**GRAPH NOTIFY-SENDING MESSAGES THROUGH GRAPH API**

**ABISHEK K**

**Register No: 1933002**

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

M.Sc. (Decision and Computing Sciences)

OF ANNA UNIVERSITY



**APRIL 2024**

**DEPARTMENT OF COMPUTING**

**COIMBATORE INSTITUTE OF TECHNOLOGY**

**(Autonomous Institution affiliated to Anna University)**

**COIMBATORE – 641014**

**COIMBATORE INSTITUTE OF TECHNOLOGY**

**(Autonomous Institution affiliated to Anna University)**

**COIMBATORE 641014**

(Bonafide Certificate)

Project Work - II

Tenth Semester

**GRAPH NOTIFY-SENDING MESSAGES THROUGH GRAPH API**

Bonafide record of work done by

**ABISHEK K**

**(Register No: 1933002)**

Submitted in partial fulfilment of the requirements for the degree of

M.Sc. (Decision and Computing Sciences) of Anna University

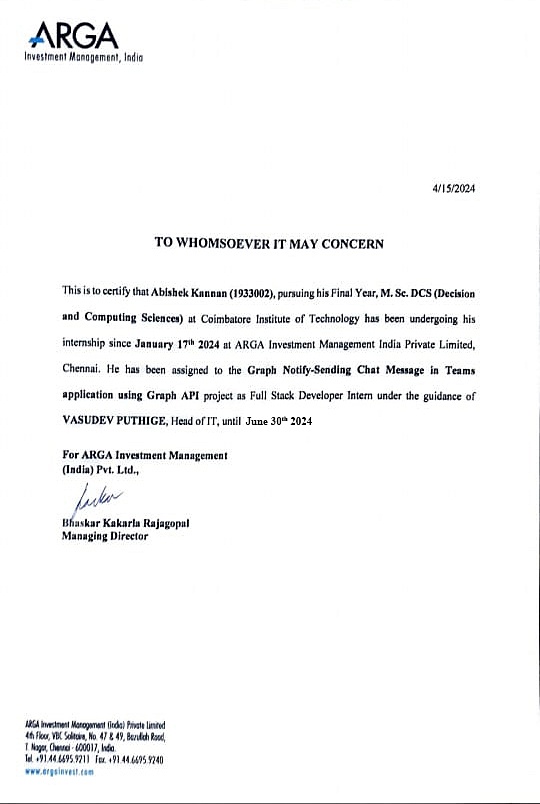
December 2022

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Faculty Guide Head of the Department

Submitted for the viva-voce held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Internal Examiner External Examiner

**CERTIFICATE:**

**CONTENTS**

**CHAPTER** **PAGE NO**

ACKNOWLEDGEMENT i

SYNOPSIS ii

PREFACE iii

**I INTRODUCTION**

1.1 ORGANIZATION PROFILE

1.2 PROBLEM STATEMENT

1.3 SYSTEM ENVIRONMENT

**II SYSTEM ANALYSIS**

2.1 SYSTEM DESCRIPTION

2.2 USE CASE MODEL

2.3 AZURE AD AND AUTHENTICATION

2.4 SOFTWARE REQUIREMENTS SPECIFICATION

2.5 TEST PLAN

**III SYSTEM DESIGN**

3.1 ARCHITECTURAL DESIGN

3.2 STRUCTURAL DESIGN

3.3 BEHAVIOURAL DESIGN

3.4 TABLE DESIGN

3.5 USER INTERFACE DESIGN

3.6 DEPLOYMENT DESIGN

3.7 NAVIGATION DESIGN

3.8 CODE DESIGN

**IV SYSTEM TESTING**

4.1 TEST CASES AND TEST REPORTS

**V SYSTEM IMPLEMENTATION**

**VI CONCLUSION**

BIBLIOGRAPHY

# ACKNOWLEDGEMENT

The opportunity to work on this project was a great chance and a hands-on experience for learning multiple things. I consider myself privileged to be a part of such a wonderful organization.

I express my thankfulness to **Dr. A. Rajeswari, Principal In-charge**, Coimbatore Institute of Technology, for permitting me to undertake this project work.

I am grateful to **Dr. K. Sakthimala**, **Dean of Computing**, Coimbatore Institute of Technology, Coimbatore for her encouragement throughout this project.

I am grateful to **Dr. A. Kannammal**, **Head, Department of Computing** (Decision and Computing Sciences), Coimbatore Institute of Technology, Coimbatore for her encouragement throughout this project.

I am indebted to my internal guide, **Mrs S. Karpagam**, Assistant Professor, Department of Computing, Coimbatore Institute of Technology, Coimbatore, for her constant support and guidance throughout the project work.

I also express my sincere thanks to my external guide **Mr. Vasudev Puthige,** IT

Lead (Software),Arga Solutions., Chennai for helping me to do this project work.

I wish to acknowledge all the other members in my team, who made work challenging and were very helpful in growing my learning curve.

Finally, I express my gratitude to my family and friends for their encouragement and support.

i.

**SYNOPSIS**

The Graph Notify project represents a pivotal advancement in internal communication strategies, harnessing the capabilities of the Microsoft Teams platform and the Graph API to facilitate seamless and efficient message transmission within organizations.At its essence, Graph Notify offers a sophisticated yet user-friendly interface embedded directly within Microsoft Teams, empowering users to compose, send, and manage messages effortlessly. Leveraging the Graph API's extensive functionalities, the system supports diverse message types, including text, multimedia content, links, and attachments, ensuring versatility in communication.

One of the project's paramount features is its robust authentication mechanism, seamlessly integrated into the system. By leveraging Microsoft's authentication services, such as Azure Active Directory, Graph Notify ensures secure access control, safeguarding sensitive information and fostering a trusted communication environment.

Moreover, the project places a strong emphasis on reliability and performance optimization. Through meticulous error handling strategies, the system adeptly manages any anomalies encountered during message transmission, guaranteeing uninterrupted communication flow. Additionally, performance optimization techniques, such as message queuing and asynchronous processing, bolster the system's efficiency, even under high loads.

In alignment with modern organizational needs, Graph Notify goes beyond mere message transmission. It offers comprehensive analytics and reporting capabilities, enabling organizations to gain insights into message engagement metrics, delivery rates, and user interactions. These insights empower organizations to refine their communication strategies, maximize engagement, and drive organizational alignment.

Overall, the Graph Notify project represents a transformative leap forward in internal communication paradigms. ii.

**PREFACE**

**CHAPTER I – INTRODUCTION** This chapter gives a detailed description of the organizational environment in which the projects were developed. It also gives a brief introduction to the problem statement, objective, and scope of the proposed system.

**CHAPTER II - SYSTEM ANALYSIS** This chapter describes the features of the proposed system. It also describes the functional and non-functional requirements of the system.

**CHAPTER III - SYSTEM DESIGN** It elucidates the design of Graph Notify with related diagrams.

**CHAPTER IV - SYSTEM TESTING** This chapter describes in detail the various performance measures used in the project for testing the accuracy of the model developed.

**CHAPTER V - SYSTEM IMPLEMENTATION** This chapter discusses the pre-installation software and its procedures.

**CHAPTER VI – CONCLUSION** Puts forth the special features and suggestions for future enhancements of this project.

iii.

**CHAPTER I**

**INTRODUCTION**

The Graph API, developed by Microsoft, serves as a unified endpoint for accessing data, intelligence, and services across various Microsoft platforms, including Office 365, Azure Active Directory, and Microsoft Teams.

It provides developers with a powerful toolset to integrate and interact with Microsoft services programmatically, enabling seamless communication, collaboration, and automation within organizations. By offering a standardized and comprehensive interface, the Graph API simplifies the development process and facilitates the creation of innovative applications and solutions that leverage the rich ecosystem of Microsoft services.

## **1.1 ORGANIZATION PROFILE**

Arga Infotech Solutions Private Limited, headquartered in Chennai, India, is a dynamic IT firm renowned for its comprehensive suite of services and innovative solutions. Established with a vision to revolutionize the digital landscape, Arga Infotech Solutions has garnered a reputation for excellence in software development, digital transformation, and consultancy services.

The company boasts a team of highly skilled professionals equipped with diverse expertise in areas such as software engineering, data analytics, cloud computing, and project management. This multidisciplinary approach enables Arga Infotech Solutions to deliver tailored solutions that address the unique challenges and requirements of clients across various industries.

Arga Infotech Solutions' service offerings span a wide spectrum, encompassing custom software development, mobile application development, web development, enterprise solutions, and IT consulting. Leveraging the latest technologies and industry best practices, the company empowers businesses to streamline operations, enhance productivity, and achieve sustainable growth.

One of the key strengths of Arga Infotech Solutions lies in its customer-centric approach. The company places a strong emphasis on understanding the specific needs and objectives of each client, forging long-term partnerships built on trust, transparency, and mutual success. By fostering open communication and collaboration, Arga Infotech Solutions ensures that every project is executed seamlessly, on time, and within budget.

Moreover, Arga Infotech Solutions is committed to staying at the forefront of technological innovation. The company invests in continuous research and development to explore emerging technologies and trends, enabling clients to harness the full potential of digital transformation. Whether it's implementing cutting-edge solutions in artificial intelligence, machine learning, blockchain, or IoT, Arga Infotech Solutions remains at the forefront of innovation, driving value for its clients in an ever-evolving digital landscape.

In essence, Arga Infotech Solutions Private Limited stands as a trusted partner for organizations seeking to leverage technology to achieve their business objectives. With its unwavering commitment to excellence, innovation, and customer satisfaction, the company continues to redefine the boundaries of possibility in the realm of IT solutions and services.

Arga Infotech Solutions operates with a philosophy deeply rooted in client-centricity, innovation, and excellence. Our primary focus is on understanding and meeting the specific needs of our clients, ensuring their satisfaction and success. We continually strive for innovation, pushing boundaries to deliver cutting-edge solutions and services that exceed expectations. At the core of our philosophy lies a commitment to quality and reliability, underpinned by rigorous processes and attention to detail. We foster a collaborative culture, valuing teamwork, communication, and shared knowledge. Integrity and ethics are fundamental to everything we do, guiding our interactions and decisions. Additionally, we prioritize continuous learning and growth, empowering our team members to develop professionally and personally. This philosophy drives our pursuit of excellence, enabling us to create lasting value for our clients and stakeholders.

