American University Of The Middle East

College of Engineering and Technology

B.Sc. Electrical Engineering

Graduation Project II

Final Report

Augmented Reality Application for Assisting Workers in the Oil and Gas Industry

Academic Year: *2023 – 2024*

Semester: *Spring 2024*

Course Title and Code: *EE402, CE477*

Name	ID	Major
Fahad alsubaiei	49149	EE
Shaheen alsharaf	50481	EE
Ahmed alhendi	58275	EE
Khalifa alkhalifa	50210	EE
Abdulaziz alfoudari	48624	EE
Supervisor:		

***Abstract*:**

This paper presents the design and implementation of a GPS-based navigation and QR code scanner-based mobile augmented reality (AR) application using Flutter to assist workers in the oil and gas industry. The application aims to enhance worker efficiency and safety by providing real-time navigation guidance and access to relevant information through augmented reality technology. The system utilizes GPS technology to provide environment awareness, allowing workers and visitors to navigate complex industrial environments easily and safely. Additionally, integrating a QR code scanner enables workers to access detailed information about equipment and procedures by scanning QR codes placed at specific locations. The application is developed using the Flutter framework, which allows for cross-platform compatibility and rapid development. The design and implementation details, including the system architecture, user interface, and functionality, are discussed in detail. The proposed application offers a user-friendly and intuitive interface, facilitating seamless interaction and enhancing worker productivity in the oil and gas industry. This application uses both marker-less and marker-based augmented reality (AR) technology, seamlessly integrating multimedia information—such as sound, text, images, and videos—onto real-world surroundings. To assess the efficacy of the application, we designed a survey questionnaire based on the Technology Access Model (TAM3). In an experiment involving 10 participants, this innovative approach for training, orientation, and maintenance operations was well received by the participants. Participants reported a greater sense of support and motivation while using the application. The positive feedback received from participants in the experimental evaluation underscores the effectiveness of the proposed solution in addressing the needs of workers in the oil and gas industry.

Table of Contents

I. Introduction

Kuwait is located in Western Asia, and it borders Saudi Arabia to the south and Iraq to the north. It is situated near the mouth of the Persian Gulf on the northern tip of Eastern Arabia. Kuwait has the sixth-largest stated crude oil reserves in the world, with 101.5 billion barrels. Kuwait was the fourth-largest OPEC producer in 2022, with 2.7 million barrels per day (bpd), placing it among the top 10 producers globally [1] About 50% of Kuwait's GDP is composed of hydrocarbons, with oil accounting for 90% of the country's export revenue and government revenue. Over the next five years, Kuwait plans to invest US$48 billion in exploration and drilling to improve its oil output from proven and mature oil resources.

In today's global economy, the oil and gas industry play a vital role, yet it faces significant challenges that hinder its operational efficiency. Among these challenges, workforce training and skill development in hazardous work environments are of paramount concern. While traditional training methods have some efficacy, they often fall short in providing immediate, context-specific learning experiences essential for safe operations. Moreover, the cost associated with on-site training per employee can be substantial, exceeding $1,000 per individual, as reported by industry sources. Additionally, there are inherent risks and potential downtime associated with on-site training sessions.

A survey conducted by the Oil and Gas Technology Centre highlighted a concerning statistic: approximately 40% of operational incidents in the industry are attributed to human error resulting from inadequate training. This staggering proportion underscores the critical importance of addressing training deficiencies within the oil and gas sector. Failure to adequately prepare the workforce can have far-reaching consequences, not only in terms of safety and operational disruptions but also in terms of financial costs and reputational damage for companies operating in this sector [2].

When performing maintenance operations in oil and gas industry, engineers and technicians encounter challenges in efficiently diagnosing, repairing, and maintaining complex machinery and equipment. Traditional manuals and printed documentation are often cumbersome, and they can lead to errors, delays, and safety hazards. There is a need for a more effective and user-friendly solution that leverages augmented reality technology to provide real-time, context-aware information and guidance to technicians, thereby enhancing their productivity, reducing errors, and improving safety [3].

In light of these challenges, there is a pressing need for innovative training solutions that can provide effective, context-specific learning experiences to oil and gas workers. By leveraging technologies such as augmented reality (AR), QR codes, and GPS-based navigation, companies can enhance their training programs, mitigate risks, and empower employees with the skills and knowledge needed to perform their duties safely and efficiently in hazardous environments.

a) Augmented Reality

Augmented reality (AR) is a technology that merges digital images and information with the real world, typically viewed through devices like phones or tablets' cameras [4]. This allows users to see virtual objects superimposed onto their physical surroundings, enriching their environment with computer-generated visuals and data. Marker-based augmented reality relies on predefined visual markers—such as images, patterns, or distinct objects—to anchor virtual content onto the real world. These markers are easily detectable and trackable by AR software, enabling the overlay of digital content when detected. However, if used outside the designated marker area, the application may not function as expected. While marker-based AR is simpler to implement and popular, it has inherent limitations. In contrast, marker-less AR relies heavily on the device's sensors, like the camera, GPS, and accelerometers, to detect and track the physical environment, accurately placing digital objects within it. This creates a seamless AR experience where digital objects appear anchored to the user's real-world surroundings. Location-based AR [5] utilizes GPS and other sensors to position virtual objects in the user's environment based on their geographical location. A prime example is the popular mobile game Pokémon GO. Hybrid augmented reality combines various AR techniques, often blending marker-based and marker-less approaches, to create more dynamic and versatile AR experiences.

b) GPS-based Navigation in Oil and Industries

GPS-based navigation plays a pivotal role in industries across various sectors due to its numerous benefits and applications. Industrial facilities, such as manufacturing plants, refineries, or construction sites, can be vast and complex [6]. A GPS-based navigation app provides detailed maps and directions to help visitors navigate these environments efficiently, reducing the risk of getting lost and minimizing disruptions to operations. Safety is paramount in industrial settings. The app can include safety information, hazard alerts, and emergency procedures specific to the site. Visitors receive real-time notifications and guidance on safety protocols, ensuring they remain aware of potential risks and adhere to safety regulations. It can also be integrated into training programs for oil and gas workers to familiarize them with site layouts, emergency procedures, and safe navigation practices. This helps enhance worker competence, situational awareness, and overall safety culture within the industry. Also, the oil and gas industry are subject to stringent regulatory requirements regarding worker safety, environmental protection, and operational standards. GPS-based navigation systems help companies comply with these regulations by providing accurate data on asset locations, personnel movements, and environmental parameters. Additionally, industrial environments are dynamic, with changes in operations, construction activities, or maintenance schedules occurring regularly. The app provides real-time updates and notifications on site conditions, closures, or operational changes, ensuring visitors have the latest information at their fingertips.

c) QR Codes in Oil and Industries

QR codes have found many applications in the oil and gas industry, facilitating various aspects of operations, safety, and maintenance [7]. QR codes are used to track equipment, machinery, and tools. We can also link the QR codes placed on equipment to maintenance procedures and inspection checklists. Maintenance technicians can scan these codes to access manuals, procedures, and relevant data, streamlining maintenance tasks and ensuring compliance with safety regulations. QR codes can be placed on safety equipment, such as fire extinguishers and emergency exits, providing quick access to safety procedures and training materials. Employees can scan these codes to access instructional videos, guidelines, and emergency contact information. QR codes are utilized for asset identification and management in pipeline systems [8]. Each section of the pipeline can be marked with a QR code containing information about its specifications, installation date, inspection history, and maintenance requirements. This aids in the efficient management and monitoring of pipeline networks.

d) Objectives

The objectives of this project are as follows:

- Design an intuitive and easy-to-use interface that allows workers to navigate industrial environments and access relevant information seamlessly.
- Integrate GPS technology to provide real-time location tracking and navigation guidance within complex industrial sites, ensuring workers can navigate safely and efficiently.
- Develop a QR code scanning feature to enable workers to access detailed information about equipment, procedures, and safety protocols by scanning QR codes placed at specific locations.
- Utilize augmented reality (AR) technology: Implement both marker-less and marker-based AR technology to overlay multimedia information—such as text, images, and videos—onto real-world environments, enhancing worker understanding and engagement.
- Utilize the Flutter framework to develop a mobile application that is compatible with both Android and iOS devices, ensuring widespread accessibility for workers using different mobile platforms.
- Assess the usability, effectiveness, and user satisfaction of the developed application through user testing and evaluation sessions with workers in the oil and gas industry.
- Ensure that the application operates smoothly, with minimal latency and downtime, even in challenging industrial environments with limited connectivity and varying lighting conditions.

II. Design Development

In this chapter, a detailed description of the design and its different elements should be given.

1. Proposed Design

a) Mobile Augmented Reality based Solution

In this project , we propose a GPS and QR code-based mobile augmented reality (AR) application aiming to assist workers in maintenance operations within the oil and gas industry. The application will open to a home screen where users can choose between GPS navigation or QR code scanning. When GPS navigation is selected, users are presented with a map interface showing their current location and nearby points of interest (POIs). POIs relevant to maintenance operations, such as equipment locations or workstations, are highlighted on the map. When QR code scanning is selected, users can activate the device's camera to scan QR codes placed on equipment or machinery. Upon scanning, relevant information about the scanned item is displayed on the screen. The application utilizes GPS technology to provide real-time location awareness to users. This allows workers to navigate complex industrial environments with ease and safety. Relevant POIs, such as equipment locations, emergency exits, or maintenance stations, are marked on the map. Users can select a POI to view additional information or receive navigation instructions. Users can scan QR codes placed on equipment or machinery to access detailed information about the scanned item. This information may include equipment specifications, maintenance history, troubleshooting guides, or safety procedures. Upon scanning a QR code, relevant AR content is superimposed onto the real-world view through the device's camera. This may include instructional videos, 3D models, or animated overlays providing step-by-step maintenance instructions.

The application utilizes markerless AR technology to overlay digital content onto the real-world environment without the need for predefined markers. This allows for seamless integration of AR content into the user's surroundings. AR overlays may include multimedia content such as instructional videos, textual instructions, interactive 3D models, or audio guides to assist users during maintenance operations.

b) Application Requirements

The main requirements of the proposed systema re listed below:

- The application should be compatible with both Android devices to cater to the diverse range of smartphones used by workers.
- The application must integrate GPS functionality to provide real-time navigation assistance within industrial environments, ensuring workers can easily locate equipment and facilities.
- The application should support QR code scanning capabilities to allow workers to access detailed information about equipment, procedures, and safety protocols by scanning QR codes placed at specific locations.
- The application should leverage augmented reality technology to overlay relevant information, such as equipment manuals, troubleshooting guides, and safety instructions, onto the real-world environment.
- The application's interface should be intuitive and user-friendly, enabling easy navigation and seamless interaction with AR overlays and information.
- The application should deliver high performance, with quick response times and minimal latency, to provide users with a smooth and seamless experience.
- The application should allow for customization to tailor the AR overlays and information presentation based on specific user roles, preferences, and operational requirements.
- Design a paper-based questionnaire to evaluate the efficacy of the application using TAM frameworks.
- Test the questionnaire with a sample of respondents to evaluated performance and identify any potential issues or confusion.
- Comprehensive training materials and user support should be provided to ensure workers can effectively utilize the application's features and maximize its benefits for maintenance operations.
- Collect the survey responses and analyze the data to identify patterns, trends, and insights related to the efficacy of the application. Use quantitative analysis for closed-ended questions and qualitative analysis for open-ended responses.

- Prepare a report summarizing the survey findings, including key findings, recommendations, and actionable insights for improving the application. Present the report to stakeholders involved in the development and implementation of the application.

c) *Application Constraints*

The constraints for the design and implementation of a GPS and QR Code-Based Mobile Augmented Reality Application for Maintenance Operations in the Oil and Gas Industry (OGAR) include:

- The project is implemented targeting the Android device as implementing it iOS may take more time.
- The handling of sensitive operational data and user information raises concerns about data security and privacy. Implementing robust security measures to protect against unauthorized access, data breaches, and privacy violations is essential.
- This project is subject to several financial constraints, as it is initiated by students and therefore operates within a limited budget. Additionally, there is a tight time constraint, requiring completion within a 16-week timeframe.
- The project involves the collection of GPS coordinates for 25 buildings, as well as the gathering of AR content relevant to each location. Furthermore, QR codes need to be installed on at least 25 devices, and pertinent data for equipment and machinery must be collected.
- The application should be tested with KoC employees, and their feedback should be collected.
- The application's performance may be affected by environmental factors such as GPS signal strength, network connectivity, and lighting conditions. Addressing these environmental constraints is essential for ensuring reliable operation in real-world conditions.
- The success of the application depends on user acceptance and adoption. Overcoming resistance to change, addressing user preferences and needs, and providing adequate training and support are essential for promoting user engagement and adoption.
- Integrating the application with existing systems, databases, and infrastructure within the oil and gas industry may pose integration challenges. Ensuring seamless interoperability and data exchange between the application and other systems is critical for its effectiveness.
- Providing ongoing maintenance, updates, and technical support for the application post-deployment is essential. Constraints related to resource availability, response time, and troubleshooting may impact the application's long-term sustainability and success.
- The application's functionality may be limited by geographical factors such as GPS coverage, geographic features, and regulatory restrictions in certain regions or countries. Adapting the application to different geographical contexts and regulatory environments is necessary for global deployment.

d) *Design Criteria*

- The application should provide an intuitive and seamless user experience, ensuring that users can navigate through the app easily and accomplish tasks efficiently.

- The UI design should be visually appealing, with clear and consistent design elements such as buttons, icons, and menus. It should also follow platform-specific design guidelines to maintain familiarity for users.
- The application should be responsive and performant, with fast loading times and smooth transitions between screens. It should also consume minimal device resources such as CPU, memory, and battery power.
- The application must provide precise GPS-based navigation to ensure workers can locate maintenance sites with high accuracy.
- The system should offer real-time updates on maintenance schedules, equipment status, and safety protocols to keep workers informed.
- The interface should be intuitive and easy to navigate, allowing workers to access information quickly and efficiently.
- Robust security measures must be implemented to protect sensitive operational data and prevent unauthorized access.
- The AR content should be of high quality, providing clear instructions, diagrams, and animations to aid workers in maintenance tasks.
- The application should be optimized to work across various devices and screen sizes commonly used in the industry.
- The design should consider accessibility features to accommodate workers with disabilities or special needs.

2. Detailed High-Level Specifications

Hihg level block diagram of the OGAR mobile application is shown in Figure 1. The system architecture makes clear that the OGAR consists of a distant data server and a mobile client software running on the Android OS platform. They are connected via a wireless network. A client submits a request, which is received by the server, which then replies by sending the client the results of the processing. Every connected client receives real-time data synchronization from the server in JSON (JavaScript Object Notation) format [9], which allows for immediate mobile device access.

