EJSIT
European Journal of Science, Innovation and Technology
ISSN: 2786-4936
www.ejsit-journal.com
Volume 3 | Number 2 | 2023

Function Point Analysis and Collaborative Study on UI Library, Open-Source DB, Cross Platform Server Environment, Object Detection Library and Cloud Computing Services

Sana Rizwan [1], Furqan Rasool Yahya [2], Moiz Rasheed [2], Hasnain Allah Rakha [2]
[1]Assistant Professor, Department of Computer Science, COMSATS University Islamabad, Lahore Campus, Pakistan
[2]Student of BSSE, Department of Computer Science, COMSATS University Islamabad, Lahore Campus, Pakistan

Abstract

A software application can only form by the integration of different platforms, environments, languages and collaborative libraries and APIs. This collaborative study presents an extensive analysis of the software to determine the complexity of any application. The metric can only calculate the exact collaboration. A client or user can identify the accurate amount of interactions with the functions and features to find the exact complexity metric. Function point analysis will be an effective way to measure software size. Internal, external, logical queries and responses will show metric results for the detailed amalgamation. To assess the challenge, use React as UI Library, MongoDB as open-source database, Node.js as cross platform open-source server environment, Coco SSD as object detection library and Firebase as cloud computing services.

To check the appropriate incorporation and ways, develop the complex commerce website and a mechanic and car washer app that incorporate advanced features such as augmented reality (AR), real-time tracking, in-app chat functionality, and car customization options. The e-commerce website is built using React for the front-end, MongoDB for the back end, and Node.js for the back-end development. The website enables the sale of old vehicle parts and allows users to customize their cars using AR technology with the Coco SSD model. Firebase will handle image uploading. The mechanic and car washer app is built using React Native, MongoDB for the back-end, and Node.js for the back-end development. The app enables users to call a mechanic or car washer and some features such as in-app chat, real-time tracking, and payment options. One of the unique features of the project is the car customization area, which incorporates Machine Learning and Color Segmentation. To enable color segmentation, an HSV model of the user's car is created, and segmentation is performed using OpenCV, Python library. This approach allows the dynamic changing of hue and saturation of the image to colorize the segmented area. Users can customize their cars by selecting various options such as rims, color, and other modifications. Additionally, the website includes a learn-to-drive module where users can learn to drive through various videos. To enable real-time chat functionality across both the website and app, socket programming has been implemented, allowing users to communicate with each other seamlessly. A function point analysis was conducted to evaluate the size and complexity of the software development project, providing valuable insights into the effort, time, and cost required for development.

The study highlights the challenges and opportunities associated with building an e-commerce website and an app with advanced features such as AR, real-time tracking, socket programming, and car customization options. The success of the website and app depends on the effective integration of different technologies and the ability to provide seamless user experience. The study's findings will help developers and stakeholders make informed decisions, optimize resource utilization, and enhance the website and app's functionality to meet the evolving needs of the customers.

Keywords: Augmented Reality; React; Socket Programming; Node.js; Express; Bootstrap; Firebase; Socket Programming’HSV model; Funtion point analysis; Open source; Document object model

Introduction

The automotive industry is undergoing a significant transformation with the adoption of advanced technologies such as augmented reality, real-time tracking, and machine learning. These technologies are changing the way people buy, sell, and customize their vehicles, leading to the development of innovative solutions that cater to evolving customer needs. In this collaborative study, we present an in-depth analysis of an e-commerce website and a mechanic and car washer app that incorporate advanced features such as AR, real-time tracking, in-app chat functionality, and car customization options.

The e-commerce website is a platform for selling old vehicle parts and enables users to customize their cars using AR technology with the Coco SSD model. The website is built using React for the front-end, MongoDB for the backend, and Node.js for the backend development. The website has three roles: admin, seller, and buyer, and includes standard e-commerce features such as payment options, shop-ping cart, search, and filter options, and more. Firebase is used for image uploading, while the car customization module incorporates machine learning and color segmentation techniques.

The mechanic and car washer app is a mobile application that allows users to call a mechanic or car washer anywhere, similar to Uber ser-vices. The app is built using React Native, MongoDB for the backend, and Node.js for the back-end development. The app has two roles: service provider and customer and includes features such as in-app chat, real-time tracking, and payment options.

Through this study, we aim to showcase the challenges and opportunities associated with building an e-commerce website and a mobile app with advanced features such as AR, real-time tracking, socket programming, and car customization options. The findings of this study will help developers and stakeholders understand the effort, time, and cost required for development and enhance the functionality of the website and app to meet the evolving needs of customers in the automotive industry.

People go from city to city to purchase these auto parts of vehicles, which can be imported or NCP vehicle parts. Actually, online marketplaces for such equipment is not easily available where people can buy the parts they need. Secondly, the availability of restored parts or secondhand parts with quality assurance is not easily present; as a result, we will specify a warehouse in addition to all other quality product standards. The system will consist of multiple modules:

E-commerce platforms are open to everyone who owns a company and wants to move their operations online. It is to make sure that it will have a practical implementation and the project will launch in the market before any evaluation, all ad-vertexing, or onsite visits, etc.

Next module is about Augmented Reality. If a customer wishes to purchase exterior items for their vehicle, such as rims, body kits, tires, or bumpers, special functions in the store allow them to do so. He or she can submit a picture of their vehicle, and the system will adjust it to fit the product using image processing. Customers who find that a product looks good on their vehicle are more likely to purchase that product.

The system has a wide range of services, including mobile car wash. Customers will be able to submit requests, and washers will be notified about requests on FCFS services. After being accepted, the chat features will be turned on, and users will be able to talk to each other. This will result in a significant increase in employment opportunities in the area.

There will also be some driving tutorials included, albeit the user will be instructed by the system to use animations. This tool will be useful for novice drivers who are just starting out and learning the rules of the road.

Building of these modules with its complete functionality using React, Mongo DB, Node.js, Coco SSD, Firebase having total number of lines of code to calculate software complexity by functional units i.e., Internal Logical Files (ILF), External Interface Files (EIF), External Inputs

(EI), External Outputs (EO) and External Enquiries (EQ). These units will identify the accurate picture of client interface interaction scenario as shown in Figure 1.

