Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 44 No. 9
September 2023

Architecture Which Suits You Best

Nishchal,
Department of Computer Science Engineering,
Chandigarh University Mohali, India

Mohammad Shuaib Khan,
Department of Computer Science Engineering,
Chandigarh University Mohali, India

Sahil Verma,
Uttaranchal University, Dehradun, India
Email: sahilverma@ieee.org

Ajay Singh
Uttaranchal University, Dehradun Email: profajaysingh@uumail.in

Abstract—The motive of this paper is to help beginners to take initial steps in building their own web application architectures. We first grow the clairvoyance for software architecture by briefly discussing terms and techniques used in building a production level application. On the basis of this clairvoyance, we show some of the evergreen models of software architecture and design Model-View-Controller, Model-View- View Model and Model-View-Provider. We discuss the problems faced while and after building your web application. We discuss concepts of Object-Oriented Programming helping in building web application architectures. We will be modifying these architectures by adding different elements according to ease of use. We temper them and use them. We discuss the beginner steps to build scalable web applications. We culminate by showing some of the advantages of our way to web application architectures, condensing our experience, and relating our way to others' present work.
Index Terms—Clean Architecture, Flutter, Dart

I. INTRODUCTION

Building a project and making it work for once is easy but building a project and making it scalable makes a software developer. While developing any kind of web application we include any business logic with UI. There is no doubt that it will work, but what if we have to add another feature in the same. Then we do changes in the same logic and it will become very messy. It will take time to render and make your application heavy. All this results in a bad user experience [22]. There are many books related to software architecture but it will take time to read all those and then decide what to implement. So, the solution to this is very simple: "Start your architecture by separating logic from applications' User Interface". While building your application just start keeping your business logic separate [23[.

Where you can do enhancements without thinking of the user interface becoming a mess. We will be taking references from MVC, MVVM as shown in Fig 4, MVP as shown in Fig 3 and using them in Flutter (Dart is used in flutter). We will be using micro services to interact with our database and other third-party software [24,32].

We use structures to store data. This will be implemented in layers like data, business logic, State, and Services. Let's take an example of a burger, there are many layers in it, like a layer of bread then above cheese then vegetable and then bread again. But what if vegetables are above the slices of bread. It will not make sense, right? So just like that, every layer has its position and

function. This structure of layers we can implement universally. We discuss every layer in detail and how Object-Oriented Programming comes into play as well.

II. PROBLEM DEFINITION

Most of the modern developers after acquiring some skills start building the project but most of the times they either don't know about the importance of the proper architecture or they just buy a lie: "We can clean it up later; we just have to get to market first!" And if they want to build that project into some start-up then it's surely wrong approach because, actions never do get cleaned up again, because trade pressures never lessen. Developers think that it's a hassle to implement things according to architecture, but it's wrong. Being fast doesn't make you software developer, being steady and consistent makes you a better software developer. Because as a developer you have to farsighted regarding the software they are building. Writing grimy code only makes developer to go fast in the short run, and just slows them down in the long run (software developing is a marathon not a sprint). Jason conducted a remarkable experiment over a duration of six days. Each day he completed a simple program to convert integers into Roman numerals as shown in Fig 1. His work was done when the predefined set of acceptance tests passed for the problem. Each day the task took a little less than half hour.

Jason used a well-known cleanliness discipline named test- driven development (TDD) on the first, third, and fifth days.

So, it is clear how implementation of proper architecture can save time, increase the quality of code and increase your productivity. But how can a beginner developer implement in the modern-day frameworks like flutter without using the outside packages?

