for a generic computation facility accessible to all kind of users who want to run programs on their smart phones and tablets with complex algorithms which require higher processing and memory resources. The examples include finding large prime numbers, execution of computation intensive and interactive codes on smart phones and tablets. The front end will be available on smart phones and tablets through which users will give the command and initiate the application. The middleware will execute the program at the back end cloud servers through calling procedures and passing necessary arguments. The results shall be displayed on mobile devices with an illusion of local execution and maintain a suitable transparency level. The project will also enhance the potential and capability of light weight smart phones and tablets with more convenience to users

DECLARATION

We very solemnly declare that the work presented herewith is the result of sole effort of our group, comprising of Nida Khan, Haziqa Sajid and Izza Rashid Dar, and is free of any kind of plagiarism in part or whole. We also declare that the dissertation has never been submitted

previously in part or whole in support of another award or qualification either at this institution or elsewhere.

DEDICATION

To our respected teachers whose kind guidance and unfailing support made this mammoth task easy for us and to our very dear parents whose unceasing prayers gave us strength and courage to complete the work of this magnitude.

ACKNOWLEDGEMENT

We are, very humbly, grateful to Almighty Allah for bestowing us with the strength and resolve to undertake and complete the project.

We owe a special debt of gratitude to our supervisor, Dr. Sarmad Sadik for the continuous supervision, motivation and support provided to us and for their continuous and valuable suggestions, guidance, and instructions from time to time right through the project.

We would also like to acknowledge and thank the beginning faculty members of the department of Computer Software Engineering for their continuous support. It was their help and guidance which helped us complete the project in due time.

Mr. Bilal Bashir (MIS Cell) Military College of Signals, who despite his busy routine as system administrator generously employed his intellectual expertise and extended his kind support to help us in networking domain. Sir we thank you for your unparalleled kindness. We would also like to acknowledge and thank the beginning faculty members of the department of Computer Software Engineering for their continuous support. It was their help and guidance which helped us complete the project in due time.

We are also thankful to our class mates, Ma’am Sehrish (BESE 14) and our family members for their support throughout the project in every possible way.

Contents Chapter 1

List of figures

List of Tables

Chapter 1 1.1 Introduction Cloud computing is rapidly gaining momentum in information and communication technology especially in the domain of distributed computing. Its potential will be truly utilized if cloud computation is mapped with mobile devices like Smart phones and tablets which have low memory and processing power as compared to demand of modern computation intensive applications. The main goal is to provide remote code compilation and execution facility for users through online IDE who want to run programs with complex algorithms which require higher processing and memory resources through web service on their mobile devices. **1.1.1. Document Conventions** The format of the document is Simple. Bold face and indentation is used on general topics and on specific points of interest. **1.2 Motivation** Students related to engineering background especially in Pakistan‟s academic community need computation intensive machines in order to truly analyze their programs and algorithms written in diverse programming languages. There is a need to provide programming language compilation and execution as a service through web browsers for those personal machines which do not have the development environment (IDE) installed especially smart phones and tablets. Secondly, the general audience who want to run computation intensive applications on mobile devices can use the proposed framework which will run their programs on cloud and provide them results on their light weight hand held devices. Era is moving towards thin client. With increase in tele-density, more people are using mobile devices each year with same proportion of increase in smart phone and tablets users. However, they are low in processing power and memory requirements. The objective of this project is to present a middleware design which supports the execution of mobile applications with a computational support of cloud computing. These back end cloud servers will be mapped with middleware on smart phones and provide results after execution of main algorithm.
10

1.3 Project Scope Our product “Cloud based Unified IDE for .NET” is an embedded web application that provides an environment for online compilation and execution of codes. It supports multi-languages which include C# and Visual Basic. Users will perform remote online compilation and execution of their codes without requiring intensive CPU and memory usage as Unified IDE is provided on the Cloud. By combining Cloud Computing technology, our web application will facilitate the remote users and remove the requirement for powerful systems and provide portability through provision of SaaS. **1.3.1 Project Vision** For Academic Institutions with a focus on NUST What To provide remote online compilation and execution of codes in specified language The Cloud based Unified IDE for .NET Languages Is A Web-based application That Provides online compilation and execution of C# and VB codes Unlike Manual installation of softwares on computer and then compilation and execution Our Product Ensures remote compilation and execution of codes which is accessible through web browser **Table 1: Project Vision 1.3.2 Project Objective** The objective of this project is to propose a middleware design which supports the execution of mobile applications with a computational support of

cloud computing. These back end cloud servers will be mapped with middleware on smart phones and provide results after execution of main algorithm. In case of scientific computation, it will not consume as much time but frequent compiling codes rendering will take more time. **1.4 Deliverables** 1.4.1. Web Application for Mobile as well as Desktop users 1.4.3. Documentation/User Manual

11

Chapter 2 2 Literature Review 2.1 Introduction Our project is an online compiler which helps to reduce the problems of portability and storage space by making use of the concept of cloud computing. It mainly deals with the development of Cloud Based Unified Integrated Development Environment for the .NET (C# and Visual Basic) languages to code, compile, run, test and debug using the browser based IDE through the Internet and a web browser. The ability to use different compilers allows a programmer to pick up the fastest or the most convenient tool to compile the code and remove the errors. Moreover, a web-based application can be used remotely throughout any network connection and it is platform independent. The problem of several compiler installations can be eliminated. Also error/output is stored in convenient way. In the today‟s world everything is online so we are creating software for the today‟s faster world called compiler as service over the cloud. The core tasks involved in our project include: **Step 1**: Language selection by the user. **Step 2**: After selecting a specific language, user shall be able to provide code in text box or uploads a code file and then presses submit button. **Step 3**: Submitted code will be compiled and executed and output will be shown to user in textbox or file. Following is the diagrammatic illustration of all these three steps. **Figure 1: Online code compilation and execution [1]. Selection of Language [2]. Insertion and submission of code [3]. Compiled and executed output of the submitted code**

12

2.2 Problem Domain Our project is centralized compiler which helps to reduce the problems of time, cost, processing power and storage space by using the concept of cloud computing. Also, the trouble of installing the compiler on each computer is avoided. The main reason for creating the project is to provide a centralized compiling scheme. The other major advantage that this system will have over the others is that it will make the users system lightweight i.e. there will be no need to maintain separate compilers SDK‟s at the client-side. Thus, for educational institutions this will prove to be highly efficient. **2.3 Related Work** Online education websites like **Coursera** [2] provide an online execution interface for specific programming language related to a particular course like CUDA for Parallel Programming hosted on Amazon EC2 cloud service. Google App Engine [3] provides support for the Java, Python and GO programming interfaces while Windows Azure [4] provides a platform for hosting and executing Java, Python, PHP and .NET applications. In **first phase** of our application we shall provide a unified interface for compilation, execution and testing of codes in Java and .NET. However, in **second phase** of our application, we shall further extend the hosting facility and allow computation intensive applications to run from mobile devices and invoking back end cloud platform for computation with higher degree of transparency and efficiency. In the related research [5], a **Hadoop** derived platform called Hyrax has been developed. This platform facilitates the cloud computing on android smart phones. Through hyrax client applications can easily use the data and perform different task on smart phone‟s heterogeneous network. It also facilitates the client applications to access and utilize distributed resources in abstract way. The design and implementation of Hyrax is described, including experiences in porting Hadoop to the Android platform and the design of mobile specific customizations. The scalability of Hyrax is evaluated experimentally

