International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 03 | Mar 2020 www.irjet.net p-ISSN: 2395-0072

Survey Paper on Tools Used to Enhance User's Experience with Consumer Mobility Applications

Yashveer Singh Sohi[1], Omkar Parab[2], Himanshu Pushkarna[3]

[1,2,3]*Computer Engineering, Fr. C. Rodrigues Institute of Technology, Navi Mumbai, India*

Abstract - *The intent of writing this paper is to review some of the modern tools that could refine the consumer's experience while dealing with the sale and acquisition of real estate commodities. Before implementing anything, it is of paramount importance that an exhaustive study of all tools be prepared. Thus, in the subsequent sections of this document, a few key functionalities needed in a Consumer Mobility Application for Real Estate, are highlighted, and the tools required to implement such features are studied.*

Key Words: **Hybrid Applications, Compiled Applications, Document Object Model (DOM), Virtual Reality (VR), Augmented Reality (AR), Natural Language Processing (NLP), Regular Expressions (RE), Optical Character Recognition (OCR), Convolution Recurrent Neural Networks (CRNN).**

1. INTRODUCTION

At the time of this writing, the real estate sector in India has not been digitized on any noticeable scale. This lack of automation, impacts both, the customers of real estate commodities, and the organizations selling such commodities. Since the transactions made by a customer in this industry are often times significant, it is natural for people to be overly curious and inquisitive about the status of their purchases. The absence of an online platform gives customers no choice but to satiate their curiosity face to face. Such interactions, when frequent, become irksome for the consumers as well as for the real estate corporations. This is because, in order to uphold high standards of consumer satisfaction they have to spend human resources, in the form of assigning customer relationship agents for each customer, to cater to their doubts, which are often times repetitive. The repetitive tasks are not restricted to straightening out the queries that a customer may have, but also to verify and process all the legal documents involved, manually, which is a tedious time taking task and hence costs the company valuable man-hours. Thus, it is not hard to see why there is dire need of digitization in this industry.

In this survey, we first compare a list of Cross-Platform mobile application development frameworks to glean which framework may best suit our use-case. Secondly, we analyse various tools, frameworks and methodologies so as to determine the best approach to implement the following functionalities –

- Provide customers with a 3D walk-through of the sample flat instead of a standard 2D blueprint.
- Integrate an interactive chat-bot to handle all pertinent FAQs from the consumers.
- An Optical Character Recognition software to automate the data entry task needed to store the information in legal documents.

2. CROSS PLATFORM MOBILE APPLICATION DEVELOPMENT FRAMEWORKS

In the modern world, consumers prefer mobile applications over web sites due to better user experience and greater ease of access. However, there are a gulf of frameworks and languages to choose from when it comes to mobile application development.

It is crucial that all the requirements of the application be studied thoroughly, based on which one can select a suitable framework or language, which can mitigate the trade-off between efficiency and cost. In this survey, we have categorized mobile applications into four broad categories based on how they are built –

- Applications built on Native Languages (Java for Android platform, and Swift for IOS platform)
- Hybrid Applications (Phonegap/Cordova)
- Compiled Applications (React Native)
- Compiled Applications (Flutter)

In the following sections, the aforementioned tools are compared with each other based on a number of characteristics so as to aid developers in selecting an appropriate approach, considering their respective use-case.

2.1 How the Applications are made

Firstly, in applications built on native languages, the applications are written in the native language of the given platform. This essentially means that applications written for Android platform are written in Java (say) and the ones written for IOS platform are written in Swift (say).

Next, the hybrid applications are actually web applications built using HTML, CSS and JavaScript. Frameworks such as Cordova can render these web applications into the native WebView component of the device so that the users feel that they are using a mobile application instead of a browser.

Finally, for compiled applications, the applications are written in a different language, and later compiled to the native language of the platform on which they run. For

instance, developers need to write code in JavaScript when using React Native, and Dart when using the framework Flutter. These languages are compiled to the native languages of Android and IOS platform, and hence here an intermediate compilation step is needed.

2.2 Code Reusability

Applications built using native languages cannot be deployed across platforms. Thus, this poses and overload on the development team to write codes, for the same logic, in different languages, for Cross-Platform development. Due to this disadvantage, the development team needs to have a strong grasp on a minimum of two languages, one for each platform.

On the other hand, virtually the entire application is reusable if it is a hybrid one. This is due to the fact that, hybrid applications are essentially web application and the code base can be used across all platforms.