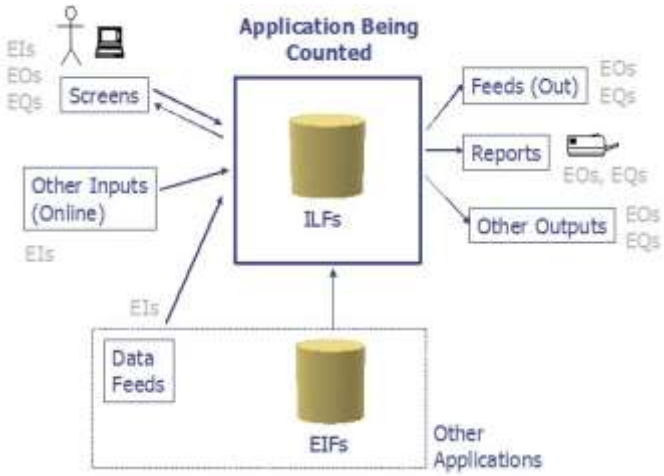

Figure 1: Function Point Units

Methodology

There is a significant degree of uncertainty regarding the requirements and functionalities of the software. This is because the project involves multiple modules and technologies such as React, Node.js, MongoDB, Firebase, OpenCV, and augmented reality, each of which presents its own set of challenges.

Furthermore, there is a risk involved in the software, particularly in the implementation of the augmented reality and machine learning modules. These technologies require a high degree of expertise and experience, and the risk of failure is relatively high.

Finally, the software requires a high degree of collaboration and communication between different teams and stakeholders, including developers, designers, project managers, and customers as shown in Figure 2.

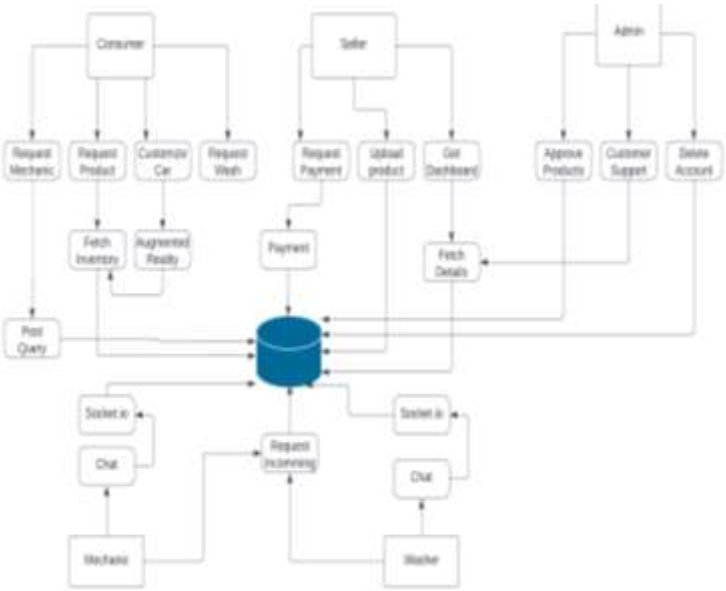

Figure 2: System Architecture

Augmented Reality

By combining elements of both virtual and physical reality, augmented reality creates a new immersive experience. Recent rapid progress in developing augmented reality has sparked the public's curiosity in cutting-edge technology (React (software), 2023).

Main feature is superimposes digitally created images onto real-world settings. Multimedia, 3D modelling, real-time tracking and registration, intelligent interaction, sensing, and more are just some of the technologies used. Investments in both technology and capital have fueled augmented realities meteoric. Whereas foreign augmented reality software firms include Metaio, Vuforia, Wikitude as shown in Figure 3. Rapid progress in the field of augmented reality technology, propelled mostly by breakthroughs in computer vision and artificial intelligence. The nature of human-computer interaction, the capabilities of display devices, and the accuracy of tracking registration have all advanced significantly. However, it's obvious that there are still many problems that require fixing in the realm of augmented reality technology.

Figure 3: Famous Pokemon Go game was developed using Augmented Reality

React/React Native

React is a JavaScript library used for building web applications and user interfaces, while React Native is a mobile application development framework used for building native mobile applications.

React works on the concept of a virtual DOM (Document Object Model). This allows React to perform efficient updates by comparing the virtual DOM to the actual DOM and only rendering the differences. React also uses components, which are reusable pieces of code that can be composed together to create complex user interfaces as shown in Figure 4.

React Native allows developers to build mobile applications using React-style components and APIs that are specific to mobile platforms. This allows developers to build native mobile applications for iOS and Android using the same JavaScript codebase. React Native also uses a virtual DOM to optimize performance and provide a smooth user experience.

React enables developers to create modular and reusable components that can be easily composed to build complex user interfaces. This makes the code more maintainable and easier to scale. It follows a unidirectional data flow, where data flows only in one direction, from parent to child components. This makes it easier to debug and maintain the codebase. React enables developers to write code in a declarative manner, which means that they can specify what they want to happen, rather than how they want it to happen. This makes the code easier to read and understand. It has a large and vibrant ecosystem of libraries, tools, and resources that will be used to enhance development productivity and solve common problems. Furthermore, it will be used to build web applications as well as native mobile applications for iOS and Android platforms using React Native.

In summary, while React is used for building web applications, React Native is used for building native mobile applications. Both frameworks use a virtual DOM and components to build reusable, efficient user interfaces.

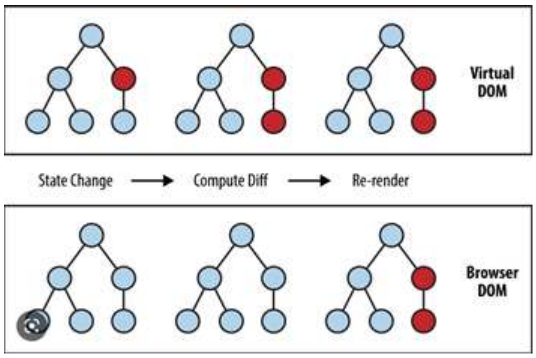

Figure 4: React DOM

Mongo DB

MongoDB is a NoSQL database that uses a document-oriented data model. This means that data is stored as JSON-like documents, rather than in tables with rows and columns like a traditional relational database. MongoDB's flexible schema design allows for easy scaling, indexing, and querying of enormous amounts of data. It is also known for its high performance, automatic sharing, and replication features, making it a popular choice for web applications, mobile apps, and other large-scale data-driven projects. MongoDB is designed to scale horizontally across multiple servers, which makes it easy to handle large amounts of data and high traffic volumes. It provides built-in replication and automatic failover, which ensures that the data remains available in the event of a hardware failure or network outage. It provides high performance and low latency for read and write operations, thanks to its ability to store data in memory and its efficient indexing capabilities. MongoDB's flexible schema and easy-to-use API make it easy for developers to work with and iterate on the data model, resulting in faster development cycles. It is a document-oriented database, which means that it maps directly to the data structures used in many modern applications. This makes it easy to work with for developers who are already familiar with JSON and other document formats. It supports ad-hoc queries, which allows developers to retrieve data quickly and easily without having to define a specific schema or data model in advance.

