A "*NEW NORMAL*" APPROACH IN POST-COVID19 ERA: ONLINE HEALTHCARE TESTING STRATEGY

Joydeep Dey

State Aided College Teacher & Former HoD,
Department of Computer Science,
M.U.C. Women's College, Burdwan, West Bengal, India
Email: joydeepmcabu@gmail.com

Abstract

The novel corona virus had shattered the entire living world. Its fast spreading transmission is still not under control. In this new normal post- Corona Virus Disease (COVID), Telemedicine is the safest mode of getting health services being confined within homes. Patients can easily consult their doctors through telemedicine software. Thus, the risks of getting COVID attacks at hospitals are completely eradicated. It ensures treatments along with social-distancing parameters. Testing is an integral component of such telemedicine software. White box testing plays a pivotal in developing such online healthcare software. It is done with intent to locate all hidden errors inside the Telemedicine system. This paper presents a testing strategy on online healthcare software in this post COVID context. It is useful for the software engineers who are engaged in developing such COVID online healthcare software for preserving the patients' data security.

Keywords*: COVID, Telemedicine, White Box Test, Control Flow Graph, Adjacency Matrix*

INTRODUCTION

Corona virus evolved to be the cause of disruptions at all corners of the world, ever since its outbreak in Wuhan of China in November, 2019. Earlier, Wuhan has been considered as the epicenter of this novel corona virus. The Indian first patient was detected in Kerala on 30th January, 2020 having a travel history from China, the then epicenter of corona virus. Pandemic due to this novel corona virus has been declared by the World Health Organization on 11th March, 2020 [1]. The total numbers of COVID affected patients are increasing at an exponential rate, which is quite high. The novel corona virus is immensely affecting loads of peoples in the world. It is a newest form of virus [2-3]. Patients are suffering a lot in this pandemic era. On account to their safety and preventive measures, they are advised to stay at their residence. Now the best option for them is to have telemedicine services in this post COVID time. On the one hand patients are not exposed to COVID attacks, and on the other hand, healthcare services are being provided through virtual platforms to them. It is the safest mode to resist the corona virus rapid transmission in the "New Normal" post-COVID era. Unless very acute emergencies, patients are advised to grab the online healthcare facilities of telemedicine during the global COVID pandemic. Telemedicine refers to the act of providing medical care and support to the

remote patients through digital connections [4]. The terms like telehealth, telecare, E-health, virtual health, electronic follow-ups, mobile health, etc are the subset of the telemedicine system. Every term has its own importance and utility in the context of providing online health services. Patients can take the online consultation with doctors by the help of Internet and computers or mobile phones [5-6].

The objectivity of this paper is addressing the testing issues in the COVID Telemedicine software during this global pandemic era. It is necessary to curtail the spread of the coronavirus. In this paper, a brief white box testing procedure analysis on COVID telemedicine system has been shown. The rate of COVID spike is under rapid acceleration. Telemedicine systems are being used on hyper digital basis. The security features are to be considered for preserving the patients' medical confidentiality. During the pandemic, the use of online communications has been increases exponentially, so as the art of intruding too inside the networks. Intruders are active to steal the medical data for data manipulation purposes. There needs to safeguard the patients' medical data indeed. To ensure this, white box testing has been used to determine the structural faults lying inside the COVID telemedicine software. It detects the hidden flaws and defects inside the COVID telemedicine software. The overall testing of such COVID Telemedicine software is an entire process to accomplish the functional and non functional aspects. White box testing includes path coverage, graph based complexity, test case generation, linearly dependent paths. The main aim is to ensure correctness, performance, reliability, and quality assurance [7]. White box COVID Telemedicine testing is a process of investigation at its internal logical and physical structure at the coding segment [8]. It is needed to have sound knowledge of the programming skills.

