Astana IT University

Issadilov Akezhan
Kanatbek Baidaulet
Ibragimov Olzhas

"Accounting of employees of the organization and their material values: The inventory control system"

6B06102 - Software Engineering

Diploma project

Supervisor
Seraliyev Zhandos
MSc Information Systems Senior lecturer

Kazakhstan Republic
Astana, 2023

CONTENTS

DEFINITIONS

Following terms are used in this work:

PostgreSQL	A commonly used solution for structured data storage and retrieval, an open-source object-relational database management system. It offers a dependable and scalable solution for controlling project inventories.
Client-Server Application	A software architecture in which client devices connect to a central server in order to access and process data. Inventory management application for this project uses a client-server paradigm, in which the front-end communicates with the back-end server.
Front-end	The user-facing component of a software application, which comprises the features that users interact with and the graphical user interface (GUI) of the application.
Back-end	The component at the server side of a software program, responsible for processing, storage, and retrieval of data. The back-end for this project is constructed using ASP.NET, a web application framework, and works with the PostgreSQL database.
HTTP Calls	The communication protocol that web browsers and servers use to communicate with one another across the internet. HTTP calls are used to send requests to the front-end and get data replies.
ReactJS	A JavaScript toolkit for making user interfaces that lets you make reusable UI components. It offers a quick and effective method for creating interactive and dynamic front-end interfaces.
ASP.NET	A framework for web applications created by Microsoft. With the help of the.NET framework and programming languages like C#, it enables developers to create dynamic

	and interactive online applications.
HTTP	Hypertext Transfer Protocol - The protocol used to send data across the internet is called the hypertext transfer protocol. It specifies the structure and guidelines for client and server communication.
RFID	A system called Radio Frequency Identification (RFID) utilises radio waves to passively identify tagged objects. It is employed in a variety of commercial and industrial applications, including inventory management and the tracking of library checkouts.
Cloud-based	Any software that is stored, controlled, and accessible through the cloud is referred to as cloud-based software. Users need an internet connection in order to access such services or applications. It is typically kept on shared computer platforms like cloud servers.
IoT	The network of physical objects—"things"—that are integrated with sensors, software, and other technologies for the purpose of communicating and sharing data with other devices and systems through the internet is referred to as the Internet of Things (IoT).
AWS services	The most complete and widely used cloud in the world, Amazon Web Services (AWS), provides over 200 fully functional services from data centres across the world.
Inventory Management	Controlling the movement of products, commodities, and resources effectively within an organisation. It entails tasks including monitoring inventory levels, controlling stock, and improving supply chain operations.

DESIGNATIONS AND ABBREVIATIONS

Following designations and abbreviations are used in this work:

AWS	Amazon Web Services
IoT	Internet of Things
HTTP	HyperText Transfer Protocol
RFID	Radio Frequency IDentification
QR	Quick Response
SMEs	Small and Medium Enterprises
AI	Artificial Intelligence
ASP.NET	Active Server Pages.NET
UI	User Interface
CSS	Cascading Style Sheets
PDF	Portable Document Format
REST API	Representational State Transfer and Application Programming Interface
JWT	JSON Web Token
AES	Advanced Encryption Standard

RSA	Rivest, Shamir and Adleman
DFD	Data Flow Diagrams
WCAG	Web Content Accessibility Guidelines
PC	Personal Computer
IT	Information Technology
MSc	Master of Science
GUI	Graphical User Interface
JS	JavaScript

ANNOTATION

The management of inventory in university housing is complicated by the arrangement and monitoring of resources across various rooms and facilities. The purpose of this project is to develop a client-server application that addresses these problems in a user-friendly and efficient manner. The application makes use of cutting-edge tools including ReactJS, ASP.NET, PostgreSQL, and AWS for hosting. The appealing and straightforward user interface of the front-end component allows users to add, edit, and remove classes and commodities. The back-end component, which also interacts with the database and ensures efficient data retrieval and storage, is in charge of handling the business logic. The front-end and back-end components are connected through HTTP calls, allowing for real-time modifications and a seamless user experience. The hosting component offers a secure and dependable deployment using AWS services by automating the deployment process and managing the servers. The system's data flow is started by user input on the front-end user interface and follows a clearly defined path via the back end to the PostgreSQL database. Overall, this client-server application provides a useful method for managing inventory in student housing, accelerating resource allocation, averting shortages, and boosting productivity. Thanks to the integration of components, modern technology, and best practices, which ensures scalability, flexibility, and maintainability, the system is well-equipped to meet the university's evolving requirements.

1 Introduction

The idea of inventory management has a long history, dating back to the earliest civilizations, when records of the materials and goods kept in warehouses were kept. These records were used to monitor the movement of supplies and make sure there were enough supplies on hand to meet the needs of the neighborhood. More sophisticated inventory management systems became necessary as businesses expanded and international trade increased. Businesses can now track and manage their inventory more easily thanks to new technologies like barcode scanning and inventory management software.

Businesses frequently measure and manage their inventory using statistics like inventory turnover and days of inventory. Inventory turnover is a ratio that measures how often a business sells and replaces its inventory. It is computed by dividing the average inventory by the cost of goods sold. A greater inventory turnover ratio shows that a company is effectively controlling its inventory and moving its goods off the shelves more quickly. On the other hand, days of inventory is a measurement of how long it takes a company to sell its goods. It is computed by multiplying the number of days in the year by the average inventory, cost of products sold, and division. A business is effectively managing its inventory and selling its products at a faster rate when it has fewer days of inventory.

Inventory management is an essential component of supply chain management and logistics in terms of its significance in businesses and organizations. A company needs effective inventory management since it enables them to maintain ideal stock levels, cut expenses, and enhance customer service. Businesses may make sure they have the appropriate products in the right quantities at the right time by maintaining accurate records of inventory levels and tracking stock movements. They may be able to do this to prevent stockouts, which may result in lost revenue, and overstocking, which may squander precious resources. A company can also identify slow-moving or out-of-date products with the use of efficient inventory management, which enables them to change their product mix and reduce losses.

Overall, inventory management is an essential task for every business that stores tangible products. An efficient system is necessary to maximize the efficiency of the business.

Any organization that stores tangible products must execute the essential task of managing their inventory; to maximize their efficiency, they must have a strong system in place. However, businesses may run into a number of issues that could have a detrimental impact on how they manage their inventory.

Overstocking is one of the most prevalent issues and happens when a company keeps an excessive amount of inventory. This wastes precious resources and raises the possibility of stock going out of date or losing value. Overstocking can result

in higher carrying expenses, such as insurance and warehousing, which can have a detrimental effect on a company's bottom line. Additionally, excess inventory can occupy space that could be used for expansion or the addition of new products.

Stockouts, which happen when a company runs out of a particular product, are another frequent issue. This may result in dropped sales and maybe disgruntled customers. Stockouts can also interfere with supply chain and production processes, resulting in delays and higher expenses. Stockouts can also cause businesses to lose out on sales opportunities, which can be detrimental to their reputation and consumer base [1].

Having inaccurate inventory data is another issue that companies could experience. Inventory management decisions may be made incorrectly when a company does not maintain reliable records of inventory levels, movements, and other information. Due of this, it may be challenging for a company to identify outmoded or slow-moving products and to estimate future demand. Another issue that firms could have is shrinkage, or the loss of goods as a result of theft, damage, or other factors. This can result in financial losses and can be challenging to monitor and stop.

Another issue that firms could experience is a lack of visibility. It might be challenging for firms to have a comprehensive perspective of their inventory in real-time without an advanced inventory management system. Because of this, it may be difficult for firms to react swiftly to shifts in demand or supply and may result in poor decision-making.

Businesses could also have trouble with limited forecasting skills. Poor inventory control can make it difficult for a company to predict future demand, which can result in stock outs or overstocking.

Limited scalability is yet another issue that organizations could experience. It may become more challenging to scale a business as it expands if it relies on manual inventory management procedures.

In the current digital era, web applications are crucial since they give customers simple online access to a variety of goods and services. They include e-commerce, social media, education, healthcare, and other industries and use cases [2].

The accessibility of web apps is one of its key benefits. Web apps are accessible from any device with an internet connection, which makes it simple to operate from a distance or while on the road. As numerous users may access and work on the same programme at once, this also enables increased cooperation and communication.

Scalability and affordability are further features of web apps. They can easily handle enormous volumes of data and users and don't need to be downloaded or installed. Web apps are simple to update and manage, and they are also

considerably less expensive to design and maintain than desktop applications.

The ability of web apps to interface with various platforms and systems is another benefit [3]. This makes it possible for several apps to communicate and share data seamlessly, which can significantly increase productivity and efficiency.

Web applications also have the potential to improve user experience. Web applications may offer users a quick, simple, and visually appealing experience by utilizing responsive design and contemporary technologies. Web applications are widely used in a variety of industries, including e-commerce, where they give customers a platform to buy products and services online, and education, where they give students a platform to access course materials and interact with their peers and teachers.

In general, web apps have many benefits over conventional desktop applications, including usability, scalability, affordability, and integration. They are widely employed across many industries to increase productivity and efficiency in the current digital era.

In order to address and improve the organization of the inventory in university rooms (practice and lecture classrooms, staff housing, etc.), this diploma project proposes an inventory system. This client-server programme avoids situations where assigned classes are left without furniture (such as tables and chairs). Authorized users can manage and arrange the inventory of rooms in the client-server application, while visitors can only browse. There are a few distinct sorts of user accounts with various levels of access. Users of the application can scan a QR code or barcode to view full information, make edits, and more. The methodology also suggests the various software development development and testing methods. In the end, a client-server application that is ready to use is developed to address the university inventory issue.

This entails identifying the crucial parties involved in inventory management as well as the systems in use at the moment for inventory tracking and management. This is crucial because it gives a general picture of the situation and makes it possible to spot any gaps or inefficiencies in the current procedure.

Analyze how well the present inventory management system is working. This entails evaluating the correctness of inventory information, spotting any tracking or accountability problems, and gauging the system's effectiveness. This is crucial since it permits the detection of any issues that might be impeding efficient and effective inventory management.

Determine what aspects of the inventory management system need work. This could entail locating areas for automation and digitization, dealing with concerns with accountability and tracking, and looking for ways to boost the system's effectiveness. This is crucial because it makes it possible to recognise any adjustments that should be made to the inventory management system.

Give suggestions for enhancing the university's inventory management system. This could involve both targeted improvements to the system and more comprehensive approaches to dealing with the problems that have been discovered. This is significant since it offers the university a strategy for enhancing its inventory management system.

Inventory management is essential for compliance and accreditation because colleges and universities need to be able to show that they are using their resources in accordance with the laws and accreditation criteria that apply. The university will be able to identify any compliance issues and create strategies to remedy them with the aid of this investigation.

Budgeting and forecasting: A successful inventory management system is essential for these two processes. The university will be assisted by this study in locating any inefficiencies in the current system and formulating plans to address them.

Improvements in Resource Management: The university can better track and manage its assets, including its equipment, supplies, and other resources, thanks to an effective inventory management system.

Overall, the analysis of the university's inventory system is critical because it gives the institution the knowledge and tools it needs to manage its resources more effectively, increase compliance and accreditation, and improve budgeting and forecasting [4]. This research helps the university operate more successfully and efficiently, which eventually boosts the institution's success.

This research paper is divided into a number of sections, each of which deals with a different component of the investigation. The pertinent theories, research, and methodology are presented in Section 1, along with any gaps in the prior literature and how the present study fills them. The system under consideration is thoroughly described in Section 2, along with its architecture, parts, and functionalities, as well as an overview of the system's current problems and difficulties, as well as the study's goals. Section 3 provides a thorough explanation of the research techniques employed, including the study design, sample, data collection, and analytic procedures. The approaches and methods used to analyze the system and data are described in Section 4, along with a thorough discussion of the data processing, modeling, and visualization techniques employed. Section 5 presents the study's conclusions, along with any statistical analysis, tables, or figures that support them. The study's primary conclusions and findings are finally summarized in Section 6, highlighting the significance of the study and its contributions to the field. Following the proper reference format, the sources used in the paper are listed in Section 7.

2 Literature Review

The literature review is a crucial part of every research effort because it offers a thorough overview of the body of knowledge and body of research that has been done on a particular issue. The purpose of the literature review in the context of this study is to examine and evaluate prior research, academic publications, books, and other pertinent sources of data regarding inventory management in university housing.

In order to guarantee the efficient use of resources in any company, including colleges, inventory management is a crucial component. To support academic activities and offer students a supportive learning environment, inventory management in university housing is essential. It can be a challenging undertaking, though, needing careful organization, planning, and execution.

We intend to acquire insights into the existing state of knowledge on inventory management in university housing by performing an extensive literature review. With the help of this review, we will be able to see where there are gaps in the literature and consider prospective directions for future research. It will also serve as a theoretical framework for our research and serve as a guide for creating our client-server inventory management application.

A variety of inventory management topics, such as best practices, difficulties, and potential solutions that are unique to university housing, will be covered in the literature review. We will review and evaluate studies that have looked at the use of technology, such as client-server applications, in controlling inventories. In addition, we'll look at how various frameworks and tools, like ASP.NET and ReactJS, can be used to create inventory management systems.

The literature analysis will also explore the advantages of adopting and hosting inventory management systems on the cloud, notably on Amazon Web Services (AWS). We will look at research that have highlighted the benefits and difficulties of cloud hosting and investigate how it can improve the system's scalability, security, and accessibility.

Overall, the evaluation of the literature will give a thorough grasp of the state of inventory management in university housing today. It will assist us in developing our client-server application in a way that incorporates best practices and takes into account the difficulties that the industry is now facing. By expanding on the knowledge and research already available, we hope to develop inventory management procedures in academic settings and offer a useful remedy for boosting effectiveness and resource utilization in university housing.

2.1 Current State of Inventarization Systems

Technology has played a vital influence in the considerable evolution of inventorization systems over time. Today, the majority of businesses manage their inventory using computerised systems. These systems are created to deliver precise and current information regarding the inventory, including stock levels, locations, and product movement.

One of the key advantages of putting in place an efficient inventory management system is cost savings. Educational institutions can lower the expenses of overstocking and understocking inventories by controlling inventory levels appropriately. Effective inventory management can also result in lower labor expenses compared to manual inventory management.

Another advantage of putting in place an efficient inventory management system is increased productivity. Inventory management can be simplified with automated solutions, freeing up workers to work on more important projects. Standardized procedures can also save effort duplication and boost inventory management effectiveness.

Another advantage of putting in place an efficient inventory management system is better resource utilization. Educational institutions can make sure that by successfully managing their inventory that.

However, due to the difficulties involved in manually managing inventory, traditional inventory management techniques have been gradually supplanted by digital inventory management systems. Comparing digital inventory management systems to traditional ones, there are a number of benefits. Particularly, digital inventory management systems outperform manual inventory management solutions in terms of accuracy, efficiency, and speed. Additionally more scalable, or able to handle bigger volumes of goods, are digital inventory management systems.

Digital inventory management systems come in a variety of forms, including cloud-based, RFID-based, and barcode-based systems. Based on cloud computing technology, cloud-based inventory management systems let users control their inventory from anywhere in the globe as long as they have an internet connection. While barcode-based inventory management systems utilize barcodes to identify and monitor inventory items, RFID-based inventory management systems use RFID tags.

The way businesses track and manage their inventory has been completely transformed by the usage of barcodes and radio-frequency identification (RFID) technologies. Products are labeled with barcodes to facilitate scanning and product identification. On the other hand, RFID tags are affixed to commodities to follow their movement and provide real-time information about their whereabouts. Because of these technology, businesses can now track inventory in real-time,

which lowers the chance of stockouts and overstocking.

However, computerized inventory systems perform more tasks than just keeping track of stock. They also provide organizations a number of advantages. They first offer real-time data on inventory levels, which is essential for monitoring stock levels and averting stock-outs. Second, the improved inventory management made possible by computerised systems makes it simpler to monitor the movement of items, stock levels, and stock locations. Third, automated inventory management solutions lessen the possibility of human mistake, which can result in incomplete inventory records.

Computerized inventory management systems have also made it simpler for businesses to manage their inventory across various locations in addition to these advantages. Computerized inventory systems have made it feasible for businesses to more effectively manage their inventory across many locations by giving real-time information about inventory levels and the movement of goods. This has allowed businesses to concentrate on other areas of their operations by saving them time and money.

Computerized inventory management systems have also made it simpler for businesses to manage their inventories in real-time. Computerized inventory systems have made it feasible for businesses to react swiftly to changes in demand by giving real-time information on inventory levels and the movement of goods. As a result, businesses have seen an increase in their bottom line by lowering the risk of stock-outs and overstocking.

Generally speaking, the usage of computerized inventory systems has developed into a crucial tool for businesses aiming to automate their inventory management procedures and boost their financial performance. Computerized inventory systems have made it simpler for businesses to manage their inventory more effectively, saving time and money in the process. These systems have the capacity to give real-time information regarding inventory levels, location, and movement of goods.

Carter (2020) provides an overview of the implementation of an effective inventory management system [5]. The author highlights the importance of having an efficient inventory management system in order to reduce costs, increase profitability, and improve customer satisfaction. The author also mentions the need for clear goals and objectives, as well as proper planning and execution, in order to successfully implement an inventory management system as well as improved accuracy, which is one of the most significant benefits of using an inventory management system. By reducing the number of errors and improving the accuracy of inventory data, businesses can ensure that they have an accurate understanding of their inventory levels at all times.

Kumar and Singh (2021) conduct a review of inventory management systems.

The authors analyze different types of inventory management systems, including perpetual and periodic inventory systems, and discuss their advantages and disadvantages [6]. The authors also examine the role of technology in inventory management and the impact of the internet of things (IoT) on inventory management practices with another major advantage of inventory management systems - increased efficiency. By automating and streamlining many of the manual processes involved in inventory management, businesses can save time and increase their overall efficiency. Better inventory control is also a key benefit of using an inventory management system.

Lee and Whang (2019) focus on the analysis of inventory management practices in small and medium-sized enterprises (SMEs) [7]. The authors find that SMEs face unique challenges in inventory management, such as limited resources and limited access to technology. The authors suggest that SMEs adopt inventory management practices that are suitable for their size and resources. The integration of technology, particularly the Internet of Things, has brought about significant improvements in the accuracy and efficiency of inventory management practices. However, small and medium-sized enterprises (SMEs) face unique challenges in inventory management, such as limited resources and limited access to technology. To overcome these challenges, SMEs must adopt inventory management practices that are suitable for their size and resources.

Manojlovich and Loughridge (2022) examine the impact of technology on inventory management systems [8]. The authors find that technology has improved inventory management by reducing the time and cost of inventory management processes, improving accuracy, and enabling real-time monitoring of inventory levels. The authors also discuss the potential challenges associated with technology-based inventory management systems, such as the need for specialized skills and the potential for data breaches. Technology has been a major driver of change in inventory management, reducing the time and cost of inventory management processes and enabling real-time monitoring of inventory levels. However, technology-based inventory management systems may also come with challenges such as the need for specialized skills and the potential for data breaches. Despite these challenges, technology has played a critical role in improving inventory management practices and ensuring that businesses have the right products in stock to meet customer demand.

Wong-On-Wing and Tam (2020) focus on the role of inventory management in supply chain management [9]. The authors argue that inventory management is a critical component of supply chain management, as it helps to ensure that products are available when needed, while reducing costs and improving overall efficiency. The authors also discuss the importance of inventory control and the need for inventory optimization in order to balance the trade-off between the

cost of holding inventory and the cost of stock-outs. They also mention that implementing an effective inventory management system is crucial for reducing costs, increasing profitability, and improving customer satisfaction. The first step towards a successful implementation is proper planning and setting clear goals. Different types of inventory management systems, such as perpetual and periodic systems, exist and each has its own advantages and disadvantages.

When taken as a whole, these sources emphasize the significance of having a successful inventory management system to enhance business operations and cut expenses. They also emphasize the special difficulties faced by small and medium-sized businesses in inventory management, as well as the role that technology plays in enhancing inventory management procedures. These publications also go over the significance of inventory optimisation and control in supply chain management.

2.2 Challenges of Implementing and Maintaining Inventorization Systems

Utilizing inventory management systems, however, is not without its difficulties. Particularly for small and medium-sized businesses, the initial installation expenses can be substantial. Technology-based inventory management systems can be vulnerable to data security concerns and need specialized knowledge, tools, and resources to install and maintain. Businesses that largely rely on technology for inventory management run the risk of losing credibility if the technology malfunctions, and some staff could be resistant to change and need extensive training to become used to new procedures and systems.

The high cost of setting up and maintaining the system is one of the biggest problems organizations encounter. For small and medium-sized firms, the cost of obtaining and deploying inventory management software, barcode scanners, and RFID tags may be prohibitively high. Additionally, expenditures for continuing maintenance, such as those associated with training staff members to utilize the system, can be substantial. For smaller organizations that do not have the capacity to invest in such a system, these fees can be a substantial hurdle.

The system's intricacy poses another difficulty. Setting up and maintaining inventory management systems calls for a certain level of technological know-how. It's possible that businesses will need to expand their staff or retrain current employees to use the system. This can be a costly and time-consuming operation. For smaller organizations with fewer resources, the level of technical skill needed to build and maintain the system can be a major obstacle.

Ensuring data accuracy is a difficulty that organizations must overcome. Inaccurate inventory records may be the consequence of incorrect data entry, scanning mistakes, and technical difficulties. Stock-outs, overstocking, and sloppy

decision-making can result from this. Moreover, since businesses might not be able to precisely estimate their stock levels, faulty inventory records can also lead to wasted income. In order to keep correct records, it is crucial to regularly monitor and update the system.

The demand for customisation presents another difficulty for businesses. Although inventorization systems have many advantages, they could not always meet the particular requirements of every firm. It may be necessary for organizations to modify the system to meet their unique requirements, which can be a challenging and time-consuming procedure. Smaller organizations that might lack the resources to spend in system modification may find this procedure to be particularly difficult.