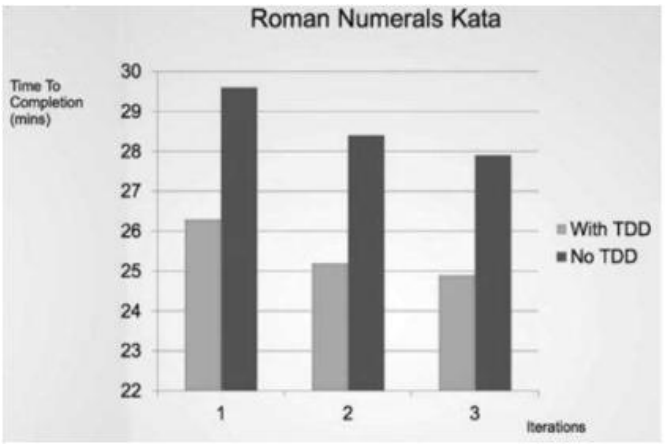

Fig. 1. Roman Numeral Experiment Graph

III. THEORY AND RELATED WORK

A. Model-View-Controller design pattern actually parts the web application into three major parts that are Model- View-Controller.

B. Model

Model contains data or model of the schema which is required to display in the User Interface. Model actually is a collection of classes that defines the business logic (business model and the data model). We make getters according to our business logic and use them in our project accordingly.

C. View

In simple terms view means user interface. View shows the data that is received from the controller as the result of an action. View is completely stateless that means it is the interface by which the user interacts. In MVC pattern View as shown in Fig 2, keeps an eye for any state change and displays updated state. If there is any change it will act according to the code returned in the controller [25,33].

D. Controller

The Controller is in charge of processing the incoming re- quests. It exercises the user's data through the Model and reverts the results to View again. It contains the business logic and all other logic. It works as a middleman between view & model.

E. Model-View-Presenter pattern is somewhat close to the MVC pattern [26,27]. It is procured from MVC pattern, the only difference is that the controller here is the presenter. This pattern parts the web application into three major parts that are Model- View-Presenter.

F. Model

Model contains data or model of the schema which is required to display in the User Interface. Model actually, is a collection of classes that defines the business logic & data (business model and the data model). We make getters according to our business logic and use them in our project accordingly.

G. View

In simple terms view means user interface. View displays the data that is received from the controller as the outcome. View is completely stateless that means it is the interface by which the user interacts. Logic is not present in the view. User actions are made from a view which is different from MVC.

H. Presenter

It is different from Controller in MVC because there the controller is in charge here View is in charge [28]. The Presenter retrieves the request from the user through View, then exercises the user's request through Model, business logic & services then revert the results to the View again. MVP is derived from MVC. It is mostly used in small applications.

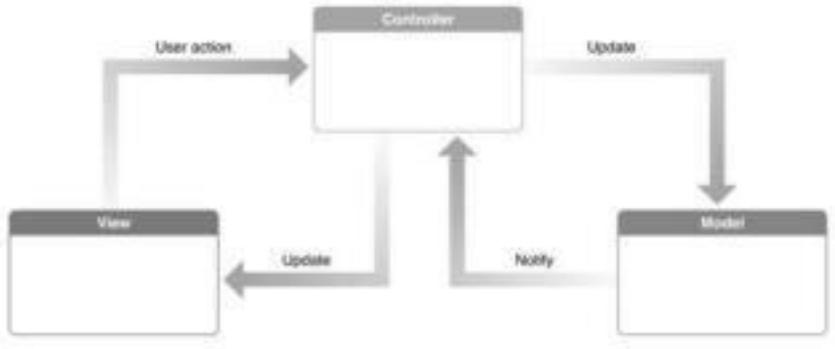

Fig. 2. MVC Diagram

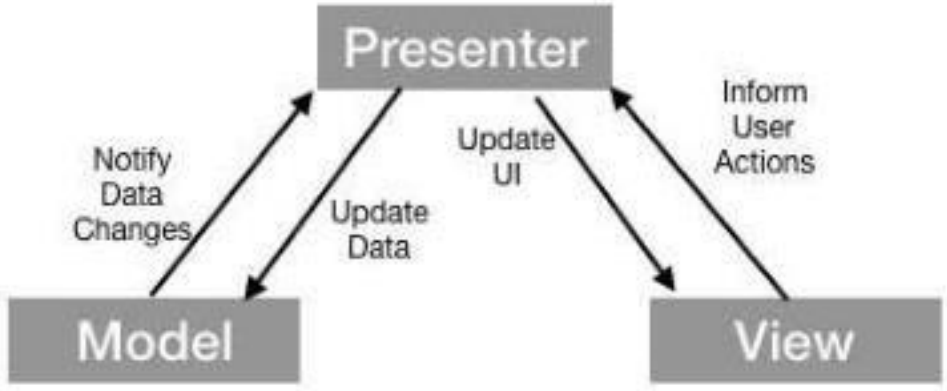

Fig. 3. MVP diagram

I.

