Dynamic Dashboard

Submitted By:
Hasin Mahjabeen
Student ID: 0242220005343009
Department of Software Engineering
Daffodil International University,
Daffodil Smart City

Supervised By:
Afsana Begum
Assistant Professor and Program Coordinator of M.Sc.
Department of Software Engineering
Daffodil International University,
Daffodil Smart City

This project report has been submitted in fulfillment of the requirements for the Degree of Masters of Science in Software Engineering.

APPROVAL

This project which is titled “Dynamic Dashboard”, submitted by Hasin Mahjabeen, ID: 0242220005343009 to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Masters of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Dr. Imran Mahmud
Associate Professor and Head
Department of Software Engineering
Daffodil International University

Chairman

Afsana Begum
Assistant Professor
Department of Software Engineering
Daffodil International University

Internal Examiner 1

Dr. Md. Fazla Elahe
Assistant Professor and Associate Head
Department of Software Engineering
Daffodil International University

Internal Examiner 2

Md. Tanvir Quader
Senior Software Engineer
Technology Team
a2i Program

External Examiner

DECLARATION

I declare that the work described in this report was done in the Software Engineering Department at Daffodil International University, under the guidance of Afsana Begum. If I used materials from other projects, I mentioned them. I promise that, to the best of my knowledge, none of this report has been submitted for a degree anywhere else before.

Supervised By:

Afsana Begum
Assistant Professor and Program Coordinator of M.Sc.
Department of Software Engineering
Daffodil International University,
Daffodil Smart City

Submitted By:

Hasin Mahjabeen
Student ID: 0242220005343009
Department of Software Engineering
Daffodil International University,
Daffodil Smart City

ACKNOWLEDGEMENT

I would like to express my gratitude to the Almighty Allah (SWT) for guiding me through this M.Sc study and the accompanying examination. A heartfelt thanks to my parents, a constant pillar of support throughout my life, and to my spouse, who stood by me during the M.Sc program, providing unwavering encouragement. Their faith in me has always been a source of confidence and conviction, motivating me to achieve my goals. Special appreciation goes to my well-wisher, whose continuous inspiration fueled my creative endeavors.

I extend my sincere respect and gratitude to my supervisor, Afsana Begum madam, for her support in completing this study and for encouraging my research efforts and project implementation.

Additionally, I would like to thank Dr. Imran Mahmud, Associate Professor and Head in Charge of the Department of Software Engineering at Daffodil International University, for motivating us to pursue quality research. Thanks also to our dedicated teachers who have supported us consistently throughout our graduate journey.

Last but not least, a big thank you to all my coursemates who completed the coursework with me as well as to all the staff of the Department of Software Engineering and Daffodil International University for their continuous efforts in providing us with excellent facilities in classrooms and laboratories.

ABSTRACT

The project, Dynamic Dashboard, is like a smart tool that helps people understand data better. Using the power of ASP.NET Core and C#, I've created a dashboard that does not only show data – it also seamlessly updates in real-time whenever new information becomes available.

What makes this dashboard special is that it's like a universal translator for data. Instead of sticking to one database system, REST API is used. This means any organization, anywhere, can easily share their data with this dashboard. It's like making data friends with everyone!

This dashboard is user-friendly, allowing people to explore and customize views effortlessly. It’s made sure that it's secure and adaptable to different organizations' needs. It's not just a tool for today; it's ready for the future of data, evolving and growing with the way people want to understand information.

In a nutshell, Dynamic Dashboard is here to make data fun and useful for everyone, no matter where they are or what kind of data they have. It's like having a super-smart friend who speaks the language of data and helps people make smarter decisions.

TABLE OF CONTENTS

Chapter 1 1
Introduction 1
1.1 Objectives 1
1.3 Differentiators 1
1.4 Project Summary 2
Chapter 2 3
System Analysis 3
2.1 Methodology Used 3
2.1.1 Incorporating Iterative and Incremental Model 3
2.1.2 Incorporating Agile Practices 3
2.1.3 Incorporating Waterfall Model 3
2.1.4 Incorporating Hybrid Approach 3
2.2 Feasibility Study 4
2.2.1 Operational Feasibility 4
2.2.2 Technical Feasibility 5
2.2.3 Economic Feasibility 5
2.2.4 Legal Feasibility 5
2.2.5 Scheduling Feasibility 5
2.3 Time Estimation 6
2.3.1 Time Estimation Calculation 6
2.4 Cost Estimation 6
2.4.1 Cost Estimation Calculation 7
Chapter 3 8
Software Requirement Specification 8
3.1 Functional Requirements 8
3.1.1 User Authentication 8
3.1.2 Dashboard 8
3.1.3 Full Chart View 8
3.1.4 Data Filtering and Sorting 8
3.1.5 Data Sources Integration 8
3.1.6 Responsive Design 8
3.1.7 Export and Sharing 8
3.2 Non-Functional Requirements 8
3.2.1 Performance 8

3.2.2 Scalability 9
3.2.3 Security 9
3.2.4 Reliability 9
3.2.5 Usability 9
3.2.6 Compatibility 9
3.2.7 Maintainability 9
Chapter 4 10
Use Case Diagram 10
4.1 Actors 10
4.2 Diagram 10
4.3 Description 10
4.3.1 Register 10
4.3.2 Login 11
4.3.3 Display 11
4.3.4 View Full Chart 11
4.3.5 Customize Chart 11
4.3.6 Filter and Short Data 12
4.3.7 Export Chart 12
4.3.8 Logout 12
Chapter 5 13
Activity Diagram 13
4.1 Registration 13
4.2 Login 14
4.3 View Dashboard 15
4.4 Customize Charts 16
4.5 Filter and Sort Data 17
4.6 Export Chart 18
Chapter 6 19
Sequence Diagram 19
Chapter 7 19
Data Flow Diagram 19
Chapter 8 20
User Manual 20
8.1 Registration Page 20
8.2 Login Page 21
8.3 Opening Dashboard 21

8.4 Opening Full View of Chart 22
8.5 Change Chart Type 22
8.6 Filter and Sort Data 23
8.7 Stop Dynamic Display and Export Chart 24
Chapter 9 25
Project Summary 25
9.1 Project Link 25
9.2 Technologies Used in Project 25
9.2.1 Backend- ASP.NET Core 25
9.2.2 Data Management- SQL Server 25
9.2.3 REST API for Global Accessibility 25
9.2.4 CanvasJS for Visualization 25
9.3 Integration 25
9.3.1 Data Source 25
9.3.2 Database Setup 26
9.3.3 IDE Selection 26
9.3.4 Project Template Selection 26
9.3.5 Project Configuration 26
9.3.6 Dependencies and Packages 26
9.3.7 Implementation of Business Logic 27
9.3.8 REST API Integration 28
9.3.9 ASP.NET Core Integration 28
9.3.10 CanvasJS Integration 29
9.4 API Implementation Guide 29
9.4.1 Prepare Stored Procedure 29
9.4.2 Configure Connection String 30
9.4.3 Implement API Controller 30
9.4.4 Configure Authentication 30
9.5 Achievements and Limitations 30
9.5.1 Achievements 30
9.5.2 Limitations 30
9.6 Future Plan 31
Conclusion 32
References 32
Account Clearance 33
Plagiarism Report 33

Chapter 1

Introduction

1.1 Objectives

The Dynamic Dashboard Implementation project is an idea to help organizations understand and use their data better. It's like creating a special tool that makes looking at data easy and fun. This tool is not just pretty to look at, but it's also flexible, so you can change it the way you like.

We know that people really need to understand their data well, so this project aims to build a special dashboard. This dashboard will make showing data easy, let you change things the way you want, and connect to different types of data sources without any trouble.

In simple words, we're making a simple and easy way for people to see and make the most out of their data.

1.2 Scopes

The scope of my dynamic dashboard project could stem from several factors:

1. **Customization and Specific Requirements**
    - Existing solutions may not be perfectly aligned with the specific needs and requirements of particular organizations or industry.
    - Building a custom dynamic dashboard allows for tailoring the solution to meet specific criteria.

2. **Global Accessibility and API Integration**
    - The project's emphasis on global accessibility through REST API integration distinguishes it from some existing solutions.
    - This feature allows any company to integrate the dashboard by conforming to a standardized API format, promoting flexibility and interoperability.

3. **Security and Compliance**
    - Some organizations may prioritize building a custom solution to ensure a higher level of control over security measures and compliance requirements, especially in industries with strict regulations.

1.3 Differentiators

The differentiators between my dynamic dashboard project and existing dashboards can be some crucial points that set my project apart and provide unique value. Here are some potential differentiators:

Key Feature	My Project	Differentiator

Real-Time Data Updates	My project offers real-time data updates.	Many existing dashboards may lack real-time updates, requiring manual refresh or relying on scheduled updates.
User Customization	I have allowed users to customize the charts.	Users may not be able to change into different types of charts.
Seamless Data Integration	I have seamlessly integrated with diverse data sources, both internal and external.	Some dashboards may have limitations in connecting to various data sources.
Global Accessibility with REST APIs	RESTful APIs were utilized for global accessibility, allowing users (organizations) from different locations to interact.	Certain dashboards may lack global accessibility features, not all organizations can implement all type dashboards tools in their software.
Responsive Data Visualization	I have implemented various visualization types using CanvasJS scripts, ensuring responsiveness.	Other dashboards may have limitations in terms of visualization options or may not be optimized for responsiveness across datasets.
Scalability	I designed with scalability in mind, accommodating an increasing number of data points.	Not all dashboards may be as scalable.

1.4 Project Summary

My project is a website made using ASP.net core and C#. It takes data stored in SQL Server and turns it into cool charts that we can see on the screen. It makes numbers and information look super nice and easy to understand by turning them into visual charts with help of CanvasJS scripts. So, instead of just staring at boring numbers, we get to see data in a way that makes sense and looks really awesome.