LITERATURE SURVEY

Barr E. T. et al. [9] have tested the oracle testing cases deployed in their paper. The bottleneck problems of the oracle system that hinders the total testing performances are the common issues dealt by them. They had given a comprehensive survey in the available approaches to test the oracle system, and to analyze their trends in the domain of software engineering. Nichola D. et al. [10] had done a grey –box functional testing for the automation of the railways system security. William N. et al. [11] had proposed a novel method for the automated structural testing path criteria. They had done dynamic analysis on different C programming language codes. Bounimova E. et al [12] had done white box fuzz testing. It had leveraged symbolic constraints on the uncovered paths lying in the software codes. They had discovered potential security vulnerabilities. Bhansali S. et al. [13] had proposed a runtime framework which includes automated data collection, process independent, and users can trace the execution of a program. Their framework had operated on a compression method with less runtime overhead, and thus, minimizing the tracking size. It enabled to know the detailed behavioral knowledge of the software. Petiot G. et al. [14] had explained an incremental verification method to generate test

cases. They had used PATHCRAWLER on various C programs. Jamrozik K. et al. [15] had proposed an augmented DSE (Dynamic Symbolic Execution) that may generate test case sets with path conditional and logical covering condition. To inspect the system's behavior for automated oracles test cases and to generate test case for regression test set, unit testing is the basic methodology that was used.

Singh A. et al. [16] said that it is better to keep sugar level of diabetic patients within normal limits in this time of post-COVID19 context. Using the telemedicine, the diabetic patients can consult their endocrinologists electronically. Hollander J. et al. [17] had proposed that telehealth system should be developed so that the patients with average or high risks can be nursed in triaged channels. It should be done by tele-video calls with prior appointments. Weight J. et al. [18] had suggested to aide services using phone, telemedicine and other internet based technologies. Training should be arranged on the implementation of ICT to the doctors and patients.

SECURITY PERSPECTIVE

Any COVID telemedicine software must ensure its functional potentials. It must follow the standards of a model software testing. The most common and formal software security testing may be classified into two parts. They are: a) proposition substantiating, and b) perfect code inspection. Mathematical theorems are used to show the transformation of a module code into rationalized formula. The axioms and rules are used to demonstrate the validity of the mathematical theorem. The vital aspect is to make the COVID Telemedicine software safe from the act of external intruding. In the same context, structural testing is an integral component that needs to be carried out. In any telemedicine software, enormous patients' data are flowing in the public network. Strong and robust telemedicine software means how to resist against the malicious attackers.

PROPOSED METHODOLOGY

COVID Telemedicine Testing Spectrum

COVID Telemedicine software testing is conducted at every stages of its SDLC (Software Development Life Cycle) with its different objectives and strategies. The following is the COVID telemedicine software spectrum that has been widely used.

***Unit testing*:** This testing is done on the individual units separately. All the modules of the COVID telemedicine are tested one by one in this way [19].

Integration testing: Here, the COVID telemedicine modules are integrated simultaneously to validate its functioning, and emphasis on the interfaces given in the structural design paradigm [19].

***System testing*:** It ensures that the COVID telemedicine software performs all the functional aspects at the end-to-end point. It helps to maintain the software in future better [19].

***Acceptance testing*:** The hospital management conducts such type of testing on the COVID telemedicine software that was built by the development team. The objective of this testing is to test whether the COVID telehealth software meets its requirements or not [19].
Regression Testing: this kind of testing is done only after making the COVID telemedicine software error free. It ensures the reliability of COVID systems [19].
Alpha Testing: The COVID telemedicine software is tested out by the development team.
Beta Testing: The COVID telemedicine software is tested out by the friendly team within the development company.
Functional Testing: It is done on the complete COVID telemedicine system. It is done to verify all the needed healthcare services provides or not [19].

Different White Box Techniques on COVID Telemedicine
The problem of intractability is solved by the following three white box testing techniques on COVID Telemedicine software [12].

