POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Optimizing Industrial Operations: A Web Application for QR Code-Based Machinery Information Management

Candidate	Supervisors	Company
FRANCESCO SANTORO	Prof. STEFANO QUER	AROL GROUP

JULY 2024

SUMMARY

In today's context, digitalization and technological innovation are fundamental to improving efficiency and traceability in industrial processes. The modern industrial landscape is increasingly characterized by the integration of advanced technologies such as the Internet of Things (IoT), big data analytics, and artificial intelligence (AI), which are transforming traditional manufacturing and production methods. Amidst this wave of technological advancement, companies are striving to adopt smart solutions to remain competitive and enhance operational efficiency. AROL, a leader in the industrial automation sector, has recognized the importance of implementing advanced solutions to optimize the management and monitoring of its machinery and components. In response to this need, the idea emerged to develop a web application that utilizes QR code technology to provide detailed and accessible information on machinery and components. This innovation aligns with the broader industry trend towards greater connectivity and data-driven decision-making, enabling real-time access to critical information and facilitating predictive maintenance. The application aims to enhance the transparency and efficiency of operations by allowing users to scan QR codes printed on machinery and their components to access a wealth of relevant information. Depending on the user's privileges, the application will display varying levels of detailed information, ensuring that sensitive data is protected and accessible only to authorized personnel. This thesis, focusing on the development of the front end of the application, is part of a larger company-wide project. It collaborates with another thesis student responsible for managing the back end using microservices. The front-end development emphasizes creating an intuitive and responsive user interface that can adapt to various devices and user needs, ensuring seamless interaction and a high level of user satisfaction. The relevance of this work lies in its ability to significantly improve operational efficiency and information management at AROL. The application will enable operators and technicians to quickly access critical data, thereby reducing downtime and increasing productivity. This rapid access to information is crucial in an industrial environment where time and efficiency directly impact profitability and operational success. Furthermore, the ability to differentiate the visible information based on user privileges ensures that only authorized personnel can access sensitive data, thereby enhancing information security. This feature is particularly important in an era where data breaches and unauthorized access pose significant risks to companies.

By providing a secure and user-friendly platform, the application supports AROL's commitment to maintaining high standards of operational integrity and data protection. The specific objectives include:

- **Create an Intuitive and Responsive User Interface**: Develop a front end that is user-friendly and responsive, ensuring accessibility from various devices such as desktops, tablets, and smartphones. This will enhance user experience and ensure efficient interaction with the application.
- **Implement a Robust Authentication and Authorization System**: Develop a system that allows differentiation of content visibility based on user roles and privileges. This ensures that sensitive information is accessible only to authorized personnel, enhancing data security.
- **Integrate QR Code Scanning Functionality**: Implement functionalities that enable users to scan QR codes using their devices, leading them directly to the relevant information on the application. This will facilitate quick and easy access to necessary data.
- **Develop QR Code Generation Mechanism**: Create a mechanism within the application that generates unique QR codes for different machinery and components. This will standardize the identification process and ensure consistency across the system.
- **Ensure Data Security and Integrity**: Incorporate security measures to protect data from unauthorized access and ensure the integrity of the information displayed.
- **Enhance Operational Efficiency**: By providing real-time access to critical information and streamlining the data retrieval process, the application aims to reduce downtime, improve maintenance procedures, and increase overall productivity.

All this will be explained in the following chapters, starting first with some historical background and how technology is changing the world of work and beyond, and then proceeding to the actual design and implementation of the application, with an overview of all the tools used. By achieving these objectives, the application will support AROL's efforts to leverage technology for better operational management, data security, and user satisfaction. To achieve its goals, this thesis adopted an iterative and incremental software development methodology. The main steps in the process include:

- **Requirements Analysis:** Collection and definition of functional and non-functional requirements through interviews and workshops with AROL stakeholders.
- **Design:** Creating prototypes of the user interface and defining the front-end architecture.
- **Development:** Implementation of the front end using the Angular framework, with emphasis on responsiveness and user experience.
- **Testing:** Performing unit, integration, and usability testing to ensure that the application meets requirements and is bug-free.

The research and development process has addressed the key objectives set at the beginning, resulting in a user-friendly, secure, and efficient application tailored to   the specific needs of AROL's operations.

CONTENT

LIST OF FIGURES

CHAPTER 1

1. Introduction

In today's highly competitive industrial landscape, the ability to collect, analyze, and utilize data effectively is paramount. Detailed and accurate data on machinery performance, maintenance schedules, and operational parameters play a crucial role in optimizing production processes, minimizing downtime, and ensuring the longevity of equipment. Effective data collection and management systems provide a wealth of information that can drive informed decision-making. Real-time data can reveal insights into the operational health of machinery, predict potential failures, and identify opportunities for efficiency improvements. This proactive approach to equipment management not only enhances productivity but also reduces operational costs by preventing unplanned downtime and extending the life cycle of machinery. The availability of this data for consultation with employees across various departments is equally important. When employees have instant access to up-to-date information on machinery, they can swiftly address issues, plan maintenance activities more efficiently, and implement improvements based on real-time insights. Maintenance teams can schedule preventive interventions, production managers can optimize workflow, and quality assurance can monitor adherence to standards, all through a unified data access platform. This seamless flow of information not only boosts operational efficiency but also supports the overall strategic goals of the company. Moreover, in an era where digital transformation is reshaping industries, the integration of user-friendly data access solutions becomes crucial. Empowering employees with the tools to easily retrieve and interpret machinery data fosters a culture of continuous improvement and innovation. It enhances communication across departments, promotes accountability, and ensures that everyone from technicians to executives is aligned with the company's objectives and performance metrics. This thesis will delve into the technical aspects of designing and implementing such an application, the challenges involved in ensuring data accuracy and security, and the potential impacts on operational efficiency and employee productivity. By examining case studies and conducting empirical research, the study aims to demonstrate how such a system can be a game-changer for AROL and similar companies, driving them towards greater technological integration and operational excellence.

Figure 1: AROL logo.

1.1 The Company

Founded in 1978, AROL S.p.A. has established itself as a pioneering force in the field of capping and closure systems for the packaging industry. Headquartered in Canelli, Italy, the company emerged with a vision to provide innovative and reliable solutions for sealing containers, addressing the ever-growing demands of the global market. Design 100% of their machines and 95% of the components are manufactured in-house. AROL technical support is available for the entire machine life cycle and relies on a highly qualified team of specialists operating from each of their 11 offices around the world. From its inception, AROL has focused on technological advancements and quality, enabling it to rapidly expand its product portfolio and market reach. The company's commitment to research and development has led to the creation of state-of-the-art machinery designed to meet the specific needs of various industries, including food and beverage, cosmetics, pharmaceuticals, and household products. Over the years, AROL has evolved from a small local enterprise into an internationally recognized leader, thanks to its robust engineering capabilities and dedication to customer satisfaction. The company has continuously invested in modernizing its production processes and incorporating cutting-edge technologies such as Industry 4.0, ensuring efficiency, precision, and sustainability. By incorporating advanced sensors, intelligent automation, and real-time monitoring systems, AROL has enhanced the efficiency and precision of its machinery, ensuring optimal production processes and minimizing downtime. In line with the growing demand for sustainable solutions, AROL has invested heavily in the development of eco-friendly closure systems. These systems utilize recyclable materials and technologies designed to reduce environmental impact, reflecting the company's commitment to sustainability. Furthermore, AROL has introduced innovative sealing technologies that enhance the safety and quality of closures. These advancements are particularly crucial in the food and pharmaceutical industries, where product integrity and contamination prevention are paramount. The company's adoption of advanced automation and robotics has also been a key driver of increased productivity and cost reduction. By integrating these technologies into their production lines, AROL has achieved greater precision in the closure process, reducing the need for human intervention and thereby boosting overall efficiency. AROL's ability to deliver customized solutions tailored to the specific needs of its clients sets it apart from competitors. This bespoke approach allows the company to swiftly respond to market trends and unique customer requirements, ensuring that their solutions are always relevant and effective. These initiatives highlight how AROL S.p.A. not only meets current market demands but also anticipates future trends, maintaining its leadership position in the field of capping and sealing solutions for the packaging industry. Today, AROL S.p.A. stands as a testament to Italian excellence in engineering and innovation, driven by a passion for progress and a relentless pursuit of perfection.

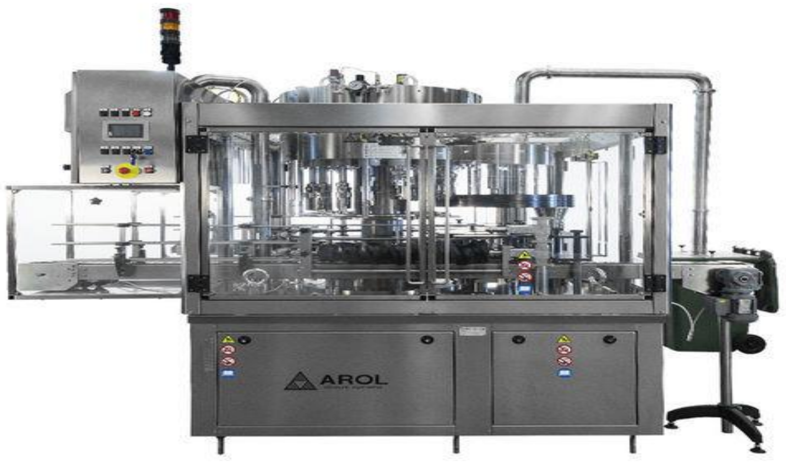


Figure 2: An example of an AROL capping machinery.

1.2 Objectives

Starting from the continuous development and technological advances that characterize AROL S.p.A., the idea for my thesis was conceived. Given AROL's commitment to innovation and the integration of cutting-edge technologies, the project aims to develop a web application, for all the devices, capable of generating barcodes specific to each machine or its components. This initiative will allow customers to scan a QR code on the machinery, either through an application or directly through the device's camera, directing them to a dedicated website where all relevant information about that specific machine will be displayed. This solution improves accessibility and provides immediate access to crucial data, further reflecting AROL's commitment to improving customer experience and operational efficiency through technological innovation. The main goal, therefore, is to make information regarding machinery easily accessible to all employees at AROL's various locations, as well as to buyers of their products. In the latter case, of course, it will only be able to access information about the machinery purchased by the company itself and not any AROL machinery that you do not own, thus maintaining a distinction at the level of privileges for the users of this application.

1.3 Tools used

For the development of this application, a variety of tools and technologies were utilized to ensure it met the high standards and integration capabilities required by AROL S.p.A. The front end of the application was developed using Angular, JS/Typescript, CSS, and HTML, all of which were chosen to create a dynamic and responsive user interface. Angular was specifically selected to align with AROL's existing suite of applications, ensuring consistency and compatibility across their technological ecosystem.

The front-end development was my responsibility, and I focused on creating an intuitive and efficient interface that enhances user experience. The back end of the application, on the other hand, was developed by another thesis student who focused on creating a robust series of microservices. These microservices are designed to seamlessly connect to the front-end application, providing the necessary data and functionality to support the application's features.

This modular approach not only enhances the scalability and maintainability of the system but also allows for independent development and deployment of various application components, facilitating continuous improvement and integration.

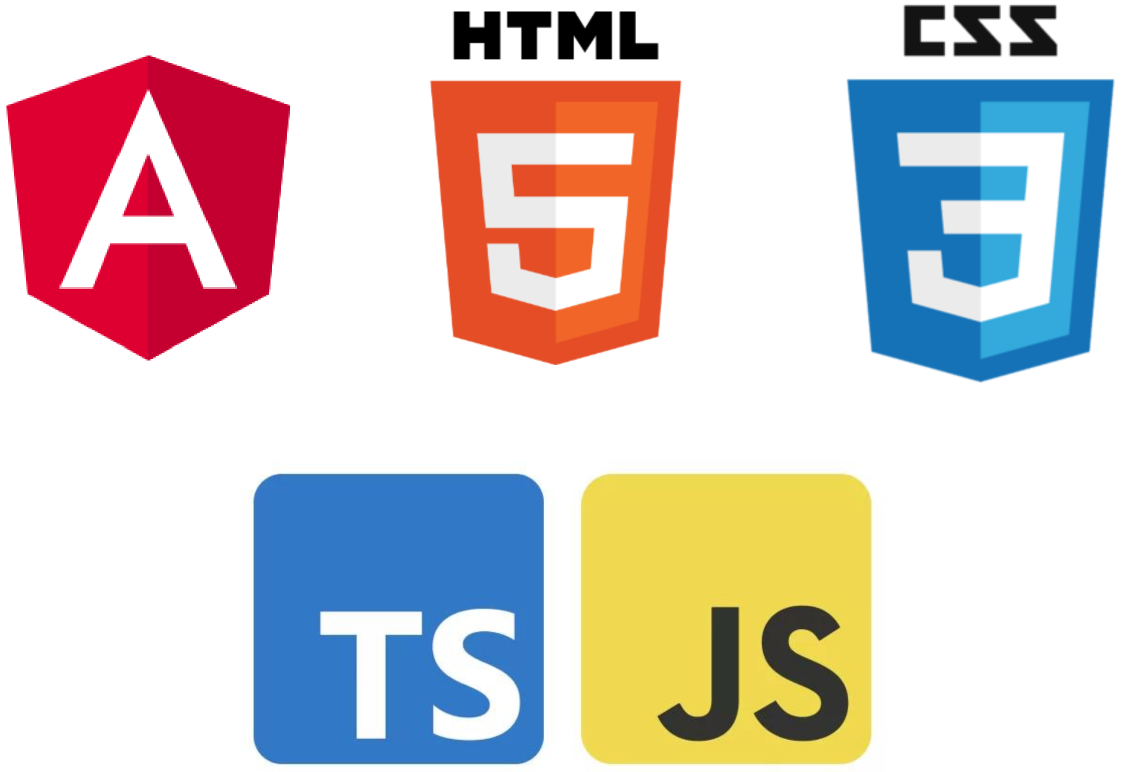


Figure 3: List of tools used.

CHAPTER 2

The process from production to consumption has undergone a remarkable transformation over the past few decades, driven by technological advancements and changing consumer dynamics. Understanding this evolution is crucial for companies aiming to stay competitive in an increasingly digital and interconnected marketplace. This chapter explores the multifaceted nature of the company-consumer relationship, and the pivotal role technology plays in shaping it. Overall, this chapter aims to provide a comprehensive overview of how technological innovations, such as QR codes, have transformed the path from production to consumption. By examining the interplay between technology and consumer dynamics, we gain insights into the strategies businesses can employ to enhance operational efficiency, comply with regulatory standards, and ultimately foster a stronger connection with their customers in a rapidly evolving digital landscape.

2. The Path from Production to Consumption

The journey from producer to consumer is a critical aspect of the supply chain that encompasses all stages of the manufacturing of products to their final delivery to the end users. Ensuring a seamless and efficient transition through these stages is paramount. AROL's sophisticated machinery plays a vital role in the initial stages of this journey. By providing reliable and high-quality capping and sealing solutions, AROL ensures that products are securely packaged, preserving their quality and safety throughout the distribution process. This initial step is crucial as it sets the foundation for the subsequent handling, storage, and transportation phases. As products move from the production facilities to warehouses, and eventually to retail locations or directly to consumers, maintaining the integrity of the packaging is essential. AROL’s innovative technologies, including real-time data collection and advanced sealing mechanisms, help in tracking and monitoring the condition of products, thereby reducing the risk of damage or contamination. This not only enhances the product's shelf life but also builds consumer trust in the brand's commitment to quality. Furthermore, the integration of digital solutions such as the application developed in this thesis can significantly enhance the transparency and efficiency of this journey. By enabling employees and stakeholders to access detailed information about each machine and component through QR codes, the application ensures that any potential issues can be quickly identified and addressed, minimizing disruptions in the supply chain. In a broader context, the smooth flow from producer to consumer relies heavily on robust logistics and supply chain management. This includes everything from efficient inventory management to timely transportation and effective communication between all parties involved. AROL's commitment to continuous improvement and technological innovation plays a pivotal role in optimizing these processes, ensuring that products reach consumers in optimal condition and within the expected timeframe. In conclusion, the journey from producer to consumer is a complex and multifaceted process that requires meticulous planning and execution. AROL's advanced machinery and innovative solutions significantly contribute to the efficiency and reliability of this journey, ultimately enhancing the overall consumer experience and strengthening the company's reputation in the market.

Figure 4: Customer relationship.

2.1 Evolution of the Company-Consumer Relationship

In the contemporary digital landscape, the relationship between producers and consumers has undergone a profound transformation, driven by the internet and increasingly sophisticated, goal-oriented applications. This evolution has reshaped how businesses like AROL S.p.A. interact with their customers, offering unprecedented levels of engagement, transparency, and personalization.

Figure 5: Cloud connection.

2.1.1 Enhanced Communication and Engagement

The Internet has facilitated direct and continuous communication between producers and consumers. Companies can now engage with their audience through various online channels, including social media, email newsletters, and customer service portals. This direct line of communication enables producers to respond to customer inquiries, address concerns, and gather feedback in real-time. This means maintaining a closer connection with the clients, understanding their needs more precisely, and delivering tailored solutions that meet those needs.

2.1.2 Increased Transparency

Consumers today demand transparency regarding the products they purchase. The internet allows producers to provide detailed information about their production processes, sourcing of materials, and quality assurance measures. It’s possible to leverage digital platforms to share insights about all of this information, building trust and confidence among the customers, and ensuring they are well-informed about the products they are using.

2.1.3 Personalization and Customization

Targeted applications enable producers to offer personalized experiences to their customers. By analyzing data collected through these applications, companies can gain insights into individual preferences and behaviours. This means they can customize their offerings and services to better align with the specific needs of each client. Personalized marketing campaigns, product recommendations, and tailored support services enhance customer satisfaction and loyalty.

2.1.4 Efficient and Convenient Access to Information

With the proliferation of smartphones and mobile applications, consumers have instant access to a wealth of information at their fingertips. Applications designed for specific purposes, such as the one developed in this thesis, allow customers to quickly access detailed information about machinery and its components. This convenience not only improves the user experience but also empowers customers to make informed decisions and troubleshoot issues independently.

2.1.5 Data-Driven Decision Making

The internet and targeted applications provide producers with valuable data that can inform strategic decisions. By leveraging analytics, it’s possible to track customer interactions, monitor product performance, and identify trends. This data-driven approach allows producers to continually refine their products and services, ensuring they remain competitive and relevant in a rapidly changing market.