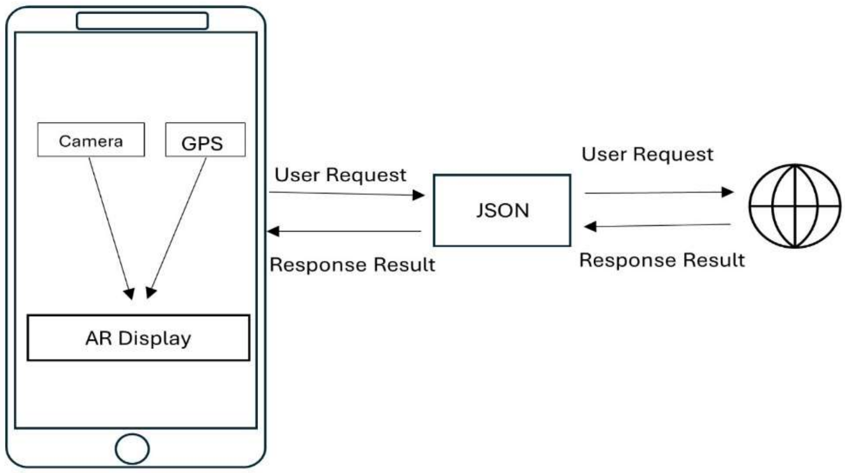

Figure 1: High level block diagram of OGAR

Table 1. Hardware specifications

S. No.	Hardware	Specification
1	RAM	16GB
2	ROM	12GB
3	smartphone camera	12MP or Higher
4	CPU	Qualcomm Snapdragon 732 or higher, iPhone 12 bionic chip or higher
5	GPU	Qualcomm Adreno 660 or higher

Flutter is an open-source UI software development toolkit created by Google. It is used to build natively compiled applications for mobile, web, and desktop from a single codebase. Flutter allows developers to write code once and deploy it on multiple platforms, including iOS, Android, web, and desktop. This is achieved through its reactive framework, which enables hot reload functionality, allowing developers to see changes instantly without losing state. Flutter applications are crafted using Dart, an object-oriented programming language [10]. Dart operates on a virtual machine implemented in C/C++. In Flutter app development, rendering widgets on the device screen and accessing services like geolocation, sensors, camera, audio, Bluetooth, etc., is facilitated using Canvas and Events. Additionally, the application is compiled using Ahead of Time (AOT) compilation rather than Just in Time (JIT) compilation, as seen in JavaScript.

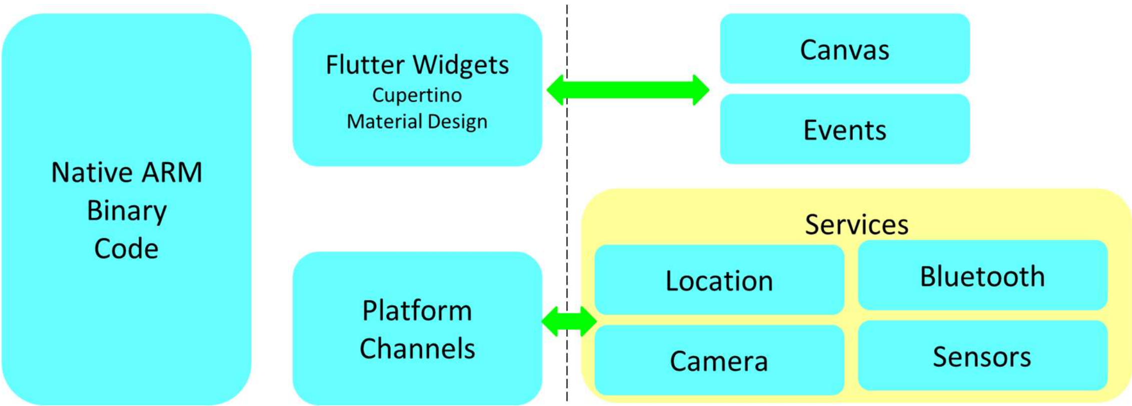

Figure 2: Flutter approach.

Table 2. Software specifications

S. No.	Software	Specification
1	Android OS	Android 11 or higher
2	Kotlin	1.4.2
3	Gradle	3.1.3
4	Android Studio	4.1 or higher
5	Flutter Student Development	Kit 2.2.3 or higher
6	Google ARCore	1.3.2

3. Detailed Low-Level Specifications

a) GPS – Global Positioning System

The global positioning satellite system, or GPS system was developed the US. There are 24 satellites in orbit around the Earth, and they are all synced to an atomic clock. Each satellite continuously sends its position coordinate and the precise time associated with the location. Every GPS handheld device obtains a position by receiving location data from a minimum of four satellites. Subsequently, the device computes a position fix at the device's location. This is achieved by squaring the satellites' known positions with the time it takes for the signal to reach the device. One satellite is utilized for time synchronization, while three satellites are used to triangulate position. Many nations have developed their own individual global positioning systems. The Russians have their own system called Glonass, the Chinese have Baidu, and the European Union has Galileo. More satellites were added by Japan, whose system is now known as qzs. To be precise, the global positioning module on a phone is called gnss instead of GPS. GPS receivers in phones are basically simply a standardized chip; I'm assuming that most are produced by Broadcom and are part of the Wi-Fi Bluetooth GPS SOC embedded into phone motherboards. this is because every positioning satellite is used by the positioning system. Wi-Fi Bluetooth GPS chip on the phone: Wi-Fi and GPS work together to establish a device's position in current computing gadgets.

b) QR Code

A QR code is an image that can be instantaneously decoded with embedded information using a smartphone's camera. Each QR code is made up of dots and black squares that together encode unique data. This code becomes readable by your smartphone and converts the data into a human-readable format. A QR reader can recognize a standard QR code by detecting the presence of three large squares located outside the code. Upon identifying these three shapes, the reader determines that everything enclosed within the squares constitutes a QR code. Subsequently, the QR reader dissects the QR code by segmenting it into a grid for analysis.

c) Flutter – Mobile Application Development Platform

During development, Flutter apps operate within a virtual machine (VM) environment, enabling stateful hot reload of changes without requiring a complete recompilation. When ready for release, Flutter apps are compiled directly into machine code, supporting Intel x64 or ARM instructions, or into JavaScript if intended for web deployment. The framework, which is open source and licensed under the permissive BSD license, boasts a robust ecosystem of third-party packages that complement the core library functionality. Flutter is architected as a modular, layered system comprising a series of standalone libraries, each relying on the layer beneath it. There is no inherent hierarchy granting privileged access to lower layers, and every component of the framework is intentionally crafted to be optional and interchangeable.

Flutter applications are packaged to the underlying operating system much like any other native application. A platform-specific embedder serves as the entry point, interfaces with the operating system for access to services such as rendering surfaces, accessibility, and input, and oversees the message event loop. The embedder is implemented in a language suitable for the platform, currently Java and C++ for Android, Objective-C/Objective-C++ for iOS and macOS, and C++ for Windows and Linux. Leveraging the embedder, Flutter code can either be integrated into an existing application as a module or constitute the entirety of the application. While Flutter includes several embedders for common target platforms, alternative embedders are also available.

At the heart of Flutter lies the Flutter engine, predominantly coded in C++, which furnishes the fundamental elements required to power all Flutter applications. Tasked with rasterizing composited scenes whenever a new frame necessitates painting, the engine serves as the groundwork for Flutter's core API implementation. This encompasses graphics functionality (utilizing Impeller on iOS and soon on Android, and Skia on other platforms), text layout, file and network I/O, accessibility features, plugin architecture, as well as a Dart runtime and compilation toolchain.

Typically, developers engage with Flutter through the Flutter framework, which furnishes a contemporary, reactive framework scripted in the Dart language. This framework encompasses a diverse array of platform, layout, and foundational libraries, organized into a hierarchical structure of layers. Progressing from the bottom to the top, these layers consist of Foundational classes and essential services such as animation, painting, and gestures, which provide convenient abstractions over the underlying foundation. It also contains the rendering layer, which abstracts the handling of layout tasks. Within this layer, developers can construct a hierarchy of render-able objects and dynamically manipulate them, with the layout automatically adjusting to reflect modifications. The widgets layer serves as a composition abstraction. Each render object in the rendering layer corresponds to a class in the widgets layer. Furthermore, the widgets layer facilitates the definition of reusable combinations of classes. This layer also introduces the reactive programming model. The Material and Cupertino libraries furnish extensive sets of controls utilizing the composition primitives of the widgets layer to implement the Material or iOS design languages.

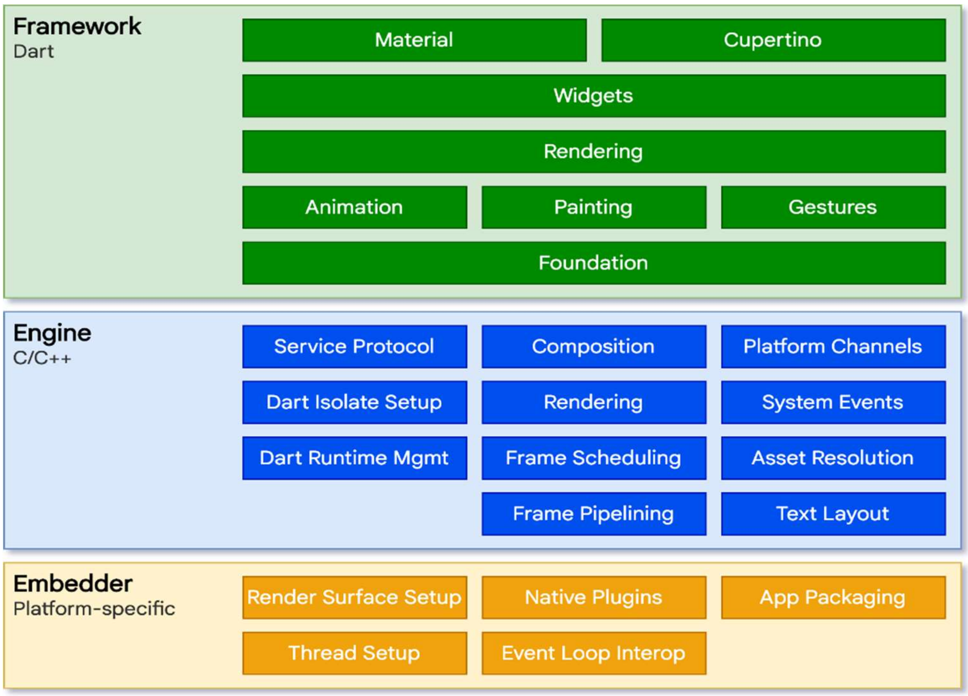

*Figure 3. Flutter architec*ture [11]

Flutter presents numerous advantages such as use of Dart programming language. Dart boasts an extensive repository of software packages, enabling developers to enhance the functionalities of their applications effortlessly. A single codebase suffices for developing applications across both Android and iOS platforms, with the potential for Flutter's expansion to additional platforms in the future. Due to its unified codebase, Flutter necessitates less testing, as automated tests need only be written once for both platforms. Flutter's straightforwardness renders it ideal for rapid development, bolstered by its capacity for customization and scalability. Developers wield complete control over widgets and layout with Flutter, fostering a highly tailored development experience. Flutter provides robust developer tools, including the remarkable hot reload feature, facilitating seamless and efficient development cycles.

d) Dart Programming Language

Google created the general-purpose programming language known as Dart, which is available as an open-source project. It well known for its versatility in building various types of applications, ranging from web and mobile to server-side applications. It features a familiar syntax inspired by languages like Java and JavaScript, making it accessible to developers with diverse backgrounds. Dart supports both object-oriented and functional programming paradigms, offering features like classes, interfaces, mixing, and asynchronous programming with Futures and Streams. Interestingly, Dart does not provide arrays, in contrast to certain other languages. However, it makes up for this by offering collections that may mimic array-like data structures, as well as optional type and generics.It's the primary language used for building applications with the Flutter framework, renowned for its fast development cycles and high-performance UIs. Additionally, Dart is known for its strong type system, enabling developers to catch errors at compile time and write more maintainable code.

III. PROJECT REALIZATION AND PERFORMANCE OPTIMIZATION

1. Planned implementation and experiments

a) Hardware Platform Selection

These are many augmented reality devices available on the market, each offering unique features and functionalities tailored to different use cases and industries [12]. Some of the popular AR devices are discussed here.

i. Wearable Devices

Wearable devices, such as smart glasses and headsets, offer hands-free AR experiences that overlay digital content directly onto the user's field of view [13]. Examples include Microsoft HoloLens, Magic Leap One, and Google Glass Enterprise Edition. These devices typically feature integrated displays, cameras, and sensors for tracking and interacting with virtual content.

ii. Automotive Head-Up Displays (HUDs)

Automotive HUDs project AR-based information, such as navigation directions, vehicle speed, and safety alerts, onto the windshield or a dedicated display in the driver's field of view. These AR-enhanced HUDs provide drivers with relevant information without distracting them from the road.

iii. Smart Cameras

Smart cameras and wearable cameras equipped with AR capabilities can capture real-world scenes and overlay digital annotations, information, or effects onto the live video feed. Devices like the Snap Spectacles and GoPro HERO cameras with AR functionality enable users to create AR-enhanced videos and photos.

iv. Immersive VR Headsets:

Immersive virtual reality (VR) headsets, such as the Oculus Rift, HTC Vive, and PlayStation VR, can also support AR experiences through passthrough video or mixed reality modes. While primarily designed for VR, these headsets can overlay real-world video feed or blend virtual and real environments for hybrid AR/VR experiences.

v. Smartphones and Tablets:

Smartphones and tablets are ubiquitous AR platforms, capable of running AR applications using built-in sensors such as cameras, gyroscopes, and accelerometers. ARKit for iOS and ARCore for Android provide software frameworks that enable developers to create AR experiences for mobile devices.

We chose handheld smartphones as the platform for our project due to their widespread accessibility. Almost everyone possesses a smartphone, eliminating the necessity to invest in specialized hardware solely for using our application. This decision ensures that our solution can reach a broader audience without requiring users to acquire additional devices, thereby enhancing convenience and usability. Additionally, leveraging smartphones as the platform enables seamless integration into users' daily lives, as they are already accustomed to using these devices for various tasks and applications. Overall, selecting smartphones as our platform aligns with our goal of maximizing accessibility and minimizing barriers to adoption for our project.

b) Android vs iOS

The selection of the target platform between iOS and Android posed another challenge for our project [14]. While a majority of Kuwaiti nationals own iPhones, we faced the reality that the overwhelming majority of workers in our target demographic use Android phones. This presented a crucial decision point for our team. Considering the widespread adoption of Android devices among our target users, we made the strategic decision to prioritize Android as our primary development platform. This choice was driven by the desire to maximize the reach and accessibility of our application among the largest segment of our user base. While recognizing the popularity of iOS devices, particularly among certain demographics, we weighed the practical considerations of reaching the broadest audience possible. By focusing on Android, we aimed to ensure that our application could effectively serve the needs of the majority of users within our target market, including workers who rely heavily on Android devices for their day-to-day activities.