- ✓ Statement coverage
- ✓ Branch coverage
- ✓ Path coverage

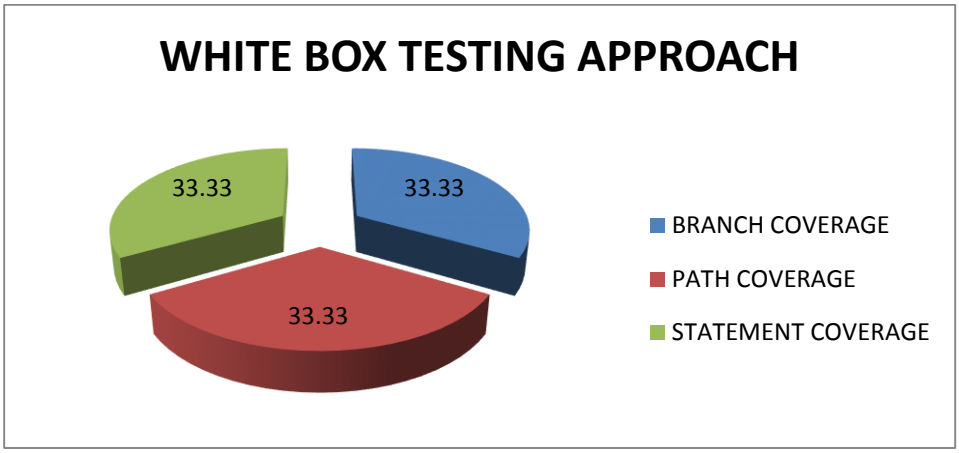

Fig 1: Three White Box Testing techniques on Post-COVID Online Healthcare Software

***Statement Coverage*:** It is also called as segment coverage. Only the true conditions are covered in this technique. It involves the execution of all the statements in the COVID Telemedicine modules minimum once.

Branch Coverage: It is also called as decision coverage. The main objective of this technique is to ensure that there should be at least once execution for each one of the possible combinations. Thus, all the statements of the COVID Telemedicine are thereby reachable codes. The formula used here is as follows.

$$BT = \frac{ND}{TD} * 100\ \% \quad \ldots\ldots\ (1)$$

; where, BT= Branch Test, ND= Number of tested decision results, TD = Total number of decision results.

Path coverage: Here, all the paths of the COVID Telemedicine are tested at least once. It is a type of structural testing which involves find all possible execution paths based on the COVID Telemedicine modules. The objective is to find all possible hidden errors inside it. Different coverage techniques under this scheme are Loop Coverage, Basis Path Coverage, and Data flow Coverage [8].