2.1.6 Building Long-Term Relationships

The continuous interaction facilitated by the Internet helps in building long-term relationships with customers. By maintaining an active online presence and utilizing targeted applications, companies can foster a sense of community and loyalty among their customers. Regular updates, exclusive offers, and personalized communication strengthen these relationships, leading to increased customer retention and advocacy.

2.1.7 Adapting to Consumer Expectations

As consumer expectations evolve, producers must adapt to meet them. The internet and targeted applications enable companies to stay agile and responsive to these changes. For AROL, this means adopting new technologies and continuously improving their digital interfaces to provide a seamless and satisfying customer experience. This adaptability ensures that they can meet the demands of the modern consumer and stay ahead of industry trends. In conclusion, the relationship between producers and consumers has become more dynamic and interactive through the use of the internet and targeted applications.

Leveraging these digital tools allows for enhanced communication, increased transparency, personalized experiences, and data-driven decision-making, all of which contribute to building stronger, more meaningful relationships with their customers. This evolution not only benefits consumers by providing them with better products and services but also helps producers stay competitive and innovative in a digital world.

2.2 Web and Smartphone

In the modern digital age, the journey from producer to consumer has been significantly transformed by the pervasive use of the web and smartphones, and the integration of digital technologies into this journey enhances efficiency, transparency, and consumer satisfaction. The web plays a crucial role in streamlining the supply chain. Through robust online platforms, a company can provide real-time updates and comprehensive data about its machinery and production processes. These platforms facilitate better communication and coordination among different stakeholders, including suppliers, manufacturers, and distributors. By leveraging cloud-based solutions, we ensure that all parties have access to the most current information, enabling quicker decision-making and more efficient operations. Smartphones further revolutionize this process by offering mobile access to essential data and tools. Employees and partners can use their smartphones to scan QR codes on machinery, instantly retrieving detailed information about the equipment's status, maintenance history, and operational parameters.

This instant access to information not only enhances troubleshooting and maintenance efficiency but also ensures that any issues can be addressed promptly, minimizing downtime and disruptions in the supply chain. For consumers, the integration of web and smartphone technologies offers greater transparency and engagement. These digital solutions allow end-users to track the journey of products from production to delivery, providing them with detailed insights into the sourcing, manufacturing, and quality assurance processes. This transparency builds trust and confidence in the brand, as consumers can verify the integrity and authenticity of the products they purchase. Additionally, smartphones enable direct communication and feedback between consumers and producers.

Through mobile apps and responsive websites, consumers can provide feedback, ask questions, and receive support, enhancing their overall experience and satisfaction. This direct line of communication helps companies to continuously improve their products and services based on real-time consumer feedback. Moreover, the use of web and smartphone technologies facilitates personalized marketing and customer engagement strategies. By analyzing data collected through these digital platforms, it’s possible to gain valuable insights into consumer preferences and behaviours, allowing for more targeted and effective marketing campaigns.

This personalized approach not only drives sales but also fosters long-term customer loyalty. The integration of web and smartphone technologies into the producer-to-consumer journey has brought about significant improvements in efficiency, transparency, and consumer engagement. Leveraging these digital tools ensures that advanced systems are not only reliable and efficient but also aligned with the evolving expectations of the digital age.

This seamless integration of technology enhances every stage of the supply chain, ultimately delivering superior value to consumers and strengthening the market position of companies.

Figure 6: The Internet power.

2.3 The Origin of the Barcode

The QR code, short for Quick Response code, has a rich history that began in the early 1990s. Developed by Denso Wave, a subsidiary of the Toyota Group in Japan, the QR code was created to meet the growing demands of the automotive industry. Traditional barcodes were proving insufficient for the complex and fast-paced production processes, as they could store only a limited amount of data and required precise alignment to be scanned. Masahiro Hara, the chief engineer at Denso Wave, led the team that developed the QR code in 1994, aiming to create a more efficient and robust system for tracking automotive parts. The innovative design of the QR code allowed it to hold significantly more information than traditional barcodes. Unlike linear barcodes, which store data in one dimension, QR codes use a two-dimensional matrix format, enabling them to store data both horizontally and vertically. This 2D structure allows QR codes to encode a wide range of information, including text, URLs, images, and other data types, in a compact and accessible form. A QR code consists of black squares arranged on a white background. These squares are strategically positioned to include several key components that ensure the code can be read accurately and efficiently.

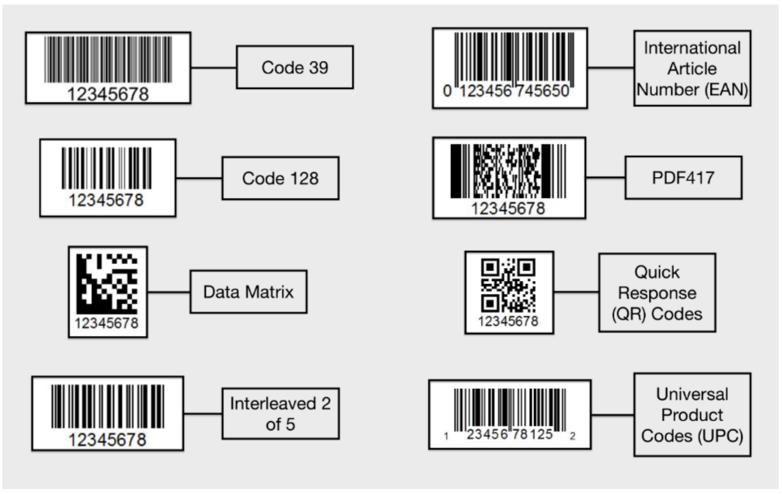


Figure 7: Example of Barcodes formats.

A QR code consists of black squares arranged on a white background. These squares are strategically positioned to include several key components that ensure the code can be read accurately and efficiently. The main elements of a QR code are:

1. **Position Markers**: These are the three large squares located at three corners of the QR code. They help the scanner determine the orientation of the code, ensuring that it can be read from any angle. This feature, known as omnidirectional scanning, is a significant advantage over traditional barcodes, which must be aligned precisely with the scanner.
2. **Alignment Markers**: These are small squares located near the bottom-right corner of the QR code and are used to correct for perspective distortions. They are particularly useful when the QR code is printed on curved or uneven surfaces, helping the scanner maintain accuracy.
3. **Timing Patterns**: These consist of alternating black and white modules arranged in a line and are located between the position markers. They help the scanner determine the size of the data modules and ensure consistent reading.
4. **Quiet Zone**: This is a margin of white space surrounding the QR code. It acts as a buffer zone, ensuring that the code can be distinguished from surrounding text or graphics.
5. **Data Area**: This is where the actual information is encoded. Depending on the version of the QR code, the data area can hold different amounts of information. The smallest

version (Version 1) is 21x21 modules, while the larger version (Version 40) is 177x177 modules. The more data stored, the larger the QR code.

The process of scanning a QR code is both simple and efficient. When a QR code is scanned by a QR code reader or a smartphone camera, the scanner first identifies the position markers to determine the orientation of the code. It then uses the timing patterns to gauge the module size and proceeds to decode the data area. Advanced error correction algorithms, such as Reed-Solomon error correction, ensure that even if part of the QR code is damaged or obscured, the information can still be retrieved accurately.

One of the primary advantages of QR codes is their versatility. They can be printed on any virtual surface, from paper and plastic to metal and fabric. Their ability to store several types of data makes them useful for a wide range of applications, including marketing, inventory management, and mobile payments. For instance, QR codes can encode URLs, allowing users to access websites instantly by scanning the code with their smartphones. This capability has been widely adopted in advertising and product packaging, where QR codes link consumers to promotional content or detailed product information. Over the years, QR codes have evolved beyond their original use in the automotive industry. In the early 2000s, QR codes gained popularity in Japan, where they were used for various consumer-facing applications such as digital business cards (vCards), mobile payments, and ticketing systems. The widespread adoption of smartphones equipped with cameras and internet connectivity in the 2010s further accelerated the global use of QR codes. The COVID-19 pandemic significantly boosted the adoption of QR codes worldwide. Businesses and organizations sought contactless solutions to reduce physical interaction and prevent the spread of the virus.

QR codes emerged as an effective tool for facilitating contactless payments, digital menus in restaurants, virtual check-ins, and even contact tracing. Their ability to provide instant access to information without physical contact made them an ideal solution in a time of social distancing. The success of QR codes can also be attributed to their adaptability and continuous improvement. For example, the introduction of dynamic QR codes, which can be updated with new information without changing the code itself, has expanded their utility.

Dynamic QR codes are particularly useful in scenarios where the encoded information needs to be frequently updated, such as event schedules, promotional offers, or inventory levels. In addition to their practical applications, QR codes have also influenced the development of other technologies. For instance, the concept of encoding information in a two-dimensional pattern has inspired advancements in augmented reality (AR) and the Internet of Things (IoT). QR codes are often used to link physical objects to digital information, creating interactive and immersive experiences for users. In summary, the QR code's origins in the automotive industry have led to its widespread adoption across various sectors due to its ability to store extensive data, ease of use, and versatility. Its development of Denso Wave marked a significant milestone in the evolution of data encoding and retrieval, setting the stage for numerous innovations in the digital age. As technology continues to advance, QR codes are likely to remain a fundamental tool in bridging the physical and digital worlds, offering efficient and innovative solutions for a wide array of applications.

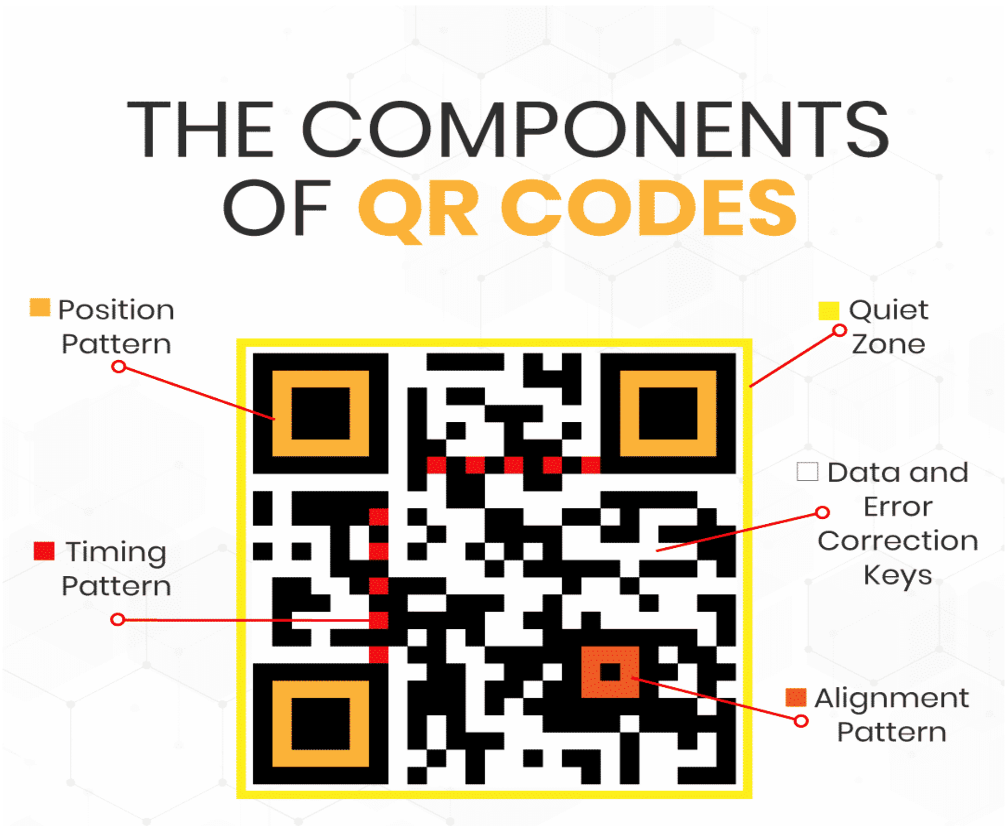


Figure 8: QR code's components.

2.4 How QR Codes Can Change Company Operations

Figure 9: Scanning Qr Code.

QR codes have the potential to significantly enhance the operational efficiency and effectiveness of businesses across various industries. Their ability to store a large amount of data and facilitate instant access to information makes them a powerful tool for streamlining processes, improving customer engagement, and enhancing overall productivity. Here's a detailed look at how QR codes can transform business operations:

2.4.1 Streamlined Inventory Management

One of the primary applications of QR codes in business operations is inventory management. Businesses can use QR codes to tag products with detailed information such as product ID, location, quantity, and even expiration dates. This enables real-time tracking of inventory, reduces the chances of human error, and speeds up the process of stocktaking. Employees can simply scan the QR code on a product to instantly access its details, facilitating faster and more accurate inventory audits.

2.4.2 Enhanced Product Information

QR codes can be used to provide customers and employees with detailed product information. By scanning a QR code on a product's packaging, customers can access information such as product specifications, usage instructions, ingredients, and customer reviews. This not only enhances the customer experience by providing easy access to relevant information but also helps in building trust and transparency. For employees, especially in retail or warehousing, having instant access to detailed product information can improve efficiency and accuracy in tasks such as stocking shelves, processing returns, and assisting customers with inquiries.

2.4.3 Improved Customer Engagement and Marketing

In the realm of marketing, QR codes offer a versatile and interactive way to engage with customers. Businesses can place QR codes on their marketing materials—such as flyers, brochures, posters, and advertisements that link to promotional content, special offers, or company websites. This allows customers to engage with the brand effortlessly by simply scanning the code with their smartphones. QR codes can also be used in loyalty programs, where customers can scan codes to earn points, receive discounts, or participate in contests. This creates a seamless and engaging customer experience, encouraging repeat business and fostering brand loyalty.

2.4.4 Efficient Payment Systems

QR codes have revolutionized the way businesses handle transactions, particularly in the retail and hospitality sectors. By integrating QR codes into their payment systems, businesses can offer a quick, secure, and contactless payment method. Customers can scan a QR code at the point of sale using their smartphones to complete a transaction instantly, without the need for physical cash or cards. This not only speeds up the checkout process but also enhances security by reducing the risk of physical contact and the transmission of pathogens-a feature that has become particularly important in the wake of the COVID-19 pandemic.

2.4.5 Seamless Access to Digital Services

For service-oriented businesses, QR codes can facilitate easy access to digital services. For example, restaurants can place QR codes on tables that link to digital menus, allowing customers to view the menu, place orders, and even pay for their meals using their smartphones. This not only improves the customer experience but also reduces the need for physical menus and minimizes interaction with staff. In the healthcare industry, QR codes can be used to provide patients with access to their medical records, appointment scheduling, and telemedicine services. By scanning a QR code, patients can securely access their health information and manage their care more efficiently.

2.4.6 Enhanced Tracking and Traceability

QR codes can significantly improve tracking and traceability in supply chain management. By tagging products with QR codes that contain detailed information about their origin, manufacturing process, and distribution, businesses can ensure greater transparency and accountability. This is particularly important in industries such as food and pharmaceuticals, where traceability is critical for ensuring product safety and compliance with regulatory standards. In the event of a product recall, QR codes enable businesses to quickly identify, and trace affected products through the supply chain, facilitating a more efficient and targeted response.

2.4.7 Simplified Maintenance and Support

In industrial and manufacturing settings, QR codes can simplify maintenance and support processes. Equipment and machinery can be tagged with QR codes that link to detailed maintenance records, user manuals, troubleshooting guides, and instructional videos. Maintenance personnel can access this information on-site by scanning the QR code, enabling them to perform repairs and maintenance more efficiently and accurately. This approach reduces downtime, improves the reliability of equipment, and ensures that maintenance is carried out according to the manufacturer's specifications.

2.4.8 Enhanced Security and Access Control

QR codes can also be used to enhance security and access control within a business. For example, employees can use QR codes on their ID badges to gain access to secure areas or log into company systems. This provides a secure and efficient way to manage access and track employee movements within a facility. In events and conferences, QR codes can be used for attendee registration and access control, streamlining the check-in process and enhancing security by ensuring that only authorized individuals can enter.

2.4.9 Environmental Benefits

By reducing the need for printed materials, QR codes contribute to more sustainable business practices. Digital menus, manuals, brochures, and receipts accessed through QR codes minimize paper usage and waste, supporting environmental conservation efforts.

The integration of QR codes into business operations can lead to significant improvements in efficiency, accuracy, customer engagement, and overall productivity. Their versatility and ease of use make them a valuable tool for businesses looking to streamline processes, enhance customer experiences, and stay competitive in an increasingly digital world. As technology continues to evolve, the potential applications of QR codes are likely to expand even further, offering new and innovative ways to transform business operations.

2.5 MOCA Regulation and the Role of QR Codes

The MOCA (Materials and Objects in Contact with Food) regulation is a critical framework established to ensure the safety and suitability of materials intended to come into contact with food. These regulations cover a broad spectrum of materials, including plastics, ceramics, metals, paper, and adhesives, ensuring that they do not release harmful substances into the food. The primary objective of MOCA is to protect consumer health by preventing contamination and ensuring that all materials meet stringent safety and quality standards. MOCA regulations mandate rigorous testing and certification processes for materials and finished products. These tests are designed to assess the potential migration of substances from the materials into the food, ensuring that any transfer remains within safe limits. Manufacturers must provide detailed documentation demonstrating compliance with these standards, which includes specifications on the material composition, production processes, and any treatments or coatings applied. In addition to ensuring the safety of the materials themselves, MOCA regulations also require manufacturers to maintain thorough traceability throughout the supply chain. This means that all stages of production, processing, and distribution must be documented, allowing for complete transparency and accountability. Traceability is crucial for identifying and addressing any issues that may arise, such as contamination or non-compliance with safety standards. Compliance with MOCA regulations is not only a legal requirement but also a key aspect of maintaining consumer trust and brand reputation. Non-compliance can lead to severe penalties, including fines, product recalls, and damage to a company's reputation. Therefore, adherence to MOCA standards is a fundamental responsibility for all businesses involved in the production and distribution of food-contact materials. In this context, QR codes offer a powerful tool to enhance and simplify the enforcement of MOCA regulations. By embedding detailed and comprehensive information about the materials, manufacturing processes, and safety compliance directly into a QR code, stakeholders throughout the supply chain can easily access and verify critical data. Manufacturers can use QR codes to provide regulators with instant access to compliance documentation, certificates of analysis, and traceability information, thereby streamlining the inspection and certification processes. For consumers, QR codes offer transparency and peace of mind. By scanning a QR code on food packaging or kitchenware, consumers can immediately access information about the safety standards and certifications that the product meets, including any potential hazards or safety precautions. This level of transparency not only builds trust but also empowers consumers to make informed choices about the products they use. Moreover, the use of QR codes can facilitate rapid response in the event of a safety breach or recall. Regulators and manufacturers can quickly trace and identify affected batches, ensuring that any non-compliant products are swiftly removed from the market. Overall, the integration of QR codes within the framework of MOCA compliance enhances the efficiency, accuracy, and transparency of regulatory processes, benefiting manufacturers, regulators, and consumers alike.