One of the biggest problems that educational institutions have with inventory management is a lack of visibility. The majority of educational institutions lack a clear understanding of their inventory levels and consumption trends. This makes it challenging to make wise decisions about inventory management and to prepare for the future. Furthermore, it is challenging to track the whereabouts of goods and determine when objects have been misplaced or lost due to the lack of visibility.

Another difficulty educational institutions encounter with inventory management is inadequate planning. The majority of institutions lack an elaborate inventory management strategy. They frequently use ad hoc methods to manage inventory, which can result in waste and inefficiency. The majority of institutions also lack a clear knowledge of the inventory levels necessary to meet their operational demands. Inventory overstocking or understocking may result from this.

Another issue educational institutions face with inventory management is inefficient operations. The majority of institutions lack standardized inventory management procedures. Ineffective inventory management may result from this, including duplication of effort, incorrect data entry, and a delay in responding to inventory problems.

And lastly, employees may oppose organizations. Some workers could be reluctant to adapt, especially if they are used to manual processes. To assist staff in adjusting to the new system and comprehending its advantages, suitable training and support must be offered. It might take a lot of time and money to complete this training and support, especially for larger organizations.

In conclusion, even though inventorization systems have many advantages for businesses, they can be difficult to adopt and keep up. When establishing and managing these systems, organizations must carefully examine the costs, complexity, data accuracy, customization, and employee resistance. By doing this, they can guarantee that the system satisfies their particular needs and adds the most value to the organization.

The difficulties in creating and maintaining inventorization systems can be overcome in a number of ways. Utilizing cloud-based inventory management solutions is one remedy. These systems don't need expensive gear or software and may be accessed from anywhere. Real-time collaboration is another benefit of cloud-based solutions, which makes it simpler for numerous personnel to view and edit inventory details. This can drastically lower the system's implementation and upkeep costs.

By leveraging a single source of data, organizations can also streamline their inventory management system. The inventory system can be integrated with other business systems, such as accounting, sales, and procurement, to achieve this. Organizations can lower the possibility of data entry errors and increase the accuracy of inventory records by linking the inventory system with other business systems. Additionally, by simplifying the system, it will be simpler to maintain.

Organizations can use routine inventory counts and reconciliations to increase data accuracy. In order to do this, the inventory must be physically counted and compared to the inventory records. Any differences can be looked into and fixed. Regular inventory counts can aid in spotting errors early and lower the likelihood of stock-outs and overstocking.

Utilizing open-source software is another potential answer to the problems associated with developing and maintaining inventorization systems. Open-source software can be updated and improved by the community and is normally free to use. This can drastically lower the system's implementation and upkeep costs.

Finally, businesses can spend money on employee assistance and training. Employees may find it easier to learn and acclimatize to the new system in this way. Organizations can lower employee resistance and increase system adoption by offering proper training and support.

In conclusion, there are a number of feasible answers to the difficulties in setting up and sustaining inventorization systems. The solutions that are most suited for their unique needs and available resources must be carefully considered by organizations. They can guarantee that the system offers the organization the most value by doing this.

2.3 Future of Inventory Management Systems

The field of inventory management has advanced significantly over time, and given how quickly technology is developing, it appears to have a promising future. The Internet of Things (IoT) is one of the burgeoning technologies that has the potential to transform inventory management. IoT allows for the interconnection of various devices, including sensors, to deliver real-time inventory data. This can aid businesses in tracking inventory even more accurately and effectively, ensuring they have the proper supply on hand at the appropriate moment. IoT can assist

businesses in automating their inventory management procedures, which requires less manual labor and increases productivity.

Artificial intelligence (AI) is a different technology that has the potential to enhance inventory management. AI can analyze data, forecast demand, and help businesses decide how much inventory to keep on hand and when to replace their stock. AI may also assist businesses in seeing potential issues before they become serious ones by spotting trends and abnormalities in inventory data, further increasing efficiency and cutting expenses. AI may further assist businesses in streamlining their inventory management procedures by determining the most effective ways to move and store stock, which will further cut costs and boost productivity.

The management of inventories may be significantly impacted by developments in 3D printing technology. Products can be created with 3D printing on demand, eliminating the need for significant stocks. In addition to lowering inventory management expenses like storage and maintenance, this also enables businesses to react more quickly to market demands. With 3D printing, businesses may even tailor products to match particular client requirements, enhancing client happiness and loyalty even further.

The continued development of current technology is also expected to have an impact on how inventory management systems develop in the future, in addition to these emerging technologies. For instance, radio-frequency identification (RFID) technologies and barcodes have already made inventory management and tracking simpler. Processes for inventory management may become even more effective and precise as these technologies develop.

In conclusion, upcoming technologies like IoT, AI, and 3D printing are poised to change the field of inventory management systems, making the future of this industry exciting. Organizations can increase productivity, cut expenses, and better serve customers by utilizing these technologies. The accuracy and effectiveness of inventory management procedures will continue to be improved by ongoing technological development, ensuring that inventory management will play a crucial part in the success of enterprises in a variety of industries.

Systems for inventory management are essential for controlling inventory and enhancing decision-making in businesses. Although installing and sustaining these systems has its difficulties, there are also solutions that can assist firms in overcoming these difficulties. A few of the options that can help businesses enhance their inventory management system include computerized inventory systems, cloud-based systems, connection with other corporate systems, and routine inventory counts.

The future of inventory management systems appears bright as technology advances. Several cutting-edge technologies, including the Internet of Things,

artificial intelligence, and 3D printing, have the potential to completely transform inventory management. To ensure the accuracy and effectiveness of their inventorization system, firms must routinely assess their inventory management system and make modifications where appropriate. By doing this, businesses can enhance their planning, financial reporting, and decision-making processes, ultimately increasing their success.

3 Analysis of existing systems

Any project must begin with a thorough examination of the existing systems in order to determine their strengths, shortcomings, and potential areas for development. In order to learn more about the methods, tools, and features already in use in the industry, it is crucial in the framework of this project to analyze the inventory management systems now in use in university housing.

Tracking, organizing, and controlling inventory items inside an organization is a complicated process known as inventory management. Inventory management is essential for ensuring that the resources needed for academic activities, such as lectures, practicals, and staff housing, are accessible in the setting of university housing. An effective inventory management system can reduce instances of equipment or facility shortages, increase operational effectiveness, and optimize resource allocation and usage.

It is crucial to perform a thorough examination of existing systems in order to create a client-server application for inventory management in university housing that works well. This analysis will aid in our comprehension of both the difficulties and constraints that might exist in actual use, as well as the functionality and features that are frequently incorporated.

Reviewing and assessing a variety of inventory management systems currently in use in university housing will be part of the investigation. In this article, we'll look at a number of different facets of these systems, such as the technology utilized in their creation, their user interfaces, their functions, and their integration with other systems or databases. By examining these systems, we hope to pinpoint the essential elements and traits required for effective inventory management in the context of university housing.

The examination will also include a critical assessment of the advantages and disadvantages of these current systems. To identify the areas where improvements may be made, we will evaluate variables including usability, scalability, security, and performance. We can create a more robust and efficient solution that tackles these problems by understanding the constraints and difficulties that currently used systems encounter.

The investigation will also look into the possibility of incorporating cutting-edge frameworks and technologies like ASP.NET, ReactJS, and AWS into the inventory management system. We can improve the system's scalability, usability, and security as well as streamline the deployment and hosting procedures by utilizing these technologies.

Overall, our client-server application's development relies heavily on the examination of the inventory management systems now in use in university housing. We can create a solution that successfully handles the particular problems encountered in the context of university housing by studying the

benefits, drawbacks, and limitations of existing solutions. The investigation will offer insightful information on technologies, functionality, and best practices, enabling us to create a complete and ground-breaking inventory management system that satisfies the unique requirements of the users.

3.1 Manual System Analysis

The university's manual inventory management system is based on paper records. This system makes extensive use of human record-keeping and data entering. The records are kept in logbooks and files, which are vulnerable to loss, destruction, and mistakes. The records are updated manually, and the inventory is tracked manually. This technique is time-consuming and inefficient, making it difficult to effectively manage the inventory.

The manual process is also prone to mistakes. The manual data entry procedure used in the manual system increases the likelihood of errors. Incorrect records created by humans might result in improper inventory management. Additionally, this method makes it challenging to prepare reports because it takes a lot of manual labor to gather data from many sources.

3.2 Automated System Analysis

The institution has put in place a partially automated system for managing inventory. The system is used to manage and track the inventory electronically. The system offers real-time data on the inventory as well as the ability to generate reports. This technique, however, has significant drawbacks.

The inability to track the movement of items is a result of the lack of a centralized inventory tracking system. There may be a disparity between the real inventory count and the recorded count, for instance, if an item is transferred from one room to another and isn't registered in the inventory system.

Essential components needed for effective inventory management are missing from the automated system in use today. It is difficult to manage the inventory properly since the system cannot track the inventory accurately. Making judgments on the inventory is challenging because the system does not provide sufficient information on its status.

The inability to deliver real-time inventory data is another significant drawback of the present inventory management system. The real inventory count and the recorded count differ because the inventory count is manually updated. Stock-outs or overstocking may result from this delay, which might cause erroneous inventory figures [10].

Additionally, the system offers a basic reporting function. The system generates simple reports, and it is difficult to get the system to provide more

precise information. The system's utility for inventory management is further constrained by its inability to interpret the data.

The fact that the current automated method is not user-friendly is another important constraint. The technology is complicated, and using it efficiently takes substantial training. Because of this, it is challenging for users who are unfamiliar with the system to use it effectively. The inventory management process is less effective since the system is slow and takes a long time to process data.

The manual inventory management system has a number of drawbacks as well as shortcomings, including some hazards. These dangers include losing something as a result of human error, theft, or damage. It is challenging to pinpoint the origin of any losses or damages in the absence of a centralized inventory control system.

Overall, there are a number of shortcomings in the current system that make the inventory management process less effective. The automated system lacks crucial characteristics needed for effective inventory management, while the manual approach is ineffective, prone to errors, and challenging to manage. A new inventory management system is therefore required to get around these restrictions and offer a more effective and efficient approach to handle goods.

There are a variety of inventory management software alternatives on the market today, each with unique features and functionalities. Zoho Inventory, QuickBooks, and Cin7 are three well-liked solutions for inventory management software.

Inventory Zoho: Businesses may manage their inventory, orders, and sales with the use of the cloud-based inventory management tool Zoho Inventory. The ability to track inventory levels in real-time, which enables users to prevent stockouts and overstocking, is one of Zoho Inventory's important advantages. Additionally, it provides multi-channel selling, which enables companies to control sales from several channels including Amazon, eBay, and their own website from a single platform. Order management, automated workflows, and reporting are further functions.

Small and medium-sized enterprises, especially those who conduct online product sales, should use Zoho Inventory. There are paid plans available for companies with bigger order volumes as well as a free plan for those with up to 20 monthly online orders. One drawback of Zoho Inventory is that some users have complained that it can be challenging to use.

QuickBooks: QuickBooks is a well-known accounting program with capabilities for inventory control. Its inventory management tools allow you to manage sales and purchase orders, track inventory levels, and produce reports. Additionally, QuickBooks enables connections to e-commerce sites like Shopify and Amazon, enabling users to sync their inventory levels between several sales channels.

Small and medium-sized enterprises that require inventory management and accounting software should use QuickBooks. It provides a range of plans, including advanced, small company, and self-employed programs. The fact that QuickBooks' inventory management features fall short of those of specialized inventory management programs like Zoho Inventory is a drawback.

Cin7: For companies that must manage inventory across many channels, Cin7 is a cloud-based inventory management system. Inventory management, order management, and fulfillment are some of its capabilities. Along with that, it provides interfaces with several e-commerce platforms and accounting programs. Cin7's capacity to handle intricate inventory procedures like dropshipping and cross-docking is one of its main benefits.

Small and medium-sized companies can use Cin7 to manage inventories across several sales channels. It provides a range of plans, including startup, growth, and premium options. Cin7 can be more expensive than other inventory management programs, especially for companies with high order quantities. This is one drawback.

It's critical to take a business's unique requirements into account when contrasting these three inventory management software alternatives. Businesses who already use other Zoho applications and require a straightforward, user-friendly inventory management solution would find Zoho Inventory to be a good fit. Businesses that are largely focused on accounting but want some fundamental inventory management functions may find QuickBooks to be a good fit. Businesses who sell through several channels and require a comprehensive solution for managing inventory and fulfillment across those channels may find Cin7 to be a good fit.

While Cin7 offers a more specialized pricing plan based on certain business needs, Zoho Inventory and QuickBooks offer tiered pricing plans based on the number of users and features required. Businesses must weigh the costs and advantages of each software choice to choose the one that best suits their requirements and financial constraints.

In conclusion, each of these inventory management software programs has particular advantages and characteristics. Small and medium-sized firms that sell products online should use Zoho Inventory, whereas those that need accounting and inventory management software should use QuickBooks, and those who need to handle complicated inventory workflows across several channels should use Cin7. In the end, the particular requirements of the firm will determine which inventory management software is used.

4 Methodology

The client-server application created for this project is meant to address the problem of inventory management in university housing, particularly in practice and lecture classrooms, staff housing, and other related areas. By creating a system that can effectively manage the university's inventory, the primary objective is to avoid instances of allocated classrooms being devoid of the necessary facilities and equipment. The application is created using ReactJS for the front-end and the ASP.NET framework based on C# for the back-end, which is coupled to a PostgreSQL database. The application's features include the ability to add, remove, and edit classes and items.

It is impossible to overestimate the significance of inventory management in any business, but it is especially vital in educational settings where efficient resource allocation is essential for both teaching and learning. Students' learning experiences and academic performance may be significantly impacted by inadequate facilities and resources in classrooms. This client-server program is being created with the intention of addressing this issue by offering a dependable and effective mechanism for managing inventory in university housing. The application is designed with the newest frameworks and technologies, guaranteeing its scalability, security, and usability.

For the back-end portion of the program, the ASP.NET framework based on C# was selected because of its high-performance capabilities, flexibility, and resilience. Additionally chosen for its dependability, scalability, and sophisticated capabilities is the PostgreSQL database. ReactJS, a well-known front-end toolkit that enables the building of incredibly responsive and interactive user interfaces, is used to build the front end.

The application's features include the ability to add, remove, and edit classes and items. This will give inventory management a central location, guarantee that each classroom has enough supplies, and stop instances of equipment shortages. A front-end, a back-end, and a database make up the system architecture, which ensures smooth data flow and effective performance.

In conclusion, the inventory management issue in university housing is addressed by this client-server application. The system offers a central location for controlling inventories, ensuring that each classroom has enough supplies, and avoiding instances of equipment shortages. The program is made to be scalable, safe, and simple to use by utilizing the most recent technologies and frameworks.

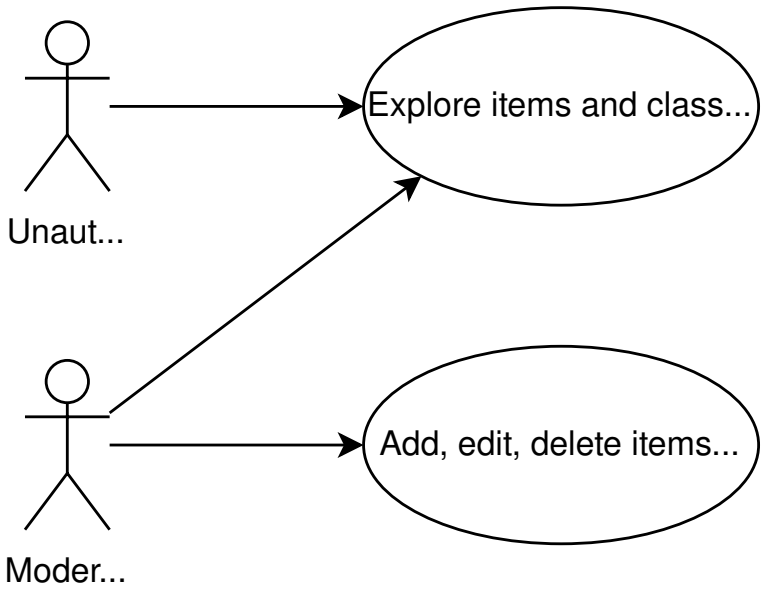

Figure 1.1 Unauthorized user and Moderator Use case diagram

Figure 1.1 describes the use case of the unauthorized user and moderator. Unauthorized users can only explore, and the Moderator can both explore the website and add edit and delete items in classrooms.

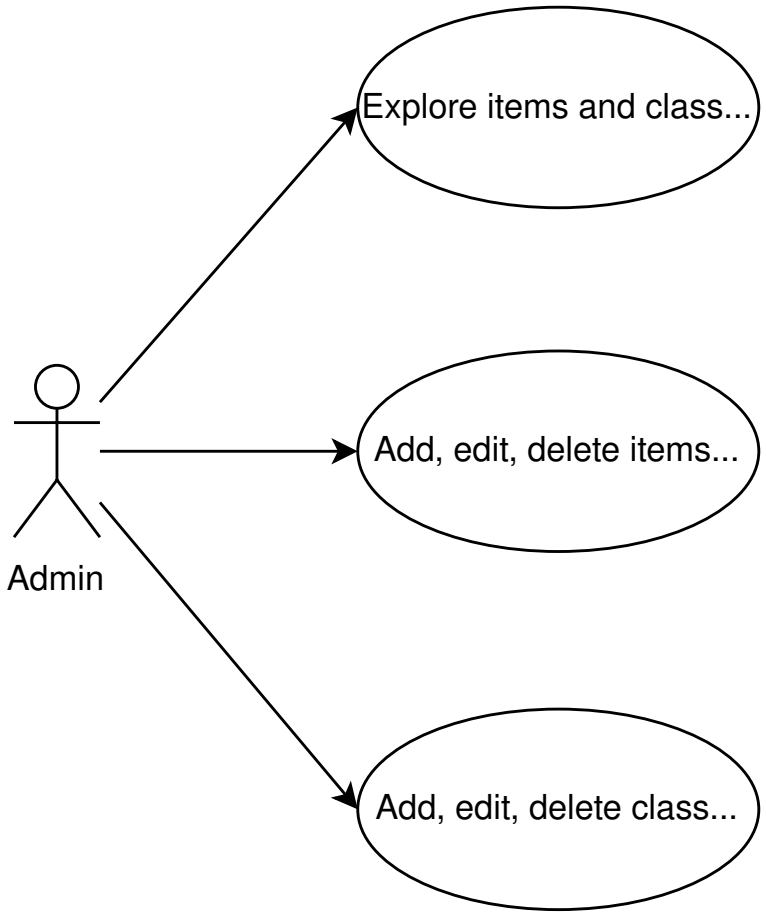

Figure 1.2 Admin Use case diagram

Figure 1.2 illustrates the use case of the Admin. Admin have the same rights as the moderator has, but in addition can do operation on classrooms too.

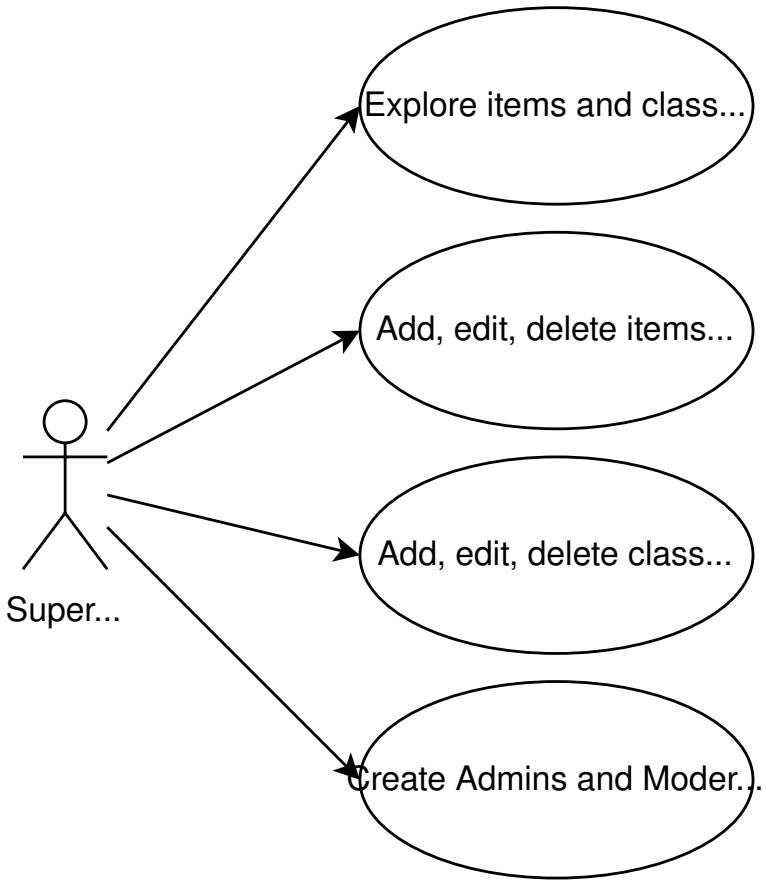

Figure 1.3 Super admin Use case diagram

Figure 1.3 shows the use case of the Super admin. Super admin have the same rights as the admin has, but in addition can create previous types of the accounts(moderator and admin).

A use case diagram is a visual representation of how users or other systems interact with a system. It presents the system's functional needs from the viewpoint of its users. Software developers frequently employ use case diagrams to visualize and explain a system's intended functionality.

A use case diagram's main components are actors, use cases, and the connections between them. Here is an explanation of each component:

Actors: Actors play the roles of the users or other systems that interact with the modeled system. A human, another system, or even a piece of equipment can be an actor. Stick figures or other icons are used to represent actors. They are linked to use cases by lines known as affiliations, which show their participation in carrying out particular actions.

Use cases: Use cases reflect particular system interactions or functionality. From the viewpoint of the user, they describe how the system behaves. Each use case denotes a specific functional component. Use cases are shown as ovals and are associated with actors.