Ultimately, our decision to prioritize Android over iOS was guided by a strategic assessment of user demographics, market dynamics, and the need to align our development efforts with the preferences and behaviors of our target audience.

c) *Software Development Platform Selection*

There are several mobile application development platforms available, each with its own set of tools, frameworks, and programming languages [15], [16]. Some of the most popular ones include:

i. React Native

Developed by Facebook, React Native [17] is a popular JavaScript framework for building cross-platform mobile applications. It allows developers to use React, a JavaScript library for building user interfaces, to create native-like mobile apps for iOS and Android.

ii. Unity

Unity is a game development platform that can also be used to create cross-platform mobile apps. While primarily known for game development, Unity offers powerful tools for creating interactive and immersive experiences across multiple platforms, including iOS, Android, and desktop.

iii. NativeScript

NativeScript is an open-source framework for building cross-platform mobile apps using JavaScript, TypeScript, or Angular. It provides access to native APIs and UI components, allowing developers to create truly native mobile apps for iOS and Android from a single codebase.

iv. SwiftUI

SwiftUI [18] is a declarative UI framework introduced by Apple for building native iOS, macOS, watchOS, and tvOS apps using Swift programming language. While SwiftUI is not cross-platform like Flutter, it offers a modern and intuitive way to create user interfaces for Apple platforms.

v. Cordova

Apache Cordova (formerly PhoneGap) is a framework for building cross-platform mobile apps using web technologies such as HTML, CSS, and JavaScript. It enables developers to create hybrid mobile apps that run on iOS, Android, and other platforms, accessing native device features through plugins.

vi. Flutter

Flutter is an open-source UI software development kit (SDK) created by Google, used for building natively compiled applications for mobile, web, and desktop from a single codebase. It allows developers to write code once and deploy it across multiple platforms, saving time and effort in development. Flutter allows for the development of cross-platform applications, meaning that a single codebase can be used to deploy apps on both iOS and Android platforms.

d) *Flutter*

Flutter is an open-source UI software development kit (SDK) created by Google, used for building natively compiled applications for mobile, web, and desktop from a single codebase. It allows developers to write code once and deploy it across multiple platforms, saving time and effort in development. Flutter allows for the development of cross-platform applications, meaning that a single codebase can be used to deploy apps on both iOS and Android platforms. This significantly reduces development time and costs, as developers do not need to write separate codebases for each platform. Flutter's hot reload feature enables developers to see changes made to the code in real-time on emulators, simulators, and physical devices. This speeds up the development process and allows for rapid iteration and experimentation. Flutter apps are compiled directly to native machine code, resulting in high-performance applications that run smoothly on both iOS and Android devices. The Flutter engine's architecture also allows for efficient rendering and animations, leading to a responsive user experience. Flutter comes with a rich set of customizable UI components called widgets, which can be easily combined and customized to create beautiful and immersive user interfaces. Additionally, Flutter's Material Design and Cupertino libraries provide pre-designed UI components that adhere to platform-specific design guidelines. Flutter has a vibrant and active community of developers who contribute to the ecosystem by creating plugins, packages, and tutorials. This community support ensures that developers have access to a wealth of resources and assistance when building Flutter applications. Flutter provides plugins that allow developers to access native device features and functionalities, such as camera, geolocation, sensors, and more. This enables developers to create feature-rich applications that leverage the full capabilities of the underlying platform. Flutter is backed by Google and has seen significant adoption by both large corporations and startups. Its growing popularity and strong support from Google ensure that Flutter will continue to evolve and improve in the future, making it a reliable choice for long-term mobile application development projects.

After considering all the above mentioned featues, Flutter is a better option for developing mobile applications because of its cross-platform interoperability, quick development cycles, high performance, rich user interface components, community support, access to native functionality, and futureproofing.

i. Flutter Directory Structure

The directory structure of an OGAR project provides information about the organization of files and folders in a logical manner. The directory structure along with the description of each description is given in Figure xx.

- **android** – Auto generated source code to create android application
- **ios** – Auto generated source code to create ios application
- **lib** – Main folder containing Dart code written using flutter framework
- **ib/main.dart** – Entry point of the Flutter application
- **test** – Folder containing Dart code to test the flutter application
- **test/widget_test.dart** – Sample code
- **.gitignore** – Git version control file
- **.metadata** – auto generated by the flutter tools
- **.packages** – auto generated to track the flutter packages
- **.iml** – project file used by Android studio
- **pubspec.yaml** – Used by **Pub**, Flutter package manager
- **pubspec.lock** – Auto generated by the Flutter package manager, **Pub**
- **README.md** – Project description file written in Markdown format

ii. Flutter Widgets

Each component of the graphical user interface is depicted by either a widget or a collection of widgets. In Flutter, a widget serves as a portrayal of a graphical element, which may include:

- Structural components such as text, images, shapes, or buttons
- Styling attributes like fonts and colors
- Schematic attributes such as borders and padding

Widgets are organized in a hierarchical manner based on composition. Each widget is nested within another, inheriting properties from its parent. Unlike traditional frameworks, Flutter lacks a distinct "application" object; instead, this role is assumed by the root widget. Interaction with the user in Flutter involves updating the widget hierarchy, achieved by comparing the new widget with the previous one, and modifying only the elements that differ. Additionally, complex widgets can be constructed by combining multiple simple widgets. It's worth noting that Flutter does not employ native widgets from the underlying platforms (Android or iOS); rather, it utilizes platform-specific representations, as Flutter encompasses a comprehensive array of widgets, layouts, and themes tailored for each platform.

StatelessWidget widgets are those that maintain a constant state and are independent of other components in the interface. Examples include Text, Icon, and IconButton. A *StatelessWidget* widget is responsible for defining a portion of the user interface by recursively assembling other widgets that comprise the interface. This recursive build process is depicted in Figure 11. In every Flutter application, the starting point is the void *main ()* function located in the *main.dart* file. This function, utilizing shorthand syntax, invokes the runApp function with the root widget class constructor as an argument. The material.dart package which facilitates rapid application development by providing a comprehensive collection of fundamental widgets.

StatefulWidget widgets are those whose state is altered through user interaction with the application, as well as by modifications to other application parameters. Each stateful widget possesses an initial state that stores pertinent information about the widget. To enable dynamic content in a user-created widget, it must inherit the StatefulWidget widget.

iii. Hot Reloading

Flutter's hot reload feature is a game-changer for mobile app development, providing developers with a fast, efficient, and intuitive way to iterate on their code and accelerate the development process. By enabling instant updates and preserving app state, hot reload enhances productivity, fosters creativity, and facilitates rapid iteration and testing, ultimately leading to the delivery of high-quality apps in less time. With hot reload, developers can make changes to their Flutter code, such as UI modifications, adding new features, or fixing bugs, and see the updates applied to the running app almost instantly. This near-instant feedback loop accelerates the development cycle and enhances productivity. Unlike a full restart, which resets the app state and navigation stack, hot reload preserves the app's state during code changes. This means that developers can continue interacting with the app and testing its functionality seamlessly while making iterative changes to the code. It can selectively reload only the modified portions of the code, rather than reloading the entire app. This results in faster update times, as only the necessary changes are applied, minimizing disruption to the developer's workflow and user experience.

e) *QR-Scanner Code Implementation*

Implementing QR code scanning and augmented reality (AR) functionality can significantly enhance the user experience of your application by bridging the gap between the physical and digital worlds.

i. QR Code Scanning

Users utilize the QR code scanner within the application to scan QR codes placed in physical or digital environments, such as posters, products, advertisements, or websites. The application decodes the information embedded in the QR code, which may include URLs, text, contact information, product details, or other relevant data. This functionality offers users a quick and convenient way to access additional information or perform actions related to the scanned content without the need for manual input or navigation.

ii. QR Code Placement

QR codes were placed on various machinery, equipment, and products within the oil and gas industry industrial setting. Each QR code contains unique information relevant to the item it's attached to. Upon scanning the system retrieves relevant data associated with the QR code. The data could be specifications, datasheet, and drawing etc. The QRcodes are placed are placed on various items as given below:

- *Equipment and Machinery:* Installed QR codes on equipment, machinery, and assets such as pumps, valves, compressors, and pipelines. Users can scan these QR codes to access maintenance manuals, operating instructions, safety protocols, and troubleshooting guides specific to each piece of equipment.

- *Safety Signage and Labels:* Place QR codes on safety signage, labels, and hazard communication materials throughout the facility. Users can scan these QR codes to access safety procedures, emergency response protocols, risk assessments, and incident reporting forms.

- *Control Panels and Instrumentation:* Attach QR codes to control panels, instrumentation panels, and process control equipment. Users can scan these QR codes to access control diagrams, process flowcharts, instrumentation schematics, and control system documentation.

- *Work Permits and Job Sites:* We place QR codes on work permits, job tickets, and job site signage at various locations within the facility. Users can scan these QR codes to access work instructions, task checklists, permit-to-work documentation, and site-specific safety guidelines.

- *Storage Tanks and Vessels:* QR codes can be placed on storage tanks, vessels, and containers used for storing chemicals, fuels, and other materials. Users can scan these QR codes to access material safety data sheets (MSDS), tank inspection reports, product specifications, and handling instructions.

- *Training and Education Areas:* Install QR codes in training rooms, educational facilities, and simulation centers used for employee training and development. Users can scan these QR codes to access training videos, interactive simulations, competency assessments, and learning resources.

- *Field Locations and Remote Sites:* QR codes can be deployed at field locations, remote sites, and offshore platforms where internet connectivity may be limited. Users can scan these QR codes to access offline-ready AR content, downloadable resources, and emergency contact information.

- *Maintenance and Inspection Points:* Place QR codes at maintenance and inspection points throughout the facility, including equipment access points, inspection checkpoints, and testing locations. Users can scan these QR codes to log maintenance activities, record inspection findings, and access historical data.

An example of QR code placed on file is shown in Figure below. The QR code is placed on the figure of rig and its corresponding data is stored in JSON files placed on the repository and available online.

Figure 4: Example of QR code

The JSON file contains a structured list of equipment or machinery within the industrial facility, each associated with a unique QR code. This file serves as a comprehensive inventory detailing the assets present in the facility and facilitates easy access to relevant information through QR code scanning. JSON, or JavaScript Object Notation, is a lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate. It is based on a subset of the JavaScript programming language and is commonly used for transmitting data between a server and a web application as an alternative to XML. The name of the equipment is provided as an array of elements separated by commas as shown in Figure fff.

```
"types": [
    "XLS Fire Alarm",
    "Flow Meter",
    "Water Injection Pump",
    "Heat Exchange Unit",
    "Oil and Gas Separator",
    "Solenoid Valve",
    "Hermetic Booster",
    "SCADA PKS",
    "Control Logix PLC",
    "PowerFlex VFD",
    "Fire Extinguisher",
    "Security Processor",
    "Drilling Rig",
    "Ball Valve",
    "Butterfly Valve",
    "Boiler",
    "Compressor",
    "Generator",
    "Synchronous Motors",
    "Gas Analyzer",
    "Radar Level Sensor",
    "Wireless Ultrasonic Sensor",
    "Fiber Disc Filter",
    "Muffin Monster",
    "Gas Liquid Separator"
]
}
```


Figure 5: JSON file contents showing equipment and machinery.

iii. Augmented Reality (AR) Content

Upon scanning a QR code, the application triggers the display of AR content overlaid onto the user's physical environment through their device's camera. We include five different multimedia elements for QR-scanner component of the mobile application. One user scans a QR code, upon successful recognition, a new screen pops up which prompts the user to select one of the elements and view its content. These multimedia elements include:

1. **URLs:** Once user clicks these options, he can access the product webpage other online resources directly from the AR experience. This option allows worker to get access resources related to product quickly without wasting any time. Furthermore, a newly hired worker does not need to spend time on its search and verification.
2. **Videos:** The videos invoke the tutorial or maintenance video of the product whose QR code is scanned, and these contents can are played within the AR environment.
3. **Drawings**: Interactive drawings or annotations are superimposed onto real-world objects by choosing this option. These drawings convey detailed information about the geometry, dimensions, tolerances, materials, and finishes of machine components, ensuring clarity and accuracy in the manufacturing process. They also provide visual representations of complex mechanical systems, components, and assemblies, allowing stakeholders to visualize how parts fit together and function within a larger system.

4. **PDF Text:** Documents or textual information contained within the QR code can be displayed as virtual text overlays. These documents may contain information about the product, can be datasheet or a product catalogue. Datasheets provide detailed specifications and parameters of electrical components, such as voltage ratings, current ratings, frequency response, temperature range, and physical dimensions. This information helps engineers and designers understand the capabilities and limitations of components and select the most suitable ones for their applications.
5. **Vendor Contact:** Contact information such as phone numbers, email addresses, or social media profiles can be displayed for easy access. Having access to vendor contact information allows engineers and technicians to reach out to technical support teams for guidance and assistance during troubleshooting efforts. Maintenance personnel can also inquire about warranty coverage, eligibility criteria, and procedures for initiating warranty claims to address malfunctioning or defective products.

By integrating AR content, the application transforms static QR codes into dynamic and interactive experiences, providing users with richer and more engaging interactions with their surroundings.

The value for each machinery or equipment is provided in the annotation. Json file. It contains a collection of objects, and each object is enclosed in curly brackets“{}” and separated by a comma from another object. Each object has five attributes, each of which is a sequence of characters enclosed within double quotes " ".


```
{
  "QRType": "Water Injection Pump",
  "video_link": "https://www.youtube.com/watch?v=XpcCUtYzwy0",
  "pdf_link": "https://github.com/MuhammadNadeemAUM/OilandGas/blob/main/QRScannerData/Pump-PBPS-100.pdf",
  "website_url": "https://ifsolutions.com/products/water-injection-pump-package/",
  "map_url": "https://github.com/MuhammadNadeemAUM/OilandGas/blob/main/QRScannerData/PumpDrawing.jpg",
  "booking_url": "https://ifsolutions.com/contact-us/"
},
{
  "QRType": "Heat Exchange Unit",
  "video_link": "https://www.youtube.com/watch?v=-SZiXB20V24",
  "pdf_link": "https://github.com/MuhammadNadeemAUM/OilandGas/blob/main/QRScannerData/Heat-exchanger.pdf",
  "website_url": "https://aager.de/ergil/heat-exchangers/",
  "map_url": "https://github.com/MuhammadNadeemAUM/OilandGas/blob/main/QRScannerData/Heat-Exchange-Drawing.jpg",
  "booking_url": "https://aager.de/contact/"
},
{
  "QRType": "Oil and Gas Separator",
  "video_link": "https://www.youtube.com/watch?v=sPDvpZiIX7o",
  "pdf_link": "https://github.com/MuhammadNadeemAUM/OilandGas/blob/main/QRScannerData/OilandGasProductionSeparator.pdf",
  "website_url": "https://www.hcpetroleum.hk/product-detail/Oil-Gas-Production-Separator",
  "map_url": "https://github.com/MuhammadNadeemAUM/OilandGas/blob/main/QRScannerData/Oilngasproductionseparator.jpg",
  "booking_url": "https://www.hcpetroleum.hk/contact-us"
 },
```


Figure 6: JSON annotation file showing values of each object.

iv. Packages

Flutter packages are pre-built libraries or modules that developers can integrate into their Flutter applications to add specific features or functionality without having to write everything from scratch. These packages cover a wide range of functionalities and can significantly speed up the development process. The code snippet provided in Figure xxx shows all the packages used when implementing QR code scanner in the mobile and their details are provided below:

- *dart:async* is a library that provides classes and utilities for working with asynchronous programming. Asynchronous programming allows code to execute concurrently without blocking the main execution thread, enabling tasks such as network requests, file I/O, and long-running computations to be performed without freezing the user interface or application.