Model-View-View Model design pattern is the same as the replacement to MVC and MVP patterns. This framework was introduced in the 1980s. It overcomes the only concern of MVC and MVP that is they helps is modularity. Modules don't have reference to their parent, they only

have relations by observables. The workflow can be seen in the diagram below.

IV. SOLUTION

A. Layer Design

It is important for your project as it will have many business rules so it should be easy to define a class having new features in mind.
There are three main components in which project is di- vided:

1. Presentation layer: The UI that displays the data to the user and shows the states or different forms of data environment Management (publish, env, config).
2. Business layer: It deals with data validation and define business rules & norms.
3. Data Access layer: It is the mechanism that connects the web application to the channel of choice to store the data.

B. Folder Design

It is important while developing a project. It's your folder structure. It means how you are going to divide folders. Below is our approach to divide folders.

C. Models

This layer contains the model of the schema and contains the getter as we required. We will call the get the data from the database and store it here then modify it with the help of get- ter and use it accordingly. It is used when we post the data when we service. It does not contain any logic. It's only used in business logic.

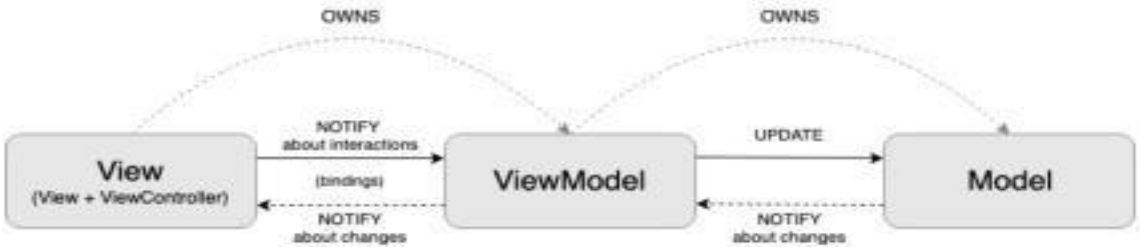

Fig. 4. MVVM Diagram

D. Common (more usage Product)

This layer is created to contain which is used in the whole project and has the same code. For example, your theme layer, localization, structure (network instance, like error), size configuration and constants etc. This layer key points to what the project needs.

E. Manager (Contains the business logic)

Manager contains all the business logic and helps manage the state of the screens. All the business logic will be present in the manager. According to the logic, the state of the screen will change. It behaves as a presenter in MVP but it is different from it. It's true, it interacts with view and request will pass through view but it is also managing state. It is achieved with the help of a provider. We will discuss it in further sections.

F. Functions (Contains some common functions)

It contains some common functions like to alter datetime and much more.

1) Screens (Contains the UI): Screen contains all the state- less screens. It solely comprises UI and nothing else. UI will be shown according to the manager or business logic. It interacts with the user. We can add any screen there and then add its logic in the respective manager without affecting any component.

2) Services (Contains the different Service): Services contain all the services that we are going to use in the project. Services like UserDataService, helps us interact with the database and all other functionality related to the database. In services we can make network image services etc. We can also make cache as a service as well. Now after folder design you need to know how to make it possible and what technology we are using to make it possible [29]. Structure design represents the page design, project design as shown in Fig 5 or any special design. So now we need to decide which state

management we are going to use. There are many like BLoC, MobX, Redux etc: But here we are going to use a provider which is officially backed and provided by google itself.

