****

Faculty of Contemporary Sciences and Technologies

**Software Engineering**

**(Sem. 5, 2022/2023)**

**Project:**

***“Banking Management System”***

**Student(s)**:

Jon Fetahi

Agon Muharemi

Arijan Pollozhani **Mentor**:

Rinor Dalipi Prof. dr. Besnik Selimi

January 2023, Tetovo

Contents

[1. Introduction 2](#_Toc126869405)

[1.1. Overview 2](#_Toc126869406)

[1.2. Stakeholders: 4](#_Toc126869407)

[2. Requirements 5](#_Toc126869408)

[2.1. Functional Requirements 5](#_Toc126869409)

[2.2. Non-functional Requirements 6](#_Toc126869410)

[2.3. User-Requirements 6](#_Toc126869411)

[3. Architecture 7](#_Toc126869412)

[3.1. Overview 7](#_Toc126869413)

[3.2. Database Design 7](#_Toc126869414)

[4. Design: 8](#_Toc126869415)

[4.1. UML diagram 8](#_Toc126869416)

[4.2 Design Patterns 11](#_Toc126869417)

[5. Implementation 12](#_Toc126869418)

[5.1. Programming Languages 12](#_Toc126869419)

[5.2. Testing strategy 13](#_Toc126869420)

[6. Testing 13](#_Toc126869421)

[6.1. Test cases 13](#_Toc126869422)

[6.2. Test results: 14](#_Toc126869423)

[7. Deployment 14](#_Toc126869424)

[7.1. Installation instructions 14](#_Toc126869425)

[7.2. Deployment process 15](#_Toc126869426)

[8. Maintenance 16](#_Toc126869427)

[8.1. Patching: 16](#_Toc126869428)

[8.2. Upgrades 16](#_Toc126869429)

[References 17](#_Toc126869430)

# Introduction

* 1. Overview

A banking system is a software application designed to handle financial transactions and banking operations in a secure, efficient, and streamlined manner. In this project we will be building a Banking Management System which will include a website and an ATM software which users will use to withdraw money from and other functions.

As software engineers our objective is to create a perfect website and an ATM software system that provides a seamless easy user experience while also meeting the demanding requirements of the financial industry and the requirements of our clients. The system must be able to handle a wide range of functions, from deposits and withdrawals to registering new users and money transfers and must ensure that all information is securely stored and protected.

We will be working in a group of four engineers, where each of us will bring unique skills to the table. By combining our knowledge and expertise, we will be able to tackle complex problems and find creative solutions. For this project we have projected an estimate 40-45 work hours per week 5 days a week, this might extend to a 60 hours per week depending on the workload we might encounter during development.

We will be working in an agile environment, which means that we will prioritize flexibility and collaboration in our approach to our project, due to the reason being that agile environments are beneficial in a way that allows us to quickly adapt to changes in project requirements and adjust changes as needed, ensuring that we will deliver the best possible outcome for our clients. Agile environments enable us to routinely review and realign our objectives, keeping us on track and concentrated on what matters most. This strategy enables us to work more quickly, effectively, and efficiently, making it the best way to succeed in today’s fast-paced, dynamic business environment.

* 1. Stakeholders:

In this project, it is crucial to know the stakeholders and their needs and expectations, by identifying who the stakeholders are it allows us to build relations, manage expectations and address any concerns they may have. This is essential to the projects success and ensures us that all parties involved are satisfied. Understanding the stakeholders is important as we can identify any potential risk and to make inform decisions to support the overall success of the project. One of the most important stakeholders is the client, the reason being that the software is build according to their requirements, in our case our client is American Express Banking located in New York City, United States,

With that being said here are all of the parties involved in the development of this software and their roles:

* **The Client**: The client is one of the most important of the stakeholders, as the project is being build based on their requirements, in this case our client is American Express Banking.
* **Customers**: These individuals or businesses are who use the banking services and offered by the system. They expect the system to be secure, reliable, and accessible.
* **Shareholders**: Shareholders are the owners of the bank and are interested in the financial performance of the system, including profitability, growth, and return on investment.
* **Employees**: Employees of the banking system include tellers, loan officers, and other staff members who interact with customers and help to manage the day-to-day operations of the bank.
* **Regulators**: Regulators are government agencies that oversee the banking industry to ensure compliance with laws and regulations, protect consumers, and maintain stability in the financial system.
* **Other Financial Institutions**: Other banks, and other financial institutions are also stakeholders in this banking system. They may interact with the system through inter-banking transactions, partnerships, and other agreements that they might have.