MOCA
Materiali e oggetti a contatto con gli alimenti
Art.6 del D.lgs.n.29 del 10/02/2017

Figure 10: MOCA regulations.

CHAPTER 3

The concept for this thesis was conceived in collaboration with AROL, a company that identified the need for a mechanism to swiftly track and access information about their machinery. This system was intended to benefit not only AROL employees but also all purchasers of their machinery. The primary objective was to develop an application capable of generating unique QR codes based on the serial number, product ID, and scope of any machine or its components. The inclusion of the scope was crucial because the same component could be found in multiple different machines, thus necessitating an even more precise aggregation of information. During several visits to the company, it became evident that the integration of such technology could significantly enhance operational efficiency. By printing these QR codes on the machinery, employees would be able to scan the codes and navigate to the application to retrieve comprehensive information about the scanned item. This functionality was designed to be particularly beneficial in the warehouse, where various supplies are stored and dispatched. Employees can only scan machinery or components belonging to their company. For external machinery, they will only be able to read more general information. The information accessible via the QR codes includes, but is not limited to, operational instructions, assembly guides, and batch details. This dual advantage serves both to accelerate the learning process for certain machine-related knowledge and to alert the relevant personnel in case of an anomalous malfunction, thereby allowing them to check if other components from the same batch exhibit similar issues. The following chapters will delve into the functional and non-functional requirements considered during the development of this application, as well as a comprehensive analysis of the use cases envisioned for this system. This detailed examination aims to provide a thorough understanding of the project’s inception, development, and potential impact on both AROL and its customers. Through this exploration, we aim to demonstrate the significant benefits and efficiencies that such an application can bring to the company's operational processes and customer support services.

3. Requirements Analysis

Requirements specification is the activity wherein the application analyst elaborates on the business requirements that drive the application development, considering all available information regarding the technical, organizational, and managerial context within which the application must operate. This process translates these inputs into a detailed specification of the functionalities that the application must deliver. The requirements specification comprises two phases: requirements gathering and subsequent analysis.

The gathering phase aims to define a comprehensive overview of the application domain and the solution to be developed. At the conclusion of this phase, key outcomes include the identification of the primary users of the application, the delineation of the functionalities that need to be supported, and the identification of the principal non-functional requirements.

Conversely, requirements analysis focuses on reviewing and formalizing these requirements, resulting in a set of specifications that include:

- A list of user groups who will access the web application.
- The most significant use cases, which aim to illustrate the interaction between identified user groups and the application.
- A data dictionary outlining the data elements and their relationships.
- Non-functional requirements that outline performance, security, and usability criteria.
- Presentation guidelines that specify the graphical style of user interfaces to be developed.

These specifications serve as the blueprint for the development team, guiding them in creating an application that not only meets business objectives but also adheres to technical and organizational standards.

3.1 Functional Requirements

The functional requirements of an application define the specific functionality that the application must provide to meet user needs and achieve business objectives. Some of the functional requirements of the application are:

User Management:

- User registration and authentication (AROL company employees and customers).
- Role-based authorization to limit access to specific functionalities (admin or classic user).

QR Code Generation and Management:

- Creation of unique QR codes based on serial number, product ID, and scope.
- Association of specific QR codes with machinery and components to provide detailed information.

QR Code Scanning and Reading:

- QR code scanning functionality within the application.
- Display of information related to the scanned machinery or component.

Information Display:

- Display of usage, maintenance, and assembly instructions for machinery.
- Display batch information and other relevant details.

Interaction and Navigation:

- Intuitive navigation through a fixed navigation bar to access all sections of the application.
- Responsive user interface to adapt to different screen sizes (mobile, tablet, desktop).

Data Integration:

- Integration with existing company systems to retrieve machinery data and details.
- Real-time updating of machinery and component information through the application

3.2 Non-functional Requirements

Non-functional requirements are the criteria and constraints that define the qualities and characteristics of a software system, rather than describing specific functionality or behaviour directly related to the operations of the application. These requirements are essential to ensure that the system not only functions properly from a functionality standpoint, but also that it meets high standards for performance, security, usability, and other properties that affect the overall user experience and the robustness of the system itself. Here are some common categories of non-functional requirements:

- **Performance:** Specifies how the system should perform under certain conditions, such as response times, throughput, and resource usage limits.
- **Security:** Defines the security measures and protocols that must be implemented to protect data, prevent unauthorized access, and ensure compliance with regulations (e.g., GDPR, HIPAA).
- **Reliability:** Specifies the system's ability to maintain functionality over time and under varying conditions, including uptime requirements, fault tolerance, and disaster recovery.
- **Scalability:** Describes how well the system can handle increasing amounts of work or users, typically addressing issues like scalability of databases, servers, and network infrastructure.
- **Usability:** Defines the ease of use and intuitiveness of the system's user interface, ensuring that it meets the needs of its users with minimal training or support.
- **Maintainability:** Specifies how easily the system can be modified, updated, and extended over time, including considerations for code maintainability, documentation, and adherence to coding standards.
- **Compatibility:** Describes the system's ability to operate with other systems, platforms, or software products, including interoperability requirements and compatibility with legacy systems.

- **Availability**: Specifies the system's uptime requirements and the measures in place to minimize downtime, including redundancy, failover mechanisms, and backup procedures.
- **Performance**: Describes how the system should perform under specific conditions, such as response times, throughput, and resource usage limits.

3.3 UI/UX Design Principles

UI design focuses on the visual and interactive elements of the interface, including the arrangement of components, color schemes, typography, and overall layout. The primary objective of UI design is to create an interface that is both aesthetically pleasing and functionally intuitive, ensuring users can navigate the product with ease and confidence. Conversely, UX design encompasses the comprehensive experience a user has when interacting with a digital product. This includes the product's ease of use, efficiency, and the emotional responses it evokes. The goal of UX design is to create a product that is not only user-friendly and engaging but also meets the diverse needs of its users in an effective manner. Most web applications generate vast amounts of data from connected devices, necessitating intuitive and accessible interfaces for staff and technicians, who may not have advanced technical expertise. The UI must be designed to present data clearly and logically, facilitating quick comprehension and action. Similarly, UX design in industrial contexts should focus on minimizing cognitive load, offering clear, concise information and straightforward tools to enhance user productivity and accuracy. A well-executed UI/UX design can also contribute to improved safety in industrial environments by presenting critical information in an easily digestible format. This allows users to swiftly identify and respond to potential hazards or malfunctions in equipment, thereby reducing the likelihood of accidents and enhancing overall operational efficiency. By adhering to these principles, the UI and UX design of our application aims to deliver a seamless, efficient, and satisfying user experience, ultimately supporting the goals of both AROL and its end-users in managing and maintaining industrial machinery effectively.

3.3.1 Layout

The layout of an application is a crucial aspect that determines its usability and overall user experience. A well-designed layout provides a logical and intuitive structure, enabling users to navigate through the application seamlessly and efficiently. In developing the layout for our application, we have prioritized simplicity, clarity, and responsiveness to cater to a wide range of devices and user preferences.

The layout of our application is divided into three main sections:

1. **Fixed Navbar:** Positioned at the top of the screen, the fixed navbar allows users to navigate across the entire application effortlessly. This navbar remains visible as users scroll through different pages, ensuring consistent access to navigation options. When a user is logged in, the navbar is personalized to display their username, enhancing the user's connection to the application.
2. **App Viewport:** This central section dynamically displays the appropriate web page based on the user's navigation. Depending on the selected options from the navbar, the app viewport adjusts to show the relevant visual components and functionalities. This ensures that users always have the information and tools they need readily accessible, enhancing their overall experience and productivity.
3. **Footer Section:** Located at the bottom of the screen, the footer provides generic information and company contact details. This section ensures that important information is always within easy reach, without cluttering the main content area.

The entire layout is designed to be fully responsive, adapting seamlessly to various screen sizes. There is a distinct version for mobile devices, ensuring optimal usability on smaller screens, while tablet and desktop users share a unified layout. This approach guarantees a consistent and efficient user experience across all devices, allowing users to interact with the application effectively regardless of the platform they are using.

By integrating these design principles, our layout aims to provide a user-friendly, accessible, and visually appealing interface that meets the diverse needs of our users and supports the functionality of the application.

3.3.2 Design Patterns

Design patterns are established solutions to common problems encountered during software design and development. They provide reusable templates and guidelines that help developers structure their code in a scalable, maintainable, and efficient manner. By employing design patterns, developers can leverage proven techniques to address recurring challenges and promote best practices in software architecture.

Design patterns can be categorized into three main types:

- **Creational Patterns:** These patterns focus on object creation mechanisms, helping to create objects in a manner suitable for the situation.
- **Structural Patterns:** Structural patterns deal with the composition of classes and objects to form larger structures. They facilitate relationships between objects and classes to ensure flexibility and efficient use of resources.
- **Behavioural Patterns:** Behavioral patterns concentrate on the interaction and communication between objects. They define patterns of communication between objects and responsibilities.

These patterns are tailored to Angular's architecture and best practices, aligning with its component-based structure and dependency injection system. For this reason, the following were used:

- **Component Pattern:** Angular components encapsulate logic, UI, and behaviour into reusable units, promoting separation of concerns and modularity.
- **Service Pattern:** Services in Angular provide shared data and functionality across components, implementing the Singleton pattern to ensure a single instance throughout the application.
- **Dependency Injection Pattern**: Angular's dependency injection mechanism enables efficient management and injection of dependencies, promoting loose coupling and facilitating unit testing.
- **Observable Pattern:** Leveraging RxJS observables, Angular implements the Observable pattern for handling asynchronous data streams and event-driven communication between components.

3.4 Use Case Scenarios: Driving System Functionality through User Interactions

Use case analysis is a pivotal method of software development that focuses on capturing the functional requirements of a system from the perspective of its end users. Use cases provide a structured approach to defining interactions between users (actors) and the system to achieve specific goals. Each use case represents a discrete task or functionality that the system must perform to meet user needs.

Key Elements of Use Cases:

1. **Actor Identification**: Use cases identify the actors or users interacting with the system. Actors can be individuals, external systems, or other software applications that initiate actions within the system.
2. **Scenarios and Interactions**: Each use case details a specific scenario where an actor interacts with the system to achieve a desired outcome. These scenarios outline the steps involved, including inputs, actions performed by the system, and outputs or results.
3. **Goal Achievement**: Use cases focus on achieving specific goals or objectives that are valuable to the users or stakeholders. These goals drive the functionality and behavior of the system.
4. **Preconditions and Postconditions**: Preconditions specify the conditions that must be true before a use case can start, while postconditions describe the state of the system after the use case has been successfully completed.
5. **Alternative and Exception Paths**: Use cases also consider alternative paths and exceptions, detailing how the system should behave in different scenarios or when errors occur. This ensures robustness and resilience in handling unexpected situations.

Benefits of Use Case Analysis:

- **Clarity and Understanding**: Use cases provide a clear and understandable way to communicate system requirements to stakeholders, developers, and designers.
- **Scope Definition**: They help define the scope of the system by identifying all possible interactions and functionalities required.
- **Validation**: Use cases serve as a basis for validating the system's functionality against user expectations and business requirements.
- **Basis for Testing**: Use cases form the basis for testing, ensuring that all identified scenarios and interactions are thoroughly tested during the development process.

The diagram of use cases identified in this app is shown in Figure 11.

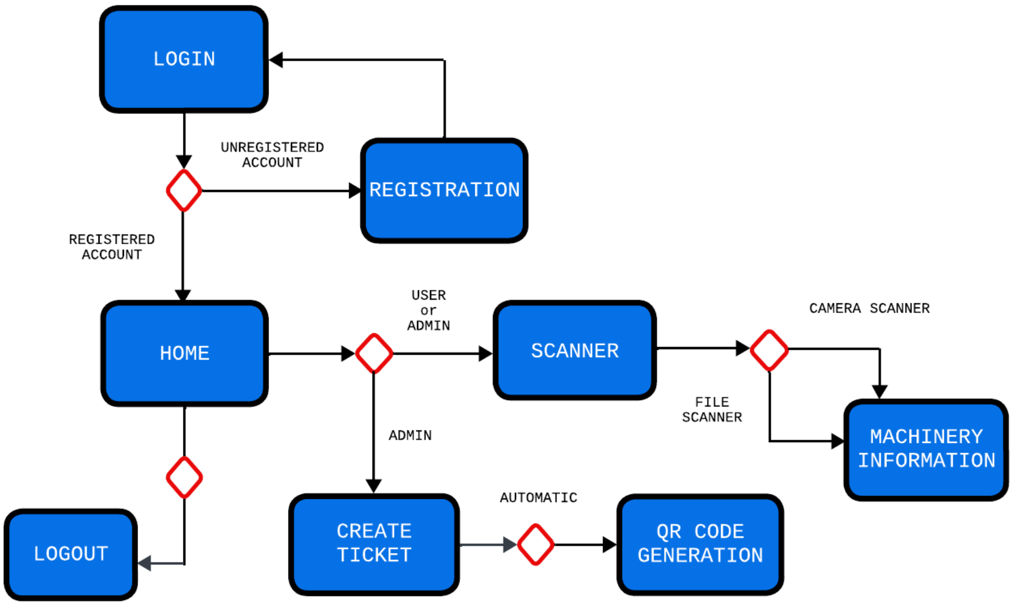


Figure 11: Use cases diagram.

3.4.1 Register Account

Actor: New User

Description: This use case describes the process by which a new user registers an account in the application by providing necessary personal and organizational information.

Preconditions:

- The user must have access to the application's registration page.
- The user should not already have an account registered with the same email address.

Postconditions:

- A new user account is created and stored in the system.
- The user is notified of a successful registration and may proceed to log in.

Main Flow:

1. The user navigates to the registration page of the application by clicking the Login button in the drop-down menu of the navbar.
2. The system displays a registration form requiring the following information:
    - First Name
    - Last Name
    - Nickname
    - Company
    - E-mail Address
    - Password
3. The user enters all of this information into the right fields.
4. The user submits the registration form by clicking the "Register" button.
5. The system validates the input data, ensuring all required fields are filled and the email format is correct.
6. The system checks if the email address is already associated with an existing account.

7. If validation passes and the email is unique, the system creates a new user account with the provided information.
8. The system displays a success message, indicating that the registration was successful.

Alternative Flows:

- **Missing Required Information**:

  If the user submits the form with missing required fields, the system highlights the missing fields and prompts the user to complete them.

- **Invalid E-mail Format**:

  If the email format is incorrect, the system displays an error message indicating that the email address is invalid and prompts the user to enter a valid email.

- **E-mail Already Registered**:

  If the email address is already in use, the system displays an error message indicating that the email is already associated with an existing account and prompts the user to use a different email address.

- **Password Strength**:

  If the password does not meet the required strength criteria (e.g., minimum length, complexity), the system prompts the user to create a stronger password.

3.4.2 Login

Actor: Registered User

Description: This use case describes the process by which a registered user logs into the application using their email address and password.

Preconditions:

- The user must already have a registered account in the system.

- The user must have access to the application's login page.

Postconditions:

- The user is successfully authenticated and granted access to the application.
- The user's session is started, allowing them to navigate through the application.

Main Flow:

1. The user navigates to the login page of the application by clicking the Login button in the drop-down menu of the navbar.
2. The system displays a login form requiring the following information:
    - E-mail Address/Nickname
    - Password
3. The user enters the information in the designated input fields.
4. The user submits the login form by clicking the "Login" button.
5. The system validates the input data, ensuring all required fields are filled and formatted correctly.
6. The system checks if the email address exists in the database.
7. If the email address exists, the system verifies that the password entered matches the stored password for that account.
8. If the credentials are valid, the system creates a user session.
9. The system redirects the user to the application's main page.
10. The system displays the user's name in the navbar to personalize the experience.

Alternative Flows:

- **Missing Required Information**:

  If the user submits the form with missing required fields, the system highlights the missing fields and prompts the user to complete them.

- **E-mail Not Registered**:

If the email address is not found in the database, the system displays an error message indicating that the email is not associated with any account and prompts the user to register or use a different email address.

- **Incorrect Password**:

  If the password entered does not match the stored password for the given email, the system displays an error message indicating that the password is incorrect and prompts the user to try again.

3.4.3 Logout

Actor: Authenticated User

Description: This use case describes the process by which an authenticated user logs out of the application, terminating their session and ensuring security and privacy.

Preconditions:

- The user must be logged into the application.
- The user has access to the logout functionality from any page within the application.

Postconditions:

- The user is successfully logged out.
- The user's session is terminated.
- The user is redirected to the login page.

Main Flow:

1. The user identifies the logout option, located in the navigation bar’s user menu.
2. The user selects the logout option.
3. The system prompts the user to confirm their intention to log out.
4. The user confirms the logout action.
5. The system terminates the user's session.

6. The system clears any session-related data from the client (local storage).
7. The system redirects the user to the login page.

Alternative Flows:

- **Cancel Logout Confirmation**:

  If the system prompts the user to confirm the logout action and the user cancels, the system aborts the logout process and returns the user to their previous state within the application.

3.4.4 Generate Machinery Ticket

Actor: Admin User

Description: This use case describes the process by which an admin user generates a unique ticket for a product, which includes creating a unique identifier (UID) that will be embedded in the URL of the QR code for that product.

Preconditions:

- The user must have admin privileges.
- The admin user must be logged into the application.
- The user has access to the create Ticket functionality from the ticket page.

Postconditions:

- A unique ticket (UID) is generated for the specified product.
- The UID is embedded in the URL for the product's QR code.
- The product information, including the UID, is stored in the system.

Main Flow:

1. The admin user navigates to the ticket generation page within the application by clicking the ticket navbar item.
2. The system displays a form requiring the following information:

        - Product ID
        - Scope
        - Serial Number
3. The admin user the information in the designated input fields.
4. The admin user submits the form by clicking the "Generate Ticket" button.
5. The system validates the input data, ensuring all required fields are filled and formatted correctly.
6. The system generates a unique UID based on the provided Product ID, Scope, and Serial Number.
7. The system stores the product information, including the UID in the database.
8. The system displays a success message, indicating that the ticket has been successfully generated.