This provider is very powerful for state management and perfect fit for flutter. So as I said earlier that we are going to take some elements from every existing design pattern and make it our own.
Here how it looks like:-

- Constants
- Model Controller
- View Services

OOPS concept will be used through-out the project
3) Delivery Design: Here it means a method to publish. Most of the time we do not think about this activity and development time gets shot up. So, we can achieve this in two ways.
Continuous Integration (CI): This way we can con-tenuously measure the quality and search-fix the possi- ble bugs in our code (unit test coverage, package build, codebase etc.).
Continuous Delivery (CD): It is a little compound. It does integration processes and deploy servers and deploy environ- ments much more.
We are using:- Application Architecture: CMVCS State Management: Provider
As shown in the Fig 6, everything is clearly divided into folders. This increases the modularity. It also helps in the future prospect. As we can add any feature without worrying about the code failure. It also helps in testing as we can test for the business logic and modularity as well. We can make custom widgets and use them in views. While using custom widget we will make them general as we will generalize abstract classes to avoid the boilerplate. Controllers contain all the business logic and no other folder contains anything related to business logic. For example, In the controller we are using a provider there. It has business logic as well, it is used to manage the state. It is the main idea for the Controller or Manager layer.

Provider Design
The provider is used for state management, and it is backed by Google officially. Here we write all the business logic and change the state of the page according to our needs. When we use this all pages or screens become stateless and they don't have to be rebuilt again and again. It makes our app seamless. Provider work for your business norms and state changes accordingly. It updates the widget according to its respective manager (Controller + Provider). To use this package, you must add it in pub spicy. You can also use streams in the provider to make it more scale-able.