A connection between an actor and a use case is represented by an association.

It indicates that the actor is involved with the use case or engages with it in some way.

Generalization/Inheritance: A triangle-shaped arrow pointing from a parent use case to a child use case indicates inheritance. It implies that the parent use case's behavior is passed down to the child use case.

Include: An open arrowhead on a dashed arrow designates an include relationship. It means that one use case makes advantage of another use case's features.

Extend: Like the include relationship, an extended relationship is displayed as a dashed arrow with an open arrowhead. However, it shows that under specific circumstances, one use case increases the functionality of another use case.

Use case diagrams give a broad overview of the functionality of the system and its interactions with different actors. They aid in comprehending the needs of the system, locating possible participants, and determining its extent. Use case diagrams can also be helpful for confirming that requirements are complete and directing the development process because they act as a starting point for additional analysis and design tasks.

4.1 System architecture

To handle inventories at the university housing, the system consists of a client-server application. ReactJS, a well-known front-end package that enables constructing user interfaces, is used to develop the client-side. The C#-based ASP.NET framework is used to implement the back-end. To store and retrieve data, the back-end interfaces with a PostgreSQL database. HTTP calls are used to communicate between the front end and the back end.

The front-end, back-end, and database are the three key parts of the system. The user interface for dealing with the system—including creating, modifying, and deleting classrooms and inventory items—is provided via the front-end. Requests from the front end are handled by the back end, which also retrieves data from the database and returns it to the front end. All of the data regarding the inventory and classrooms is kept in the database.

The networking layer, which permits communication between the client and server, is a part of the system architecture in addition to the primary components. The front-end can send requests to the back-end and get results from the server thanks to HTTP protocol-based communication. The security procedures that guarantee the confidentiality and integrity of the data being transferred are also part of the networking layer.

The data storage layer, which comprises of the PostgreSQL database, is also part of the system architecture. The database is in charge of maintaining the integrity of the data as well as storing and retrieving data. It also maintains

data about the classrooms, including the room number, capacity, and technical amenities, as well as data about the inventory, including the quantity of products, location, and status [11].

The middleware layer, which serves as a link between the front-end and the back-end, is also a part of the system architecture. The HTTP requests and answers are handled by the middleware layer, which is also in charge of parsing the data and sending the requests to the back-end. Additionally, it manages server answers and formats data before transferring it to the front-end.

The application logic layer, which is in charge of handling the system's business logic, is also a part of the system architecture. It outlines the policies and guidelines for managing the inventory, such as how to add, amend, and remove classes and inventory items. The methods for figuring out which resources are available in each classroom and avoiding equipment shortages are also part of the application logic layer.

In conclusion, the modular, scalable, and secure system architecture of the inventory management application is designed. It has several levels that combine to provide a strong and effective system for controlling the inventory in the university housing. The system architecture makes sure that resources are allotted effectively and that the inventory is maintained correctly, increasing the overall effectiveness of the university housing.

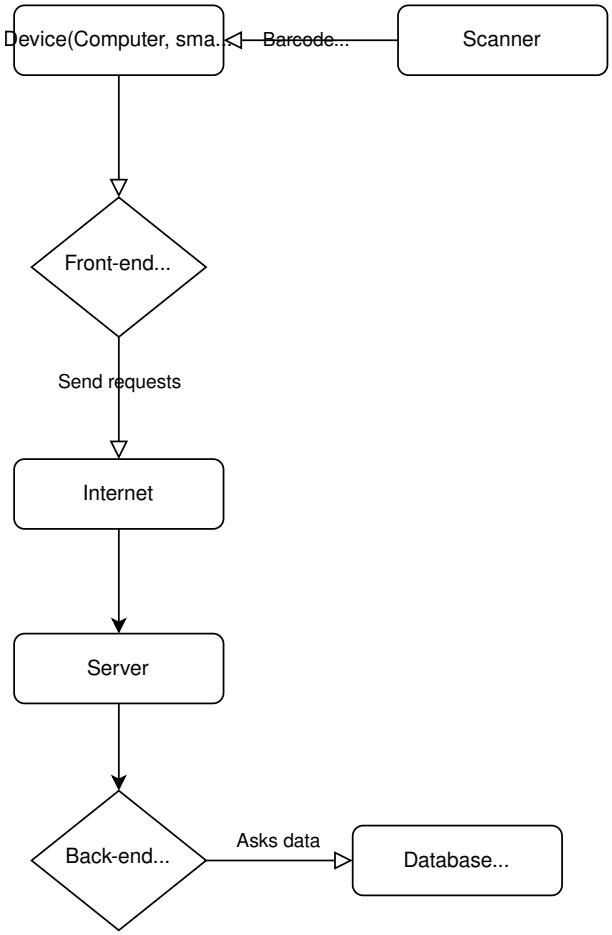

Figure 1.4 High level system overview diagram

Figure 1.4 above explains the High level overview of the system including the stack that have been used in the development of the inventorization system for the university.

A description of a system that offers a thorough grasp of its elements, operations, and relationships is known as a high level system overview. It frequently takes the form of a diagram or flowchart that shows the key elements of the system and how they interact.

The numerous parts that make up a system, such as its hardware and software components, data flow, user interface, integration points, security, and business processes, are all detailed in a high level system overview. Servers, databases, and applications are some of the system components that cooperate to make the system work. A flowchart or diagram that depicts how data goes through the system from input to output and how it is processed or converted along the way serves as an example of the data flow. The way people engage with the system, for as through a web interface or a mobile app, is how the user interface is described. In order to maintain a smooth information flow, the integration points identify other systems or services that are integrated with the system, such as payment gateways or shipping providers. The system's authentication, authorisation, and encryption processes are described in the security component in order to safeguard data and thwart illegal access. The workflows and procedures supported by the system, such as order processing or inventory management, are included in the business processes component. An extensive understanding of the system, its capabilities, and its constraints is given to stakeholders in a high-level system overview, assisting in conversations, decision-making, and problem-solving.

A high level system overview's goal is to give stakeholders a comprehensive understanding of the system's operation and objectives. It can be used to speed up discussions and decision-making, spot possible problems or opportunities for development, and explain the system's strengths and weaknesses to others.

4.2 Components

Front-end Component

The system's user interface is handled by the front-end component. It is made up of a number of modules that interact with one another to offer a seamless user experience. The following are the front-end component's key modules:

Each module interacts with the others in the system's architecture, which was created with modularity in mind to offer a smooth user experience. The system's user interface is rendered by the ReactJS module, which is assisted by the HTTP Calls module, which uses HTTP calls to the back-end to fetch and transmit data. The UI Design module is in charge of creating the system's user interface,

and it makes use of CSS and Bootstrap to make sure that the components are suitably styled and responsive. Together, these components make sure that the user interface is both useful and aesthetically pleasing.

The login module ensures a secure login process for users by validating their login information before granting them access to the system. It has a login button, a password input field, and an email input field. Before the user can continue, the email and password input fields are checked to see if they comply with the requirements.

The user can create, view, edit, and delete classrooms using the classroom module. It comprises a form where the user may enter the name, location, capacity, and other pertinent information about the classroom. The user can click on a specific classroom to change its details after viewing a list of all the available classrooms. A classroom can also be removed from the system by the user.

The user can control the inventory in each classroom using the inventory module. In each classroom, the user has the ability to add, view, edit, and delete items. The inventory module has a form on which the user may enter the name, description, amount, and other pertinent information about the object. The user can click on a specific item to update its details after viewing a list of all the things that are currently available in each classroom. A system item may also be deleted by the user.

The user can create reports on the inventory in each classroom using the reports module. The user can produce reports on the goods that are available in each classroom, how often they are used, and other pertinent information. The form in the reports module lets the user choose the parameters for creating the report. The report is then either downloaded as a PDF file or displayed on the screen.

Overall, by enabling the user to effectively manage the inventory in each classroom, the front-end part of the system offers a smooth user experience. It offers all the necessary functionalities for controlling the inventory in the university housing while being user-friendly and aesthetically pleasing.

Back-end Component

The system's business logic is handled by the back-end component. It is made up of many modules that cooperate to manage the data and carry out different actions. The following are the primary modules of the back-end component:

Three core modules make up the system's back-end: a C# module, a PostgreSQL database module, and a REST API module. The C# module is in charge of implementing the system's business logic, using a number of sub-modules to manage data and carry out activities. The system's data are managed by the PostgreSQL database module, which also ensures data consistency and

integrity. The REST API module uses ASP.NET to establish a RESTful API that the front-end component may use, exposing the system's capabilities as a REST API. These modules function flawlessly as a unit to give the system a stable and scalable back-end.

The C# module is made up of numerous sub-modules that deal with various areas of the business logic of the system. The task of maintaining classroom-related data, including the addition, modification, and deletion of classrooms, falls under the purview of the Classrooms module. The Inventory module is in charge of maintaining the university's inventory, which includes adding, updating, and removing goods. Only authorized users are able to access the system since the user authentication process is handled by the authentication module. Before any input data is recorded in the database, it is validated by the Validation module, which manages this procedure.

The PostgreSQL database module is in charge of looking after the system's data. It keeps track of all the system's data, including user, inventory, and classroom data. Various methods, including constraints, triggers, and foreign keys, are used to assure data consistency and integrity. Additionally, it offers the system a scalable and trustworthy data storage solution, ensuring that the data can be accessed promptly and effectively. The REST API module uses ASP.NET to establish a RESTful API that the front-end component can access in order to expose the system's capabilities as a RESTful API. The front-end component can interact with the system through a variety of endpoints provided by the API, including APIs for adding, modifying, and removing classrooms and items. Additionally, it offers endpoints for data retrieval, user authentication, and other system-related tasks. The front-end component and API communicate using HTTP requests and replies, making the system usable from any device with an internet connection.

Barcode Scanning Component

The system's barcode scanning component is essential since it makes inventory tracking precise and efficient. The inventory items' barcodes must be scanned by the barcode scanning component. It consists of a USB-connected barcode scanner for the computer system. To update the inventory, the barcode scanner delivers the scanned data to the front-end component, which then sends it to the back-end component. The component makes use of barcode scanning technology, which has become a popular way for many sectors to track inventory. When an item is scanned with the barcode scanner, which is connected to the front-end component, data is passed to the back-end component for processing. The database's inventory is then updated via the back-end component, making sure it is current and accurate. The system's productivity is increased by the barcode

scanning component since it removes the need for human data entry, lowers the possibility of errors, and guarantees correct and current inventory management.

4.3 Hosting Component

The system is kept on the university's servers by the hosting component. It is made up of a number of smaller parts that work together to install the system on the servers. The following are the primary hosting component sub-components:

To make sure that the system is correctly distributed and accessible to users, the Deployment Module and Server Module collaborate closely [12]. Using platforms like Jenkins and Docker, the Deployment Module automates the deployment process. By making sure the system is implemented regularly and dependably, it lowers the possibility of mistakes or discrepancies. The Server Module is in charge of overseeing the servers that house the system. It guarantees that the system is available to users and that the servers are up and functioning. The Server Module also makes sure that the servers are safe and that they deliver the performance standards set by the system. These two modules work in tandem to make sure the system is deployed and accessible to users in a secure and dependable manner.

4.4 Security Component

The security element is in charge of making sure the system is secure. It is made up of a number of smaller parts that work together to defend the system from different threats. The following are the primary security component sub-components:

Various components keep the system's security up to date [13]. Using tools like JWT and OAuth, the Authentication Module is in charge of making sure that only authorized users can access the system. Users' permissions are managed by the authorization module, which makes sure they can only access the areas of the system to which they have been granted access. The Encryption Module employs encryption methods like AES and RSA to safeguard sensitive data. The system ensures that user data is secure and only accessible by those with the appropriate authorisation by integrating these modules.

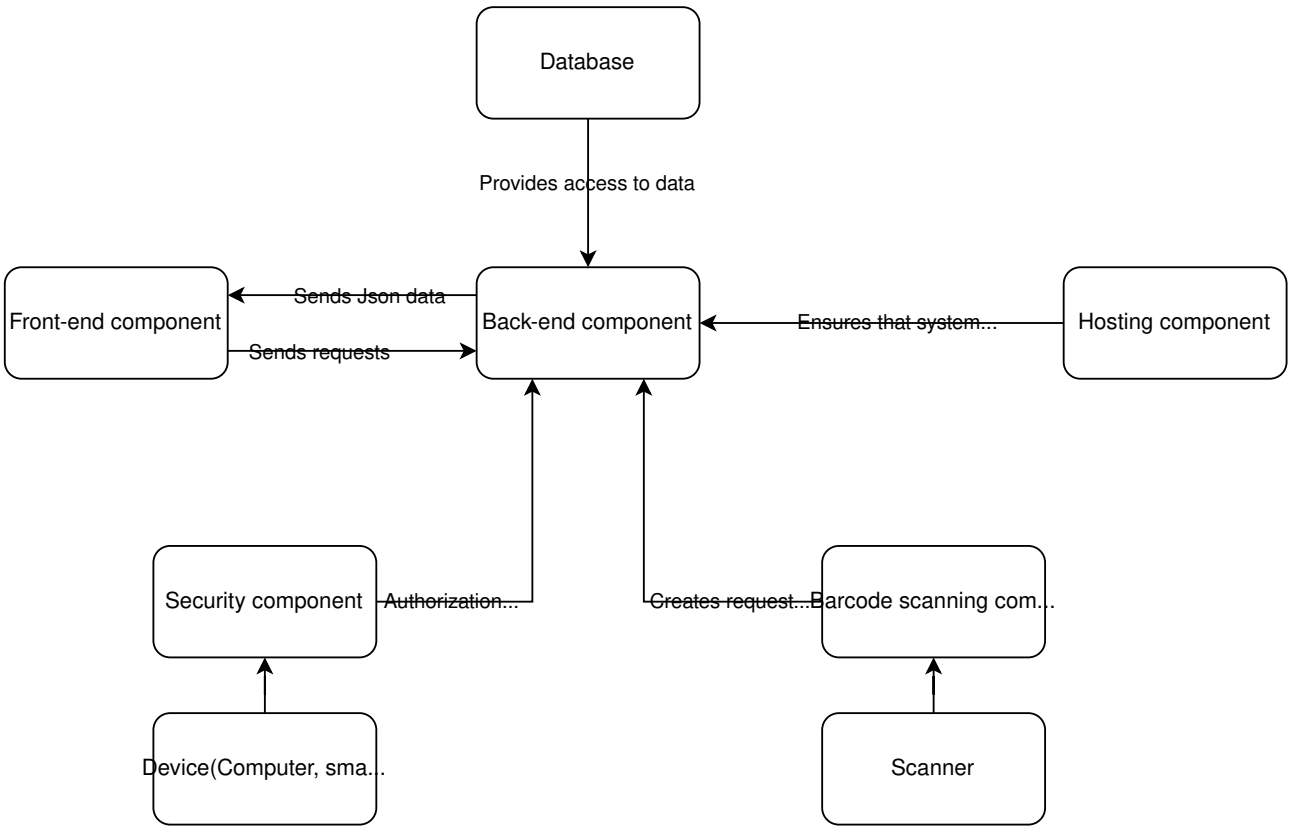

Figure 1.5 Component interaction

Figure 1.5 shows the interaction between components that were described above

The interactions and links between software components or objects within a system are shown visually in a component interaction diagram, sometimes referred to as a communication diagram or collaboration diagram. It highlights the dynamic features of a system by showing how parts or objects work together to produce a certain capability or behavior.

A component interaction diagram places more emphasis on message exchange than internal structure of the components or objects. During the execution of a use case or scenario, it records the runtime behavior and communication flow.

The primary components and ideas of a component interaction diagram are broken down as follows:

Components/Objects: Software elements that interact with one another to complete a job or accomplish a goal are represented by components or objects. They are shown as rectangles or icons that have their names written on them.

Lifelines: Lifelines show the instances of the various items or components that are involved in an interaction. Vertical lines extending from the component or object symbol are used to represent them. Lifelines display a component or object's presence and longevity throughout the interaction.

Messages: Messages are a representation of the interaction or communication between parts or things. They are shown as arrows that have labels that describe the type of communication (such as method calls, events, or signals). Lifelines

are connected horizontally, and messages flow between them to show the order of interactions. Asynchronous (non-blocking) or synchronous (blocking) messages are both possible.

Activation boxes: Also called activation bars or execution occurrences, activation boxes show the duration of time that a component or object is actively processing or carrying out a message. They represent the length of an operation or method call and are displayed as vertical rectangles on a lifeline.

Return messages show the reactions to the outcomes of an earlier message. They are shown as dashed arrows flowing back to the caller, which indicate the method call return path.

Collaborations: Collaborations show how various parts of a system interact and relate to one another. They offer a comprehensive view of how several elements or objects interact to carry out a certain action or behavior.

Component interaction diagrams are helpful for studying the interactions between components or objects, detecting message flows, and comprehending the behavior of a system during runtime. They can aid in the planning and improvement of inter-element communication as well as the identification of possible problems or bottlenecks in the architecture or design of the system.

4.5 Data flow

The user's input on the front-end interface initiates the flow of data within the system. To build, remove, and change classes and items, the user can enter data. Using HTTP requests, the ReactJS-built front end communicates with the back end. The HTTP calls are received by the back-end created with the ASP.NET framework, which then processes the data appropriately. The PostgreSQL database is then used to store the data. The data necessary for the system to operate must be stored and retrieved by the database.

When a user asks data from the system, the back-end receives the request and transmits the requested data to the front-end via HTTP calls after retrieving it from the database. The user is subsequently presented with the desired data via the front-end.

The request is sent from the front-end to the back-end whenever a user wishes to add, remove, or change a classroom or an item. After processing the request and carrying out the required operations on the database's data, the back-end replies to the front-end, letting it know whether the operation was successful or not.

The system's internal data flow is essential to the application's smooth operation. Each component's input and output are closely correlated, therefore any interruptions in the data flow might result in mistakes and problems for the system [14]. Data management and retrieval are more efficient and dependable by

structuring the processing and storing of the data.

The hosting component, which is crucial in assuring the availability and accessibility of the system's data, is also included in the data flow. The hosting element makes certain that the system is set up and operating on the servers, enabling users to access and modify data as necessary.

Security measures that guarantee the availability, confidentiality, and integrity of data are also necessary for the data flow. To prevent unauthorized access and preserve sensitive data, the system includes security mechanisms such user identification, role-based access control, and encryption.

Overall, a key factor that affects the application's effectiveness, dependability, and security is the data flow within the system.

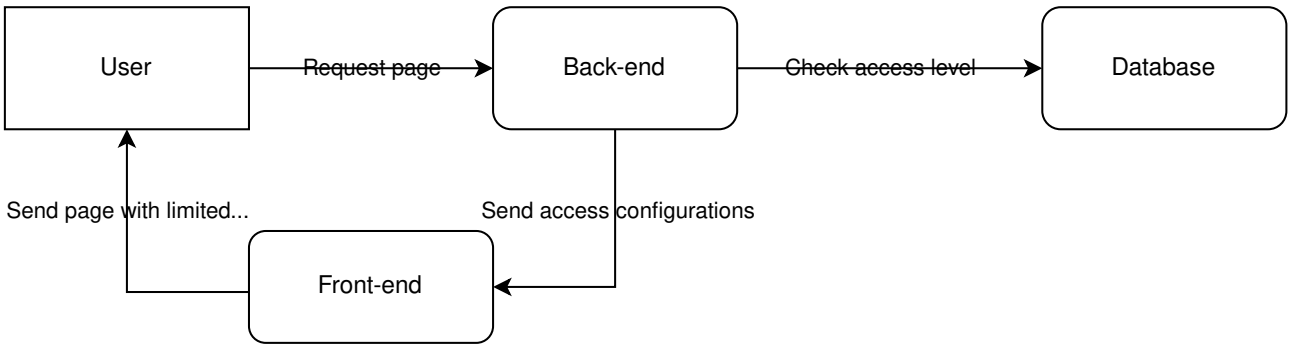

Figure 1.6 Data flow 1 diagram

Figure 1.6 is the data flow of the page request action.