#### **1.1.1 TEAM PROFILE**

#### ***1. Software Development Service:***

#### Arga's software development team is dedicated to crafting user-centric applications that enhance user experience and deliver comprehensive solutions. They specialize in integrating various components seamlessly to create cohesive and intuitive software products.

#### ***2. Embedded Design Service:***

#### Arga's embedded design team specializes in the development of complex imaging systems, RF technologies, and managing aspects like signal integrity, thermal dynamics, and EMI/EMC considerations. Their focus is on delivering solutions that meet strict timelines and adhere to specified form factors.

#### Arga Infotech Solutions prides itself on its ability to cater to diverse client needs by offering a comprehensive range of services across software development and embedded design domains. With a commitment to quality, timeliness, and client satisfaction, Arga is poised to deliver cutting-edge solutions that meet the evolving demands of the industry.

## ***1.2 PROBLEM STATEMENT:***

The Graph API project aims to tackle the intricate challenge of seamlessly integrating and harnessing the vast array of data, intelligence, and services dispersed across various Microsoft platforms. This entails overcoming barriers related to disparate data silos, complex authentication mechanisms, and diverse service endpoints within Office 365, Azure Active Directory, and Microsoft Teams. The project seeks to streamline access to these resources, enabling developers to create cohesive and interconnected solutions that leverage the full potential of Microsoft's ecosystem for enhanced productivity, collaboration, and innovation.

***1.2.1. PROBLEM OBJECTIVE:***

The Graph API endeavors to provide developers with a unified and simplified interface for accessing data, intelligence, and services across Microsoft platforms, fostering seamless integration and enabling the creation of innovative solutions that enhance productivity, collaboration, and user experiences.

***1.2.2. SCOPE:***

The project encompasses the development and deployment of the Graph API, focusing on providing a comprehensive set of endpoints and functionalities for accessing and interacting with data, intelligence, and services across Microsoft platforms. This includes authentication mechanisms, data querying and manipulation, integration with various Microsoft services, and support for diverse application scenarios, catering to the needs of developers and organizations seeking to leverage the power of Microsoft's ecosystem.

***1.2.3. LIMITATIONS***

The limitations of the Graph API encompass coverage constraints, rate limits, access restrictions based on permissions, authentication complexities with OAuth 2.0, challenges with data consistency, service downtime impacting availability, a learning curve for developers, and dependency on platform updates.

## ***1.3 SYSTEM ENVIRONMENT***

This section describes the system environment used for the development of the project.

***1.3.1 HARDWARE AND SOFTWARE ENVIRONMENT***

#### HARDWARE CONFIGURATION

* Processor : Intel Core i5 CPU 7200U @ 2.50GHz 2.71 GHz
* Hard Disk : 1 TB
* RAM : 8 GB

#### SOFTWARE CONFIGURATION

* Operating System : Windows 10
* Languages : C#, MS SQLDB, Angular JS,SCSS
* Tool used : Microsoft Visual studio Code

\

**CHAPTER II**

# SYSTEM ANALYSIS

System analysis is a process of studying the system in order to identify the goals, purpose of the system and better understanding of the system’s requirements. This chapter gives a brief discussion about the detailed study of the proposed system and the different functionalities involved in the system.

## ***2.1 SYSTEM DESCRIPTION***

Graph Notify revolutionizes internal communication by seamlessly integrating with Microsoft Teams and harnessing the power of the Graph API. This innovative system provides organizations with a comprehensive platform to send messages, notifications, and alerts directly through Teams channels, enabling seamless collaboration and information dissemination.

At its core, Graph Notify offers a user-friendly interface embedded within Microsoft Teams, allowing users to compose, send, and manage messages effortlessly. Leveraging the Graph API's extensive functionalities, the system ensures secure access and efficient delivery of messages, empowering teams to stay connected and informed in real-time.

One of the key strengths of Graph Notify lies in its ability to target messages effectively. Whether it's notifying specific teams about project updates, sending alerts for critical incidents, or broadcasting company-wide announcements, the system offers flexibility and customization to meet diverse communication needs.

Moreover, Graph Notify enhances collaboration by facilitating interactive communication within Teams channels. Users can engage in discussions, share insights, and collaborate on projects seamlessly, driving productivity and innovation across the organization.

With robust authentication mechanisms and stringent security measures, Graph Notify ensures that sensitive information remains protected at all times. By adhering to industry best practices and compliance standards, the system offers peace of mind to organizations concerned about data privacy and security.

In summary, Graph Notify serves as a powerful tool for organizations looking to streamline internal communication, foster collaboration, and drive productivity. By leveraging the capabilities of Microsoft Teams and the Graph API, it empowers teams to communicate effectively, stay informed, and achieve their goals with greater efficiency.

#### ***APPLICATION DOMAIN DESCRIPTION:***

***2.1.1. C#:***

C# (pronounced as "C sharp") is a versatile and powerful programming language developed by Microsoft in the early 2000s. It is designed to be simple, modern, and efficient, with syntax similar to other C-style languages like C, C++, and Java. C# is widely used for developing a wide range of applications, including desktop software, web applications, mobile apps, cloud-based services, and even games.

One of the key features of C# is its strong typing system, which provides compile-time type checking to ensure type safety and prevent common programming errors. This makes C# code more robust and reliable, especially in large-scale projects.

C# is also an object-oriented programming (OOP) language, meaning it supports the concepts of classes, objects, inheritance, and polymorphism. This enables developers to write clean, modular, and reusable code, leading to easier maintenance and scalability of software projects.

Another important aspect of C# is its integration with the .NET framework, which is a comprehensive platform for building and running applications on Windows. The .NET framework provides a rich set of libraries and APIs for tasks such as file I/O, networking, database access, and user interface development, making it easier for developers to create feature-rich applications.

Moreover, C# has evolved over the years with the introduction of new language features and enhancements. For example, C# 8.0 introduced nullable reference types, asynchronous streams, and pattern matching enhancements, further improving the language's expressiveness and productivity.

In addition to its technical capabilities, C# enjoys strong support from the developer community and a rich ecosystem of tools, libraries, and resources. This makes it an attractive choice for both beginners and experienced developers looking to build modern, high-performance applications across various platforms.

Overall, C# continues to be a popular and widely adopted programming language, playing a crucial role in the Microsoft ecosystem and contributing to the development of innovative software solutions across industries.

***2.1.2.Angular JS:***

AngularJS is an influential open-source JavaScript framework developed and maintained by Google, renowned for its ability to streamline the development of dynamic and interactive web applications. With a focus on enhancing HTML syntax, AngularJS empowers developers to build robust single-page applications (SPAs) with ease.

At the core of AngularJS lies its powerful features, including two-way data binding, which enables seamless synchronization between the model and the view, reducing the need for manual DOM manipulation. This feature enhances productivity and simplifies the development process by automatically updating the UI in response to changes in the underlying data.

Furthermore, AngularJS promotes modularity through its modular architecture, allowing developers to break down complex applications into smaller, reusable components. This approach fosters code reusability, maintainability, and scalability, making it easier to manage large-scale projects.

Another key aspect of AngularJS is its dependency injection system, which facilitates the management of dependencies and promotes the writing of testable and maintainable code. By decoupling components and services, AngularJS enhances code organization and promotes best practices in software development.

Additionally, AngularJS offers a rich set of built-in directives, such as ng-repeat, ng-model, and ng-if, which extend HTML with custom attributes and behaviors. These directives enable developers to create dynamic and responsive user interfaces, enhancing the overall user experience.

Moreover, AngularJS provides various built-in services for common tasks, including HTTP requests, routing, form validation, and more. These services simplify the implementation of core functionalities, allowing developers to focus on building innovative features and functionalities.

In summary, AngularJS revolutionizes web development by providing developers with a comprehensive toolkit for building modern web applications. Its features, including two-way data binding, modularity, dependency injection, directives, and services, empower developers to create dynamic, interactive, and scalable applications that meet the demands of today's digital landscape.

***2.1.3. SCSS:***

SCSS, short for "Sassy CSS," is a preprocessor scripting language that is interpreted or compiled into CSS. It adds functionality and features to CSS, making it more dynamic and maintainable. SCSS introduces concepts such as variables, nesting, mixins, inheritance, and functions, which allow developers to write CSS in a more modular and reusable way.

One of the key benefits of SCSS is its ability to use variables, allowing developers to define values once and reuse them throughout the stylesheet. This not only improves code consistency but also makes it easier to update styles across the entire project.

Additionally, SCSS supports nesting, which allows CSS rules to be nested within one another, mirroring the structure of the HTML markup. This makes the code more readable and helps developers better organize their stylesheets.

Mixins and inheritance are another powerful feature of SCSS. Mixins allow developers to define reusable blocks of CSS properties and apply them to multiple selectors, reducing code duplication. Inheritance, on the other hand, enables styles to be inherited from one selector to another, promoting a more modular and DRY (Don't Repeat Yourself) coding approach.

SCSS also supports functions, which can be used to perform calculations, manipulate colors, and apply various transformations to CSS values. This enhances the flexibility and expressiveness of CSS, allowing developers to create more dynamic and interactive stylesheets.

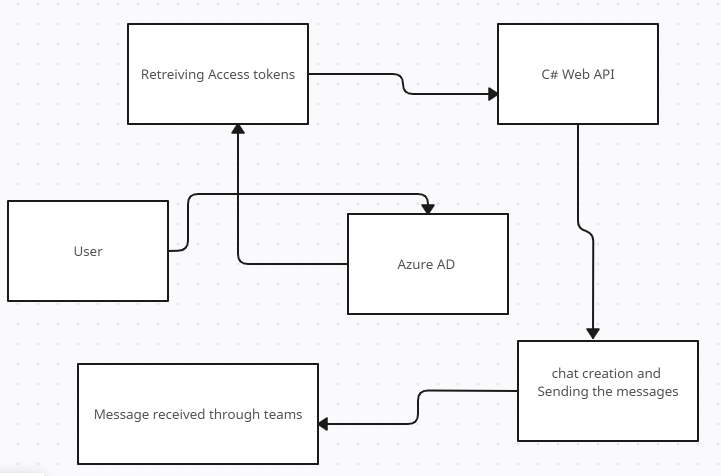
Overall, SCSS is a valuable tool for front-end developers, offering enhanced capabilities and productivity benefits compared to traditional CSS. By leveraging its features, developers can write cleaner, more maintainable code and streamline the process of styling web applications and websites.