Chapter 2

System Analysis

2.1 Methodology Used

In my project, I found it practical to use elements from various SDLC models. This approach, known as a hybrid or blended SDLC model, acknowledges that different phases of a project may benefit from different methodologies which are mentioned below:

2.1.1 Incorporating Iterative and Incremental Model

I have implemented an iterative and incremental approach when dealing with the data from Bangladesh Bank. The process involved fetching data, modifying it, and preparing stored procedures for CanvasJS. This iterative method allowed for continuous improvement and refinement.

2.1.2 Incorporating Agile Practices

As I worked alone on the project, the need for rapid development and adaptability aligned well with Agile principles. The quick turnaround from requirement analysis to development and testing within three months reflects the agility inherent in the development process.

2.1.3 Incorporating Waterfall Model

My project has a clear breakdown of tasks in phases such as data modification, database design, REST API implementation, ASP.NET Core integration, and CanvasJS visualization resembling a somewhat sequential progression, drawing inspiration from the Waterfall model for certain phases.

2.1.4 Incorporating Hybrid Approach

Within this chapter, I had a dedicated section to explicitly address the hybrid or blended approach I took. I explained how elements from both the Iterative and Agile Models were harmoniously integrated to meet the project's unique requirements effectively.

The hybrid or blended approach I had applied seems suitable, considering the nature of the dynamic dashboard project, where adaptability, continuous improvement, and rapid development are essential.

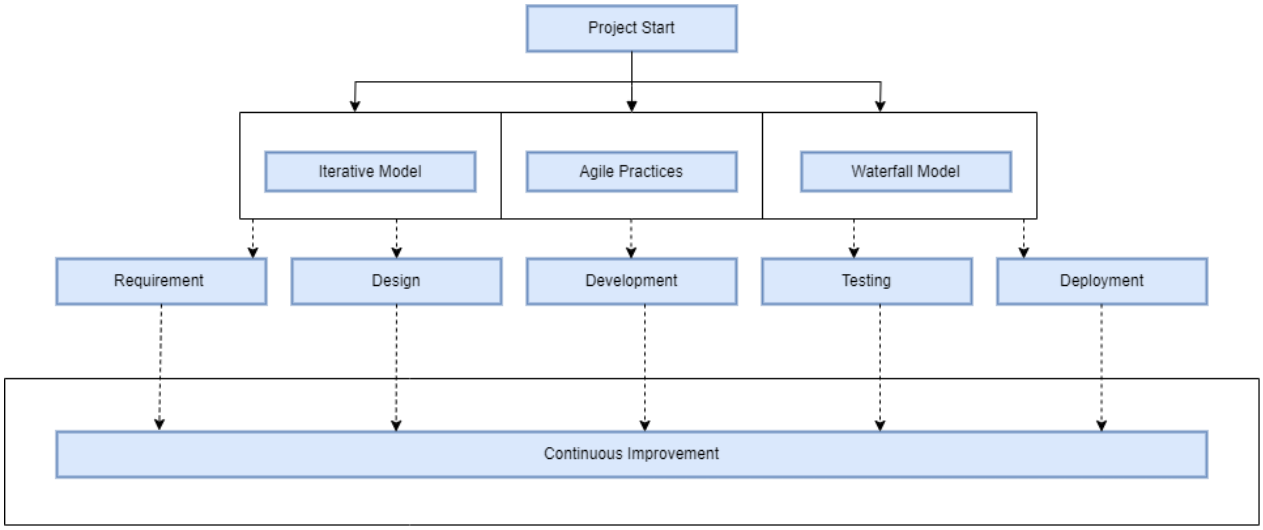


Fig: A visual representation of a hybrid model for the dynamic dashboard project involves illustrating the integration of different methodologies.

2.2 Feasibility Study

Project management involves planning, organizing, and overseeing resources to achieve specific goals and objectives, ensuring the successful completion of a project.

A feasibility study is an initial assessment of a proposed project's merits and viability. It explores technical, economic, financial, legal, and environmental aspects to provide decision-makers with an independent evaluation, helping them decide whether to proceed with the project.

Conducting a feasibility study for my dynamic dashboard project is a crucial step in ensuring its success. Here's what I can achieve through this study:

2.2.1 Operational Feasibility

Objective	Outcome
Assess how well the project fits into the existing operations of the organization.	Ensure that the dynamic dashboard integrates smoothly with current workflows, promoting user acceptance and enhancing operational efficiency as well as it should support in any operating system environment such as windows and linux.

2.2.2 Technical Feasibility

Objective	Outcome
Evaluate the technical aspects, including technology choices, required skills, and system compatibility.	Confirm that using ASP.net Core and C# is technically feasible and that the development team possesses or can acquire the necessary technical skills. In this project we will use ASP.net Core and C#.

2.2.3 Economic Feasibility

Objective	Outcome
Analyze the project's costs against expected benefits to determine its financial viability.	Understand the economic impact of the project, considering development costs, potential returns, and overall return on investment.

2.2.4 Legal Feasibility

Objective	Outcome
Examine legal considerations to ensure compliance with laws and regulations.	I have identified and addressed legal requirements related to data privacy, licensing, and any other relevant legal aspects. This dashboard will be accessed by legal users of any organizations who will have a valid user list.

2.2.5 Scheduling Feasibility

Objective	Outcome
Assess whether the project can be completed within a reasonable timeframe.	I successfully developed the project within three months, working on it alone. By efficiently managing tasks like requirement analysis, gathering, design, development, and testing, I ensured that each step took the least amount of time possible. This demonstrated the feasibility of completing the project within the planned schedule, taking into account dependencies, potential risks, and overall project timelines.

2.3 Time Estimation

Objective	Outcome
Estimate the time required for development, testing, and implementation.	I figured out how much time I needed to create, test, and launch the project. After careful planning, I came up with a detailed schedule that covers all the steps in building the project. This includes developing different parts, testing them out, and finally, how long the entire project would take to finish. This way, I had a clear roadmap to follow and could ensure the project stayed on track.

2.3.1 Time Estimation Calculation

SL	Activity	Preceding Activity	Hours Count
A	Requirement Analysis	None	80
B	Data Collection and analysis	A	70
C	Diagrams Design	A, B	96
D	Database Design	A, C	50
E	Front-end Design	A, C	345
F	Back-end Programming	A, C, D	300
G	Testing	F	60
		Total hours	**1001 hrs**
		Equivalent to days	**41 days (avg.)**

2.4 Cost Estimation

Objective	Outcome
Estimate both development and operational costs associated with the project.	I have identified the financial resources needed for various tasks such as analysis, collection, design, development and testing in budget planning.

2.4.1 Cost Estimation Calculation

Activity	Hours Count	Cost Per Hour (in BDT)	Fixed Price Cost (in BDT)	Estimation Cost (in BDT)
Requirement Analysis	80	20		1600.00
Data Collection and analysis	70	10		700.00
Diagrams Design	96	10		960.00
Database Design	50	15		750.00
Front-end Design	345	35		12075.00
Back-end Programming	300	40		12000.00
Testing	60	30		1800.00
Hardware-Laptop			70000	70000.00
Windows Install			12500	12500.00
Others			5000	5000.00
Total hours	**1001 hrs**		**Total Cost Estimation**	**117385/- Taka**

Conducting a feasibility study ensures that my dynamic dashboard project is well-planned, viable, and aligns with the goals and resources of any organization. It serves as a foundation for successful project execution and minimizes risks associated with uncertainties.

Chapter 3

Software Requirement Specification

3.1 Functional Requirements

3.1.1 User Authentication

- Users should be able to create a new profile by registration.
- Users should be able to log in and log out securely.

3.1.2 Dashboard

- Users should be able to view all charts in a single page.
- All charts should dynamically update in real-time whenever there are changes in the data or new information is introduced.

3.1.3 Full Chart View

- Display real-time or near-real-time data using charts etc.
- Support for various visualization types (bar charts, line graphs, pie charts, etc.).
- Provide the capability to halt real-time dynamic updates for charts when necessary.

3.1.4 Data Filtering and Sorting

- Allow users to filter and sort data within the dashboard.
- Implement date range filters, category filters, etc.

3.1.5 Data Sources Integration

- Connect to different data sources by using REST API.
- Provide the ability to import and integrate external data.

3.1.6 Responsive Design

- Ensure the dashboard is accessible and usable on various devices (desktop, tablet, mobile).

3.1.7 Export and Sharing

- Allow users to export charts (e.g., as JPEG, PNG).
- Allow users to print charts.

3.2 Non-Functional Requirements

3.2.1 Performance

- Ensure fast loading times for the dashboard.
- Optimize data retrieval and rendering processes.

- Supports data loading 24/7, ensuring consistent and reliable performance.

3.2.2 Scalability

- Design the system to handle an increasing number of users and data points.
- Consider scalability in terms of both hardware and software.

3.2.3 Security

- Implement secure authentication mechanisms for registration and login.
- Protect against common security threats like SQL injection, cross-site scripting (XSS), etc.

3.2.4 Reliability

- Minimize downtime by implementing error handling and recovery mechanisms.
- Regularly backup and ensure data integrity.

3.2.5 Usability

- Design an intuitive and user-friendly interface.
- Provide documentation or tooltips for features.

3.2.6 Compatibility

- Ensure compatibility with different web browsers.
- If applicable, consider compatibility with different operating systems.

3.2.7 Maintainability

- Design the system with modularity for easier maintenance and updates.
- Document code and configurations for future reference.