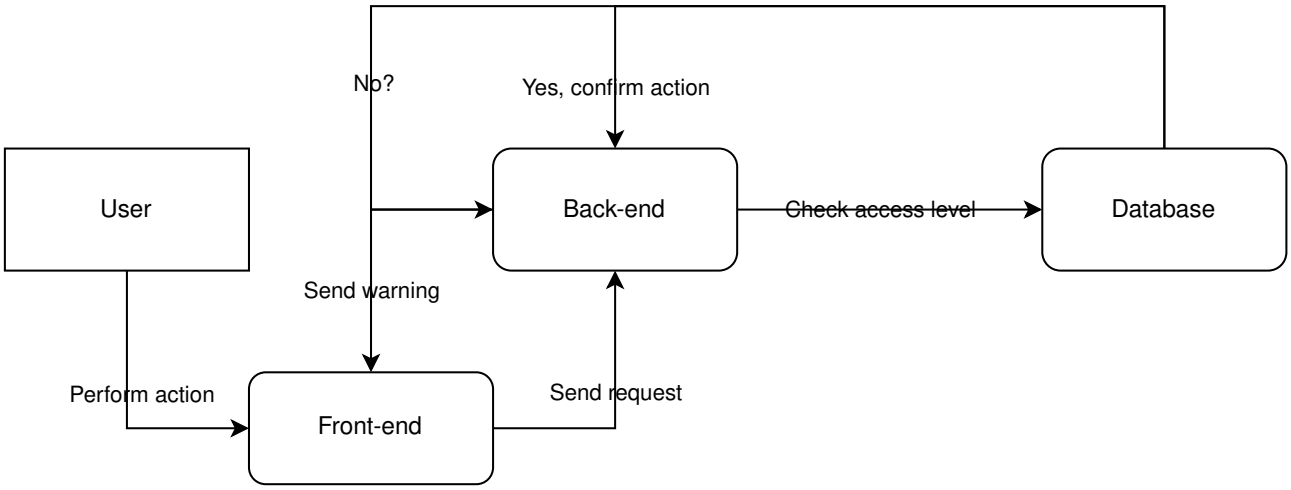

Figure 1.7 Data flow 2 diagram

Figure 1.7 is the data flow of the add, edit and delete action

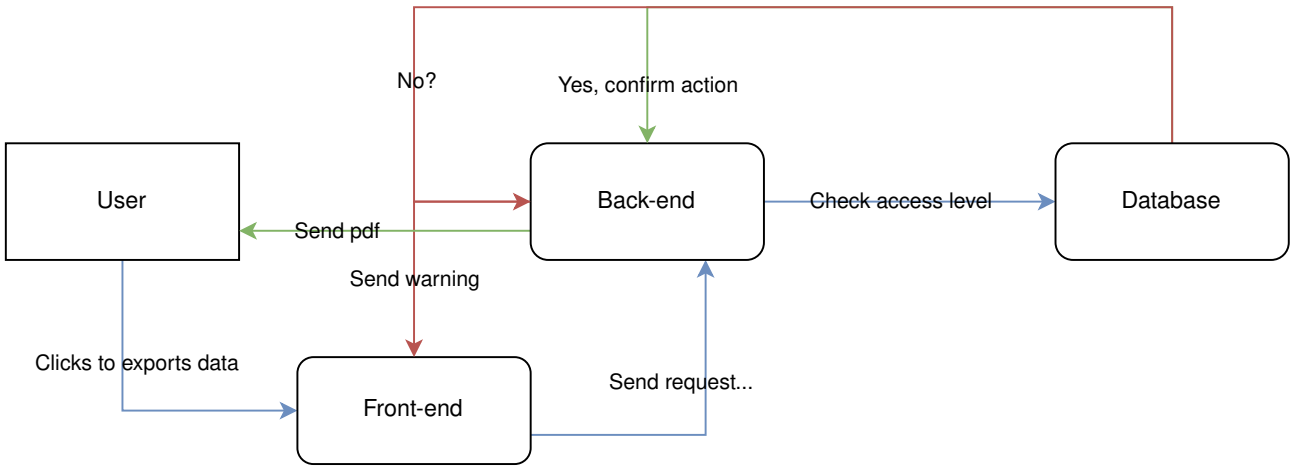

Figure 1.8 Data flow 3 diagram

Figure 1.8 describes the data flow of the data exporting case.

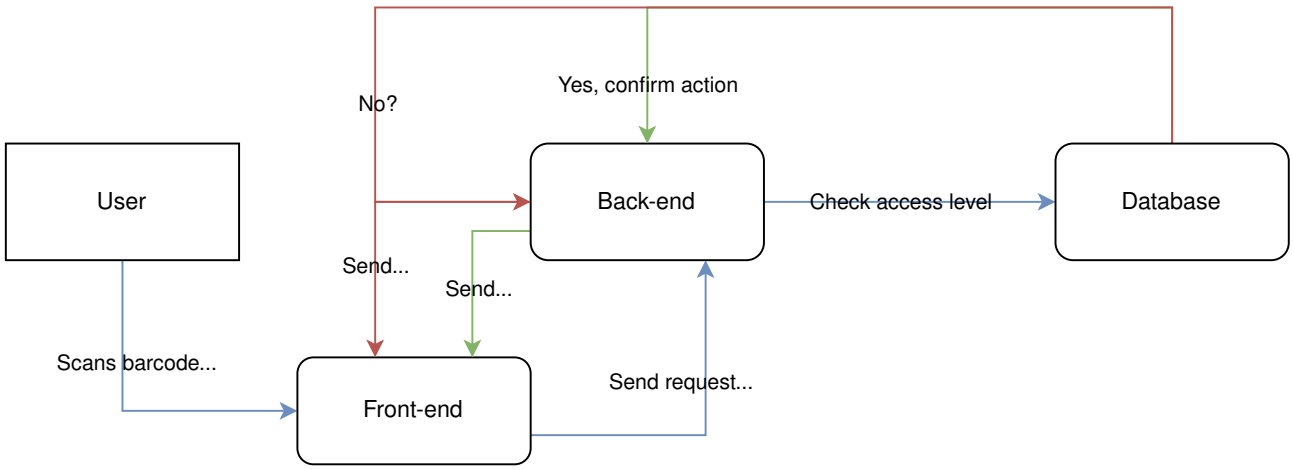

Figure 1.9 Data flow 4 diagram

Figure 1.9 is the data flow of performing action to multiple objects by scanning barcodes.

A data flow diagram (DFD) is a graphic that shows how data moves through a system. It demonstrates the flow of data between processes, how it is stored or converted, and how different system elements interact with one another. To model and comprehend the data flow and logic of a system, DFDs are frequently used in system analysis and design.

The essential elements and ideas of a data flow diagram are broken down as follows:

Processes: The functions or actions that change or manipulate data within the system are represented by processes. They could involve computations, data transformations, validations, or other processes applied to input data to create output data. Processes are shown as rectangles or circles with labels describing their roles.

Data Flows: Data flows show how information is moved around processes, outside entities, and data repositories. They serve as an example of the transmission of information from one component to another. Data flows are shown as arrows with labels that identify the kind of data being transported.

Data sources or destinations that interact with the system but are not a part of it are known as external entities. They can be users, external systems, or other entities that interact with the system and send or receive data. Rectangles with labels are used to represent external entities.

Data Stores: Data stores are the locations in the system where data is kept. They could be files, databases, or any other kind of storage system. The persistence of data within the system is demonstrated by data stores, which are represented as rectangles with labels.

Labels for Data Flow: Data flow labels offer additional details about the sent data. They provide a description of the data's content, structure, or format, aiding in the clarification of its meaning and use.

Context Diagram: The highest-level DFD that offers a system-wide overview is the context diagram. It represents the interactions between the system and external entities and depicts the system as a single process. Understanding the system's boundaries and domain is made easier by the context diagram.

DFDs can be divided into several levels of detail through decomposition. The context diagram can be developed into more intricate diagrams as the system is further examined, showing new data flows and subprocesses. It is possible to comprehend the system's functionality at a more detailed level thanks to this hierarchical deconstruction.

Clear understanding of the data flow and processing logic within a system is facilitated by data flow diagrams. They support the process of identifying data inputs, outputs, and transformations and reveal potential bottlenecks or design flaws. DFDs are used to make sure that data is handled correctly and flows within the system easily during system analysis, requirements collection, and communication with stakeholders.

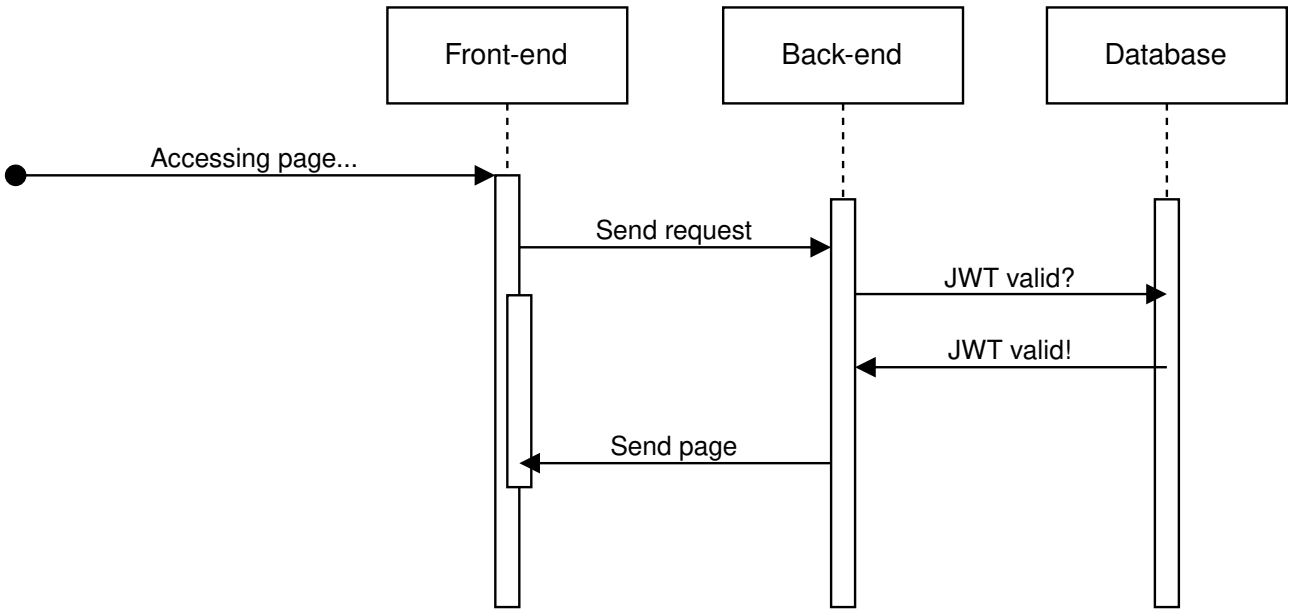

Figure 2.1 Sequence diagram 1

Figure 2.1 This sequence diagram shows the sequence of actions done in order to access the page via JWT token.

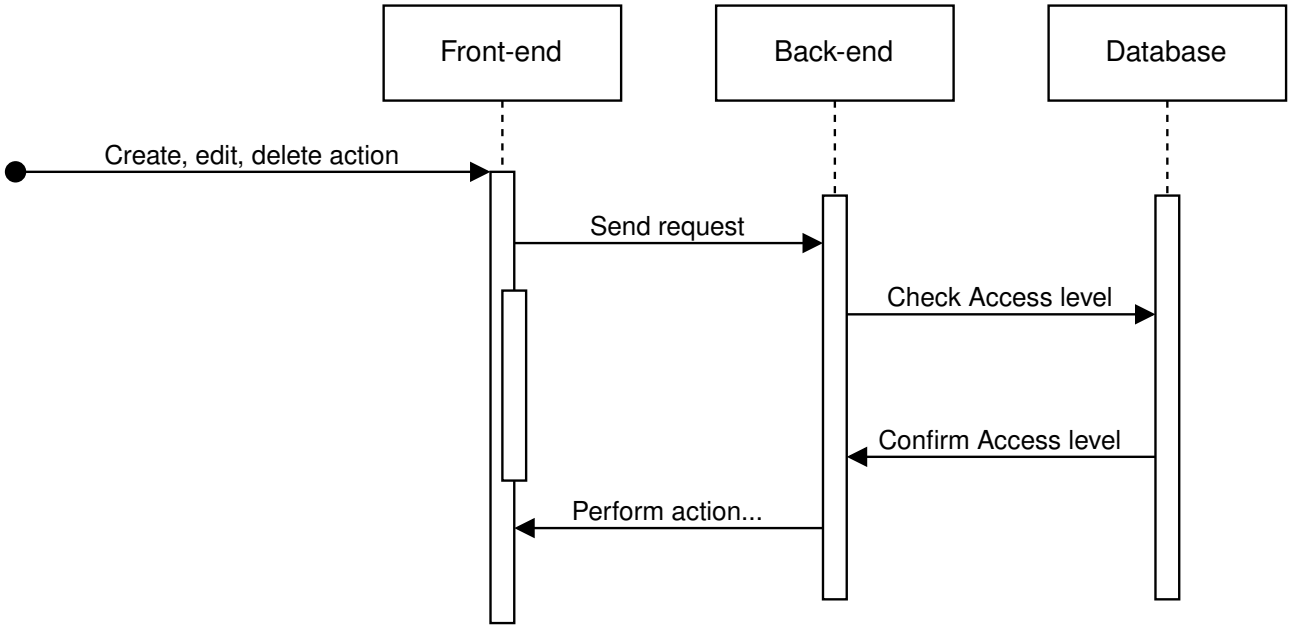

Figure 2.2 Sequence diagram 2

Figure 2.2 The sequence diagram above illustrates the sequence of actions done in order to manipulate(add, edit, delete the classrooms or items)

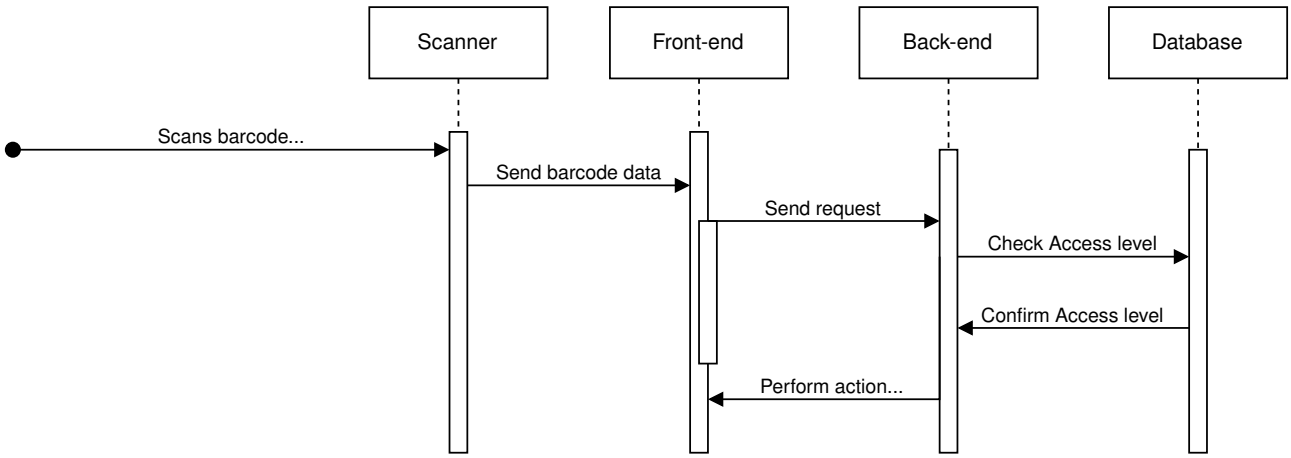

Figure 2.3 Sequence diagram 3

Fig 2.3 This sequence diagram shows the sequence of actions done to objects.

A sequence diagram is a sort of interaction diagram that depicts the communications and interactions between system elements over a predetermined time period. It emphasizes the chronological flow of events and the communication between many entities, showing how several items work together to complete a task or scenario visually.The essential elements and ideas of a sequence diagram are broken down as follows:

Lifelines: The elements or objects participating in the interaction are represented by lifelines. The name of the object or component is written on top of the vertical lines that represent them. Lifelines display an object's existence and lifetime across the course of the sequence diagram.

Activation Boxes: Also referred to as execution occurrences or activation bars, activation boxes show the duration of time that an object is actively carrying out a method or operation. On the lifeline, they are shown as horizontal rectangles or bars that show when an object is acting in a particular way.

Messages: Messages show how items interact or communicate with one another. They show how requests or information travel from one object to another. Arrows that point in the direction of communication and start at the sender's lifeline and stop at the receiver's lifeline are used to indicate messages. The goal or content of the message might be stated in the label of the arrow.

Return messages: A return message shows a previous communication's answer or outcome. They are shown as arrows that point backward from the sender to the receiver, showing the communication's return path.

Self-Invocation: When an object calls a method or action on itself, this is known as self-invocation. It is shown in a sequence diagram by a message arrow that circles back to the same lifeline.

Combined Fragments: In a sequence diagram, combined fragments are used to depict alternate routes, loops, or conditional statements. They enable

the depiction of intricate control flow scenarios, including parallel execution, conditionals, and loops.

Sequence diagrams offer a live perspective of the interactions and teamwork that take place between several items or components during a specific scenario or use case. They aid in comprehending the sequence of events, message transmission, and overall control flow inside a system. Sequence diagrams are useful for developing, describing, and evaluating a system's behavior as well as for identifying potential problems or communication bottlenecks between various items or components.

4.6 Data exporting

In many firms and industries, exporting data to a PDF file containing tables is a regular necessity. Tables are a common approach to organize and show data in a logical and intelligible fashion, whereas Portable Document Format (PDF) is a file format that is frequently used for documents that need to be easily shared or printed. Businesses can present data in a consistent, professional manner by exporting it to a PDF file with tables, which also makes it simple for users to access and share data in a standardized format.

Depending on the program or application being used, there are various ways to export data to a PDF file containing tables. Utilizing a PDF creation tool, which is frequently included into the application or software, is one conventional approach. Users of the program can choose the data to export and alter the table's design and formatting. The program creates the PDF file, complete with the table and any other pertinent information, after the settings have been selected [15].

Using a third-party library or API is another way to export data to a PDF file containing tables. To create PDF files, including tables, several programming languages and platforms have libraries or APIs available. Although they can require more technical knowledge to utilize, these libraries or APIs frequently offer more sophisticated capabilities and customization choices than the built-in PDF creation tools.

It's crucial to take the table's formatting and style into account when exporting data to a PDF file with tables. Tables should be logically arranged and simple to comprehend, with distinct headers and labels for each column and row. The table should be formatted correctly to fit the page or screen, with consistent font size and style throughout.

Furthermore, when exporting to a PDF file with tables, the data itself must be taken into account. To make sure the data is accurate and full before exporting, it should be cleansed and sorted. Double-checking any calculations or formulas will help to assure their accuracy. Any outliers or anomalies should be looked into and, if necessary, corrected. Businesses may make sure that the information displayed

in the PDF file is correct, trustworthy, and usable by following these procedures.

It's crucial to take the table's design into account when exporting data to a PDF file with tables. With distinct and unambiguous titles, rows, and columns, the table should be aesthetically pleasing and simple to read. Key data points or trends can be highlighted with color, but it's crucial to make sure the color scheme is consistent and doesn't interfere with the table's reading.

Additionally, it is crucial to make sure that the exported information is useable and accessible to all users, including those with impairments, when exporting data to a PDF file with tables. Following accessibility rules, such as those offered by the Web Content Accessibility rules (WCAG), will help you achieve this. The standards include suggestions for improving content's accessibility for people with disabilities, including utilizing high contrast colors, making sure tables are correctly constructed, and offering alternative language for images.

Additionally, several libraries and tools for creating PDFs allow for the insertion of interactive features in the PDF file that is exported. This can contain clickable links, bookmarks, and search capabilities, all of which can improve the PDF file's usability and accessibility. For instance, users can utilize bookmarks to easily move to particular table sections or the search feature to look for particular data points.

The security ramifications of exporting data to a PDF file containing tables should also be taken into account. It could be essential to incorporate additional security measures, including password protection or encryption, depending on how sensitive the data being exported is. These precautions can aid in preventing data from being intercepted or altered while in transit and ensuring that it is only accessible to authorized users.

In conclusion, many firms and industries have a common and crucial necessity for data exporting to a PDF file with tables. Table-based PDF files can be created using a variety of approaches, including built-in PDF production tools and third-party libraries or APIs. It is crucial to take into account both the correctness and completeness of the data being presented as well as the formatting and style of the table when exporting data to a PDF file with tables. Businesses may make sure that their PDF files with tables are polished, accurate, and simple to use by adhering to these best practices.

Many institutions use inventory management systems because it's crucial for efficient operation to keep track of inventory items including furniture, equipment, and supplies. These systems often offer a standardized and organized method for tracking and recording inventory goods, which helps to streamline the inventory management process. The ability to export data to a PDF file with tables is one of these systems' key features; it can be especially helpful for creating reports, summaries, and inventory lists.

Inventory managers may be able to provide expert, educational, and simple-to-read reports by using PDF files with tables. These reports can contain a variety of inventory data, including item description, location, number, and status. This data can be sorted and put into tables for simpler presentation and analysis. In addition, PDF files are simple to share with other employees or departments, making them the perfect format for team-based inventory management.

Additionally, using PDF files for inventory management can improve the system's precision and effectiveness. Inventory managers may immediately check and confirm inventory data by exporting it to a PDF file, which lowers the possibility of mistakes and discrepancies. Additionally, the inclusion of tables in PDF files can make it simple to search, sort, and filter data, which makes it simpler to recognize and locate particular inventory items.

The usage of PDF files with tables can provide various additional advantages for inventory management in universities, in addition to the ones already described. The ease with which inventory data can be archived and retrieved is one such benefit. Because PDF files are often smaller than other file formats, they are simple to store and send. Since PDF files may be used to establish digital archives of inventory information that can be easily accessed and retrieved as needed, this can be very helpful for colleges that need to manage significant volumes of inventory data.

Additionally, tables in PDF files can improve the accessibility of inventory information. Since PDF files can be viewed on many different platforms, including as PCs, tablets, and smartphones, inventory managers can access and evaluate inventory data more easily from any location. Additionally, PDF files are simple to print, making it possible to give relevant staff members or departments hard copies of inventory reports.

It is also important to note that using PDF files with tables can improve the confidentiality of inventory data. PDF files can be password-protected or encrypted to make sure that only authorized people can access crucial inventory data. This can be especially crucial in academic settings where inventory data may contain private information like equipment serial numbers or financial data.

In conclusion, using PDF files with tables can have a variety of advantages for university inventory management. PDF files containing tables can be a useful tool for inventory managers in universities since they make the process of archiving, retrieving, and making inventory information accessible more accurate and efficient. Universities may make sure their inventory is properly handled and that they have access to the information they need to make decisions about their inventory needs and requirements by making use of this feature.

4.7 Interfaces

The system's user interface is made to be simple to use and intuitive. The user interface was created using ReactJS, a well-liked front-end toolkit for creating user interfaces, and is accessible through a web browser. The user interface enables people to communicate with the system by sending queries and getting immediate results.

The inventory management portion and the room management section are separated on the system's primary interface. While the room management area enables users to view and manage the rooms themselves, the inventory management section enables users to view and manage the inventory of each room.

A table listing every object currently in the room may be found in the inventory management area, along with tools for adding, removing, and changing items. The table will refresh in real-time to only display the matched items when users use the search bar to look for particular things.

A list of all rooms is provided in the room administration area, along with tools for adding, removing, and changing rooms. The list will refresh in real-time to only display the matching rooms if users use the search field to look for particular rooms.