***2.1.4.MS SQL DB:***

Microsoft SQL Server (MS SQL Server) is a powerful relational database management system (RDBMS) developed by Microsoft. It provides a comprehensive platform for storing, managing, and analyzing structured data. Key features include support for Transact-SQL (T-SQL), scalability options, high availability solutions, robust security features, built-in business intelligence tools, and seamless integration with the Microsoft ecosystem. MS SQL Server is widely used in enterprises for its reliability, performance, and extensive feature set, making it a preferred choice for managing critical business data.

## ***2.2 USE CASE MODEL:***

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.



**Figure 2.2.1 Use Case Model(Project Flow)**

***2.3. Azure AD Authentication and Authorization:***

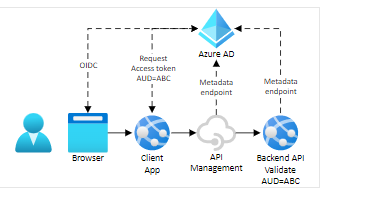
Azure Active Directory (Azure AD) is Microsoft's cloud-based identity and access management service, offering comprehensive authentication and authorization functionalities for cloud applications and resources.

Authentication in Azure AD involves verifying the identity of users attempting to access resources or applications. It supports various authentication methods, including username and password, multi-factor authentication (MFA), OAuth, and OpenID Connect. Users can authenticate using their Azure AD credentials, social media accounts, or external identity providers. Azure AD also provides seamless single sign-on (SSO) capabilities, allowing users to access multiple applications with a single set of credentials.

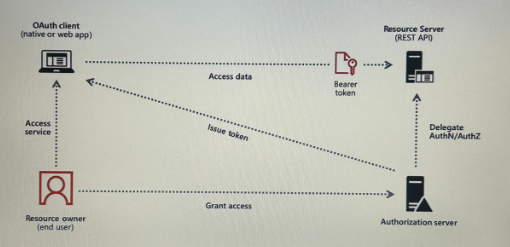
Authorization in Azure AD governs access to resources based on predefined permissions and roles. It employs security principles such as role-based access control (RBAC) and conditional access policies to enforce access policies. Administrators can assign users to roles, grant specific permissions, and define access policies based on factors such as user identity, device health, and location. This enables organizations to control access to sensitive data and resources effectively, mitigating security risks.

Furthermore, Azure AD offers advanced security features such as identity protection, privileged identity management (PIM), and threat detection, enhancing the overall security posture of organizations. It integrates seamlessly with Microsoft's cloud services, including Azure, Office 365, and Dynamics 365, as well as with third-party applications and services through industry-standard protocols.

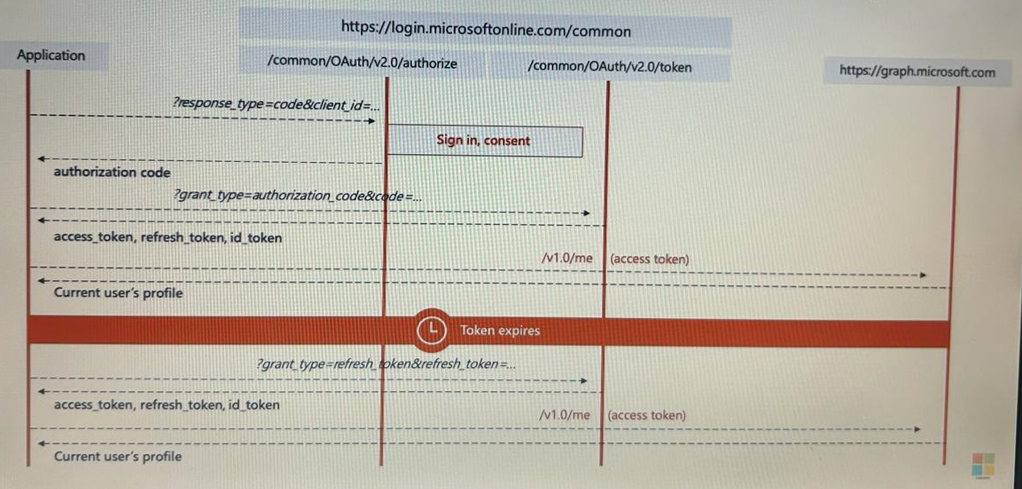
In summary, Azure AD provides a robust and scalable identity and access management solution for organizations of all sizes. Its comprehensive authentication and authorization capabilities, coupled with advanced security features, make it a preferred choice for securing cloud-based applications and resources in today's digital landscape.

****

**Figure 2.3.1 Use Case Model(Authentication and Authorization Flow)**

****

**Figure 2.3.2 Use Case Model(Authentication Flow)**

****

**Figure 2.3.3 Use Case Model(Authorization Flow)**

## ***2.4 SOFTWARE REQUIREMENTS SPECIFICATION:***

***2.4.1. FUNCTIONAL REQUIREMENTS :***

***Login:***

A login page must be designed with which the user can login with the provided credentials.

***Profile:***

Here the Profile where the Authenticated users details are specified after they login.

***Logout:***

Once the user logged in they must logout and then they are authenticated.

***2.4.2. NON- FUNCTIONAL REQUIREMENTS***

***Friendly:***

The application must be user friendly and interactive. It must also be easy to navigate. Since the majority of the end users are normal peoples, it must be easy for them to perform their desirable operations.

***Usability:***

This application aims to help the users to book appointments and manage their medical records and it provides all possible features to achieve this.

***Availability:***

With the server’s uptime as 99.9%, this application is operable and available to the all the users with a stable internet connection.

***Security:***

The security of the user is given prime importance and this application ensures the user’s data security. The data of the user cannot be accessed by anyone without the knowledge of the user.

***Privacy:***

The privacy of the user is maintained as the users can access only the book appointment and manage their members.

***Compatibility:***

The application is compatible across all browsers and mobile devices.

***Performance:***

This application has the capability of providing optimum performance despite the workload. It has a quick response time concerned with the user interface and provides the resultant computed data at that time.

***Scalability:***

This application implements horizontal scaling by increasing the resources to not affect the performance.

***Reliability:***

This application is fool proof and bug free thus maintaining reliability.

***Maintainability:***

This application’s code is readable as it is refactored from time to time. It is easy to repair and enhance. The bugs are fixed from time to time and the existing functionalities are enhanced.

***Durability:***

This application ensures serviceability and hence aims to serve users for a long period of time.

**ASSUMPTIONS:**

***Resources:***

* End users will be available to test during the time they agree to

***Delivery:***

* Project servers arrive configured as expected
* Correct number of downloads happen on the desired day.

***Budget – estimated cost of the project***

* Project costs will stay the same as initially budgeted costs
* Training will be conducted internally with no additional training costs incurred

***Finances – funding to complete the project***

* Funding for licenses will be provided by various departments as needed

***Scope – the scope of the you’re going to deliver***

* The project scope will not change once the stakeholders sign off on the scope statement

***Schedule: tasks, duration and dependencies needed to complete the project***

* Materials will arrive as planned within the project schedule
* Vendor contracts will be fully executed within two months of vendor selection.

## **2.5 TEST PLAN**

Test planning is the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project. It also defines the size of the test effort. It is the main document often called a master test plan or a project test plan and is usually developed during the early phase of the project.

***2.5.1 TEST SCOPE***

Factors influencing test scope are the size of the project, complexity of the project, budget for the project, time scope for the project, number of staff, why tests at different levels, software development naturally split into phases, easily tracking bugs, ensuring a working system, software reuse more practical. Testing takes 30 percent time of development.

***2.5.2 TESTING TECHNIQUES***

##### ***Manual Testing***

Manual Testing finds out the defects or bugs in the application. Testing manually plays an important role for the end-user and verifies that all the features of the application are working correctly. Test cases are executed manually without using any automation tools. And problems faced by customers are rectified manually.

##### ***Unit Testing***

Unit Testing is done at the end of each development level to test particular functions and sub-modules.

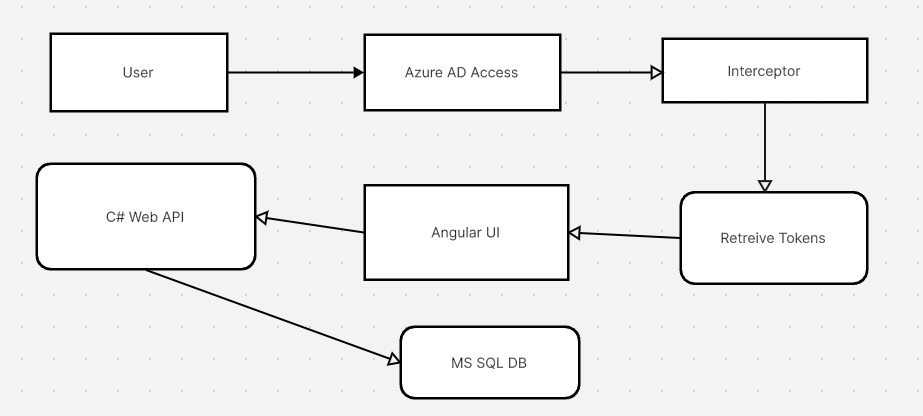
**CHAPTER III**

# SYSTEM DESIGN

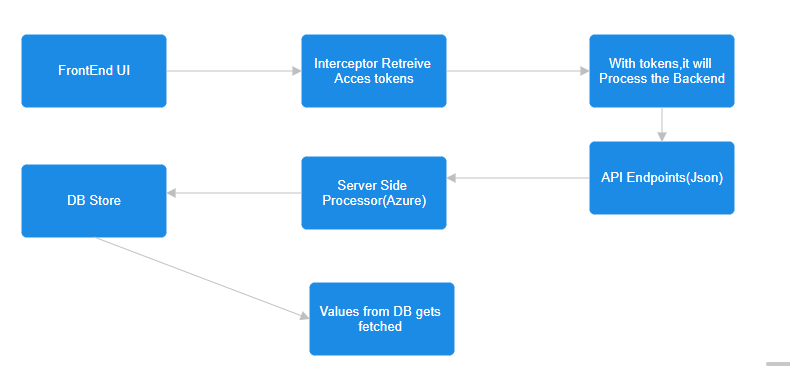
Systems design identifies the software components, specifies relationships among components, properties of both components and relations, defines program structure and provides a blue-print for implementation. The following chapter deals with the various design issues that guides the interface development of the application.