Firebase

Firebase is a mobile and web application development platform, owned by Google, which provides a range of services and tools for building and deploying applications (Firebase, 2023). Firebase provides a real-time database that enables developers to store and synchronize data in real-time across multiple clients, including web, mobile, and desktop applications. It provides a built-in authentication system that makes it easy to manage user authentication and authorization for your application. It provides a hosting service that allows developers to deploy and host their applications, including web apps, static content, and dynamic content quickly and easily. It endows a comprehensive analytics solution that enables developers to track user behavior, engagement, and retention for their application. It presents a serverless computing solution, known as Firebase Cloud Functions, which allows developers to write and deploy server-side code without having to manage servers or infrastructure. Firebase integrates seamlessly with other Google services, such as Google Cloud Storage, Google Cloud Functions, and Google Cloud Messaging, making it easy to build and deploy applications on the Google Cloud Platform. With addition, it supports for a wide range of platforms, including

web, iOS, Android, and Unity, making it easy to build and deploy applications across multiple platforms.

Socket Programming

Socket programming is a way of communicating between two computers over a network using sockets, which are endpoints of a two-way communication link between a client and a server (Sankar, 2022). In socket programming, a client program sends a request to a server program through a socket. The server program receives the request, processes it, and sends a response back to the client program. The client program receives the response and may send additional requests to the server. Sockets are typically implemented using the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) network protocols. TCP is a connection-oriented protocol that provides dependable, ordered delivery of data between applications, while UDP is a connectionless protocol that provides unreliable, unordered delivery of data as shown in Figure 5.

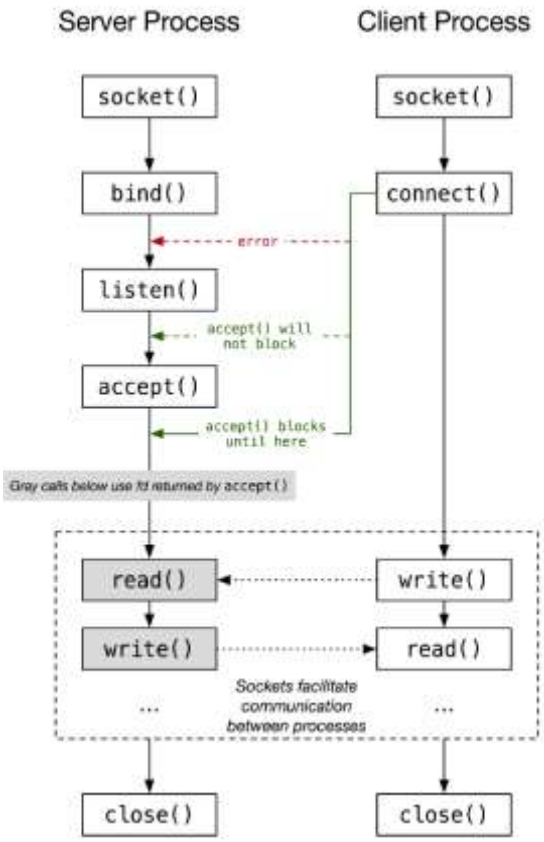

Figure 5: Working of Socket Programming

Bootstrap

Bootstrap is a popular open-source front-end development framework that allows developers to create responsive and mobile-first web pages quickly and easily (Bootstrap (front-end framework), 2023). Bootstrap provides a set of pre-designed UI components, such as buttons, forms, and navigation menus, which can be easily customized and integrated into a website as shown in Figure 6. Bootstrap also includes a responsive grid system that allows developers to create flexible and fluid layouts that adapt to different screen sizes and devices. It is designed to be modular, so developers can choose to use only the components they need, rather than including the entire framework.

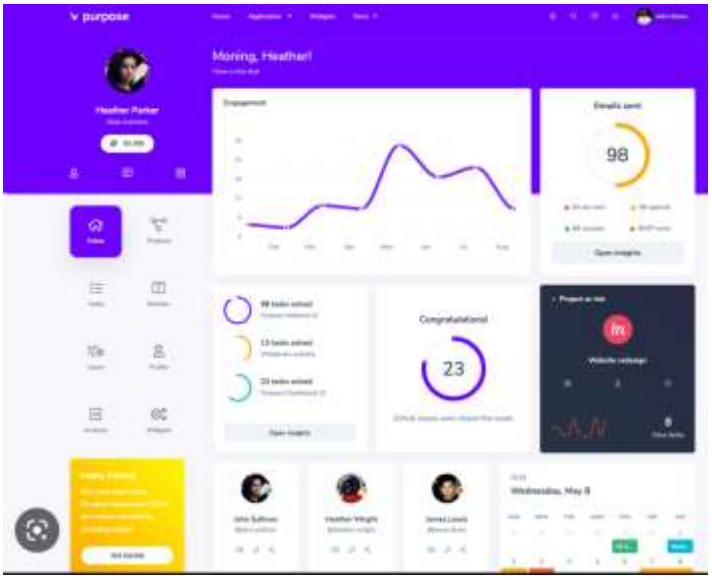

Figure 6: Interface using Bootstrap

TensorFlow

TensorFlow allows developers to build and train various machine learning models, including deep neural networks, to solve a wide range of problems such as image and speech recognition, natural language processing, and many others. TensorFlow (2023) provides a variety of tools and APIs for building and deploying machine learning models, including high-level APIs such as Keras and TensorFlow Estimators, as well as lower-level APIs for more advanced users. It can perform distributed computing across multiple CPUs and GPUs, which makes it suitable for large-scale machine learning tasks. TensorFlow also provides support for mobile and embedded devices through Tensor-Flow Lite, which allows models to deploy on devices with limited computational resources. It is a powerful and flexible framework that has gained widespread adoption in the machine learning community due to its ease of use, scalability, and versatility.

OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library[1]. It is written in C++ and has bindings for Python and other languages. OpenCV has a wide range of functions and algorithms for image and video processing, such as image filtering and transformation, object detection and tracking, camera calibration, and machine learning. It is widely used in various fields such as robotics, surveillance, and computer vision research. OpenCV also provides a graphical user interface (GUI) for visualizing and evaluating its algorithms. It is a powerful and popular tool for computer vision and image processing tasks in Python.

