Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 116 (2017) 222–231

Procedia Computer Science

www.elsevier.com/locate/procedia

2nd International Conference on Computer Science and Computational Intelligence 2017, ICCSCI 2017, 13-14 October 2017, Bali, Indonesia

Harmonik = ++(Web IDE)

Budi Yulianto[1]*, Harjanto Prabowo[2], Raymond Kosala[3], Manik Hapsara[4]

[1,4] Computer Science Department, School of Computer Science, Bina Nusantara University, Indonesia
[2] Management Department, BINUS Business School, Undergraduate Program, Bina Nusantara University, Indonesia
[3] Computer Science Department, Faculty of Computing and Media, Bina Nusantara University, Indonesia

Abstract

IDE (Integrated Development Environment) that is needed by developers to write sourcecode for computer programming is also starting to shift from desktop to web platform. As implemented in the cloud, Web IDE (WIDE) can be accessed online through browsers and on mobile devices. WIDE takes many advantages and becomes popular in recently years. Unfortunately, there is no tool found yet in currently WIDE to support translating novice programmers' algorithm into programming language before they become experts in techniques of writing computer programs. This study goal is to design a tool to support novice programmers' early learning of programming before they step into coding, compiling, running, and debugging in WIDE. Software development method used in this study is Rational Unified Process (RUP). Conclusion of this study is block-code tool can help novice programmers to learn programming language in early phase, and motivate them better.

Peer-review under responsibility of the scientific committee of the 2nd International Conference on Computer Science and Computational Intelligence 2017.

Keywords: Web IDE, Online IDE, Cloud IDE, Mobile IDE, Browser IDE, Integrated Development Environment

1. Introduction

The trend and benefit of cloud computing has migrated many desktop applications into the cloud[1]. IDE (Integrated Development Environment) that is needed by developers to write sourcecode for computer programming is also starting to shift from desktop to web platform. Some advantages taken from Web IDE (WIDE) are capability of

* Corresponding author. Tel.: +62-21-5345830; fax: +62-21-5300244.
E-mail address: [1] budi.yulianto@binus.edu;

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the 2nd International Conference on Computer Science and Computational Intelligence 2017.
10.1016/j.procs.2017.10.044

managing sourcecode files on cloud; no need to install, config, and maintain many compilers/SDK[2]; virtually huge computing power; easy collaboration and communication with partners; widely online resources; accessible through many devices[3]; easy to integrate with online services[4]; accessible anytime and anywhere[5]; no licensed copies of the required software (just browser)[6,7], same environment development to reduce errors; and increase economical return in user side by reducing maintenance and operational costs of infrastructure[8]. Unfortunately, WIDE is also bringing disadvantages such as lack of advanced features for expert developer; less security for server if no access restriction management[1]; internet connection required[9]; unable to diagnose problems which require access to logs and process inspection tool; and hard to do GUI programming[10].

Some WIDE are Cloud9, CodeRun Studio, Eclipse Orion, eXo Cloud IDE, etc[2,4,6]. Cloud9 IDE supports HTML for web development, and real time collaboration. Code Run Studio supports ASP .NET, and C# .NET language, and it allows sharing of code through hyperlinks. Eclipse Orion is focusing on web development (HTML and JavaScript). eXo Cloud IDE supports Java programming but it does not support real time collaboration.

To develop good programming skills, novice programmers must be provided with a tool to translate their algorithm into programming language before they become experts in techniques of writing computer programs[11,12]. Without any assistance, they usually face many obstacles associated with this phase. Furthermore, there is no that feature found yet in currently WIDE. It needs to support their early learning of programming language before they step into coding, compiling, running, and debugging. Based on that, research question in this study is "how is the tool to be developed on WIDE for novice programmer and what are popular tools developed in the current WIDE?".

2. Related Works

To answer that research question, this study refers to some previous studies. Finding previous studies (papers) is done through Systematic Literature Review (SLR) method by entering keywords in the search engine of some popular web publishers, such as IEEE, ACM, ScienceDirect, Springer Link, and Taylor & Francis. To complete the sources, searching additional papers is also conducted on Google Scholar in addition to the popular web publishers, and traced its references. Keywords entered for searching are "(web OR mobile OR online OR cloud) AND (IDE OR compiler OR programming OR code)".