### **3.1 ARCHITECTURAL DESIGN**

This application utilizes layered architecture because the code of this application can be decomposed into several layers with each layer designed to serve a purpose and provide service to the next layer. The other important use of this layer is that it provides a layer of abstraction from each layer.



**Figure 3.1 Architectural design**

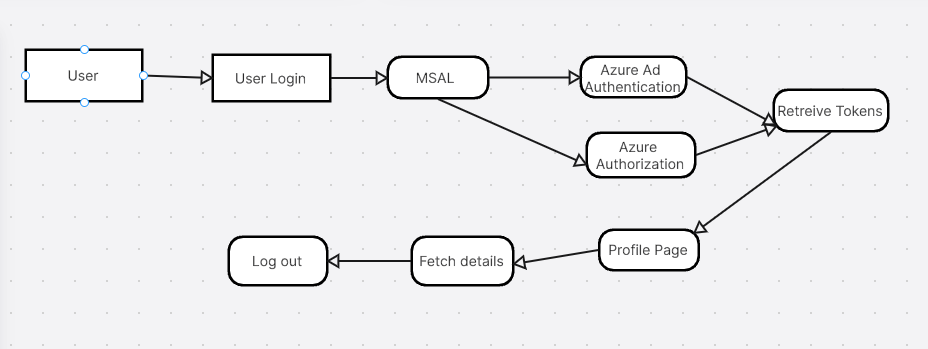
 **Figure 3.2 Architectural design flow of Frontend to Backend**

#### Top of Form

# 3.2 STRUCTURAL DESIGN:

***3.2.1 DATAFLOW:***

***LEVEL 1 DFD:***



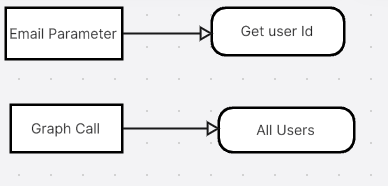
**Figure 3.2.1 Level 1 DFD**

***MODULE DESIGN***

A modular design is an approach for product designing which is used to produce a complete product by integrating or combining smaller parts that are independent of each other. Each of these individual components is then integrated (or assembled) together to form the final product.

***Email :***

Using the email parameter is a common and convenient way to uniquely identify users in systems where email addresses serve as primary identifiers. It simplifies the process of retrieving user information and enables seamless integration with other systems and services.



**Figure 3.3.1 Module design(Get user)**

***From User Id:***

This parameter represents the unique identifier of the user who is sending the message. It could be an ID assigned by the system or an identifier unique to your application's user management system.

***To User ID:***

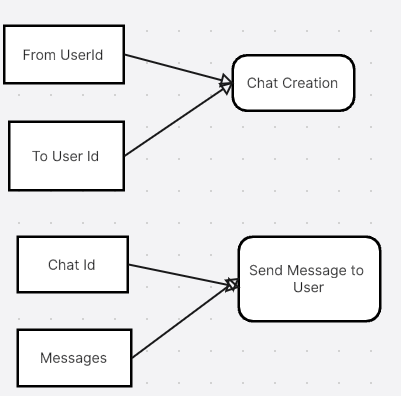
This parameter identifies the user or recipient who will receive the message. Similar to the From User ID, it could be a system-assigned ID or a custom identifier within your application.

***Chat Id:***

In the context of a chat system, the Chat ID refers to the unique identifier of the chat room or conversation where the message will be sent. This allows the system to route the message to the appropriate chat room or thread.

***Message Content:***

This parameter contains the actual content of the message that will be sent. It typically includes text or multimedia content, such as images or files, that the sender wants to communicate to the recipient(s).



**Figure 3.3.2 Module design(Chat Creation-One to one)**

***Email:***

Using the email parameter is a common and convenient way to uniquely identify users in systems where email addresses serve as primary identifiers. It simplifies the process of retrieving user information and enables seamless integration with other systems and services.

***Team Id:***

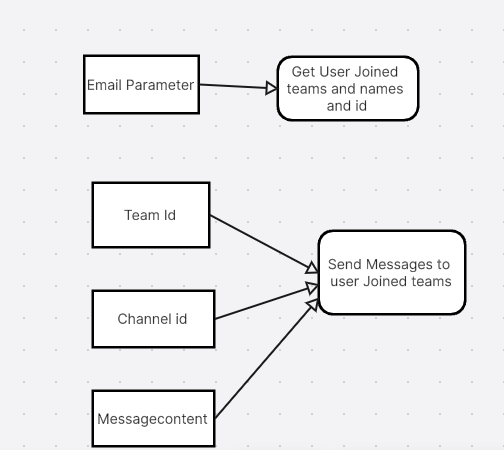
The Team ID is essential for managing team-specific settings, permissions, and resources. It helps identify the members, channels, and content associated with a particular team. When interacting with the Microsoft Teams API or developing custom integrations, specifying the Team ID enables actions such as sending messages, accessing team resources, or managing team settings. Overall, the Team ID plays a crucial role in organizing and managing collaboration within Microsoft Teams, providing a structured framework for teamwork and communication.

***Channel Id:***

he Channel ID facilitates targeted communication and content sharing within these specific subgroups or topics. It helps distinguish between different channels within the same team and ensures that messages and activities are directed to the appropriate channel. When sending messages, uploading files, or performing other actions within Microsoft Teams, specifying the Channel ID ensures that the content is delivered to the correct channel, fostering organized and efficient collaboration within teams.

***Message Content:***

This parameter contains the actual content of the message that will be sent. It typically includes text or multimedia content, such as images or files, that the sender wants to communicate to the recipient(s).



**Figure 3.3.3 Module design(Chat Creation-One to one)**

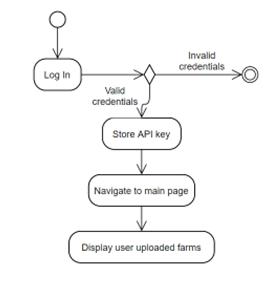
### **3.3 BEHAVIOURAL DESIGN**

#### **ACTIVITY DIAGRAM**

The activity diagram represents the behaviour of the system. It shows the workflow and step-by-step activities and actions performed. it also describes the business processes, use cases and to document the implementation of the system process

***1. LOG IN:***

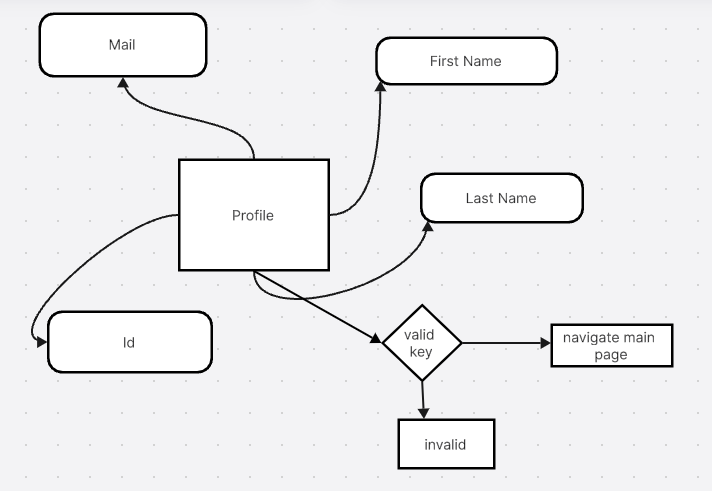
The user tries to log in the application. When the user enters the valid and right credentials then the module stored the API key which is encrypted and then it navigates to the main page of the app.



##### **Figure 3.4.1 Log in Activity diagram**

***2. Profile:***

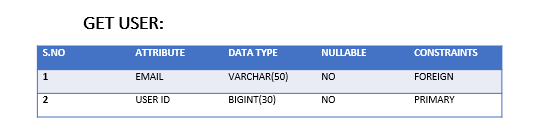
A profile page typically serves as a centralized location within a platform or application where users can view and manage their personal information, preferences, and settings. It provides users with an overview of their account details and allows them to update or modify their profile as needed.



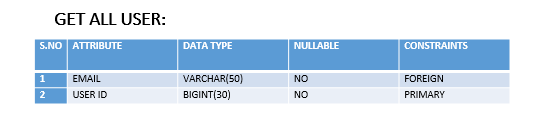
##### **Figure 3.4.2 Profile in Activity diagram**

### **3.4 TABLE DESIGN**

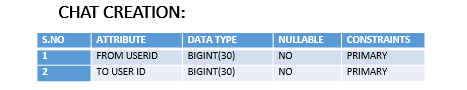
Table-driven design is an approach to software development engineering that is aimed at simplifying and generalizing applications by separating the program control variables and parameters (rules) from the code and placing those in separate external tables. The main objective is to decouple the program control data from the application logic and to put emphasis on modularity to ease change management.