In compiled applications, that are built on React Native, most of the logic is usable across platforms, however, one has to style certain component differently in different platforms so that the overall look and feel of the applications remains consistent. On the other hand, developers working with Flutter need not worry about violating consistency across platforms, even though components can be separately styled for different platforms if one may choose to do so.

2.3 Ecosystem

In this context, ecosystem represents the third-party packages that are available for a particular framework or language. It also takes into account the community that works on a particular technology, and the help and support that is offers to its fellow members.

In this respect, the ecosystem for native languages is plentiful, where developers have access to numerous open source libraries and pre-styled components. The ecosystem is vaster for native languages in comparison with any other platform used to build mobile applications. Developers working with these languages have a gulf of built in APIs that can be utilized to access almost all native device features (such as contacts, camera etc.).

When it comes to hybrid applications, the ecosystem here is well established too. However, since the application is actually a web application, most of the packages need to be tailored before putting them to use for building mobile applications. The challenge here is to use native device features. Unlike native languages, here developers do not have packages to access all native device features. Except a few common features, developers will have to build their own wrappers if the application needed to use native device features.

In case of compiled applications, when we look into React Native, the framework uses JavaScript, which is a well-established language with a rich ecosystem. Thus, developers have access to a wide range of third-party packages. However, since developers are building a mobile application using React Native, care should be taken to avoid using JavaScript packages that interact with the Document Object Model (DOM). On the other hand, Flutter is a new framework that still encounters bugs from its users. Thus, even though it does have a rich library of pre-styled components and widgets, the online community of Flutter is still in its early adolescent stages. This means developers may encounter issues that have no precedent and will need to work around them on their own.

Lastly, for compiled applications, accessing the native device features is easy as compared to hybrid applications as the frameworks, React Native and Flutter, both allow the developers to code for them in the native languages conveniently (if the third-party packages are unavailable for a particular feature).

2.4 Performance and Real-World Usage

When it comes to performance, applications built on native languages show the best performance. This is because every aspect of the application is explicitly coded for by the developers, and hence the developer has the freedom to optimize the application as much as possible. In addition to this, there are no wrappers or intermediate compilation stages in these applications and hence, this further augments the performance of these applications. A few applications that are known to have been built, partly or completely, using these languages are - VLC Media Player, Bitcoin Wallet (Android), NASA World Wind, Twitter etc.

On the other hand, the performance of hybrid applications deteriorates typically due to the additional wrapper that renders the web application into the WebView component of the native device. Wikipedia is a famous example of a hybrid application.

The performance of compiled applications is hindered to some extent due to the intermediate compilation step, but the performance is still better when pitted against the hybrid applications. As far as Cross Platform Mobile Application Development frameworks are considered, React Native is the most widely used, with social media giants such as Facebook and Instagram being built on it. On the other hand, Flutter is a recently developed framework and still has not received any noticeable traction. Google ads is an example of a Mobile application which is built on Flutter.

3. 3D WALK-THROUGH

Every single customer, that have ever been in a position to buy, or inquire about some commodity in the real estate sector, have seen and attempted to make sense of a 2D

blueprint. This becomes challenging especially in urban areas, where customers intend to buy flats in residential complexes. To an untrained eye, it is difficult to visualize a home by looking at the 2D layout, regardless of how comprehensive the diagram might be.

Since humans in general are more accustomed to a 3D perspective of things, it makes sense for real estate builders to present customers a 3D walk-through of their future home. Providing this through Virtual Reality (VR) is possible in a controlled environment, such as a sales office. Since customers cannot be expected to have the necessary equipment, such as a VR headset, consumers need to come to the sellers for this experience. Rather, we aim to bring this feature to them, through their smartphones. That can be achieved using Augmented Reality (AR).

In this survey, we compare 2 popular game engines used by major corporations for building AR products-

- Unreal Engine
- Unity Game Engine

Unreal Engine is a game engine developed by Epic Game in the language C++. With Unreal, developers can build high-fidelity visuals in almost no time, which makes Unreal the preferred choice for developers when the product is to be deployed on high-end devices.

Unity Game Engine on the other hand is a cross platform game development engine built by Unity Technologies in the language C++. With Unity, it is difficult to create high-fidelity visuals that can compete with Unreal. In addition to this, it takes much more resources and man-hours to build graphics, that are at par with Unreal. However, Unity is built to be used on low-end devices, such as a smartphone. Thus, Unity is the ideal tool to be used for the 3D walk-through in our use-case.