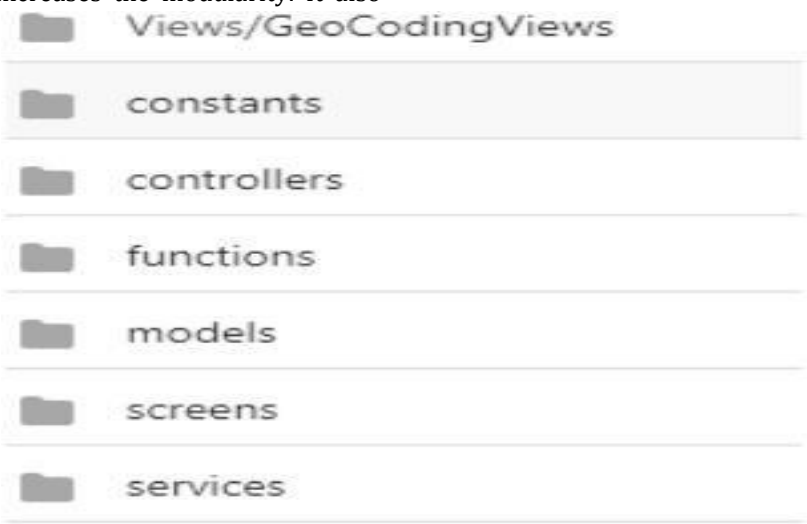

Fig. 5. Project Design

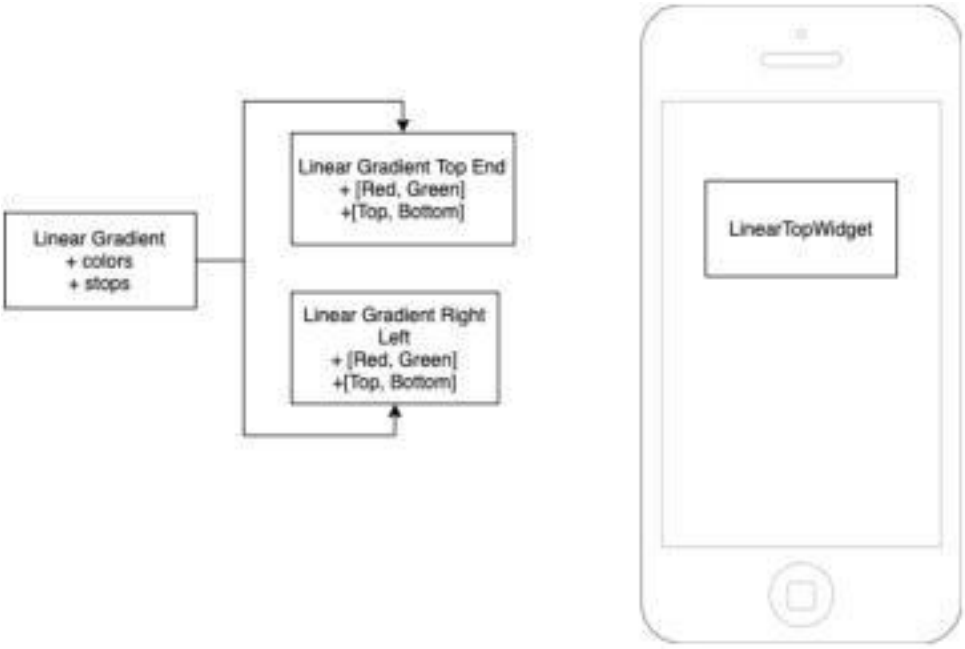

Fig. 6. Flow Control

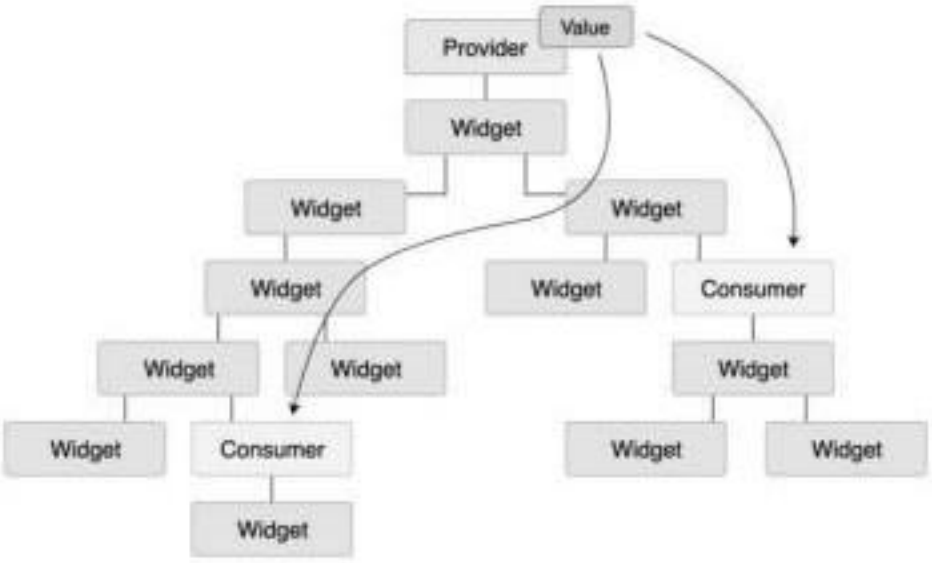

Fig. 7. Provider Tree

We will extend the Change Notifier class in manager [30.31]. We have to initialize this class constructor above in the widget tree. As shown in Fig 7. So, the widget below can use it. It is the most effective and mostly used provider. With the help of this you can tackle many logic and state related problems. You need to notify any function if there is any change in their respective variable, just write a function and after its done just use notify. It can be done by using notify Listeners () function as shown in Fig 8.

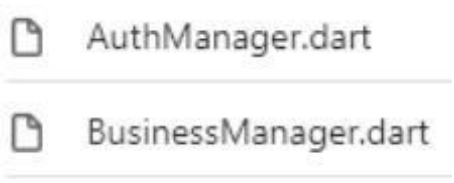

Fig 8: Change Notifier Folder

V. CONCLUSION

So, it is clear how you can start, choose and make your architecture which suits you best. Here we used Provider but there are other methods as well. Just follow which suits you best but make sure to build the next project with proper architecture for good practice. Making the project work once doesn't make you a good developer but making it scalable, reliable one should follow a proper architecture. You don't have to spend sleepless nights to finish reading the entire book collection in one go. But having access to them will definitely help (maybe have a small library at your office/home).

Develop a learning mindset and absorb things from these books as you build things. Most of the time, you may have to unlearn, relearn, and combine some of these concepts over time, so be prepared for that too.

VI. ACKNOWLEDGEMENT

The research for this topic behind this paper would not have been done without the guidance of my mentor, Mohammad Shuaib. His knowledge related to the research paper and the way of presenting their ideas for the paper help us a lot and give us a motivation to complete the paper on time.