1. Requirements

## Functional Requirements

As mentioned, the requirement specification is the key to making the project successful, therefore is crucial that the requirements are properly defined and met, the Website software needs to accomplish most of the functions, while the ATM will accomplish less than the website online. The Functional Requirements for the website are:

* **To have a Landing Page:** for this project to look completed the client has requested for a landing page to be implemented so that the users can access it first and from there to navigate to different pages.
* **Register New Clients:** Users can use the website to activate their account online by inputting their personal information and they will be given an account which they can access with username and password protection and pin protection if they are accessing from ATM.
* **Transfer Money to different accounts:** Users will have a function to transfer money to different account, by knowing the other recipients Account Number and CUSIP number, which is an American banking number, they will be able to transfer money quickly with a few clicks of a button.
* **Payment/Transaction History:** With being created an account users will also have a page where they can see their transaction history, they will be shown the date and time when the transaction occurred, the amount they spent or gained and it will show if the spent or gained in that certain transaction, all the data will be shown with a click of a button named “Show History”.
* **Check Balance:** With having an account and being able to transfer money, users will also have a page where they will be shown the amount of funds the user currently has, with a replica of a Debit or Credit Card shown in the display above, and it’s worth mentioning that the currency will be in USD$.
  1. Non-functional Requirements

Non-functional requirements are as important as functional requirements, these requirements refer to requirements that specify the quality attributes and constraints of a system, when it comes to security, reliability etc.

Here are the Non-functional requirements for our project:

* **Security and Reliability:** The software needs to be secured due to the reason that clients have personal information stored, and the clients need to be assured that their information will not be leaked in any way.
* **Maintainability:** It makes it possible to add or change features confidently, while not avoid making improvements for fear of breaking the existing software.
* **Scalability:** It will be designed for large scale use, to the ever-growing clients that will use this software in the future.
* **Usability**: Ensuring that the system is easy to use and provides a user-friendly interface for performing banking activities.

## **User-Requirements**

When it comes to user requirements it refers to the specific needs and expectations of the users who will interact with the system. In the context of our system the user requirements would include:

* **Account Registration**: Ability for customers to apply for loans online and to track the status of their loan applications.
* **Account Management**: Ability for customers to view their account information, including balances, transaction history, and account statements.
* **Money Transfers**: Ability for customers to transfer funds between their own accounts and to external accounts.
* **Customer Service**: Ability for customers to communicate with customer service representatives and receive support through the Contact Us page.
* **Mobile Access**: Ability for customers to access the software from their mobile devices, such as smartphones and tablets.
* **Cash Withdrawal**: Ability for users to withdraw cash from their accounts using the ATM system.

1. Architecture:

## Overview

Considering that our banking system includes a website, and an ATM software, the system will consist of several key components. The system would have a centralized database that stores customer and transaction information. The Website and the ATM system would be connected to the database through a network and with the help of APIs, allowing for real-time updates to customer accounts and transaction information.

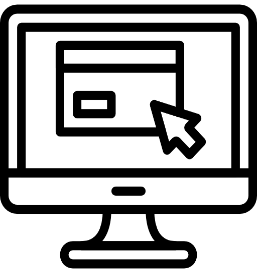
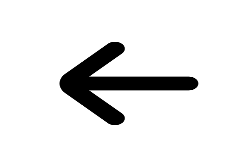
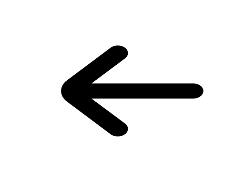
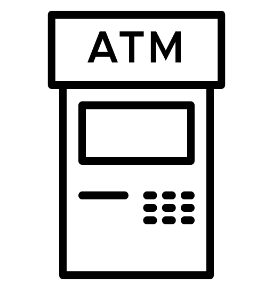


Figure 1. System Interaction

Customers would have access to their accounts via the website component using a user-friendly interface, enabling them to see account balances, conduct transactions, and view transaction history.

While through the ATM system component customers would have physical machine access to their accounts, enabling them to conduct operations like cash withdrawals, transferring money and check the balances of their accounts.