- *dart:convert* is a built-in library that provides support for encoding and decoding various data formats, such as JSON, UTF-8, and base64. It contains classes and functions for converting between different representations of data, enabling communication between different systems and components. One example is jsonEncode() function that converts Dart objects to JSON strings, while the jsonDecode() function converts JSON strings to Dart objects. These functions allow developers to serialize and deserialize data in JSON format, facilitating communication with web services, APIs, and other platforms.
- *dart:developer* library in Dart provides a set of tools for debugging and profiling Dart applications. It includes functions and classes for inspecting the runtime environment, collecting performance metrics, and monitoring code execution.
- *package:flutter/material.dart* library is a core part of the Flutter framework and provides widgets, classes, and utilities for building user interfaces (UIs) using the Material Design guidelines. It includes a wide range of pre-designed UI components, such as buttons, text fields, dialogs, and navigation bars, as well as layout widgets for arranging UI elements in rows, columns, grids, and more.
- *package:ar_location_view/ar_location_view.dart* library provides widgets and utilities for integrating augmented reality (AR) location-based views into Flutter applications. This package allows developers to create AR experiences that overlay digital information and graphics onto the real-world environment, enhancing location-based applications with interactive, location-aware content. The package includes the ARLocationView widget, which is the core component for displaying AR views within Flutter applications. This widget renders a live camera feed from the device's camera and overlays digital content, such as markers, labels, and annotations, onto the camera feed based on the user's location and orientation.
- *package:geolocator/geolocator.dart* provides utilities for accessing the device's location information, including latitude, longitude, altitude, speed, and heading. This package allows developers to incorporate location-based functionality into Flutter applications, such as mapping, navigation, and location-aware services. It also includes methods for retrieving the device's current location using various sources, such as GPS, Wi-Fi, and cellular networks. Developers can use the getCurrentPosition() function to fetch the device's current coordinates asynchronously, or the getLastKnownPosition() function to retrieve the most recent cached location data.
- *package:http/http.dart* provides utilities for making HTTP requests and interacting with web APIs from Flutter applications. This package allows developers to perform various HTTP operations, such as sending GET and POST requests, handling responses, and processing data formats like JSON, XML, and form-urlencoded data. The package includes functions for making HTTP requests to web servers and APIs. Developers can use the get() function to send a GET request, the post() function to send a POST request, and similar functions for other HTTP methods like PUT, DELETE, and PATCH.
- *package:mobile_scanner/mobile_scanner.dart*. The MobileScanner class contains a method scanDocument() that uses the device's camera (you may need to use plugins like camera or image_picker for this functionality) to capture an image of a document

and return it as a File object. *Mobile_scanner.dart* is the main Dart file where you define the functionality of the mobile_scanner package. It would include classes and functions related to mobile scanning functionality.

- *Package: url_launcher:* Provides functions for launching URLs in the device's default web browser.

```
import 'dart:async';
import 'dart:convert';
import 'dart:developer';
import 'dart:typed_data';
import 'package:arapp/gps_module/pdf_viewer.dart';
import 'package:arapp/gps_module/video_player.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:mobile_scanner/mobile_scanner.dart';
import 'package:url_launcher/url_launcher.dart';
import '../models/QRModel.dart';

class QRScanner extends StatefulWidget {
  const QRScanner({Key? key}) : super(key: key);

  @override
  State<QRScanner> createState() => _QRScannerState();
}

class _QRScannerState extends State<QRScanner> {
  MobileScannerController cameraController = MobileScannerController(
      detectionSpeed: DetectionSpeed.noDuplicates
  );
```

Figure 7: Packages used for QR scanning.

The following is the snippet of the code in Figure 8 used to retrieve annotations from a JSON file stored on a GitHub repository. We used the http package to make a GET request to fetch the JSON data and then parse it using the *dart:convert* library. This code fetches the JSON data from the specified URL, parses it into a list of *AnnotationModel* objects, and displays them in a *ListView*.

The code in Figure 9 essentially handles the processing of scanned QR codes by checking their type, finding matching data in a list, and generating buttons based on the available data associated with that QR code type. This Dart code is part of a function that processes a scanned QR code. This line checks if the scanned QR code's value (barcodeValue) is contained within the list of known QR types. If the QR code value is found in the list of known QR types, it assigns the value of the scanned QR code to qrDataString. The subsequent block of code iterates over a list of QR data objects (qrData) to find a match for the scanned QR code type. If a matching QR data object is found, it checks each optional property (video_link, pdf_link, website_url, map_url, booking_url) of the QR data object. If a property is not null (i.e., it has a value), it adds a button to buttons with the corresponding label and a callback to open the URL associated with that property. It breaks out of the loop after processing the first matching QR data object.

```
Future<List<AnnotationsModel>> getAnnotations() async {
  try {
    var url =
        "https://raw.githubusercontent";
    http.Response response = await http.get(Uri.parse(url));
    var data = jsonDecode(response.body.toString());
    if (response.statusCode == 200) {
      log(data.toString());
      List<AnnotationsModel> fetchedAnnotations = [];
      for (var item in data) {
        fetchedAnnotations.add(AnnotationsModel.fromJson(item));
      }
      return fetchedAnnotations;
    } else {
      // Handle error response
      throw Exception('Failed to load annotations');
    }
  } catch (e) {
    // Handle exceptions
    rethrow;
  }
}
```

Figure 8: Code snippet for fetching annotations from web repository.

```
Future<void> _showBarcodeAlert(BuildContext context, String? barcodeValue) async {
  String? qrDataString;
  List<Widget> buttons = [];

  // Check if the scanned barcode value is one of the available QR types
  if (qrTypes.contains(barcodeValue)) {
    qrDataString = barcodeValue;
    // Find matching QR type
    for (var qr in qrData) {
      if (qr.qr_type == barcodeValue) {
        // Add buttons if the type matches
        if (qr.video_link != null) {
          buttons.add(_buildLinkButton('Video', () => _launchUrl(Uri.parse(qr.video_link!))));
        }
        if (qr.pdf_link != null) {
          buttons.add(_buildLinkButton('PDF', () => _launchUrl(Uri.parse(qr.pdf_link!))));
        }
        if (qr.website_url != null) {
          buttons.add(_buildLinkButton('Website', () => _launchUrl(Uri.parse(qr.website_url!))));
        }
        if (qr.map_url != null) {
          buttons.add(_buildLinkButton('Map', () => _launchUrl(Uri.parse(qr.map_url!))));
        }
        if (qr.booking_url != null) {
          buttons.add(_buildLinkButton('Booking', () => _launchUrl(Uri.parse(qr.booking_url!))));
        }
        break;
      }
    }
  } else {
    qrDataString = 'Unknown';
  }
```

Figure 9: Code for handling for scanned QR

When the camera identifies a QR code, it typically triggers a notification banner on the real-world image providing a message that code was successfully scanned and printing the name of equipment identified. It also shows the AR contents option available as shown in Figure . User can select any of the option and view the contents which are stored online or alternately he can close the app and return to the previous page where he can access the page again. This reduced the footprint of the mobile application, but it need access to internet to operate. Once user has accessed the information, he can return to previous page.

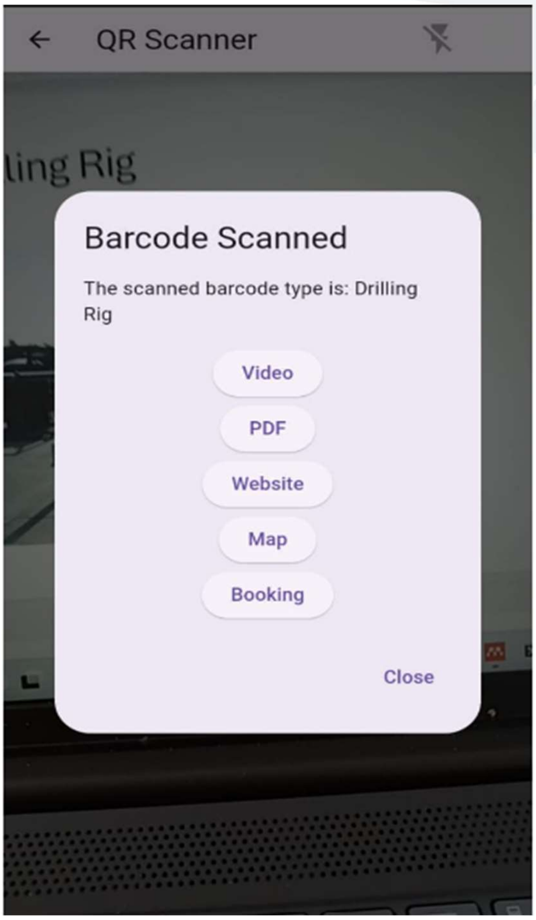

Figure 10: Screenshot showing result of scanned QR code.

f) GPS-based Navigation Implementation

GPS-based navigation offers valuable insights about a location, particularly in industrial settings. Upon approaching a specific building or area within the facility, augmented reality (AR) contents are activated, enriching the user experience. These AR elements encompass a variety of informative resources, including URLs, videos, maps, PDF texts, and booking options. This integration of AR technology serves multiple purposes, such as facilitating training sessions, enhancing operational efficiency, improving overall user experience, and enabling remote assistance. By seamlessly merging GPS navigation with AR functionalities, users can access pertinent information and resources tailored to their location, thereby optimizing their interactions within the industrial environment.

i. Location Selection

To gather location data for our GPS-based navigation project, we began by conducting a search on Google Maps using the keyword "KoC," which stands for Kuwait Oil Company. This search led us to discover various locations associated with the KoC facility, each marked on the map interface as shown in Figure 11. We captured the precise coordinates of these locations, we utilized a straightforward method: right-clicking on each desired location on the map, enabling us to copy the coordinates for subsequent use. This streamlined process proved effective in extracting the necessary location data essential for our navigation project.

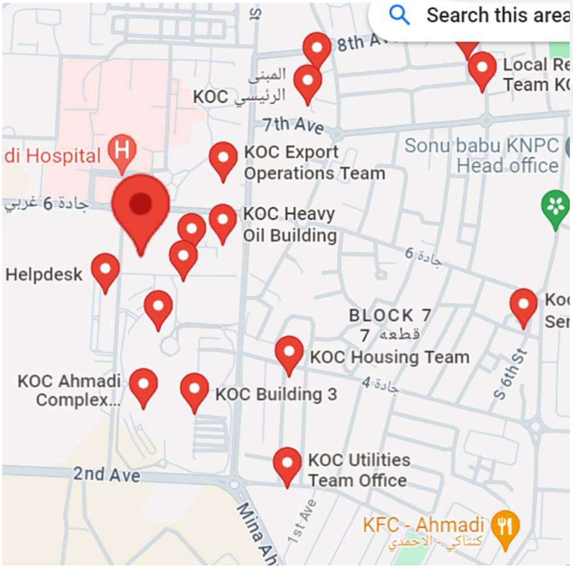

Figure 11: KoC location coordinates using Google maps.

The JSON file contains a structured list of locations within the KoC facility, each associated with a unique gps coordinate. This file serves as a comprehensive inventory detailing the location present in the facility and facilitates easy access to relevant information through QR code scanning. The name of location is provided as an array of elements separated by commas as shown in Figure 12. It has twenty-five entries and gps coordinates are obtained by right clicking the location pinned on the map.

```
"types": [
  "Ahmadi Complex",
  "Building 3",
  "IT Helpdesk",
  "Main Building",
  "Heavy Oil Building",
  "Headquarters",
  "Utilities Team",
  "Magwa Office",
  "Ahmadi Complex 2",
  "Hospital",
  "Ahmadi Complex 5",
  "Security Center",
  "Export Team",
  "Community Services",
  "Housing Team",
  "Local Relations",
  "Ahmadi Complex Gate 3",
  "Public Relations",
  "Recruitment Facility",
  "Scrap Yard 3",
  "Data Center",
  "Drilling Group",
  "Referral Abroad",
  "Office Complex",
  "option 25"
```

Figure 12: JSON files listing all location names

ii. AR Contents

Upon reaching close to location, the application triggers the display of AR content overlaid onto the user's physical environment which is the name of the file fetched from the JSON file. We include five different multimedia elements for GPS-navigation component of the mobile application. The AR values for each location object are provided in the snippet of JSON file shown in Figure 13


```json
{
  "AnnotationType": "Ahmadi Complex",
  "latitude": 29.088603256528724,
  "longitude": 48.056185672211626,
  "video_link": "https://www.youtube.com/watch?v=4wztchlc5EQ",
  "pdf_link": "https://github.com/salmanrazzaq1993/koc/blob/main/KOCAhmadiComplex.pdf",
  "website_url": "https://www.kockw.com/sites/EN/Pages/Default.aspx",
  "map_url": "https://github.com/salmanrazzaq1993/koc/blob/main/KOC%20Ahmadi%20Complex.jpg",
  "booking_url": "https://mrr.aum.edu.kw/booked/Web/?"
},
{
  "AnnotationType": "Building 3",
  "latitude": 29.084768566196384,
  "longitude": 48.05758050621762,
  "video_link": "https://www.youtube.com/watch?v=N-fTXAFKues&t=22s",
  "pdf_link": "https://github.com/MuhammadNadeemAUM/ARapp/blob/main/Pdfdocuments/Sports_Centre.pdf",
  "website_url": "https://www.kockw.com/sites/EN/Pages/Default.aspx",
  "map_url": "https://github.com/salmanrazzaq1993/koc/blob/main/KOCBuilding3.jpg",
  "booking_url": "https://www.aum.edu.kw/english/campus-life/athletics-sports-center/outdoor-facilities"
},
{
  "AnnotationType": "IT Helpdesk",
  "latitude": 29.087931800997133,
  "longitude": 48.055087853737525,
  "video_link": "https://www.youtube.com/watch?v=dUC3r6UOPb0",
  "pdf_link": "https://github.com/salmanrazzaq1993/koc/blob/main/KOCITHelpdesk.pdf",
  "website_url": "https://www.kockw.com/sites/EN/Pages/Default.aspx",
  "map_url": "https://github.com/salmanrazzaq1993/koc/blob/main/KOCITHelpdesk.jpg",
  "booking_url": "https://www.aum.edu.kw/english/contact-us"
}
```


Figure 13: JSON annotation file showing values of each location.

iii. Packages

The snippet of code showing different packages and libraries used when implementing the gps-navigation component of the application as shown in Figure 14.

dart:convert is a built-in library that provides support for encoding and decoding various data formats, such as JSON, UTF-8, and base64. It contains classes and functions for converting between different representations of data, enabling communication between different systems and components. One example is jsonEncode() function that converts Dart objects to JSON strings, while the jsonDecode() function converts JSON strings to Dart objects. These functions allow developers to serialize and deserialize data in JSON format, facilitating communication with web services, APIs, and other platforms.