Function Point Analysis

The process of function point estimation is to gather available documentation, determine the counting scope, boundaries, functional user requirements, Identify and classify the base functional components, calculate the functional size, document the function points, and finally report the result. FPA is classified into two major functional components i.e., measure the data functions and measure the transactional functions. Internal grouping of data is called Internal Logical Files (ILF), whereas external groupings of data is called External Interface Files (EIF).

[1] https://opencv.org/

There are three measures of transactional functions: External Inputs (EI), External Outputs (EO) and External Inquires (EQ).

Implementation

What is API and Reason to use API?

An API, or Application Programming Interface, is a set of protocols, routines, and tools that enables software applications to communicate with each other. APIs also make it easier to develop applications, as developers can simply use pre-built APIs rather than having to write all the code themselves. API provides a standardized way for applications to communicate with each other. This means that different parts of the application can be developed independently and then connected via the API. It allows for the separation of concerns between the front-end and back-end of the application.

Combination of MongoDB with Express and Node

The combination of Node.js, Express, and MongoDB provides a robust and scalable platform for building APIs and server-side applications, allowing for efficient handling of data requests and seamless integration with a variety of client-side applications.

Implementation of API using MongoDB, Express and Node

API plays a crucial role in facilitating communication between users and the database. Built with Mongo technology, it serves as the vital bridge that enables seamless data transmission and retrieval. Since for e-commerce an API is must so developing one was necessary. The API used here will begin the implementation with the definition of a Model, which serves as the schema in a relational database. By leveraging the power of this technology, we can ensure a smooth and efficient user experience that enables users to seamlessly interact with the platform.

Data variables/attributes such as username, full name, phone and etc., are selected as test data. After defining them we create a new collection which is a table in relational database having title “Paiyauser” as shown in Figure 8 of Model.

Figure 8: Model of API

After defining Model, define basic routes with different kinds of request that are going to be passed to the database in easy words. These are database manipulation techniques DML. There are four types of post which is to insert new data into database, update database, delete the unwanted data from database and last one is fetch data from database as shown in Figure 9. Node.js will help to build these routes. Various routes are given below in catch and block format.

Figure 9: Routes in API

Muller is used to upload files in databases such as images as shown in Figure 10.

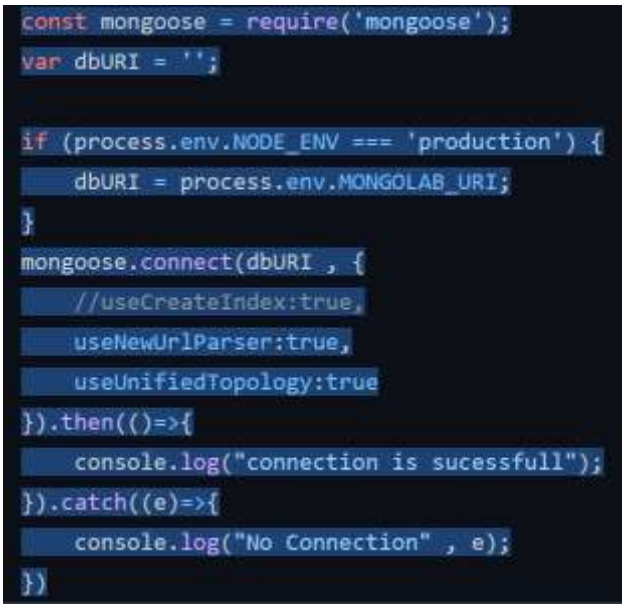

Figure 10: Routes in API

After creating Model and route, need to make connection with MongoDB Atlas or Compass to work with URL, which was generated by the MongoDB itself. However, URL present in .env file to maintain privacy of database. The variable dB URI stores the URL of MongoDB database, which was fetched from the env file as mentioned in Figure 11.

Figure 11: Connect MongoDB and bind URI

Augmented Module and Car Customization

What is Augmented Reality and use of AR technology?

Augmented reality (AR) involves the use of computer-generated sensory input, such as sound, video, graphics or GPS data, to enhance and augment a user's real-world experience. AR has the potential to revolutionize many industries and provide new and exciting opportunities for innovation and growth.

Why use CocoSSD model?

Coco SSD (Common Objects in Context Single Shot Detector) is a pre-trained machine learning model that can be used for object detection in images and videos. It can detect and classify various objects in an image and return their location and classification data. When integrated with augmented reality technology, Coco SSD can help detect specific objects in real-time and overlay relevant virtual content on top of them. Coco SSD uses a deep neural network to perform object detection and classification, and it is based on the TensorFlow framework. By leveraging the capabilities of Coco SSD and TensorFlow, developers can create powerful augmented reality applications that can detect and interact with specific objects in real-time.

CocoSSD model comparison with other models

CocoSSD is a state-of-the-art object detection model that can detect and localize objects in an image. It is a single-shot detector model that is fast and efficient, making it suitable for real-time applications. It is trained on the Common Objects in Context (COCO) dataset, which is a large-scale object detection, segmentation, and captioning dataset. In comparison to other object detection models, such as YOLO (You Only Look Once) and Faster R-CNN (Faster Region-based Convolutional Neural Network), CocoSSD is faster and lightweight. It can run on lower-end devices and still deliver satisfactory results. YOLO is also a single-shot detector,

but it can be slower than CocoSSD, especially on large images. On the other hand, Faster R-CNN is more accurate than CocoSSD, but it is also slower and requires more computational resources.

Overall, the choice of object detection model depends on the specific application requirements, such as accuracy, speed, and device capabilities. CocoSSD is a desirable choice for real-time applications that require fast and accurate object detection on lower-end devices.

OpenCV Library Comparison with Other Tools

OpenCV library is a widely used computer vision library, and it has several features that make it a popular choice among developers for computer vision applications. Here are some comparisons with other popular computer vision libraries:

1. OpenCV vs. TensorFlow: TensorFlow is a popular deep learning library that is primarily used for building and training neural networks. While OpenCV also supports deep learning models, its focus is on image and video processing, and it is more geared towards traditional computer vision tasks such as object detection and tracking.
2. OpenCV vs. PyTorch: PyTorch is another popular deep learning library that is known for its ease of use and flexibility. Like TensorFlow, PyTorch is more geared towards deep learning tasks, whereas OpenCV is more focused on traditional computer vision tasks.
3. OpenCV vs. scikit-image: scikit-image is a Python library that provides tools for image processing and computer vision. While scikit-image has some useful features, such as support for segmentation and feature extraction, OpenCV is more comprehensive and provides a wider range of tools for image and video processing.