## Database Design

As referred to the figure above, both the Website and the ATM system would communicate through a centralized database using secure protocols to ensure the confidentiality and integrity of customer data. The system will be deigned in a way such that it would have security mechanisms in place to authenticate customers, prevent unauthorized access to customer data, and ensure the privacy of customer transactions.

Considering that it will be used by a banking company, the database needs to be able to handle large volume of data, and expectations to grow during the future, that’s why when building we need to take into consideration scalability, a scalable system should be able to handle increased traffic, data storage, and computational requirements without sacrificing performance or availability, including adding additional resources (such as servers, storage, and network bandwidth).

1. Design:
   1. UML diagram

With the help of UML, we can visualize use cases for the system, here are some example UMLs to better understand how the system works.

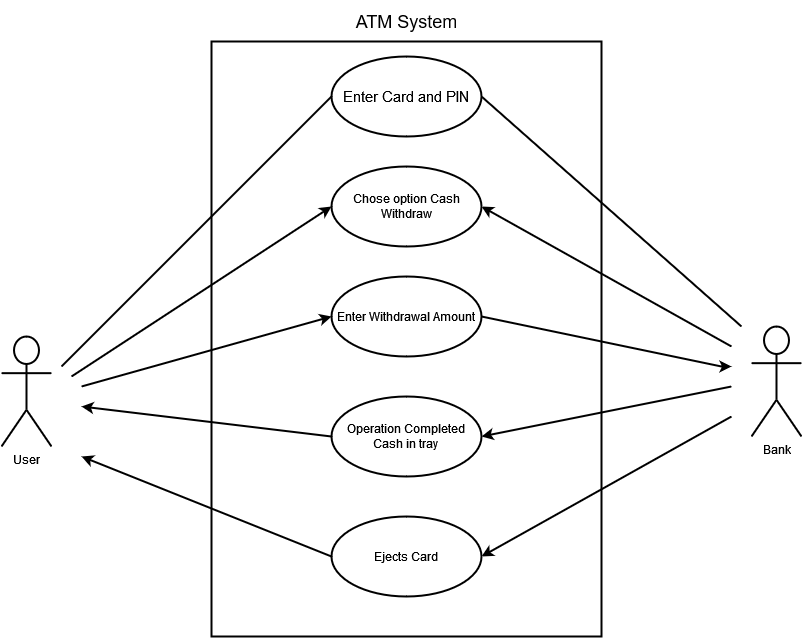
-Use Case for Withdrawing cash from ATM

Figure 2. UML 1

Here we can see the user inserting his personal Card and entering his PIN number, then choses the option for withdraw, enters the amount of money he wants to withdraw, the bank records his transaction, verifies it, and deposits the cash in the tray, then the card is ejected out for the user to take back.