I am very grateful to those who helped me during this research. Each of my friends and teachers of my university provide me great personal as well as professional knowledge for writing a research paper at a journal level. Their key learnings and their optimal approaches helped me in writing my very first research paper. They have shown me what a good researcher should be.

I am also thankful to my parents, whose love, guidance and moral support for whatever I pursue. They are the best role models for me. So, I am very grateful to them.

VII. APPENDIX

Uncle Bob's clean architecture is the best way to implement low level designs and maintain the SOLID principles. We haven't talked about it here but it plays a vital role in software development.

You can also use High level designs by following the understanding of SOLID principles in any language you just need to know how the layers are divided and used.

REFERENCES

[1] Cognitive Human-Computer Interaction, Vol. 1 , No. 2 , (2021) : 46-56 (Doi : DOI: https://doi.org/10.54216/JCHCI.010201)

[2] Vijay K, Collaborating The Textual Reviews Of The Merchandise and Foretelling The Rating Supported Social Sentiment, Journal of Cognitive Human-Computer Interaction, Vol. 1 , No. 2 , (2021) : 63 - 72 (Doi : DOI: https://doi.org/10.54216/JCHCI.010203)

[3] R. Venkatesan, Althaaf Shaik, Suraj Kumar, Vipul Guria , Abhishek Raj, Intelligent Smart Dustbin System using Internet of Things (IoT) for Health Care, Journal of Cognitive Human-Computer Interaction, Vol. 1 , No. 2 , (2021) : 73 - 80 (Doi : https://doi.org/10.54216/JCHCI.010204)

[4] P. Kavitha , R. Subha Shini , R. Priya, An Implementation Of Statistical Feature Algorithms For The Detection Of Brain Tumor, Journal of Cognitive Human-Computer Interaction, Vol. 1 , No. 2 , (2021) : 57 - 62 (Doi : DOI: https://doi.org/10.54216/JCHCI.010202)

[5] Sonia Jenifer Rayen, Survey On Smart Cane For Visually Impaired Using IOT, Journal of Cognitive Human-Computer Interaction, Vol. 1 , No. 2 , (2021) : 81 - 85 (Doi : DOI: https://doi.org/10.54216/JCHCI.010205)

[6] J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidanceandcontrol,"Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.

[7] Technology Strategy Patterns: Architecture as Strategy [2018] by Eben Hewitt.