Overall, the user interface is made to be straightforward and simple to use, with all relevant data and options displayed in a logical and clear manner. The system's interface is essential since it makes it simple and effective for users to engage with and manage the inventory and rooms [16].

Before accessing the inventory and room management areas, users can log in and authenticate themselves using the system's authentication interface. By ensuring that only authorized users may access the system, the authentication interface improves security and prevents unauthorized access to sensitive data.

The login page, choices for changing passwords, and options for creating new accounts are all included in the authentication interface. A user is led to the main interface, where they may access the inventory and room management areas, after logging in.

The authentication interface is a crucial part of the system since it guarantees that only authorized users can access and edit the inventory and room data, lowering the possibility of data breaches and other security issues.

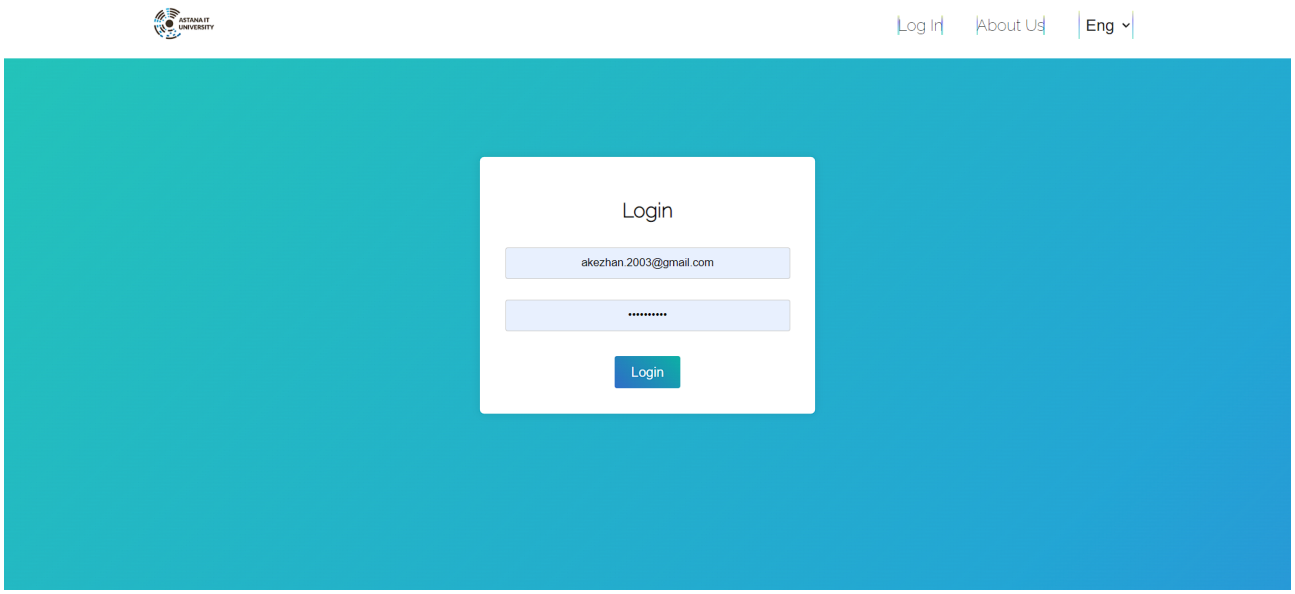

Figure 3.1 Login page

Figure 3.1 illustrates the login form, where user can fill the email and password to login.

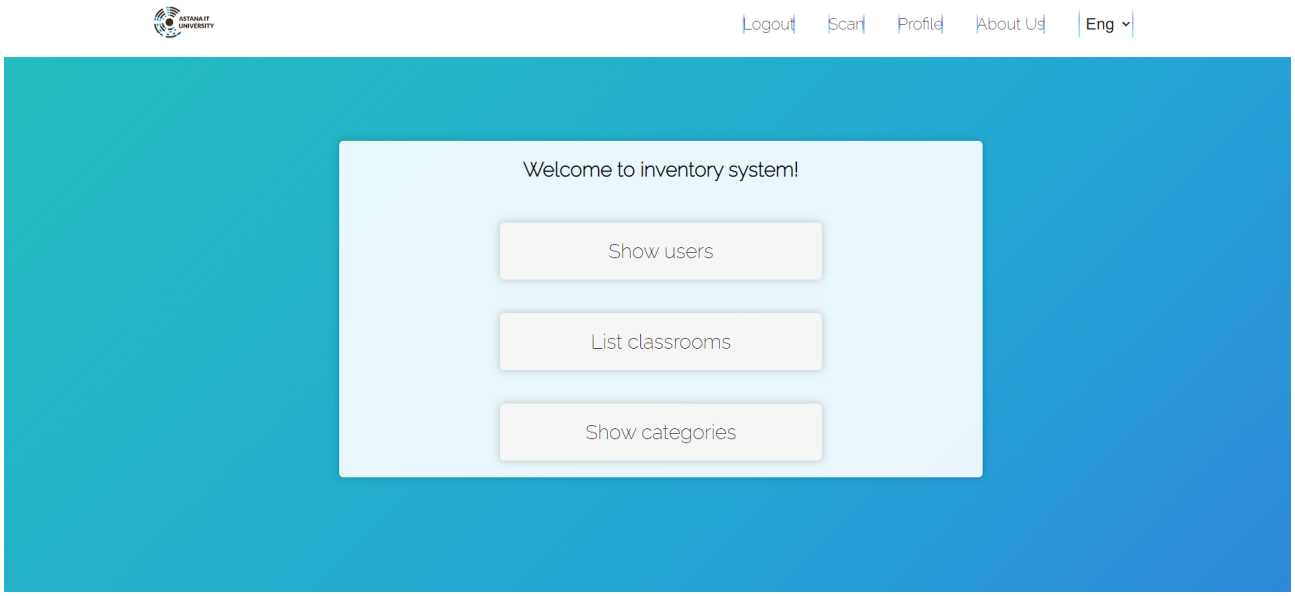

Figure 3.2 Admin panel page

Figure 3.2 shows the admin panel of the inventory management application, where user can see the three options, 'Show users', 'List classrooms' and 'Show categories'.

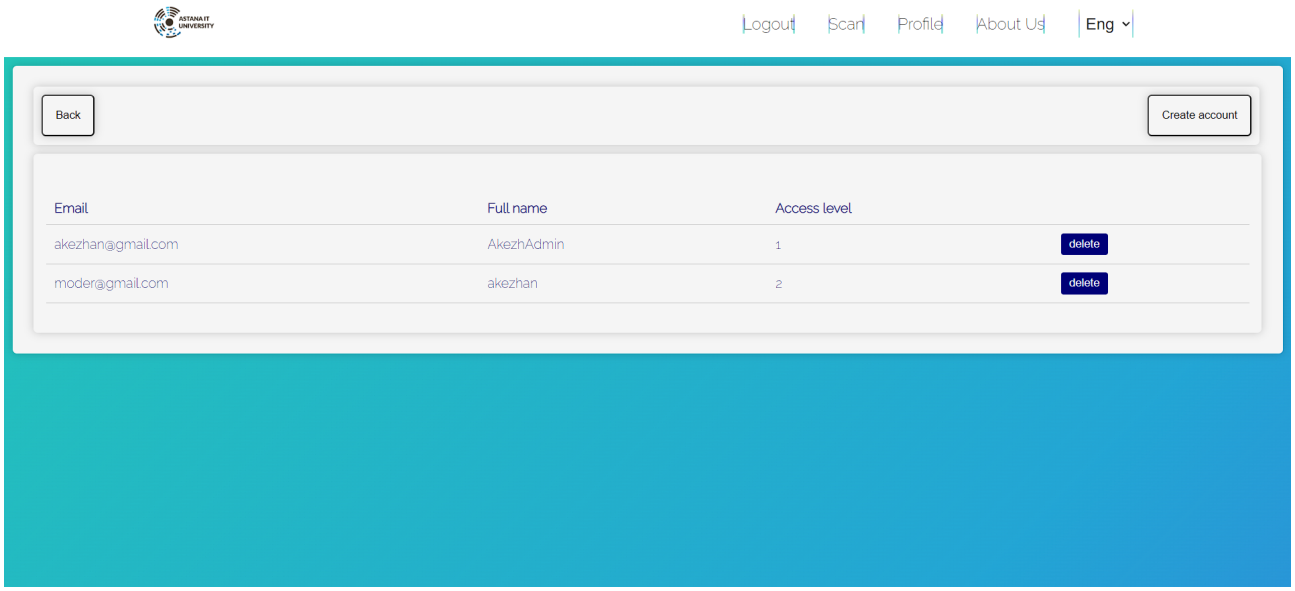

Figure 3.3 Users list page

Figure 3.3 demonstartes list of users, their email, full name and access level, and only superadmin can create or delete the user.

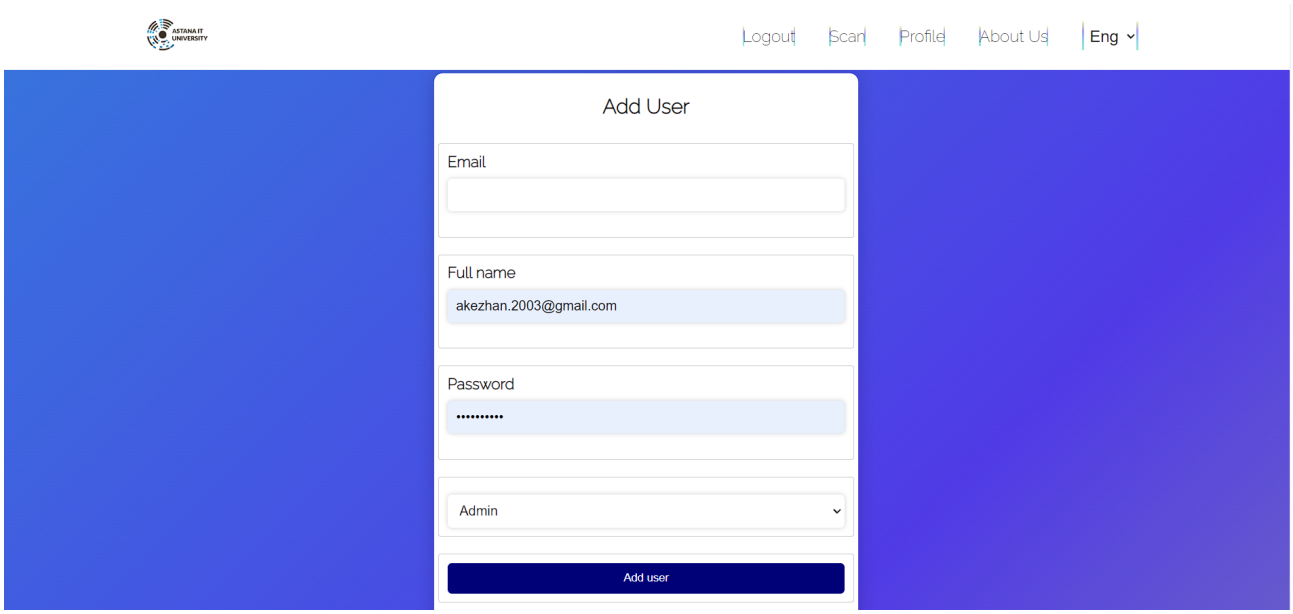

Figure 3.4 Create user page

Figure 3.4 displays the create user page, where superadmin can add user, by filling the email, full name, password and role fields.

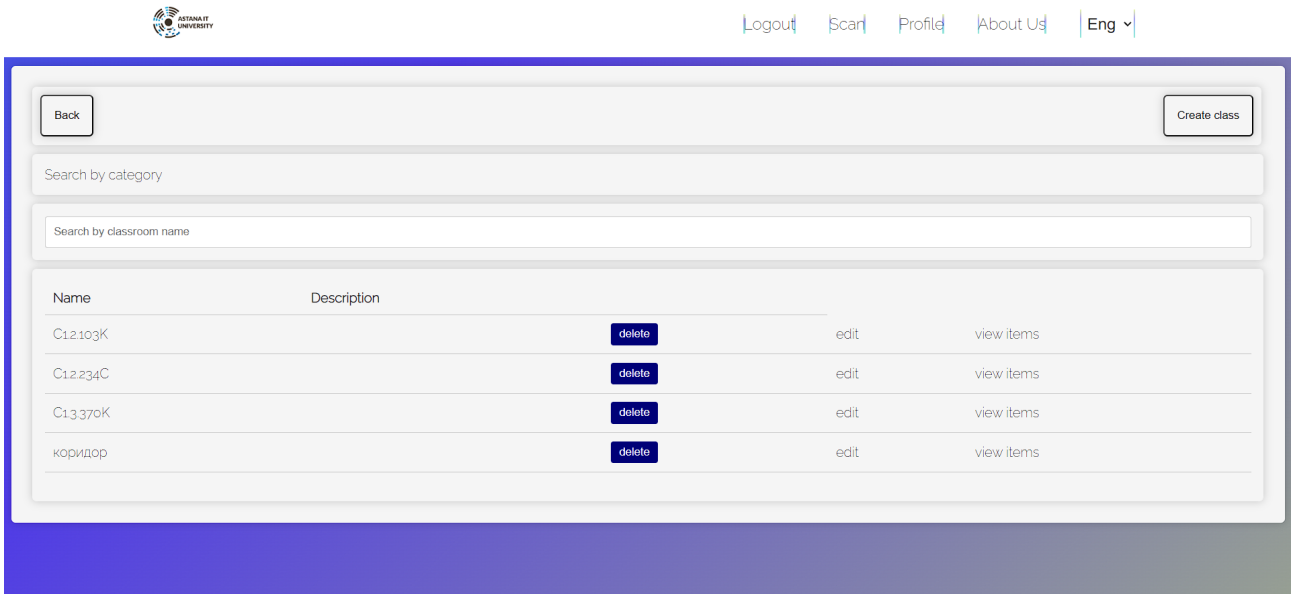

Figure 3.5 Class list page

Figure 3.5 represents list of the classes, admin and superadmin can see the name and description of the class, whilst having permission to edit the name and description of the class. By clicking view items button both can navigate to the list of items to corresponding class.

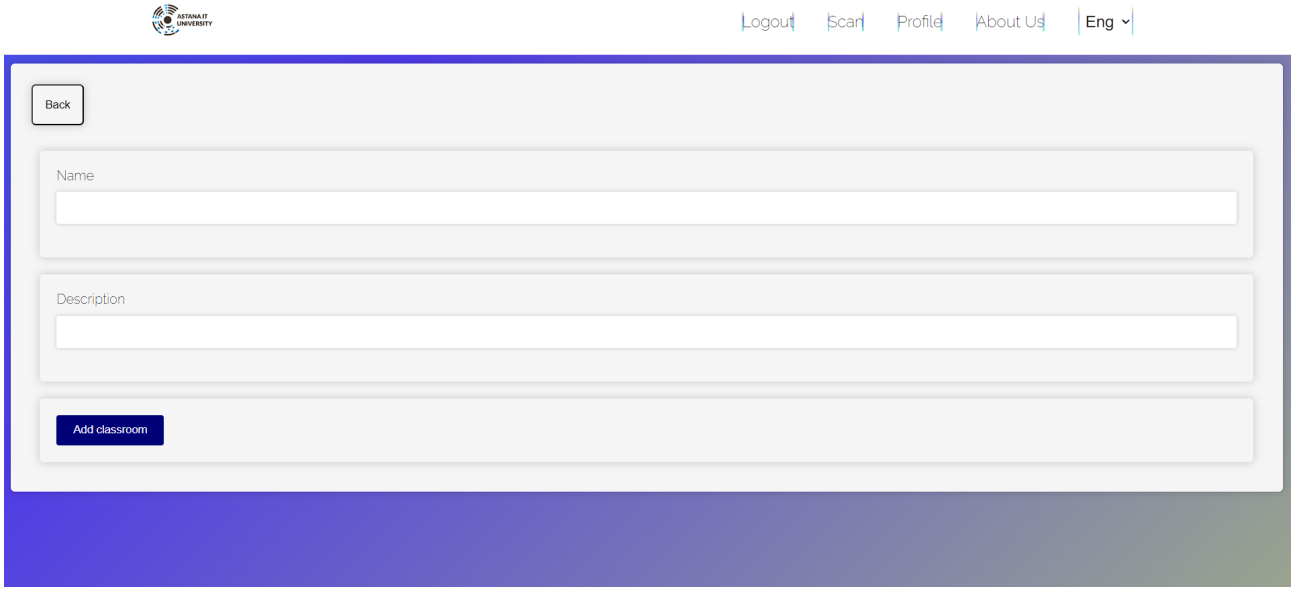

Figure 3.6 Create class page

Figure 3.6 illustrates create class page, where admin and superadmin can fill the name and description of the class.

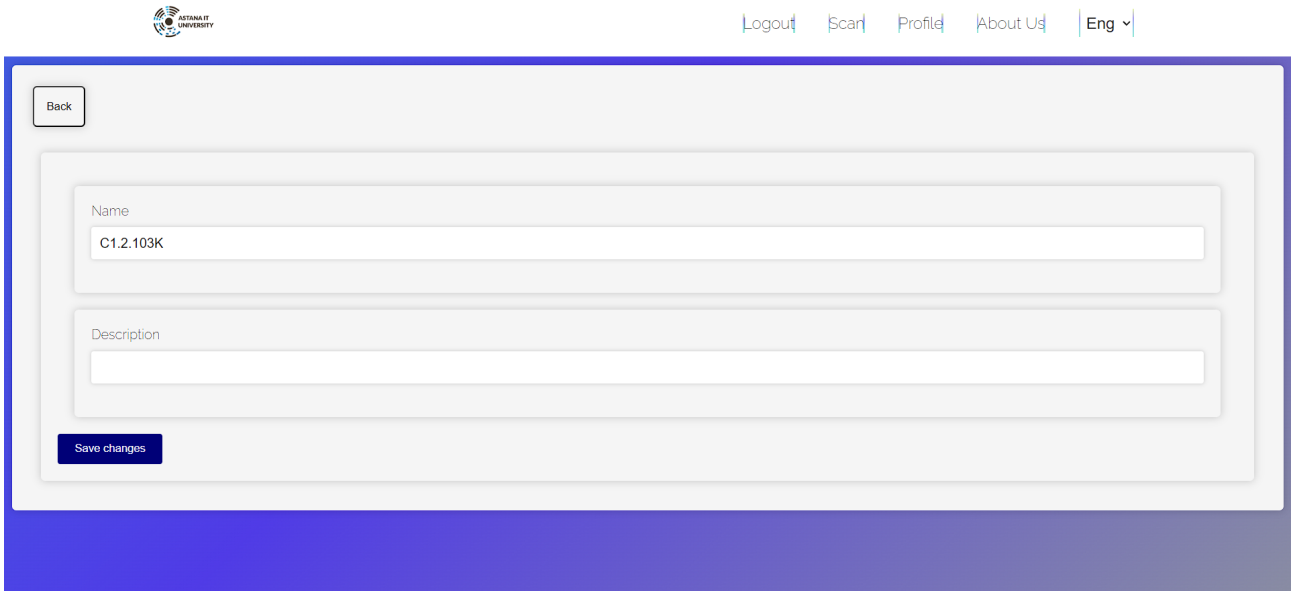

Figure 3.7 Edit class page

Figure 3.7 shows edit class, where admin and superadmin can edit the name and description of the class.

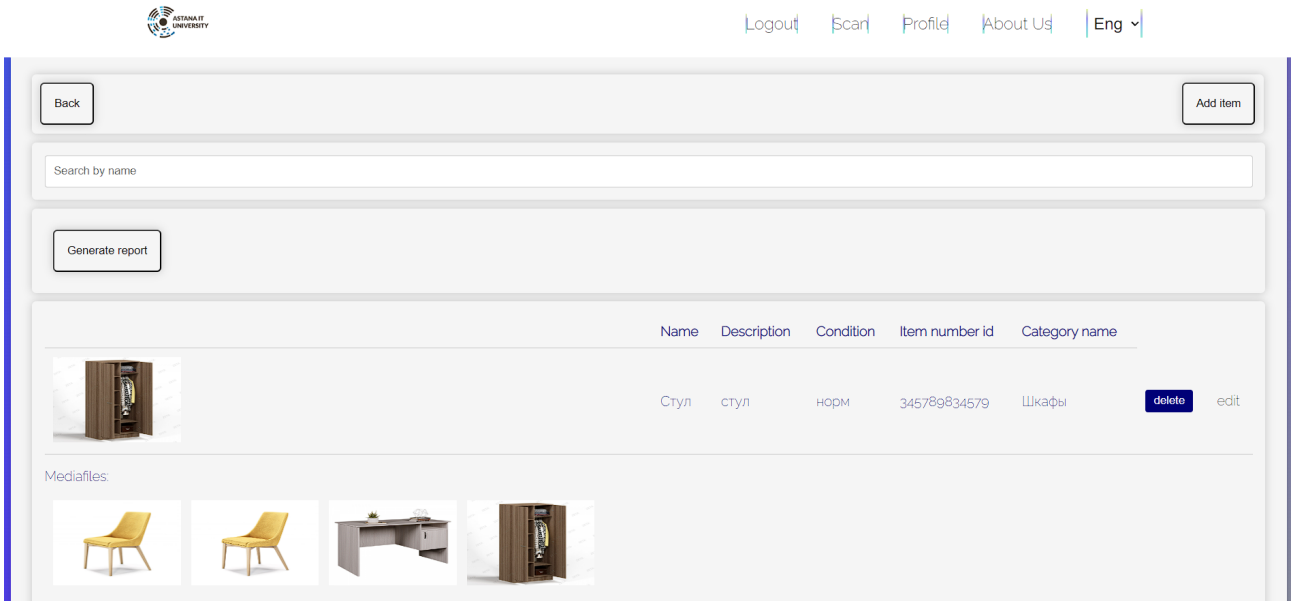

Figure 3.8 Items list page

Figure 3.8 displays items list page, in the items list page users are able to see the image, description, name and item number id of the item. Users can edit, delete and add the item.