and compared to that of Hadoop. Although the performance of Hyrax is poor for CPU-bound tasks, it is shown to tolerate node-departure and offer reasonable performance in data sharing. A distributed multimedia search and sharing application is implemented to qualitatively evaluate Hyrax from an application development perspective.
13
In another approach [6], researchers have revisited general concept of offloading computation to remote servers by focusing on a largely unsolved problem: how to automatically determine whether and when smartphone applications will benefit from offloading? This is an especially relevant and challenging problem today as (1) modern mobile applications tend to have complex interactions with users and advanced capabilities (e.g., GPS and camera) and hence cannot be offloaded as a whole; (2) whether an application component, e.g., a method call, will benefit from offloading depends on its execution time on Smartphone and the size of state to be shipped, which in turn depend on the input parameters. A design and implementation has been presented of XRay, an event-tracing-based profiling tool that identifies methods in a Smartphone application that can be offloaded to a remote server, and determines whether and when offloading the methods will benefit the application. Their experiments of applying XRay to a set of smartphone applications show that after a small number of offline profiling runs,XRay can automatically generate offloading decision logic for each remotable method that makes correct offloading decisions in future online executions of these applications under a priori unknown input parameters and network conditions. The problem of extending the battery lifetime for a portable computer by off loading its computation to a server [7] has been presented in domain of computational off load. Depending on the inputs, computation time for different instances of a program can vary significantly and they are often difficult to predict. Different from previous studies on computation off loading, this approach does not require estimating the computation time before the execution. The program was executed initially on the portable client with a timeout. If the computation is not completed after the timeout, it is off loaded to the server. Firstly, the timeout was set to be the minimum computation time that can benefit from offloading. Later on, the online statistics of the computation time were also considered and the optimal timeout was identified. Similarly in another approach [8], a mobile device based virtual cloud computing framework has been provided but it has limitations in terms of application processing requirements and availability. With the advancement in HTML5 [9], more advanced mobile cloud computing platforms are being developed [10] to handle the issues of computation intensive applications on mobile devices. There are various online services [11][12][13] which provide remote compilation for Java, HTML, PHP and various .NET based programming languages.
14
Our work will present a unified cloud based IDE framework for multiple languages including .NET (C# & Visual Basic) especially for academic community in Pakistan as well as support for execution of computation intensive programs on lightweight based heterogeneous devices including smartphones and tablets. Customized View for smartphone and tablet users is the innovation part in our proposed project. Uptil now online IDE`s are provided for desktop users. Our project will provide the users with detailed view for desktop users but along with that customized view for smartphones and tablets is also being provided for processing of computation intensive programs requiring extensive processing and memory power. The diagrammatic illustration of Customized view for Smartphone and tablet is given below: **Figure 2: Code Insertion Figure 3: Compiled and executed result**

2.4 Shortcomings/issues

- One

challenge is that in case of scientific computation, it will not consume lot of time but frequent compiling, complex algorithmic and highly interactive codes will take more time. • The contents of our web application will be in English language only. • Our online web application will be unavailable in case of maintenance and testing issues of server. No backup server configuration is provided.
15
2.5 Proposed Project Our system is an embedded web application that provides a **Cloud based Unified IDE for .NET languages**. It is a platform for compilation and execution of codes of C# and Visual Basic without need of high resources at the user end. The core features of our project include: • Remote Online compilation • Remote Online execution • Detailed view for Desktop users • Customized view for Smartphone users **2.6 Deliverables** The deliverables of our project are as follows: **Deliverable Name Deliverable Description** Software Requirements Specification(SRS) Document Complete Description of WHAT system will do, who will use it. Detailed description of functional and non-functional requirements and system features Analysis Document Detailed requirement analysis and analysis models are included Design Document Complete description of How the system will do. Design models are included. Code Complete code with the APIs Testing Document Whole system is tested corresponding to the specifications. System is tested at all levels of Software Development Life Cycle (SDLC) Complete System Complete working system(embedded web application) **Table 2: Deliverables**
16
2.7 Technological Requirements Our project requires following software and hardware requirements. **2.8 Software Requirements** The software(s) required for the implementation of our project includes: **2.8.1 Operating System 2.8.1.1 For End User** • Microsoft Windows • Linux/Unix • iOS **2.8.1.2 For System** • Microsoft Windows Azure **2.8.2 Network System** Protocols for communication: • TCP/IP • HTTP • HTTPS **2.8.3 Programming Interface** Programming interfaces for project are: • Microsoft ISAPI dll libraries • Visual Studio 2012 **2.9 Hardware Requirements** The Hardware required for the implementation of our project include: • Smart Phone and tablets having iOS as its operating system. • Microsoft Windows Azure Server on which our system will be deployed. • The client machine and internet with which the user will be able to interact with our web application. • Windows Azure Database server to store the logs of users.
17
Chapter 3 3 Overall Description 3.1 Product Perspective The overall functioning of the system is to compile and execute the users‟ code without high resource consumption at the client end. The system is designed for academic niche and will be deployed on the MS Windows Azure Server for compilation and execution of codes. **Figure 4: System Overview Figure 5: Middleware**
18
3.2 Product Functions The major functions that will be provided by our product are: • Compilation of code • Execution of code • Designing of interfaces at client and server ends (middleware) • The backend controller of the system will assign the request to appropriate compiler module for computation • Designing of Output view • Generation of results • Configuration of Output view (i.e. Detailed and customized view) • Maintenance of the database of users • Generation of user profile reports maintaining history logs **3.3 User Classes**

and Characteristics The system is being developed for different classes of the users which are: • **End Users** o Desktop as well as Smartphone users o End users are those who send compilation and execution requests • **Administrator** o Administrator makes the application accessible/open to the users and makes configuration and updation at the middleware and server sides. • **Database Administrator** o User‟s data will be stored in database and managed by database administrator.
19
3.4 Operating Environment Microsoft Platform • Browser: HTML aware **Particulars Client System Server System** OS Windows XP and Higher Windows AzureServer Processor Pentium 4 , 2 GHz i5 processor , 2.5 GHz Hard Disk 100GB 500 GB RAM 1GB (recommended) 4GB RAM (minimum) 8 GB (recommended) **Table 3: Operating Environment 3.5 Design and Implementation Constraints** The system will be designed and developed under the following constraints: • **Text Area space:** Lines of code are limited according to text area size
• **Languages:** The system will support two languages C# and VB • **Server:** The system will operate on Microsoft Windows Azure Server • **Compilation Time:** The system will consume time up to 15 seconds in case of frequent compiling, highly interactive and complex algorithmic codes **3.6 User Documentation** Following user documents will be provided with the system on deployment: • User manual including all the details of the working of web application and step by step demonstration will be provided along with the system • Help documents will also be part of the deliverables.
20
• Online help feature will highlight the system features, working, procedures and solutions to the problems if encountered (sample codes will also be provided). **3.7 Assumptions and Dependencies** Compilation and execution of the code depends on the compilation tools and softwares installed at the server end. Our system that is a middle ware between client and server receives requests from the client and interacts with the server for executing the results of those requests. Middleware and server remain active for request acceptance and executions. • Internet connection availability • Active browser availability • Windows server status is „up‟ • Although basic password authentication and security mechanisms will be used to protect the system from unauthorized access; functionality such as email notifications are also assumed to be applied by the University network team • Server status should be „up‟ for successful delivery of services; in the event of the server failing due to an error might result in the system becoming temporarily unavailable
21
Chapter 4 4 Software Requirements Specification 4.1 System Features Use Case Diagram Figure 6: UseCase Diagram of whole System
22
4.1.1 Login 1. Description and Priority This feature allows the authentication of the registered users as well as registration of new users. User provides username and password which is sent to the database server for authentication. **Priority** = 1 (minimum priority) **2. Stimulus/Response Sequences • Stimulus:** The user will click the “Sign in” option • **Response:** A window will appear asking for username and password • **Stimulus:** The user will provide username and password and click “Sign in” button • **Response:** Username and password shall be forwarded to server for authentication**.** In case of successful authentication, user shall be logged in. In case of