Table 1. List of Selected Papers

No	ID	Title & Reference Index	Publisher	No	ID	Title & Reference Index	Publisher
1	10A	Adinda: a knowledgeable …[17]	IEEE	24	13L	Compilers on Cloud[29]	Other
2	10B	JavaWIDE: innovation in …[14]	ACM	25	14A	TouchDevelop: create rich …[30]	ACM
3	11A	Collabode: collaborative …[18]	ACM	26	14B	.NET Compiler using Cloud …[8]	Other
4	11B	Real-time collaborative …[19]	ACM	27	14C	Lessons from a web-based IDE …[31]	ACM
5	11C	CEclipse: An Online IDE for …[1]	IEEE	28	14D	Online Java Compiler Using …[32]	Other
6	11D	Online C/C++ compiler using …[2]	IEEE	29	14E	Learning and practicing object …[33]	IEEE
7	12A	Specification engineering and …[20]	IEEE	30	14F	Improved Interaction in Web …[34]	Other
8	12B	CoRED: browser-based …[3]	ACM	31	14G	Developing a SAAS-Cloud …[35]	Other
9	12C	Implementation of Browser …[15]	Other	32	14H	Web Based IDE[9]	Other
10	12D	Software development …[4]	ACM	33	14I	Web Based Integrated …[10]	Other
11	12E	Web Based 'C' IDE: Approach[5]	Other	34	15A	Automating Repetitive Tasks …[36]	IEEE
12	12F	Designing IDE as a Service[6]	Other	35	15B	The Cloud Based Compiler for …[37]	Other
13	13A	Specification and reasoning in …[21]	IEEE	36	15C	Web2Compile-CoT: A Web …[38]	Spr. Link
14	13B	Browser Based IDE to Code in …[16]	Spr. Link	37	15D	CodeR: Real-time Code Editor …[39]	Sci. Dir.
15	13C	WIDE an interactive Web …[11]	IEEE	38	15E	Web Based Interface …[40]	Other
16	13D	C/C++ Cloud Compiler Using …[7]	Other	39	15F	Survey on Web Based Interface[41]	Other
17	13E	Cloud Based Compiler[22]	Other	40	15G	Online C, C++, Java Compilers …[42]	Other
18	13F	Online C, C++, Java Compilers …[23]	Other	41	15H	Web Based IDE …[43]	Other
19	13G	Cloud Documentation and …[24]	Other	42	16A	Jimbo: A Collaborative IDE …[44]	IEEE
20	13H	An Effective C, C++, PHP, Perl …[25]	Other	43	16B	Selecting the most appropriate …[45]	IEEE
21	13I	Compiler as Service over Cloud[26]	Other	44	16C	A Web-Based IDE for …[46]	ACM
22	13J	Online Java Compiler Using …[27]	Other	45	16D	Online Editor for Compiling …[47]	Other
23	13K	An interactive Web-based IDE …[28]	IEEE				

In summary, the steps of the SLR method are as follows. Hundreds to thousands of papers are displayed on every web publishers. There are hundreds of papers with the title approaching to answer the research question. Furthermore, the abstract of them are read and which are approaching to answer the research question are selected. Then, the contents of them are read and 45 papers that can answer the research question are obtained (Table 1). They are from IEEE publishers (11 papers), ACM (8), Springer Link (2), ScienceDirect (1), and Others / Google Scholar (23). The ID column consists of 2 digits of the year the paper is published, and a letter as sequester/differentiator. For example, ID '10A' means that the paper was published in 2010. Detail information of the each selected paper can be seen on the Reference section at the end of this paper.

The next step is forming the roadmap of papers by tracing their references. Roadmap is used to know the progress of research that has been done from each previous study. The results of the tracing are shown in Table 2. The 'Predecessors ID' column refers to its predecessor paper ID (obtained by tracing their references). There are some papers that do not have corresponding predecessor papers from Table 1 (marked with dash symbol '-').

Table 2. Predecessor of Selected Papers (Roadmap)

No	ID	Predecessors ID	No	ID	Predecessors ID	No	ID	Predecessors ID
1	10A	-	16	13D	-	31	14G	13B, 11B, 11C
2	10B	-	17	13E	-	32	14H	13J, 12E, 13E, 13G
3	11A	10A	18	13F	-	33	14I	13I, 13L, 11D, 13J
4	11B	10A, 11A	19	13G	11D	34	15A	-
5	11C	10A, 10B	20	13H	11D	35	15B	-
6	11D	-	21	13I	11D	36	15C	12F
7	12A	-	22	13J	11D	37	15D	12D, 11B
8	12B	-	23	13K	10A, 11B, 11C	38	15E	13L, 13H, 13J, 12D
9	12C	-	24	13L	11D, 13J	39	15F	13I, 13L, 13H, 13J, 10A
10	12D	10A	25	14A	-	40	15G	13J, 13F, 11D, 13H, 13G, 13E
11	12E	11C	26	14B	-	41	15H	13I, 13L, 13H, 13J, 12D, 10A
12	12F	10A, 10B, 11B, 11C, 12B	27	14C	14A	42	16A	-
13	13A	-	28	14D	11D	43	16B	-
14	13B	-	29	14E	13K, 11B	44	16C	-
15	13C	-	30	14F	12E, 11B, 13B, 13J, 12F	45	16D	11D, 13G, 13H

The roadmap diagram is shown in Figure 1. The top of it shows the year the paper published. There is no difference in the meaning of the line style (colour, pattern, thickness) used. These styles are to facilitate visualization so that the lines are not overlap each other. Figure 1 shows that WIDE researches have begun since 2010 and began to be popular in 2013. As for every paper are supporting the next researches.