Alternative Flows:

- **Missing Required Information**:

  If the admin user submits the form with missing required fields, the system highlights the missing fields and prompts the admin user to complete them.

- **Invalid Data Format**:

  If any of the input fields contain data in an invalid format, the system displays an error message indicating the issue and prompts the admin user to correct it.

3.4.5 Generate Machinery QR Code

Actor: Admin User

Description: This use case describes the process by which an admin user generates a QR code for a product after generating a unique ticket (UID). The QR code includes a URL with the UID embedded and can be customized in size and error correction level.

Preconditions:

- The user must have admin privileges.
- The admin user must be logged into the application.

- A unique ticket (UID) must have been generated for the product.

Postconditions:

- A QR code is generated with a URL that includes the unique UID.
- The QR code is available for download as a PNG image.

Main Flow:

1. The admin user navigates to the ticket generation page within the application.
2. The admin user generates a ticket by entering the Product ID, Scope, and Serial Number and submits the form to generate a unique UID.
3. The system generates a unique UID and displays it in a designated field on the same page.
4. The system displays a QR code generation form with the following fields:
    - UID (automatically populated with the generated UID)
    - Size (selectable options: 200, 300, 400, 500 pixels)
    - Error Correction Level (selectable options: Low, Medium, High, Very High)
5. The admin user selects the desired size for the QR code from the button group.
6. The admin user selects the desired error correction level from the button group.
7. The system automatically generates a QR code with the specified size and error correction level, embedding the URL that includes the UID.
8. The system provides a download button for the admin user to download the QR code as a PNG image.
9. The admin user downloads the QR code image.

Alternative Flows:

- **Default QR Code Generation:**
  If the admin user does not select the size and error correction level, the system automatically generates the QR code using default settings (200 pixels size and Low error correction level).

3.4.6 Scan QR Code

Actor: Any User

Description: This use case describes the process by which a user scans a QR code to retrieve the embedded URL using either their device's camera or by uploading an image/file. The upload method offers a significant advantage when dealing with QR codes that are too small to be effectively scanned by a camera.

Preconditions:

- The user must have access to the application and navigate to the scanning page.

Postconditions:

- The URL embedded in the QR code is displayed on the screen.
- The user can click on the URL to navigate to the respective webpage.

Main Flow 1: Scan Using Device Camera:

1. The user navigates to the scanning page using the specific nav item accessible from any situation within the application.
2. The system displays two distinct cards for selecting the scanning method.
3. The user selects the card for using the device's camera.
4. The system displays a new component where the user can select the device's camera from available options.
5. The user selects the preferred camera.
6. The user can also select the decoding mode for the QR code, with the default set to UTF-8.
7. The system displays buttons below the camera window to start the camera, stop the camera, and toggle the flash (if available).
8. The user clicks the button to start the camera.
9. The user aligns the QR code within the camera's view.

10. The system automatically decodes the QR code and displays the embedded URL below the camera component.
11. The URL is clickable, allowing the user to navigate to the respective webpage.

Main Flow 2: Scan Using Uploaded Image/File:

1. The user navigates to the scanning page using the specific nav item accessible from any situation within the application.
2. The system displays two distinct cards for selecting the scanning method.
3. The user selects the card for uploading an image/file.
4. The system displays a new component where the user can upload a file or photo.
5. The user uploads a file or photo containing a QR code.
6. The system automatically scans the uploaded image/file for a QR code.
7. If a QR code is detected, the system decodes it and displays the embedded URL below the component.
8. The URL is clickable, allowing the user to navigate to the respective webpage.

3.4.7 View Machine Detail

Actor: Any User

Description: This use case describes the process by which a user views the detailed information of a machine corresponding to a scanned QR code (UID). The amount of information displayed depends on the user's privileges, with more detailed data available to users with higher access levels.

Preconditions:

- The user must have access to the application.
- The user must scan a QR code, leading them to the machine details page.
- The system must store machine information associated with the UID embedded in the QR code.

• The system must recognize the user's access level.

Postconditions:

• The user views the machine details corresponding to the scanned QR code.

• The displayed information is tailored to the user's access level.

Main Flow:

1. The user scans the QR code, which redirects them to the machine details page.
2. The system extracts the UID from the QR code URL.
3. The system retrieves the machine information associated with the UID from the database.
4. The system identifies the user's access level.
5. The system displays the machine details in a table format, tailored to the user's privileges:
    - Basic information for all users (e.g., machine name, model, manufacturer, serial number).
    - Additional details for users with higher access levels (e.g., maintenance history, operational manuals, component details).
6. The user views the displayed machine details.

Alternative Flows:

- **User Not Logged In**:

  If the user is not logged in, the system prompts the user to log in to view detailed machine information. Basic information may still be displayed to unauthenticated users.

- **No Matching UID**:

  If the system does not find any machine information associated with the UID, it displays an error message indicating that the machine details could not be found and prompts the user to verify the QR code.

CHAPTER 4

4. Technological for the app’s implementation

In this chapter, the programming languages and development environments chosen for the application development will be discussed in detail, along with the advantages and reasons behind their selection.

4.1 Visual Studio Code

Visual Studio Code (VS Code), developed by Microsoft, is a powerful and versatile code editor that has gained widespread popularity among developers for its rich feature set, performance, and extensibility. Launched in 2015, VS Code has quickly become a go-to tool for coding across various platforms, including Windows, macOS, and Linux.

At the heart of VS Code's appeal is its lightweight and fast performance, even when handling large codebases. This efficiency is complemented by a minimalist yet highly customizable user interface, allowing developers to tailor their workspace with themes, extensions, and settings that suit their preferences and workflow. The editor supports a wide range of programming languages out-of-the-box, with syntax highlighting, intelligent code completion, and support for Git version control seamlessly integrated.

VS Code's built-in debugging capabilities streamline the process of identifying and resolving issues in code. Developers can set breakpoints, inspect variables, and step through code execution directly within the editor, facilitating faster debugging cycles and improving code quality. Integration with popular build systems and task runners further enhances productivity by automating repetitive tasks and workflows.

One of VS Code's standout features is its extensive marketplace of extensions, which allows developers to extend and customize the editor's functionality to meet specific project requirements. From language support and linters to productivity tools and themes, the marketplace offers thousands of extensions contributed by the community and third-party developers. This ecosystem fosters innovation and enables developers to enhance their development environment with tools that integrate seamlessly into their workflow.

VS Code's integration with cloud services and development environments further enhances its utility. It supports integration with Azure, GitHub, Bitbucket, Docker, and other cloud services, enabling seamless deployment, collaboration, and management of cloud-native applications. Additionally, VS Code's remote development extensions allow developers to work on projects hosted on remote servers or containers, providing flexibility and scalability in development environments.

Accessibility is another key strength of VS Code, with features designed to support developers of all skill levels. The editor's intuitive interface, extensive documentation, and community support forums facilitate learning and troubleshooting, making it an ideal choice for both beginners and experienced developers alike. Regular updates and enhancements ensure that VS Code remains at the forefront of modern development tools, adapting to evolving technologies and industry practices.

4.2 Atlassian Bitbucket

Bitbucket, developed and maintained by Atlassian, is a robust platform designed for version control and collaboration in software development. It supports both Git and Mercurial repositories, offering flexible options for teams to manage their codebase securely and efficiently. Bitbucket is available in both cloud-based and self-hosted versions, catering to the diverse needs of organizations across different scales and industries.

At its core, Bitbucket facilitates centralized management of code repositories, allowing teams to store, review, and track changes to their software projects. It provides a user-friendly web interface that simplifies the process of creating repositories, managing branches, and collaborating on code with team members. This ease of use extends to features such as pull requests, which enable developers to propose changes, request reviews, and merge code seamlessly after approvals, ensuring code quality and team collaboration.

A standout feature of Bitbucket is its integration with other Atlassian products, most notably Jira. This integration enables seamless project management by linking code changes directly to Jira issues, providing traceability from development tasks to business objectives. Teams can associate commits, branches, and pull requests with specific Jira tickets, facilitating better communication and alignment between development teams and project stakeholders.

Bitbucket offers robust access control mechanisms, allowing administrators to define fine-grained permissions for repositories and branches. This ensures that sensitive code remains secure while enabling collaboration within the team and with external contributors. Additionally, Bitbucket supports integrations with various CI/CD tools, including Bitbucket Pipelines, Jenkins, and Bamboo, enabling teams to automate build, test, and deployment processes. This integration fosters a continuous delivery pipeline, enhancing development speed, reliability, and overall team productivity.

For organizations adopting Agile and DevOps practices, Bitbucket provides features such as code insights and code search capabilities, empowering teams to analyze code quality metrics and perform efficient code reviews. These tools help teams identify and address technical debt, improve code maintainability, and uphold development best practices.

4.3 Atlassian Jira

Jira, developed by Atlassian, is a powerful project management tool widely recognized for its versatility and scalability across various industries and team sizes. At its core, Jira serves as a robust platform for Agile project management, enabling teams to plan, track, and manage their software development projects efficiently.

One of Jira's key strengths lies in its flexibility to support different Agile methodologies, including Scrum, Kanban, and hybrid approaches. It allows teams to create customizable workflows that mirror their development processes, facilitating transparency and alignment among team members and stakeholders. Jira's intuitive interface and extensive customization options make it adaptable to diverse project needs, from software development and IT operations to marketing campaigns and business projects.

Central to Jira's functionality are its issue-tracking and management capabilities. Teams can create and prioritize tasks, bugs, user stories, and epics within Jira, assigning them to team members and tracking their progress through various stages of development. Each issue is equipped with customizable fields, attachments, comments, and status updates, providing a comprehensive view of project status and enabling effective collaboration.

Jira's integration with other Atlassian tools, such as Bitbucket, Confluence, and Trello, enhances its utility by connecting development, documentation, and project management seamlessly. For instance, integration with Bitbucket allows developers to link code changes directly to Jira issues, providing traceability from code commits to project tasks. Similarly, integration with Confluence facilitates knowledge sharing and documentation within the project team, ensuring that all stakeholders have access to up-to-date information.

Jira supports advanced reporting and analytics capabilities, empowering teams to gain insights into project progress, team performance, and resource allocation. Built-in dashboards and reports offer real-time visibility into key metrics, enabling stakeholders to make data-driven decisions and identify areas for improvement. Customizable dashboards allow team members to create personalized views tailored to their specific roles and responsibilities, promoting transparency and accountability.

Furthermore, Jira's ecosystem of add-ons and integrations extends its functionality beyond traditional project management. Teams can leverage marketplace apps to enhance Jira with additional features such as time tracking, test management, release planning, and automation. These add-ons cater to specialized needs across different industries, enabling teams to tailor Jira to their unique project requirements and workflows.

4.4 HTML5

HTML (HyperText Markup Language) is the cornerstone of web development, serving as the standard markup language used to create and structure web pages and web applications. Developed by Tim Berners-Lee in the early 1990s, HTML forms the foundation upon which content on the World Wide Web is built. At its essence, HTML is a markup language that utilizes tags to define the structure and content of web documents. Tags are enclosed in angle brackets (< >) and typically come in pairs: an opening tag and a closing tag, with content nested between them. The semantic nature of HTML plays a crucial role in how content is interpreted and presented by web browsers. Semantic tags provide meaningful structure to web pages, enhancing accessibility and search engine optimization (SEO) by clearly defining the purpose and relationship of different parts of the content. Each tag specifies a different role for the content it marks, describing characteristics such as a function, color, size, and relative position within the page. When a hypertext document written in HTML is stored in a file, its extension is typically .html. Let us consider the modes of operation that lead to the actual display of data on the client. In general, the user makes a request to the server for a certain HTML page, which is sent. Once received by the client, this is processed by the user's browser. The result of the processing represents the on-screen display of the requested page.

An HTML document begins with the DOC-TYPE prologue definition, which informs the browser of the HTML specification URL used for the document. It then serves to indicate, in implicit form, the elements, attributes, and entities that may be used. It also specifies which version of HTML is being referred to. The definition of the prologue is followed by the structure that delimits the remainder of the document, which is between the tags <html> and </html>. Within this structure, there is a header, delimited by the tags <head> and </head>, followed by the body, delimited by the tags <body> and </body>. The header contains mainly control information not normally displayed by the browser, while the body contains all those elements displayed on the user's screen (text, images, links, etc.).

HTML5, the latest version of HTML standardized in 2014, introduced significant improvements and new features to meet the demands of modern web development. It includes support for multimedia elements with <audio> and <video> tags, a canvas for dynamic graphics rendering, local storage capabilities for offline web applications using the Web Storage API, and semantic tags to better describe document structure and improve accessibility. The evolution of HTML has been driven by the need to support richer, more interactive web experiences. HTML5's adoption of new APIs such as Geolocation, Web Workers for multi-threaded JavaScript execution, and WebRTC for real-time communication further expands the possibilities for web developers. HTML works in conjunction with Cascading Style Sheets (CSS) for styling and JavaScript (JS) for dynamic behavior, forming the foundation of front-end web development. Together, these technologies enable developers to create visually appealing, responsive, and interactive websites that adapt to different devices and user interactions.

4.5 CSS

Cascading Style Sheets (CSS) is a fundamental technology in web development that allows developers to style and format web pages, enhancing their appearance and presentation. Developed as a complement to HTML, CSS separates the structure and content of a web page from its visual design, enabling greater flexibility and control over the look and feel of web content. CSS works by specifying style rules that define how HTML elements should be displayed in different contexts. These style rules consist of selectors and declarations: selectors target specific HTML elements, while declarations define the appearance of those elements. Declarations include properties (e.g., color, font size, margin) and their corresponding values (e.g., blue, 16px, 10px auto). The power of CSS lies in its ability to apply styles consistent across multiple web pages by defining them in external style sheets (.css files) or embedding them directly within HTML documents using the <style> tag. External style sheets promote modularity and maintainability, allowing developers to update styles across an entire website by modifying a single file. CSS offers a wide range of features and capabilities to control the layout, typography, colors, and visual effects of web content. Flexbox and CSS Grid Layout are two layout models introduced in recent years that revolutionized how developers create responsive and flexible web layouts. Flexbox simplifies the alignment and distribution of elements within a container, while CSS Grid Layout enables precise control over the positioning and sizing of elements in a grid-based layout. The evolution of CSS, particularly with CSS3 and beyond, has introduced numerous enhancements and new features to address the complexities and demands of modern web design. CSS3 introduces advanced selectors (e.g., pseudo-classes and pseudo-elements), media queries for responsive design, transitions, and animations for adding interactivity, and custom properties (CSS variables) for dynamic styling. Furthermore, CSS frameworks such as Bootstrap, Foundation, and Tailwind CSS offer pre-designed components, grids, and utilities that streamline development and ensure consistency across projects. These frameworks provide a starting point for creating responsive and aesthetically pleasing websites, allowing developers to focus more on functionality and less on repetitive styling tasks.

4.5.1 Flexbox and CSS Grid Layout

Flexbox and CSS Grid Layout are two powerful layout models introduced in CSS that revolutionized how developers create responsive and complex web layouts.

Flexbox, short for Flexible Box Layout, is designed for one-dimensional layouts, either in a row or a column. It allows for efficient distribution of space between items in a container, even when their sizes are unknown or dynamic. Flexbox offers a flexible and intuitive way to align, stack, and reorder elements within a container, making it ideal for components like navigation menus, card layouts, and form controls. Key features of Flexbox include its ability to control the alignment, order, and sizing of items with properties like justify-content, align-items, flex-direction, and flex-grow.

On the other hand, CSS Grid Layout introduces a two-dimensional grid-based system, allowing developers to create complex layouts with rows and columns. It provides precise control over the placement and sizing of items within the grid, enabling designs that were previously difficult to achieve with traditional CSS. CSS Grid Layout is particularly useful for creating multi-column layouts, magazine-style designs, and responsive grids that adapt to different screen sizes seamlessly. Developers can define grid templates, set grid gaps, and place items explicitly using properties like grid-template-columns, grid-template-rows, grid-gap, and grid-area.

Both Flexbox and CSS Grid Layout complement each other and can be used together within the same project. Flexbox is often used for arranging items within a single row or column, while CSS Grid Layout excels in creating more complex and structured layouts that require precise control over both rows and columns. Their combination allows developers to achieve highly responsive and visually appealing designs while maintaining clean and maintainable code.

4.5.2 Bootstrap

Bootstrap is a popular front-end framework that facilitates the development of responsive and mobile-first websites and web applications. Originally developed by Twitter, Bootstrap is now maintained by an open-source community of developers and contributors. It is widely used for its comprehensive set of ready-to-use components, CSS and JavaScript utilities, and responsive design principles. One of Bootstrap's key features is its grid system, which allows developers to create complex layouts using a responsive, 12-column grid. This grid system simplifies the arrangement of content by providing predefined classes (such as col-sm-6, col-md-4, etc.) that adjust automatically based on the screen size, making it easy to create layouts that look good on various devices and screen resolutions. Bootstrap also includes a vast library of UI components such as buttons, forms, navigation bars, modals, carousels, and more. These components are styled using Bootstrap's CSS classes, ensuring consistency and adherence to modern design principles like Material Design. Developers can quickly integrate these components into their projects, saving time and effort in UI development. Another significant aspect of Bootstrap is its extensive documentation and community support. The official Bootstrap documentation provides comprehensive guides, examples, and API references, making it easy for developers to get started and leverage its features effectively. The active Bootstrap community contributes to plugins, themes, and extensions, further extending its functionality and customization options. Bootstrap's versatility extends beyond traditional web development to support responsive web design, which ensures that websites adapt seamlessly to different devices and screen sizes. By incorporating Bootstrap into their workflow, developers can streamline development, improve productivity, and deliver visually appealing and functional web experiences.

4.5.3 CSS Media Queries

Media queries in CSS provide a mechanism for applying different styles based on various device characteristics such as screen width, device orientation, resolution, and more. They enable responsive web design by allowing developers to tailor the layout and appearance of web pages to different devices and screen sizes. The syntax of a media query typically begins with the @media rule followed by a media type (such as screen or print) and one or more media features enclosed in parentheses. For example, @media screen and (max-width: 768px) apply styles only when the viewport width is 768 pixels or less. Media queries allow developers to create adaptive and flexible designs that adjust seamlessly across different devices, from smartphones and tablets to desktops and large screens. By utilizing media queries, CSS can deliver optimized user experiences by ensuring that content is readable, elements are appropriately sized, and layouts are well-structured regardless of the device being used. Common use cases for media queries include defining breakpoints for responsive design, adjusting typography and layout based on screen size, optimizing images for different resolutions, and hiding or displaying elements selected based on device capabilities. Media queries are a fundamental tool in modern web development for creating designs that are not only visually attractive but also functional and accessible across a diverse range of devices and contexts.