The *dart:developer* library provides a set of tools for debugging and profiling Dart applications. It includes functions and classes for inspecting the runtime environment, collecting performance metrics, and monitoring code execution.

The *package:flutter/material.dart* provides widgets, classes, and utilities for building user interfaces (UIs) using the Material Design guidelines. It includes a wide range of pre-designed UI components, such as buttons, text fields, dialogs, and navigation bars, as well as layout widgets for arranging UI elements in rows, columns, grids, and more.

The *package:ar_location_view/ar_location_view.dart* provides widgets and utilities for integrating augmented reality (AR) location-based views into Flutter applications. This package allows developers to create AR experiences that overlay digital information and graphics onto the real-world environment, enhancing location-based applications with interactive, location-aware content. The package includes the ARLocationView widget, which is the core component for displaying AR views within Flutter applications. This widget renders a live camera feed from the device's camera and overlays digital content, such as markers, labels, and annotations, onto the camera feed based on the user's location and orientation.

The *package:geolocator/geolocator.dart* provides utilities for accessing the device's location information, including latitude, longitude, altitude, speed, and heading. This package allows developers to incorporate location-based functionality into Flutter applications, such as mapping, navigation, and location-aware services. It also includes methods for retrieving the device's current location using various sources, such as GPS, Wi-Fi, and cellular networks. Developers can use the getCurrentPosition() function to fetch the device's current coordinates asynchronously, or the getLastKnownPosition() function to retrieve the most recent cached location data.

The *package:http/http.dart* provides utilities for making HTTP requests and interacting with web APIs from Flutter applications. This package allows developers to perform various HTTP operations, such as sending GET and POST requests, handling responses, and processing data formats like JSON, XML, and form-urlencoded data. The package includes functions for making HTTP requests to web servers and APIs. Developers can use the get() function to send a GET request, the post() function to send a POST request, and similar functions for other HTTP methods like PUT, DELETE, and PATCH. *AnnotationsModel.dart* is used to define models or data structures representing annotations or markers used in mapping or location-based applications.

```
import 'dart:convert';
import 'dart:developer';
import 'package:flutter/material.dart';
import 'package:ar_location_view/ar_location_view.dart';
import 'package:geolocator/geolocator.dart';
import 'package:http/http.dart' as http;
import '../models/AnnotationsModel.dart';
import 'annotation_view.dart';

class GPSAnnotationScanner extends StatefulWidget {
  const GPSAnnotationScanner({super.key});

  @override
  State<GPSAnnotationScanner> createState() => _GPSAnnotationScannerState();
}
```

Figure 14: Snippet of code showing Packages used for gps-navigation

iv. Fetching Annotations

The following is the snippet of the code used to retrieve annotations from a JSON file stored on a GitHub repository. We used the http package to make a GET request to fetch the JSON data and then parse it using the *dart:convert* library. This code fetches the JSON data from the specified URL, parses it into a list of *AnnotationModel* objects, and displays them in a *ListView*.

```
class _GPSAnnotationScannerState extends State<GPSAnnotationScanner> {

  List<Annotation> annotations = [];
  List<String> annotationTypes = [];

  Future<void> fetchAnnotationTypes() async {
    try {
      final response = await http.get(Uri.parse('https://raw.githubusercontent
      if (response.statusCode == 200) {
        final jsonBody = jsonDecode(response.body);
        log(jsonBody.toString());
        final typesFromJson = List<String>.from(jsonBody['types']);
        setState(() {
          annotationTypes = typesFromJson;
        });
      } else {
        throw Exception('Failed to fetch annotation types');
      }
    } catch (e) {
      throw Exception('Error: $e');
    }
  }
```

Figure 15: Code for fetching annotations.

v. Range Setting

The range of GPS navigation is set by using the geolocator package which accesses location services and calculate the distance between two geographical coordinates. It fetch the current device location and calculates the distance between the current location and a target location. *_getCurrentLocation()* method retrieves the current device location using the getCurrentPosition method from Geolocator. The distance between the current location and the target location is calculated using the *distanceBetween* method from Geolocator. We compare the calculated distance with a predefined range threshold *(_rangeThreshold)* to determine if the current location is within range. The range of gps service in this application is set to 3000 meters as shown in

```
@override
Widget build(BuildContext context) {
  return SafeArea(
    child: Scaffold(
      body: ArLocationWidget(
        annotations: annotations,
        showDebugInfoSensor: false,
        maxVisibleDistance: 3000,
        annotationViewBuilder: (context, annotation) {
          return AnnotationView(
            key: ValueKey(annotation.uid),
            annotation: annotation as Annotation, // Cast annotation to Annotation type
          );
        },
        onLocationChange: (Position newPosition) {
          WidgetsBinding.instance.addPostFrameCallback((_) {
            setState(() {
              // Update annotations based on the new position
              annotations.forEach((annotation) {
                double distanceInMeters = Geolocator.distanceBetween(
                  newPosition.latitude,
                  newPosition.longitude,
                  annotation.position.latitude,
                  annotation.position.longitude,
                );

                // For example, let's update the visibility of annotations based on distance
                if (distanceInMeters <= 3000) {
                  annotation.isVisible = true; // Show annotation if within 3000 meters
                } else {
                  annotation.isVisible = false; // Hide annotation if further away
                }
              });
            });
          });
```

Figure 16: Code for setting gps-navigation range.

g) *Integration of Mobile Application Components*

In the last phase of implementation, we integrated GPS navigation and QR code scanning functionalities into a single application. We created separate screens for each feature and then navigate between them using Flutter's navigation system. We used the geolocator package for GPS navigation and the *qr_code_scanner* package for QR code scanning. We then implemented the GPS navigation functionality on one screen. This screen will display the user's current location and provide navigation features such as route calculation, distance calculation, and direction display. Similarly, we implemented the QR code scanning functionality on another screen. This screen uses the device's camera to scan QR codes and extract information from them. We used Flutter's navigation system (e.g., *Navigator.push*) to navigate between the GPS navigation screen and the QR code scanning screen based on user actions or application logic. Both screens are implemented separately and do not communicate with each other. The required dependencies (*geolocator and qr_code_scanner*) are added to *pubspec.yaml* file. The Dart code in Figure 17 defines a *HomePage* widget, which serves as the main screen of the application. class HomePage extends *StatefulWidget*. represents the HomePage widget, which is a *StatefulWidget*. It overrides the createState method to create an instance of *_HomePageStat*e, which represents the mutable state of the HomePage widget. _HomePageState class extends the State class and represents the mutable state for the HomePage widget and overrides the build method to define the UI of the HomePage. build method returns a SafeArea widget, which ensures that its child's edges are inset by sufficient padding to avoid system UI elements. Inside SafeArea, there's a Scaffold widget, which provides a basic layout structure for the home screen. The body of the Scaffold contains a Center widget, which centers its child vertically and horizontally. Inside the Center widget, there's a Row widget, which arranges its children in a horizontal row. The mainAxisAlignment property is set to MainAxisAlignment.spaceBetween, which evenly spaces the children along the main axis (horizontally) with extra space placed between them. Within the Row, there are two ElevatedButton widgets wrapped in SizedBox widgets to specify their dimensions. One button is for QR code scanning, and the other is for GPS navigation.

```dart
class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    return SafeArea(
        child: Scaffold(
          body: Center(
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                SizedBox(width: 20,),
                SizedBox(
                height: 150,
                  child: ElevatedButton(
                    child: Text('QR Scanner'),
                    onPressed: (){
                      Navigator.push(context, MaterialPageRoute(builder: (context) => QRScanner()));
                    },
                  ),
                ),
                SizedBox(
                  height: 150,
                  child: ElevatedButton(
                    child: const Text('GPS Guide'),
                    onPressed: (){
                      Navigator.push(context, MaterialPageRoute(builder: (context) => const GPSAnnotationScanner()));
                    },
                  ),
                ),
                SizedBox(width: 20,),
              ],
            ),
          ),
        )
    );
  }
}
```

Figure 17: Dart code for homepage widget

The code of *main.dart* is shown in Figure 18 which is the entry point of the application. The main function is executed when the application is started. Inside this function, we call the runApp function and pass an instance of MyApp. *runApp(const MyApp())* function takes a widget as an argument and starts the Flutter application with that widget as the root of the widget tree. In this case, it starts the application with an instance of MyApp. *class MyApp extends StatefulWidget* represents the root widget of the application. It extends *StatefulWidget*, indicating that it has mutable state. It overrides the createState method to create an instance of _MyAppState, which represents the mutable state of the MyApp widget. _MyAppState class extends the State class and represents the mutable state for the MyApp widget. Currently, it doesn't have any state or logic defined in its initState method. build method returns a MaterialApp widget, which is the root of the widget tree for the Flutter application. It specifies the HomePage widget as the home screen of the application and sets debugShowCheckedModeBanner to false to hide the debug banner in the top-right corner of the screen.

```
import 'package:KoCAR/home_page.dart';
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatefulWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {

  @override
  void initState() {
    super.initState();

  }

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: HomePage(),
    );
  }

}

```

Figure 18. main.dart file

The Figure 19 shows the screenshot of gps-based navigation page where a circle and red dots are shown which represent the range and twenty-five locations, respectively.

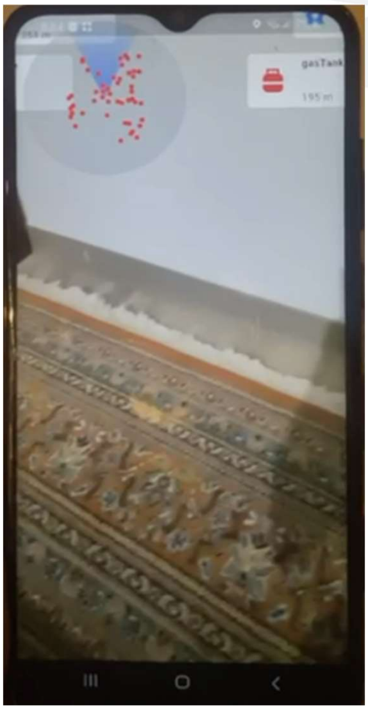

Figure 19. GPS-based navigation screen

Upon reaching closer to the location, its name is displayed and upon clicking, it displays the name of the location as well the AR field of the object location. The screenshot in Figure 20 shows that the user is nearing the admin building and also indicate that information in the form of video and document are available.

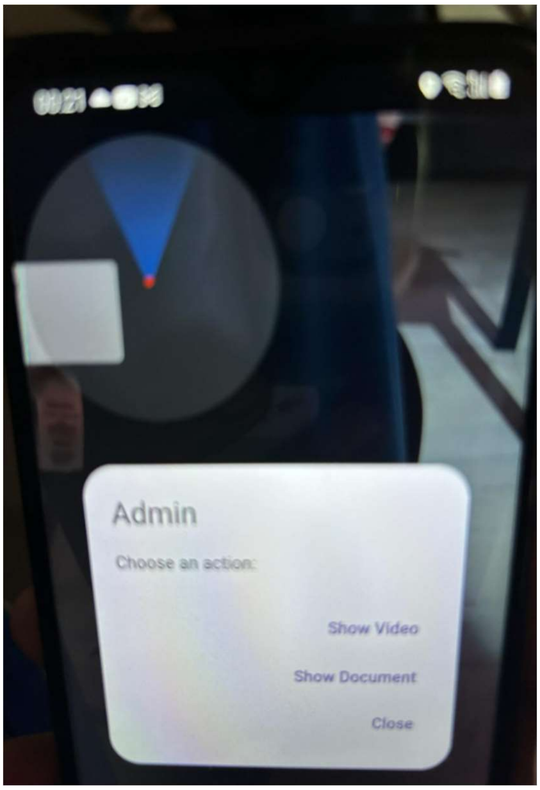

Figure 20. Screenshot when a location is identified.