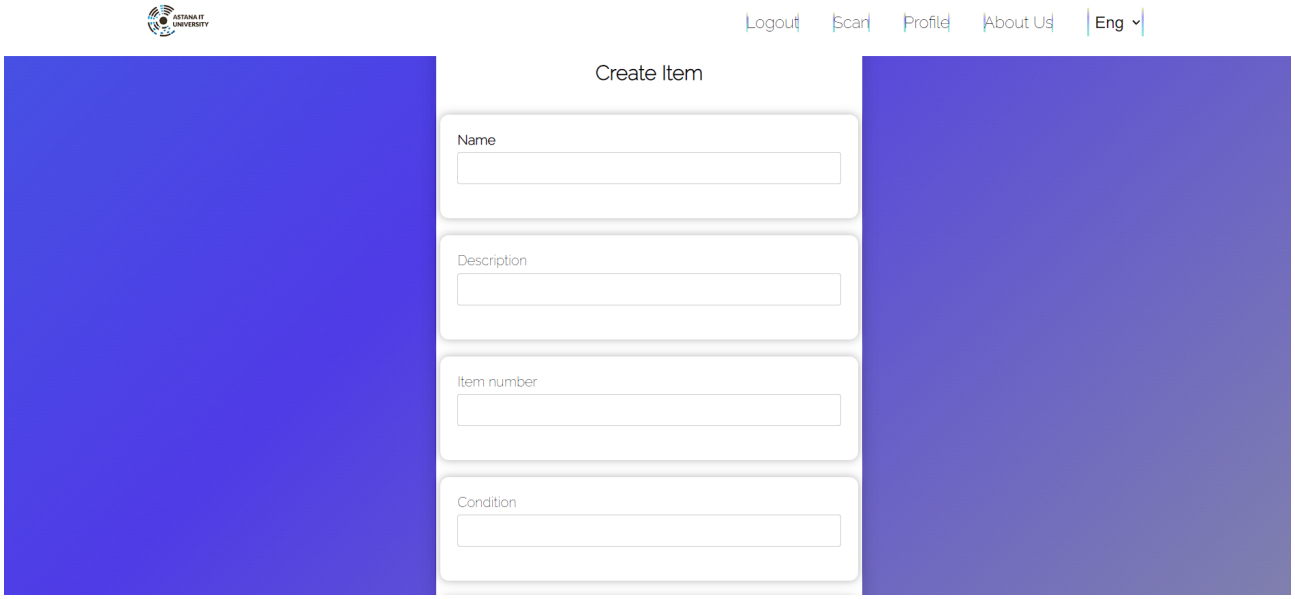

Figure 3.9 Create item page

Figure 3.9 illustartes create item page, where users can create item by filling the data of the item.

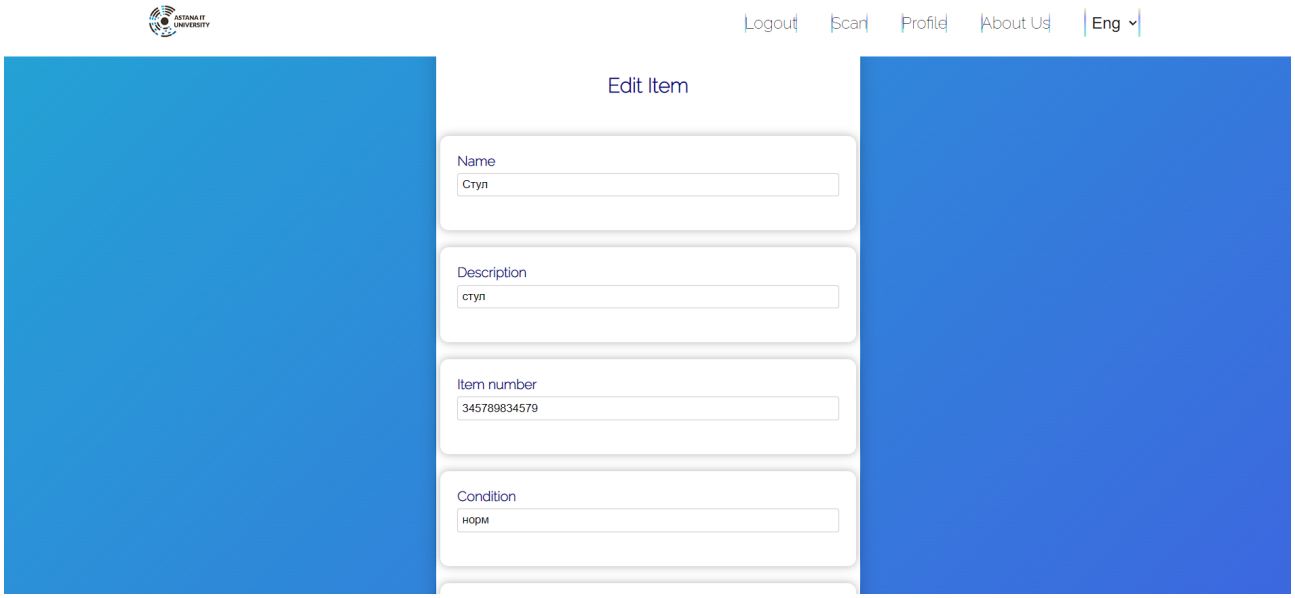

Figure 3.10 Edit item page

Figure 3.10 demonstartes the edit item page, where users can edit the data of the item.

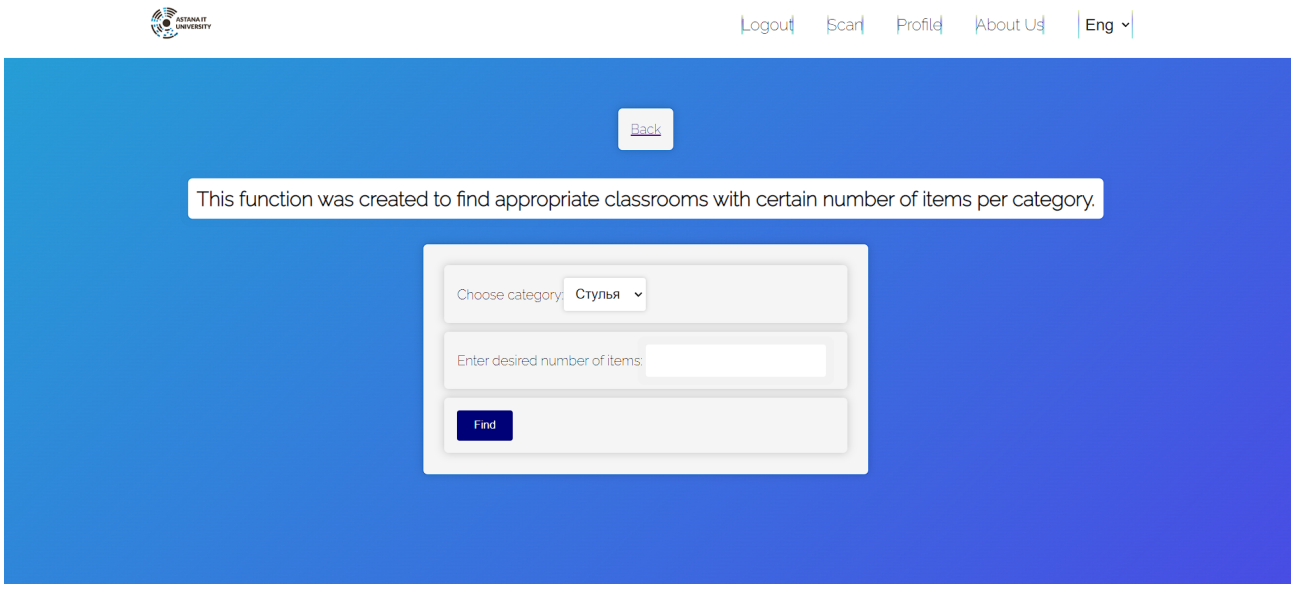

Figure 3.11 Search by category page

Figure 3.11 represents the search by category page, where users can search by choosing the category and the minimum amount of item for choosen category.

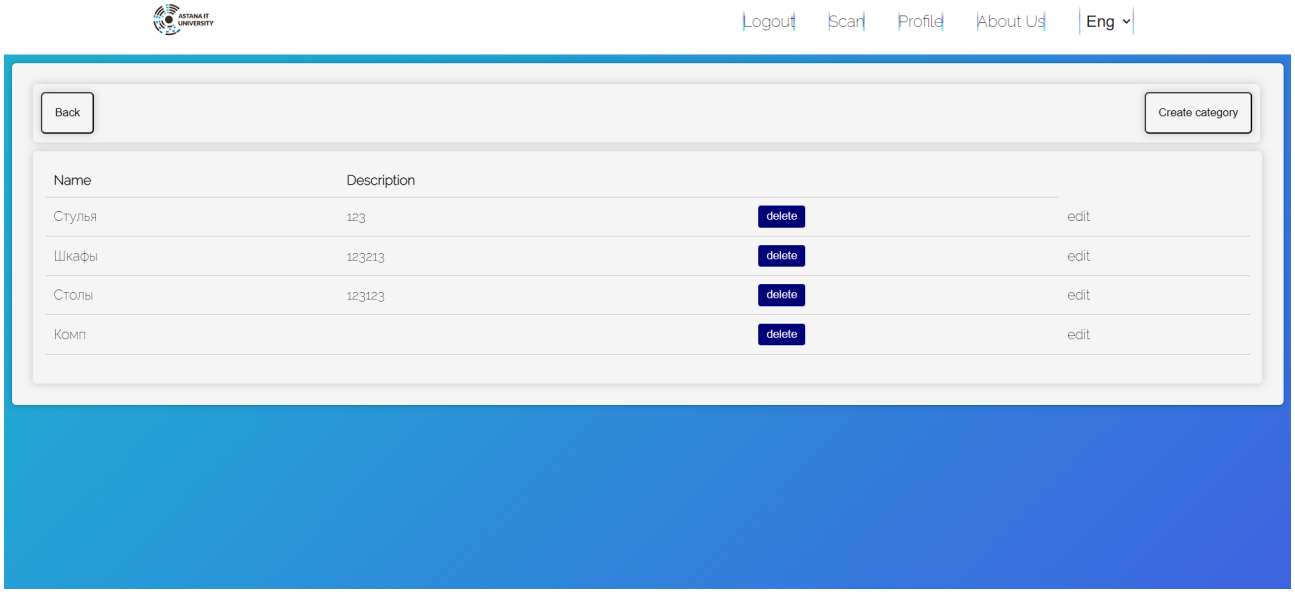

Figure 3.12 Category list page

Figure 3.12 shows the category list page, where users can see the name and description of the category, also can edit and delete the category.

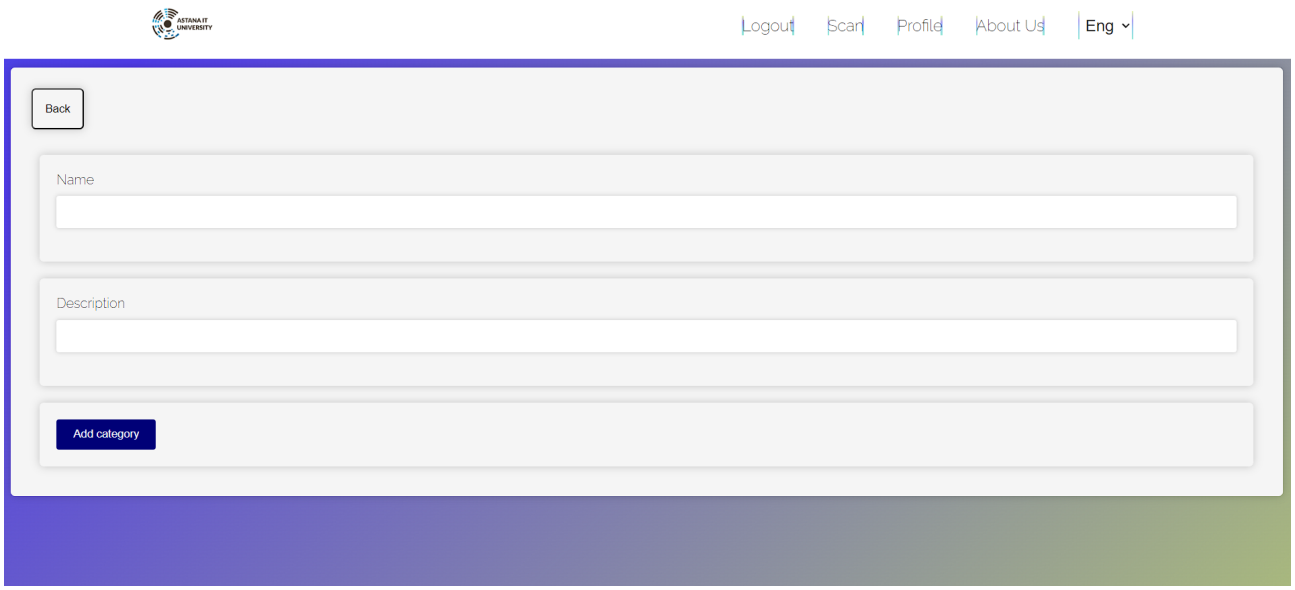

Figure 3.13 Create category page

Figure 3.13 represents create category page, where user can create a category by filling the name and description.

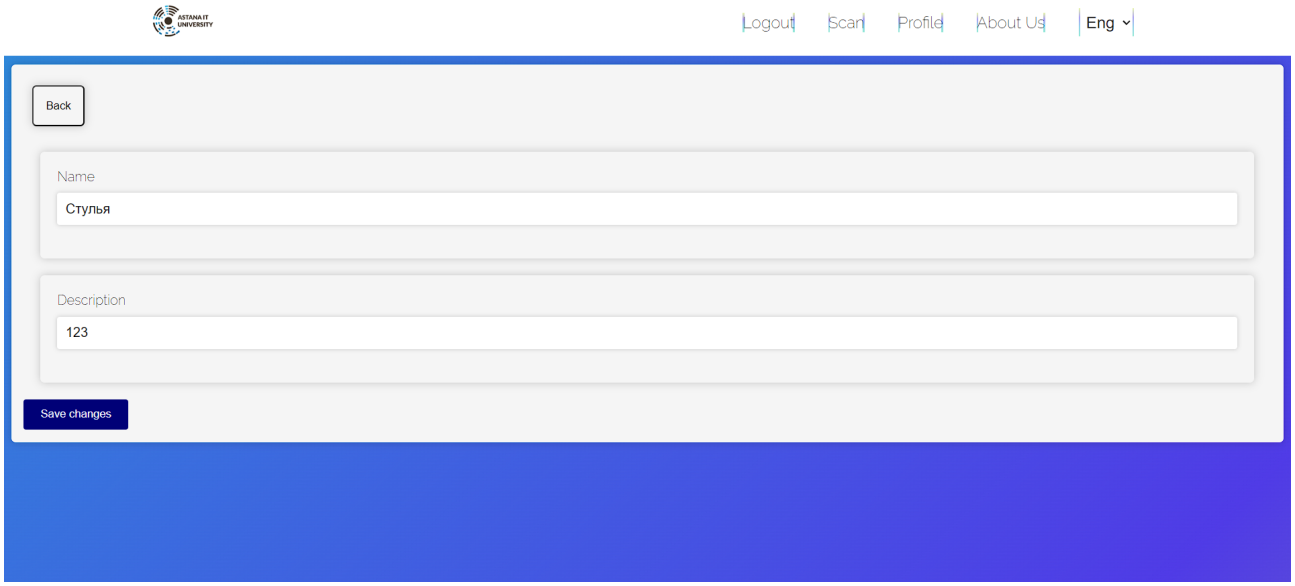

Figure 3.14 Edit category page

Figure 3.14 demonstrates edit category page, where users can edit the name and description of the category

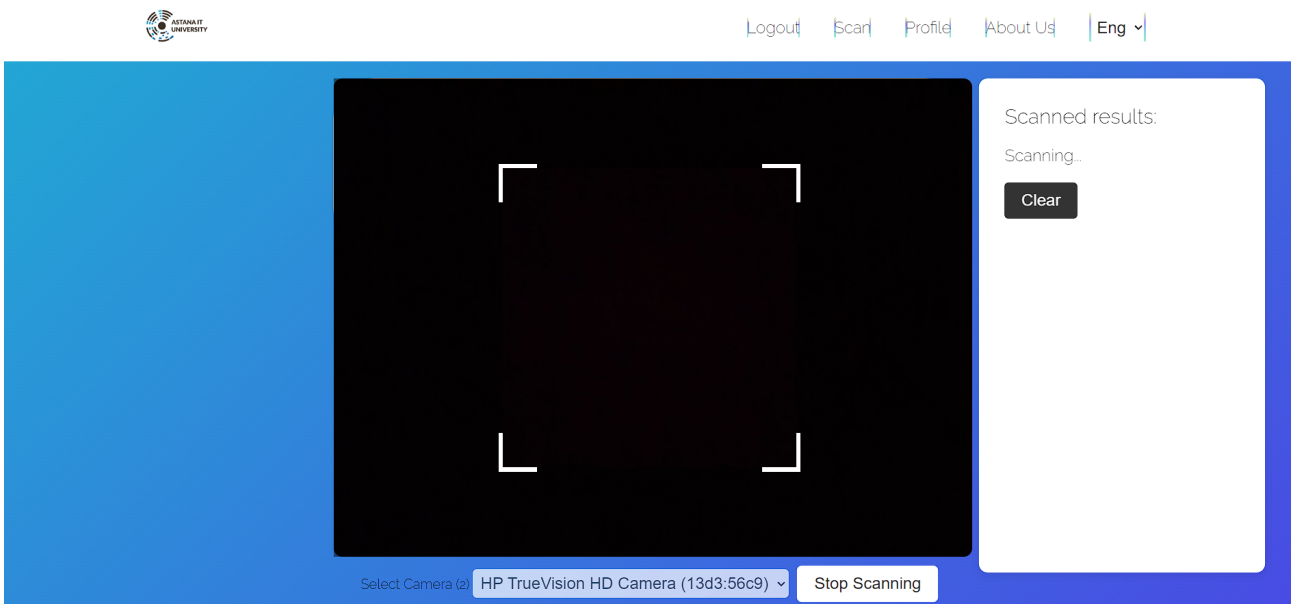

Figure 3.15 Scan page

Figure 3.15 displays scan page, where user can scan the barcode of the item, after which he/she can see the information of the item if the item is present, if it is not users can add the item.

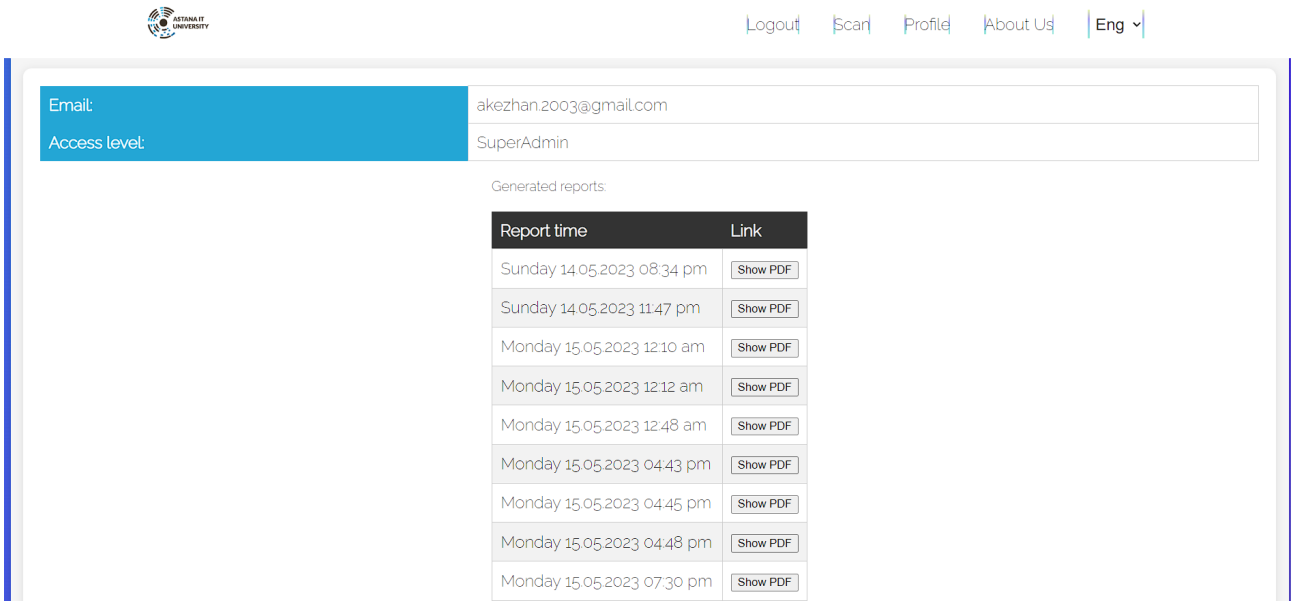

Figure 3.16 Profile page

Figure 3.16 shows profile page, where users can see the information of the user and generated report with the information at what time the report was generated, also users can see and download the report.