[8] Software Architecture in Practice Book by Len Bass, Paul Clements, and Rick Kazman
[9] Fundamentals of Software Architecture: An Engineering Approach.
[10] Medvidovic, N. and Taylor, R.N., 2010, May. Software architecture: foundations, theory, and practice. In 2010 ACM/IEEE 32nd International Conference on Software Engineering (Vol. 2, pp. 471-472). IEEE.
[11] Sood, M., Verma, S., Panchal, V.K.: Optimal path planning using hybrid bat algorithm and cuckoo search. Int. J. Eng. Technol. 7(4.12), 30–33 (2018).
[12] Kumar, P.; Verma, S. Detection of wormhole attack in VANET. Natl. J. Syst. Inf. Technol. 2017, 10, 71.
[13] Batra, Isha, Sahil Verma, Arun Malik, Kavita, Uttam Ghosh, Joel J. P. C. Rodrigues, Gia Nhu Nguyen, A. S. M. Sanwar Hosen, and Vinayagam Mariappan. 2020. "Hybrid Logical Security Framework for Privacy Preservation in the Green Internet of Things" Sustainability 12, no. 14: 5542.
[14] Gaba S, Verma S. Analysis on Fog Computing Enabled Vehicular Ad hoc Networks. Journal of Computational and Theoretical Nanoscience, 2019, 16(10), pp. 4356–4361.
[15] N. Kaur and S. Verma, "Detection of plant leaf diseases by applying image processing schemes," Journal of computational and theoretical nanoscience (JCTN), vol. 16, no. 9, pp. 3728–3734, 2019. [16] Michael Jouravlev: "Redirect After Post," theserverside.com, August 1, 2004.
[17] Alex Yarmula: "Strong Consistency in Manhattan," blog.twitter.com, March 17, 2016
[18] Jim Gray: "The Transaction Concept: Virtues and Limitations,"at 7th Interna- tional Conference on Very Large Data Bases (VLDB), September 1981.
[19] Mark Seaborn and Thomas Dullien: "Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges," googleprojectzero.blogspot.co.uk, March 9, 2015.
[20]Ben Laurie: "Scheitle, Q., Gasser, O., Nolte, T., Amann, J., Brent, L., Carle, G., Holz, R., Schmidt, T.C. and Wählisch, M., 2018, October. The rise of certificate transparency and its implications on the Internet ecosystem. In Proceedings of the Internet Measurement Conference 2018 (pp. 343-349).Laurie, B., 2014. Certificate transparency. Communications of the ACM, 57(10), pp.40-46.
[21] Srinivasan, K., Garg, L., Datta, D., Alaboudi, A. A., Jhanjhi, N. Z., Agarwal, R., & Thomas, A. G. (2021). Performance comparison of deep cnn models for detecting driver's distraction. CMC-Computers, Materials & Continua, 68(3), 4109-4124.
[22] Khalil, M. I., Jhanjhi, N. Z., Humayun, M., Sivanesan, S., Masud, M., & Hossain, M. S. (2021). Hybrid smart grid with sustainable energy efficient resources for smart cities. sustainable energy technologies and assessments, 46, 101211.
[23] A. Almusaylim, Z., Jhanjhi, N. Z., & Alhumam, A. (2020). Detection and mitigation of RPL rank and version number attacks in the internet of things: SRPL-RP. Sensors, 20(21), 5997.
[24] Shah, I. A., Sial, Q., Jhanjhi, N. Z., & Gaur, L. (2023). The Role of the IoT and Digital Twin in the Healthcare Digitalization Process: IoT and Digital Twin in the Healthcare Digitalization Process. In Digital Twins and Healthcare: Trends, Techniques, and Challenges (pp. 20-34). IGI Global.
[25] Keshav Kumar et al "A Survey of The Design and Security Mechanisms of The Wireless Networks and Mobile Ad-Hoc Networks" 2020 IOP Conf. Ser.: Mater. Sci. Eng. 993 012063
[26] Srivastava, A., Verma, S. et al. "Analysis of Quality of Service in VANET", in Materials Science and Engineering Conference Series, 2020, vol. 993, no. 1, p. 012061. doi:10.1088/1757-899X/993/1/012061.
[27] Kavita et al. "A research paper on "A Fault Tolerant Approach For Load Balancing In Grid Environment" in IJERT Volume 1 Issue 9, November- 2012, ISSN: 2278-0181.
[28] Kumar, Parteek et al. "Detection of Wormhole Attack in VANET." National Journal of System and Information Technology 10.1 (2017): 71.
[29] Dogra, V. et al. "Understanding of Data Preprocessing for Dimensionality Reduction Using Feature Selection Techniques in Text Classification". In: Peng, SL., Hsieh, SY., Gopalakrishnan, S., Duraisamy, B. (eds) Intelligent Computing and Innovation on Data Science. Lecture Notes in Networks and Systems, vol 248. Springer, Singapore, 2021.
[30] Rani, P. et al. "Mitigation of black hole attacks

using firefly and artificial neural network", Neural Comput & Applic (2022). https://doi.org/10.1007/s00521-022-06946-7
[31] Jhanjhi, N. Z., Brohi, S. N., Malik, N. A., & Humayun, M. (2020, October). Proposing a hybrid rpl protocol for rank and wormhole attack mitigation using machine learning. In 2020 2nd International Conference on Computer and Information Sciences (ICCIS) (pp. 1-6). IEEE.
[32] K. Hussain, S. J. Hussain, N. Jhanjhi and M. Humayun, "SYN Flood Attack Detection based on Bayes Estimator (SFADBE) For MANET," 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 2019, pp. 1-4, doi: 10.1109/ICCISci.2019.8716416.
[33] Shah, I. A., Sial, Q., Jhanjhi, N. Z., & Gaur, L. (2023). Use Cases for Digital Twin. In Digital Twins and Healthcare: Trends, Techniques, and Challenges (pp. 102-118). IGI Global