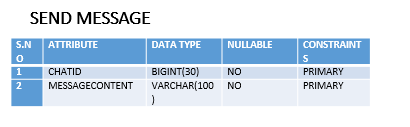
##### **Figure 3.4.1 Get user**



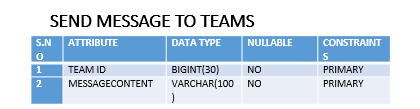
##### **Figure 3.4.2 Get All user**



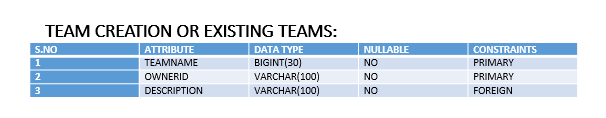
##### **Figure 3.4.3 Chat Creation**



##### **Figure 3.4.4 Send Message**



##### **Figure 3.4.5 Send Message to teams**

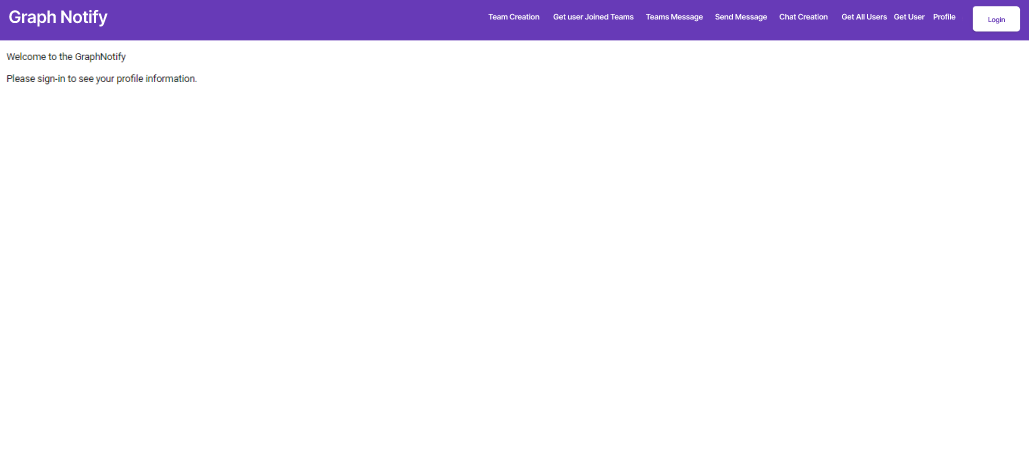


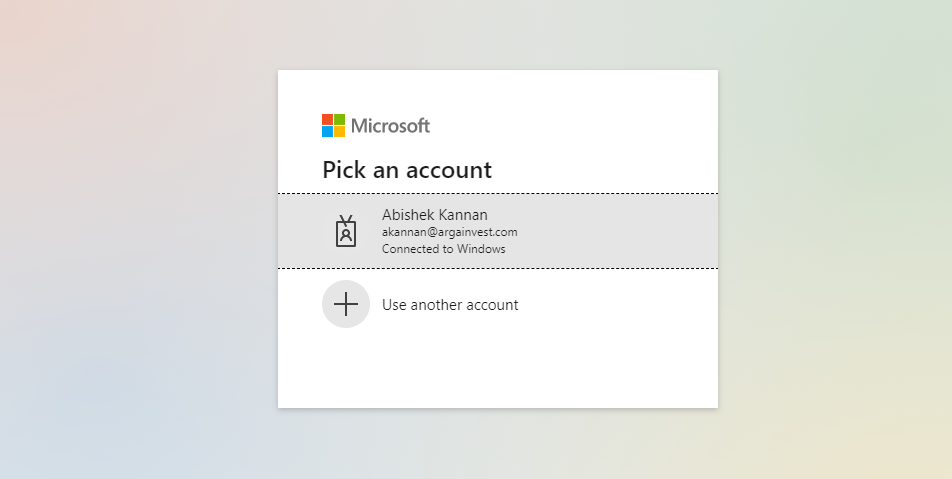
##### **Figure 3.4.6 Team Creation**

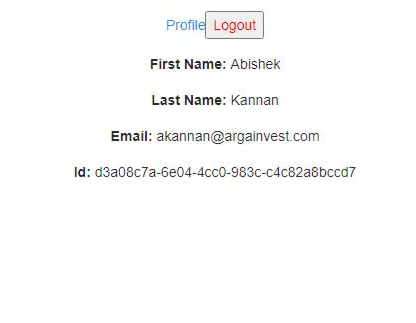
**3.5 USER INTERFACE DESIGN**

The user interface (UI) is the point of human-computer interaction and communication in a device. This can include display [screens](https://www.techtarget.com/whatis/definition/screen), [keyboards](https://www.techtarget.com/whatis/definition/keyboard), a mouse and the appearance of a [desktop](https://www.techtarget.com/searchenterprisedesktop/definition/desktop). It is also the way through which a user interacts with an [application](https://www.techtarget.com/searchsoftwarequality/definition/application) or a [website](https://www.techtarget.com/whatis/definition/Web-site).