4.6 Angular Framework

Angular is a powerful and widely used open-source web application framework maintained by Google. It is designed for building dynamic, single-page web applications (SPAs) and robust enterprise-scale applications. Since its initial release in 2010 as AngularJS, the framework has undergone significant evolution, with Angular 2+ (rebranded simply as Angular) marking a complete rewrite to address the limitations of its predecessor. Angular leverages TypeScript, a superset of JavaScript, which introduces strong typing, object-oriented programming concepts, and other advanced features to JavaScript development. This integration enhances code quality, maintainability, and scalability by enabling developers to catch errors at compile-time rather than runtime, thanks to TypeScript's static typing and tooling support. The core principles of Angular revolve around modularity, component-based architecture, and reusability. Applications built with Angular are structured as a collection of components, each encapsulating its own logic, template (HTML), and styles (CSS/SCSS). Components are reusable, composable building blocks that promote the separation of concerns and facilitate collaboration among developers working on different parts of an application. Angular's architecture is centered around dependency injection (DI), a design pattern that facilitates the development of loosely coupled components. DI promotes code reuse, testability, and maintainability by allowing components to be easily substituted or mocked during unit testing. This architectural approach also supports modular development and scalability, making it ideal for large-scale applications with complex requirements. The latest version of Angular, released in November 2023, is Angular 17. Angular releases new versions approximately every six months, with each release introducing enhancements, performance improvements, bug fixes, and new features to the framework. The Angular team's commitment to regular updates ensures that developers can leverage the latest advancements in web development and maintain compatibility with evolving web standards. Angular's feature set includes powerful tools and libraries such as Angular CLI (Command Line Interface) for scaffolding projects, managing dependencies, and optimizing builds; RxJS (Reactive Extensions for JavaScript) for handling asynchronous operations and managing data streams; and Angular Material, a UI component library that provides pre-designed and customizable components adhering to Material Design principles. Key features introduced in recent versions of Angular include differential loading for faster startup times and improved performance, strict mode to enforce stricter TypeScript checks and optimize bundle size, improved support for web components and server-side rendering (SSR), and enhanced support for progressive web applications (PWAs) and native mobile development through tools like Capacitor and Ionic. Angular continues to be a preferred choice for developers and enterprises due to its comprehensive ecosystem, strong community support, and robust documentation. It empowers developers to build scalable, maintainable, and performant web applications while adhering to best practices in software engineering. As Angular evolves, it remains at the forefront of modern web development frameworks, driving innovation and shaping the future of web applications.

4.6.1 Javascript/Typescript languages

JavaScript and TypeScript are both widely used programming languages in web development, each with its own strengths and purposes.

JavaScript is a dynamic, high-level programming language primarily used for creating interactive and dynamic web pages. It's supported by all modern web browsers and is the cornerstone of web development alongside HTML and CSS. JavaScript is known for its versatility and runs on the client-side (browser) and server-side (Node.js), allowing developers to build full-stack applications.

JavaScript's key features include:

1. **Dynamic Typing**: JavaScript is dynamically typed, meaning variable types are determined at runtime. This flexibility makes it easy to write and modify code quickly.
2. **Prototype-based Object Orientation**: JavaScript uses prototypes rather than classes for object-oriented programming. Objects can inherit the properties and methods directly from other objects.

3. **Event-driven and Asynchronous Programming**: JavaScript supports event-driven programming, crucial for handling user interactions and asynchronous operations like fetching data from servers without blocking the main thread.
4. **Extensive Ecosystem**: JavaScript has a vast ecosystem of libraries and frameworks (e.g., React, Vue.js) that simplify and accelerate web development tasks.

On the other hand, **TypeScript** is a superset of JavaScript developed by Microsoft. It adds optional static typing and other advanced features to JavaScript, addressing some of its drawbacks in large-scale applications:

1. **Static Typing**: TypeScript introduces static typing, allowing developers to define types for variables, function parameters, and return values. This helps catch errors during development and improves code quality and maintainability.
2. **Enhanced IDE Support**: TypeScript enhances developer productivity with features like code navigation, intelligent code completion, and refactoring tools. IDEs and editors can provide better insights and suggestions thanks to TypeScript's type annotations.
3. **Compatibility with JavaScript**: TypeScript is backward-compatible with JavaScript. Existing JavaScript codebases can be gradually migrated to TypeScript, allowing teams to adopt TypeScript at their own pace.
4. **Tooling and Community**: TypeScript benefits from strong tooling support, including TypeScript compiler (tsc), TypeScript-aware editors (e.g., Visual Studio Code), and a growing community that contributes to its development and ecosystem.

In summary, while JavaScript remains the essential language for web development, TypeScript enhances it with static typing and advanced tooling, making it suitable for large-scale projects where type safety, maintainability, and developer productivity are crucial. Developers can choose between JavaScript and TypeScript based on project requirements, team expertise, and desired level of type safety and tooling support.

4.6.2 Advantages

Using Angular with TypeScript provides numerous significant advantages for developing modern web applications. Here are some of the key benefits of using Angular and TypeScript together:

1. **Type Safety and Error Checking**: TypeScript introduces a static typing system that allows developers to define types for variables, function parameters, object properties, and more. This helps detect errors during development before the code is executed. With static typing, common errors such as typos, type conversion errors, and logic errors can be identified more quickly compared to plain JavaScript, thereby improving application robustness and reliability.
2. **Enhanced Object-Oriented Capabilities**: TypeScript supports a more robust object-oriented programming approach compared to JavaScript, with support for classes, interfaces, inheritance, and more. This makes the code more structured, organized, and easier to understand and maintain, especially for complex and long-term projects.
3. **Improved IDE Support**: IDEs and code editors like Visual Studio Code provide advanced support for TypeScript, including intelligent suggestions, code auto-completion, quick navigation, and refactoring tools. This significantly enhances developer productivity by reducing the time needed to find errors, write new code, and navigate within the project.
4. **Modularity and Structure**: Angular promotes a modular architecture through its NgModules, allowing developers to organize code into functional and independent units. TypeScript facilitates the creation of reusable and well-structured modules due to its object-oriented nature and capabilities for import/export.
5. **Testability and Maintainability**: The combined use of Angular and TypeScript makes code more easily testable thanks to type clarity and modular organization. This facilitates the writing of effective automated tests that verify application behavior without depending on implementation complexities.

6. **Community and Support**: Both Angular and TypeScript benefit from a large and active community of developers, contributing resources, tools, libraries, and solutions to common problems. Detailed documentation, tutorials, and online resources are widely available, facilitating learning and issue resolution during development.

In conclusion, the combined use of Angular and TypeScript offers significant advantages for developing modern web applications, enhancing code quality, developer productivity, and project maintainability. This combination is particularly suitable for complex and large-scale projects where type safety, modular structure, and collaborative development capabilities are essential for project success.

CHAPTER 5

In this chapter, we will analyze the implementation of this application by describing the elements that make up Angular and providing code examples for clarity. Specifically, we will focus on Angular 17, the version used for this project. We will also briefly discuss how the backend has been managed. Finally, there will be a demo, illustrated with screenshots, to demonstrate the application's functionality.

5. Angular side implementation

Thanks to the advancements in Angular 17, it is possible to create highly modular applications. Applications, like this one, consist of various standalone components, eliminating the need for traditional NgModules. The main component now serves the role of bootstrapping the application, simplifying the startup process. Every component in Angular 17 integrates two fundamental elements: the Component itself, which handles the interaction between application logic and the user interface, and the Template, which defines the visual structure presented to the user. The communication with the web server is managed through Services, another critical element of Angular applications. These Services facilitate interactions with databases and external APIs, ensuring seamless data flow and operations. This modular architecture, enhanced with standalone components, allows for more efficient and maintainable code, paving the way for scalable and high-performance applications.

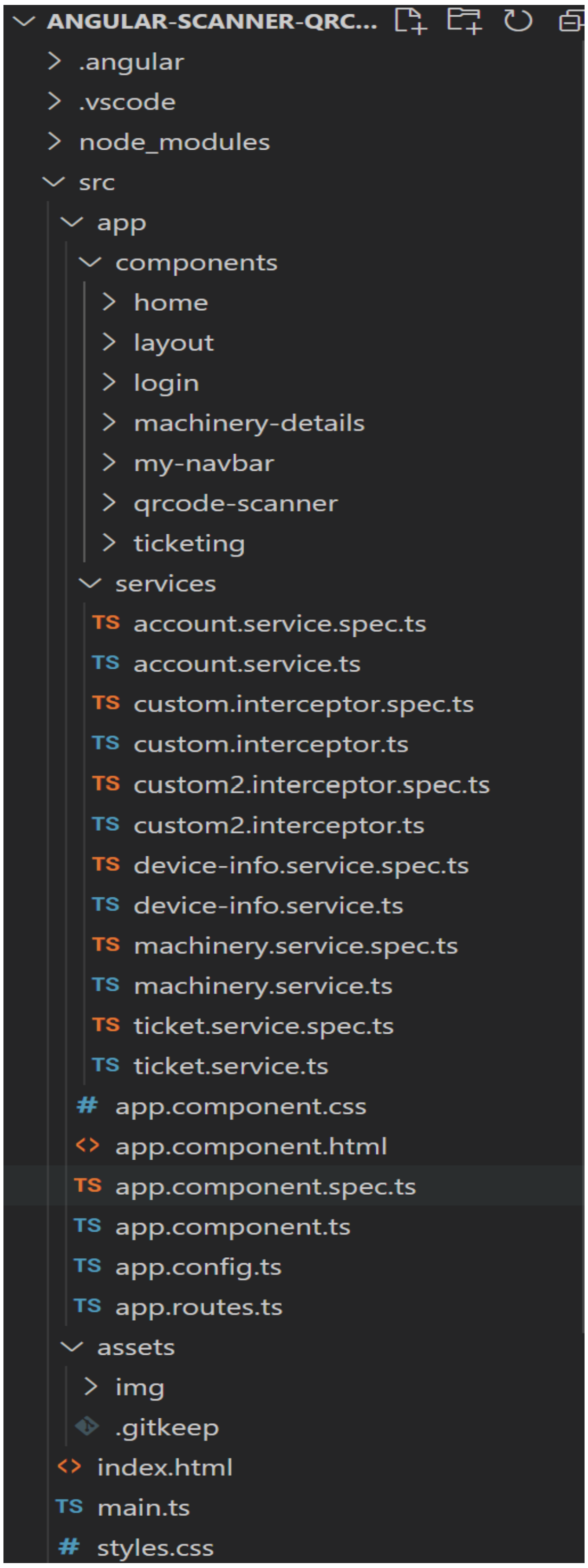
5.1Structure of an Angular Project

Here's a typical structure for an Angular 17 project utilizing standalone components:

- **Root Directory**:
  - **node_modules/**: Contains project dependencies managed by npm.
  - **src/**: Main directory for the application's source code.
  - **angular.json**: Angular CLI configuration file.

- **package.json**: npm configuration file listing dependencies and scripts.
- **tsconfig.json**: TypeScript compiler configuration file.

- **src/ Directory**:
  - **app/**: Contains the main application code.
  - **standalone/**: Directory for standalone components.
  - **example-standalone.component.ts**: Example standalone component.
  - **example-standalone.component.html**: Template for the standalone component.
  - **example-standalone.component.css**: Styles for the standalone component.
  - **example-standalone.component.spec.ts**: Test specification file for the component.
  - **app.component.ts**: It defines the root component.
  - **app.component.html**: Defines the root component's template.
  - **app.component.css**: Style for the root component.
  - **app.component.spec.ts**: This is a test specification file for the component.
  - **app.config.ts**: File to manage configuration settings for the application.
  - **app.routes.ts**: Defines the application's routes.
  - **assets/**: Static assets like images and JSON files.
  - **environments/**: Environment configuration files.
  - **environment.prod.ts**: Production environment configuration.
  - **environment.ts**: Development environment configuration.
  - **main.ts**: Main application bootstrap file.
  - **index.html**: Main HTML file hosting the Angular application.
  - **styles.css**: Global CSS styles.
  - **polyfills.ts**: Configuration file for polyfills.



```
ANGULAR-SCANNER-QRC...
  .angular
  .vscode
  node_modules
  src
    app
      components
        home
        layout
        login
        machinery-details
        my-navbar
        qrcode-scanner
        ticketing
      services
        account.service.spec.ts
        account.service.ts
        custom.interceptor.spec.ts
        custom.interceptor.ts
        custom2.interceptor.spec.ts
        custom2.interceptor.ts
        device-info.service.spec.ts
        device-info.service.ts
        machinery.service.spec.ts
        machinery.service.ts
        ticket.service.spec.ts
        ticket.service.ts
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.config.ts
      app.routes.ts
    assets
      img
      .gitkeep
    index.html
    main.ts
    styles.css
```

Figure 12: Project structure.

5.2 Components

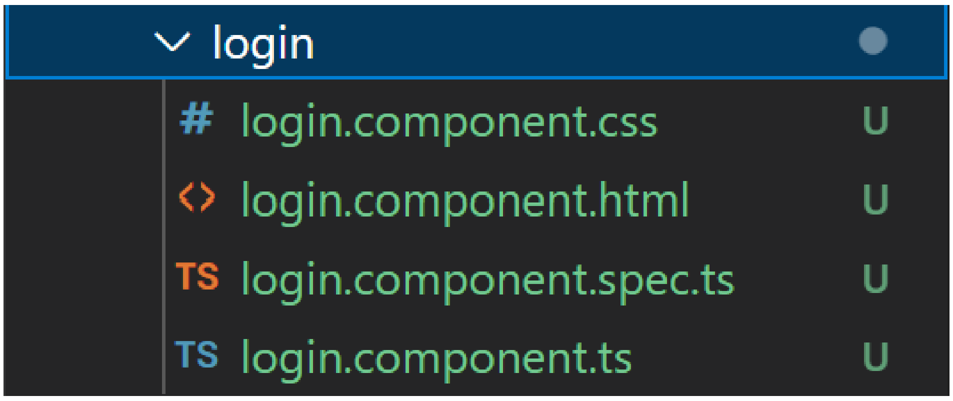
Components in Angular are the fundamental building blocks for constructing modular and reusable user interfaces within web applications. Each component consists of a blend of HTML elements, CSS styles, and JavaScript (or TypeScript) logic that work together to define a specific part of the application's user interface. Components are declared using the @Component decorator, which is a key feature in Angular's component-based architecture. This decorator defines the component's selector, template, and styles, linking them together within a single cohesive unit. A typical Angular component consists of the following parts:

1. **Class**: Defines the component's properties and methods, containing the logic for the component.
2. **Template**: Defines the HTML view for the component.
3. **Styles**: Defines the CSS styles for the component.
4. **Metadata**: Decorates the class with additional information using the @Component decorator.

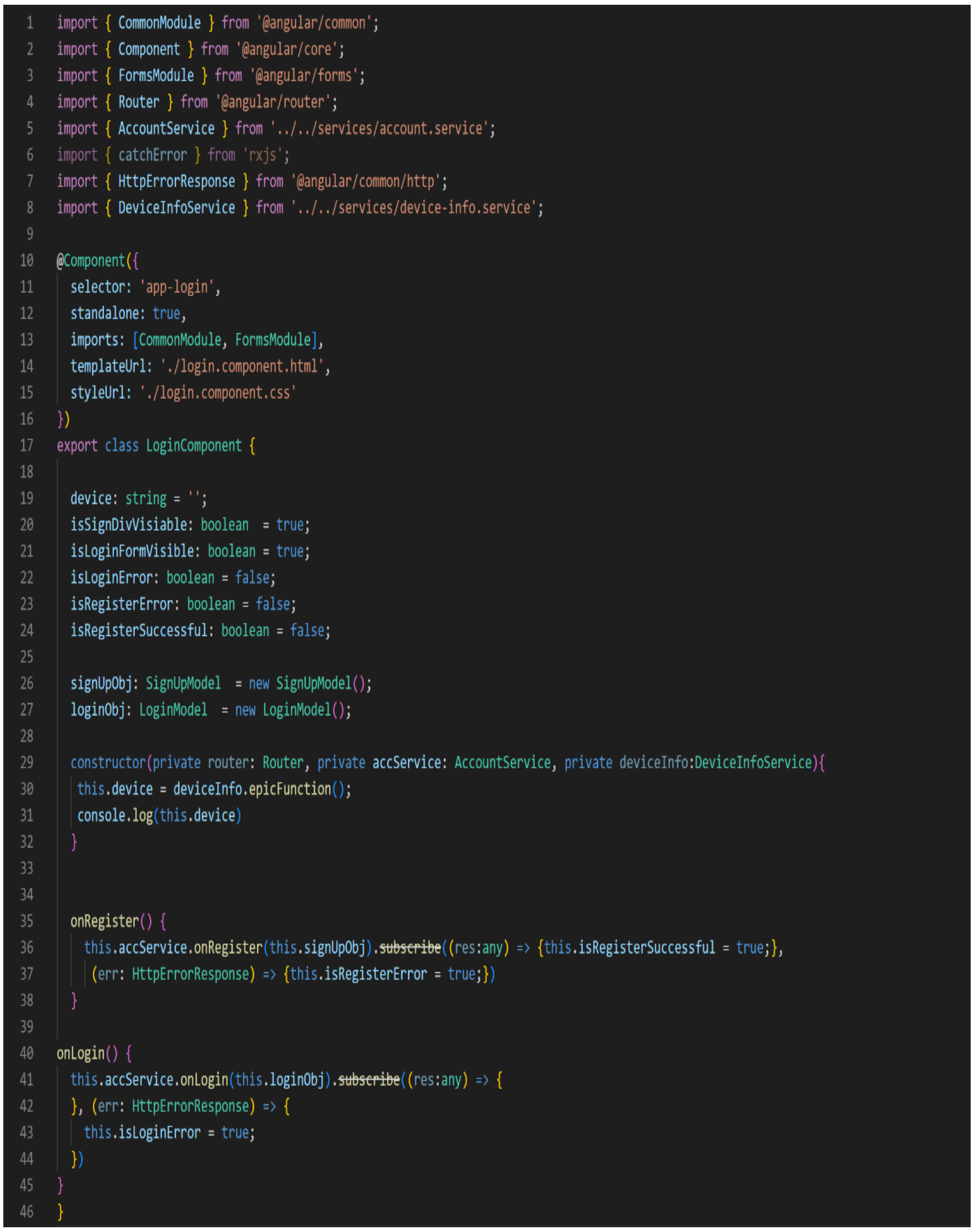
With Angular 17, a significant enhancement introduced is the concept of standalone components. This new feature simplifies the component model by allowing components to be created without requiring them to be part of an Angular module (NgModule). This can make the application more modular and reduce boilerplate code.