unsuccessful authentication, user shall return back to login page and will be asked to enter username and password again. **3. Functional Requirements** The functional requirements for this feature are: • **REQ-1:**User shall provide username • **REQ-2:**User shall provide password • **REQ-3:**Server shall monitor the users requests • **REQ-4:**Username and password shall be stored in Database Server **4.1.2 Language Selection 1. Description and Priority** This feature provides the user with language selection capability as per the code which is to be compiled and executed. **Priority** = 5 (maximum priority)
23
2. Stimulus/Response Sequences • Stimulus: The user will select the language from the drop down list • **Response:** The language will be selected and appear in menu **3. Functional Requirements** The functional requirements for this feature are: • **REQ-1:**Server shall provide compilation and execution tools and APIs • **REQ-2:**The history logs shall be maintained in database for registered users **4.1.3 Remote Online Compilation and Execution 1. Description and Priority** This feature allows the online compilation and execution of code provided by the user. The code is passed on to computation server through middleware for compilation and execution. **Priority** = 5 (maximum priority) **2. Stimulus/Response Sequences • Stimulus:** The user uploads the code file or enters directly the code in interface text area and clicks the "Submit Code" button • **Response:** The output results or errors (if any) of the code will be displayed in a text area **3. Functional Requirements** The functional requirements for this subsystem are: • **REQ-1:**User shall upload a code file or directly insert code in text area • **REQ-2:** User shall select the relevant programming Language • **REQ-3:**Server shall provide compilation and execution tools and APIs • **REQ-4:**the history logs shall be maintained in database for registered users
24
4.1.4 Customized/Reduced view for mobile users 1. Description and Priority This feature allows redirection of mobile users to a mobile-enabled web version (custom web application) that is customized/reduced view. **Priority** = 4 **2. Stimulus/Response Sequences • Stimulus:** User will browse for the web application through Smartphone or mobile device • **Response:** A customized view of the web application will appear **3. Functional Requirements** Functional requirements for this subsystem are: • **REQ-1:** The Smartphone or mobile device shall have internet availability • **REQ-2:** The user shall request for the web application **4.1.5 Detailed view for Desktop users 1. Description and Priority** This feature directs desktop users to a full web version of web application that is detailed view. **Priority** = 5 **2. Stimulus/Response Sequences • Stimulus:** User will browse for the web application • **Response:** A detailed view of the web application will appear **3. Functional Requirements** Functional requirements for this subsystem are: • **REQ-1:** The desktop shall have internet availability • **REQ-2:** The user shall request for the web application
25
4.1.6 Online Help 1. Description and Priority This feature provides the user with online help. Facilitate users on how to use the application with ease and other helping tutorials to solve problems if encountered. Sample codes will also be provided. **Priority** = 4 **2. Stimulus/Response Sequences • Stimulus:** User will click the online help option • **Response:** Online help interface will appear • **Stimulus:** User will specify the type of problem or selects from the help categories • **Response:** Respective solution will appear on the online help interface **3. Functional Requirements** Functional requirements for this subsystem are: • **REQ-**

1: Online help should be available for user‟s assistance **4.2 Non Functional Requirements** **4.2.1 Performance Requirements** Performance requirements are given as follow: • They system should be able to handle 100 users for execution simultaneously. • The system should compile and execute the code for two languages which are C# and Visual Basic. • System is being developed in distributed environment; language compilers for two different languages will be working in parallel.
26
• Remote and onsite support on 20 hours bases should be provided, so system work 20 hours with proper functionality. Remaining hours in case of maintenance and testing of computation server. • Response time of our system is 5s. **4.2.2 Security Requirements** The security involved in this system is the authentication of the database administrator to log into the system. The system will keep the log of access in the database and log will be read-once so no one can alter the log. It will protect its resources against malicious applications. **4.2.3 Software Quality Attributes 4.2.3.1 Correctness** The system produces sufficiently correct results as desired. The correctness of the system under discussion will be 99% for normal computation codes compilation and execution requests. The system will consume 20 to 30 seconds in case of frequent compiling, highly interactive and complex algorithmic codes causing delay on its performance. **4.2.3.2 Flexible** Since software is flexible in nature, thus the system can incorporate up to 90% changes that are relevant to system. For new changes the flexibility rate will be dynamic from 80% to 90%. i.e. enhancing the web application functionality by adding more languages and making it compatible with OS like android. **4.2.3.3 Maintainability** After the release of the system, the system will be maintained after 6 months for bugs or system performance issues. These problems will be corrected at its best by the developer team. The system is able to correct the errors after its release up to 55%. **4.2.3.4 Availability** The system will be available to the user 22 hours a day if there is no power and internet disruption. Remaining hours are for maintenance and testing of computation and database servers per week.
27
4.2.3.5 Robustness After the failure occurs at server end, the system will be able to recover within 2 hours. **4.2.3.6 Usability** The web application will be easy to use after training/demo to the beginners for 3 to 4 hours. Technical personnel related to software field will require 1-2 hour training. **4.2.3.7 Portability** Our web application is based on PaaS which doesn't require installation of softwares at each client side. Since it facilitates the remote users and removes the requirement for powerful systems, so it provides portability up to 95%.
28
Chapter 5 5 System Design Specifications 5.1 Work Breakdown Structure **Figure 7: Work Break Down Structure**
29
5.2 Architecture Diagram **5.2.1 System Block Diagram Figure 8: System Overview** The **figure 8** shows the main scenario in which a user/student can enter his code related to programming language (C#, VB.NET) and system will pass it on through middleware for execution on computation server. This whole architecture will be acting as a cloud environment which will be communicating with user through HTTP over the internet acting as a unified service. The request from user will be received by interface of middleware and transferred to compiling tools through controller module. The controller will identify the particular programming language compiler and transfer the code for compiling and execution. If the code is
30

already compiled like in binary form then it will be transferred to Executor for result/report generation and transfer it back to user through Output View creator module. **5.2.2 High Level Design Diagram (Modules Identification)** **Figure 9: High Level Design** **5.2.3 Abstract Architecture** **Figure 10: Abstract Architecture**

31

The abstract architecture is discussed as shown in figure 10. The user shall interact with the interface handler at front end of application for input. The input can be of two types, (1) User enters directly the code in interface text box and selects the relevant programming language. (2) As a second option, user can upload the code file into the system for compilation and execution. After receiving input from interface handler component, the request handler will generate a customized request encoding the device details along with code data. This module will send the specific request to Request Executor residing on cloud service as back end of application. The protocol for transportation will be SOAP (Simple Object Access Protocol) based on HTTP which is standard approach used in web services for communication and collaboration. The request executor will unpack the incoming request and invoke the relevant third party component as specified by user in selection of programming language. The code shall be compiled and in case of any error, user is informed about the current status. In case there is no error in code, the program will be executed on underlying cloud servers and the results will be sent back to user device. The request handler at device will process the response and upgrade the user interface screen with the output received after successful program execution. The equipment required will be Microsoft Server 2012 computation server which provides an integrated support for cloud computing. The proposed middleware will be residing on this cloud server and it will be accessible through Tablets/Smartphones as well as traditional desktops/laptops. Now days Microsoft Windows Azure is a popular technology in cloud computing, So we will be using public cloud for deployment purposes. Microsoft Azure Websites is a cloud computing based platform for hosting Websites, created and operated by Microsoft. Microsoft Azure Websites is a platform as a service (PaaS) which allows publishing Websites running on multiple frameworks and written in different programming languages (.NET, node.js, php, Python and Java), including Microsoft proprietary ones and 3rd party ones. Microsoft Azure is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed datacenters.