2. Design Analysis and Feedback

In this section, we will present an evaluation of the effectiveness and user experience of OGAR mobile application. Systematic evaluation of the design, functionality, and user experience of the application and feedback can be incorporated into design process, creating an engaging, intuitive, and impactful for AR application users.

a) *Mobile Application Evaluation*

The primary focus of the mobile application evaluation centers on assessing the user experience facilitated by the interaction process utilizing the augmented reality feature. This encompasses evaluating specific performance criteria that may impact the augmented reality functionality. Evaluation allows for the assessment of user satisfaction levels with the application. By gathering feedback from users through surveys, interviews, or analytics, developers can gain insights into user preferences, pain points, and overall satisfaction with the app's features and functionality. This feedback is invaluable for making necessary adjustments and improvements to enhance user experience and satisfaction. It also serves as a means of continuously improving the mobile application over time. By identifying areas for enhancement based on user feedback, usage data, and market trends, developers can iteratively refine the application to meet evolving user needs and expectations. This iterative process of improvement ensures that the app remains relevant, competitive, and aligned with user preferences. In summary, the evaluation of a mobile application is essential for ensuring user satisfaction, driving continuous improvement, prioritizing features, enhancing user engagement, and fostering innovation and differentiation. By systematically evaluating the application and incorporating feedback into the development process, developers can create a mobile application that meets user needs, exceeds expectations, and achieves long-term success.

i. Testing Procedure

To evaluate the effectiveness and usability of the prototype AR application, a testing phase was conducted involving a group of Kuwait Oil Company (KoC) employees. The experiments took place in a real-world setting, with participants from the local audience on-site at KoC in AlAhmadi. Participants were invited to take part in the testing session, and their consent was obtained before their involvement. Recruitment was made on the recommendation of the administration of KoC. Upon agreeing to participate, the participants were instructed to download the prototype AR application onto their mobile devices. Detailed instructions may have been provided to ensure successful installation from an app store or through a direct download link. Each participant conducted individual tests using their smartphone. Once the application was installed, participants were asked to test it in a real-world environment relevant to their work at KoC. This involved visiting different locations, pieces of machinery, and instruments within the company's facilities where AR content was intended to be utilized. Participants were given specific tasks or scenarios to complete using the AR application. These tasks may have included locating specific equipment or facilities, accessing digital overlays of relevant information, or interacting with AR-enhanced features designed to assist with job-related activities. Participants received instructions about the test objectives and were instructed to adhere to the provided questionnaire supervised by the facilitator. The facilitator documented notes and recorded any issues or suggestions explicitly mentioned by the participants. Following the testing session, participants were asked to complete a questionnaire or survey to provide feedback on their experience with the AR application. The questionnaire may have included questions about usability, effectiveness, user satisfaction, perceived usefulness, and suggestions for improvement.

ii. Questionnaire Design

To evaluate the effectiveness and acceptance of the application among participants, we developed a comprehensive questionnaire tailored to assess various aspects of their experience. The questionnaire was designed with the participant characterization in mind, and it included demographic questions to gather information about the participants, such as their age, gender, job role, and level of experience with similar technologies. This demographic data provided context for interpreting their responses and understanding any potential correlations between user characteristics and application perception. Likert scale questions were utilized to quantitatively measure participants' attitudes and perceptions towards different aspects of the application. Participants were asked to rate their agreement or disagreement with statements using a predefined scale, typically ranging from strongly disagree to strongly agree. These questions assessed factors such as perceived usefulness, ease of use, satisfaction, and intention to use the application in the future. In addition to Likert scale questions, the questionnaire included open-ended questions to elicit qualitative feedback and insights from participants. These questions encouraged participants to provide detailed comments, suggestions, and critiques about their experience with the application. Open-ended responses allowed for a deeper understanding of participants' thoughts, preferences, and areas for improvement. Participants were assured of the anonymity and confidentiality of their responses to encourage honest and unbiased feedback. This confidentiality measure aimed to create a safe environment for participants to express their opinions freely without fear of repercussion.

The questionnaire was structured based on the Technology Acceptance Model (TAM) [19], a widely used theoretical framework for understanding users' acceptance and adoption of technology. TAM posits that perceived usefulness and perceived ease of use are key determinants of users' intention to use technology. Therefore, the questionnaire was designed to capture participants' perceptions of these factors and their impact on their acceptance of the application.

Table 3. Survey questions.

Demographic	
Q1.	Gender
Q2.	Company Name
Q3.	Job Title
Q4.	Years of Experience in the Oil and Gas Industry:
Q5.	Education
Usefulness	
Q6.	Using OGAR improves my job performance.
Q7.	The OGAR application addresses my job-related needs.
Q8.	Using OGAR saves me time.
Q9.	Using OGAR reduces my time on unproductive activities
Q10.	Using OGAR improves the quality of the work I do.

Q11	Using OGAR increases my productivity.
Q12	Using OGAR makes it easier to do my job.
Ease of Use	
Q13.	It is easy to use OGAR.
Q14.	Interacting with the OGAR application is often frustrating.
Q15	I needed help when using OGAR.
Q16.	Interacting with the OGAR application requires a lot of my mental effort.
Q17.	The OGAR application often behaves in unexpected ways.
Q18.	I find it cumbersome or difficult, to use the OGAR application.
Q19.	My interaction with the OGAR application is easy for me to understand.
Q20.	The OGAR application provides helpful guidance in performing tasks.
ATU	
Q21.	I believe it is a good idea to use the OGAR application.
Q22.	I think it is valuable to use the OGAR application.
BIU	
Q23.	I plan to use the OGAR application in the future.
Q24.	I'd love to use multimedia materials in my class.
Q25.	I will recommend using the OGAR application in the future.
Relevance	
Q26.	In my job, the usage of the OGAR is relevant.
Q27	In my job, the usage of the OGAR is important.
Open-ended	
Q28.	W hat kind of information would you like to see on the application?
Q29.	What will you suggest to improve this application?

iii. Results and Discussion

In this section, we delve into the outcomes of the survey conducted, which involved the participation of 10 individuals who tested the application and subsequently provided feedback through the survey. Figure 21 illustrates the gender distribution among the participants, revealing that 80% of the participants were male, while the remaining 20% were female. The surveyed participants encompassed a diverse range of individuals, including employees of KoC, visitors, and contract employees. Their educational backgrounds varied, with the majority holding bachelor's degrees. Additionally, the participant pool comprised individuals with diplomas, certificates, and expertise in electrician and mechanic roles, as well as a participant with a master's degree. The participants' experience ranged from individuals with no prior experience, such as AUM students visiting KoC, to those with a maximum of 10 years of experience. This diversity in experience levels contributed to a comprehensive assessment of the application's usability and effectiveness across different user demographics and skill sets.

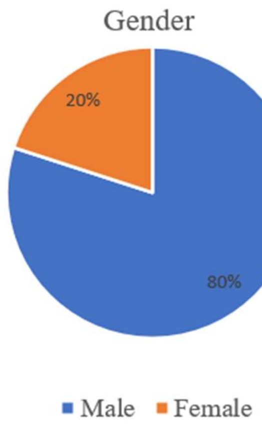

Figure 21: Gender breakdown of participants.

Figure 22 displays the mean scores obtained from participant responses regarding the perceived usefulness of the application. Overall, the majority of participants expressed a positive opinion, as indicated by mean scores exceeding 3 for most questions. However, it's noteworthy that question Q10, which pertains to the quality of work, received a relatively lower score. This discrepancy in scores could be attributed to various factors. One potential explanation for the lower score on Q10 is the sensitive nature of questioning one's work quality, which might evoke concerns about job security or performance evaluation. Participants may have felt hesitant to provide candid feedback, opting instead to remain neutral to avoid potential repercussions. It's essential to recognize the potential influence of social dynamics and organizational culture on participant responses. In workplace settings, individuals may be inclined to withhold negative feedback to preserve harmony or avoid confrontation. Additionally, concerns about anonymity and confidentiality may contribute to participants' reluctance to provide candid assessments of work quality. Addressing concerns related to job security and fostering open communication channels can help cultivate a culture of constructive feedback and continuous improvement within the organization.

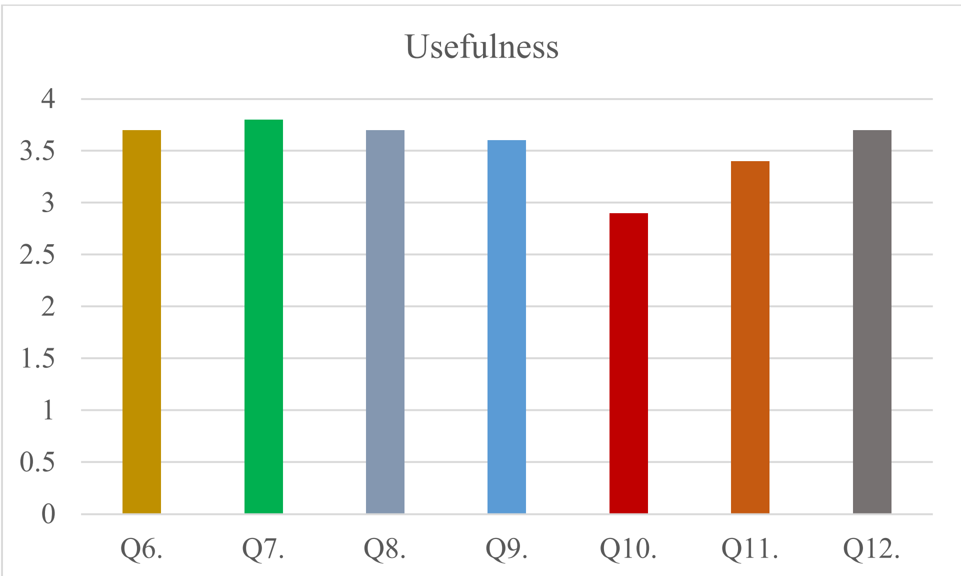

Figure 22. Responses to the survey questions related to usefulness.

The mean score of likert score for questions related to ease of use is provided in Figure 23 The average score for each question is above 3 indicating that the participant found it easy to use the OGAR mobile application. The ease of use of a mobile application significantly impacts its evaluation and overall success. When users find an app intuitive and easy to navigate, they are more likely to have a positive experience and perceive the application favorably. Conversely, if an app is difficult to use or requires extensive effort to accomplish tasks, users may become frustrated and abandon the application altogether. Applications that are easy to use tend to have higher user engagement rates. When users can quickly access desired features and functions, they are more likely to engage with the app frequently and for longer durations. Ease of use plays a crucial role in determining the adoption rate of a mobile application. Users are more inclined to download and continue using an app that offers a seamless and hassle-free experience from the outset.

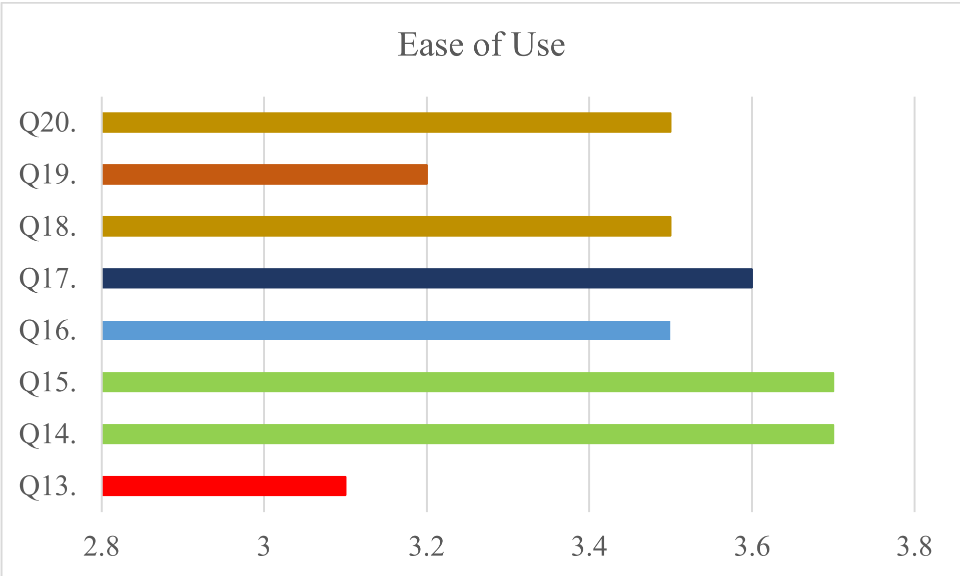

Figure 23. Mean score of responses to the questions related to ease of use

ATU (Actual Use), BIU (Behavioral Intention to Use), and relevance are key constructs in the Technology Acceptance Model (TAM), a widely used framework for understanding users' acceptance and adoption of new technologies. The score for questions related to these constructs are provided in Figure 24. Q21 and Q22 are related to ATU with mean score for Q21 much higher than Q22 indicating that participant think it can be used in industry and low score of Q22 is due to the fact that this application might not be successful for some of the participants due to the nature of the job as some of the participant find it useful in actual setting than others. For example, a maintenance engineer might find it more useful in an actual setting than computer operators.

Q23-Q25 represent the mean scores of participants' responses to questions regarding their behavioral intention to use the application. These questions are designed to assess participants' willingness or inclination to use the application in the future. The mean scores for Q23-Q25 are consistently higher than 3.5 across all cases, indicating a positive attitude towards using the application. A mean score higher than 3.5 suggests that participants generally expressed a favorable intention to use the application in various scenarios or contexts. This indicates a strong inclination among participants to engage with the application and incorporate it into their routine or workflow.

Q26-Q27 represents the mean scores of participants' responses to questions concerning the relevance of the application for the oil and gas industry. These questions aim to gauge participants' perceptions of how well the application aligns with the specific needs and requirements of the oil and gas sector. The mean scores for Q26-Q27 are 3.8 and 3.9, respectively, indicating a high level of perceived relevance of the application to the industry. These scores suggest that participants generally view the application as highly relevant and well-suited to address the unique challenges and demands of the oil and gas industry. This indicates that participants perceive the application as valuable and potentially impactful in enhancing efficiency, productivity, or safety within the industry context.

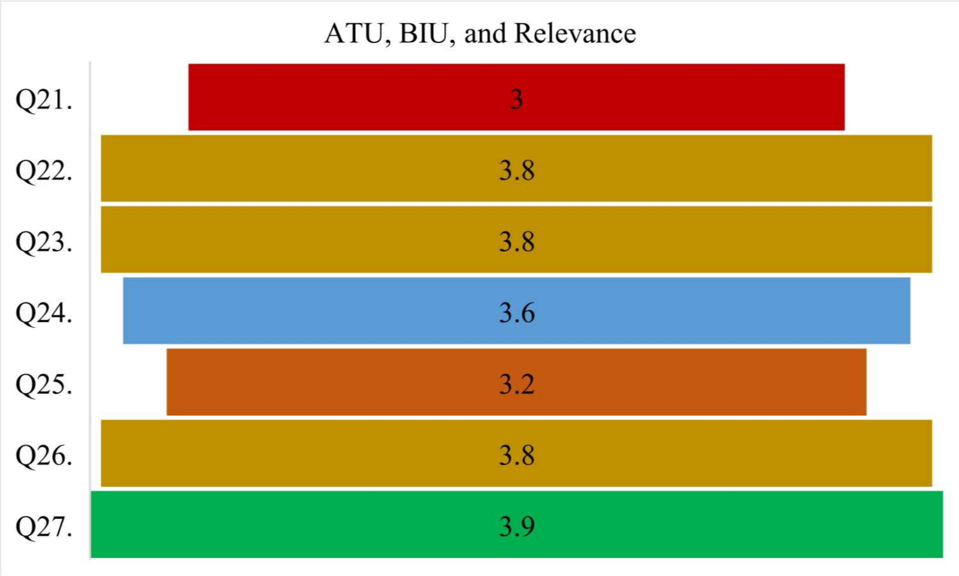

Figure 24: Mean score of responses to the questions related ATU, BIU, and relevance

Responses to open-ended questions provide valuable insights into users' preferences and suggestions for improving the mobile application for use in the oil and gas industry. Here is an elaboration on the responses provided:

- Users express a need for comprehensive instructions within the application. This indicates a desire for clear guidance on how to navigate the app's features and functionalities effectively. Detailed instructions can help users better understand how to utilize the application to its full potential and enhance their overall user experience.
- Users emphasize the importance of having detailed information available within the application. This suggests a preference for comprehensive data or content related to various aspects of the oil and gas industry. Providing detailed insights or specifications can help users make informed decisions and perform tasks more efficiently within the app.
- Users suggest incorporating video content within the application to facilitate better understanding of complex concepts or procedures. Videos can serve as visual aids to complement textual information and provide users with a more engaging and immersive learning experience. Including instructional or informational videos can enhance user comprehension and retention of key information.
- Users emphasize the importance of having a well-organized user interface (UI) within the application. This indicates a preference for a clean and intuitive layout that facilitates easy navigation and access to relevant features or content. An organized UI

can streamline the user experience and make it easier for users to find what they need within the app.