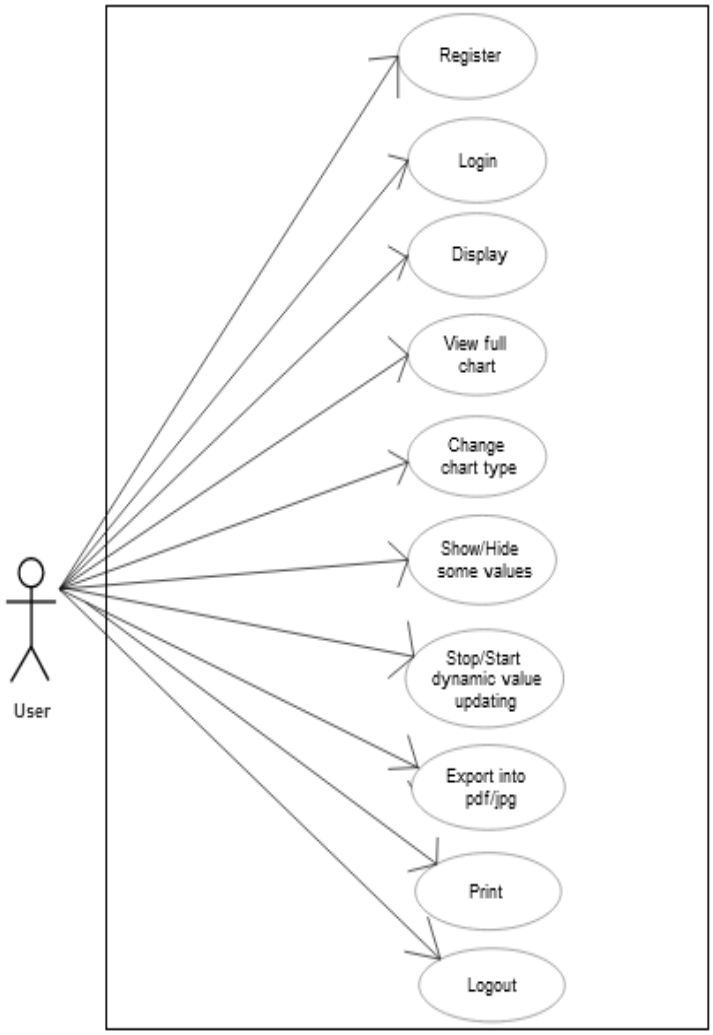
Chapter 4

Use Case Diagram

4.1 Actors

User: Represents the end user interacting with the dynamic dashboard.

4.2 Diagram



4.3 Description

4.3.1 Register

- Description: Allows the user to create a new profile by registration.
- Actors: User.
- Preconditions: The user must be on the register page.

- Flow of Events:
    - User opens the registration page.
    - User gives all the information.
    - All information should be valid.
    - User submits the button.

4.3.2 Login

- Description: Allows the user to login to the dashboard by using credentials.
- Actors: User.
- Preconditions: The user must be registered.
- Flow of Events:
    - User gives his user id and password.

4.3.3 Display

- Description: Allows the user to access the main dashboard.
- Actors: User.
- Preconditions: User is authenticated.
- Flow of Events:
    - User logins into the dynamic dashboard.
    - User opens the dashboard.
    - System retrieves and displays the default dashboard layout.
    - Various charts are rendered on the dashboard.

4.3.4 View Full Chart

- Description: Allows the user to open the full view by clicking any charts from the main dashboard..
- Actors: User.
- Preconditions: User is authenticated and must be on View Full Chart page.
- Flow of Events:
    - User opens the full chart view page.
    - System retrieves and displays the default dashboard layout.
    - Various chart options are available.

4.3.5 Customize Chart

- Description: Enables users to personalize the charts.
- Actors: User.
- Preconditions: User is authenticated and must be on View Full Chart page after clicking any charts from the main dashboard.
- Flow of Events:
    - User accesses the customization settings.
    - System allows various types of charts.
    - System allows show data of different years (parameter value change).
    - System allows to stop and start the dynamic motion when real-time data is updated.
    - User saves the customized layout.

4.3.6 Filter and Short Data

- Description: Permits users to filter data displayed on the dashboard.
- Actors: User.
- Preconditions: User is authenticated and must be on View Full Chart page after clicking any charts from the main dashboard.
- Flow of Events:
    - User interacts with filtering options.
    - System updates the displayed charts based on the selected filters.

4.3.7 Export Chart

- Description: Allows users to export charts from the dashboard.
- Actors: User.
- Preconditions: User is authenticated.
- Flow of Events:
    - User selects data or specific graphs.
    - System exports the selected data or graphs in a chosen format (e.g., JPEG, PNG, Print).

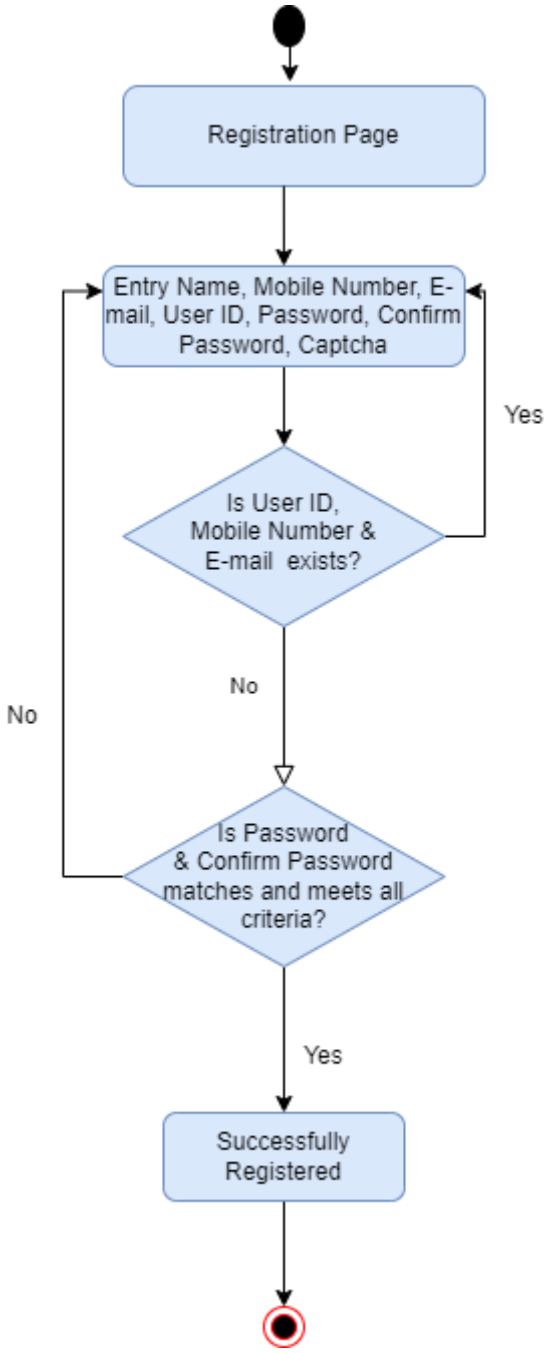
4.3.8 Logout

- Description: Allows the user to log out to the system by clicking on the Logout button on the menu bar.
- Actors: User.
- Preconditions: The user must be logged on.
- Flow of Events:
    - Users will click on the Logout button.
    - Users will be able to log out of the system.