32

5.3 Interaction among modules (Abstract View) **Figure 11: Interaction amongst modules (abstract view)**

33

5.4 System Architecture This section provides a detailed and comprehensive architectural overview of the system. **5.5 MVC Architectural Pattern is used for this project** Architecture of Cloud Based Unified IDE for .NET Languages can be modeled using **Model-View-Controller**. Interface of the system is distinct from the application logic. Interface forms the View of the architecture. Model has the application data .Controller controls the coordination between the view and the model. There is no direct communication between model and the view all the communication is directed using the controller. The system is divided into modules as per the main functionality of the system. These functionalities can be used separately as off-the shelf components .Interface of this system has no application logic embedded in it so the system architecture can easily be made using Model View Controller approach. **Figure 12: Model View Controller** 1. **Model** in this system is the **Executor**, which further includes C# and VB.NET

modules which are encapsulating the application logic. These classes process the requests made by the user through the controller. After processing Model gives the retrieved compilation and Execution results back to the controller in unrendered form. The controller then passes on the results to the view to present them in proper view to the user. 2. **View** in this system is **View** class containing displayResult() and setView() function which ensures the availability and display of detailed / customized view to user as per his requirement. The result/report generator class displays the execution results. These functions handle the processing of interface but no application logic is involved in it.
34
3. **Controller** in this system is **RequestHandler** which includes request_login() , request_file() from user, submit_code() sends code for compilation, invoke_compile() and invoke_execute() are the main functions which invoke compilation and execution. Controller only invokes the functions of View and Model but do no processing. **Figure 13: MVC of Cloud Based Unified IDE**

5.6 User Interface Design Our work will present a unified cloud based IDE framework for multiple languages including .NET (C# & Visual Basic) especially for academic community in Pakistan as well as support for execution of computation intensive programs on lightweight based heterogeneous devices including smartphones and tablets. Customized View for smartphone and tablet users is the innovation part in our proposed project. Until now online IDE`s are provided for desktop users. Our project will provide the users with detailed view for desktop users but along with that customized view for smartphones and tablets
35
is also being provided for processing of computation intensive programs requiring extensive processing and memory power. When a user clicks on “Login” option, the following login interface will appear which is similar in both customized and desktop view. **Figure 14: Login Interface 5.6.1 Customized View** After successful login into the system, the diagrammatic illustration of Customized view for Smartphone and tablet is as illustrated in figure 15. **Figure 15: Code insertion** The user will insert the code and request for code compilation and execution. The system will compile and execute the code and display the result as shown in the figure 16. **Figure 16: Compiled and executed result**
36
As our web application is meant for authorized users only, so sign up is must and when user will browse for our web application he has to login first if he is registered one. In case if the user is not the registered one, he has to sign up first for compilation and execution of codes. **5.6.2 Detailed View** After successful login the user will perform the following steps: **Step 1:** The user will select the language whose code is to be compiled. Our web application supports two types of languages which are C# and VB. The interface will appear to the user as: **Figure 17: Language Selection Step 2:** The input can be of two types: • User enters directly the code in interface text area • As a second option, user can upload the code file into the system for compilation and execution. **Figure 18: Code Insertion**
37
Step 3: The user will submit the code by clicking the “Submit Code” button. System will pass it on through middleware for execution on computation server. The output results after execution will be then displayed to the client on the new web page in text area. In case of any error, the errors will be displayed in text area. **Figure 19: Output after compilation and execution 5.7 Design Details 5.7.1 Design Patterns Name:** Controller Pattern **Problem:** What first object beyond the UI layer receives and coordinates ("controls") a system operation (Code compilation

and execution)? **Solution:** Whenever an input system event is generated by pressing the "Submit" Button which is in the **View** of the system, request for the processing will be made to the **Controller** which in this scenario is RequestHandler - a usecase controller class. RequestHandler will then forward this request to the Executor class which is **Model** in this case, will then process the request using its functions and data that are compilation APIs. After processing Model will return the result to View class which will then select the appropriate view out of customized and detail view. Processing in the Model includes compilation and execution of the code By using this Controller pattern interface logic will be separated from the business logic. It will increase the potential for reuse and pluggable interfaces. It provides low coupling, high cohesion and high reusability.
38
Figure 20: Controller Pattern Name: Facade Pattern **Problem:** There may be undesirable coupling to many things in the subsystem, or the implementation of the subsystem may change. What to do? **Solution:** In the UI Layer, for the compilation and execution of two different languages (c# and vb) codes, we will define a single entry point for these languages in View by providing the drop down menu. The drop down menu will identify the language track and presents a single unified interface and will be responsible for collaborating with the subsystem components. At the Domain Layer, RequestHandler class will select the module according to language selected. By using this Façade pattern,clients will be shielded from subsystem components, thereby reducing the number of objects that clients deal with and making the subsystem easier to use. Also, it will promote weak coupling between the subsystem and its clients.
39
Figure 21: Facade Pattern 5.7.2 Class Diagram Cloud based Unified IDE comprises of the following classes: • User • RequestHandler • Executor • CSharp • Visual Basic • View • Desktop View • Customized View • Result and Report Generator Class Diagram is given on next page:
40
+login()+signUp()+getHelp()+viewResults()+submitCode()+UploadCodeFile()+logout()+selectLanguage()-Username : string-Password : string-Email : string**User**+get_username()+get_password()+get_email()+request_login()+request_signUp()+request_file()+request_code()+invoke_compile()+invoke_execute()+select_view()+display_compilation_result()+invoke_validate()+display_execution_result()-filename : string-code : string-result : string**Request Handler**+compile_code()+execute_code()+compile_file()+execute_file()+submit_result()-filename : string-language : string**Executor**+compile_java_code()+compile_java_file()+execute_java_code()+execute_java_file()-**Java**+compile_csharp_code()+compile_csharp_file()+execute_csharp_code()+execute_csharp_file()**CSharp**+compile_visualBasic_code()+compile_visualBasic_file()+execute_visualBasic_code()+execute_visualBasic_file()**Visual Basic**+get_result()+set_result()-result : string**Results and Report Generator**+setView()+displayResult()-generated_results : string**ViewDetailed ViewCustomized View**1..*111111111+validate_user()+insert_info()+update_info()+delete_info()-username : string-password : string-email : string**Database**11 **Figure 22: Class Diagram**
41
5.7.2.1 System Classes Description Name Description User This class contains all the functions related to the functionality of users i.e. login() functions allows user to enter username and password and get logged into the system. For unregistered users signUp functionality is

provided , user can get registered by entering his username, password and email id. If user is having problem understanding or navigating between pages getHelp() function helps by giving user demo on how to use the web application. selectLanguage and submitCode allow user to select one language (c#, vb) and submit it to the computation server. UploadCodeFile() allows user to upload code file and results are diaplayed through viewResults(). finally logout() allows user to log out of the system. **RequestHandler** This class servers as a controller. It requests login / signup through requestlogin() and requestsignup() and gets data from user through getusername(),getpassword(),getemail() and then calls invoke_validate() function of database in order to validate the user. It requests file and code through requestfile() and requestcode() and then invokecompile() and invokeexecute(). After compilation and execution is done it selects_view() and displays result through display_compilation_result() and display_execution_result(). **Database** This class performs all the main functions of database as validation of user`s information through Validate_user(), insertion of user`s information through insert_info() and updation and deletion from update_info() and delete_info(). **Executor** This class performs the main functions of compilation and execution. compile_code() and execute_code() if code is provided by user in textarea. compile_file() and execute_file() if code is provided in file. Result is submitted through submit_result().