The extraction of the papers is done to obtain the tools (features) developed and the software architecture used in each WIDE development process. The most widely developed tools are Editor (44 papers: 10A, 10B, 11A, 11B, 11C, 11D, 12A, 12B, 12C, 12D, 12E, 12F, 13A, 13C, 13D, 13E, 13F, 13G, 13H, 13I, 13J, 13K, 13L, 14A, 14B, 14C, 14D, 14E, 14F, 14G, 14H, 14I, 15B, 15D, 15E, 15F, 15G, 15H, 16A, 16C, 16D), Compile & Run (37 papers: 10A, 10B, 11A, 11B, 11C, 11D, 12C, 12D, 12E, 12F, 13C, 13D, 13E, 13F, 13H, 13I, 13J, 13K, 13L, 14A, 14B, 14C, 14D, 14E, 14F, 14G, 14H, 14I, 15B, 15D, 15E, 15F, 15G, 15H, 16A, 16C, 16D), and Output Preview (24 papers: 10A, 10B, 11A, 11B, 11C, 12D, 12E, 13C, 13E, 13F, 13H, 13I, 13J, 13L, 14F, 14G, 14H, 15D, 15E, 15F, 15G, 15H, 16C, 16D). The tools that need to be considered in the development are File Management (13 papers: 11D, 12C, 12F, 13E, 13G, 13L, 14H, 14I, 15B, 15E, 15F, 15G, 15H), Debug (12 papers: 11C, 12D, 13A, 13G, 13K, 14A, 14C, 14E, 14I, 15B, 15G, 16C), Collaboration (10 papers: 10A, 10B, 11A, 11B, 12B, 12F, 13G, 13K, 15D, 16C), Register and Log (9 papers: 12C, 12E, 12F, 13G, 13L, 14B, 14I, 15B, 15G), and Communication (7 papers: 10A, 12C, 13C, 13E, 13K, 14E, 16A). Editor is a tool for user to type in sourcecode, Compile & Run is required to compile and execute sourcecode, and Output Preview displays execution results to users. Features File Management is required to manage (save, open, and delete) files created on the server, Debug for tracing variable value when the execution took place, Collaboration allows multiple users to edit/type on the same sourcecode, Register & Login allows users to register and login to the system, and Communication as a medium of communication between users.

The software architecture is not clearly explained in the most papers so analysing and classifying it into 3 types are needed, namely MVC (15 papers: 11C, 11D, 12C, 13E, 13G, 13I, 13L, 14E, 14F, 14H, 14I, 15D, 15E, 15G, 15H), Three Tier (5 papers: 10A, 13C, 13F, 13H, 13J), and Two Tier (4 papers: 12B, 12E, 12F, 13D). 21 papers do not explain it in the development process. This paper does not specify the theory of each architecture as it is not the focus of this study, so readers can learn them on the internet if necessary. Most studies use MVC as their software architecture and it can be considered for readers in developing WIDE[13].