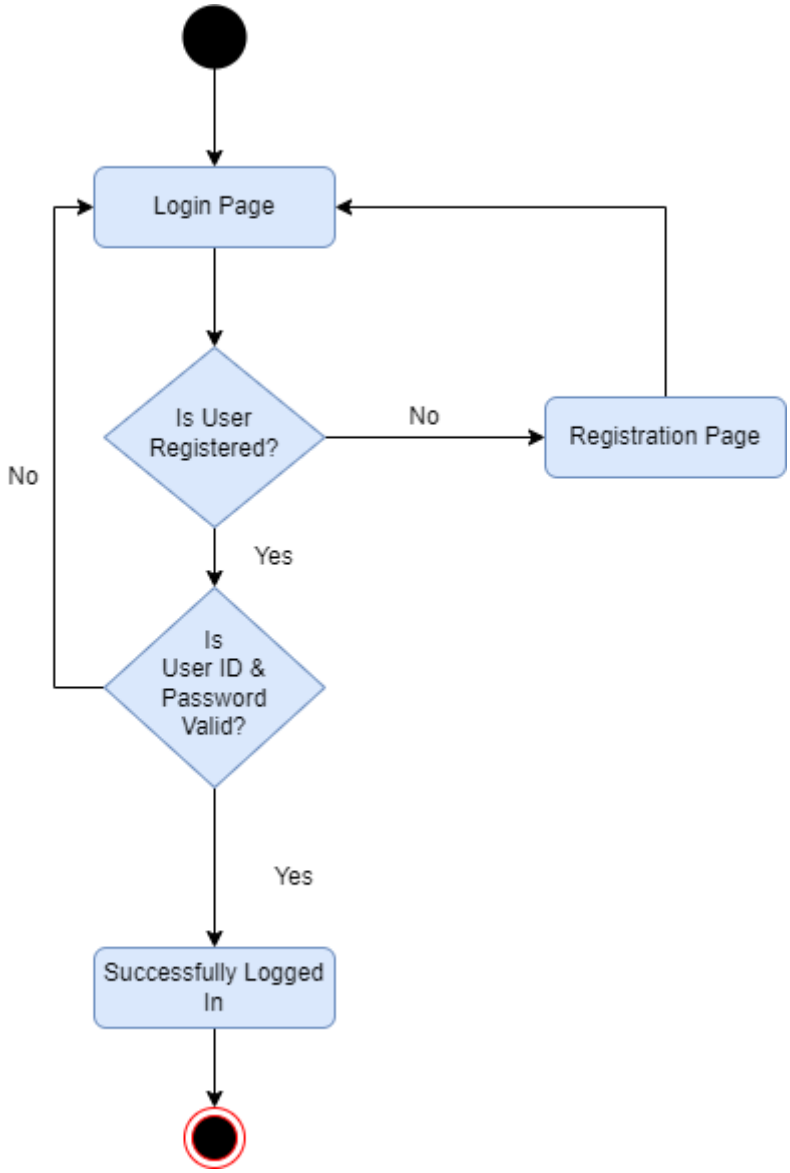
Chapter 5

Activity Diagram

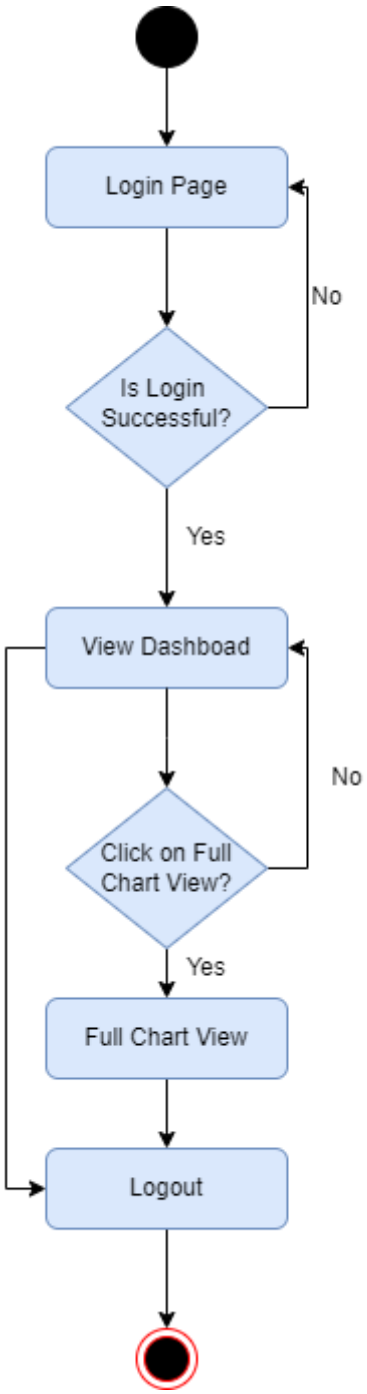
4.1 Registration



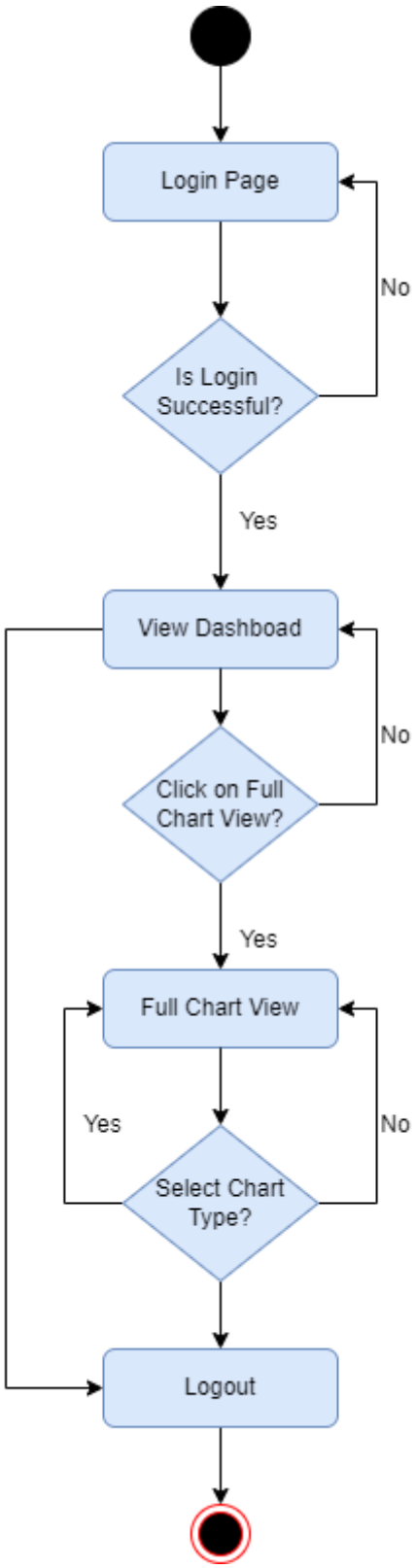
4.2 Login


Login Page
Is User Registered?
No
Registration Page
No
Yes
Is User ID & Password Valid?
Yes
Successfully Logged In