CSharp This class is inherited from executor class. This class contains the functionality of compilation and execution of csharp code. compile_ csharp _code() and execute_ csharp _code() if csharp code is provided by user in textarea. compile_ csharp _file() and execute_ csharp _file() if csharp code file is provided. **Visual Basic** This class is inherited from executor class. This class contains the functionality of compilation and execution of visual basic code. compile_visualBasic _code() and execute_ visualBasic _code() if visual basic code is provided by user in textarea. compile_ visualBasic _file() and execute_ visualBasic _file() if visualBasic code file is provided. **View** This class handles both the views of the web application through setView() i.e. detailed (desktop) view and customized view(for smartphone users). This class also displays compilation and execution results to the user through displayResult(). **Detailed View** This class is inherited from View class . it is responsible for the generation of detailed view to desktop users and display results to users through setView() and displayResult(). **Customized View** This class is inherited from View class . it is responsible for the generation of customized view to smartphone users and display of results to users through setView() and displayResult(). **Results & Report Generator** This class is responsible for the generation of compilation and execution results and reports. The two function used for this purpose are get_result() which gets result from executor (compiled file) and set_result() changes it into text file. **Table 4: System Class Description**

5.7.3 Database Design 5.7.3.1 Entity Relationship Diagram Online unified IDE requires user to log into the system for compilation and execution of code. User credentials are verified and sign up facility is provided for unregistered users. The user selects a language and submits the code. The system identifies the language track on the basis of language selection and uses appropriate compilation tools for compilation and execution of codes. Then the system displays results. EDR of the system is given below: **Figure 23: ERD of Cloud Based IDE 5.7.3.2 Entities & Attributes 1.** User • User_id • Uname • Password **2.** Database

3. Code • LOC • FileType (.txt file with C# or VB code) **4.** Language • Type (Programming Language) • Name (CSharp and VisualBasic) **5.** System (Unified IDE for .Net) **6.** Results (Code Output or Errors if any) **5.8 UML Diagram 5.8.1 Use Case Diagram Figure 24: Use case Diagram**
45
5.8.2 Use case Specification 5.8.2.1 Actors • User • Administrator **5.8.2.2 Use Cases** • Login • Authenticate Users • Register • Display Tutorials • Identify Language Track • Select C# • Select VB • Submit Code • Compile Code • Execute Code • View Generated Results and Reports • View Detailed • View Customized • Maintain History Logs The key uses cases depict the main functionality of our system that is Login, Register, Compile Code and Execute Code. **1. Login Use Case Specification Use Case ID** 1 **Use Case Name** Login **Actor(s)** User **Description** This feature allows the authentication of the registered users. User provides username and password which is sent to the
46
database server for authentication. **Pre-Conditions** 1. Web application must be opened. 2. User must be registered. **Post-Conditions** If the use case is successful the actor has the choice to enjoy any of the above mentioned features of application. **Normal Flow (Primary Scenario)** The use case starts when an actor opens the web application and request for login in to the application: 1. The user will click the “login” button. 2. A login page will appear asking for username and password. 3. The user will provide username and password and click “submit” button. 4. Username and password shall be forwarded to server for authentication**. Alternative Flow(s) a. Missing Functionality** If in step 1 of the basic flow i.e. pressing the login button does not open the login page the use case ends with a failure condition **b. Wrong Functionality** If the wrong username and password is provided then user shall return back to login page and will be asked to enter username and password again. Also the user will be given an option to register to the website. **c. Successful Condition** If use case was successful, user will be login successfully. **d. Failure Condition** If not, user will not be login. **Table 5: Use Case Login**
47
2. Register Use Case Specification Use Case ID 2 **Use Case Name** Register **Actor(s)** User **Description** This feature allows the registration of new users. User provides username, password and email which is sent to the database server for registration. **Pre-Conditions** Web application must be opened. **Post-Conditions** If the use case is successful the actor has the choice to access any of the above mentioned features of application. **Normal Flow (Primary Scenario)** The use case starts when an actor opens the web application and request for sign up for the application: 1. The user will click the “Register” option 2. A registration page will appear asking for username, password and email. 3. The user will provide username, password and email id and click “Register” button 4. Username, password and email shall be forwarded to server for registration**. Alternative Flow(s) a. Missing Functionality** If in step 1 of the basic flow i.e. pressing the register button does not open the register page the use case ends with a failure condition **b. Wrong Functionality** In case of unsuccessful registration, user shall return back to sign up page and will be asked to enter username, password
48
and email again. **c. Successful Condition** In case of successful registration, user shall be registered **d. Failure Condition** If not, user will not be registered. **Table 6: Use Case Register 3. Compilation Use Case Specification Use Case ID** 3 **Use Case Name** Compilation **Actor(s)** User **Description** This feature allows the online compilation of code provided by the user. The

code is passed on to computation server through middleware for compilation. **Pre-Conditions** 1. Web application must be opened. 2. User must be logged in. **Post-Conditions** If the use case is successful the actor can further access the execution code facility. **Normal Flow (Primary Scenario)** The use case starts when an actor opens the web application and login into the application: 1. The user uploads the code file or enters directly the code in input text area and clicks the “Submit Code” button. 2. The code is passed on to computation server through middleware for compilation. 3. The output results or errors (if any) of the code will be displayed in output text area.
49
Alternative Flow(s) a. Missing Functionality If file is not uploaded successfully then the user will be asked to upload again or enter code in text area. **b. Wrong Functionality** If the language of the uploaded file or input code is other than the two languages specified then the use case ends with a failure condition. Also if the selected language and input code language mismatch again failure occurs. **c. Successful Condition** If the input code is submitted successfully to the computation server, the code will be compiled. **d. Failure Condition** If there is an error in code submission to the computation server, the use case fails. **Table 7: Use Case Compilation 4. Execution Use Case Specification Use Case ID** 4 **Use Case Name** Execution **Actor(s)** User **Description** This feature allows the online execution of code provided by the user. The code is passed on to computation server through middleware for execution. **Pre-Conditions** 1. Web application must be opened. 2. User shall be logged in
50
3. The code should be compiled **Post-Conditions** If the use case is successful the user can access the compilation and execution result of the code. **Normal Flow (Primary Scenario)** The use case starts when an actor opens the web application and login into the application: 1. The user uploads the code file or enters directly the code in interface text area and clicks the “Submit Code” button 2. The code is passed on to computation server through middleware for execution. 3. The output results or errors (if any) of the code will be displayed in a text area. **Alternative Flow(s) a. Missing Functionality** If file is not uploaded successfully then the user will be asked to upload again or enter code in text area. **b. Wrong Functionality** If the language of the uploaded file or input code is other than the two languages specified then the use case ends with a failure condition. Also if the selected language and input code language mismatch again failure occurs. **c. Successful Condition** If the input code is submitted successfully to the computation server, the code will be executed. **d. Failure Condition** If there is an error in code submission to the computation server, the use case fails. **Table 8: : Use Case Execution**
51
5.9 Sequence Diagram

5.9.1 Use case: Login

Figure 25: Login Sequence Diagram User requests for login and enters his/her credentials. The request is forwarded to the RequestHandler as it is the controller class. RequestHandler then submits name and password of the user to the database server for authentication and validation. The RequestHandler invoke the validate function. Then the database verifies the username and password and on valid search returns the Login successful message to the user. And on invalid username and password, Login Failure Message is returned.
52
5.9.2 Use case: Register