Overall, OpenCV is a powerful and versatile library that offers a range of tools for image and video processing, making it a popular choice for developers working on computer vision applications.

Implementation of AR with the help of CocoSSD model

Utilization of the CocoSSD library to detect whether the user uploaded a car image or not, allowing them to further customize their vehicle by adding rims and other augmented technology (What Object Categories, 2018). The CocoSSD model provides accurate and efficient object detection in real-time, making it an ideal solution for basic needs. With its ability to detect objects within an image, the system was able to incorporate a user-friendly interface for customizing their car using augmented technology. The main advantage of the model is to create a canvas which is a board for painting. Below Figure 12 shows the implementation of the canvas.

```
const handleMouseMove = (event) => {
  if (isDragging && draggedObject) {
    //const img = imgUrl;
    const canvas = canvasRef.current;
    const context = canvas.getContext("2d");
    const mouseX = event.nativeEvent.offsetX;
    const mouseY = event.nativeEvent.offsetY;

    context.clearRect(0, 0, canvas.width, canvas.height);
    context.beginPath();
    const img = document.getElementById("input-image");
    context.drawImage(img, mouseX, mouseY , 90 , 90);
    setDraggedObject({ x: mouseX, y: mouseY });
  }
};
```

Figure 12: Canvas

Canvas use in this case study is 2D, whereas, in future this study will enhance to 3D model.

- const handleMouseDown = (event) => {: This is a function declaration that takes an event as an argument. The event in this case is the mouse down event on the canvas element.
- const canvas = canvasRef.current;: This line gets a reference to the canvas element using the canvasRef object.
- const context = canvas.getContext("2d");: This line gets the 2D rendering context for the canvas element. This context is used to draw on the canvas.
- const mouseX = event.nativeEvent.offsetX;: This line gets the X position of the mouse on the canvas element as shown in Figure 14.
- const mouseY = event.nativeEvent.offsetY;: This line gets the Y position of the mouse on the canvas element.
- const img = document.getElementById("input-image");: This line gets a reference to an HTML image element with an id of "input-image".
- context.fillStyle = \url('${img}')`;: This line sets the fill style of the context to a URL of the img` variable.
- context.beginPath();: This line starts a new path on the context.
- context.drawImage(img, mouseX, mouseY , 100 , 100);: This line draws the image on the canvas at the position of the mouse with a width and height of 100 pixels.
- setIsDragging(true);: This line sets the isDragging state to true.
- setDraggedObject({ x: mouseX, y: mouseY });: This line sets the draggedObject state to an object with the X and Y position of the mouse.

An image was loaded to canvas which has a background and mouse drag and drop was allowed to fit rims on selected vehicle as shown in Figure 13. Even though no one can simply start modifying the images when they have inserted one, some procedure must follow. First, check the image with the help of Coco model to determine if the image is car or not? The model checks the image against the other images in its database when the model determines what image it assigns a class to that image and then will perform further action against it, whether wanted the object in picture of the following class or not . Below is the implementation of it.

```
const detectRims = async () => {
  setLoading(true);

  // Load the COCO-SSD model
  const model = await cocoSsd.load();

  // Load the input image
  const img = document.getElementById("input-imagee");
  console.log("loaded img is : " , img);
  // Detect objects in the input image
  const predictions = await model.detect(img);

  // Filter the predictions to only include rims
  const rimsPredictions = predictions.filter(
    (prediction) =>  prediction.class === "car"
  );
console.log("rimPrediction is : " , rimsPredictions);
  if (rimsPredictions.length > 0) {
    setRimsDetected(true);
```

Figure 13: Model Detection -- Rims Prediction

```
  // Get the coordinates of the rims
  const rimsCoords = rimsPredictions[0].bbox;
    console.log(rimsCoords);
  // Create a canvas element and draw the input image on it
  const canvas = document.createElement("canvas");
  canvas.width = img.width;
  canvas.height = img.height;
  const ctx = canvas.getContext("2d");
  ctx.drawImage(img, 0, 0);


} else {
  setRimsDetected(false);
  alert("No Car Found in Image !");
  setUserCar("");
}

setLoading(false);
```

Figure 14: Create Element

By using await, Coco model can check image and move to other line of code that was pre-defined set. Only the condition works only if it belongs to the class car and set rimsPrediction to car object. If the condition is true, then length will be of 3 because word "car" is stored in it. This will confirm the relative image that belongs to the class car. If the image does not belong to the class car then error is simply generated and user cannot customize the car as seen in If/Else loop in Figure 14. Simply drag and drop the selected rims from inventory and apply it on rims. This will show how to manage mouse function. Their functionality is dependent on the x and y axis as object moves with the pointer below is the implementation as shown in Figure 15 & 16.

```
const handleMouseDown = (event) => {
  const canvas = canvasRef.current;
  const context = canvas.getContext("2d");
  const mouseX = event.nativeEvent.offsetX;
  const mouseY = event.nativeEvent.offsetY;
  context.fillStyle = `url('${img}')`;
  context.beginPath();
  const img = document.getElementById("input-image");
    context.drawImage(img, mouseX, mouseY , 100 , 100);

  setIsDragging(true);
  setDraggedObject({ x: mouseX, y: mouseY });
};
```

Figure 15: Implementation of different mouse functions

```
const handleMouseMove = (event) => {
  if (isDragging && draggedObject) {
    //const img = imgUrl;
    const canvas = canvasRef.current;
    const context = canvas.getContext("2d");
    const mouseX = event.nativeEvent.offsetX;
    const mouseY = event.nativeEvent.offsetY;

    context.clearRect(0, 0, canvas.width, canvas.height);
    context.beginPath();
    const img = document.getElementById("input-image");
    context.drawImage(img, mouseX, mouseY , 90 , 90);
    setDraggedObject({ x: mouseX, y: mouseY });
  }
};

const handleMouseUp = () => {
  setIsDragging(false);
  setDraggedObject(null);
};
```

Figure 16: Drag and Set Image – Handle Mouse Move

System's interface of the module detecting image from inventory by matching the uploaded image. Below is the front end of the module. The red car image is uploaded which is set inside a canvas of rectangle type in Figure 17.

Figure 17: Coco Model Image detection in process

The CocoSSD model is trying to detect the object in the Figure 18 given below, provided by the user.

Figure 18: Successful Image detection

The above coco model successfully detects the images and allows drag and drop of the items in the inventory section as shown in Figure 19. A user has successfully placed the rim on their vehicle.