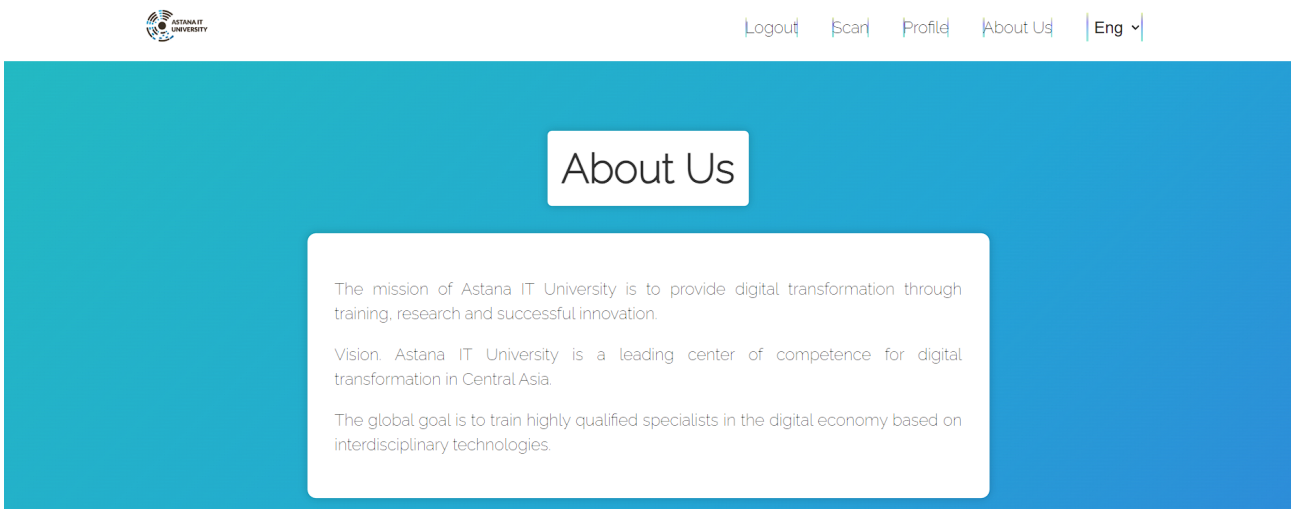

Figure 3.17 About us page

Figure 3.17 illustrates about us page, where users can see the information about the Astana IT University.

4.8 Integration

The system's many parts are integrated so they can operate as a cohesive whole to accomplish the system's objectives. The user interface can interact with the back-end server and database to retrieve and store data because the front-end and back-end components communicate via HTTP calls. The front-end server receives requests from the back-end server, processes them, and provides the results back to the front-end server for display.

The user may simply interact with the system thanks to the user-friendly interface provided by the ReactJS-based front-end component. It retrieves information from the PostgreSQL database from the back-end server and then shows the user the outcomes. User input is also handled by the front-end component before being forwarded to the back-end for processing.

A set of APIs are provided by the ASP.NET-based back-end component to handle requests from the front-end component. These APIs are in charge of communicating with the database, getting data, processing it, and sending the outcomes back to the front-end. A barcode scanning module is also part of the back-end system; it can be used to scan inventory items and update the database as a result.

The system's various parts all function flawlessly together to offer the university a productive and successful inventory management solution. The system's integration also makes room for future growth and development. For instance, it might be possible to expand the back-end component's features or modules without substantially altering the front-end interface. Similar to how changes to the back-end functionality could be performed without changing the front-end experience.

Scalability is another benefit of the system's integration because more servers or database instances may be added to meet growing demand without interfering with the system's operation. The system is future-proof and is simple to maintain and update as needed thanks to the usage of contemporary web technologies like ReactJS and ASP.NET.

In conclusion, the overall functionality and usefulness of the system are significantly influenced by the integration of the various system components. The effective data processing and management made possible by the front-end, back-end, and database components also makes it possible for the system to be expanded and improved in the future.

The system's main attributes and elements consist of:

The system is designed using a client-server architecture, in which the server handles data storage and processing while the client gives users a simple way to interact with the system. To do this, the system's back-end is built using the C#-based ASP.NET framework, which offers a stable environment for web applications. Data consistency and integrity are guaranteed by the use of a PostgreSQL database to store the data. ReactJS is used in the system's front-end development, which gives users a dynamic and responsive user interface. As a result, the system offers options to add, remove, and change classrooms and goods, aiding in the organization of the inventory in the university housing. Additionally, it avoids the issue of assigning academic groups to classrooms with subpar physical and technical amenities.

The system's main functions and elements collectively offer a complete solution for the management of inventories in university housing. To guarantee data consistency and integrity, it offers a user-friendly interface, a strong back-end, and a trustworthy database.

5 Data collection

To answer research questions, inform decisions, or acquire insights into a certain topic or event, data collecting is the process of obtaining information or data from numerous sources. The methods used to collect the data, which might be qualitative or quantitative, include surveys, interviews, observation, and experimentation.

The quality and relevance of the data collected impact the validity and trustworthiness of the research findings, making effective data gathering a critical component of research. In order to acquire accurate and relevant data, it is crucial to ensure that the data collecting process is well-planned and carefully carried out.

Any research project, including diploma works and theses, must include data collection. It entails compiling information or data from diverse sources in order to support or respond to research hypotheses. There are many different ways to collect data, including surveys, interviews, observation, and experimentation. The data might be qualitative or quantitative.

Successful data collecting is crucial for the completion of thesis and diploma projects. The validity and reliability of the research findings are based on the standard and applicability of the data that was gathered. In order to acquire accurate and relevant data, it is essential to make sure that the data collection procedure is well-planned and carefully carried out.

Supporting the study issue or hypothesis being studied is one of the main reasons why data collecting is important for diploma work. Data offer proof that can be used to support hypotheses and provide answers to research inquiries. It would be challenging to reach meaningful conclusions or make important discoveries without data.

Additionally, gathering and evaluating data offers proof that enhances the work's legitimacy and worth. The reliability and worth of the diploma work are increased by providing proof for the arguments and assertions made in it. This increases the research's significance and increases its value to graduate programs and future employers.

Additionally, gathering and interpreting data is an essential component of research and shows that the student has the required research abilities. Employers and graduate schools highly value the ability to plan and execute research studies, gather pertinent data, and conduct data analysis. It demonstrates that the student is able to do research and generate high-caliber work.

Last but not least, the information gathered for a diploma project might add to the body of knowledge already present in the student's field of study. This contribution could be substantial and could influence future studies. The success of diploma work and thesis research therefore depends on the efficient and ethical data collection, and it is crucial to take into account the ethical considerations

surrounding data collection, including obtaining participants' informed consent, maintaining confidentiality, and ensuring that the data is used only for its intended purpose.

A discussion with the university's IT department was conducted to collect data for the diploma project. The investigation's goals were to learn more about the university's current inventory management system, gain new perspectives on it, and decide whether a new system was necessary. The department at the university verified that it has a system in place for keeping track of inventories, but it is just for computers and a few other pieces of specialised equipment. Consequently, a more complete inventory management system that can handle all kinds of goods is required.

The IT department was also open to using any technology, whether a website or a desktop or mobile application, to create the new inventory management system. However, as barcode technology is compatible with their current scanning equipment, they advised using it to scan the objects directly.

The department did not have any particular preferences for the website's design, but they underlined the value of having a user-friendly, straightforward interface. Finally, they verified that the university has a server on which to host a website. Overall, the discussion with the IT team yielded insightful information that will guide the creation of the university's new inventorization system.

In conclusion, gathering data is a crucial component of thesis and diploma work. It offers proof, backs up the research topic or hypothesis, exemplifies research abilities, and adds to the body of knowledge in the student's field of study. Students should take the time to organize and carry out their data gathering procedure effectively to obtain accurate and valuable information. Effective and ethical data collection is vital for successful diploma work and thesis research.

6 Technology comparison

Any company that wants to properly manage its inventory must invest in inventory management solutions. By using these technologies, businesses can monitor their inventory levels in real-time and make sure they always have enough product to satisfy client demand. Businesses have a variety of technological solutions to select from when putting in place an inventory management system, each with certain benefits and drawbacks. For instance, barcode scanning technology is a well-liked choice that enables companies to properly and instantly track inventory levels, ensuring that they always have the right amount of merchandise on hand. RFID technology, on the other hand, provides even better accuracy and may be used to monitor inventory levels across numerous sites. In the end, the technology selected will depend on the particular requirements of the company and the kind of inventory they are handling. However, it is obvious that a strong inventory management system is an essential resource for any company hoping to thrive in the cutthroat business environment of today.

6.1 Barcode scanning technology

The way that firms handle their inventory has been changed by barcode scanning technology. Due to the technology's capacity to scan barcodes and update inventory levels in real-time, which has simplified the inventory management process, it has grown in popularity. There are two varieties of barcode scanners: portable and stand-mounted. They allow for simple data transfer and can be connected to a computer or mobile device. The primary benefits of barcode scanning technology are its speed and high accuracy, which assist to cut down on errors and save time.

The simplicity and usability of barcode scanning technology are two of its main advantages. Barcode scanners are a practical solution for organizations of all sizes because they are generally accessible and reasonably priced. Additionally, barcode scanning technology enables companies to precisely and quickly update inventory levels, lowering the chance of stockouts and overstocking. This aids companies in avoiding losses brought on by overstocking, which can tie up funds and increase storage costs. Stockouts, on the other hand, can result in missed sales and a drop in customer satisfaction, both of which can harm a company's reputation.

Using barcode scanning technology has additional benefits in addition to accuracy and speed. For instance, firms can watch inventory patterns and make wise decisions thanks to barcode scanning technology. Businesses can determine which products are selling well and which products may need to be restocked or withdrawn from inventory by evaluating inventory data. Businesses can boost their profitability by making informed judgments regarding their inventory levels with the aid of this information. The use of barcode scanning technology to lessen

employee workload is another benefit. Employees can focus more on other duties by automating the inventory management process instead of manually checking inventory levels. This can boost output and cut labor expenses, which is especially advantageous for small enterprises.

Last but not least, barcode scanning technology can raise client satisfaction. Businesses may ensure that products are constantly in stock and lower the chance that customers will be dissatisfied by out-of-stock items by carefully tracking inventory levels. Barcode scanning technology can also speed up checkout times and decrease line length by cutting down on the time it takes to scan and process merchandise. Better client experiences and greater customer loyalty may result from this.

In conclusion, barcode scanning technology offers various benefits to organizations, such as accuracy, simplicity, inventory tracking, a reduction in employee workload, and increased client pleasure. Businesses can streamline their inventory management procedures, lower expenses, and boost profitability by utilizing barcode scanning technology.

6.2 RFID technology

The management of inventory systems by enterprises has been completely transformed by RFID technology. RFID technology, which uses radio waves to identify and monitor inventory goods, is now the technology of choice for sectors like healthcare, manufacturing, and logistics. In order to update inventory levels in real-time, products are given RFID tags, which are subsequently scanned by RFID readers. Since the technology offers so many advantages, supply chain management cannot function without it.

The ability to scan several items simultaneously is one of the most important benefits of RFID technology because it significantly lowers labor costs and boosts productivity. Because of the time and money savings resulting from this, businesses can operate more profitably. Businesses may track inventory items without making direct touch with them because to the ability of RFID tags to be read from a distance. When handling a large inventory, this tool adds another level of convenience.

The real-time information that RFID technology delivers on inventory levels is another advantage. Businesses can respond promptly to changes in demand thanks to this data, which improves the effectiveness of their supply chain. Businesses may make better decisions regarding their inventory, such as whether to refill or when to lower inventory levels, by having access to this data. Businesses may improve their supply chains with the use of this information, which will ultimately result in higher performance.

Additionally, inventory management security has been enhanced using RFID technology. Businesses can easily monitor their inventory and lower the risk of

theft and loss by attaching RFID tags to products. As organizations struggle to preserve the security of their inventory during transportation and storage, this technology has become more and more important.

Finally, RFID technology has radically changed how companies operate their inventory management systems. RFID technology has turned into a crucial tool for companies to better their supply chain management since it offers real-time data, increased effectiveness, and greater security. RFID technology will surely be essential to organizations' success as they continue to struggle with inventory management.

6.3 IoT technology

With real-time data on inventory levels and a host of advantages for organizations in a variety of industries, the Internet of Things (IoT) is revolutionizing the way businesses manage their inventory management systems. In industries like agricultural and food processing, where it's critical to keep an eye on the temperature and moisture levels of inventory goods, IoT devices, such as sensors and cameras, can be used to watch inventory levels and offer real-time data on stock levels.

The capacity of IoT technology to gather and analyze vast amounts of data is one of its key benefits. By detecting important trends and patterns, this data can then be used to optimize the inventory management procedures. Businesses may get a complete picture of their inventory levels and improve the efficiency of their supply chain operations by deploying IoT devices to monitor inventory levels in real-time. Additionally, IoT technology enables companies to follow inventory goods along the supply chain, offering insightful data into the effectiveness of their operations and spotting future development opportunities.

IoT technology improves inventory management accuracy, decreasing the possibility of stockouts and overstocking. Businesses can swiftly adapt to changes in demand and prevent expensive operations disruptions by giving real-time data on inventory levels. IoT devices may also keep an eye on inventory conditions like temperature and moisture, ensuring that goods are stored properly and lowering the chance of deterioration or damage. The use of IoT technology to provide supply chain transparency is another advantage. Businesses can increase insight into their operations and find possible bottlenecks or inefficiencies by tracking inventory items across the supply chain. Then, they can use this knowledge to streamline their processes and boost overall performance.

In conclusion, organizations aiming to enhance their inventory management procedures can gain a variety of advantages from the Internet of Things. It is the perfect choice for sectors where inventory tracking is crucial because of its capacity to gather and analyze real-time data on inventory levels and conditions. Businesses can utilize IoT technology to promote accuracy, transparency, and

operational efficiency, which boosts productivity and profitability.

The business goals and requirements will determine the best technology to use for your real-time inventory system. While RFID technology is great for organizations that need to manage inventory products at different stages of the supply chain, barcode scanning technology is ideal for those who need to swiftly and accurately update inventory levels. IoT technology is an excellent choice for companies who need to gather a lot of data for analysis and monitor inventory items in real-time.

When deciding whether to deploy an inventory management system, businesses can make an informed choice by being aware of the benefits and drawbacks of each technology solution. When selecting a technology solution, it is crucial to take into account elements like cost, usability, and the unique needs of your company. Businesses can make sure they have enough inventory to fulfill client demand and streamline their inventory management procedures by putting the correct technologies in place.

7 Conclusion

The creation of a client-server application to handle inventory in university housing was a key component of the inventorization project's success. As a result of the application's sophisticated resource management system, academic groups are assigned to classrooms with enough technical and physical amenities.

The project used cutting-edge tools like ReactJS, ASP.NET, PostgreSQL, and AWS to create a scalable and reliable solution. The ReactJS-built front-end component has an aesthetically pleasing and user-friendly interface that enables users to add, update, and remove classrooms and items. The business logic is handled by the back-end component, which is built using ASP.NET, and it interacts with the PostgreSQL database to ensure quick data retrieval and storage.

Through HTTP calls, the front-end and back-end components are integrated, allowing for real-time modifications and a fluid user experience. The user input on the front-end interface initiates the system's data flow, which proceeds along a clearly defined path through the back end to the PostgreSQL database. By doing this, the system's data consistency and dependability are guaranteed.

In order to guarantee the system's uptime and security, the hosting component is essential. The deployment procedure is automated using AWS services, lowering the possibility of mistakes and inconsistencies. The servers are managed to maintain availability and performance while fulfilling the needs of the system. As a result, users can use the system from any location at any time while feeling secure in its dependability. The inventory project has addressed the difficulties the institution had in controlling inventory in its lodging. It offers a single platform for inventory management, avoiding situations where facilities and equipment in designated classrooms are inadequate. The university may effectively manage its resources by using the application, which improves efficiency and maximizes the use of facilities that are already available.

The inventorization project has also shown the advantages of adopting contemporary frameworks and technology for creating a sophisticated inventory management system. A visually appealing and responsive user interface was produced as a result of using ReactJS for the front-end component. This improves the user experience overall and makes sure that users can engage with the system and explore it with ease.

The ASP.NET framework-based back-end component has given the system's business logic a strong foundation to be implemented on. The usage of C# has made it possible to process and handle data in an expedient manner, guaranteeing that tasks like adding, removing, and updating classrooms and items are completed correctly and on schedule.

The front-end, back-end, and database have all been integrated into the system

in a seamless and effective manner. The system's internal data flow has been carefully thought out, making sure that user input is accurately processed, saved, and retrieved before being accurately shown to the user. This seamless connection has improved the system's overall performance and dependability.

The institution has been able to overcome the difficulties associated with inventory management in its accommodations thanks to the inventorization effort. This client-server application has given the university more oversight and control over its resources. The system's capacity to stop instances of inadequate facilities and equipment in designated classes has greatly enhanced the university's overall operation and organization.

Additionally, the project has offered insightful knowledge and lessons for upcoming improvements and expansions. In order to further enhance the inventory management system and create new applications for the university, the project team has accumulated knowledge and experience working with contemporary technologies and frameworks.

The inventory management project has, overall, been successful in reaching its goals of creating a client-server application to manage inventories in university housing. The system's effectiveness and usability have been boosted by its user-friendly interface, excellent data flow, smooth integration, and safe hosting. The institution has considerably improved its inventory management procedures by putting this system in place, which has improved resource allocation, increased efficiency, and improved the learning environment for its students.

BIBLIOGRAPHY

1 *Ahmed E. R., Alabdullah T. T. Y Shaharudin M. S.* Approaches to Control Mechanisms and Their Implications for Companies' Profitability: a Study in UAE / Alabdullah T. T. Y Shaharudin M. S. Ahmed, E. R. // *Journal of accounting Science.* — 2020. — Vol. 4, no. 2. — Pp. 11–20.

2 *Dechow P. M., Ge W. Larson C. R. Sloan R. G.* Predicting material accounting misstatements / Ge W. Larson C. R. Sloan R. G. Dechow, P. M. // *Contemporary accounting research.* — 2011. — Vol. 28, no. 1. — Pp. 17–82.

3 *Shin S., Ennis K.L. Spurlin W.P.* Effect of inventory management effificiency on profitability: current evidence from the US manufacturing industry / Ennis K.L. Spurlin W.P. Shin, S. // *Journal of Economics and Economic Education Research.* — 2015. — Vol. 16, no. 1. — Pp. 98–106.

4 *Wang Q., Wu J. Zhao N. Zhu Q.* Inventory control and supply chain management: A green growth perspective / Wu J. Zhao N. Zhu Q. Wang, Q. // *Resources, Conservation and Recycling.* — 2019. — Vol. 145. — Pp. 78–85.

5 *Carter, C.* Implementing an effective inventory management system / C. Carter // *Journal of Business Retail Management Research.* — 2020. — Vol. 15, no. 2. — Pp. 1–9.

6 *Kumar A., Singh R.* A review of inventory management systems / Singh R. Kumar, A. // *International Journal of Supply Chain Management.* — 2021. — Vol. 10, no. 4. — Pp. 327–339.

7 *Lee H., Whang S.* An analysis of inventory management practices in small and medium-sized enterprises / Whang S. Lee, H. // *International Journal of Production Economics.* — 2019. — Vol. 210. — Pp. 99–109.

8 *Manojlovich M., Loughridge M.* The impact of technology on inventory management systems / Loughridge M. Manojlovich, M. // *Journal of Applied Business Research.* — 2022. — Vol. 37, no. 3. — Pp. 327–341.

9 *Wong-On-Wing B., Tam C. M.* The role of inventory management in supply chain management / Tam C. M. Wong-On-Wing, B. // *International Journal of Logistics Management.* — 2020. — Vol. 31, no. 1. — Pp. 56–71.

10 *Ahmed E. R., Aiffin K. H. B. Alabdullah T. T. Y. Zuqebah A.* Zakat and Accounting Valuation Model / Aiffin K. H. B. Alabdullah T. T. Y. Zuqebah A. Ahmed, E. R. // *Journal of Reviews on Global Economicse.* — 2016. — Vol. 5. — Pp. 16–24.

11 *Bi G., Song L. Fei Y.* Dynamic mixed-item inventory control with limited capital and short-term financing. / Song L. Fei Y. Bi, G. — Annals of Operations Research, 2020. — Vol. 284. — Pp. 99–130.

12 *Gupta R., Singh V. P. Singh G. K.* Inventory control system using lean manufacturing principles: a review / Singh V. P. Singh G. K. Gupta, R. // *International Journal of Lean Six Sigma.* — 2017. — Vol. 8, no. 1. — Pp. 82–104.

13 *Yusyawiru N., Setyaningrum D.* Analysis of Internal Control System on Inventory at University of X. / Setyaningrum D. Yusyawiru, N. // Prosiding CELSciTech / CELSciTech. — Vol. 3. — 2018. — Pp. 6–13.

14 *Wang L., Li M. He Y.* Research on the design of an inventory control system based on big data / Li M. He Y. Wang, L. // *International Journal of Distributed Sensor Networks*. — 2019. — Vol. 15, no. 2. — Pp. 101–107.

15 *Aro-Gordon S., Gupte J.* Review of modern inventory management techniques / Gupte J. Aro-Gordon, S. // *Global Journal of Business Management*. — 2016. — Vol. 1, no. 2. — Pp. 1–22.

16 *Kim D. Y., Park S. J.* Development of an integrated inventory control system for a small business / Park S. J. Kim, D. Y. // *Journal of Industrial Engineering and Management*. — 2019. — Vol. 12, no. 2. — Pp. 262–282.