PROPOSED FLOW DIAGRAM OF COVID TELEMEDICINE

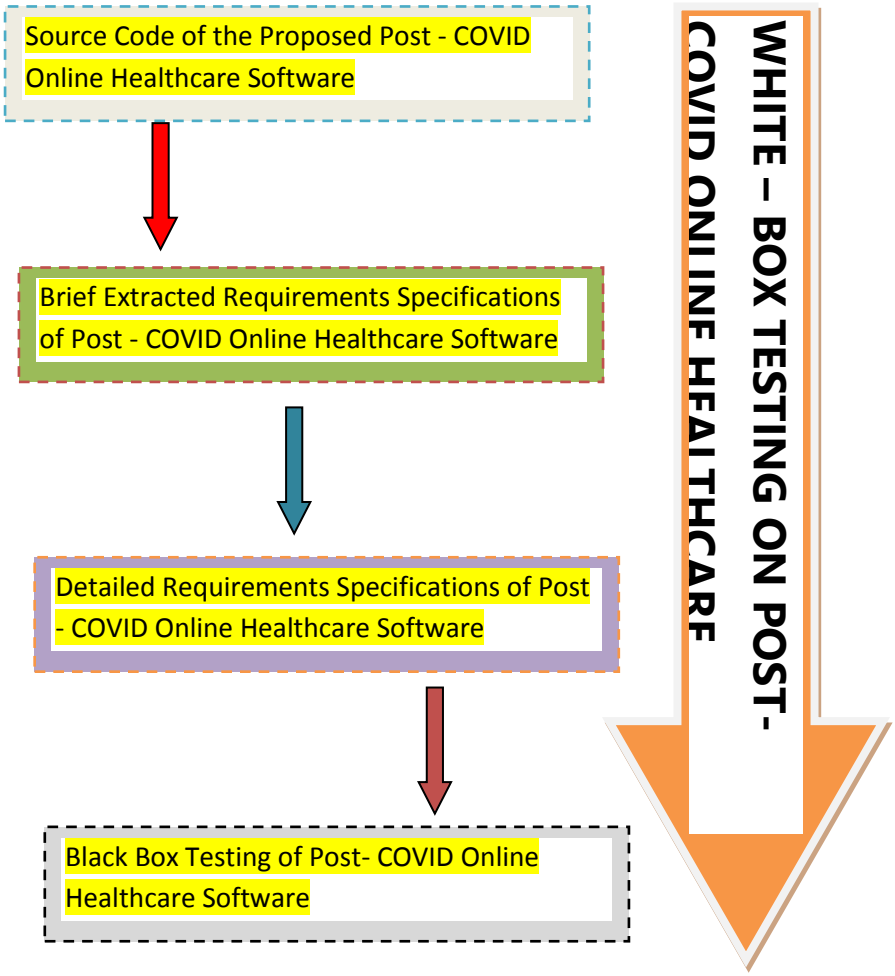

Fig. 2: Testing Flow of COVID Online Healthcare System

RESULTS SECTION

A Control Flow Graph (CFG) based on the post-COVID online healthcare software has been used as base. A CFG is a directed graph comprised of N and E. N denotes the finite set of nodes, and E denotes the finite number of directed edges. It may be drawn as per the following four steps [20].

a. **Nodes Distribution**: All the participating nodes of the post-COVID Telemedicine software are serially arranged first.
b. **Ordering of Nodes**: All the said participating nodes of the post-COVID Telemedicine software are sorted in ascending sequence.
c. **Node tracing**: On the basis of nodes' functionalities, myriad structural testing of the post-COVID Telemedicine modules is created.
d. Notation of Edges: Edges are drawn from nodes to nodes for the sequential execution of the post-COVID telemedicine modules.

In the era of Post-COVID New Normal, using the following online healthcare modules, the average number of patients availed is given below. Module for the average number of patients attended COVID Telemedicine is given in the following algorithm 1. Module for finding a doctor by the patient is given in the following algorithm 2.

Proposed Algorithm* 1**: ***Module for COVID Telemedicine Average patients

$Input(s)$: $Number\ of\ Patients\ (data), Number\ of\ Days\ (Nd)$

$Outputs$: $Average\ No.of\ Patients(AVG)$

1. $float\ COVID\_TEL\_AVG\ (\ float\ *\ data,\ int\ Nd)$
2. {
3. $int\ i, Sum = 0$
4. $for\ (i\ =\ 0\ ;\ i\ <\ Nd;\ i++)$
5. $\quad Sum\ =\ Sum\ +\ Data[i]$
6. $return\ (\ Sum\ /\ Nd)$
7. }

Proposed Algorithm* 2**: ***Module for Finding Doctor by patients

$Input(s)$: $Set\ of\ Doctors(docset), Doctor\ Name(\ Key)$

$Outputs$: $Online\ Appointment\ or\ Error\ Message$

1. $void\ Find\_Doc\_COVID\_Tel\ (String\ *\ docset[\ ],\ int\ no, String\ Key)$

2. {
3. $int\ i$
4. $for(\ i = \ 0;\ i\ <\ no\ ;\ i++)$
5. {
6. $if(\ String.Equals(\ Key, docset(i), 0)$
7. $Flag = 1$
8. $Book\_Appointment(\ )$
9. $Exit\ Loop(\ )$
10. }
11. $if(\ Flag\ == 0)$
12. print " No Doctor Found Online"
13. }

Control Flow Graph: Average Patients of COVID Telemedicine

The CFG for the above figure 3 can be drawn as below.