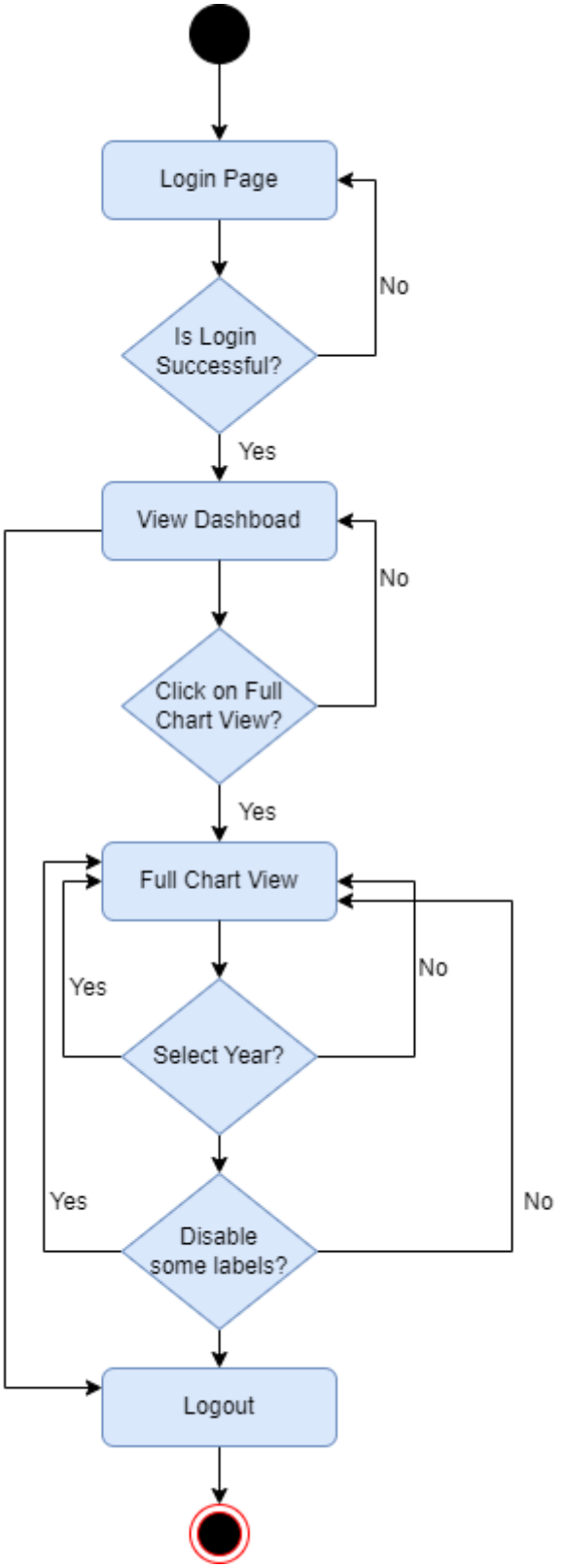
4.3 View Dashboard



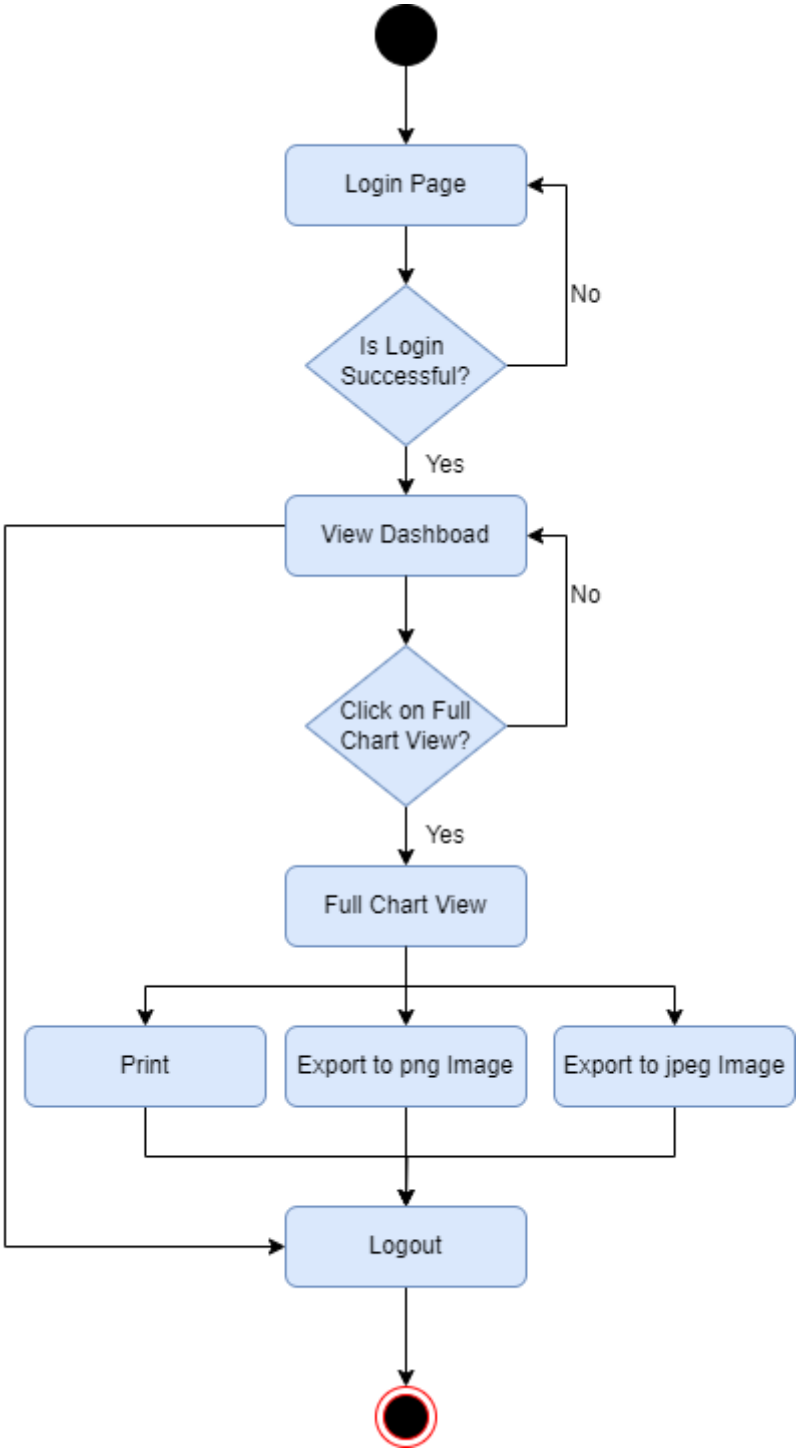
4.4 Customize Charts



4.5 Filter and Sort Data

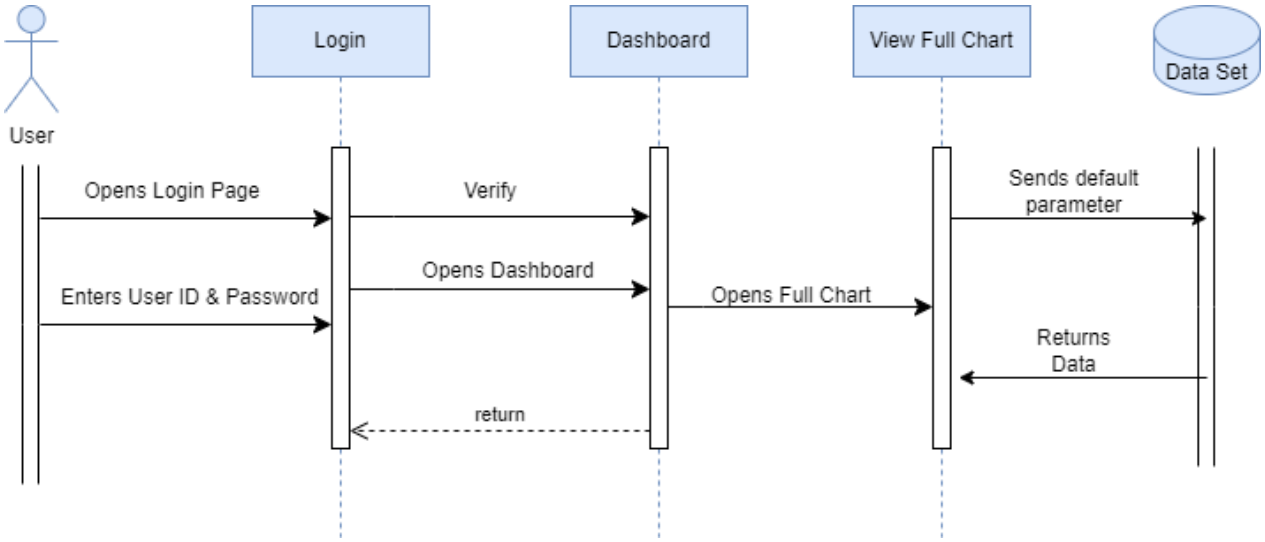


4.6 Export Chart



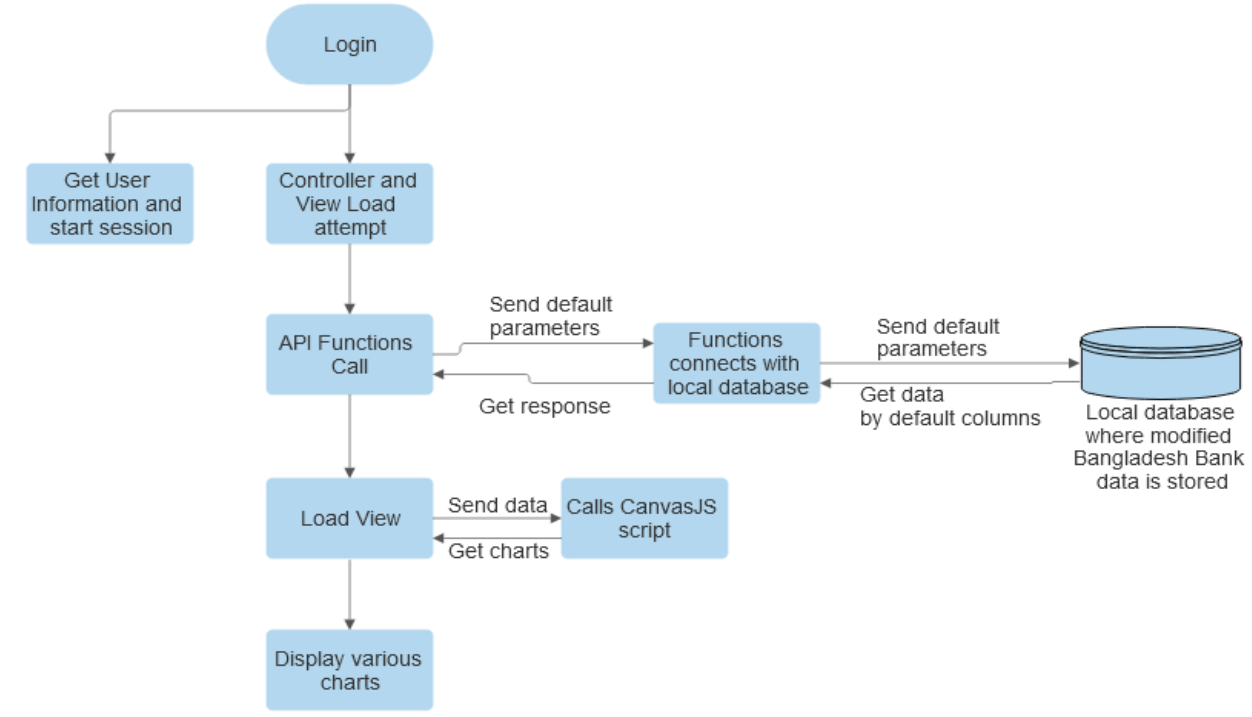
Chapter 6

Sequence Diagram



Chapter 7

Data Flow Diagram



Chapter 8

User Manual

8.1 Registration Page

- I am opening the registration page in order to create my profile.
- I am giving my full name, valid mobile number, valid Email address, User ID which is unique and password to register.

Fig-1: Registration Page

8.2 Login Page

- After successful registration, I am opening the login page.
- I am giving my registered user id and password.

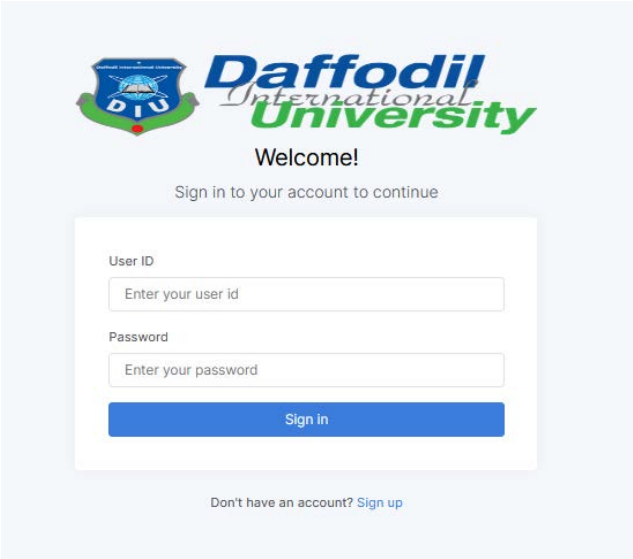


Fig-2: Login Page

8.3 Opening Dashboard

- After successful login, it will redirect to the dashboard page.
- I can view all types of charts in a single display.
- I can see that data is updating whenever any information has come, removed or updated.

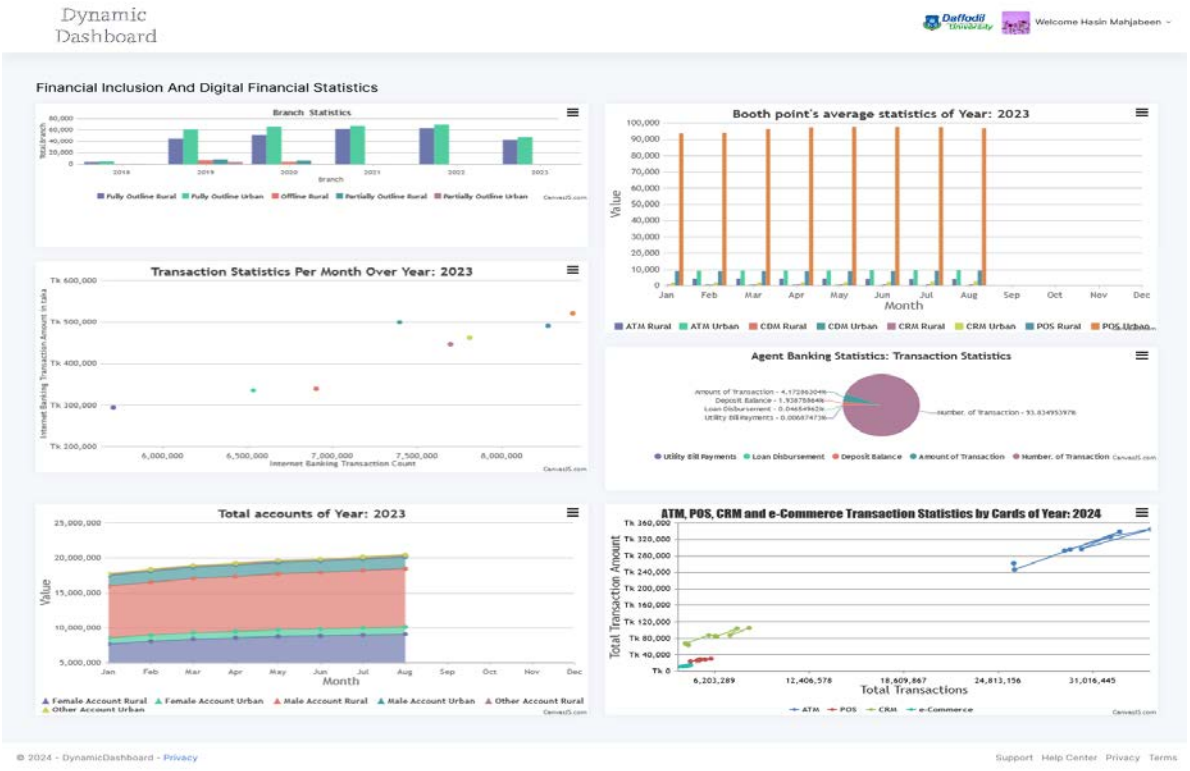


Fig-3: Dashboard Home Page

8.4 Opening Full View of Chart

- If I want to see a full view of any chart, I click on that chart on the home page and it will redirect me to the full view page.
- I can see various options to modify that chart.