**Home:**

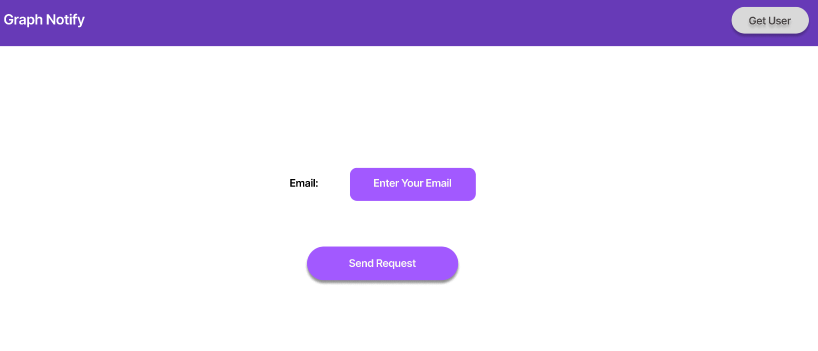
****

**Profile:**

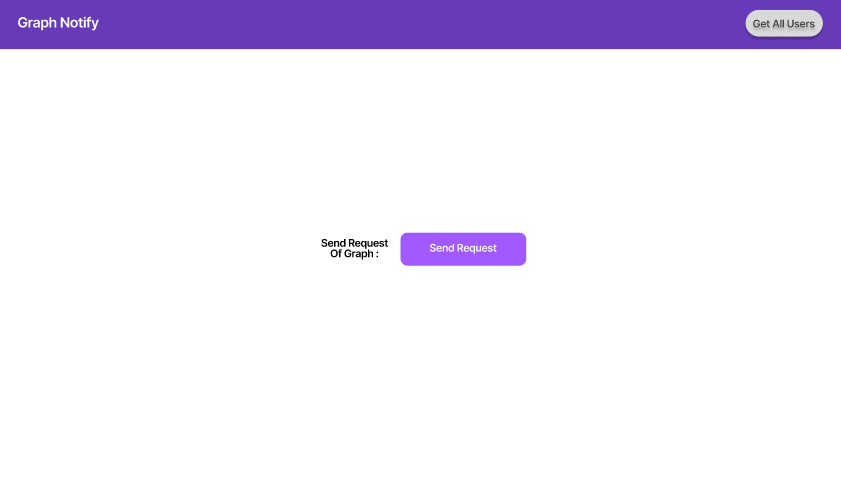
****

**Access Token retrieval:**

****

**GetUser:**

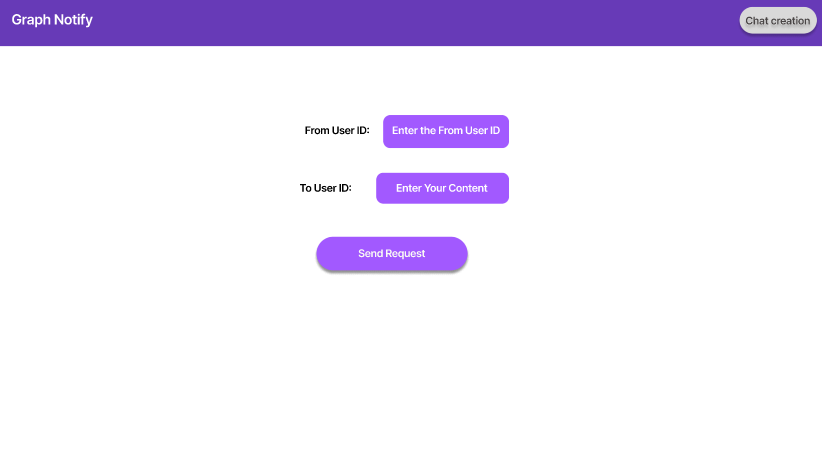
**Get User Output:**

**Get All Users:**

**Get All Users Output:**

****

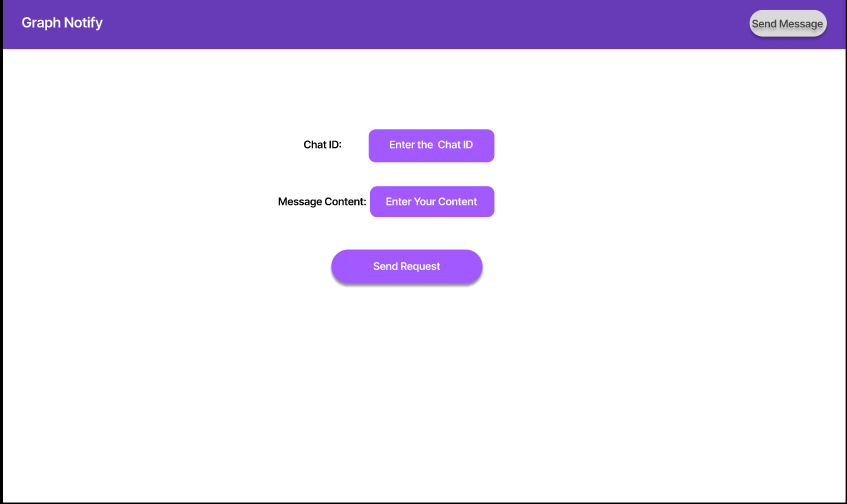
**Chat Creation:**

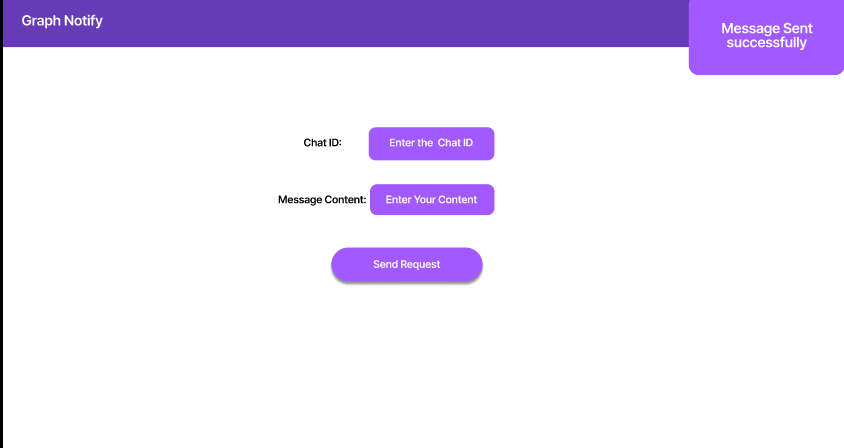
****

**Chat Creation Output:**

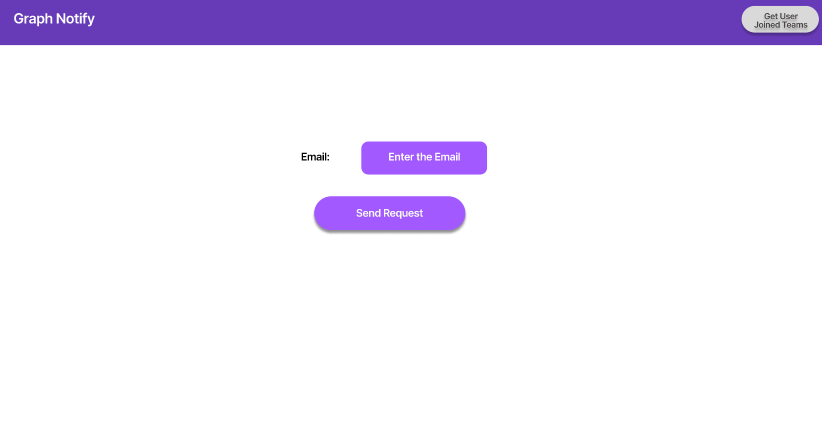
****

**Send Message:**

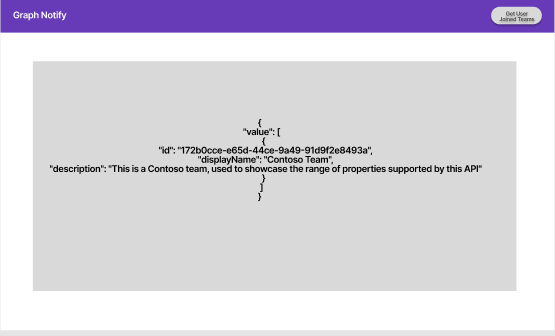
****

****

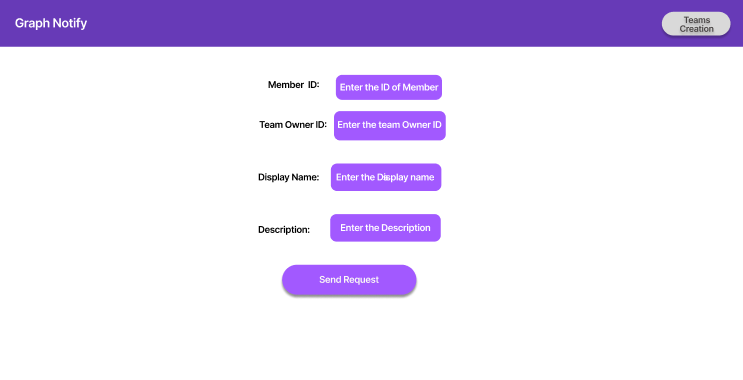
**Get User Joined Teams:**

****

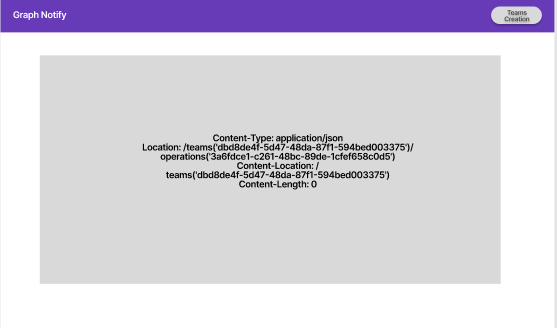
**Get User Joined Teams Output:**

****

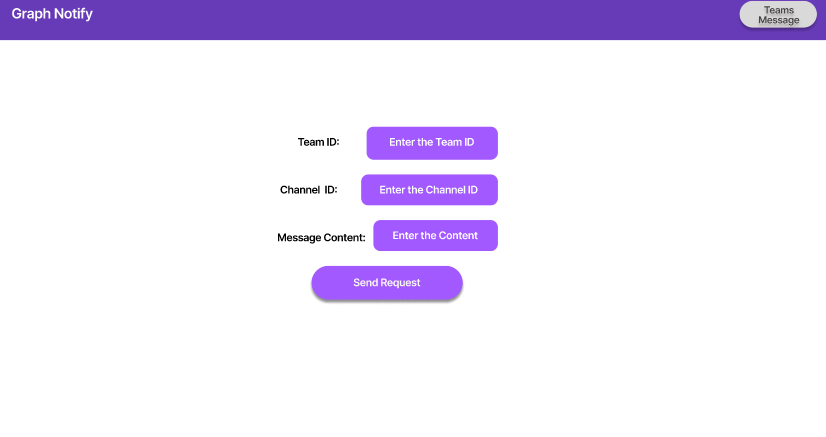
**Teams Creation:**

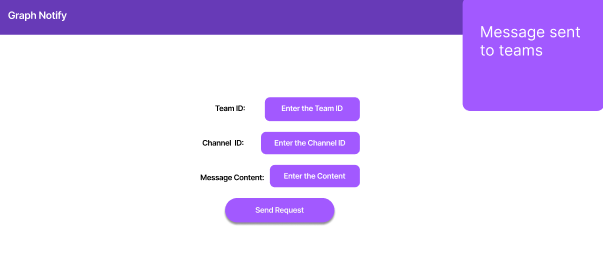
****

**Teams Creation Output:**

****

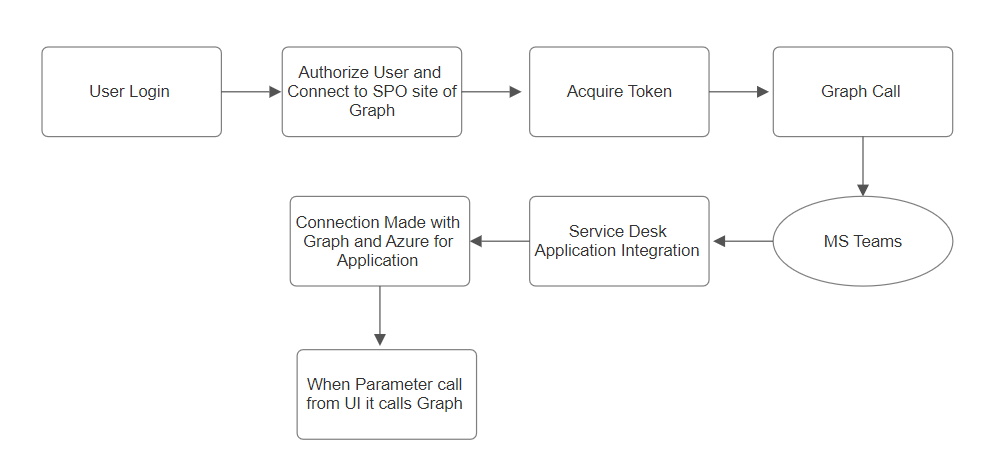
**Teams Message:**

****

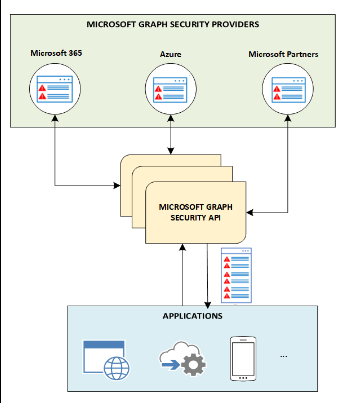
****

**3.6. DEPLOYMENT DESIGN:**

The deployment design of Graph Notify involves planning and executing the deployment process to ensure smooth operation in the production environment. This includes determining infrastructure requirements, ensuring scalability and high availability, implementing robust security measures, automating deployment processes, setting up monitoring and logging, establishing backup and disaster recovery strategies, conducting thorough testing and validation, and documenting procedures for maintenance and support.



**Fig 3.7.DFD OF INTEGRATION DESIGN**



**Fig 3.7.PROCESS OF DEPLOYMENT DESIGN**

**3.7.NAVIGATION DESIGN:**

To send messages to Microsoft Teams using the Microsoft Authentication Library (MSAL) and Graph API, the process involves integrating authentication, API initialization, message preparation, and API request handling.

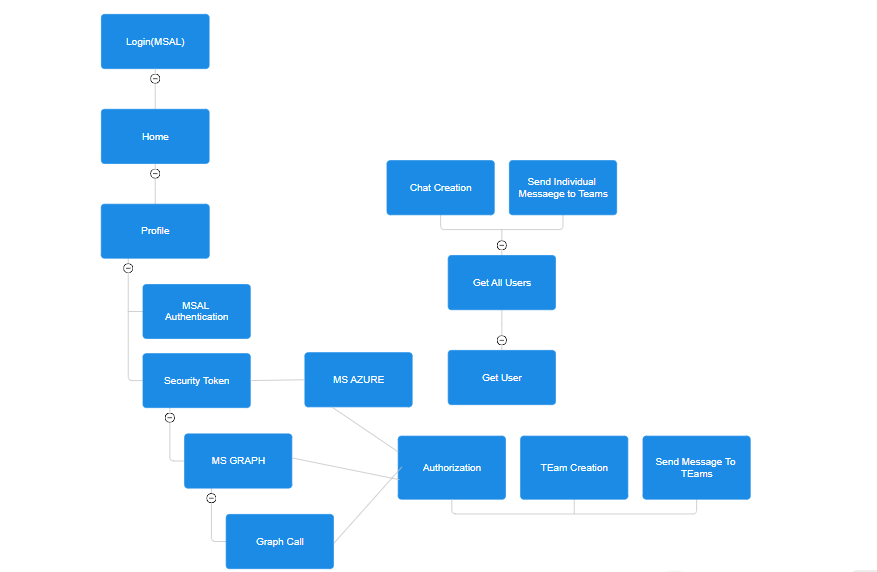
First, users authenticate themselves using MSAL, which initiates a sign-in process where they enter their credentials. Upon successful authentication, MSAL retrieves an access token needed to authorize API requests.