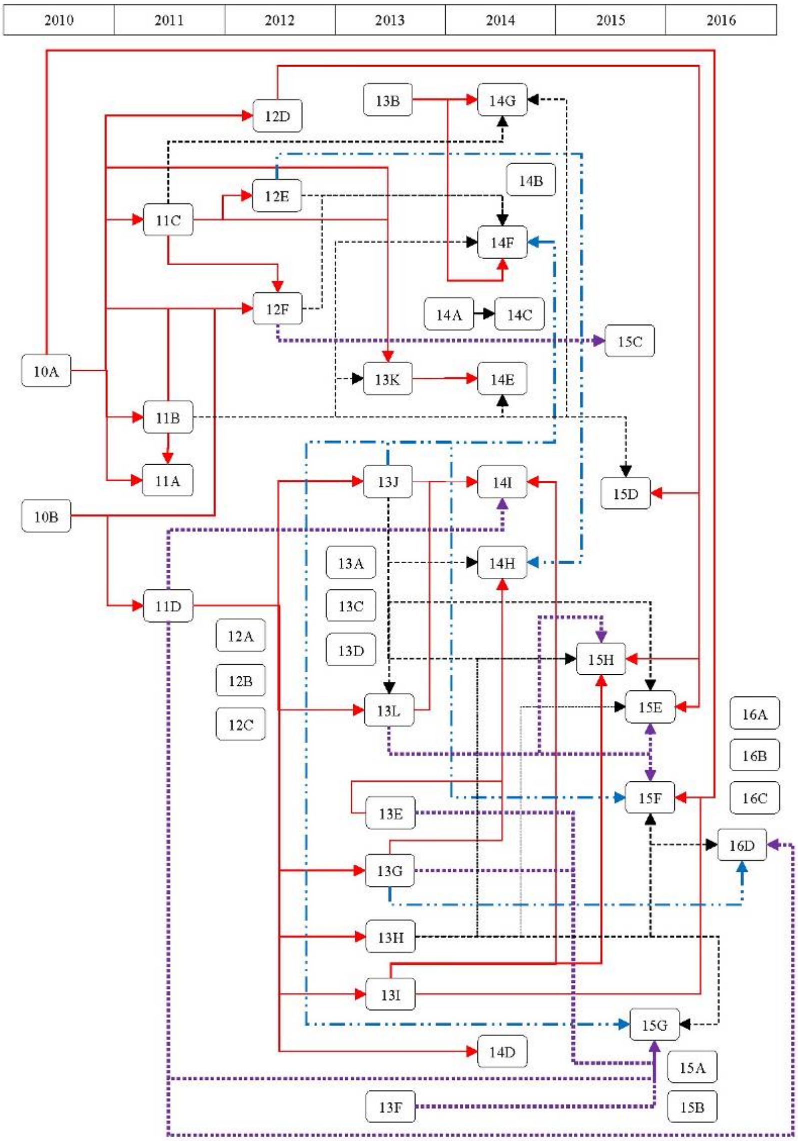


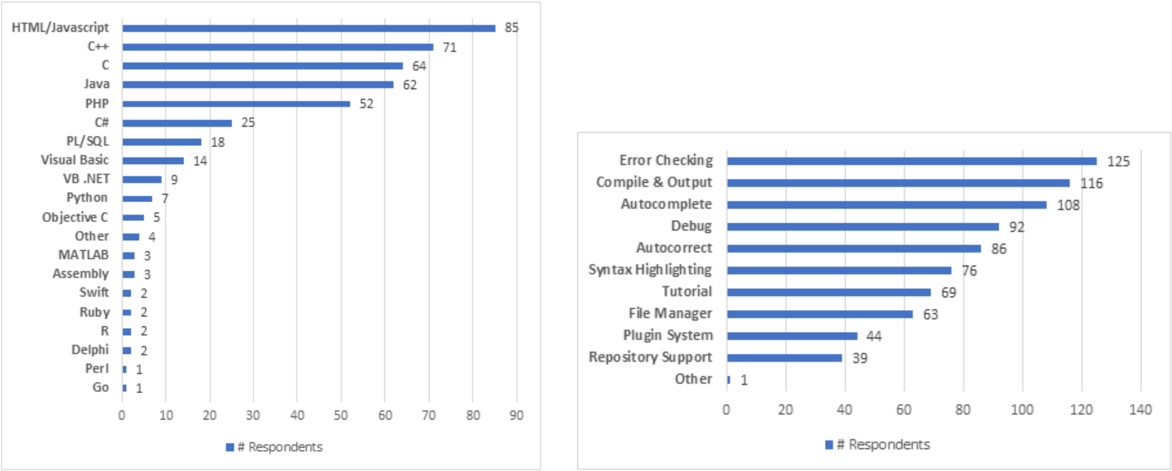
Figure 1. Predecessor of Selected Papers (Roadmap)

Extraction is also done to get the programming languages provided on the developed WIDE. Java (25 papers: 10A, 10B, 11A, 11B, 11C, 12A, 12B, 12C, 12D, 12F, 13A, 13E, 13G, 13I, 13J, 13L, 14D, 14G, 14I, 15B, 15D, 15F, 15G, 15H, 16D), C (18 papers: 11D, 12E, 13C, 13D, 13F, 13H, 13K, 13L, 14E, 14F, 14G, 14H, 14I, 15D, 15F, 15G, 15H, 16D), and C++ (14 papers: 11D, 13D, 13E, 13F, 13H, 13K, 14E, 14G, 14I, 15D, 15F, 15G, 15H, 16D) are the most programming languages provided on the WIDE. The programming languages to be considered for development are Python (6 papers: 13H, 14I, 15E, 15F, 15H, 16D), HTML (5 papers: 13E, 14I, 15E, 15H, 16A), PHP (5 papers: 13E, 13G, 13H, 14I, 15F), Perl (5 papers: 13H, 14I, 15E, 15H, 16D), Ruby (5 papers: 13H, 14I, 15E, 15F, 15H), and .NET (5 papers: 13I, 14B, 14I, 15F, 15H).

Based on the extraction from previous researches, no feature has been found for transferring algorithm into the programming language (sourcecode). This feature will be developed based on block-code that allows users to translate algorithm into pseudocode and then converted into sourcecode form. Users are expected to learn from the sourcecode generated by the system and then try to modify it (learning by doing) before start to code for first. This sequence of learning that starts from transferring algorithm into pseudocode, then into programming language is a ‘harmony’ way as is expected as a ‘pre-increment’ of existing WIDE technology. That is why we call it Harmonik as ++(Web IDE).

3. Software Development Methodology

The software development methodology used in this research is Rational Unified Process (RUP) with 4 phase: inception, elaboration, construction, and transition. In the inception phase, scope definition and collection requirements are done by extracting previous 45 papers and then distributing questionnaire to students. In the second phase, remaining requirements are collected and application system is modelled by using Unified Modelling Language (UML). The third phase carried out the development of features. In this phase also made implementation planning, testing and bug fixing, and user manual documents updating. In the fourth phase, implementation of the system is done then released the application to the user for early beta testing. Details of each phase, UML, and development process are not described in this paper as it is not in the focus of the discussion.



(a) **(b)**

Figure 2. (a) Programming Languages Learned by Respondents, and (b) Features Needed by Respondents

Questionnaire is randomly distributed to 130 first-year students from the Department of Computer Science and Information Systems. The results of the questionnaire show that the programming languages that students are studying through the WIDE (Figure 2.a) are HTML / JS (85 respondents), C++ (71), C (64), Java (62), PHP (52), and C# .NET (25). The features desired by the respondents are on the WIDE shown in Figure 2.b.