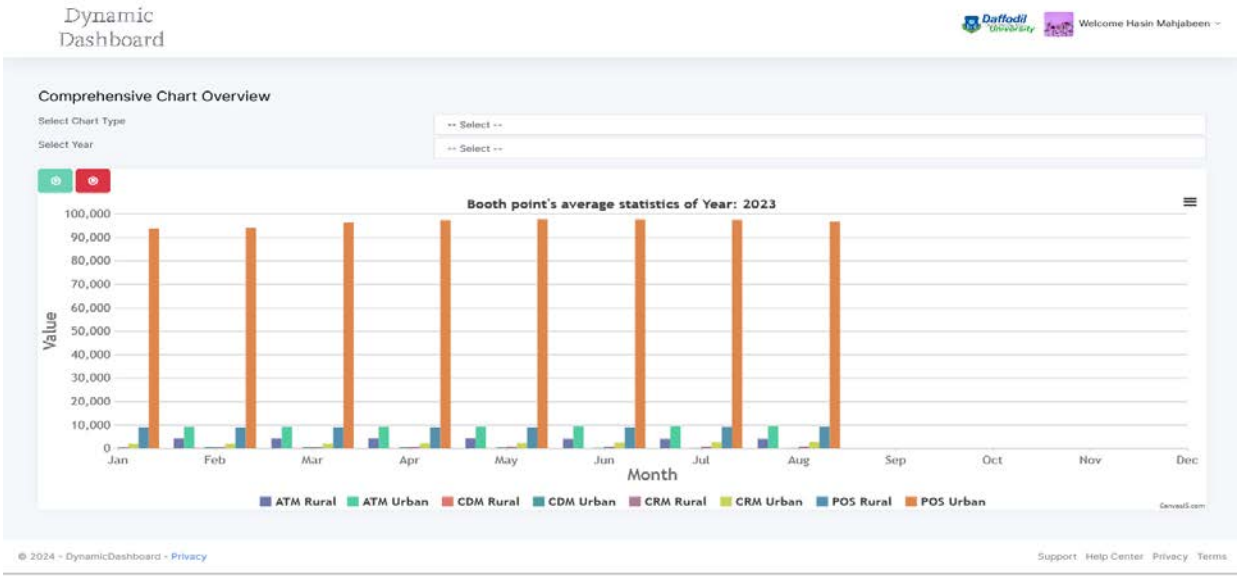


Fig-4: Full Chart View [For example, clicking on 2nd chart from Dashboard (Fig-3) will lead to this overview]

8.5 Change Chart Type

- If I want to change the chart type to check another chart to understand data better, I can select the type.
- Various types are available such as bar chart, column chart, pie chart, line chart, scatter chart, stacked area chart.
- After selecting the type, the chart is converted into a new chart with the same data.

Fig-5: Example -1 : Select Chart Type and View Full Chart (StackedArea Chart)

Fig-6: Example -2 : Select Another Chart Type and View Full Chart (Bar Chart)

8.6 Filter and Sort Data

- If I want to filter some data, I can select Year (Fig-7).
- If I want to skip some labels, I can click on labels, data of clicked labels will be hidden (Fig-8).

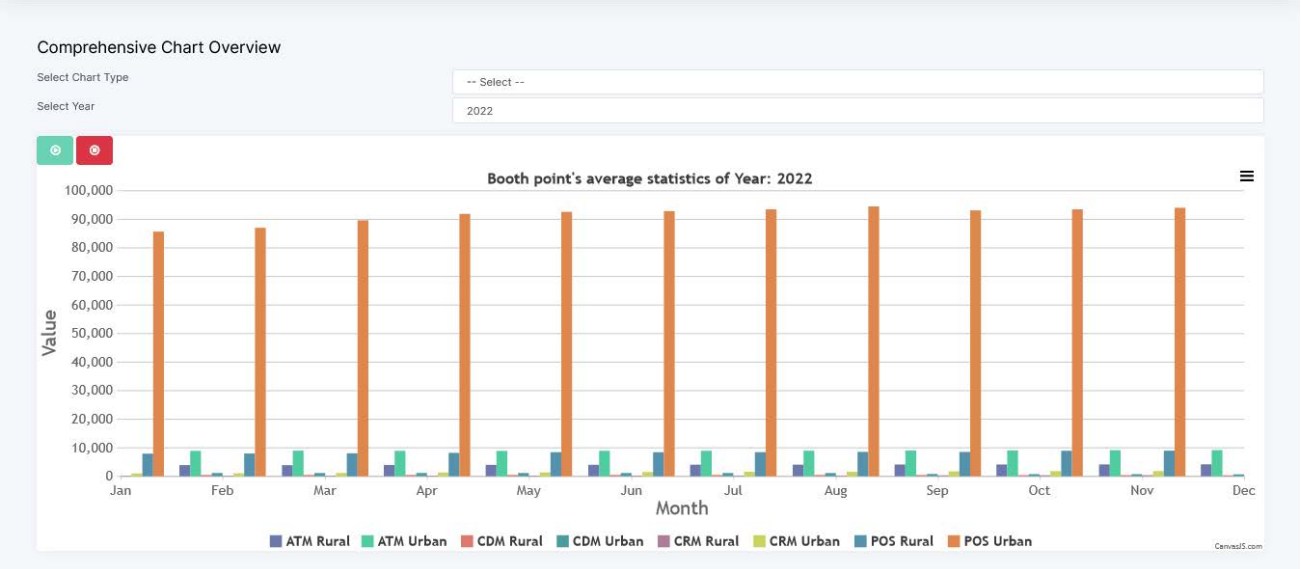


Fig-7: Example -1 : Select Year (For example: 2022) and view data of 2022.

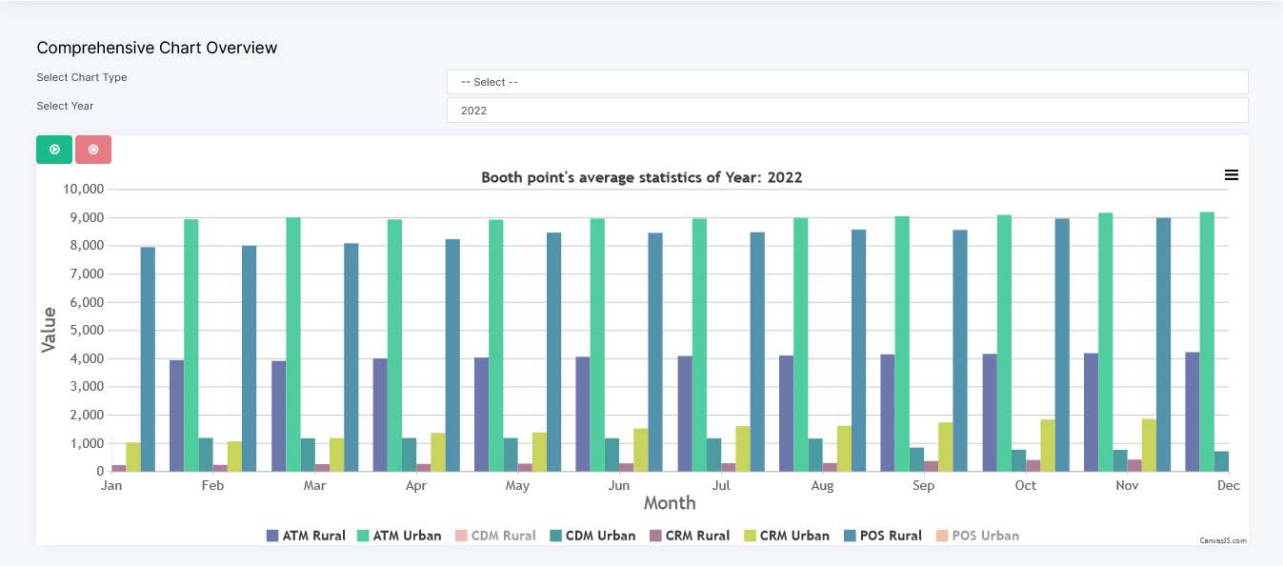


Fig-8: Example -2 : Click on some labels, for example, CDM Rural and POS Urban. We are now viewing data of 2022 but without those labels.

8.7 Stop Dynamic Display and Export Chart

- If I want to export a specific chart, I can stop the dynamic display by clicking the red stop icon.
- I click on the hamburger icon to select the export type option to get an image or printed version.

Fig-9: 1. To Stop the dynamic motion of displaying data, click on the Green/Red button on the left side. 2. To export the selected chart, click on the hamburger icon on the right side and select print or image format (png or jpeg).

Chapter 9

Project Summary

9.1 Project Link

https://drive.google.com/drive/folders/1e5J4DPzKohFDVh_LFiqko8_0koCtFBTW?usp=sharing

9.2 Technologies Used in Project

9.2.1 Backend- ASP.NET Core

ASP.NET Core is a cross-platform, high-performance framework for building modern, cloud-based, and internet-connected applications. I have used this as this allows development in the windows platform as well as this has high performance, making it suitable for handling the dynamic and real-time aspects of a dashboard application.