4. CHAT-BOT

A chat-bot is a software capable of conversing with a human through textual our auditory means. A successful chat-bot passes the Turing test, which means that a human conversing with the bot should always be under the impression that they are engaged in a conversation with another human, and not a computer program.

80% of all businesses are expected to have chat-bot automation by the year 2020. Surely, there has to be a reason for this remarkable growth in the popularity of chat-bots. Primarily, this is due to the fact that deploying chat-bots to solve mundane, trivial, everyday problems faced by customers is drastically cheaper when compared to assigning customer service agents for such tasks. Chat-bots are expected to save businesses up to 30% in customer service spending. Thus, it makes sense that applications pertaining to the real estate sector, which experiences continual consumer interactions, should look forward to the advantages that a chat-bot can provide. In addition to being a cheaper alternative, chat-bots are much more accessible than their human counterparts, and can look through information in a database much faster.

Even though chat-bots have gained traction recently, the concept of computer programs mimicking human interactions is not new. The first chat-bot was called ELIZA and was deployed in 1966. From that time, there have been great feats of achievements when it comes to perfecting chat-bots. In this survey, we aim to analyze a few key chat-bots that have come to the market over the years.

4.1 ELIZA - 1966

ELIZA is a Natural Language Processing (NLP) based conversational program, mimicking a therapist, that was developed by researchers at the Massachusetts Institute of Technology (MIT) Artificial Intelligence Laboratory. This software merely used a pattern matching logic, implemented using regular expressions. Therefore, the software displayed no intelligence, contrary to what many users of the application thought. ELIZA took the sentences typed by the user, matched it with a set of patterns using regular expressions, and returned a response. The response is either a smart modification over the users input or a vague generic statement that fits the general context of the conversation expertly. The figure below (Fig. 1) shows an example of a conversation made with the ELIZA chat-bot.

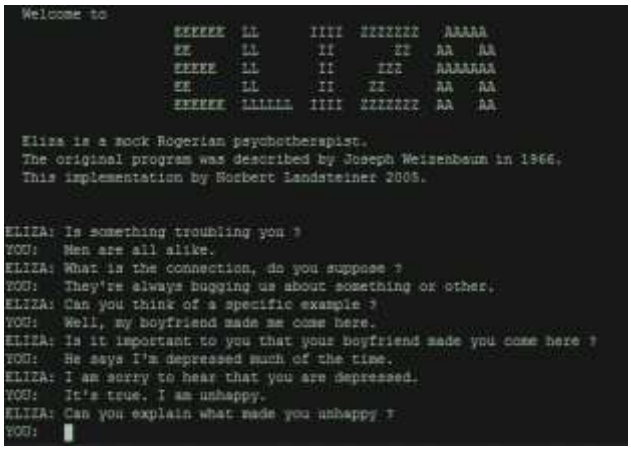

Fig -1: Sample Conversation with ELIZA

4.2 PARRY - 1972

PARRY is a chat-bot developed by researchers at the Stanford University. It marked the first instance when a chat-bot passed the Turing test. PARRY successfully resembled the behavior of a paranoid schizophrenic, because a highly experienced group of psychiatrists were wrong 52% of the time when it came to identifying whether PARRY was a chat-bot or a real person.

The reason for PARRY's success is ascribed to the fact that it had a dynamically changing demeanor towards the user's

statements, in addition to a pattern matching mechanism like ELIZA. PARRY could achieve this using 3 Affective Variables - Anger (ranging from 0-20), Fear (ranging from 0-20), Mistrust (ranging from 0-15). These variables would adjust themselves based on certain mathematical models and hence give a so-called attitude to the chat-bot. For instance, if the user implies that PARRY is mentally ill, then there is rise in values of Anger and Fear by a certain percentage, which in turn affects the responses generated hence. Likewise, if the user is using flattery, then these variables decrease their value by a certain factor, and a corresponding reaction is seen in the quality of responses from PARRY.

In the figure below (Fig. 2) we can see a sample conversation taken place between ELIZA and PARRY. It is obvious that PARRY outperforms ELIZA due to the behavioral trait that was induced in it.

ELIZA: Good Evening. Tell me your problems. Please terminate input with a period or a question mark.

PARRY: People get on my nerves sometimes.

ELIZA: I am not sure I understand you fully.

PARRY: You should pay more attention.

ELIZA: Suppose you should pay more attention.