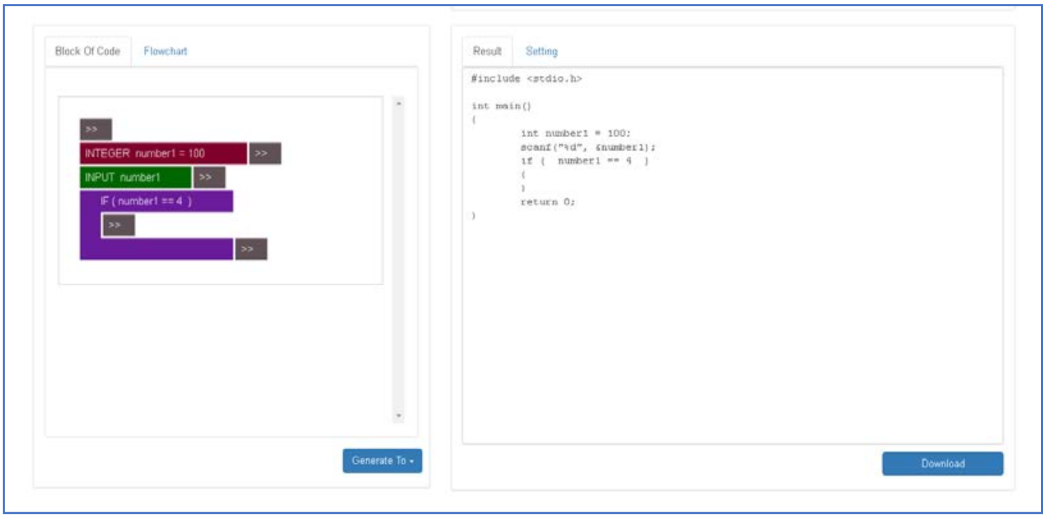
4. Proposed System

The proposed system consists of 2 main components based on the results of the SLR and questionnaires: tools (features) and programming languages provided (Table 3). The tools that are developed include Editor (including Code Template), Compile and Run, Output Preview, Debug and Input (including Error Checking), Register and Login, File Management, Collaboration and Communication, and Tutorial. In addition to these tools, a block-code tool to translate algorithm into the programming language (sourcecode) is developed so that users can learn from the system-generated sourcecode and then be modified (learning by doing). The programming languages provided are Java, C, C++, Python, C# .NET, HTML & JS, PHP, Perl, and Ruby. These programming languages selection, in addition to SLR and questionnaire results consideration, is also considered by ranking of Top 10 TIOBE programming languages of December 2016.

Table 3. Tools (Features) and Programming Languages Provided in Proposed System

No	Tools	Top 5 in SLR	Top 5 in Questionnaire	No	Prog. Lang.	Top 5 in SLR	Top 5 in Questionnaire
1	Editor	Yes (1)	Yes (3,5,6)	1	Java	Yes (1)	Yes (4)
2	Compile	Yes (2)	Yes (2)	2	C	Yes (2)	Yes (3)
3	Run	Yes (2)	Yes (2)	3	C++	Yes (3)	Yes (2)
4	Output	Yes (3)	Yes (2)	4	Python	Yes (4)	No (10)
5	File Management	Yes (4)	No (8)	5	HTML	Yes (5)	Yes (1)
6	Debug	Yes (5)	Yes (4)	6	PHP	Yes (5)	Yes (5)
7	Collaboration	No (6)	-	7	Perl	Yes (5)	No (19)
8	Register	No (7)	-	8	Ruby	Yes (5)	No (16)
9	Login	No (7)	-	9	.NET	Yes (5)	No (6)
10	Communication	No (8)	-	10	JS	No (6)	Yes (1)
11	Error Checking	-	Yes (1)	11	PL/SQL	-	No (7)
12	Tutorial	-	No (7)	12	VB	No (7)	No (8)

The system is implemented on a server with hardware specifications (for 1000 users) of 8 GB RAM, Intel Core i7 4770 3.9 GHz processor, and 5 GB storage, and software specifications of XAMPP v3.2.2 and compilers for each programming language. Users require 1 GB of RAM hardware, 1.5 GHz Intel Pentium 4 processor (for PC/laptop) or dual core 1.2 GHz (for mobile phone), and 1 GB of storage to run the browser. The first look of a Harmonik application is a block-code (Figure 3) that allows users to translate algorithms into blocks of code (such as pseudocode). After that, users can generate it into the sourcecode according to the selected programming language. The generated sourcecode is then transferred on the next page which is divided into 4 sections, namely Tutorial, Editor, Setting (including Debug and Input), and Output Preview (Figure 4).



```
#include <stdio.h>

int main()
{
        int number1 = 100;
        scanf("%d", &number1);
        if (  number1 == 4  )
        {
        }
        return 0;
}
```

Figure 3. Block-Code Page

Users can enter data/value on Input tab and debug. In the debug process, users can trace in previous or next step. The value of variable will be displayed according to the position of the tracing line. On the Settings tab, users can choose which programming language to use, set the theme (font size, background, and text color) of Editor and Output Preview.

Beside on laptop/PC, Harmonik can also be used on mobile devices (Figure 5). The use of mobile devices is recommended to test the existing copy-paste sourcecode rather than to type from scratch due to screen limitations (small font size), keypad, or typing easiness by using 2 thumbs. For security reason, some classes and libraries have been removed from each compiler's SDK, such as libraries that can access to system directory, file processing, hardware access, and other issues[1]. System will terminate the execution for more than 10 seconds processing to prevent the infinite loop or server overload.