9.2.2 Data Management- SQL Server

SQL Server is a relational database management system developed by Microsoft. It provides a robust and scalable platform for managing and storing data. As it seamlessly integrates with ASP.NET Core, streamlining database connectivity and operations, I have utilized it to establish a connection between the database and REST API. This integration allows my application to scale and efficiently manage growing data loads over time.

9.2.3 REST API for Global Accessibility

REST API is an architectural style for designing networked applications. I used RESTful APIs to enable global accessibility and data exchange as this is very simple to design and implement, making them easy to scale and maintain.

9.2.4 CanvasJS for Visualization

CanvasJS is a JavaScript library for creating interactive and dynamic charts and graphs. It is often used for data visualization in web applications. I have used CanvasJS because it provides a variety of chart types and customization options, allowing for rich and visually appealing data representation. It also allows real-time data updates as well as it is very easy to integrate into web applications.

9.3 Integration

9.3.1 Data Source

- The project began by selecting a sample dataset from the Bangladesh Bank website.
- I have modified the dataset and restructured and refined certain columns to suit the project's requirements.

9.3.2 Database Setup

- The modified dataset was uploaded to SQL Server 2012 with a database named “Dashboard”.
- I included some tables and stored the data in a structured manner in order to fit into my project.
- I also created some stored procedures to facilitate efficient retrieval of data for API calls.
- I have ensured that all my stored procedures return a list with columns x, y and labels only which supports visualization of charts.

9.3.3 IDE Selection

- I have used Visual Studio 2022 which is the best IDE to build rich, beautiful, cross platform applications for Windows, Mac, Linux, iOS, and Android.

9.3.4 Project Template Selection

- I started by choosing the ASP.NET Core project template that best suited my requirements. Visual Studio 2022 provides a variety of templates, and I carefully selected the one that aligned with the goals of my dynamic dashboard project.

9.3.5 Project Configuration

- Once the template was selected, I configured the project settings, including the project name, location, and solution structure. This step laid the foundation for organizing my codebase efficiently.

9.3.6 Dependencies and Packages

- I navigated through the NuGet Package Manager to manage project dependencies. This involved adding essential packages for functionalities like data management, REST API integration, and visualization.

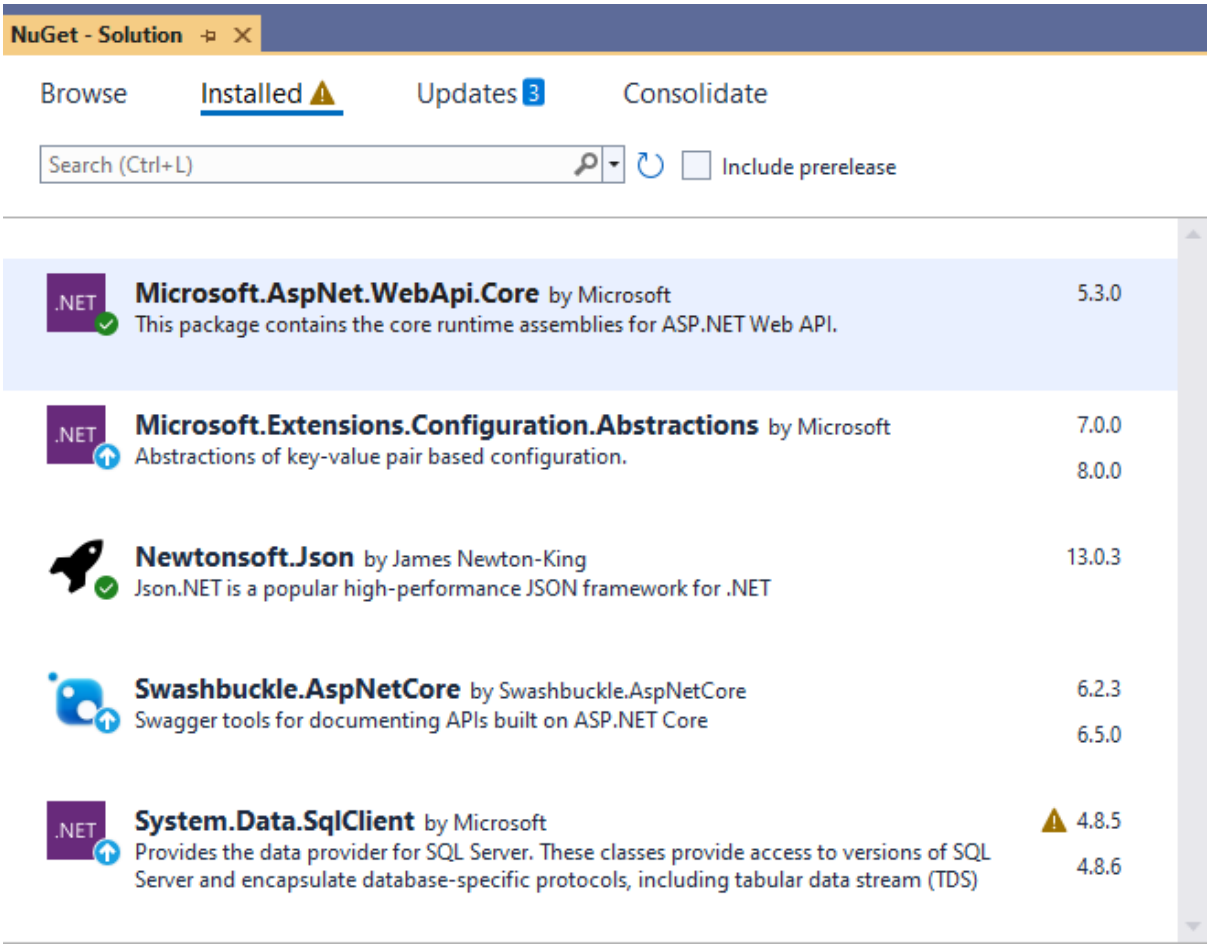


Fig: Installed package list for my project- Dynamic Dashboard

9.3.7 Implementation of Business Logic

- I have implemented the core business logic within the main solution, ensuring seamless communication between the data models and the views. This step involved coding the backend functionalities that would drive the dynamic dashboard.
- Business layers are:
    - MVC (Views and controller)
    - Model (data models)
    - Biz (Business Logic)
    - DAL (Data access layer)
    - REST API Project (Can be used in external, but I have used under the same solution)

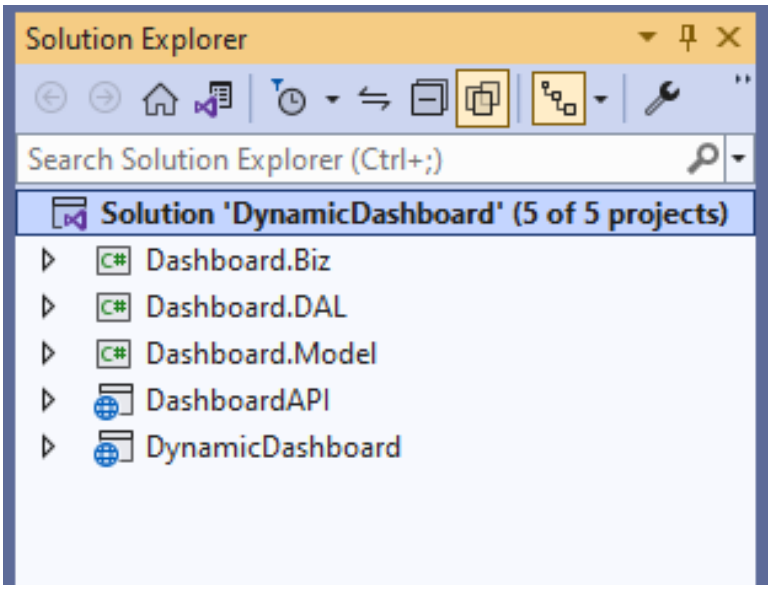


Fig: Business Logic Implementation with various projects

9.3.8 REST API Integration

- I have implemented some sample API methods to generate lists from the stored procedures within the SQL Server database.
- The API was designed to handle specific parameters (for example, x-axis value, y-axis value, label etc.) from external calls, ensuring flexibility and usability and enabling communication between the database and the ASP.NET Core backend.

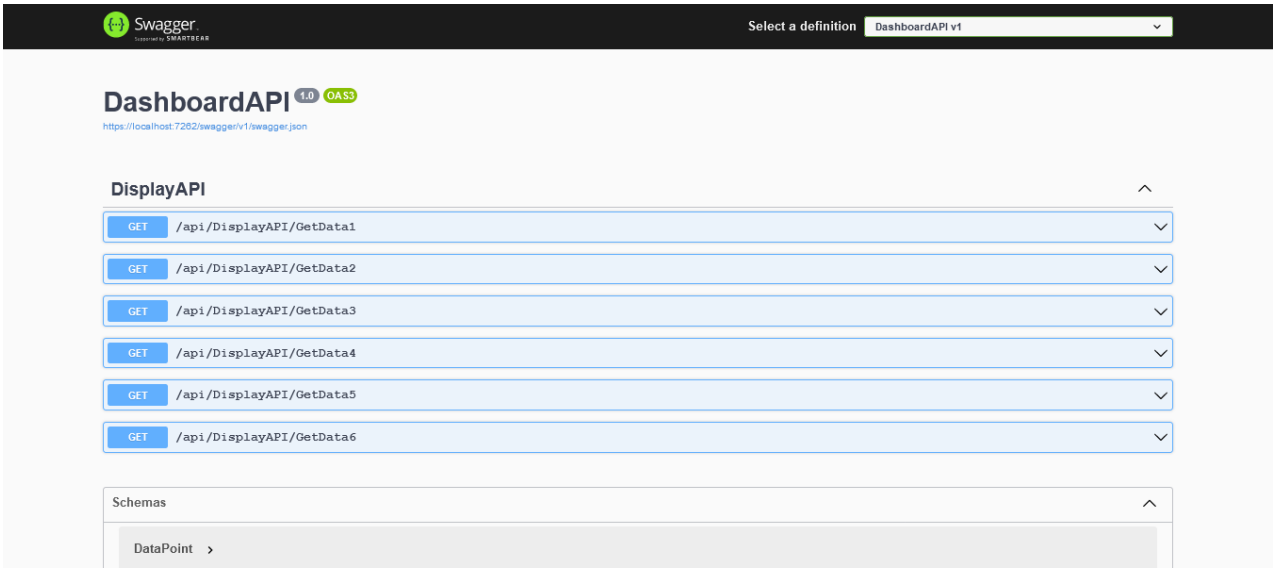


Fig: DashboadAPI view in Swagger

9.3.9 ASP.NET Core Integration

- I have used the Model-View-Controller (MVC) architectural pattern in the ASP.NET Core platform.
- Controller methods were used to handle incoming requests, interact with the REST API, and process data for the frontend visualization.