Figure 26: Register Sequence Diagram As the users need to register to our system for compilation and execution of their codes, so our system provides the user with SignUp facility. User requests for signup and enters his/her credentials that is username, email

and password. The request is forwarded to the RequestHandler as it is the controller class. RequestHandler then submits name, password and email of the user to the database server for registration. The database server invoke the set_username, set_password and set_email function. After successful insertion in database a message is returned to the user indicating successful sign up. In case of failure, a sign up failure message is returned back to the user.
53
5.9.3 Use case: Compile Code **Figure 27: Compile Code Sequence Diagram** After selecting a language, user uploads a code file or inserts code in text area and clicks the submit button. Submit_code() function is called. The request is forwarded to the RequestHandler. RequestHandle invokes the invoke_compile() function and request is forwarded to the Executor. The Executor selects among the two modules that is CSharp and VisualBasic according to the language selected. The Executor invokesthe respective compile() function and requests the ResultReportGenerator to generate result. The ResultReportGenerator submits the result to the RequestHandler. According to the type of user, RequestHandler calls select_view() and requests the View to setview(). The view sfter setting view calls the display_result() function and the RequestHandler calls the displaycompilationResult() and returns the Result back to the User.
54
5.9.4 Use case: Execute Code **Figure 28: Execute Code Sequence Diagram** After selecting a language, user uploads a code file or inserts code in text area and clicks the submit button. Submit_code() function is called. The request is forwarded to the RequestHandler. RequestHandle invokes the invoke_compile() and invoke_execute() function and request is forwarded to the Executor. The Executor selects among the two modules that is CSharp and VisualBasic according to the language selected. The Executor invokes the respective compile() and execute() function and requests the ResultReportGenerator to generate result. The ResultReportGenerator submits the result to the RequestHandler. According to the type of user, RequestHandler calls select_view() and requests the View to setview(). The view sfter setting view calls the display_result() function and the RequestHandler calls the displaycompilationResult() and returns the Result back to the User.
55
5.10 Activity Diagram **Figure 29: Activity Diagram**
56
5.11 Data Flow Diagram **Figure 30: Data Flow Diagram**
57
Chapter 6 6 System Implementation 6.1 Technology Used 6.1.1 Programming Language Used ASP.Net is used as programming language to develop Web pages. Java Script is used as client side scripting language. SQL language is used for managing data held in Relation Database Management System (RDBMS). **6.1.2 Development Tools** Microsoft Visual Studio 2012 and Windows Azure SQL server are used to develop the system. **6.1.3 Database** Database is developed and managed in MS Windows Azure SQL. **6.1.4 Operating System** 1. On server side, Windows Azure server is used as operating environment. While on client side, website application is tested for all browsers on Microsoft Windows and iOS. 2. Figure below the interaction of application with little information about technologies too. **Figure 31: Client Server Architecture**
58
6.2 Complete System Implementation The application consists of two servers. One server is used for both database and the application hosting. Database Server is created by translating the

schema into tables and relations. Application Server contains five modules implemented. Application Server will be discussed in detail in the further sub headings. **6.2.1 Administrator module** Module is linked with database and forms the data access layer of the application. Buisness layer consists of all the functions that are then accessed from the view. Firstly ,administrator View Layer consists of Login page. Usernames and passwords are stored in database so every time the user logs-in, usernames and passwords are compared with the values in database. Figure below shows the login page. **Figure 32: Login** After successful login the home page appears, which contains language selection feature, code editor feature, input feature and compilation feature.
59
6.2.2 User module This module is also linked with database and which forms the data access layer of the application. Buisness layer consists of all the functions that are accessed from the view. User module consists of login page. Username and password are compared with database. For unregistered users sign up page is provided. This webpage requires the user to enter username, password and email address which are stored in database and hence user is registered. Figure below shows the sign up page. **Figure 33: Sign Up 6.2.3 Language Selection module** After successful login, the first feature provided to user is language selection. This module provides the user with two options to select from: C# and Visual Basic. If C# option is selected then C# sample code appears in code editor. If Visual Basic option is selected then VB sample code appears in code editor. If Visual Basic option is selected then VB sample code appears in code editor. Figure below shows the language selection module.
60
Figure 34: Language Selection 6.2.4 Input Handling module This module handles the inputs which are to be provided by the user. Code is pasted in code editor then user clicks on show inputs button. Textboxes appear on the screen according to the input code provided by user. User is prompted to enter inputs in respective textboxes and click submit input button. All inputs are submitted to windows azure server for further handling. Figure below shows the input handling module. **Figure 35: Input Handling**
61
6.2.5 Compilation and Execution module This module takes user code and user given inputs as input. The parser parses the code. The input to the code generator typically consists of a parse tree or an abstract syntax tree. The tree is converted into a linear sequence of instructions; usually in an intermediate language such as three addresses code is compiled. **Figure 36: Compilation and Execution Module**
62
Chapter 7 7 System Implementation 7.1 Over View 7.1.1. Testing of the software projects involve different levels of testing to make sure that the software which is being developed is error and fault free. Cloud based Unified IDE for .NET Languages has different modules which were developed separately depending up on the functionalities. Therefore testing of all the modules has to be done and testing while integrating all the modules. The different levels at which testing was done are discussed here. **7.2 Unit Testing** Unit testing involves the testing of each module at the completion and sometimes during the development of the module. **7.2.1 Database Connectivity Testing** Test Case Name Database Connectivity Testing Test Case ID 1 Description This test case is used to check the database connectivity issues. Testing Technique Used White Box Testing Preconditions Accessed the application for database connectivity Input Values Connection string Valid Inputs Valid Connection string Steps - Create an object on Datacontext() - Call login function on it - Run the application and debug mode -

Check at every point by value of connection in local variables that is the database connected. Expected Output Connection string in local variables will have a
63
some valid values and same connection string value as given in input Actual Output Connection String has a value of User=haziqa and Password = 12345 Status PASS **Table 9: Database Connectivity Testing** **7.2.2 Login Feature Testing** Test Case Name Login Feature Testing Test Case ID 2 Description This feature asks the user to enter his/her credentials for login. The user can register to the database if not registered. This test case is aimed to check that this feature works according to the user requirements. Testing Technique Used Black Box Testing Preconditions The system is running and the database is connected Input Values Username string Password string Valid Inputs Valid/Authorized Username string Valid/Authorized Password string Steps - Enter Username - Enter Password - Press Login button Expected Output The user credentials will be passed to the server for verification. The valid users will be directed to CodeEditor webpage Actual Output Connection String has a value of User=haziqa and Password = 12345 Status PASS **Table 10: Login Feature Testing**
64
Figure 37: Login Feature Testing **7.2.3 Data Insertion Testing** Test Case Name Data Insertion Testing Test Case ID 3 Description This test case is used to check the database insertion issues. Testing Technique Used White Box Testing Preconditions System is running and database is connected. Input Values Valid insertion string Valid Inputs Valid insertion string Steps -Create an object on Datacontext -Create an object of insertion -Call insert() function on it -Run the application and debug mode -Check at every point by values of insertion in local variables that is the data is inserted into database. Expected Output Insertion strings in local variables will have some valid values and same insertion strings value as
65
given in input. Actual Output Insertion String has a value of Username :haziqa Password : 12345 email : haziqa512@yahoo.com Status PASS **Table 11: Data Insertion Testing** **7.2.4 SignUp Feature Testing** Test Case Name SignUp Feature testing Test Case ID 4 Description This feature asks the user to enter his/her username, password and email id. This test case is aimed to check that this feature works according to user requirements. Testing Technique Used Black Box Testing Preconditions System is running and database is connected. Input Values Username, password, email,country,address Valid Inputs -Username, password and address should have alphanumeric characters. -Email id should be in format like: someone@yahoo.com Steps -User clicks on sign up. -SignUp webpage appears. -User is asked to enter username, password and email id. -User clicks submit button. Expected Output User data is stored in database on server. Actual Output User data is stored in database on server. Status PASS **Table 12: SignUp Feature Testing**
66
Figure 38: Sign Up Feature Testing **7.2.5 Language Selection Feature Testing** Test Case Name Language Selection Feature Testing Test Case ID 5 Description This feature asks the user to select language from drop down list according to the code he/she wants to get compiled and Executed. Two options are provided that is CSharp and Visual Basic. This test case is aimed to check that this feature works according to the user requirements. Testing Technique Used Black Box Testing Preconditions The System is running, the tester is logged in and the database is connected. Input Values -Visual Basic -CSharp
67