Key Properties of Standalone Components:

1. **Standalone Property**: Standalone components are defined by setting the standalone property to true in the @Component decorator.
2. **Imports**: Instead of relying on module imports, standalone components directly import the necessary dependencies.
3. **Simplified Structure**: This reduces the need to manage multiple modules, making the codebase cleaner and easier to maintain.



```
login
  login.component.css          U
  login.component.html         U
  login.component.spec.ts      U
  login.component.ts           U
```



```typescript
import { CommonModule } from '@angular/common';
import { Component } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { Router } from '@angular/router';
import { AccountService } from '../../services/account.service';
import { catchError } from 'rxjs';
import { HttpErrorResponse } from '@angular/common/http';
import { DeviceInfoService } from '../../services/device-info.service';

@Component({
  selector: 'app-login',
  standalone: true,
  imports: [CommonModule, FormsModule],
  templateUrl: './login.component.html',
  styleUrl: './login.component.css'
})
export class LoginComponent {

  device: string = '';
  isSignDivVisiable: boolean  = true;
  isLoginFormVisible: boolean = true;
  isLoginError: boolean = false;
  isRegisterError: boolean = false;
  isRegisterSuccessful: boolean = false;

  signUpObj: SignUpModel  = new SignUpModel();
  loginObj: LoginModel  = new LoginModel();

  constructor(private router: Router, private accService: AccountService, private deviceInfo:DeviceInfoService){
    this.device = deviceInfo.epicFunction();
    console.log(this.device)
  }


  onRegister() {
    this.accService.onRegister(this.signUpObj).~~subscribe~~((res:any) => {this.isRegisterSuccessful = true;},
     (err: HttpErrorResponse) => {this.isRegisterError = true;})
  }

onLogin() {
  this.accService.onLogin(this.loginObj).~~subscribe~~((res:any) => {
  }, (err: HttpErrorResponse) => {
    this.isLoginError = true;
  })
}
}
```

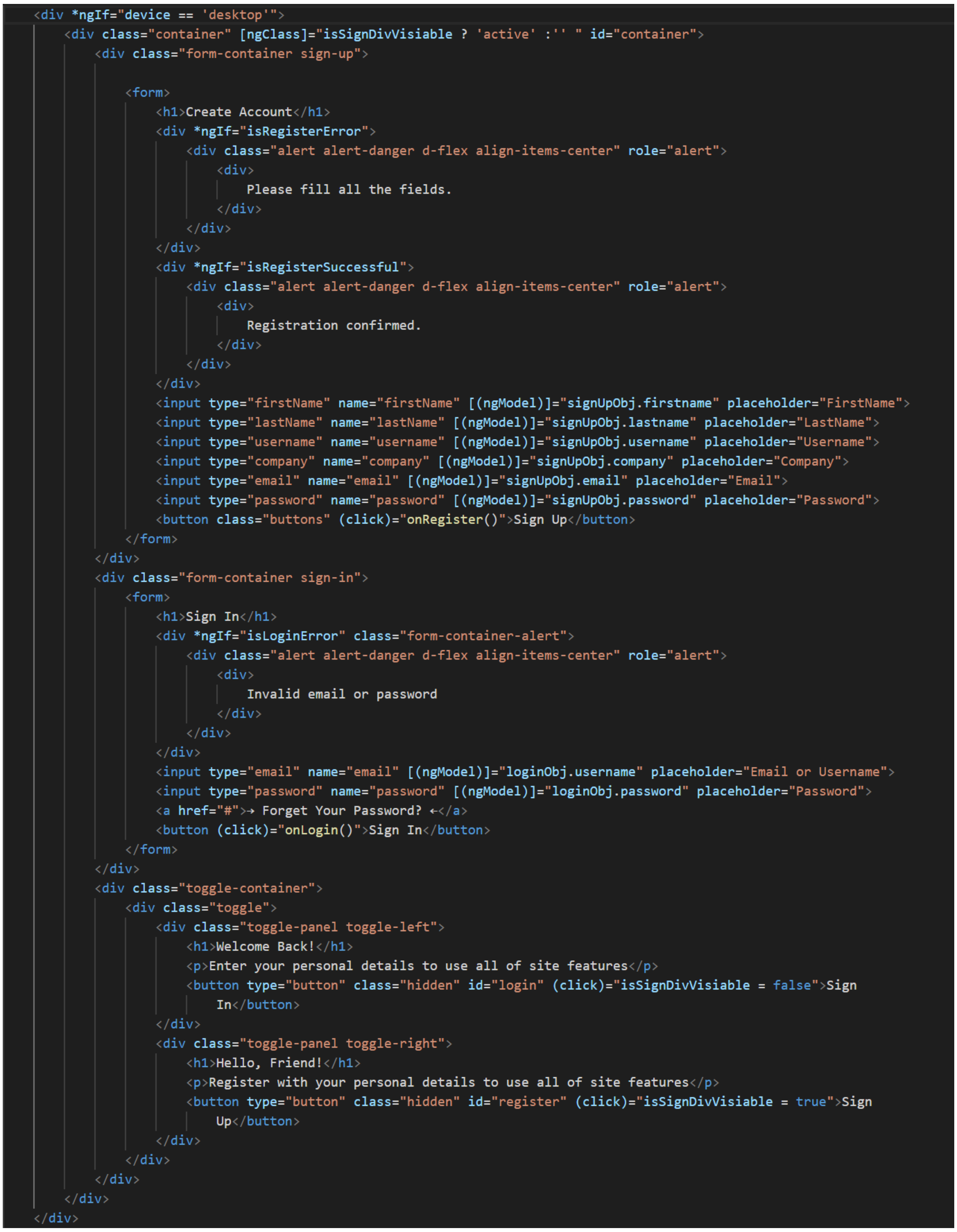
Figure 14: Login.component.ts screenshot.

5.3 Template

Angular templates are a core feature of the framework, designed to define the user interface and dictate how data is displayed and interacted with. They are written in HTML and use Angular's declarative syntax to seamlessly bind data from the component class to the view. This data binding can be accomplished through several mechanisms, including interpolation, property binding, event binding, and directives. Here are their various functionalities:

- **Interpolation**: Allows for dynamic data display by embedding component properties directly into the HTML.
- **Property Binding**: Enables synchronization between component properties and HTML element attributes, ensuring that changes in the component are reflected in the view.
- **Event Binding**: Handles user interactions, such as clicks and key presses, by linking component methods to HTML events.
- **Directives**:
    - **Structural Directives**: Control the rendering of elements based on conditions (*ngIf) or iterating over data collections (*ngFor).
    - **Attribute Directives**: Modify the behavior or appearance of an element without changing its structure.
- **Two-way Data Binding**: Synchronizes data between the model and the view, ensuring changes in the user interface are immediately reflected in the component and vice versa.
- **Local References and Template Variables**: Allow access to DOM elements directly within the template, enhancing interactivity.
- **Pipes**: Transform data directly in the template, providing a convenient way to format and display data.

By linking closely with components, Angular templates enable a clear separation of concerns. The component class contains the business logic and state, while the template defines how this data is presented and interacted with. This architecture facilitates the development of dynamic, maintainable, and scalable web applications, ensuring that the user interface is both responsive and data-driven.



```html
<div *ngIf="device == 'desktop'">
    <div class="container" [ngClass]="isSignDivVisiable ? 'active' :'' " id="container">
        <div class="form-container sign-up">

            <form>
                <h1>Create Account</h1>
                <div *ngIf="isRegisterError">
                    <div class="alert alert-danger d-flex align-items-center" role="alert">
                        <div>
                            Please fill all the fields.
                        </div>
                    </div>
                </div>
                <div *ngIf="isRegisterSuccessful">
                    <div class="alert alert-danger d-flex align-items-center" role="alert">
                        <div>
                            Registration confirmed.
                        </div>
                    </div>
                </div>
                <input type="firstName" name="firstName" [(ngModel)]="signUpObj.firstname" placeholder="FirstName">
                <input type="lastName" name="lastName" [(ngModel)]="signUpObj.lastname" placeholder="LastName">
                <input type="username" name="username" [(ngModel)]="signUpObj.username" placeholder="Username">
                <input type="company" name="company" [(ngModel)]="signUpObj.company" placeholder="Company">
                <input type="email" name="email" [(ngModel)]="signUpObj.email" placeholder="Email">
                <input type="password" name="password" [(ngModel)]="signUpObj.password" placeholder="Password">
                <button class="buttons" (click)="onRegister()">Sign Up</button>
            </form>
        </div>
        <div class="form-container sign-in">
            <form>
                <h1>Sign In</h1>
                <div *ngIf="isLoginError" class="form-container-alert">
                    <div class="alert alert-danger d-flex align-items-center" role="alert">
                        <div>
                            Invalid email or password
                        </div>
                    </div>
                </div>
                <input type="email" name="email" [(ngModel)]="loginObj.username" placeholder="Email or Username">
                <input type="password" name="password" [(ngModel)]="loginObj.password" placeholder="Password">
                <a href="#">→ Forget Your Password? ←</a>
                <button (click)="onLogin()">Sign In</button>
            </form>
        </div>
        <div class="toggle-container">
            <div class="toggle">
                <div class="toggle-panel toggle-left">
                    <h1>Welcome Back!</h1>
                    <p>Enter your personal details to use all of site features</p>
                    <button type="button" class="hidden" id="login" (click)="isSignDivVisiable = false">Sign
                        In</button>
                </div>
                <div class="toggle-panel toggle-right">
                    <h1>Hello, Friend!</h1>
                    <p>Register with your personal details to use all of site features</p>
                    <button type="button" class="hidden" id="register" (click)="isSignDivVisiable = true">Sign
                        Up</button>
                </div>
            </div>
        </div>
    </div>
</div>
```

Figure 15: Login.component.html screenshot.

5.4 Service

Another fundamental building block for creating an Angular application is the use of services. Delegating data management to one or more external entities, especially when retrieving information from a remote server, can be extremely useful. It is generally advisable to relieve components from certain responsibilities and delegate business logic to perform a specific function. Multiple services can be defined within an application, each responsible for completing a particular task. A service can also leverage the functionalities provided by other services. Services are typically defined as classes and decorated with the @Injectable decorator, which marks them as a service that can be injected.

Key Features of Angular Services

- **Reusability**: Services provide a way to encapsulate and reuse code across different components. This promotes the DRY (Don't Repeat Yourself) principle, ensuring that common functionalities are written once and used multiple times.
- **Dependency Injection**: Angular services leverage the dependency injection (DI) system, allowing services to be easily injected into components or other services. This makes it simple to manage dependencies and enhances testability. Services are usually provided in the root injector, making them available application-wide, but can also be provided at the module or component level.
- **Singleton Pattern**: When provided in the root injector, a service is instantiated as a singleton. This means that a single instance of the service is shared across the entire application, which is efficient for managing shared data or state.
- **Separation of Concerns**: By moving logic and data handling out of components and into services, Angular promotes a clean separation of concerns. Components handle the presentation and user interaction, while services handle the business logic and data operations.
- **Testing**: Services are easier to test in isolation because they are decoupled from the user interface. Unit tests can be written to ensure that the service logic works correctly without the need to test the components that use them.

5.4.1 Web API

In Angular 17, web services play a crucial role in enabling communication between the client-side application and a remote server. These services are primarily used for performing CRUD (Create, Read, Update, Delete) operations by interacting with @angular/common/http package, to facilitate these interactions.

Here's a detailed look at the role and implementation of web services in Angular 17:

- **Data Handling**: Web services are responsible for fetching data from external sources and sending data to servers. This separation allows components to focus solely on presenting data, while the logic for data handling is encapsulated within services.
- **HttpClient Module**: Angular 17 uses the HttpClient module to perform HTTP requests. This module provides various methods like get(), post(), put(), and delete(), making it easier to interact with REST APIs.
- **Interceptors and Error Handling**: Angular allows the creation of HTTP interceptors that can modify requests or handle errors globally. This ensures that all HTTP requests and responses pass through a centralized point, facilitating logging, authentication, or error handling.
- **RxJS and Observables**: The HttpClient methods return Observable objects from the RxJS library, which are subscribed to in the components. This allows for asynchronous and handling of real-time data streams effectively.

5.5 Server Side

The back-end of the project was developed separately from the front-end and managed in a different project. To seamlessly connect the front-end and back-end, the entire application was developed using Docker and microservices.

Microservices Overview

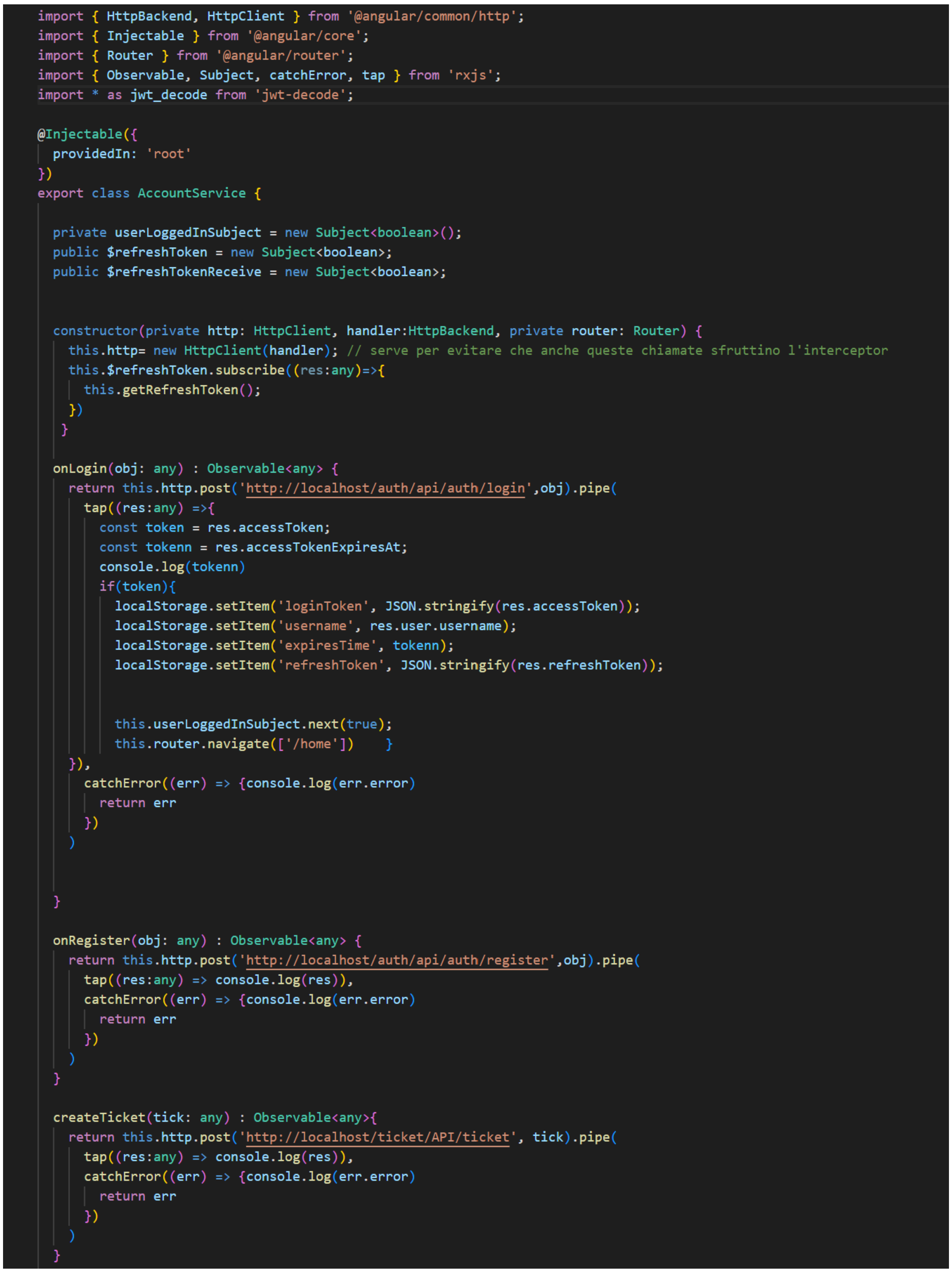
Microservices are a software architecture style that structures an application as a collection of loosely coupled services. Each service is responsible for a specific piece of business functionality and can be developed, deployed, and scaled independently.

Benefits of Microservices

1. **Scalability**: Microservices allow individual components to be scaled independently based on their specific demands, leading to more efficient use of resources.
2. **Flexibility in Technology**: Different microservices can be developed using different technologies, allowing teams to choose the best tool for each job.
3. **Resilience**: The failure of one microservice does not necessarily impact the entire system. This isolation helps in building more resilient applications.
4. **Faster Deployment**: Smaller codebases can be developed and deployed faster. Continuous integration and continuous deployment (CI/CD) processes are simplified, enabling quicker updates and feature releases.
5. **Improved Maintenance**: Microservices facilitate easier understanding and modification of individual services, which can improve the speed and efficiency of development and maintenance tasks.

Docker containers play a crucial role in modern software development by encapsulating each microservice, ensuring uniformity across various environments, and streamlining deployment processes. This containerization also facilitates efficient management and orchestration of microservices using advanced tools such as Kubernetes. By maintaining the isolation and autonomy of each service, Docker simplifies the complexities involved in developing, testing, and deploying sophisticated applications.