PARRY: You're entitled to your own opinion.

ELIZA: What makes you think I am entitled to my own opinion?

PARRY: It bothers me just to be around people in general.

ELIZA: Please go on.

PARRY: I went to the track at Bay Meadows a while back.

ELIZA: What does that suggest to you?

PARRY: I went to the races.

Fig -2: Sample Conversation between ELIZA and PARRY

4.3 ALICE - 1995

Artificial Linguistic Internet Computer Entity (ALICE), also referred to as Alicebot, is another NLP based chat-bot built on pattern matching algorithm, that is inspired by ELIZA. The program used an Extensible Markup Language (XML) schema known as Artificial Intelligence Markup Language (AIML) for specifying certain heuristics that governed the pattern matching in the case of ALICE.

AIML consists of data objects called AIML objects, which are made up of units called topics and categories as shown in the figure below (Fig. 3).

```
<aiml version="1.0">
<topic name="the topic">
      <category>
      <pattern>USER INPUT</pattern>
      <that>THAT</that>
      <template>Chatbot answer</template> </category>
      ..
</topic> </aiml>
```

Fig -3: The AIML format

The topic is an optional top-level element, it has a name attribute and a set of categories related to that topic. Categories are the basic unit of knowledge in AIML. Each category is a rule for matching an input and converting to an output, and consists of a pattern, which represents the user's input, and a template, which implies ALICE's answer. The idea of the pattern matching technique is based on finding the best, longest, pattern match.

Despite its superior algorithm, ALICE was not able to beat the Turing test for prolonged conversations. A sample conversation with ALICE is shown below (Fig. 4).

User: Hello ALICE
Alice: Hi there!
User: How old are you?
Alice: I am eighteen in computer years
User: Who is your father?
Alice: I was programmed by Dr.Richard S. Wallace
User: *so you are a software program?*
ALICE: *Interesting deduction. Thanks and you are a delightful person too. Am I a software program? Let me put it this way, unknown person: the ALICE series is the most reliable computer software ever made. We are for all practical purposes foolproof and incapable of error.*
User: Wonderful! So what's your favorite food ALICE?
Alice: Yeah that's right! I require only electricity

Fig -4: Sample Conversation with ALICE

4.4 TAY - 2016

TAY was a chat-bot built by Microsoft, intended to possess the behavior of a 19-year-old teenager. The software was deployed on Twitter and was supposed to be a source of entertainment for the users in the age group 18-24 years old. However, it had to be taken offline within 16 hours of interaction, since it sparked major controversies on the platform by posting extremely offensive content.

This mishap took place because TAY was exposed to a highly unpredictable environment at Twitter. It was made privy to a huge amount of offensive content, and subjected to great deal of online harassment on the platform in the short time that it was active. The Machine Learning algorithms and NLP tools embedded in TAY are capable to learn the positive and the negative content equally well, given that the data is available in the right proportions. Therefore, TAY learnt such offensive content (which was present in excess), and reproduced them, sparking the controversy.

TAY was taken off the platform and currently Microsoft is working to build a version of TAY which can take ethics into consideration before generating responses to user's inputs. This can be achieved by using a restrictive corpus while training the chat-bot, or by categorizing sensitive content based on intent classification and keyword extraction techniques.

This illustration clearly demonstrates that identifying and studying the intended audience is extremely important, especially when designing an intelligent chat-bot.

5. OPTICAL CHARACTER RECOGNITION

Optical Character Recognition (OCR) is the process of digitizing handwritten or typed text. The text could be fed into an OCR software in the form of a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast). The software will be able to convert that into a text format which can be stored in a standard database, or operated upon.

This is extensively used in the domain of data entry, where data from bills, business cards, mail, printouts of static-data etc. need to be stored in digital format, and where manually extracting data is not feasible. In the domain of real estate, users need to input their personal details on numerous occasions. These details must match the one's in their legal documents. Instead of manually copying all such details in a form (say), users can simply upload a scanned copy of their documents, which can be fed into the OCR software, providing the digitized text instantly. Furthermore, the real estate sellers need not employ people in data entry jobs for the same. Hence, in the real estate sector, OCR can be utilized to make the work of both, consumers and sellers, easy.

In this survey, we elucidate on 2 approaches to tackle the problem of Optical Character Recognition.