Valid Inputs One of these two languages must be selected from drop down list. Steps After successful login: -User selects a language from the two options -Respective Code Editor is displayed -User uploads code file or copies code in text area Expected Output User selects a language from the two options Respective Code Editor is displayed Actual Output Respective Code Editor is displayed Status PASS **Table 13: Language Selection Feature Testing Figure 39: Language Selection Feature Testing 7.2.6 Show Input Feature Testing** Test Case Name Show Input feature testing Test Case ID 6 Description This feature asks the user to submit inputs in textboxes. This test case is aimed to check that this
68
feature works according to user requirements. Testing Technique Used Black Box Testing Preconditions System is running. User is logged in, Language is selected and respective code is inserted in Code Editor Input Values User input Valid Inputs User input Steps -User selects language -User enters code in textbox -User clicks on Show Input button -User enters respective inputs in textboxes -User presses submit button Expected Output Input textboxes appear Actual Output Input textboxes appear Status PASS **Figure 40: Show Input Feature Testing Figure 41: Show Input Feature Testing**
69
7.2.7 Submit Input Feature Testing Test Case Name Submit Input feature testing Test Case ID 7 Description This feature submits the input provided by the user to the server side. This test case is aimed to check that this feature works according to user requirements. Testing Technique Used Black Box Testing Preconditions System is running. User is logged in, Language is selected and respective code is inserted in Code Editor and inputs are provided by user in input textboxes. Input Values User inputs in textboxes Valid Inputs User inputs in textboxes Steps -User clicks on Show Input button -User enters respective inputs in textboxes -User presses submit button Expected Output Input textboxes disappear and input is submitted to server. Actual Output Input textboxes appear and input is submitted to server. Status PASS **Table 14: Submit Input Feature Testing 7.2.8 Compilation and Execution Feature Testing** Test Case Name Compilation and Execution Feature Testing Test Case ID 8 Description This feature compiles the code provided by the user either by uploading text file or copying code in text area. This test case is aimed to check that this feature works according to the user requirements.
70
Testing Technique Used Black Box Testing Preconditions The System is running, the tester is logged in and the database is connected, input must be submitted if any. Input Values Code file according to selected Language Valid Inputs Upload Code file or copy code in text area according to selected Language Steps After successful login: -User selects a language from the two options -Respective Code Editor is displayed -User uploads code file or copies code in text area -User submits the input -User clicks the compile button Expected Output Build is successful Actual Output Build is successful Status PASS **Table 15: Compilation and Execution Feature Testing 7.2.9 CodeCompilationExecution Testing** Test Case Name CodeCompilationExecution Testing Test Case ID 9 Description This test case is used to check the code compilation issues. Testing Technique Used White Box Testing Preconditions The System is running, the database is connected, the user is logged in and the code as well as inputs must be submitted if any Input Values Code according to selected Language Valid Inputs Valid Code file or code in text area according to selected Language Steps After language identification and code submission:
71
-Code is inserted into the string -Code is tokenized and parsed -Then code compilation is done -Then the compiled code is executed and execution output is returned Expected Output -Result of

the code along with Build successful message in case of no errors in the code -Executed code errors after compilation and execution as an output Actual Output -Result of the code along with Build successful message in case of no errors in the code -Executed code errors after compilation and execution as an output Status PASS **Table 16: CodeCompilationExecution Testing Figure 42: CodeCompilationExecution Testing**
72
7.2.10 Performance Testing Test Case Name Performance Testing Test Case ID 10 Description This feature allows the tester to check the performance of application in terms of load testing and stress testing Testing Technique Used Black Box Testing Preconditions N/A Input Values URL of website Valid Inputs N/A Steps All buttons are clicked one by one under different scenarios to check the system response Expected Output As shown below Actual Output As shown below Status PASS **Table 17: Performance Testing Execution Result Figure 43: Performance Testing (1)**
73
Figure 44: Performance Testing (2) Figure 45: Performance Testing (3)
74
Figure 46: Performance Testing (4) Figure 47: Performance Testing (5)
75
7.3 Integration Testing In the integration testing, incremental testing technique is used. Incremental testing technique is in correspondence with the system development. With each module developed, it is integrated with main system and integration testing is performed to ensure the performance of system. Integration testing is also used to ensure that system‟s non-functional requirements are also met at each step. Cloud based Unified IDE for .NET Languages is mainly divided into five modules. • Administrator Module • User Module • Language Selection Module • Input Handling Module • Compilation and Execution Module **Increment 1: Administrator Module Test Case ID Input(s) Initial Condition Output(s) Sequence of Actions Result 11** -Username -Password System is running and database is connected -Administrator status logged in -Language selection option is opened -Webpage for login appears PASS **Table 18: Administrative Module Increment 2: Integration with User Module Test Case ID Input(s) Initial Condition Output(s) Sequence of Actions Result 12** -Username -Password System is running and database is connected -System user status logged in -Language selection option is opened -Webpage for login appears -Username and password entered PASS
76
13 -Username -Password -email id System is running and database is connected -System user status registered -Login web page appears -Webpage for sign up appears -Username, password and email address entered PASS **Table 19: Integration with User Module Increment 3: Integration with Language Selection Module Test Case ID Input(s) Initial Condition Output(s) Sequence of Actions Result 14** -C# -Visual basic System is running and database is connected and user is logged in -Language selected -Language selection drop down list appears -A language is selected from the list PASS **Table 20: Integration with Language Selection Module Increment 4: Integration with Input handling module Test Case ID Input(s) Initial Condition Output(s) Sequence of Actions Result 15** Code inserted into code editor -System is running, database is connected -Language is selected Input textboxes appear if any -Code is inserted in code editor -Show input button is pressed -Input textboxes appear PASS
77