Figure 19: Rim Placed on vehicle

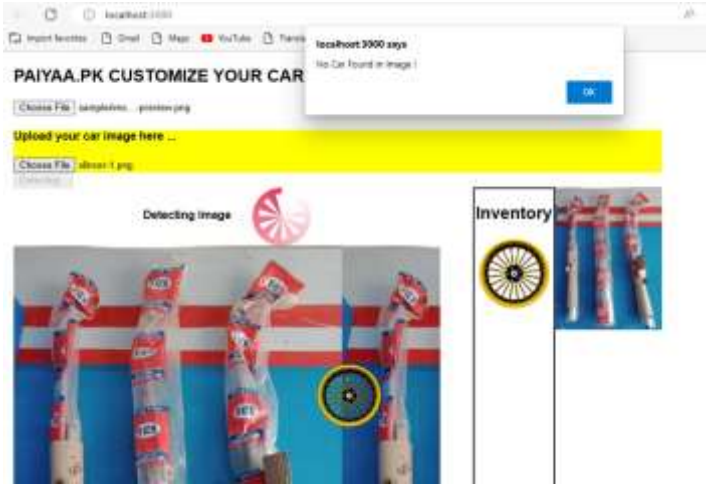

Figure 20: Error detection

In the above Figure 20, an image of the motorcycle parts has been uploaded that is not directly concerned to the required image pattern and do not belong to the class car. Hence error is generated, and we cannot place our rim.

Coco-model class documentation can help to determine the related objects as shown in Figure 21.

Search:

ID	OBJECT (PAPER)	OBJECT (2014 REL.)	OBJECT (2017 REL.)	SUPER CATEGORY
1	person	person	person	person
2	bicycle	bicycle	bicycle	vehicle
3	car	car	car	vehicle
4	motorcycle	motorcycle	motorcycle	vehicle
5	airplane	airplane	airplane	vehicle
6	bus	bus	bus	vehicle
7	train	train	train	vehicle
8	truck	truck	truck	vehicle
9	boat	boat	boat	vehicle
10	traffic light	traffic light	traffic light	outdoor
11	fire hydrant	fire hydrant	fire hydrant	outdoor
12	street sign	-	-	outdoor

Figure 21: Coco Model classes Identification

Implementation of car customization with the help of OpenCV

In the context of car color customization, OpenCV can be used to isolate and modify the color of a car body in real-time video. This is achieved by using various image processing techniques, such as color filtering, edge detection, and image segmentation as given in Figure 22. By applying these techniques, we can create a mask that isolates the car body, and then modify the color of the car body in the masked region.

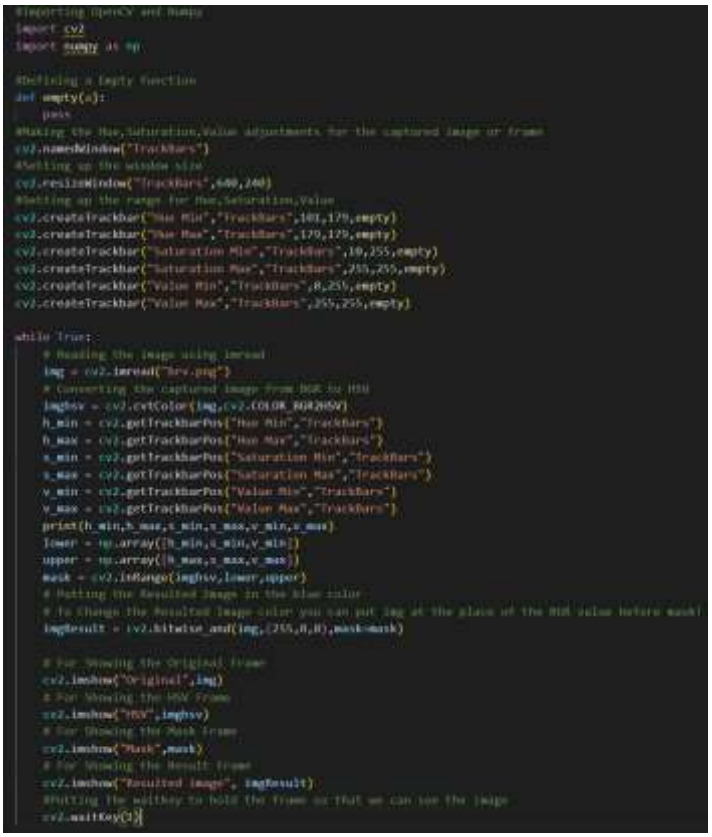

Figure 22: Code for car customization using OpenCV

- The necessary libraries are imported - cv2 for OpenCV and numpy for numerical operations.

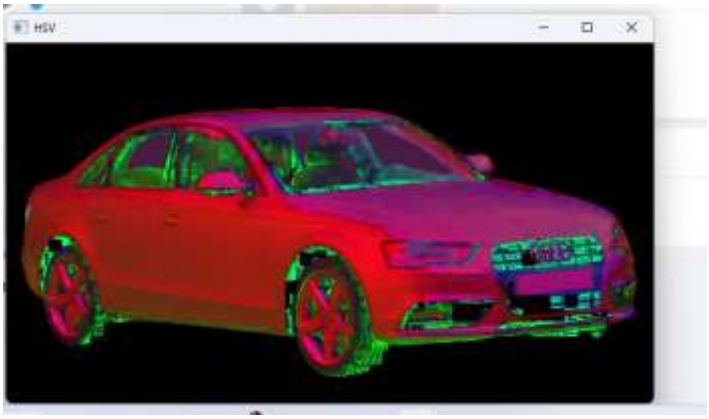

Figure 23: Car OpenCV applied operations

- An empty function is defined for future use.

- A window named "TrackBars" is created, which will allow the user to adjust the HSV values to detect a specific color. The size of the window is set to 640x240.
- Six trackbars are created using the cv2.createTrackbar() method for Hue Min, Hue Max, Saturation Min, Saturation Max, Value Min, and Value Max. These trackbars are used to adjust the HSV values for color detection. The values for each trackbar are initially set to default values as shown in Figure 23.
- An infinite loop is started, which reads an image from the file "brv.png" using the cv2.imread() method.
- The image is then converted from the BGR color space to the HSV color space using cv2.cvtColor() method.
- The current position of each trackbar is obtained using cv2.getTrackbarPos() method and used to create an array of lower and upper bounds for HSV values.
- Using these bounds, a mask is created using the cv2.inRange() method.
- The bitwise AND operation is performed on the original image and the mask, which results in a black and white image with only the detected color visible.
- Four images are displayed using the cv2.imshow() method - the original image, the HSV image, the mask image, and the resulted image.
- The waitKey() method is used to hold the frame for a specified time so that the image can be seen.
- The code runs until the user closes the window.