* + Use Case for transferring money

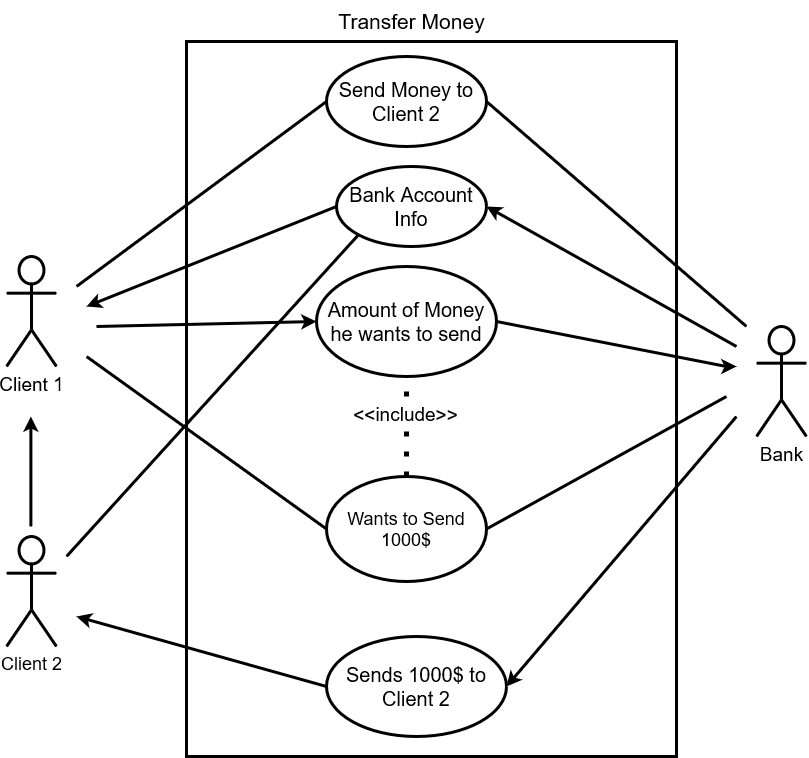


Figure 3. UML 2

Here we can see a use case when a client wants to transfer money to another client, first he inputs the information of the client he wishes to send, including the amount of money he wants to send, the system records the transaction verifies it and the money is sent to the other client.

* + Chart

    Description automatically generatedSequence diagram for registering users

Figure 4. Sequence Diagram

This sequence includes the bank employee registering the account for the customer.

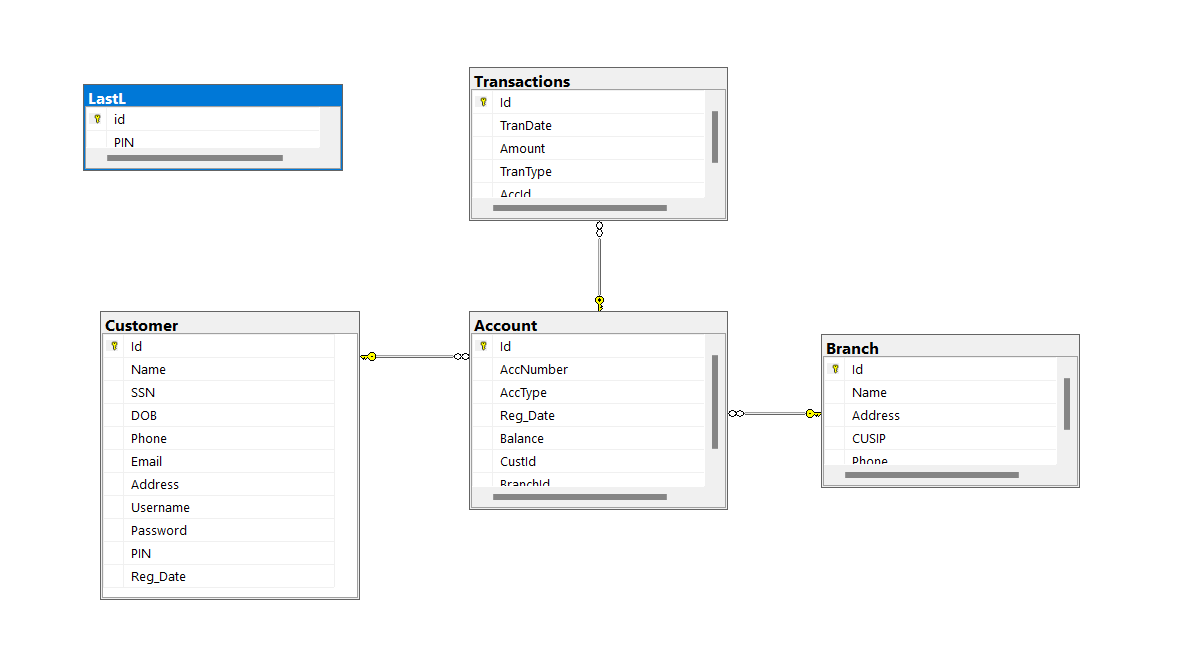
* + Class Diagram

Figure 5. Class Diagram

## Design Patterns

We as a team have decided to employ the Model-View-Controller (MVC) pattern for the banking system's architectural design. Where the website is broken down into three distinct components or parts using the MVC architectural pattern: the Model, which represents the data and business logic; the View, which shows the data to the user; and the Controller, which processes user input and modifies the Model and View as necessary.

Diagram

Description automatically generated

This pattern offers better concern separation, increased maintainability, and improved testability, among other advantages for software development.

The banking system can be created and developed in a modular and scalable way by utilizing the MVC pattern, making it simpler to add new features or update existing ones without affecting the rest of the system.

In the case of our banking project, the Model component would represent the customer and transaction data and other data that will be stored, the View component would display the user interface, and the Controller component would manage user input and direct interactions between the Model and View. One advantage while using this design pattern is that we can simply change the user interface or add new features thanks to this architecture without having an impact on the underlying data and business logic.