Next, initialize the Microsoft Graph API client using MSAL. Define the appropriate scopes and permissions required for sending messages to Teams. Identify the specific endpoint in the Graph API for sending messages to the desired Teams channel.Craft the message payload, including text, attachments, and formatting, according to the message requirements.

Make an HTTP POST request to the identified Graph API endpoint, passing the message payload and the obtained access token in the authorization header.Handle the API response to confirm the message was sent successfully or address any errors encountered during the process.Ensure robust error handling mechanisms are in place to manage issues like token expiry, network failures, or invalid requests.

Provide clear feedback to users after attempting to send a message, displaying success messages or error notifications as needed.Maintain security by securely storing and managing access tokens and implementing data encryption techniques if handling sensitive information.

Overall, the process involves seamless integration of authentication, API usage, message preparation, and error handling to enable efficient communication with Microsoft Teams via the Graph API and MSAL.



**3.8.CODE DESIGN:**

**PSEUDO CODE :**

***3.8.1. LOGIN***

// Import the necessary MSAL library

import { PublicClientApplication } from 'msal';

// Initialize MSAL with your application's client ID and other configuration options

const msalConfig = {

auth: {

clientId: 'your\_client\_id',

authority: 'https://login.microsoftonline.com/your\_tenant\_id',

redirectUri: 'https://your\_app\_domain.com/redirect',

},

};

// Create a new instance of PublicClientApplication

const pca = new PublicClientApplication(msalConfig);

// Define the login function that initiates the authentication flow

function login() {

// Specify the scopes/permissions needed for your application

const loginRequest = {

scopes: ['openid', 'profile', 'User.Read'],

};

// Call MSAL's loginPopup method to trigger the login flow

pca.loginPopup(loginRequest)

.then(response => {

// Handle successful login

console.log('Login successful');

console.log('Access token:', response.accessToken);

// Redirect user to home page or perform other actions after successful login

window.location.href = 'https://your\_app\_domain.com/home';

})

.catch(error => {

// Handle login error

console.error('Login error:', error);

// Display error message to user or retry login

});

}

// Example usage: Call the login function when a login button is clicked on the login page

const loginButton = document.getElementById('loginButton');

loginButton.addEventListener('click', login);

***3.8.2. AUTHORIZATION AND AUTHENTICATION OF MSAL WITH AZURE AND GRAPH API:***

// Import necessary libraries

import { PublicClientApplication, AuthenticationResult } from 'msal';

// Define Azure AD and Microsoft Graph API configuration

const azureConfig = {

auth: {

clientId: 'your\_client\_id', // Your Azure AD application (client) ID

authority: 'https://login.microsoftonline.com/your\_tenant\_id', // Your Azure AD tenant ID

redirectUri: 'https://your\_app\_domain.com/auth-callback', // Redirect URI after authentication

},

graph: {

endpoint: 'https://graph.microsoft.com/v1.0', // Microsoft Graph API endpoint

scopes: ['User.Read', 'Mail.Read'], // Scopes required for accessing Graph API resources

},

};

// Create a new instance of PublicClientApplication

const pca = new PublicClientApplication(azureConfig);

// Define function to initiate user authentication and obtain access token

function authenticateAndAuthorize() {

// Specify the scopes/permissions required to access Microsoft Graph API

const loginRequest = {

scopes: azureConfig.graph.scopes,

};

// Call MSAL's loginPopup method to trigger the authentication flow

pca.loginPopup(loginRequest)

.then((response: AuthenticationResult) => {

// Check if authentication was successful and access token was obtained

if (response && response.accessToken) {

// Access token obtained successfully

const accessToken = response.accessToken;

console.log('Access token:', accessToken);

// Use the obtained access token to make requests to Microsoft Graph API

fetch(`${azureConfig.graph.endpoint}/me`, {

headers: {

Authorization: `Bearer ${accessToken}`,

},

})

.then(response => response.json())

.then(data => {

// Handle API response (e.g., display user info)

console.log('User info:', data);

})

.catch(error => {

// Handle API request error

console.error('Graph API request error:', error);

});

} else {

// Authentication failed or access token was not obtained

console.error('Authentication failed or access token not obtained.');

}

})

.catch(error => {

// Handle authentication error

console.error('Authentication error:', error);

});

}

// Example usage: Call the authenticateAndAuthorize function when authentication is needed (e.g., on page load)

authenticateAndAuthorize();

***3.8.3. GET ACCESS TOKEN THROUGH USER LOGIN:***

// Import necessary libraries

import { PublicClientApplication, AuthenticationResult } from 'msal';

import axios from 'axios'; // For making HTTP requests

// Define Azure AD and Microsoft Graph API configuration

const azureConfig = {

auth: {

clientId: 'your\_client\_id', // Your Azure AD application (client) ID

authority: 'https://login.microsoftonline.com/your\_tenant\_id', // Your Azure AD tenant ID

redirectUri: 'https://your\_app\_domain.com/auth-callback', // Redirect URI after authentication

},

graph: {

endpoint: 'https://graph.microsoft.com/v1.0', // Microsoft Graph API endpoint

scopes: ['User.Read', 'Mail.Read'], // Scopes required for accessing Graph API resources

},

};

// Create a new instance of PublicClientApplication

const pca = new PublicClientApplication(azureConfig);

// Define function to initiate user authentication and obtain access token

function authenticateUserAndStoreToken() {

// Specify the scopes/permissions required to access Microsoft Graph API

const loginRequest = {

scopes: azureConfig.graph.scopes,

};

// Call MSAL's loginPopup method to trigger the authentication flow

pca.loginPopup(loginRequest)

.then((response: AuthenticationResult) => {

// Check if authentication was successful and access token was obtained

if (response && response.accessToken) {

// Access token obtained successfully

const accessToken = response.accessToken;

console.log('Access token:', accessToken);

// Send the access token to the backend for storage

sendTokenToBackend(accessToken);

} else {

// Authentication failed or access token was not obtained

console.error('Authentication failed or access token not obtained.');

}

})

.catch(error => {

// Handle authentication error

console.error('Authentication error:', error);

});

}

// Define function to send access token to backend for storage

function sendTokenToBackend(accessToken) {

// Make HTTP POST request to backend API endpoint to store the access token

const backendUrl = 'https://your\_backend\_api.com/store-token';

axios.post(backendUrl, { accessToken })

.then(response => {

// Handle successful token storage response from backend

console.log('Access token stored in backend successfully:', response.data);

})

.catch(error => {

// Handle token storage error

console.error('Error storing access token in backend:', error);

});

}

// Example usage: Call the authenticateUserAndStoreToken function when authentication is needed (e.g., on user login)

authenticateUserAndStoreToken();

***3.8.4. GET USER:***

// Import necessary libraries

import { GraphServiceClient } from '@microsoft/microsoft-graph-client'; // Example library for making Graph API requests

// Define function to retrieve user details by email address

async function getUserDetailsByEmail(accessToken, userEmail) {

// Create a new instance of GraphServiceClient using the provided access token

const graphClient = GraphServiceClient.init({

authProvider: (done) => {

done(null, accessToken); // Provide access token to GraphServiceClient

}

});

try {

// Make a request to Microsoft Graph API to get user details by email address

const userDetails = await graphClient

.users

.getByEmail(userEmail)

.select('displayName, jobTitle, mail, mobilePhone') // Specify which user properties to retrieve

.get();

// Return the retrieved user details

return userDetails;

} catch (error) {

// Handle any errors that occur during the API request

console.error('Error retrieving user details:', error);

return null; // Return null or handle error accordingly

}

}

// Example usage: Call the getUserDetailsByEmail function with an access token and email address parameter

const accessToken = 'YOUR\_ACCESS\_TOKEN'; // Replace with your valid access token obtained through authentication

const userEmail = 'user@example.com'; // Email address of the user whose details you want to retrieve

getUserDetailsByEmail(accessToken, userEmail)

.then(userDetails => {

if (userDetails) {

// User details retrieved successfully, do something with the details

console.log('User details:', userDetails);

// Example: Display user details in the frontend UI

displayUserDetails(userDetails);

} else {

// Handle case where user details could not be retrieved

console.error('User details not found or retrieval failed.');

}

})

.catch(error => {

// Handle any unexpected errors

console.error('An error occurred:', error);

});

// Function to display user details in the frontend UI (example)

function displayUserDetails(userDetails) {

// Example: Update HTML elements with user details

document.getElementById('displayName').textContent = userDetails.displayName;

document.getElementById('jobTitle').textContent = userDetails.jobTitle;

document.getElementById('email').textContent = userDetails.mail;

document.getElementById('phone').textContent = userDetails.mobilePhone;

}

***3.8.5. GET ALL USERS:***

// Import necessary libraries

import { GraphServiceClient } from '@microsoft/microsoft-graph-client'; // Example library for making Graph API requests

// Define function to retrieve all users

async function getAllUsers(accessToken) {

// Create a new instance of GraphServiceClient using the provided access token

const graphClient = GraphServiceClient.init({

authProvider: (done) => {

done(null, accessToken); // Provide access token to GraphServiceClient

}

});

try {

// Make a request to Microsoft Graph API to get all users

const users = await graphClient

.users

.request()

.get();

// Return the array of users

return users;

} catch (error) {

// Handle any errors that occur during the API request

console.error('Error retrieving users:', error);

return null; // Return null or handle error accordingly

}

}

// Example usage: Call the getAllUsers function with a valid access token

const accessToken = 'YOUR\_ACCESS\_TOKEN'; // Replace with your valid access token obtained through authentication

getAllUsers(accessToken)

.then(users => {

if (users) {

// Users retrieved successfully, do something with the users

console.log('Users:', users);

// Example: Display users in the frontend UI

displayUsers(users);

} else {

// Handle case where users could not be retrieved

console.error('Failed to retrieve users.');

}

})

.catch(error => {

// Handle any unexpected errors

console.error('An error occurred:', error);

});

// Function to display users in the frontend UI (example)

function displayUsers(users) {

// Example: Loop through the array of users and display each user's properties

const userListElement = document.getElementById('userList');

users.forEach(user => {

const userElement = document.createElement('div');

userElement.textContent = `User: ${user.displayName}, Email: ${user.mail}`;

userListElement.appendChild(userElement);

});