Appendix A Frontend Code

1.1 Multi Languages

```
# Shows how a React component's language change capability
operates. The language may be selected using a choose
dropdown element and the method handleLanguageChange.
When the value of the choose dropdown menu changes, the
handleLanguageChange method is called.
Using localStorage.getItem('lang'), the function verifies
the local storage's current language value.
When the current language is changed to English ('en'), it
changes to Russian ('ru'). It uses
localStorage.setItem('lang', 'ru') to change the language in
the localStorage.
The language in the i18next library is updated by using
i18n.changeLanguage('ru').
Using the setLanguage function (assuming setLanguage is a
state setter function), it changes the language state to
"ru".
The language state variable (if it stores the current
language) is the value set for the value attribute.
The handleLanguageChange method is assigned to the onChange
event, so when the select value changes, the language
changes.
The text inside the choice> elements is translated using the
t function given by i18next (t("Eng") and t("Rus")). The
option> elements indicate the language options ('en' and
'ru').

const handleLanguageChange = (e) => {
        if(localStorage.getItem('lang') === 'en') {
            localStorage.setItem('lang', 'ru');
            i18n.changeLanguage('ru', (err, t) => {
                if (err) return console.log(err);
            });
            setLanguage('ru');
        } else {
            localStorage.setItem('lang', 'en');
            i18n.changeLanguage('en', (err, t) => {
                if (err) return console.log(err);
```

```
            });
            setLanguage('en');
        }
    };

return (
        <div className="navbar">
            <div className="navbar-container">
                <div className={`navbar-links ${isOpen ?
                    "active" : ""}`}>
                    <ul>
                        {isAuth ? (
                        <li>
                            <select value={language} onChange=
                            {handleLanguageChange}>
                                    <option value="en">{t("Eng")}
                                    </option>
                                    <option value="ru">{t("Rus")}
                                    </option>
                            </select>
                        </li>
                    </ul>
                </div>
            </div>
        </div>
    );
};

# Russian language translation materials are contained in
the object resources. The translation key is where the
translations are kept. The keys'show users', 'list
classrooms', and 'Show categories' all have translations in
the resources object. Russian is used to specify these
translations.

const resourses = {
    ru: {
        translation: {
            'show users': 'Показать пользователей',
            'list classrooms': 'Список кабинетов',
```

```
            'Show categories': 'Показать категории',
      }
    }
}

# Russian language translation materials are contained in
the object resources. The translation key is where the
translations are kept. The keys'show users', 'list
classrooms', and 'Show categories' all have translations in
the resources object. Russian is used to specify these
translations.

i18next.
    use(initReactI18next).
    init({
    resources: resourses, lng: localStorage.getItem('lang')
    ,interpolation: {
        escapeValue: false
    }
})

# You can see the code for the admin panel page, for
instance, here.
import NavLink and any other required dependencies. Use the
"react-router-dom" to define your navigation links.
The "useTranslation" method from "react-i18next" is used to
access translation features.
The translation functions are accessed within the component
using the useTranslation hook, and the t function is
retrieved from them.
Each NavLink element is a link to a certain admin-related
page. Calling the t function with the appropriate
translation key as a parameter returns the text that appears
inside the NavLink> components.

import { NavLink, Navigate } from "react-router-dom"
import { useTranslation } from "react-i18next";

const AdminPage = () => {
    const {t} = useTranslation()
```

```
    return (
        <div className="admin-page">

        <div className="admin-link1">
            <NavLink to={'/admin/listusers'}>{t('show users')}
            </NavLink>
        </div>
        <div className="admin-link2">
            <NavLink to={'/admin/listclass'}>{t('list
            classrooms')}</NavLink>
        </div>
        <div className="admin-link3">
            <NavLink to={'/admin/listcategory'}>{t('Show
            categories')}</NavLink>
        </div>
        </div>
    )
}

export default AdminPage
```

1.2 Navigation bar

```
# the navigation bar's source code for a web application.
For menu management, navigation, language selection, and
authentication, it provides a range of logic and routines.
 Logic and each component's function are described one by
 one:
1) useContext(AuthContext): To access the AuthContext that
the AuthContext component is supplying, this line uses the
useContext hook. The context is used to get the isAuth
boolean value and the logout function.
2) useNavigate(): The useNavigate hook is used to access the
navigate method, which permits programmatic switching
between several routes.
3) useState(false): This line changes the state variable
isOpen's default value from true to false. It controls
whether the navigation menu is visible or hidden.
4) handleMenuClick(): This function is invoked when the menu
```

icon is clicked. SetIsOpen modifies the value of isOpen, which controls how the navigation menu appears.
5) handleLogoutClick(): This function is triggered when the "Logout" option is chosen. It calls the logout function from the AuthContext to fulfill the logout logic.
After logging out, the user is sent to the home page (navigate("/")).

A Link> component with the AITU logo on it takes users to the "admin/adminpanel" route.
The menu icon (either FaBars /> or FaTimes />) is displayed depending on the value of isOpen.
The navigation links are conditionally shown based on the isAuth value. If the user is approved, links to "Logout," "Scan," and "Profile" are displayed; otherwise, a link to "Log In" is displayed.
You may obtain the translated text for each link by using the t function and the relevant translation key.
A permanent "About Us" link may also be found in the navigation bar.
The navigate feature initiates navigation to the relevant routes when a link is clicked.
After logging out, the user is sent to the home page (navigate("/")).

```
import React, { useContext, useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { AuthContext } from "../../context/AuthContext";
import { FaBars, FaTimes } from "react-icons/fa";
import logo from "./aitu.png";

const NavigationBar = () => {
    const { isAuth, logout } = useContext(AuthContext);
    const navigate = useNavigate();
    const [isOpen, setIsOpen] = useState(false);
    const { t } = useTranslation();
    const [language, setLanguage] =
    useState(localStorage.getItem('lang'));

    const handleMenuClick = () => {
```

```
    setIsOpen(!isOpen);
};

const handleLogoutClick = () => {
    logout();
    navigate("/");
};
return (
    <>
        <div className="navbar">
            <div className="navbar-container">
                <div className="navbar-logo">
                    <Link to="admin/adminpanel">
                    <img src={logo} alt="AITU Logo"/>
                    </Link>
                </div>
                <div className="navbar-menu-icon" onClick=
                {handleMenuClick}>
                    {isOpen ? <FaTimes /> : <FaBars />}
                </div>
                <div className={`navbar-links ${isOpen ? "active"
                    <ul>
                        {isAuth ? (
                            <>
                                <li onClick=
                                {handleLogoutClick}>
                                    {t("Logout")}
                                </li>
                                <li onClick={() =>
                                navigate("/scan")}>
                                    {t("Scan")}
                                </li>
                                <li onClick={() =>
                                navigate('/userinfo')}>
                                    {t("Profile")}
                                </li>
                            </>
                        ) : (
                            <>
                                <li onClick={() =>
```

```
                                navigate("/")}>
                                    {t("Log In")}
                                </li>
                            </>
                        )}
                        <li onClick={() =>
                        navigate("/aboutus")}>
                            {t("About Us")}
                        </li>

                    </ul>
                </div>
            </div>
        </div>
    </>
  );
};

export default NavigationBar;
```

1.3 Scan

```
# Represents a website where QR codes may be scanned. The
operation of scanning is handled by the Html5QrcodePlugin, and
the results of the scan are shown by the ResultContainerPlugin.
1) useState([]): This line sets the decodedResults state
variable to an empty array. The results of the decoded QR code
scans will be stored.
2) onNewScanResult(decodedText, decodedResult): This function is
invoked each time the Html5QrcodePlugin returns a new scan
result.
It accepts as inputs the decodedText and decodedResult.
By employing the spread operator to append the decodedResult to
the previous array of results, the method changes the
decodedResults state.

The scanner plugin and the result container plugin are enclosed
by the scanner-page-container div.
When the decodedResults array has fewer than 10 items, the
Html5QrcodePlugin component is conditionally displayed inside
```

the container. The QR code scanning capability is configured, and the onNewScanResult callback is made available to handle fresh scan results.
A number of settings are available for the plugin, including fps (frames per second), qrbox (the size of the QR code scanning area), and disableFlip (whether to turn off rotating the camera preview).
Rendering always occurs for the ResultContainerPlugin component. The clearResult method is given to clear the results by setting decodedResults to an empty array. It then uses the results prop to show the scanned results.

Scanner page:

```
import { useEffect, useState } from "react"
import Html5QrcodePlugin from "../../../plugin/ScannerPlugin"
import ResultContainerPlugin from "../../../plugin/FilterPlugin";


export default function ScannerPage() {
    const [decodedResults, setDecodedResults] = useState([]);
    const onNewScanResult = (decodedText, decodedResult) => {
        setDecodedResults(prev => [...prev, decodedResult]);
    }


    return (
        <>
        <div className='scanner-page-container'>
            {decodedResults.length < 10 && <Html5QrcodePlugin
            fps={20}
            qrbox={350}
            disableFlip={false}
            qrCodeSuccessCallback={onNewScanResult}
            />}
            <ResultContainerPlugin clearResult={() =>
            setDecodedResults([])} results={decodedResults}/>
        </div>
        </>
    )
```

```
}
```

Scanner plugin:

```
import { Html5QrcodeScanner } from 'html5-qrcode';
import { useEffect } from 'react';

const qrcodeRegionId = "html5qr-code-full-region";

// Creates the configuration object for Html5QrcodeScanner.
const createConfig = (props) => {
    let config = {};
    if (props.fps) {
        config.fps = props.fps;
    }
    if (props.qrbox) {
        config.qrbox = props.qrbox;
    }
    if (props.aspectRatio) {
        config.aspectRatio = props.aspectRatio;
    }
    if (props.disableFlip !== undefined) {
        config.disableFlip = props.disableFlip;
    }
    return config;
};

const ScannerPlugin = (props) => {

    useEffect(() => {
        // when component mounts
        const config = createConfig(props);
        const verbose = props.verbose === true;
        // Suceess callback is required.
        if (!(props.qrCodeSuccessCallback)) {
            throw "qrCodeSuccessCallback is required callback.";
        }
        const html5QrcodeScanner = new
        Html5QrcodeScanner(qrcodeRegionId, config, verbose);
        html5QrcodeScanner.render(props.qrCodeSuccessCallback,
```

```
        props.qrCodeErrorCallback);

        // cleanup function when component will unmount
        return () => {
            html5QrcodeScanner.clear().catch(error => {
                console.error("Failed to clear html5QrcodeScanner. ",
            });
        };
    }, []);

    return (
        <div id={qrcodeRegionId} className='scanner-container' />
    );
};

export default ScannerPlugin;
```

The code is made up of two React components called ResultContainerTable and ResultContainerPlugin that are in charge of handling and presenting the outcomes of QR code scanning.
1) results from filterResults:
This method returns a filtered array of results from an input array of results. The implementation given just produces a new array and copies every element from the input array, yielding the same outcomes.
2) mode(array): The mode (most common element) of an array is determined by this function. Iterating over the items of an array provided as input, it establishes the mode. If the mode element is present, the method returns it; otherwise, it returns null.
3) ResultContainerTable(data), creates a table to display the results of the scan. The results array is represented by the data prop. The table, thead, and body elements of HTML are used to specify the table's structure. Utilizing the findings, the results are mapped over.Each result from the map() function is displayed as a row in the table along with the decoded text and the QR code's format.
4) ResultContainerPlugin(props): The container for displaying

the scanned results and controlling associated actions is represented by this component. To manage HTTP requests and gain access to the authentication context, it makes use of the useHttp and useContext hooks. To offer translation capabilities, utilize the useTranslation hook.
Calling the filterResults method on the props.results array returns the filtered results.
If there are less than 10 filtered results, the component conditionally presents a "Scanning..." message; otherwise, it produces an empty div>.
The props are activated by the clear-button button.The clearResult() method resets a number of results-related variables and states.
If there are 10 filtered results, the ItemCard component is conditionally displayed to show the finalResult obtained by using the mode method on the filteredText array.

Filter plugin:

```
import React from 'react';
import { useState, useCallback } from 'react';
import { useHttp } from '../hooks/http-hook';
import ItemCard from
'../Component/PageComponents/Scanner/ItemCard';
import { useContext } from 'react';
import { AuthContext } from '../context/AuthContext';
import { useTranslation } from "react-i18next";


function filterResults (results) {
    let filteredResults = [];
    for (var i = 0; i < results.length; ++i) {
        filteredResults.push(results[i]) }
    }
    return filteredResults;
}

function mode(array)
{
    if(array.length == 0)
```

```
        return null;
    var modeMap = {};
    var maxEl = array[0], maxCount = 1;
    for(var i = 0; i < array.length; i++)
    {
        var el = array[i];
        if(modeMap[el] == null)
            modeMap[el] = 1;
        else
            modeMap[el]++;
        if(modeMap[el] > maxCount)
        {
            maxEl = el;
            maxCount = modeMap[el];
        }
    }
    return maxEl;
}

const ResultContainerTable = ({ data }) => {
    const {t} = useTranslation()
    const results = filterResults(data);
    return (
        <table className='qrcode-result-table'>
            <thead>
                <tr>
                    <th>#</th>
                    <th>{t('Decoded Text')}</th>
                    <th>{t('Format')}</th>
                </tr>
            </thead>
            <tbody>
                {
                    results.map((result, i) => {
                        return (
                            <tr key={i}>
                                <td>{i + 1}</td>
                                <td>{result.decodedText}</td>
                                <td>
                                {result.result.format.formatName}
```

```
                                </td>
                            </tr>
                        );
                    })
                }
            </tbody>
        </table>
    );
};

const ResultContainerPlugin = (props) => {
    const {t} = useTranslation()
    const {request} = useHttp()
    const {token} = useContext(AuthContext)
    let finalResult = ''
    let results = filterResults(props.results);
    let filteredText = [];
    results.map(x => filteredText.push(x.decodedText))
    const [data, setData] = useState([])
    let showItem = false

    if(filteredText.length === 10) {
        finalResult = mode(filteredText)
        showItem= true
    }
    return (
        <div className='result-container'>
            <div className='result-header'>{t('Scanned
            results:')} </div>
            <div className='result-section'>
                {filteredText.length < 10 ? <div>
            {t('Scanning...')}</div> : <div></div>}
            </div>
            <button className='clear-button' onClick={()=> {
                props.clearResult()
                finalResult = ''
                filteredText = []
                results = []
                showItem = false
            }}>{t('Clear')}</button>
```

```
            {showItem && <ItemCard number={finalResult}/>}
        </div>
    );
};

export default ResultContainerPlugin;
```

1.4 Report generator

```
# The ListItemsPage component of the code has a generateReport
function that is in charge of producing reports.

GenerateReport:
This asynchronous method generates a report by
sending a POST request to the /api/reports/ endpoint.
When the "Generate report" button is pressed, it is activated.
Using the request function from the useHttp hook, an HTTP
request is made inside the function.
The body of the request contains the classroom ID (id).
By adding the token in the request headers, the request is
authenticated.
Using NotificationManager.success, a success notice is shown if
the request is successful.
If a request-related issue arises, it is recognized and dealt
with without generating an error notice.
In general, the generateReport function accomplishes its goal of
starting the production of a report by making an HTTP request to
the proper endpoint and responding to it appropriately.

    const generateReport = async () => {
        try {
            const response = await request('/api/reports/','POST',
            {classroom: id},
            {'Authorization': 'Bearer ' + token})
            NotificationManager.success('You can check your
            report in profile!')
        }catch(e) {}
    }
```

User page

A list of reports is rendered in the UserPage component using the code you supplied.
claims State:
Using the useState hook, the reports state is started as an empty array.
It will keep a list of the reports that were retrieved from the server.
Fetching and Storing Reports: A request is sent to get user information, including a list of reports, inside the fetchInfo function.
The setUserData method is used to save the received data in the userData state.
The setReports method is used to retrieve the list of reports from the userData object and save it in the reports state.
Creating the Reports List:
A message reading "No reports..." is displayed if there are no reports (reports.length === 0).
A table is produced to display the report information if there are reports (reports.length!== 0).
The reports array is mapped over to produce the list variable.
Each report is displayed as a table row (tr>) with a button to see the PDF report and the report's date and time (x.dateTime).
Window.location.replace(x.reportUrl) is used to open the report URL in a new window when the button is clicked.
Overall, the UserPage component retrieves the list of reports from the server, stores them in the reports state, and renders them as a table.

list of reports:

```
const list = reports.map(x => {
        return <tr>
            <td><Moment format={'dddd DD.MM.YYYY hh:mm a'} date=
            {x.dateTime}/>

            </td>
            <td><button onClick={() =>
            window.location.replace(x.reportUrl)}>{t('Show PDF')}
```

```
            </button></td>
        </tr>
    })
```

1.5 Image viewer

```
# The ListItemsPage component's code implements an image viewer
using the react-simple-image-viewer package.
listOfImages State: Using the useState hook, an empty array is
initialized for the listOfImages state.
It will keep a list of the images that will be shown in the
image viewer.
Opening the Image Viewer: Using the useState hook, the
isViewerOpen state is set to false.
The useState hook sets the currentImage state's initial value to
0.
The useCallback hook is utilized to specify the openImageViewer
function.
It requires an index argument that represents the index of the
displayed picture.
The currentImage state is changed to the supplied index inside
the openImageViewer method.
To open the image viewer, the listOfImages and isViewerOpen
states are modified appropriately.
The currentImage state is reset to 0 and the image viewer is
closed by the closeImageViewer function.
The ImageViewer component's src attribute is set to the
listOfImages state.
The ImageViewer component's currentIndex prop is set to the
currentImage state.
There are also more components available for customisation, like
disableScroll, closeOnClickOutside, and backgroundStyle.
To handle shutting the image viewer, the onClose prop of the
ImageViewer component is assigned to the closeImageViewer
function.
Users may explore and browse through a picture gallery using the
capability of the image viewer. The image URLs that will be
shown are changed in the listOfImages state, and the image
viewer is opened and closed based on the isViewerOpen state.
```

```
    Image viewer:

const [listOfImages,setList] = useState([])
    const [currentImage, setCurrentImage] = useState(0);
     const [isViewerOpen, setIsViewerOpen] = useState(false);
     const openImageViewer = useCallback((index) => {
            setCurrentImage(index);
            console.log(listOfImages);
            setIsViewerOpen(true);
        }, []);

        const closeImageViewer = () => {
            setCurrentImage(0);
            setIsViewerOpen(false);
        };


    useEffect(() => {
        fetchItems()
    }, [fetchItems])

    Return:
      <div className="list-items-page">
            {isViewerOpen && (
        <ImageViewer
          src={ listOfImages }
          currentIndex={ currentImage }
          disableScroll={ true }
          closeOnClickOutside={ true }
          backgroundStyle={{backgroundColor: "rgba(0,0,0,0.8)"}}
          onClose={ closeImageViewer }
        />
      )}
```

1.6 Search

```
For example, search by name function

# When the search status changes, the search feature is
```

activated. To do this, a dependency array is added to the useEffect hook:

```
    useEffect(()=>{
        searchItems()
    }, [search])

# The search logic is carried out through the searchItems function. It sends an API call with the parameters id and search value to the address /api/item/search/:id/:search. The fetchItems method is used to get all things if the search value is empty. Following that, the response data is set in the item's state:

    const searchItems = async () => {
        try{
            if(search.search.length === 0) {
                fetchItems()
                return
            }
            const data = await request('/api/item/search/'+id+
            '/'+search.search,'GET
            ', null, {'Authorization': 'Bearer ' + token})
            setItems(data)
        }catch(e){}
    }

# The ListItemsPage component has the feature to search by name. The useState hook is used to set the search input's initial state: const [search, setSearch] = useState('search: ''). The current value of the search input is stored in the search state.

The search input's change event is handled by the handleChange function. The new value entered by the user updates the search state:
    function handleChange(e){
        setSearch({...search, [e.target.name]: e.target.value})
    }

# The placeholder property shows the "Search by name"
```

translation key that is available. The input's value is set to search.search, which is the value of the search state object's search property. As a result of the handleChange function's connection to the onChange event, when the user inputs, the input can change the search state.

```
    return (
            <div>
            <input placeholder={t('Search by name')} className =
            'list-items-input1' type={'text'} name={'search'}
            value={search.search} onChange={handleChange}/>
            </div>
    )
}
```

Appendix B Backend Code

2.1 Entities

```
#Every table stored in database has it's entity class with
multiple datatypes attached, for example the Classroom entity
class:

public class Classroom : Entity
    {
        public string ClassroomName { get; set; }
        public string IconUrl { get; set; }
        public string Description { get; set; }
    }
```

2.2 Providers

```
#Every entity has has it's provider class, which stores
functions that performs actions and changes data in
database tables. Here is how the EntityClassroomProvider
initialized:

public class EntityClassroomProvider :
EntityProvider<ApplicationContext, Classroom, Guid>
                , IClassroomProvider{}

#As an example it has SearchClassroomByQuery function:

public async Task<List<SearchResult>> SearchClassroomByQuery
    (string category, int numberOfItems)
        {
            try
            {
                var result = _context.SearchResults.FromSql
                <SearchResult>($"select distinct classrooms.*,
                count(*) as number_of_items from classrooms
                join items on items.classroom_id =
                classrooms.id join categories on
                categories.id = items.category_id where
                categories.name = {category} group by
                classrooms.id having count(*) >=
```

```
                {numberOfItems}").ToList();

                return result;
            } catch(Exception)
            {
                throw;
            }
        }
```

2.3 ApplicationContext

```
#In the Providers folder the ApplicationContext class
exists, where Dbsets initialized:

public DbSet<User> Users { get; set; }
public DbSet<Image> Images { get; set; }
public DbSet<Item> Items { get; set; }...

#with the unique rows adjusted:

modelBuilder.Entity<User>().HasIndex(u => u.Email).IsUnique();
modelBuilder.Entity<Classroom>().HasIndex
        (u => u.ClassroomName).IsUnique();
```

2.4 Services

```
#The services folder contains in total 3 services. They are:
AwsS3UploadService, which uploads images to the S3 storage,
ReportGeneratorService, which generates data reports
and UserAuthenticationService, which is responsible for
authentication, createion and decryption of JWT tokens.

public class AwsS3FileUploadService
public class ReportGeneratorService
public class UserAuthenticationService

#The activities are implemented via functions such as
Authenticate function in UserAuthenticationService:

public async Task<(string, int?)> Authenticate
```

```
                (string login, string password)
{
    try {
        var admin = await _userProvider
                    .GetByEmail(login);

        if (!BCrypt.Net.BCrypt.Verify(password, admin
                                        .Password)) {
            throw new ArgumentException
                    ("Incorrect password");
        }

        return (GenerateJwtToken(admin.Email,
        admin.Role), (int) admin.Role);
    }
    catch (ArgumentException e)
    {
        return (null, null);
    }
}
```