- Users express a need for tutorials or guided walkthroughs within the application. Tutorials can help users familiarize themselves with the app's features and functionalities, allowing them to learn how to use the application effectively at their own pace. Providing interactive tutorials or step-by-step guides can help users overcome any learning curves associated with the app and improve their overall user experience.
- The requirement for the application to be accessible without an internet connection underscores the importance of offline functionality and accessibility. In oil and gas workers often operate in remote locations or areas with limited connectivity, so having access to critical information offline is essential. The application must ensure that users can retrieve data, access instructions, or perform tasks even when internet connectivity is unavailable. Furthermore, internet connectivity can be unreliable in certain environments, such as offshore platforms or remote drilling sites. By enabling offline access, the application will enhance operational reliability by ensuring that users can continue to use essential features and functionalities without interruptions caused by network issues.

3. Design Optimization and Improvements

a) *Desing Optimization*

Optimizing the design of a mobile AR (augmented reality) application involves improving its functionality, usability, performance, and overall user experience. We plan to enrich as well as simplify the user interface (UI) to ensure it is intuitive and easy to navigate. Minimize clutter, use clear visual cues, and prioritize essential functions to enhance usability.

In oil and gas workers often operate in remote locations or areas with limited connectivity, so having access to critical information offline is essential. The application must ensure that users can retrieve data, access instructions, or perform tasks even when internet connectivity is unavailable. Furthermore, internet connectivity can be unreliable in certain environments, such as offshore platforms or remote drilling sites. By enabling offline access, the application will enhance operational reliability by ensuring that users can continue to use essential features and functionalities without interruptions caused by network issues.

When making AR contents available online, it will increase the app footprint as well as the latency. We intend to optimize the delivery of AR content to minimize latency and improve responsiveness. Utilize techniques such as preloading assets, caching data, and implementing efficient data streaming to deliver a smooth AR experience.

b) *Improvements*

We intend to improve the application by providing offline access to data thus making it accessible without an internet connection. In oil and gas workers often operate in remote locations or areas with limited connectivity, so having access to critical information offline is essential. The application must ensure that users can retrieve data, access instructions, or perform tasks even when internet connectivity is unavailable. Furthermore, internet connectivity can be unreliable in certain environments, such as offshore platforms or remote drilling sites. By enabling offline access, the application will enhance operational reliability by ensuring that users can continue to use essential features and functionalities without interruptions caused by network issues.

Based on user recommendation, we intend plan to incorporate video content within the application to facilitate better understanding of complex concepts or procedures. Videos can serve as visual aids to complement textual information and provide users with a more engaging and immersive learning experience. Including instructional or informational videos can enhance user comprehension and retention of key information.

In the future, we intend to incorporate marker-less AR for equipment and machinery as adding a QR code is a cumbersome process. We plan to use ML models for real-time object detection and recognition within the AR environment. This will allow users to interact with physical objects or scenes captured by the device's camera. For example, ML algorithms can identify specific objects or products and overlay relevant information or actions on top of them.

IV. General Discussion

1. Final Cost Analysis and Discussion

OGAR is a mobile-based application and does not require additional hardware beyond what is already available on the user's device such as smartphone [20]. It does not require any peripheral devices, any development or integration of hardware. Furthermore, this application was developed by the student, therefore, there was no need to hire developers, designers, and other technical professionals thus no expenditure on salaries, contractor fees, or outsourcing expenses. All the development frameworks, tools, and platforms used were available free of costs. All the data collected can was hosted on publically available repositories such as GitHub that means costs associated with setting up and maintaining servers, databases, APIs, and cloud services for data storage, processing, and communication between the app and external systems was saved. Only expenses which might occur int the current form is fee associated with publishing the app on app stores like Apple App Store and Google Play Store, as well as any promotional or marketing expenses to increase visibility and downloads. Additionally, there can recurring expenses such as for maintaining and supporting the application post-launch, including software updates, bug fixes, security patches, and customer support.

2. Commercializing the Project and Relevance to Region

Kuwait, being an oil-producing nation with numerous refineries and related facilities employing a sizable workforce, mandates safety compliance, and efficient operation. However, maintenance process often requires related information on the field and if this information is missing it can cause considerable delays. Furthermore, these tasks are carried out by the contract labour who are not aware of the locations and find it hard to navigate. It facilitates faster task execution with consistent precision, thereby enhancing efficacy. Additionally, automating repetitive and labor-intensive tasks allows organizations to reduce labor costs, minimize errors, and optimize resource utilization, leading to overall cost savings. Moreover, the system seamlessly integrates with existing infrastructure or operates as a standalone solution, featuring easy installation and a user-friendly interface. Capable of generating alerts, it ensures safety compliance and reduces accident rates. Also, Kuwait's oil companies have embarked on a digital transformation journey to digitize their operations, aligning with Kuwait Petroleum Corporation's (KPC) future vision. This product harmonizes perfectly with the sector's digital future, bolstering safety, security, and risk management.

Our business model emphasizes innovation to meet client requirements competitively and sustainably [21]. Employing a dual strategy, we aim to enhance profitability. Continuous product improvement through feature enrichment enables us to adjust pricing, while sourcing inventory cost-effectively and leveraging affordable labor helps reduce development costs. Providing comprehensive customer support and service, including system installation and regular maintenance, is pivotal. This ensures uninterrupted system operation, enhancing customer satisfaction, trust, and market reputation. Establishing a network within the industry and forging partnerships with relevant organizations are crucial for product promotion and business expansion. Active engagement and collaboration with stakeholders, including participation in industry events, drive adoption of our OGAR application. We are committed to perpetual product enhancement by regularly updating it with new features, security upgrades, and performance optimizations, informed by user feedback and technological advancements.

3. Related Engineering Standards

Engineering standards are established guidelines, specifications, and criteria that define the minimum acceptable levels of quality, safety, reliability, and performance for engineering projects, products, processes, and systems.

- ### ISO/IEC 25010 - Software Quality Models

This standard defines software quality models and provides a framework for evaluating the quality characteristics of software products [22]. It is part of the ISO/IEC 25000 series, which encompasses software product quality standards and guidelines. It outlines eight primary quality characteristics, often referred to as "quality attributes" or "quality factors," which are used to assess the overall quality of a software product. These quality characteristics are further subdivided into sub characteristics, which provide more detailed criteria for evaluation.

- ### IEEE 2048.101- Augmented Reality on Mobile Devices

This standard outlines the overall technical structure, constituent elements, integration methods, and core business operations of augmented reality systems implemented on mobile devices. It sets forth the technical prerequisites, encompassing functional specifications, performance criteria, and associated testing methodologies. The intended scope of applications and services covers, among other things, the creation, enhancement, and administration of augmented reality and mixed reality functionalities within mobile applications.

- ### IEEE 829 - Standard for Software and System Test Documentation

This IEEE standard delineates the structure of a series of documents intended for utilization across eight distinct phases of software testing and system testing. Each stage has the potential to generate its unique document type, thereby facilitating comprehensive documentation throughout the testing process.

- IEEE 1471 – Description of Complex Software Systems

This standard provides recommendations for architectural description and documentation of complex software systems, offering guidelines for designing the architecture of GPS-based mobile applications to ensure scalability, maintainability, and interoperability.

- IEEE 2600 - encoding and decoding requirements for QR codes

This standard specifies the encoding and decoding requirements for QR codes, including data formats, error correction methods, and symbology specifications. It ensures consistency and interoperability among QR code scanners and generators, allowing seamless exchange of information across different devices and platforms. Adhering to IEEE 2600 ensures that QR code scanners and applications can accurately read and interpret QR codes encoded with various types of data, such as URLs, text, contact information, and multimedia content. Additionally, it facilitates the implementation of error detection and correction mechanisms to improve the reliability of QR code scanning, especially in challenging environments with poor lighting or damaged codes.

- IEEE 1857 – Framework for Augmented Reality

This standard specifies a framework for augmented reality applications, encompassing various aspects such as data formats, metadata, and communication protocols. It aims to ensure interoperability and compatibility among AR systems and components.

- IEEE 2015 – AR Performance and Evaluation

This standard defines augmented reality (AR) performance metrics and evaluation methods, providing guidelines for assessing the effectiveness and quality of AR applications. It covers aspects such as tracking accuracy, rendering quality, and user experience.

4. Engineering Considerations:

a) Relevant Ethical aspects and Codes of Ethics [PI-4.a]

Below are some key aspects from the IEEE Code of Ethics [23] that apply to the design of mobile applications.

i. Privacy and Security

IEEE Code - I.1: "To hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment."

Relevance: Engineers and technologists shall prioritize the safety, health, and welfare of the public in the performance of their professional duties. When developing QR scanner-based AR applications, developers should ensure that the technology enhances user experiences, respects privacy, and contributes positively to society.

ii. Professional Development

IEEE Code - I.6: "To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations."

Relevance: Engineers and technologists shall support the professional development of their colleagues and peers and foster a culture of lifelong learning and continuous improvement within their organizations and communities. Developers of QR scanner-based AR applications should collaborate with multidisciplinary teams, share knowledge and best practices, and engage in ongoing education and training to enhance the quality and effectiveness of their work.

iii. Public Awareness

IEEE Code - I.2: "To improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems."

Relevance: Engineers and technologists shall seek to promote the responsible and equitable use of technology to address societal challenges and improve the quality of life for all people. When deploying navigation-based AR for industry visits, professionals should consider the social and ethical implications of the technology, including its impact on job displacement, economic inequality, and access to information and opportunities.

iv. Honesty

IEEE Code - I.2: "To uphold the highest standards of integrity, responsible behavior, and ethical conduct in professional activities."

Relevance: Researchers should present survey findings accurately and transparently, avoiding exaggeration or misrepresentation of results. This includes accurately reporting response rates, data analysis methods, and limitations of the survey.

b) Impact of proposed solution [PI-4.b]

Discuss the impact (positive or negative) of your design on the global, economic, environmental, and societal contexts.

i. Societal Impacts

- *Improved User Experience:* Surveys can lead to enhancements in mobile app usability, accessibility, and functionality, ultimately improving the overall user experience and satisfaction.
- *Assist workers and visitors:* AR technology can facilitate immersive and interactive learning experiences for industry visitors, enabling hands-on exploration of complex concepts and processes.
- *Empowerment:* QR-based systems empower employees by giving them access to relevant information and resources, fostering a culture of knowledge-sharing and continuous learning.
- *Improved Safety and Risk Reduction:* AR enhances workplace safety by delivering contextualized safety information, hazard alerts, and procedural guidance to workers. AR-enabled safety training and simulations allow employees to practice emergency procedures and hazardous tasks in a controlled virtual environment, reducing the risk of accidents, injuries, and occupational hazards.
- *E-Waste:* As AR devices and accessories reach the end of their life cycle, they may contribute to electronic waste (e-waste) if not properly disposed of or recycled. E-waste disposal can pose significant environmental and health risks due to the presence of

hazardous materials such as heavy metals and toxic chemicals. Implementing take-back programs, recycling initiatives, and sustainable product design can help minimize the environmental impact of AR-related e-waste.

ii. Economic Impacts

- *Job Creation:* A thriving mobile app ecosystem supported by effective user feedback mechanisms can stimulate job creation across various sectors, including software development, user experience design, marketing, and customer support.
- *Cost of Implementation:* Implementing navigation-based AR for industry visits may require significant upfront investment in hardware, software, development, and training. Organizations need to consider the costs and benefits of deploying AR technology and assess its potential return on investment.
- *Reduced Operational Cost:* AR technology can help businesses reduce operational costs and improve efficiency by streamlining processes, enhancing productivity, and minimizing errors. For example, AR-enabled maintenance and repair procedures can decrease downtime, optimize resource utilization, and lower maintenance costs for industrial equipment and machinery.
- *Reduced Downtime:* Quick access to maintenance procedures and troubleshooting guides can minimize equipment downtime, maximizing operational efficiency and profitability.

iii. Environmental Impacts

- *Resource Efficiency:* By enabling targeted improvements based on user feedback, surveys can help reduce the environmental footprint of mobile apps by minimizing unnecessary resource consumption (e.g., data, energy) associated with inefficient or unused features.
- *Energy consumption:* The use of AR-enabled devices and infrastructure may lead to increased energy consumption, particularly if the devices require frequent charging or if additional infrastructure such as servers or networking equipment is needed to support the AR applications.
- *Energy Savings:* Efficient equipment operation, enabled by quick access to operating instructions and best practices, can lead to energy savings and reduced carbon emissions.
- *E-Waste:* As AR devices and accessories reach the end of their life cycle, they may contribute to electronic waste (e-waste) if not properly disposed of or recycled. E-waste disposal can pose significant environmental and health risks due to the presence of hazardous materials such as heavy metals and toxic chemicals.

c) Alignment of proposed solution with the Engineering codes of Ethics and taking into consideration the impacts of the design [PI-4.c]

i. Alignment taking into account the IEEE code of Ethics.

The collection of survey data adhered to the principle of anonymity, ensuring that respondents' identities were not disclosed or linked to their responses. This means that participants were not required to provide personal information, such as their names or contact details, when completing the survey. Additionally, measures were taken to safeguard the confidentiality of the data collected. This included securely storing the data and restricting access to authorized personnel only. By upholding these principles of anonymity and confidentiality, the integrity of the survey process was maintained, promoting honest and candid responses from participants without fear of repercussions or breaches of privacy.

We improve technical skills in mobile app development by learning new programming languages, frameworks, and tools. Gain expertise in platform-specific development environments, APIs, and design guidelines for iOS and Android. Understand the differences between the two platforms and optimize apps accordingly for performance, user experience, and compatibility. Explore cross-platform development frameworks such as React Native, Flutter, or Xamarin to build apps that run on multiple platforms using a single codebase. Learn the nuances of cross-platform development to maximize code reuse and minimize development time and effort.

By honestly reporting survey results, you demonstrate integrity by adhering to ethical principles and values. Integrity entails being truthful, transparent, and accountable for the accuracy and reliability of the information you provide. It ensures accuracy and objectivity in presenting findings without distortion, bias, or manipulation. It also reflected on our professionalism by upholding professional standards and norms.

ii. Alignment considering design impacts.