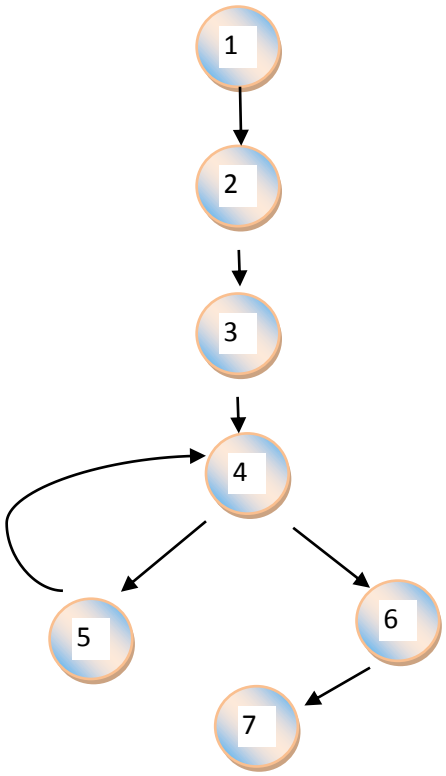

Fig. 3: CFG for COVID_AVG_PATIENT

Control Flow Graph: Searching Doctors in COVID Telemedicineby the Patients

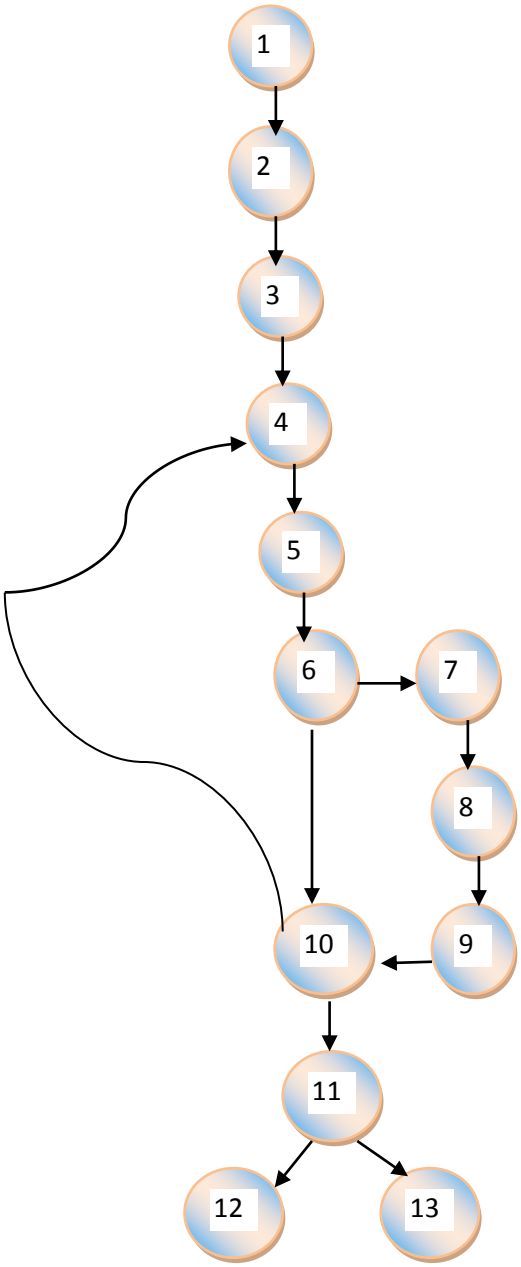

Fig. 4: CFG for Finding Doctor by Patient

Simulation of Control Flow Graph in Online Healthcare Software

Control flow graphs are the basic component for code optimization of any online software. Static analysis is made on the CFGs available. The same is true for online healthcare software in this post COVID era too. It is used to encapsulate data per each function module. Syntactic structures are well observed through it. Test case generations for online healthcare systems are done on the CFGs. Different software parameters were taken into result analysis then. The objectivity of this section is to make the healthcare software error free from its accurate functional perspectives.

Representation in terms of Adjacency Matrix

An adjacency matrix of a CFG is a square matrix of the order (N*N), where N denotes the total number of nodes that took part in the CFG. Each cell of the adjacency matrix is filled with either 0 or 1. If there exists a direct link between the pair of nodes, then that allocated cell will be filled with 1, otherwise 0. In the same way, table 1 contains the adjacency matrix of COVID_AVG_PATIENT. The following table 2 contains the adjacency matrix for searching a doctor module by the patients.

Tab. 1: Adjacency matrix of COVID_AVG_PATIENT with respect to Fig 3.