5. Experimental Results

Experimental results of this study are done in 2 ways: questionnaire (qualitative) and trial (quantitative). Based on the results of the evaluation questionnaire, 71.9% of respondents agree that Harmonik can help them to learn programming languages anywhere, anytime, and with any device. In addition, 93.8% of respondents agree that Harmonik can motivate them to learn programming languages better.

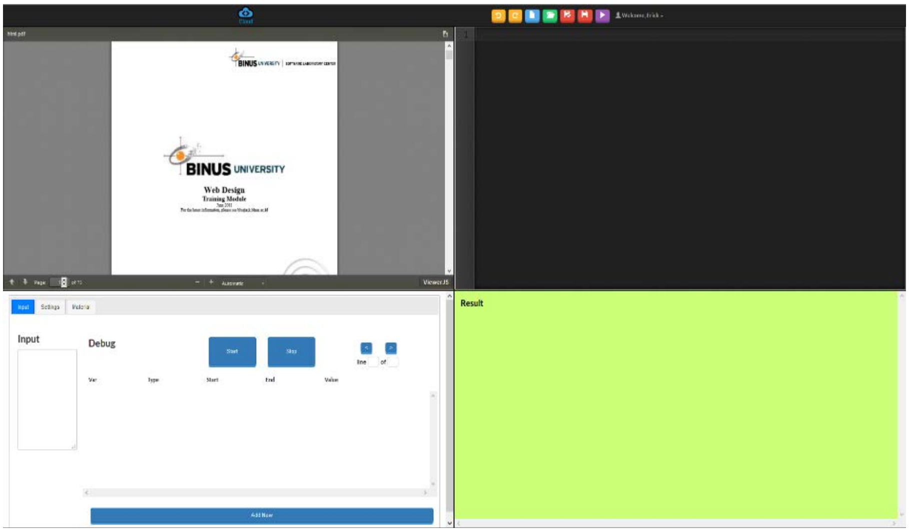
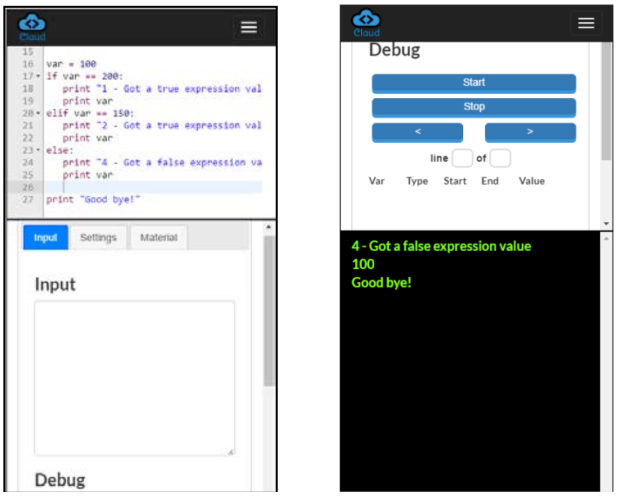


Figure 4. Application Main Page



```
var = 100
if var == 200:
   print "1 - Got a true expression val
   print var
elif var == 150:
   print "2 - Got a true expression val
   print var
else:
   print "4 - Got a false expression va
   print var

print "Good bye!"
```

```
4 - Got a false expression value
100
Good bye!
```

Figure 5. Application Page on Mobile Phone

Trial is conducted to determine the speed (time) of execution (compile and run) on Harmonik and native applications (desktop) on PC. Trial is performed by using Bubble Sort algorithm to order 150000 random numbers in Java, C, C++, C#, and Python on the same hardware (server) specifications and operating conditions. Each trial is performed 10 times for each programming language and then an average value is calculated. Calculation on Harmonik is done on 3 phases, first phase (P1) when client (browser) sends data (sourcecode) and received by server, second phase (P2) when server executes (compile, run, output) sourcecode, and third phase (P3) when server sends data back (output / Result) to the client. Calculation on each native application is only done on second phase when server executes (compile, run, output) sourcecode.

Comparison of the execution speed calculation is based on the time difference of phase 2 between Harmonik and native application (△1 = P2 Native – P2 Harmonik), and based on the time duration between total duration (3 phases) in Harmonik and phase 2 in native application (△2 = P2 Native - [P1+P2+P3] Harmonik) where using Harmonik must accept additional time consequences of sending and receiving data via internet. The positive mark of difference results show the execution time in Harmonik is faster than native application. Table 4 shows that the execution time in Harmonik is relatively bit faster than in the native application. It possibly caused by memory taken by native application is greater than Harmonik (only execute and grab the output).

Table 4. Comparison of Average Execution Time (Harmonik vs Native Application)

No	IDE	Prog. Lang.	P1 (hh:mm:ss)	P2 (hh:mm:ss)	P3 (hh:mm:ss)	P1+P2+P3 (hh:mm:ss)	△1 (sec)	△2 (sec)
1	Harmonik	C	00:00:01	00:01:00	00:00:01	00:01:02	+17	+15
	Native			00:01:17				
2	Harmonik	C++	00:00:01	00:01:29	00:00:01	00:01:31	+32	+30
	Native			00:02:01				
3	Harmonik	C#	00:00:01	00:01:00	00:00:01	00:01:02	+78	+76
	Native			00:02:18				
4	Harmonik	Java	00:00:01	00:00:41	00:00:01	00:00:43	+9	+7
	Native			00:00:32				
5	Harmonik	Python	00:00:01	00:00:40	00:00:01	00:00:42	+17	+15
	Native			00:00:57				