- Empowerment: When collecting data, special care was taken so that no important equipment is missed, and all necessary information was included. Especially the information that is frequently accessed or vital for operation.
- Reduced downtime: It is suggested that information for debugging and troubleshooting of common faults occurring in each equipment is included. This quick access to information will help workers to fix it quickly without wasting any time.
- User experience: We conducted utility survey as well as final evaluation survey to enhanced user experience.
- Jobs creation: The development of application created jobs for developers, graphic designers, data collection, and analysts.
- Increased Productivity: AR provides real-time information overlaying physical objects or environments, facilitating faster decision-making and task completion. By overlaying digital data in the real world, AR enhances situational awareness and enables workers to accomplish tasks more efficiently.
- Sustainable Practices: AR promotes sustainable practices by reducing reliance on paper-based documentation. By providing access to information digitally, AR helps create a paperless environment, minimizing the need for printing and reducing paper waste. Additionally, remote access to information through AR reduces the need for physical transportation to access manuals or documents, further contributing to environmental sustainability.

V. Project Management

Project management is the process of planning, organizing, executing, and controlling resources to achieve specific goals within a defined timeframe and budget. It involves the application of knowledge, skills, tools, and techniques to manage project activities effectively and efficiently.

1. The Schedule of GP2 Tasks

Following is the list of tasks completed by the team:

- Platform Selection
- GPS Navigation Implementation
- QR Scanner Implementation
- Equipment & Location Data
- Integration
- Application Testing
- Questionnaire Design
- Data Collection & Analysis

The Gantt chart for project management outlining the tasks and completion times are provided in . A Gantt chart is a type of bar chart that represents a project schedule. It illustrates the start and finish dates of the various elements of a project, such as tasks, milestones, and resources, over a specified time period [24].

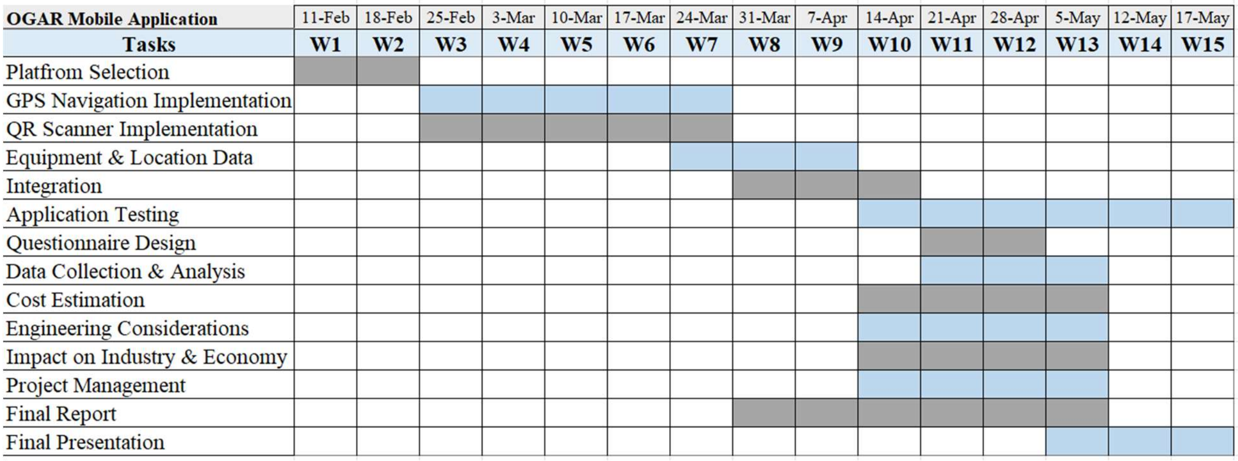

Figure 25. Gantt chart for project management

2. Encountered Problems and Proposed Solutions

Briefly discuss encountered problems during your project: managing problems, task distributions, obstacles, etc. , showing student leadership and collaboration skills.

- We encountered several challenges throughout our AR project, which necessitated effective teamwork, task delegation, and management. Below are some of the major obstacles we faced and how we overcame them:
- One significant challenge revolved choosing the suitable mobile application development platform suitable for our project. After trying various environments, we ended up with Flutter through the hit and trial method.
- We also faced problems in setting up the environment and installing the correct version of libraries and other necessary software packages. Handling the mismatch between them was particularly daunting for beginners. However, we sought guidance from our supervisor and others, which helped us resolve this issue.

- Another issue was the selection of target platforms from iOS and Android. Although, the majority of Kuwaiti nationals own iPhones but overwhelming majority of workers use Android phones and this was the reason behind our selection.
- Another significant challenge stemmed from the platform used for the project, as most group members primarily used desktops, requiring frequent consultations with supervisors and team members. Borrowing laptops from family members raised privacy concerns and logistical issues. To address this, we dedicated a laptop specifically for project-related tasks.
- Our busy schedules at the university made it challenging to meet outside of scheduled meetings. To overcome this, we arranged off-campus meetings to discuss project-related issues.
- One of the major issues we faced was the data collection which required visiting KoC and asking the workers to use our application and provide feedback.

In summary, by leveraging each team member's strengths and fostering a collaborative environment, we successfully navigated the encountered challenges and achieved our project goals.

3. Students' Responsibilities and Contribution

Achievement and contribution of each student.

Name	ID	Responsibilities
Fahad alsubaiei	49149	• Platform Selection • GPS Navigation Implementation • QR Scanner Implementation • Equipment & Location Data • Integration • Application Testing • Questionnaire Design • Data Collection & Analysis • Finalize project report (20%) • Prepare final presentation (20%)
Shaheen alsharaf	50481	• Platform Selection • GPS Navigation Implementation • QR Scanner Implementation • Equipment & Location Data • Integration • Application Testing • Questionnaire Design • Data Collection & Analysis • Finalize project report (20%) • Prepare final presentation (20%)
Ahmed alhendi	58275	• Platfrom Selection • GPS Navigation Implementation • QR Scanner Implementation • Equipment & Location Data • Integration

		• Application Testing • Questionnaire Design • Data Collection & Analysis • Finalize project report (20%) • Prepare final presentation (20%)
Khalifa alkhalifa	50210	• Platfrom Selection • GPS Navigation Implementation • QR Scanner Implementation • Equipment & Location Data • Integration • Application Testing • Questionnaire Design • Data Collection & Analysis • Finalize project report (20%) • Prepare final presentation (20%)
Abdulaziz alfoudari	48624	• Platfrom Selection • GPS Navigation Implementation • QR Scanner Implementation • Equipment & Location Data • Integration • Application Testing • Questionnaire Design • Data Collection & Analysis • Finalize project report (20%) • Prepare final presentation (20%)

VI. CONCLUSION AND FUTURE WORKS

In conclusion, this paper introduces a comprehensive solution for enhancing worker efficiency and safety in the oil and gas industry through the utilization of advanced mobile technology. The developed GPS-based navigation and QR code scanner-based augmented reality (AR) application, implemented using the Flutter framework, represents a significant step forward in providing real-time guidance and access to critical information within industrial environments. By integrating GPS technology, the application offers users a heightened level of surrounding awareness, allowing for seamless navigation through complex industrial sites. The inclusion of a QR code scanner further enhances the accessibility of relevant information, enabling workers to access detailed instructions and procedures with ease. The application's design prioritizes user-friendliness and intuitive interaction, ensuring that workers can effectively leverage its capabilities without extensive training. Through the integration of both markerless and marker-based AR technology, the application provides an immersive experience that enhances user engagement and understanding of critical procedures. The positive feedback received from participants in the experimental evaluation underscores the effectiveness of the proposed solution in addressing the needs of workers in the oil and gas industry.

In the future, continued refinement and optimization of the application based on user feedback will further enhance its utility and impact in real-world industrial settings. We intend to incorporate machine learning into the application to provide AR content for equipment without placing markers on it.

VII. References

[1] W. Zuoqian *et al.*, "Global oil and gas development in 2022: Situation, trends and enlightenment," *Pet. Explor. Dev.*, vol. 50, no. 5, pp. 1167–1186, 2023.

[2] C. D. Nwankwo, A. O. Arewa, S. C. Theophilus, and V. N. Esenowo, "Analysis of accidents caused by human factors in the oil and gas industry using the HFACS-OGI framework," *Int. J. Occup. Saf. Ergon.*, vol. 28, no. 3, pp. 1642–1654, 2022.

[3] R. Palmarini, J. A. Erkoyuncu, R. Roy, and H. Torabmostaedi, "A systematic review of augmented reality applications in maintenance," *Robot. Comput. Integr. Manuf.*, vol. 49, pp. 215–228, 2018.

[4] O. Hugues, P. Fuchs, and O. Nannipieri, "New augmented reality taxonomy: Technologies and features of augmented environment," *Handb. Augment. Real.*, pp. 47–63, 2011.

[5] S. M. H. Asraf, A. F. M. Hashim, and S. Z. S. Idrus, "Mobile application outdoor navigation using location-based augmented reality (AR)," in *Journal of Physics: Conference Series*, IOP Publishing, 2020, p. 22098.

[6] L. F. de Souza Cardoso, F. C. M. Q. Mariano, and E. R. Zorzal, "A survey of industrial augmented reality," *Comput. Ind. Eng.*, vol. 139, p. 106159, 2020.

[7] S. Sotnik and A. Andreiev, "QR codes in production." ХНУРЕ, 2023.

[8] S.-M. Jeon and K.-S. Kim, "Design and Implementation of Piping Spool Management Android Application using QR Code," *J. Digit. Contents Soc.*, vol. 13, no. 4, pp. 609–616, 2012.

[9] P. Bourhis, J. L. Reutter, and D. Vrgoč, "JSON: Data model and query languages," *Inf. Syst.*, vol. 89, p. 101478, 2020.

[10] G. Bracha, *The Dart programming language*. Addison-Wesley Professional, 2015.

[11] H. Dagne, A. A. Biya, A. Tirfie, W. W. Yallew, Z. Andualem, and B. Dagnew, "Knowledge, Attitude, and Practice of Pediculus Capitis Prevention and Control and Their Predictors among Schoolchildren in Woreta Town, Northwest Ethiopia, 2018: A School-Based Cross-Sectional Study," *Int. J. Pediatr.*, vol. 2020, p. 3619494, 2020, doi: 10.1155/2020/3619494.

[12] T. A. Syed *et al.*, "In-depth review of augmented reality: Tracking technologies, development tools, AR displays, collaborative AR, and security concerns," *Sensors*, vol. 23, no. 1, p. 146, 2022.

[13] H. Kim, Y. Kwon, H. Lim, J. Kim, Y. Kim, and W. Yeo, "Recent advances in wearable sensors and integrated functional devices for virtual and augmented reality applications," *Adv. Funct. Mater.*, vol. 31, no. 39, p. 2005692, 2021.

[14] G. Terzopoulos, I. Kazanidis, M. Satratzemi, and A. Tsinakos, "A comparative study of augmented reality platforms for building educational mobile applications," in *Internet of Things, Infrastructures and Mobile Applications: Proceedings of the 13th IMCL*

Conference 13, Springer, 2021, pp. 307–316.

[15] I. Dalmasso, S. K. Datta, C. Bonnet, and N. Nikaein, “Survey, comparison and evaluation of cross platform mobile application development tools,” in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, 2013, pp. 323–328.

[16] M. A. M. M. Alsaid, T. M. Ahmed, S. Jan, F. Q. Khan, and A. U. Khattak, “A Comparative Analysis of Mobile Application Development Approaches: Mobile Application Development Approaches,” *Proc. Pakistan Acad. Sci. a. Phys. Comput. Sci.*, vol. 58, no. 1, pp. 35–45, 2021.

[17] M. Kuitunen, “Cross-Platform Mobile Application Development with React Native.” 2019.

[18] M. Rebouças, G. Pinto, F. Ebert, W. Torres, A. Serebrenik, and F. Castor, “An empirical study on the usage of the swift programming language,” in *2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER)*, IEEE, 2016, pp. 634–638.

[19] N. Marangunić and A. Granić, “Technology acceptance model: a literature review from 1986 to 2013,” *Univers. access Inf. Soc.*, vol. 14, pp. 81–95, 2015.

[20] M. A. Murad, N. A. S. Abdullah, and M. M. Rosli, “Software Cost Estimation for Mobile Application Development-A Comparative Study of COCOMO Models,” in *2021 IEEE 11th International Conference on System Engineering and Technology (ICSET)*, IEEE, 2021, pp. 106–111.

[21] U. Killström, H. Virola, L. Galli, O. Immonen, O. Pitkänen, and B. Kijl, “Business models for new mobile applications and services,” 2006.

[22] J. Estdale and E. Georgiadou, “Applying the ISO/IEC 25010 quality models to software product,” in *Systems, Software and Services Process Improvement: 25th European Conference, EuroSPI 2018, Bilbao, Spain, September 5-7, 2018, Proceedings 25*, Springer, 2018, pp. 492–503.

[23] M. Davis, “Professional Codes of Ethics,” in *The Routledge Handbook of the Philosophy of Engineering*, Routledge, 2020, pp. 580–591.

[24] V. Krcheva, M. Dzidrov, S. Srebrenkoska, and D. Krstev, “Gantt chart as a project management tool that represents a clutch hub manufacturing process,” *Balk. J. Appl. Math. Informatics*, vol. 6, no. 2, pp. 67–78, 2023.

VIII. Appendices

OGAR: Augmented Reality Application for Assisting Workers Oil and Gas

1. Gender
   - o Male o Female o Others

2. Company Name:

3. Job Title:

4. Years of Experience in the Oil and Gas Industry:

5. Education
   - o Certificate o Diploma o Bachelor's o Masters o PhD

- usefulness

6. Using OGAR improves my job performance.
   - o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree
7. The OGAR application addresses my job-related needs.
   - o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree
8. Using OGAR saves me time.
   - o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree
9. Using OGAR reduces my time on unproductive activities (for example looking for datasheet, user manual, datasheet).
   - o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree
10. Using OGAR improves the quality of the work I do.
   - o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree
11. Using OGAR increases my productivity.
   - o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree
12. Using OGAR makes it easier to do my job.
   - o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree

- **Ease**

13. It is easy to use OGAR.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

14. Interacting with the OGAR application is often frustrating.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

15. I needed help when using OGAR.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

16. Interacting with the OGAR application requires a lot of my mental effort.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

17. The OGAR application often behaves in unexpected ways.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

18. I find it cumbersome or difficult, to use the OGAR application.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

19. My interaction with the OGAR application is easy for me to understand.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

20. The OGAR application provides helpful guidance in performing tasks.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

- **ATU**

21. I believe it is a good idea to use the OGAR application.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

22. I think it is valuable to use the OGAR application.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

- **BIU**

23. I plan to use the OGAR application in the future.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

24. I'd love to use multimedia materials in my class.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

25. I will recommend using the OGAR application in the future.

    - Strongly Disagree
    - Disagree
    - Neutral
    - Agree
    - Strongly Agree

- **Relevance**

26. In my job, the usage of the OGAR is relevant.

   o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree

27. In my job, the usage of the OGAR is important.

   o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree

28. What kind of information would you like to see on the application?

29. What will you suggest improving this application?