Figure 24: A normal image of vehicle in silver color

Normal image of car is uploaded as shown in Figure 24, which is to be colored in blue. Hsv image of the car which determines which parts of the car are to be repainted or not. The green color shows the parts that will be painted in Figure 23. And the final painted color car is shown in Figure 25, still there needs to be a fixing as it has failed to detect rim, mirror and other parts of vehicle as separate objects.

Figure 25: A final image generated

Implementation of E-commerce web

React comparison with other frameworks

React is a popular JavaScript library used for building user interfaces. Here is a comparison of React with other popular frontend frameworks:

- Angular: Angular is a full-featured framework that provides a lot of out-of-the-box features for building complex web applications. React, on the other hand, is a library that provides a lightweight solution for building user interfaces. Angular follows a more opinionated approach to development, whereas React gives developers more flexibility to choose their own approach.
- Vue.js: Vue.js is a progressive JavaScript framework that is similar to React in many ways. Vue.js has a smaller learning curve than React and provides a more structured approach to development. React, on the other hand, has a larger ecosystem and more community support, making it a more popular choice for building large-scale applications.
- Ember.js: Ember.js is a full-featured framework that provides a lot of conventions and out-of-the-box features for building complex applications. It is more opinionated than React and requires developers to follow strict conventions. React, on the other hand, is more flexible and allows developers to choose their own approach to building applications.
- jQuery: jQuery is a popular JavaScript library that is used to manipulate the DOM and add interactivity to web pages. It is not a full-featured framework like React, and is typically used for smaller projects or for adding interactivity to existing web pages. React, on the other hand, is a more comprehensive library that provides a complete solution for building user interfaces.

Implementation of front-end using React

Admin panel was developed first for the ecommerce website the functionalities of admin panel is to accept/reject mechanic and washer forms who will apply for the job other is to approve the products on website or block users account which do not obey the fair policy.

The implementation of accepting/rejecting user forms using axios to send request to API which we developed is shown in Figure 26.

Figure 26: Admin Panel

A request is made to API via axios to fetch all the forms from database so admin can accept them or reject them. This code is an asynchronous function fetchData that uses the Axios library to fetch data from an API endpoint hosted on Heroku. The API endpoint returns a list of users in JSON format.

When fetchData function is requested, it makes a GET request to the API endpoint using the Axios get method with the URL of the API endpoint as its parameter. If the request is successful, the response data is passed to the setPosts function, which sets the state variable

posts with the fetched data. If the request fails, an error message is logged to the console using the console.log method.

In summary, this code fetches data from a remote API endpoint and sets it as state in a React component using the setPosts function as shown in Figure 27.

```
const fetchData = async () =>{
  axios.get('https://paiyaa-pk-api.herokuapp.com/users')
    .then(res=>{

        setPosts(res.data)
    })
    .catch(err=>{
        console.log(err)
    })
}
```

Figure 27: Axios request code

Similarly other axios requests are also made for accepting or rejecting the form.

Implementation of android app

What is React Native?

React Native is a mobile application development framework based on React, a popular JavaScript library for building user interfaces. React Native allows developers to build mobile applications for both iOS and Android platforms using a single codebase, making it a popular choice for cross-platform app development. React Native provides a set of pre-built components that can be easily combined to create complex user interfaces, along with access to native APIs for functionality like camera, location, and sensors. It also supports hot reloading, allowing developers to see changes in real-time as they make modifications to the code.

Implementation of front-end for android app using React Native

Our focus is on mechanic and car washer functionality for android app, authorization of mechanic and car washer with all security checks will monitor completely using react native as shown in Figure 28.

Figure 28: Mobile app login page

First the user basically needs to register with the system once the user register then they need to login into the system after login following page occurs.

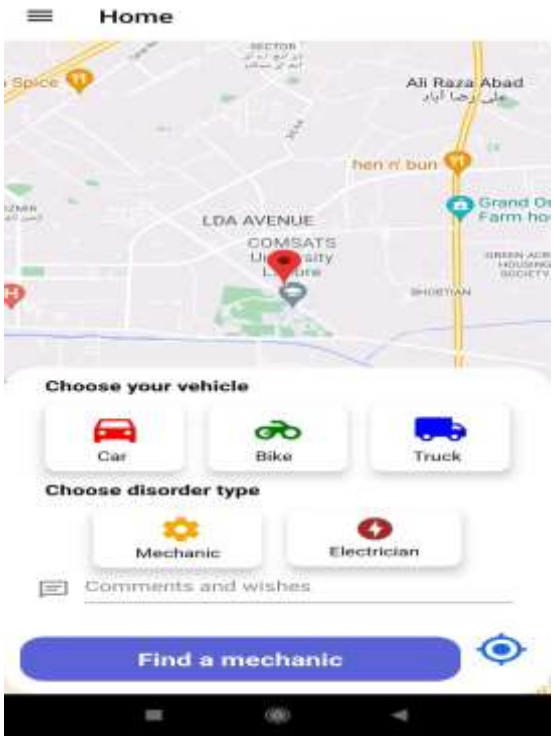

Figure 29: Mobile app home page

The user needs to turn on the location services on their phone so maps can fetch their location. Once the location is enabling the map to fetch their location and the user can make request according to their need as defined in Figure 29.

The application provide service to different user in English and urdu languages for the ease of app usage and decrease the communication barrier as shown in Figure 31 (a) & (b).

Figure 30: User Location Fetch code

In Figure 30, React Native module uses the useState hook to manage the state of the mapRegion, latitude, and longitude variables. The initial value of the mapRegion variable is an object that contains default values for latitude, longitude, and delta values for both.

The component also defines two functions, userLocation() and getAddressFromCoordinates(). The userLocation() function uses the Location API to request permission for the device's location and then retrieves the user's current location using the getCurrentPositionAsync() function. The retrieved location is then used to update the mapRegion state and the latitude and longitude states.

The getAddressFromCoordinates() function uses the fetch() method to make an HTTP request to the LocationIQ API, passing the current latitude and longitude values as parameters. The response from the API is then converted to a JSON object using the json() method, and the display_name field is extracted from the JSON object and set as the value of the setAddress state variable.

Overall, this code snippet demonstrates how React Native can be used with the Location and fetch APIs to retrieve and display the user's current location and address information.