Note: Native C, C++, and C# are using Visual Studio 2013; Native Java is using Eclipse Neon; Python is using IDLE

6. Conclusion

The conclusion of this research shows that Harmonik as one of WIDE can help users to learn programming language anywhere, anytime, and with any device. In addition, Harmonik can also motivate users to learn the programming language better. As for the execution speed, the use of Harmonik tends to be relatively bit faster than the use of native application. Harmonik and other WIDE researches slowly but surely will remove the curse that learning programming languages depend on a particular platform and break the myth that coding is difficult to do via mobile devices. Further research is suggested to measure the impact of WIDE for learning outcome (value and passing rate) improvement in learning programming language.

7. Acknowledgement

This work is funded by the Ministry of Research, Technology, and Higher Education of the Republic of Indonesia through Doctoral Dissertation Grant (contract number of 039A/VR.RTT/VI/2017); and supported by Doctor of Computer Science, Bina Nusantara University.

8. References

1. Wu L, Liang G, Kui S, Wang Q. CEclipse : An Online IDE for Programing in the Cloud. 2011 IEEE World Congress on Services (SERVICES), IEEE. 2011 Jul.
2. Ansari AN, Patil S, Navada A, Peshave A, Borole V. Online C/C++ Compiler using Cloud Computing. 2011 International Conference on Multimedia Technology (ICMT), IEEE. 2011 Jul.
3. Lautamäki J, Nieminen A, Koskinen J, Aho T, Mikkonen T. CoRED – Browser-based Collaborative Real-Time Editor for Java Web Applications. Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, ACM. 2012 Feb.
4. Kats LCL, Vogelij RG, Kalleberg KT, Visser E. Software development environments on the web: a research agenda. Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming and software, ACM. 2012 Oct.
5. Emani S, Pokale NB, Chetwani A, Patwari A. Web Based ‘C’ IDE: Approach. International Journal on Computer Science and Engineering (IJCSE). 2012; 4(3).
6. Aho T, Ashraf A, Englund M, Katajamaki J, Koskinen J, Lautamaki J, et al. Designing IDE as a Service.

Communications of Cloud Software. 2011; 1(1).

7. Tambe SB, Sutar S, Nirmal MD. C/C++ Cloud Compiler Using MainFrame. International Journal of Computer Technology and Electronics Engineering (IJCTEE). 2013; 3.
8. Garawad KM, S. G..NET Compiler using Cloud Computing. International Journal of Research in Engineering and Technology. 2014 May; 3(5).
9. Gaikwad T, Dhavale P, Gaware K, Shivale N. Web Based IDE. International Journal of Research in Information Technology. 2014 Apr; 2(4).
10. Dutta M, Sethi KK, Khatri A. Web Based Integrated Development Environment. International Journal of Innovative Technology and Exploring Engineering. 2014; 3(10).
11. Mohammed S, Abdelazziz AM. WIDE an interactive Web integrated development environment to practice C programming in distance education. 2013 1st international conference of the Portuguese Society for Engineering Education (CISPEE), IEEE. 2013 Oct.
12. Yulianto B, Prabowo H, Kosala R. Comparing the effectiveness of digital contents for improving learning outcomes in computer programming for autodidact students. Journal of e-Learning and Knowledge Society. 2016; 12(1).
13. B. Yulianto, H. Prabowo, R. Kosala and M. Hapsara. MOOC architecture model for computer programming courses. 2016 International Conference on Information Management and Technology (ICIMTech), IEEE. 2016.
14. Jenkins J, Brannock E, Dekhane S. JavaWIDE: innovation in an online IDE: tutorial presentation. Journal of Computing Sciences in Colleges. 2010; 25(5).
15. Gadhikar LM, Vincent D, Mohan L, Chaudhari MV. Implementation of Browser Based IDE to Code in The Cloud. International Journal of Advances in Engineering & Technology. 2012 Nov; 5(1).
16. Gadhikar LM, Mohan L, Chaudhari M, Sawant P, Bhusara Y. Browser Based IDE to Code in the Cloud. New Paradigms in Internet Computing, Springer Berlin Heidelberg. 2013.
17. Deursen Av, Mesbah A, Cornelissen B, Zaidman A, Pinzger M, Guzzi A. Adinda: A Knowledgeable, Browser-Based IDE. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, ACM. 2010 May; 2.
18. Goldman M, Little G, Miller RC. Collabode: Collaborative Coding in the Browser. Proceedings of the 4th international workshop on Cooperative and human aspects of software engineering, ACM. 2011 May.
19. Goldman M, Little G, Miller RC. Real-time collaborative coding in a web IDE. Proceedings of the 24th annual ACM symposium on User interface software and technology, ACM. 2011 Oct.
20. Cook CT, Harton H, Smith H, Sitaraman M. Specification Engineering and Modular Verification Using a Web-Integrated Verifying Compiler. 2012 34th International Conference on Software Engineering (ICSE), IEEE. 2012 Jun.
21. Cook CT, Carver JC, Hollingsworth J. Specification and reasoning in SE projects using a Web IDE. 2013 IEEE 26th Conference on Software Engineering Education and Training (CSEE&T), IEEE. 2013 May.
22. Abdulla S, Iyer S, Kutty S. Cloud Based Compiler. International Journal of Students' Research in Technology & Management. 2015; 1(3).
23. Doke P, Shingote S, Kalbhor S, Singh A, Yeole H. Online C, C++, Java Compilers Using Cloud Computing - A Survey. International Journal of Advances in Engineering science and Technology. 2013.
24. Raut N, Parab D, Sontakke S, Hanagandi S. Cloud Documentation and Centralized Compiler for Java & PHP. International Journal Of Computational Engineering Research. 2013; 3(3).
25. Pabitha M, Selvakumar T, Devi SP. An Effective C, C++, PHP, Perl, Ruby, Python Compiler using Cloud Computing. International Journal of Computer Applications. 2013; 69(7).
26. Harshal S, Chakrapani R, Ajay A, Sharad R. Compiler as Service over Cloud. International Journal of Computer Applications. 2013; 70(1).
27. Patel M. Online Java Compiler Using Cloud Computing. International Journal of Innovative Technology and Exploring Engineering (IJITEE). 2013.