Figure 6. Model-View-Controller Pattern

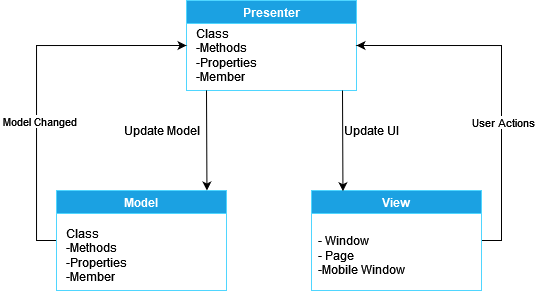
As for the ATM system, we will use the “Model-View-Presenter” design pattern, an MVP pattern is commonly used in Windows Forms. The presentation’s code oversees most of the business logic. As you can see, oftentimes the View sits in the presentation layer, so this design can be redundant. [1]

Figure 7.Model-View-Presenter Pattern

1. Implementation:
   1. Programming Languages.

For building the project, our team has chosen for the Front-End side to use a combination of ASP.NET, ASP.NET is a free web framework for building great websites and web applications using HTML, CSS, and JavaScript. The ASP.NET MVC gives you a powerful, pattern-based way to build dynamic websites that enables a clean separation of concerns and that gives you full control over markup. ASP.NET MVC includes many features that enable fast, friendly development for creating sophisticated applications that use the latest web standards. [2]

We will also utilize JavaScript for any client-side scripting or programming, as for styling and designing the layout of the Front-End, we will also use the Bootstrap CSS stylesheets,

Figure 8.ASP.NET

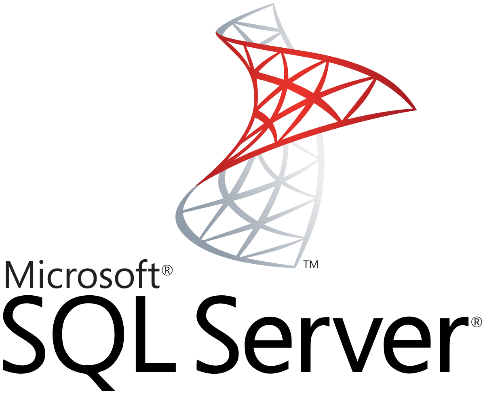
Bootstrap is a free, open-source front-end development framework for the creation of websites and web apps. Designed to enable responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template designs. [3]

Icon

Description automatically generated One of the key features of Bootstrap is its redundancy, meaning that it provides a number of pre-built components and styles that can be used to build a user interface. This helps the developing team by reducing the amount of time and effort required to build a user interface from scratch and helps us to ensure that the user interface is consistent and aesthetically pleasing.

For the ATM system we have chosen to use WinForm, a powerful platform for building desktop applications.

Figure 9. Bootstrap Logo

Windows Forms is a UI framework for building Windows desktop apps. It provides one of the most productive ways to create desktop apps based on the visual designer provided in Visual Studio. Functionality such as drag-and-drop placement of visual controls makes it easy to build desktop apps. [2]

And lastly, to store and manage all of the customer and transaction data, we have chosen to use SQL Server as our database platform. SQL Server is one of the main relational database management systems on the market that serves a wide range of software applications. For companies, using this tool is essential because of the facilities it offers and the utilities it has, SQL Server is a robust and reliable database management system that is optimized for large-scale data storage and retrieval. [4]

Figure 10.SQL Server

* 1. Testing strategy

For testing we have decided for our project to run Unit Testing method over Integration Testing, the reason being that unit tests are faster than integration tests, and with unit testing we are able to test small pieces or modules individually, rather multiple modules together. Then we can verify that each module is working properly, and if the one specific module isn’t working properly, we can target that specific module without affecting the entirety of the project.

The tests are going to be in the form of a Test Automation Strategy, meaning that the tests that will be written will be automated so they can be run quickly and easily, and so that they can be regularly executed to detect any issues that might arise as the code develops.

Given that the developing is being written in the .NET framework we will use the built in MS Unit framework to write the testing codes.

1. Testing:
   1. Test cases

Given that we are going to be using Unit Testing we are going to write testing for some key function’s ore testing cases, we have outlined some test cases that we have tested so far for example:

**Login**: Test the login functionality for both customers

**Email Validation**: Test the given email for registration is a valid email.

**Username Validation and Length:** Test the given username length is acceptable and is a valid username not containing any symbols or numbers.