NODE	N1	N2	N3	N4	N5	N6	N7
N1	0	1	0	0	0	0	0
N2	0	0	1	0	0	0	0
N3	0	0	0	1	0	0	0
N4	0	0	0	0	1	1	0
N5	0	0	0	1	0	0	0
N6	0	0	0	0	0	0	1
N7	0	0	0	0	0	0	0

Tab. 2: Adjacency matrix of FIND_DOC with respect to Fig 4.

NODE	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13
N1	0	1	0	0	0	0	0	0	0	0	0	0	0

N2	0	0	1	0	0	0	0	0	0	0	0	0	0
N3	0	0	0	1	0	0	0	0	0	0	0	0	0
N4	0	0	0	0	1	0	0	0	0	1	0	0	0
N5	0	0	0	0	0	1	0	0	0	0	0	0	0
N6	0	0	0	0	0	0	1	0	0	1	0	0	0
N7	0	0	0	0	0	0	0	1	0	0	0	0	0
N8	0	0	0	0	0	0	0	0	1	0	0	0	0
N9	0	0	0	0	0	0	0	0	0	1	0	0	0
N10	0	0	0	0	0	0	0	0	0	0	1	0	0
N11	0	0	0	0	0	0	0	0	0	0	0	1	1
N12	0	0	0	0	0	0	0	0	0	0	0	0	0
N13	0	0	0	0	0	0	0	0	0	0	0	0	0

From the above stated tables 1 and 2, the cyclomatic complexity can be calculated as below.

Tab. 3: Connection List with respect to Table 1.

NODE	N1	N2	N3	N4	N5	N6	N7	CONNECTION (C)	C-1
N1	0	1	0	0	0	0	0	1	0
N2	0	0	1	0	0	0	0	1	0
N3	0	0	0	1	0	0	0	1	0
N4	0	0	0	0	2	1	0	3	2
N5	0	0	0	1	0	0	0	1	0
N6	0	0	0	0	0	0	1	0	0
N7	0	0	0	0	0	0	0	0	0
								Sum Total=	2

Tab. 4: Connection List with respect to Table 2.

NODE	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	CON(C)	C-1
N1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
N2	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
N3	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0
N4	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0
N5	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
N6	0	0	0	0	0	0	1	0	0	1	0	0	0	2	1
N7	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
N8	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
N9	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
N10	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
N11	0	0	0	0	0	0	0	0	0	0	0	1	1	2	1
N12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sum Total=															2

It is clear from the above mentioned tables 3 and 4 that the cyclomatic complexity of the Control Flow Graph for COVID_AVG_PATIENTS is equals to 2. In this method, the number of links participating to a node is counted. This is done on all the participating nodes of the control flow graph. In each row, such links values are subtracted by one. Lastly, the column sum is done to generate the cyclomatic complexity of the COVID_AVG_PATIENT module. It helps in the white box testing applications. Similarly, the cyclomatic complexity of the Control Flow Graph for FIND_DOC is 2.

CONCLUSION

In this new normal of Post-COVID era, there has been acceleration towards the digital health care systems. COVID Telemedicines are offering the healthcare services to the patients from the remote locations. To have efficient COVID Telemedicine software, white box testing in an integral component. This paper has presented to calculate cyclomatic complexity of post COVID online healthcare module using adjacency matrix. It would be fruitful for the software developers in different phases of the post-COVID SDLC (Software Development Life Cycle).

REFERENCES:

1. World Health Organization. WHO announces COVID-19 outbreak a pandemic; 2020. Available from: http://www. euro.who.int/en/health-topics/health-emergencies/coronaviruscovid-19/news/news/2020/3/who-announces-covid-19-outbreaka-pandemic, accessed on June 03, 2020.
2. Peeri NC, Shrestha N, Rahman MS, Zaki R, Tan Z, Bibi S, et al. The SARS, MERS and novel coronavirus (COVID-19) epidemics, the newest and biggest global health threats: What lessons have we learned? Int J Epidemiol 2020. pii: dyaa033.
3. Ruan S. Likelihood of survival of coronavirus disease 2019. Lancet Infect Dis 2020. doi.org/10.1016/ S1473-3099(20)30257-7.
4. Judd E. Hollander, Brendan G. Carr, Virtually Perfect? Telemedicine for Covid-19, N Engl J Med 2020; 382:1679-1681, April 2020.
5. Ramdas Ransing, Frances Adiukwu, Victor Pereira-Sanchez, Rodrigo Ramalho, Laura Orsolini, André Luiz Schuh Teixeira, Jairo M. Gonzalez-Diaz, Mariana Pinto da Costa, Joan Soler-Vidal, Drita Gashi Bytyçi, Samer El Hayek, Amine Larnaout, Mohammadreza Shalbafan, Zulvia Syarif, Marwa Nofal, Ganesh Kudva Kundadak. (2020) Mental Health Interventions during the COVID-19 Pandemic: A Conceptual Framework by Early Career Psychiatrists. *Asian Journal of Psychiatry* 51, 102085.
6. Angela Barney, Sara Buckelew, Veronika Mesheriakova, Marissa Raymond-Flesch. (2020) The COVID-19 Pandemic and Rapid Implementation of Adolescent and Young Adult Telemedicine: Challenges and Opportunities for Innovation. *Journal of Adolescent Health*.
7. Mohd. Ehmer Khan, "Different Approaches to White Box testing Technique for Finding Errors," IJSEIA, Vol. 5, No. 3, pp 1-13, July 2011.
8. Mohd. Ehmer Khan, "Different Forms of Software Testing Techniques for Finding Errors," IJCSI, Vol. 7, Issue 3, No 1, pp 11-16, May 2010.
9. Barr, E.T., Harman, M., McMinn, P., Shahbaz, M., Yoo, S. (2015). The oracle problem in software testing: A survey. IEEE Transactions on Software Engineering, 41(5), 507–525. https://doi.org/10.1109/TSE.2014.2372785.
10. D. Nicola et al. "A grey-box approach to the functional testing of complex automatic train protection systems." Dependable Computing-EDCC 5. Springer Berlin Heidelberg, 2005. 305-317.
11. Williams, N., Marre, B., Mouy, P.: On-the-Fly Generation of K-Path Tests for C Functions. In: ASE 2004. IEEE, Los Alamitos (2004).

12. Bounimova, E., Godefroid, P., Molnar D. (2013). Billions and billions of constraints: Whitebox fuzz testing in production. In: International conference on software engineering, IEEE, ICSE, pp. 122–131. https://doi.org/10.1109/ICSE.2013.6606558.
13. S. Bhansali, W. Chen, S. De Jong, A. Edwards, and M. Drinic. Framework for instruction-level tracing and analysis of programs. In Second International Conference on Virtual Execution Environments VEE, 2006. Pages 154–163https://doi.org/10.1145/1134760.1220164.
14. Petiot, G., Kosmatov, N., Giorgetti, A., Julliand, J.: How Test Generation Helps Software Specification and Deductive Verification in Frama-C. In: Seidl, M., Tillmann, N. (eds.) TAP 2014. LNCS, vol. 8570, pp. 204–211. Springer, Heidelberg (2014).
15. Jamrozik, K., Fraser, G., Tillman, N., de Halleux, J.: Generating Test Suites with Augmented Dynamic Symbolic Execution. In: Veanes, M., Viganò, L. (eds.) TAP 2013. LNCS, vol. 7942, pp. 152–167. Springer, Heidelberg (2013).
16. Singh A. , Gupta R. ,Ghosh A. , Misra A., Diabetes in COVID19 Prevalence, Pathophysiology, Prognosis and practical considerations' , Diabetes & Metabolic Syndrome:Clinical Research & Reviews, 2020 Apr 9;14(4):303-310. doi: 10.1016/j.dsx.2020.04.004.
17. Judd E. Hollander, Brendan G. Carr,'Virtually Perfect? Telemedicine for COVID19',The New England Journal of Medicine. April 30 2020; 382:1679-1681 DOI: 10.1056/NEJMp2003539.
18. Wright J., Caudill R.,Remote Treatment Delivery in Response to theCOVID19 Pandemic, Psychotherapy and Psychosomatics, 2020;89:130–132 https://doi.org/10.1159/000507376.
19. P. Jorgensen, Software testing: a craftman's approach, CRC Press, 2002. p. 359.