28. Tran HT, Dang HH, Do KN, Tran TD, Nguyen V. An interactive Web-based IDE towards teaching and learning in programming courses. 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE), IEEE. 2013 Aug.
29. Arshad AM, Arshiya K, Sana S, Zainab M. Compilers on Cloud. International Journal of Engineering Research and Technology, ESRSA Publications. 2013 Sep; 2(9).
30. Tillmann N, Moskal M, Halleux Jd, Burckhardt S, Ball T, Bishop J. TouchDevelop: create rich mobile apps on touch devices. Proceedings of the 1st International Conference on Mobile Software Engineering and Systems, ACM. 2014 Jun.
31. Fahndrich M. Lessons from a web-based IDE and runtime. Proceedings of the ACM SIGPLAN 2014 Workshop on Partial Evaluation and Program Manipulation, ACM. 2014 Jan.
32. Verma A, Garg N. Online Java Compiler Using Cloud Computing. International journal of engineering technology, Management and Applied Sciences. 2014; 2(6).
33. Nguyen V, Dang HH, Do KN, Tran TD. Learning and practicing object-oriented programming using a collaborative web-based IDE. 2014 IEEE Frontiers in Education Conference (FIE) Proceedings. 2014.
34. Hegde R, K. K. Improved Interaction in Web-Based Cloud IDE. International Journal of Innovative Technology and Exploring Engineering (IJITEE). 2014 Sep; 4(4).
35. Mutiara AB, Refianti R, Witono BA. Developing a SAAS-Cloud Integrated Development Environment (IDE) for C, C++, and Java. Journal of Theoretical and Applied Information Technology. 2014; 68(1).
36. Sun Y, Chen D, Xin C, Jiao W. Automating Repetitive Tasks on Web-Based IDEs via an Editable and Reusable Capture-Replay Technique. 2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSAC), IEEE. 2015 Jul; 2.
37. Bazaz Y, Rajput H, Yadav A, Kojar M, Pranav. The Cloud Based Compiler for Multi Language. The International Journal of Science and Technoledge. 2015; 3(3).
38. Santos A, Farias M, Rocha G, Pereira MV, Farias CMd, França TC, et al. Web2Compile-CoT: A Web IDE for the Cloud of Things. International Conference on Internet and Distributed Computing Systems, Springer International Publishing. 2015 Sep.
39. Kurniawan A, Kurniawan A, Soesanto C, Wijaya JEC. CodeR: Real-time Code Editor Application for Collaborative Programming. Procedia Computer Science. 2015; 59.
40. Jayaraju P, Prakash V. Web Based Interface Implementation for: Ruby, Perl, Python, VB & HTML. International Research Journal of Engineering and Technology (IRJET). 2015 Sep; 2(6).
41. Jayaraju P, Prakash V. Survey on Web Based Interface. International Journal of Advance Research in Computer Science and Management Studies. 2015 Sep; 3(9).
42. V. SC, K. DC, P. SR. Online C, C++, Java Compilers Using Cloud Computing. International Journal of Computer Science and Mobile Computing. 2015 Aug; 4(8).
43. Jayaraju P, Prakash V. Web Based IDE Implementation for C, C++, C#, VB, Java, Perl, Python, Ruby, HTML, CSS, JavaScript. International Journal of Innovative Science, Engineering & Technology. 2015; 2(10).
44. Ghorashi S, Jensen C. Jimbo: A Collaborative IDE with Live Preview. 2016 IEEE/ACM Cooperative and Human Aspects of Software Engineering (CHASE), IEEE/ACM. 2016 May.
45. Škorić I, Pein B, Orehovački T. Selecting the most appropriate web IDE for learning programming using AHP. 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). 2016 May.
46. Armendariz D, Malan DJ, Onken N. A Web-Based IDE for Teaching with Any Language (Abstract Only). Proceedings of the 47th ACM Technical Symposium on Computing Science Education, ACM. 2016 Feb.
47. Kawale R, Soni P, Suryawanshi G, Balbudhe P. Online Editor for Compiling and Executing Different Languages Source Code. International Journal of Advanced Research in Computer Science and Software Engineering. 2016; 6(3).