**Transfer Properties and Funds:** Test the transfer of funds, and the account that they are transferring to are valid information’s given

* 1. Test results:

After conducting the series of test that are given above, we encountered many tests that resulted in failure, despite the initial setback, our team set back to be able to find solutions to address these issues and to make the system more functional. We were able to successfully address the problems and achieve our objectives thanks to our dedication to excellence and ambition to provide a high-quality product. We were able to troubleshoot and rectify issues as seen by the test results that indicated a considerable improvement once the modifications were applied.

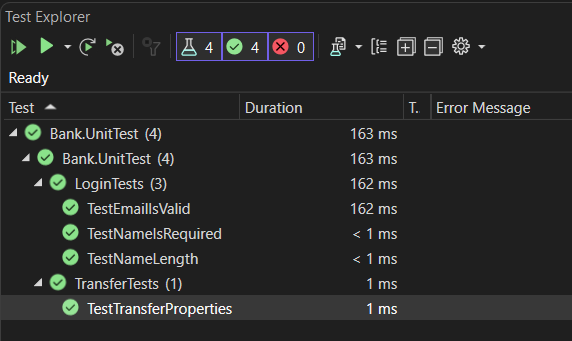


Figure 11. Test Results

1. Deployment:
   1. Installation instructions

After the project is completed in development and testing, the next stage of the process is deployment, or rather delivering the project to the client.

The installation process will only apply for the ATM software, which will be in an executable file, and for the software to work properly with the ATM machine, we will need to send groups of engineers around the country of the United States to install the software manually on site and conduct any necessary configurations so that the systems work properly together.

7.2. Deployment process

As for the online website and database system, after concurring with our client we feel the best decision is for the remaining systems to be deployed in cloud rather than having an on-premises data center.

Icon

Description automatically generatedWe came to the conclusion after our team of analysts provided us with a report stating that, the system in which we have built will be used by over 100 million users around the country, and this number can increase according to the report provided to us by our company’s Analysts. The reason we went with cloud deployment is that servers on cloud are easy to upgrade and expand and it is cheaper, and will be maintained by the cloud providers, while if we are using an on-premises data center the ability to upgrade and expand is costly and time consuming, given that new pieces of hardware need to be bought if there is the need to accommodate new users, and there is the need of having an IT department that will maintain the servers are operational 24/7.

Figure 12. Microsoft Azure

For the cloud providers we have chosen Microsoft Azure platform, given that the project we have developed is built with Microsoft development tools we feel its only fair if we use Azure for our cloud providers.

And with using Azure, we can see from the activity log, visualized data that is being used and have our team of analysts conduct research for if the servers need to be expanded in capacity, also given we hold sensitive information about our customer, azure provides data protection encryption at rest aligned with 256-bit AES standard and encryption in transit through TLS 1.2. You can also have double encryption by securely managing your keys using Azure Key Vault to encrypt the keys that you are using.