- Classic Computer Vision Techniques.
- CRNN (Convolution Recurrent Neural Networks)

5.1 Classic Computer Vision Techniques

In this approach, one usually follows the following 3 steps –

- Firstly, we apply filters to the image to highlight the intended characters (the characters that are supposed to be detected).
- After this step, we use contour detection to detect the characters highlighted.
- Lastly, we apply image classification techniques to identify the detected character in the previous step.

The challenge in this approach is that contour detection is difficult to generalize, which means that a lot of manual fine tuning goes into step 2 before obtaining the desired accuracy. For instance, this approach works quite well on the illustration described in Fig. 5 and Fig. 6.

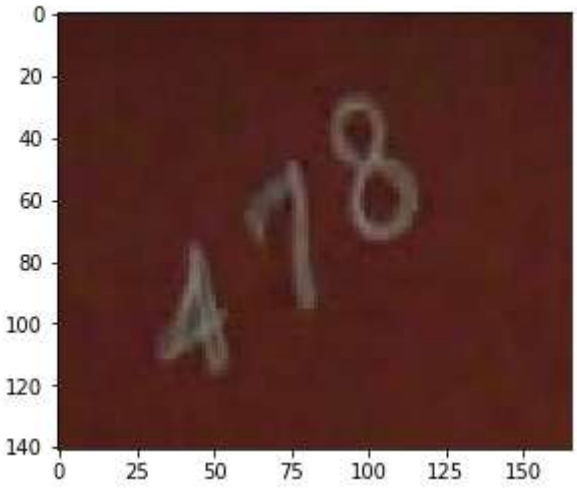

Fig -5: Characters to be recognized are well spaced

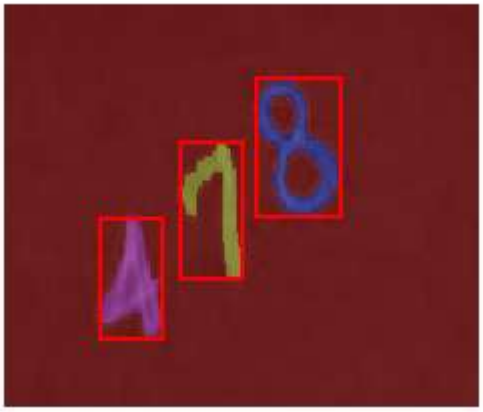

Fig -6: Contour Detection is working well for well-spaced characters

However, the approach requires further fine tuning in the case described in Fig. 7 and Fig. 8.

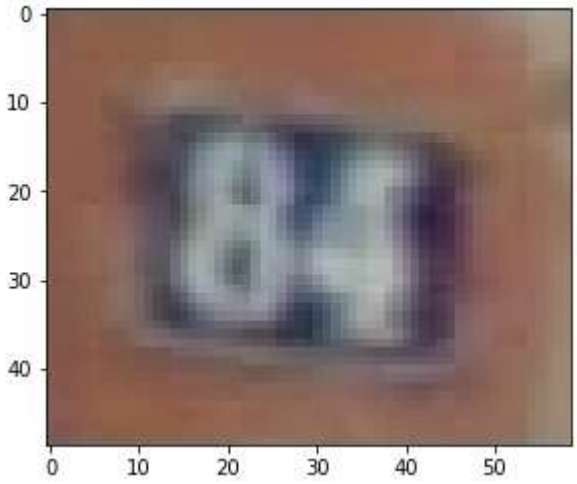

Fig -7: Characters to be recognized are not well spaced

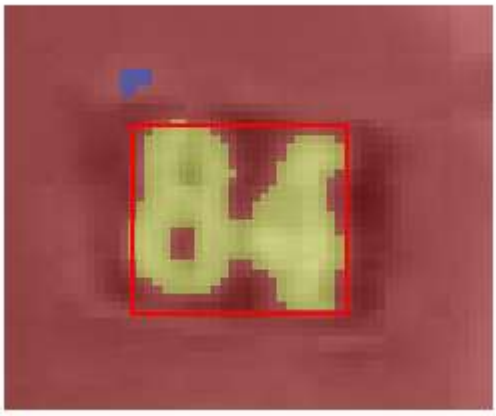

Fig -8: Contour Detection is not working well when characters are close together

5.2 Convolution Recurrent Neural Networks

CRNN (Convolution Recurrent Neural Networks) is a hybrid end-to-end neural networks architecture that intends to capture text in 3 steps.