}

***3.8.6.CHAT CREATION BETWEEN TWO USERS AND SEND THE MESSAGE:***

// Import necessary libraries

import { GraphServiceClient } from '@microsoft/microsoft-graph-client'; // Example library for making Graph API requests

// Define function to create a chat with a user and send a message to a Teams channel

async function createChatAndSendMessage(accessToken, recipientEmail, teamsChannelId, messageBody) {

// Create a new instance of GraphServiceClient using the provided access token

const graphClient = GraphServiceClient.init({

authProvider: (done) => {

done(null, accessToken); // Provide access token to GraphServiceClient

}

});

try {

// Step 1: Create a chat with the specified recipient (user) using the Microsoft Graph API

const chatPayload = {

chatType: 'oneOnOne', // Create a one-on-one chat

members: [

{

email: recipientEmail // Email address of the chat recipient (user)

}

]

};

const chat = await graphClient

.chats

.request()

.add(chatPayload);

// Step 2: Send a message to the specified Teams channel using the Microsoft Graph API

const messagePayload = {

body: {

content: messageBody // Message content to be sent

}

};

const sendMessageResponse = await graphClient

.teams

.getById(teamsChannelId) // Retrieve the specified Teams channel by ID

.messages

.request()

.add(messagePayload);

// Return the chat and message details if needed

return { chat, sendMessageResponse };

} catch (error) {

// Handle any errors that occur during the API requests

console.error('Error creating chat or sending message:', error);

return null; // Return null or handle error accordingly

}

}

// Example usage: Call the createChatAndSendMessage function with a valid access token and parameters

const accessToken = 'YOUR\_ACCESS\_TOKEN'; // Replace with your valid access token obtained through authentication

const recipientEmail = 'recipient@example.com'; // Email address of the chat recipient (user)

const teamsChannelId = 'TEAMS\_CHANNEL\_ID'; // ID of the Teams channel where the message will be sent

const messageBody = 'Hello, this is a test message!'; // Message content to be sent

createChatAndSendMessage(accessToken, recipientEmail, teamsChannelId, messageBody)

.then(response => {

if (response) {

// Chat created and message sent successfully

console.log('Chat and message details:', response);

// Example: Display success message or handle further actions

} else {

// Handle case where chat creation or message sending failed

console.error('Failed to create chat or send message.');

}

})

.catch(error => {

// Handle any unexpected errors

console.error('An error occurred:', error);

});

***3.8.7.CREATE TEAM AND SEND THE CHAT OR IN EXISTING TEAM GET THE DETAILS OF THE USER JOINED TEAMS THROUGH MAIL ID:***

FUNCTION createTeam(teamName, privacySetting, creatorEmail):

// Function to create a new team

team = new Team(teamName, privacySetting, creatorEmail)

return team

FUNCTION addMemberToTeam(team, memberEmail):

// Function to add a member to a team

team.addMember(memberEmail)

FUNCTION sendChatMessage(team, senderEmail, message):

// Function to send a chat message to a team

if team.hasMember(senderEmail):

chatMessage = new Message(senderEmail, message)

team.sendMessage(chatMessage)

else:

displayError("Sender is not a member of this team.")

// Example Usage:

teamName = "ProjectX Team"

privacySetting = "Private"

creatorEmail = "admin@example.com"

team = createTeam(teamName, privacySetting, creatorEmail)

addMemberToTeam(team, "user1@example.com")

addMemberToTeam(team, "user2@example.com")

addMemberToTeam(team, ["user3@example.com")](mailto:\"user3@example.com\"))

sendChatMessage(team, "admin@example.com", "Hello team! Let's get started on ProjectX.")

**CHAPTER IV**

# **SYSTEM TESTING**

System testing is a generic metric of performing a variety of tests on a system to explore functionality or to identify problems. A series of systematic procedures are referred to test how the system should perform and where common mistakes may be found by entering data that may cause the system to malfunction or return incorrect information. The purpose of testing is quality assurance, verification and validation, or reliability estimation. A Test Plan documents the strategy that will be used to verify and ensure that a product meets its design specifications.

### **4.1 TEST CASES AND REPORTS**

Test cases are built around the requirements and specifications i.e., what the system is supposed to do. Test cases are generally driven from the external descriptions of the framework and design parameters. Table 4.1 contains the list of test cases and their

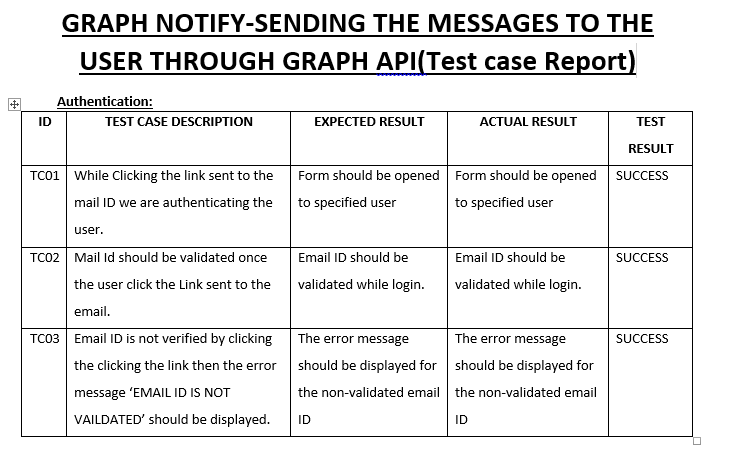
respective test reports.

Each test case contains item criteria such as:

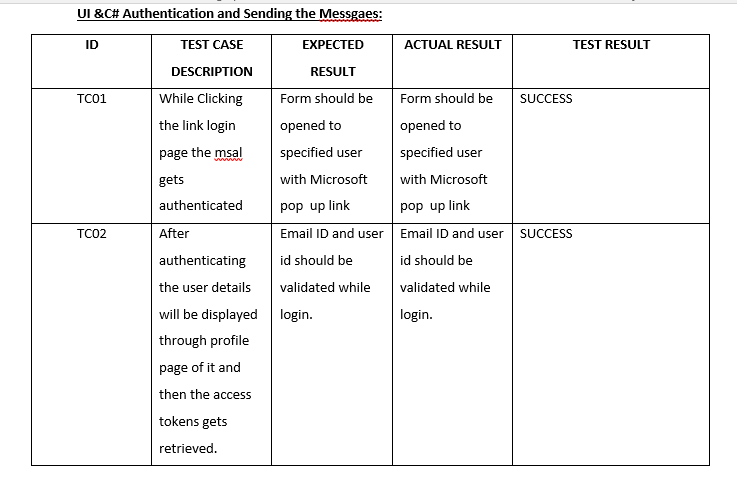
* PASS
  + All expected results are achieved and/or all unexpected events are resolved.
* PASS WITH EXCEPTIONS
  + Unexpected events require alternative procedures that have been

implemented and those events are called as Exceptions.

* FAIL
  + Testing process response does not confirm the expected results.

****

**Table 4.1 – Test Cases and Report**

** Table 4.2 – Test Cases and Report**

**CHAPTER V**

# SYSTEM IMPLEMENTATION

Implementation is the process of converting a new or a revised system design into an operational one. It is the most crucial stage in achieving a new successful system and in giving confidence on the new system for the teams that it will work efficiently and effectively. In this phase, one can build the components either from scratch or by composition. Given the architecture document from the design phase and requirement document from the analysis phase, one can build exactly what has been requested. This stage of project is a true display of the defining moments that make a project a success or a failure. This stage is defined as the system or system modifications being installed and made operational in a production environment.

The phase is initiated after the system has been tested and accepted by the user. This phase continues until the system is operating in production in accordance with the defined user requirements. An implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through programming and deployment. Many implementations may exist for a given specification or standard. The implementation of Graph Notify involves integrating it with Microsoft Teams through the Microsoft Graph API. This includes setting up authentication for secure access, developing the functionality to send messages and notifications, managing user permissions, customizing notification settings, thorough testing, debugging for any issues, deploying the application, and continuously monitoring its performance. Throughout this process, the goal is to ensure seamless communication and collaboration within Teams channels, enhancing productivity and efficiency within the organization.

**CHAPTER VI**

# CONCLUSION

#### Graph Notify represents a powerful solution for enhancing communication and collaboration within organizations using Microsoft Teams. By leveraging the capabilities of the Microsoft Graph API, Graph Notify enables seamless integration with Teams, allowing users to send targeted messages, notifications, and alerts directly within Teams channels. This facilitates real-time communication, keeps team members informed, and fosters collaboration across projects and initiatives. With features such as customizable notification settings, integration with external systems, and future enhancements like integrating with Microsoft Planner, Graph Notify is poised to further streamline communication and task management processes, ultimately enhancing productivity and efficiency within the organization. Overall, Graph Notify empowers teams to stay connected, informed, and productive in today's dynamic work environment.

#### **6.1 FUTURE ENHANCEMENT**

In future enhancements, Graph Notify aims to integrate seamlessly with Microsoft Planner, enhancing task management capabilities within the Teams environment. This integration will enable users to receive task notifications directly within Teams channels or as personal messages, keeping them informed about task updates, deadlines, and assignments. Users will also have the ability to interact with Planner tasks directly from the notification, such as marking tasks as complete or assigning them to team members, without leaving the Teams interface.

# BIBLIOGRAPHY

***REFERENCE BOOKS:***

1. Learning Angular JS:Felipe Coury ,”The ng-book: A Complete Book on Angular”,Createspace Independent Publishing Platform,2018.

***JOURNALS:***

1. Learning Graph API:Arnab Sinha ,IJRAR(International World Wide Web Conferences),535,2018.

***MANUALS:***

1.Learning MS Azure:”What Graph API need to use to get user details and how to code in powerapps to get user details”,Community Support ,2020.

***WEBSITES:***

1. <https://developer.microsoft.com/en-us/graph/graph-explorer>

2. <https://learn.microsoft.com/en-us/graph/api/chat-post?view=graph-rest-1.0&tabs=http>

3.<https://learn.microsoft.com/en-us/graph/api/chat-post-messages?view=graph-rest-1.0&tabs=http>

4.<https://www.w3schools.com/angular/>

5.<https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-8.0&tabs=visual-studio>

6.<https://learn.microsoft.com/en-us/azure/app-service/overview-authentication-authorization>