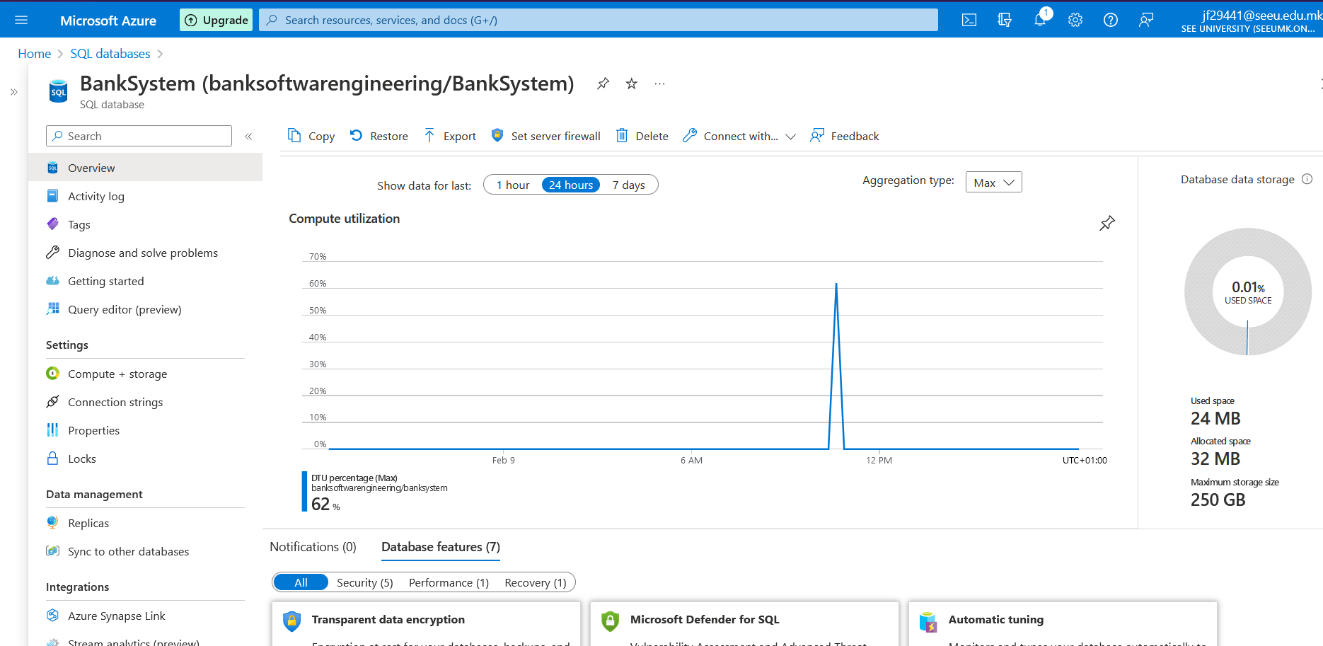


Figure 13.SQL Database after deployment on cloud

1. Maintenance:
   1. Patching:

After deployment the next phase is the most challenging, it’s maintaining the system so that its up to date, and to not have issues. We in agreement with the client, we are responsible for maintaining the system after deployment.

There are a few of patching processes that we will take into consideration if there are needs patching.

**Vulnerability Assessment**: Identifying any potential vulnerabilities in the system is the first step in the patching procedure. Regular security audits, penetration testing, or keeping an eye out for any security concerns that have been reported can all help with this.

**Patch creation**: After a vulnerability has been found, the next step is to create a patch to address the problem. Working with the program vendor or writing the fix internally may be required. Before being put into use, the patch needs to be thoroughly tested.

**Deployment**: To prevent any disturbance to the banking activities, the patch should be installed during a maintenance window. To ensure that the patch does not have a detrimental effect on the system, the deployment should be extensively tested.

Even after the issues are address, we will continue to monitor the system after the patch has been deployed, for any potential security threats or any other issues that may arise.

* 1. Upgrades

In terms of upgrading the system, our policy is to regularly do maintenance checks on all the system components that we have developed, to discover which component needs fixing or is outdated and needs to be upgraded to the news version that is available in the market.

However, to make upgrades to the software we must go over a few things, for example:

* **Planning**: A plan should be in place prior to the upgrade to guarantee that it is carried out successfully and with the least possible impact on the banking operations.
* **Preparation** The next step is to get the system ready for the upgrade. Data backups, system testing, and any required software updates that may all be part of this process.
* **Implementation of the Upgrade**: After all necessary planning has been done, the Upgrade can be put into action. This can entail introducing fresh hardware to the system or installing or updating fresh software.
* **Testing**: After the upgrade has been put into place, it is crucial to give the system a complete test run to make sure everything is operating as it should and that no new features are malfunctioning.
* **Deployment**: The upgrade may be implemented after testing is finished. To minimize any interference with banking activities, this should be done during a maintenance window.

# References

|  |  |
| --- | --- |
| [1] | bignerdranch, “bignerdranch.com,” [Online]. Available: https://bignerdranch.com/blog/which-net-design-pattern-is-best-for-your-next-project/. |
| [2] | microsoft, “ASPNET,” [Online]. Available: https://learn.microsoft.com/en-us/aspnet/overview. |
| [3] | techtarget, “What Is Bootstrap,” [Online]. Available: https://www.techtarget.com/whatis/definition/bootstrap. |
| [4] | intelequia, “What is SQL Server,” intelequia.com/, [Online]. Available: https://intelequia.com/en/blog/post/2948/what-is-microsoft-sql-server-and-what-is-it-for. |