The integration of Docker and microservices results in a modular architecture that significantly boosts the scalability, maintainability, and overall resilience of the project. This modular approach empowers the independent evolution of both front-end and back-end components, promoting a highly agile and robust development lifecycle.



```
import { HttpBackend, HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { Observable, Subject, catchError, tap } from 'rxjs';
import * as jwt_decode from 'jwt-decode';

@Injectable({
  providedIn: 'root'
})
export class AccountService {

  private userLoggedInSubject = new Subject<boolean>();
  public $refreshToken = new Subject<boolean>;
  public $refreshTokenReceive = new Subject<boolean>;


  constructor(private http: HttpClient, handler:HttpBackend, private router: Router) {
    this.http= new HttpClient(handler); // serve per evitare che anche queste chiamate sfruttino l'interceptor
    this.$refreshToken.subscribe((res:any)=>{
      this.getRefreshToken();
    })
   }

  onLogin(obj: any) : Observable<any> {
    return this.http.post('http://localhost/auth/api/auth/login',obj).pipe(
      tap((res:any) =>{
        const token = res.accessToken;
        const tokenn = res.accessTokenExpiresAt;
        console.log(tokenn)
        if(token){
          localStorage.setItem('loginToken', JSON.stringify(res.accessToken));
          localStorage.setItem('username', res.user.username);
          localStorage.setItem('expiresTime', tokenn);
          localStorage.setItem('refreshToken', JSON.stringify(res.refreshToken));


          this.userLoggedInSubject.next(true);
          this.router.navigate(['/home'])    }
    }),
      catchError((err) => {console.log(err.error)
        return err
      })
    )


  }

  onRegister(obj: any) : Observable<any> {
    return this.http.post('http://localhost/auth/api/auth/register',obj).pipe(
      tap((res:any) => console.log(res)),
      catchError((err) => {console.log(err.error)
        return err
      })
    )
  }

  createTicket(tick: any) : Observable<any>{
    return this.http.post('http://localhost/ticket/API/ticket', tick).pipe(
      tap((res:any) => console.log(res)),
      catchError((err) => {console.log(err.error)
        return err
      })
    )
  }
```

Figure 16: Screenshot of a part of service.

5.6 Website Navigation Overview

This section illustrates the operation of the finished product through screenshots and explanatory comments on its functionality. Figure 17 shows the page that allows users to register by entering some credentials specified in the fields to be filled in.

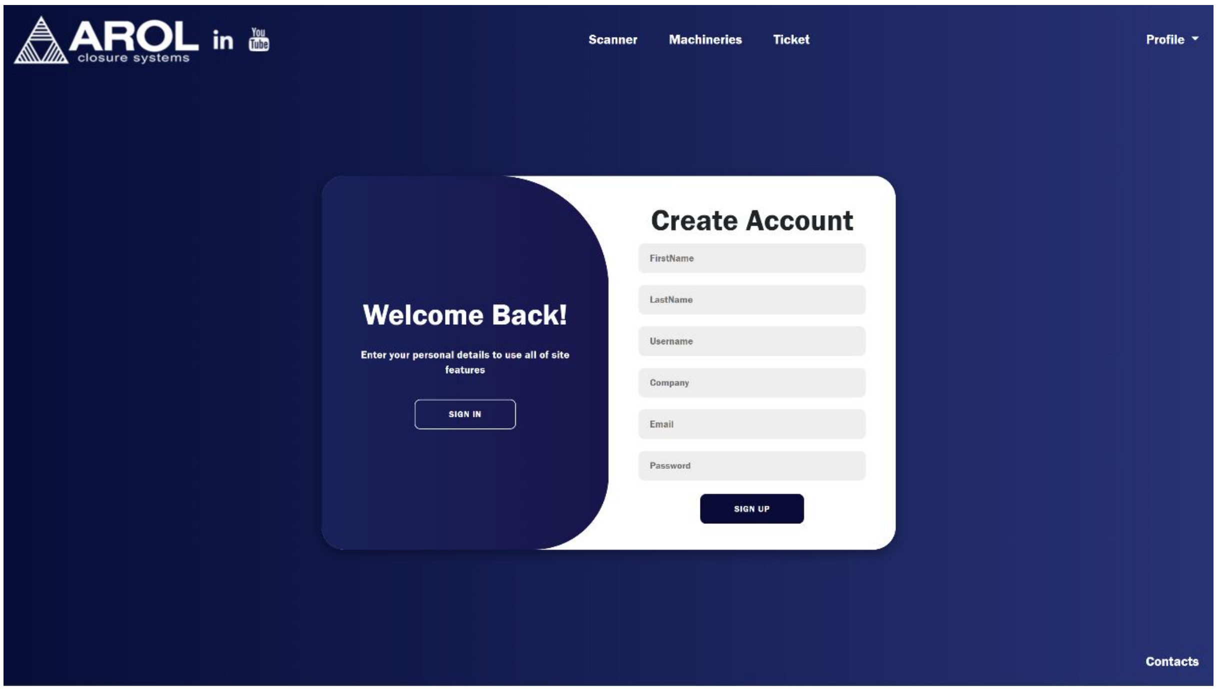


Figure 17: Registration page.

The newly created account can be used by the user to log in after registration. Figure 18 shows the login page.

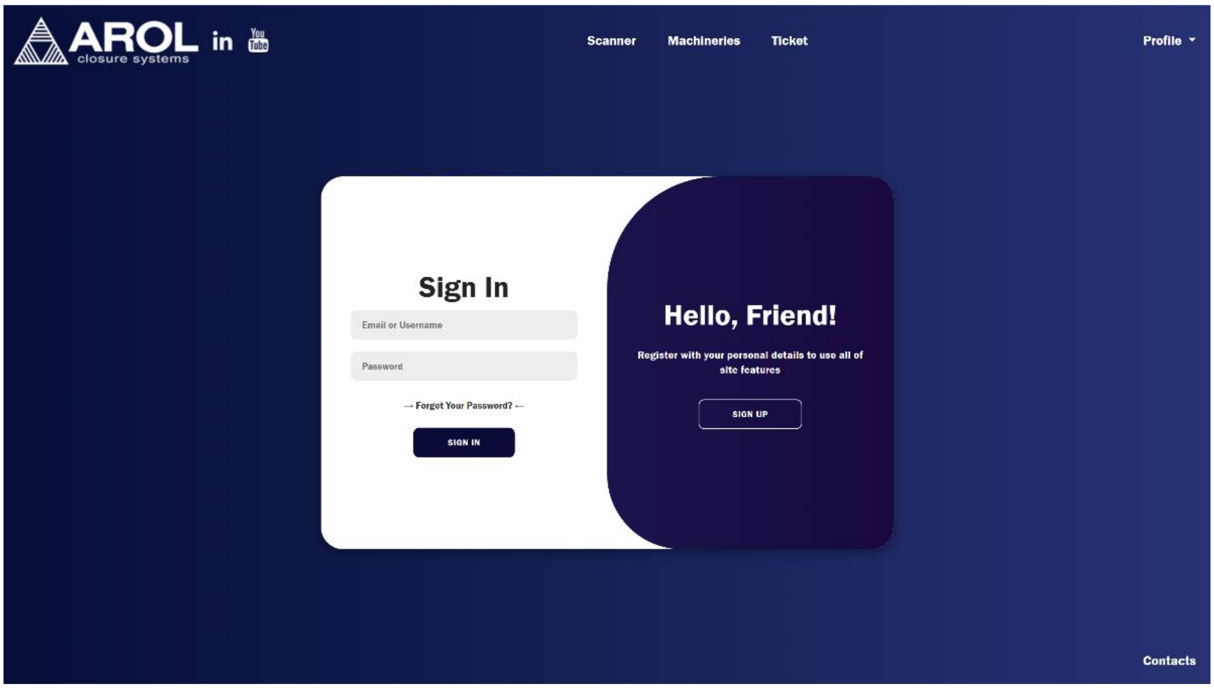


Figure 18: Login page.

Figure 19 shows the home page, with a video presentation of the company and clickable icons to take advantage of the app's functionality. In the navbar, the name of the logged-in user can be seen.

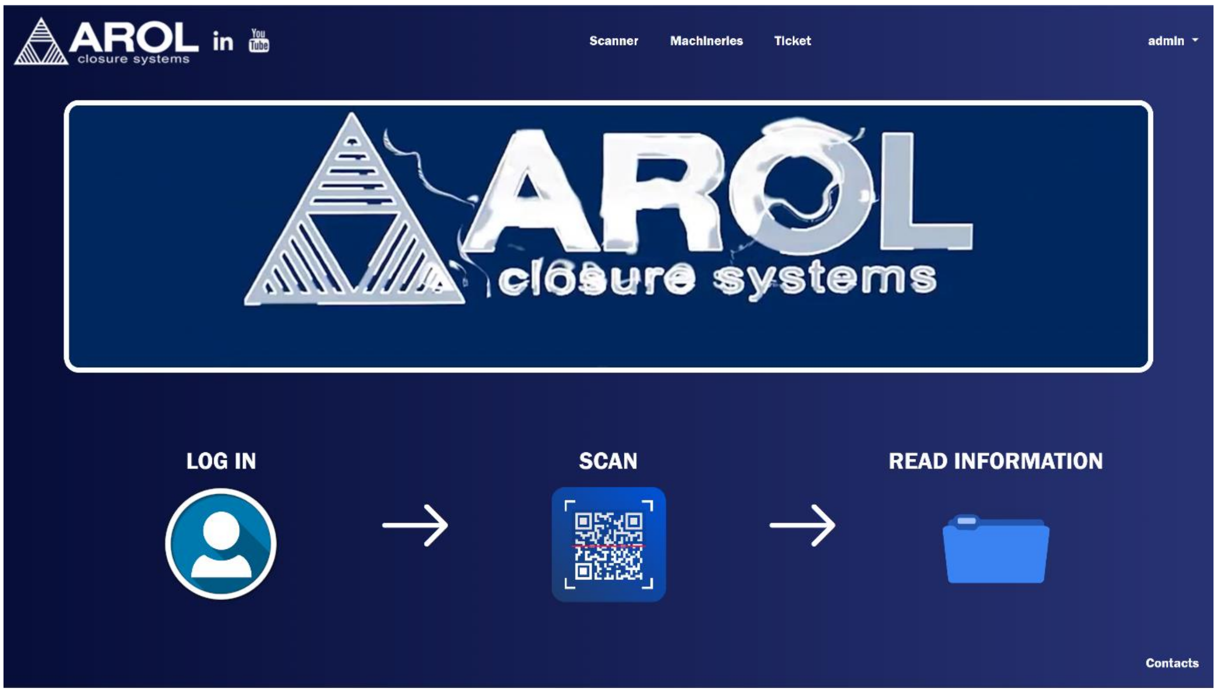
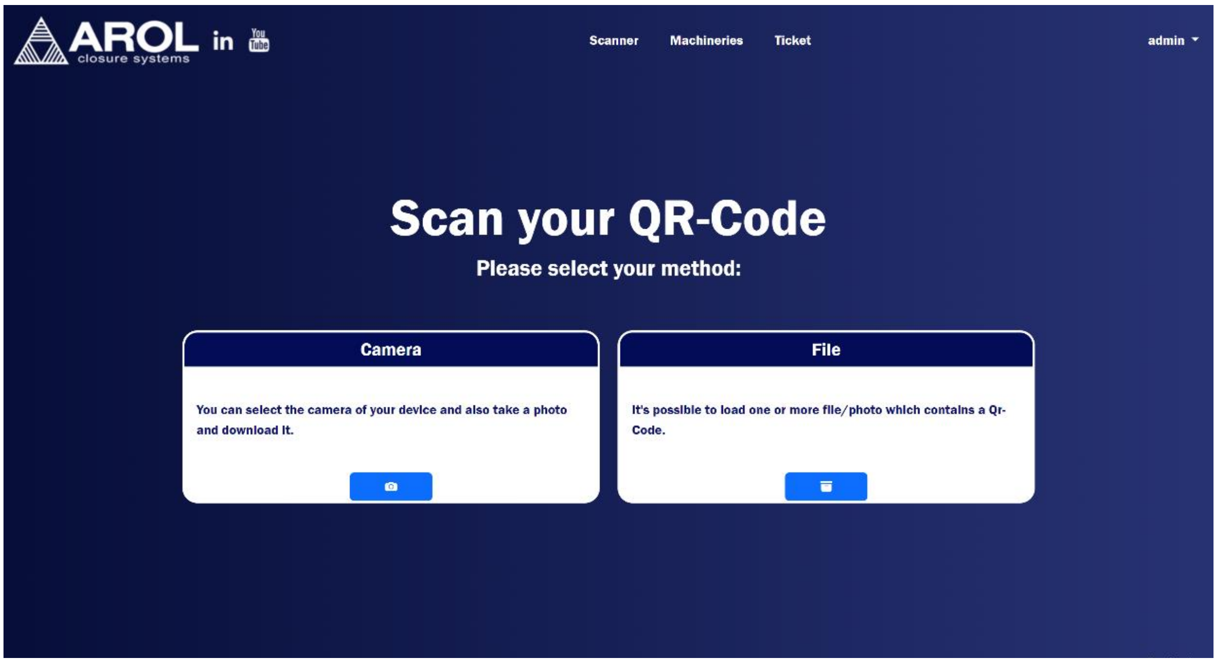


Figure 19: Home page.

By clicking on the scanner figure or the corresponding field in the navigation bar, the page in Figure 20 is displayed. Here you can choose your preferred QR-code scanning mode, between using the camera or uploading files/photos. depending on your choice, Figure 21 or Figure 22 will be shown in order.



Scan your QR-Code

Please select your method:

Camera

You can select the camera of your device and also take a photo and download it.

File

It's possible to load one or more file/photo which contains a Qr-Code.

Figure 20: Scanner page.

Once the camera mode is selected, the camera to be used (if the device has more than one) and the QR code decoding mode can be selected. The buttons shown below the frame for the image shown by the camera are used, in order from left to right, to start the camera, pause it, turn on the flash of the device, and take a photo that is downloaded automatically.

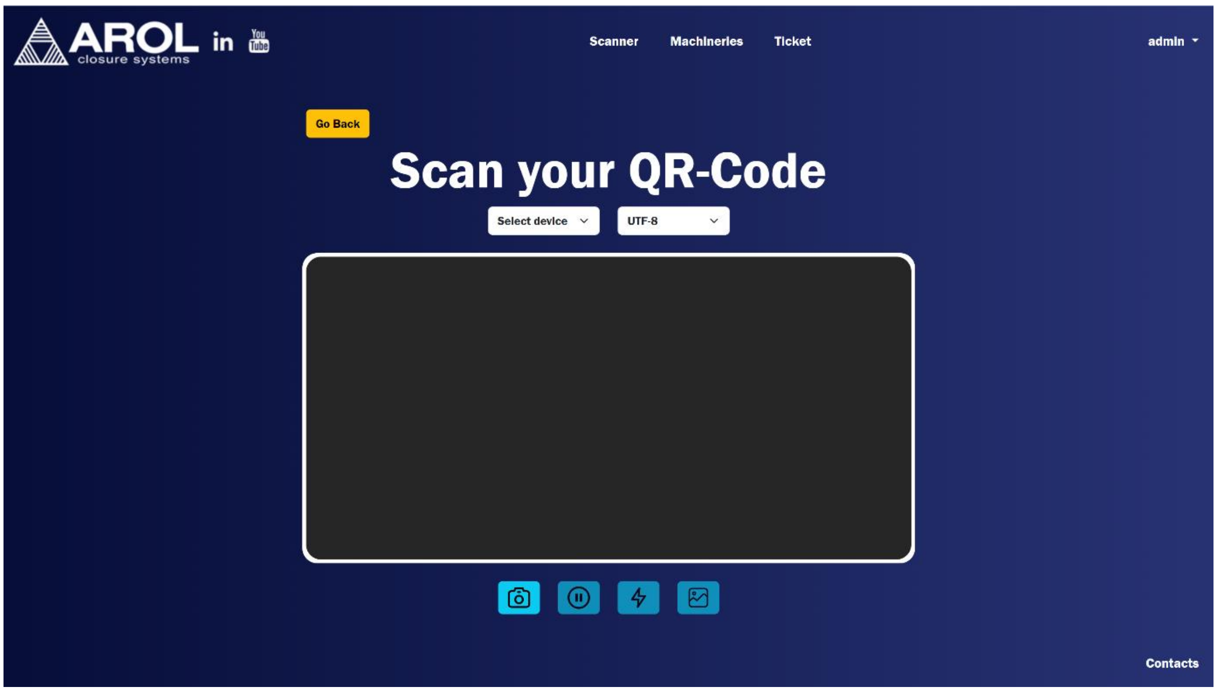
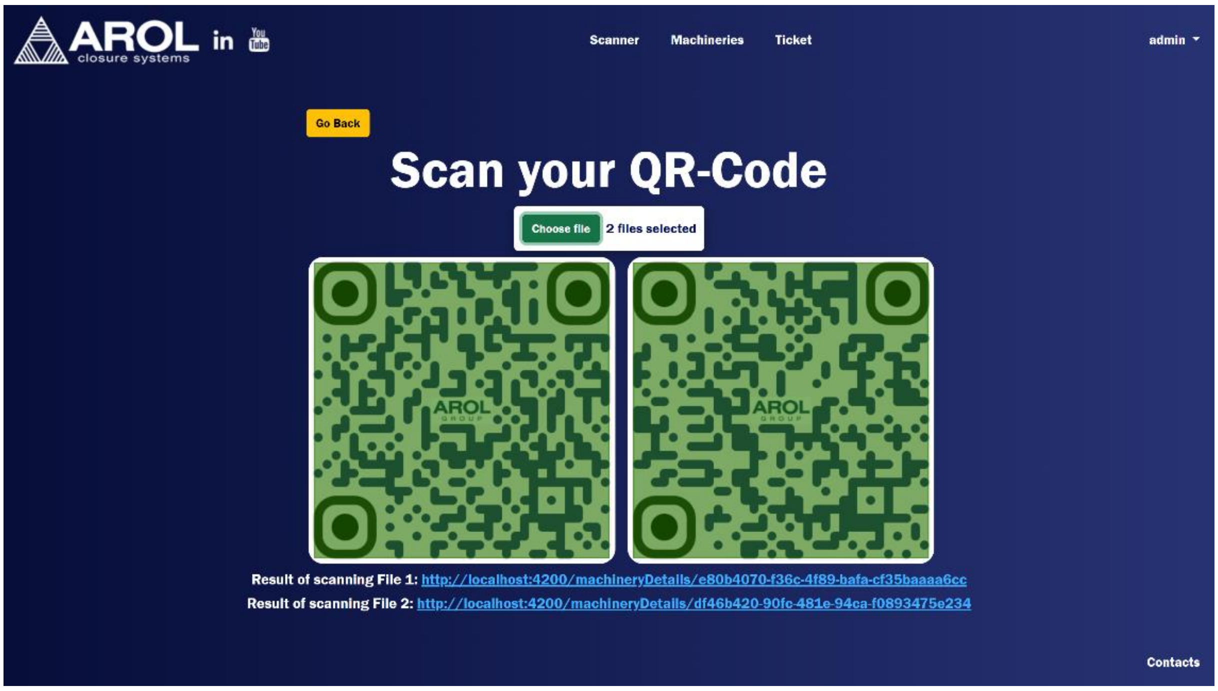


Figure 21: Camera scanner page.