16 Input inserted into input textboxes -System is running, database is connected -Language is selected -Code is inserted in code editor -Show input button is pressed Input submitted to server -Input textboxes appear -Input is inserted into textboxes -Submit input button pressed and input submitted to server -Input textboxes disappear PASS **Table 21: Integration with Input Handling Module Increment 5: Integration with Compilation/execution module Test Case ID Input(s) Initial Condition Output(s) Sequence of Actions Result 17** -Code in code editor and inputs in textboxes from user System is running, database is connected -Language is selected -Code is inserted in code editor -Inputs are inserted in textboxes and submitted to server Code is compiled and output displayed in output textbox -Compile button is pressed -Output results are shown in output textbox PASS **Table 22: Integration with Compilation/Execution Module**
78
7.4 System Testing 7.4.1. System testing is the level of testing which comes when the whole system has been developed and integrated. The complete system was tested in different inputs with different conditions to verify that those conditions do not disrupt the performance of the system. **Test Case 18** Test Case ID 18 Test Case name Login Input(s) Select Login link and enter Username and Password. Then click on Login button Output Code Editor Page opens Result Success **Table 23: Login system test Test Case 19** Test Case ID 19 Test Case name Sign Up Input(s) Select Sign Up link and then enter Username, Password and Email. Then click on Register button Output Code Editor Page opens Result Success **Table 24: SignUp System Test Test Case 20** Test Case ID 20 Test Case name Language Selection Input(s) Select Language from the dropdown list Output Code Editor appears along with the sample code of that language in textbox Result Success **Table 25: Language Selection System Test**
79
Test Case 21 Test Case ID 21 Test Case name Show Input Input(s) Enter the code in Code Editor textbox and then click on Show Input button Output Input textboxes appear Result Success **Table 26: Show Input System Test Test Case 22** Test Case ID 22 Test Case name Submit Input Input(s) Enter the inputs in the textboxes and click on Submit Input button Output Input Textboxes disappear Result Success **Table 27: Submit Input System Test Test Case 23** Test Case ID 23 Test Case name Compilation and Execution Input(s) Click on Compile Button Output Output result appears in the output textbox Result Success **Table 28: Compilation and Execution System Test Test Case 24** Test Case ID 24 Test Case name Clear code Input(s) Click on Clear button Output Code from Code editor disappears Result Success **Table 29: Clear Code System Test**
80
Test Case 25 Test Case ID 25 Test Case name Sample Codes Input(s) Click on Sample Codes link Output Sample Codes webpage appears Result Success **Table 30: Sample Codes System Test Test Case 26** Test Case ID 26 Test Case name Screenshots Input(s) Click on Screenshots link Output Webpage of screenshots appears Result Success **Table 31: ScreenShots System Test**
81
8 Conclusion and Future Work 8.1. Conclusion 8.1.1. The Cloud Based Unified IDE for .NET Languages has Five modules and we have implemented all modules namely Language Selection module, Compilation and Execution module, Administrator module, User module, Input Handler module. Our product “Cloud based Unified IDE for .NET” is an embedded web application that provides an environment for online compilation and execution of codes. It supports multi-languages which include C# and Visual Basic. Users performs remote online compilation and execution of their codes without requiring intensive CPU and memory usage as

Unified IDE is provided on the Cloud that is Microsoft Windows Azure Platform. By combining Cloud Computing technology, our web application facilitates the remote users and removes the requirement for powerful systems and provides portability through provision of SaaS. Our application supports Windows users, iOS users as well as Android Users. **Future Work** 8.2.1. Future work on this project can include adding more options of programming Languages like C++, Java and COBOL. A new feature Joint code Development can be introduced to facilitate multiple users to work in team supporting versioning system of codes.
82
Appendix How to use Software The main interface of Cloud Based Unified IDE for .NET languages is as follows. This is the Home page of the web application. **Figure 48: System Interface** From Home webpage user must go to Login webpage. On Login webpage user enters his login name and password. **Figure 49: Login Webpage**
83
Login is must for using the facility of Code Editor. In case user is not registered, he must go to Sign Up Webpage and enter username, password, email address, select country from dropdown list and enter full address. Sign Up webpage is as follows. **Figure 50: Sign Up webpage** After successful login, user goes to Editor Webpage. Editor Webpage is shown below. On this webpage user selects a language from dropdown menu i.e. C#, visual basic. User can also select a theme from the list of themes provided in dropdown list. **Figure 51: Select Language And Themes**
84
After selecting language user inserts code to be compiled in Code Editor. **Figure 52: Code Editor** Then user clicks on show input button. Input textboxes appear on webpage, user enters input into respective textboxes and presses submit input button. After submitting input, user clicks on compile button. **Figure 53: Show and Submit Input**
85
After compilation, output result is shown in output textbox. **Figure 54: Output** Sample Codes Webpage is provided with samples of C# and visual basic codes. **Figure 55: SampleCodes WebPage**
86
Screen Shots Webpage shows the steps that how user can use web application **Figure 56: ScreenShots Webpage** .About Webpage gives information about our web application. Successive versions of development are provided on this Webpage along with description of web application. **Figure 57: About Webpage**
87
About webpage also contains Contact Us facility. User can send his queries at the link provided on Webpage. **Figure 58: Contact Us**
88
Glossary HTML Hyper Text Markup Language HTTP Hypertext Transfer Protocol (HTTP) is a widely used communications protocol for communication over a computer network, with especially wide deployment on the Internet IP Internet Protocol API Application Programming Interface (ISAPI) MCS Military College of Signals NUST National University of Science and Technology SRS Software Requirements Specification SaaS Software as a Service TCP Transmission Control Protocol VB Visual Basic OS Operating System GUI Graphical User Interface FRs Functional Requirements NFRs Non Functional Requirements DBMS Database Management System UML Unified Modeling Language JavaScript Client side scripting

language used to create dynamic web content and user interface SQL Structured Query Language
89
Windows Azure Cloud Server Response time The time a system or functional unit takes to react to a given input Throughput The amount of data transferred from one place to another or processed in a specified amount of time Asp.net Active Server Pages (ASP) is a server side object-oriented scripting technology, an event-driven programming model for developing Web pages where .NET is a framework which provides a platform for developing multi-tier application Black box Testing Testing emphasizes on the external behavior of the software entity White Box Testing Testing emphasizes on the internal behavior of the software entity **Table 32: Glossary**
90
Bibliography [1] Wikipedia, Online, http://en.wikipedia.org/wiki/Cloud_computing, 2012 [2] Coursera, Online: https://www.coursera.org/, 2012 [3] Google App Engine, Online: https://cloud.google.com/products/, 2012 [4] Windows Azure, Online: http://www.windowsazure.com/en-us/, 2012 [5] EugeneE. Marinell, "Hyrax: CloudComputingonMobileDevices usingMapReduce", Online: http://www.contrib.andrew.cmu.edu/~emarinel/masters_thesis/emarinel_ms_thesis.pdf, September 2009 [6] Abhinav Pathak, Y. Charlie Hu, Ming Zhang, Paramvir Bahl and Yi-Min Wang "Enabling Automatic Offloading of Resource Intensive Smartphone Applications", Tech. rep., 2011. [7] Changj Xian , Yung-Hsiang Lu and Zhiyuan Li, "Adaptive computation offloading for energy conservation on battery-powered systems", In 13th International Conference on Parallel and Distributed Systems (ICPADS 2007), December 5-7, 2007, Hsinchu, Taiwan. [8] Gonzalo Huerta-Canepa, Dongman Lee, "A Virtual Cloud Computing Provider for Mobile Devices", Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, USA, 2010. [9] Stelios Xinogalos, Kostas E. Psannis, Angelo Sifaleras, "Recent advances delivered by HTML 5 in mobile cloud computing applications: a survey" ACM Proceedings of the Fifth Balkan Conference in Informatics, Serbia, 2012. [10] Fan Yang, Zhengping Qian, Xiuwei Chen, Ivan Beschastnikh, Li Zhuang, Lidong Zhou, Guobin Shen, "Sonora: A Platform for Continuous Mobile-Cloud Computing" Microsoft Research technical teport MSR-TR-2012-34, March 2012. [11] Compilr, 2013. Online: https://compilr.com/ [12] Ideone, 2013. Online: www.ideone.com [13] CompileOnline: www.compileonline.com