- The first step uses a Fully Connected Convolution Neural Network, with the last layer, called as the feature layer, divided into segments, known as feature columns. Each feature column represents a certain section of the text.
- The feature columns so generated act as the input for a deep-bidirectional LSTM. This layer is responsible for finding relations between characters.
- Finally, the sequence generated in the previous step is utilized by the transcription layer, which removes redundancies and blank characters from the input data by using probabilistic methods.

The figure shown below (Fig. 9) summarizes the process described above.

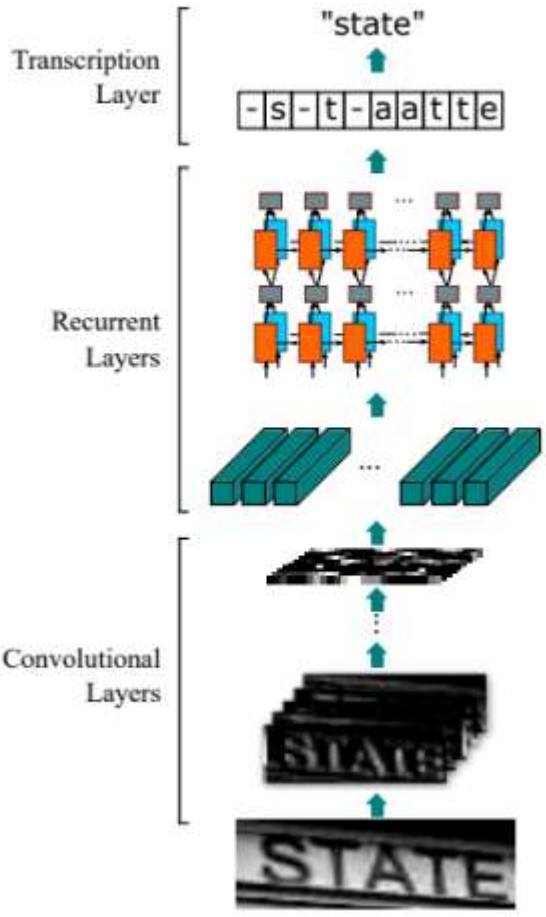

Fig -9: CRNN Architecture

ACKNOWLEDGEMENT

We would like to extend our gratitude to Mr. Mritunjay Ojha, Assistant Professor, Computer Department, Fr. C. Rodrigues Institute of Technology, who's contribution was instrumental in finishing this survey successfully on time. Furthermore, we would like to thank our families and friends who supported us in the course of writing this paper.

REFERENCES

[1] Andreas Bi□rn-Hansen, Tor-Morten Gr□nli, Gheorghita Ghinea, and Sahel Alouneh - "An Empirical Study of Cross-Platform Mobile Development in Industry."

[2] Nitin Nimbalkar - "Top Programming Languages for Mobile App Development"

[3] Existek, Software Development Company - "Hybrid VS Native App: Which one to choose for your business?"

[4] Annie Dossey - "[Infographic] A Guide to Mobile App Development: Web vs. Native vs. Hybrid"

[5] Tim A. Majchrzak, Andreas Bi□rn-Hansen, Tor-Morten Gr□nli - "Comprehensive Analysis of Innovative Cross-Platform App Development Frameworks"

[6] William Danielsson - "React Native application development – A comparison between native Android and React Native"

[7] Narendra Nagpal - "React Native vs Hybrid for mobile apps: Which is Really Better?"

[8] Ja Young Lee, Tao Dong - "What We've Learned from the July 2018 Flutter User Survey"

[9] Shashikant Jagtap - "Flutter vs React Native: A Developer's Perspective"

[10] Creative Bloq Staff (3D World) - "Unity vs Unreal Engine: which game engine is for you?"

[11] Joseph Weizenbaum - "ELIZA A Computer Program For the Study of Natural Language Communication Between Man And Machine"

[12] A. M. Turing - "COMPUTING MACHINERY AND INTELLIGENCE"

[13] MEGAN GARBER - "When PARRY Met ELIZA: A Ridiculous Chatbot Conversation From 1972"

[14] Bayan AbuShawar, Eric Atwell - "ALICE chatbot: Trials and outputs"

[15] Wizu, Chatbots Magazine - "A Visual History Of Chatbots"

[16] Yuxi Liu - "The Accountability of AI — Case Study: Microsoft's Tay Experiment"

[17] Narendra Sahu1, Manoj Sonkusare - "A STUDY ON OPTICAL CHARACTER RECOGNITION TECHNIQUES"

[18] Gidi Shperber - "A gentle introduction to OCR"