- Authentication and authorization mechanisms were implemented within ASP.NET Core to ensure secure data access, guaranteeing that only authorized users could interact with the dynamic dashboard.

9.3.10 CanvasJS Integration

- I have fetched data from controller to view of my ASP.NET Core project.
- I have implemented the CanvasJS script to render various types of charts using this data.
- Real-time updates were facilitated through CanvasJS, ensuring the dynamic nature of the dashboard.

9.4 API Implementation Guide

To guide organizations on implementing my project and configuring it with their solutions, I am providing a sample format and steps to follow.

Below is a simple example template that organizations can use as a reference:

9.4.1 Prepare Stored Procedure

Organizations need to create a stored procedure that adheres to the standard columns (x, y, labels). This stored procedure will be called by my API.

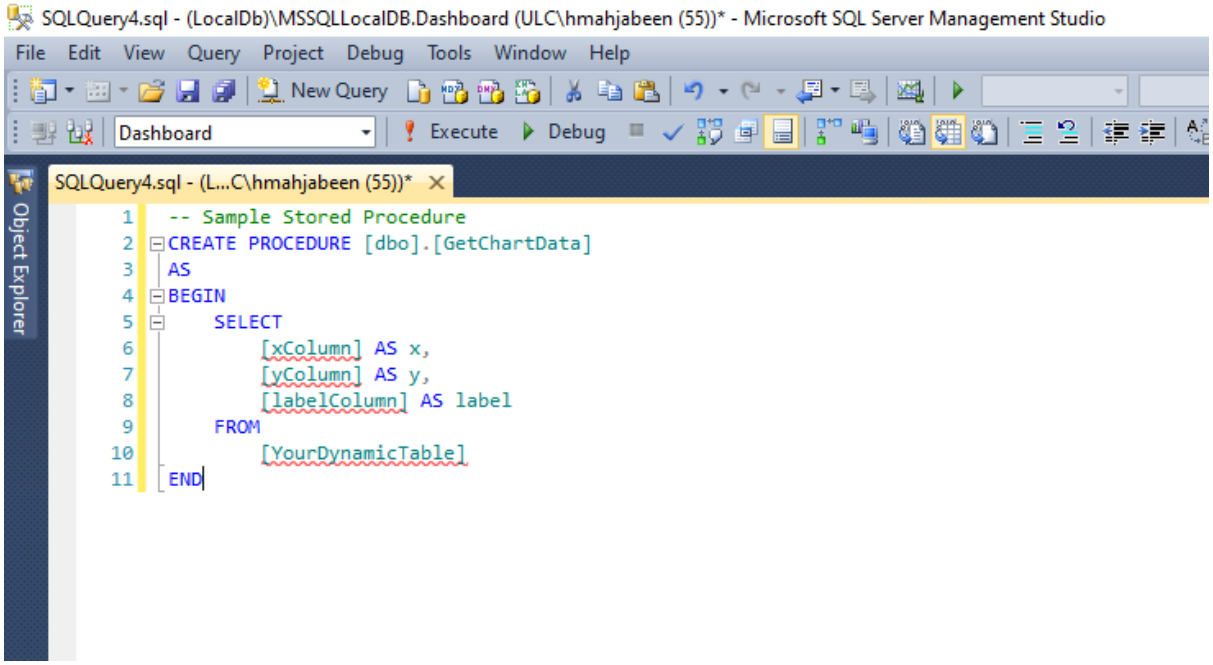


Fig: Sample Stored Procedure Format

9.4.2 Configure Connection String

```
"ConnectionStrings": {
  "DBConnection": "Server=(localdb)\\mssqllocaldb;Database=Dashboard;Trusted_Connection=True;",
```

Fig: Update connection string of organization where data is stored in API solution.

9.4.3 Implement API Controller

Organizations need to create an API controller that adheres to the stored procedures and return the data in a standardized format (x, y, labels).

9.4.4 Configure Authentication

Organizations need to configure authentication to interact with your API securely. Provide information about basic authentication credentials provided by my end.

By following these steps, organizations can integrate my dynamic dashboard API into their solutions securely with proper authentication and adherence to the standardized data format.

9.5 Achievements and Limitations

9.5.1 Achievements

- I have achieved successful implementation of charts in the dashboard using CanvasJS scripts, enhancing data visualization with dynamic and visually appealing charts.
- I have successfully ensured a seamless data flow from the source to the frontend, enhancing the overall efficiency and reliability of the system.
- Real-time data updates have been successfully implemented in the dashboard. I have set the flexibility to halt real-time updates that allows users to focus on specific data when needed.
- I have successfully used global accessibility through REST APIs, enabling any organization or users to interact with and retrieve data from the dashboard.

9.5.2 Limitations

- The dashboard may face limitations in accommodating all types of datasets, as certain datasets may not be suitable for the chosen visualization methods.
- Some charts may not be easily convertible to other types, leading to potential challenges when attempting to switch from one chart type to another (e.g., converting a pie chart to a bar chart).
- The dashboard's loading speed may be impacted when dealing with extensive datasets, potentially resulting in slower performance.
- The project has a restricted set of user roles, potentially limiting the differentiation of permissions and access levels among users.
- The user can not update their profile now, can’t change password or try “forget password” at this moment.

- The dashboard may not be fully optimized for mobile devices, potentially leading to suboptimal user experiences on smaller screens.

9.6 Future Plan

1. Integrate more chart types in the dashboard as well as include some grids if needed.
2. Add options to convert chart data to tables to analysis data properly.
3. Add more options to customize charts and data.
4. All browsers and mobile devices would be responsive.
5. User roles and permissions can be applied in order to limit viewing sensitive charts.
6. Analysis other tools except canvasJS scripts.

Conclusion

In summary, this documentation captures the essence of my dynamic dashboard project. It unfolds the intricacies of my journey from data fetching to visualization, employing iterative and Agile methodologies. The selection of technologies like ASP.NET Core, SQL Server, CanvasJS, and REST APIs underscores a thoughtful approach tailored to project needs.

The dynamic dashboard is a tool where organizations interact with data. It ensures real-time data presentation, global accessibility through REST APIs, and the ability to stop updates for focused analysis. While acknowledging limitations, such as data suitability and potential slow loading, the project opens avenues for future enhancements.

Looking ahead, the roadmap includes strengthening user authentication, enhancing dashboard customization, and integrating with diverse data sources. In conclusion, the dynamic dashboard is a solution poised to empower organizations with smarter data utilization, reflecting a journey of learning, adaptability, and impactful development.

References

1. Data Source: Bangladesh Bank Open Data Initiative
   https://www.bb.org.bd/en/index.php/econdata/index
   https://www.bb.org.bd/econdata/fin_digitalfstat/fin_digitalfstat.xlsx

2. CanvasJS:
   https://canvasjs.com/

3. ASP.NET Core
   https://dotnet.microsoft.com/en-us/apps/aspnet

4. Visual Studio 2022
   https://visualstudio.microsoft.com/vs/

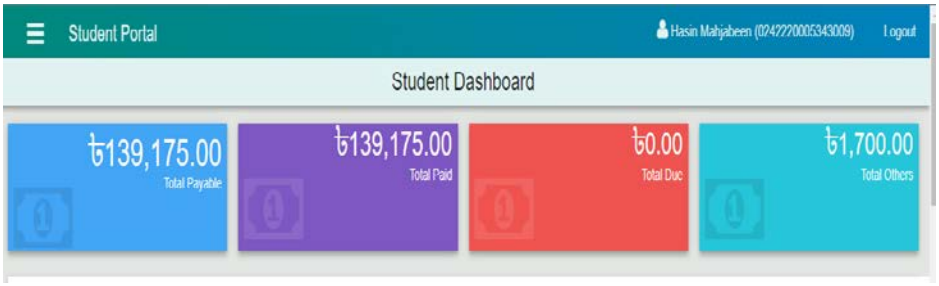
5. Microsoft SQL Server 2012
   https://www.microsoft.com/en-us/sql-server/sql-server-downloads

6. Diagram Draw:
   https://www.drawio.com/

7. Project Cost Calculation Idea from:
   https://www.zoho.com/books/free-project-estimate-calculator/

8. Project Time Calculation Idea from:
   https://bambooagile.eu/estimate/calculator/steps/question/1?reset=true

Account Clearance



Plagiarism Report

0242220005343009

ORIGINALITY REPORT

15%	14%	2%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.daffodilvarsity.edu.bd:8080 Internet Source	9%
2	www.vskills.in Internet Source	1%
3	Submitted to University of South Australia Student Paper	1%
4	Submitted to University of Greenwich Student Paper	1%
5	Submitted to Middlesex University Student Paper	<1%
6	Submitted to Hoa Sen University Student Paper	<1%
7	123dok.com Internet Source	<1%
8	visualstudio.microsoft.com Internet Source	<1%
9	dspace.bracu.ac.bd:8080 Internet Source	<1%