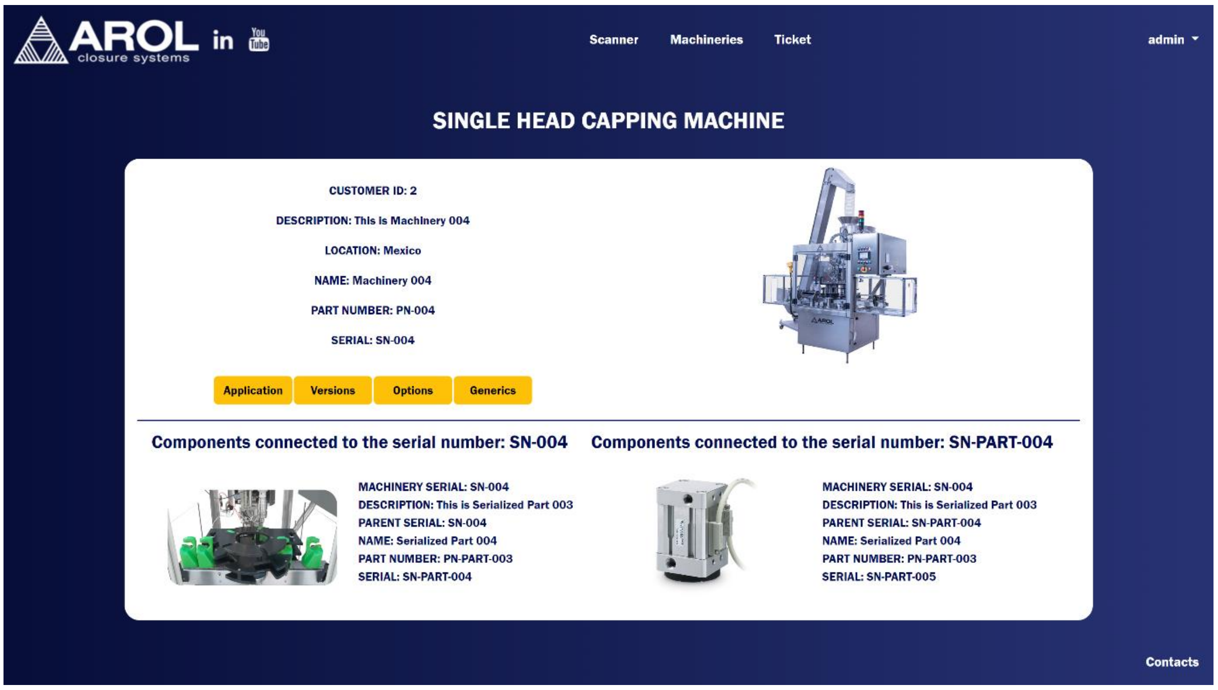
If file mode is chosen, one or more files/photos can be uploaded and will automatically be scanned. If a QR code is found within the file/photo, it will light up green (as it also does in camera mode when a QR code is scanned) and its content will be printed at the bottom of the page, which can be clicked to navigate to the desired page.



Result of scanning File 1: http://localhost:4200/machineryDetails/e80b4070-f36c-4f89-bafa-cf35baaaa6cc
Result of scanning File 2: http://localhost:4200/machineryDetails/df46b420-90fc-481e-94ca-f0893475e234

Figure 22: File scanner page.

Clicking on the scanning result, if it has a valid URL, i.e. if it is a QR code generated for one of the company's machines, will take you to the machine detail page, with all the information about it (see Figure 23). In addition to the specific information on the framed machinery, there is some more general information that can be displayed via the various buttons on the screen. In addition to the specific machine, other 'child' components will also be shown if it has any.



SINGLE HEAD CAPPING MACHINE

CUSTOMER ID: 2
DESCRIPTION: This is Machinery 004
LOCATION: Mexico
NAME: Machinery 004
PART NUMBER: PN-004
SERIAL: SN-004

Components connected to the serial number: SN-004

MACHINERY SERIAL: SN-004
DESCRIPTION: This is Serialized Part 003
PARENT SERIAL: SN-004
NAME: Serialized Part 004
PART NUMBER: PN-PART-003
SERIAL: SN-PART-004

Components connected to the serial number: SN-PART-004

MACHINERY SERIAL: SN-004
DESCRIPTION: This is Serialized Part 003
PARENT SERIAL: SN-PART-004
NAME: Serialized Part 004
PART NUMBER: PN-PART-003
SERIAL: SN-PART-005

Figure 23: Machinery details page.

Figures 24 and 25 show an example of the responsiveness of the application, showing how the navbar appears when the screen shrinks.

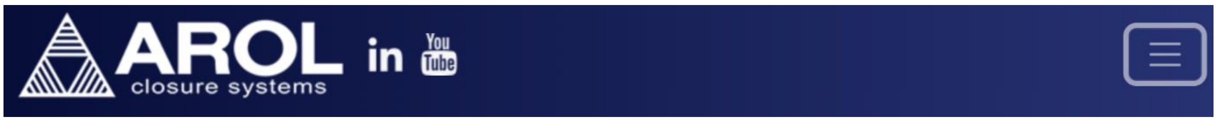


Figure 24: Navbar component when the screen is reduced.

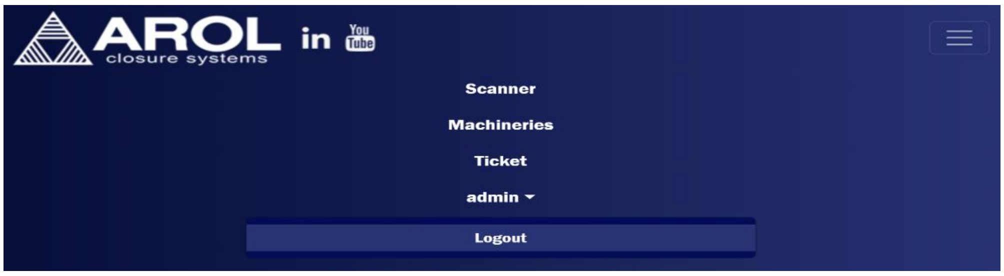
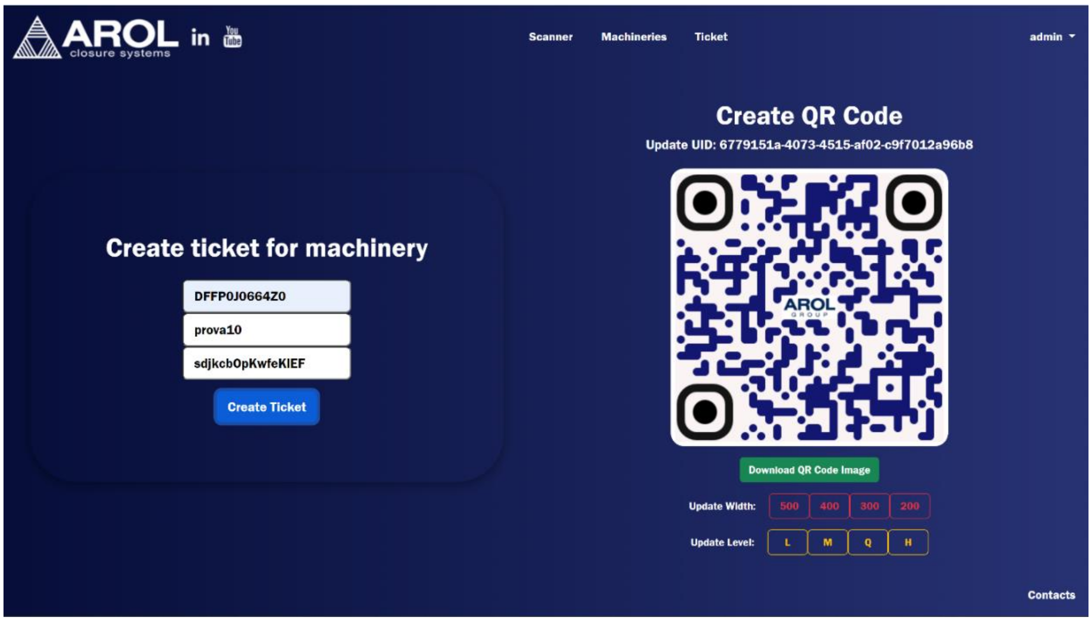


Figure 25: Navbar menu with reduced screen.

When the user is an admin, it is also possible to access the ticket creation page (Figure 26). Here, by filling in a few fields, it is possible to generate a unique ticket for a machine. Once the ticket is generated, the UID displayed within the QR code on the page will automatically update (thus changing its content). After selecting the size and error correction level, it is possible to download the generated QR code, which will be ready to be printed on the relevant machine.



DFFP0J0664Z0

prova10

sdjkcb0pKwfeKIEF

Update UID: 6779151a-4073-4515-af02-c9f7012a96b8

Figure 26: QR code creation page.

Finally, it is possible to log out at any time using the field in the navbar menu (Figure 27). A message will be displayed first, asking to confirm or cancel the operation (Figure 28).

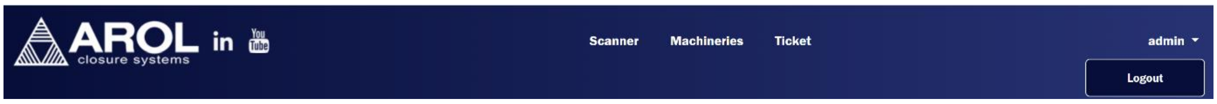
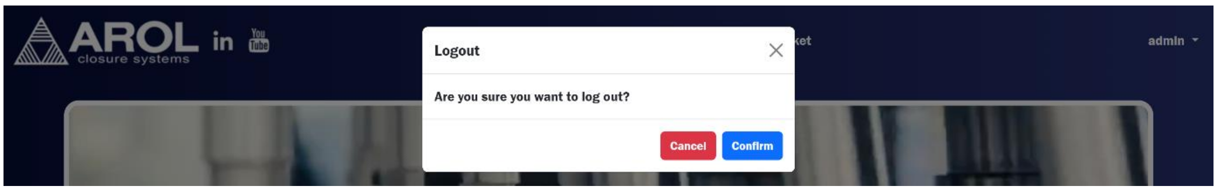


Figure 27: Logout option.



Are you sure you want to log out?

Figure 28: Confirm or cancel logout.

6. Conclusions and future work

In conclusion, the development of the web application for AROL represents a significant advancement in enhancing efficiency and traceability in industrial processes through the integration of QR code technology. This application has effectively addressed the critical need for a system that allows rapid and precise access to comprehensive information regarding machinery and components, thus significantly improving decision-making and operational efficiency. The strategic use of advanced technologies like Angular and TypeScript, combined with a responsive and intuitive design, ensures a robust and user-friendly interface tailored to the diverse requirements of AROL's workforce. Looking towards future developments, several enhancements could further elevate the functionality and impact of this application. One promising direction is the integration of real-time analytics and machine learning algorithms. These technologies could be used to predict maintenance needs, optimize operations, and reduce downtime, thereby ensuring machinery operates at peak efficiency. Another area for development is the enhancement of QR code scanning capabilities. Integrating augmented reality (AR) could provide a more interactive and immersive experience for users, making it easier to locate and identify machinery and components within the warehouse. This would be particularly beneficial during the identification and retrieval phases, improving accuracy and speed. Expanding the application to include localization features could also be immensely beneficial. By incorporating tools that track the precise location of machinery and components within the warehouse, the application could assist in identifying where specific items are stored and streamline the shipping process. This feature would be especially useful during inventory management and order fulfillment, ensuring that items are correctly identified, picked, and shipped efficiently. Additionally, further development could focus on making the application more versatile and accessible. This includes expanding language support to cater to a more diverse workforce and providing personalized user experiences based on historical data and usage patterns. These enhancements would make the application more user-friendly and ensure it meets the specific needs of each user, improving overall satisfaction and productivity. Continual feedback from users will be critical in driving these future enhancements. By actively involving users in the development process, AROL can ensure that the application evolves to meet emerging needs and technological advancements. This iterative approach will help maintain the relevance and effectiveness of the application, ensuring it continues to provide value in a rapidly changing industrial landscape.

In summary, while the current development marks a substantial improvement in operational efficiency and information accessibility for AROL, future enhancements will focus on predictive maintenance, advanced QR code capabilities, localization features, and personalized user experiences. These developments will further cement AROL's position at the forefront of industrial automation, leveraging cutting-edge technology to achieve operational excellence and innovation.

7. References

[1] https://www.sweetprocess.com/sop-software-for-small-business/

[2] https://gr0.com/blog/what-is-geofencing

[3] https://perception-point.io/guides/phishing/quishing-how-qr-code-phishing-works-and-4-ways-to-prevent-it/

[4] https://www.qodenext.com/blog/find-out-the-difference-between-rfid-and-qr-code/

[5] https://goo.by/blog/why-qr-code-is-important-for-digital-marketing-and-customer-engagement

[6] https://docshipper.com/guest-blogging/procurement-sourcing-trends-2023/

[7] https://www.fraugroup.com/blog/food-safety-in-production-plants-moca-compliance-and-certification/

[8] https://adamshalalmeat.com/#gsc.tab=0

[9] https://medium.com/@sudarakakalindu20/software-design-patterns-3e314b98c28e

[10] https://www.dadegrees.com/design-patterns-interview-questions-c/

[11] https://www.tutorialsfreak.com/java-tutorial/java-design-patterns

[12] https://www.geeksforgeeks.org/html-tutorial/

[13] https://medium.com/@digvijayackrolix/flutter-angular-or-react-which-is-the-right-framework-for-your-next-web-and-mobile-app-a9d9f2f3fea1

[14] https://topperworld.in/introduction-to-javascript/

[15] https://digitaledgewizards.com/java-script-for-beginners/

[16] https://dev.to/henrylehd/-understanding-javascript-and-typescript-a-beginners-guide-4o58

[17] https://www.naukri.com/code360/library/features-of-javascript

[18] https://pravin-hub-rgb.github.io/BCA/resources/sem4/web_tbc404/unit4/index.html

[19] https://medium.com/@rs4528090/understanding-typescript-a-comprehensive-guide-851697a8e424

[20] https://medium.com/@eng.agamil/azure-service-fabric-the-complete-blueprint 102d9476b55e

[21] https://dev.to/oloruntobi600/microservices-with-spring-boot-5457

[22] https://www.oyova.com/blog/ecommerce-microservices/

[23] https://angular.dev/overview

[24] https://www.arol.com/it/arol-canelli

[25] https://www.html.it/guide/guida-html5/

[26] https://www.html.it/guide/guida-css-di-base/

[27] https://www.typescriptlang.org/

[28] https://www.uxdesigninstitute.com/blog/ux-design-principles/

[29] https://it.wikipedia.org/wiki/Visual_Studio_Code

[30] https://www.atlassian.com/it/software/bitbucket

[31] https://www.atlassian.com/it/software/jira/guides/getting-started/introduction#what-is-jira-software

[32] https://www.newebsolutions.com/css-grid-flexbox-come-sfruttarli/

[33] https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

[34] https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_media_queries/Using_media_queries

[35] https://startup-house.com/blog/benefit-of-typescript

[36] https://qrpurple.com/

[37] https://www.uniqode.com/what-is-qr-code

[38] https://www.uniqode.com/blog/qr-code/how-much-data-can-qr-code-hold

[39] https://www.cmswire.com/customer-experience/ui-vs-ux-design-whats-the-difference/

[40] https://dev.to/topefasasi/js-design-patterns-a-comprehensive-guide-h3m

[41] https://ahmadhi.medium.com/revamping-your-codebase-leveraging-template-design-pattern-and-refactoring-6b8b82a5044a

[42] https://www.neatcode.org/design-patterns/#:~:text=Creational%20patterns%20focus%20on%20object,communication%20and%20interaction%20between%20objects

[43] https://www.paidant.com/blog/visual-studio-vs-visual-studio-code-the-epic-showdown-every-programmer-must-witness/

[44] https://en.wikipedia.org/wiki/HTML

[45] https://vlkedu.com/blog/what-are-jira-software-and-confluence/

[46] https://www.blissdrive.com/seo/why-do-web-developers-use-html-css-and-javascript/

[47] https://www.tutorialspoint.com/software_architecture_design/component_based_architecture.html

[48] https://cloudvandana.com/angular-components-best-practices-for-development/

[49] https://medium.com/codex/typescript-vs-javascript-a-comparison-of-the-two-languages-and-their-strengths-and-weaknesses-71fa9a7f9029

[50] https://relationshipbetween.com/difference-between-javascript-and-vs-typescript/?utm_content=cmp-true

[51] https://kurtwanger40.medium.com/how-to-lazy-load-standalone-components-ff6b2298259f

[52] https://www.mega.com/blog/what-is-microservices-architecture

[53] https://www.geeksforgeeks.org/monolithic-vs-microservices-architecture/

[54] https://www.angularminds.com/blog/angular-17-new-features

[55] Figure 1: https://www.arol-group.com/images/AROL-1024x205.png

[56] Figure 2: https://img.directindustry.it/images_di/photo-mg/28105-18669473.jpg

[57] Figure 3:

https://upload.wikimedia.org/wikipedia/commons/thumb/c/cf/Angular_full_color_logo.svg/250px-Angular_full_color_logo.svg.png https://www.adm.ee/wordpress/wpcontent/uploads/2023/12/javascript_and_typescript-1.jpg

https://raw.githubusercontent.com/github/explore/80688e429a7d4ef2fca1e82350fe8e3517d3494d/topics/css/css.png

https://e7.pngegg.com/pngimages/780/934/png-clipart-html-logo-html5-logo-icons-logos-emojis-tech-companies-thumbnail.png

[58] Figure 4: https://miro.medium.com/v2/resize:fit:730/1*bCrbhKFOoPc6d56F6h3OYQ.png

[59] Figure 5: https://www.uxmatters.com/IoT_Hero.png

[60] Figure 6: https://blog.vfccu.org/content/images/2022/03/internet-of-things.jpg

[61] Figure 7:

https://gototags.com/wp-content/uploads/sites/12/2022/08/barcode_formats.png

[62] Figure 8:

https://koronapos.com/wp-content/uploads/2021/09/1196925_2-New-Infographs_1_092921.png

[63] Figure 9:

https://www.onoratoinformatica.it/wp-content/uploads/2022/07/QR-Codephishing.png

[64] Figure 10:

https://www.vendingnews.it/wp-content/uploads/2019/04/MOCA-600-min.png