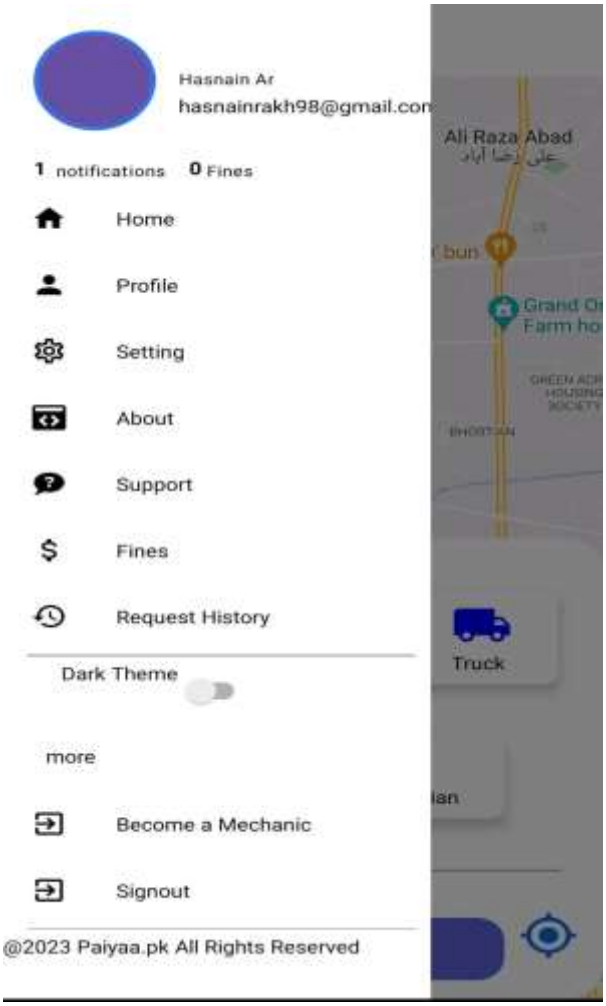

Figure 31 (a): Login domain in English

Figure 31 (b): Login domain in Urdu Language

Function Point Calculation

All the interfaces and reports will be complete, app has various module consoles i.e., Ecommerce website, learn driving, Mechanic/ Washer app and Admin with appropriate screens given below.

Ecommerce website
Login: 1
Signup: 2
Forgot Password: 2
Search Products: 2
View Product details: 1
AR: 1
Manage cart: 1
Payment: 5
Become a seller: 1
Manage products: 1
Edit product details: 1
Add new product: 1
Logout: 1

Learn Driving
Watch driving tutorials: 2
Driving test: 2
Generate Certificate: 1

Mechanic/Washer App
Login: 1
Signup: 2
Forgot Password: 2
Find Mechanic: 5
Update Profile: 2
View Request History: 3
Manager appointments: 2
Chat with mechanic/washer: 1
Customer support: 1
Become a mechanic/washer: 1
Accept/decline customers: 4
Print Challan for fine: 2
Logout: 1

Admin
Login: 1
Accept/decline pending applications: 2
Manage users: 2
Manage Products: 2
Update/delete slide show: 1
Customer service center: 2
Logout: 1

According to the information, application boundary has approximate function point counts as shown in Table 1. Weighting factors are constants that is described by the organization weighting factors. These weighting factors have low, average and high complexity as shown in Figure 32.

Table 1: Functional Points and App Screen Interaction

Functional Units	**FP Screens**
External Inputs	22
External Outputs	10
External Inquiry	12
Internal Logical Files	14
External Interface Files	18

Functional Units	Weighting factors		
	Low	Average	High
External Inputs (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiries (EQ)	3	4	6
External logical files (ILF)	7	10	15
External Interface files (EIF)	5	7	10

Figure 32: Functional units with weighting factors

Specifically for the concerned app and its features average is the weighting complexity. Therefore, Unadjusted Function Point (UFP) is given below.

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} Z_{ij} W_{ij}$$

i = 1-5 that shows total number of functional units, j=1-3 that shows total number of weighting complexity. Z is actual function points of app and w is the weighting complexity values.

UFP = 22*4+10*5+12*4+14*10+18*7 = 452

Software application has disparity is complexities and these variations depends upon 14 factors that denote as $\sum F_i$ having range:

0 - No Influence
1 - Incidental
2 - Moderate
3 - Average
4 - Significant
5 - Essential

Complexity Adjustment Factor (CAF) will be calculated as
CAF = 0.65 + (0.01 * F)
CAF = 0.65 + (0.01 * 41) = 1.06
Final Function Point (FFP) = UFP + CAF
FFP = 452 + 1.06 = 453.16 ≈ 453

Function points will measure the app's functionality from the user's perspective. It determines the approximate cost of development early in software development in the working process.

Conclusions

The development of an e-commerce website and mechanic and car washer app involved a number of complex and interrelated tasks, and as such there were a variety of potential problems and challenges that could have arisen over the app's growth. One of the primary concerns was the integration of different technologies, as the project relied on a number of different platforms and tools to achieve its goals. This required careful planning and coordination to ensure that each component of the project was able to work together seamlessly. Another challenge is to estimate the software development and working Cost/ Budget by user's point of view for the component detail based on the accurate Line of code (LOC) along with the number of interfaces interact with software system.

Acknowledgment

The Authors wish to thank Department of Computer Science COMSATS University Islamabad, Lahore Campus, Pakistan for support and help in learning Programming languages, Frameworks, Computing capacities, Cloud Working, End – End Integration, Augmented reality and Software Hardware Configuration etc.

References

Bootstrap (front-end framework). (2023). In *Wikipedia* https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

Chen, Y., Wang, Q., Chen, H., Song, X., Tang, H., & Tian, M. (2019, June). An overview of augmented reality technology. In *Journal of Physics: Conference Series* (Vol. 1237, No. 2, p. 022082). IOP Publishing.

Firebase. (2023). In *Wikipedia*. https://en.wikipedia.org/wiki/Firebase

MongoDB Atlas. (2023). Fully managed MongoDB in the cloud. https://www.mongodb.com/cloud/atlas/lp/try4?utm_source=google&utm_campaign=search_gs_pl_evergreen_atlas_core_prosp-brand_gic-null_emea-pk_ps

React (software). (2023). In *Wikipedia*. https://en.wikipedia.org/wiki/React_(JavaScript_library)

Sankar, H. H. (2022). What is Socket Programming in C? https://www.scaler.com/topics/socket-programming-in-c/

TensorFlow. (2023). Github. https://github.com/tensorflow/tensorflow

What Object Categories / Labels Are In COCO Dataset? (2018). Amikelive, Technology Blog. https://tech.amikelive.com/node-718/what-object-categories-labels-are-in-coco-dataset/