NORTH SOUTH UNIVERSITY

Center of Excellence in Higher Education

Final Report

North South University
Department of Electrical and Computer Engineering
CSE 299: Junior Project Design
Faculty: RIH

Project Name : NSU-Connect

Project Members:

Shadman Sakib – 2014310042

Sami Md Ragib Anzum – 1931743042

CSE299, Section – 8 , Group – 7

Submitted Date: 28-10-2023

Faculty Advisor

____

____

ECE Department
Summer 2023

DECLARATION

This is to certify that this Project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. Any material reproduced in this project has been properly acknowledged.

Students' names & Signatures

1.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _

2.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _

3.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _

APPROVAL

We, ______________, ____________ and ___________, members of CSE 299 (Junior Project Design) from the Electrical and Computer Engineering department of North South University; have worked on the project titled "________________" under the supervision of _____________ as a partial fulfillment of the requirement for the degree of Bachelors of Science in Engineering and has been accepted as satisfactory.

Supervisor's Signature

....

____

____

Department of Electrical Engineering & Computer Science

North South University

Dhaka, Bangladesh.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude and appreciation to North_South University for providing me with the opportunity to undertake the Junior Design Project, CSE299. This project, titled "NSU-Connect," was a significant endeavor that allowed me to apply my knowledge and skills in the field of web development.

I am profoundly thankful to my esteemed faculty member, Rifat Ahmed Hassan (RIH), for his invaluable guidance, support, and mentorship throughout this project. His feedback were instrumental in shaping the direction of this endeavor and led to the completion of the project in time.

I am also indebted to the entire CSE299 Sec-8 Class for their collaboration, and insights which made this project successful. I am grateful for the opportunity to work alongside such talented individuals.

I extend my sincere thanks to North South University for fostering an environment of learning and innovation, which allowed me to explore and expand my skills. The knowledge and experiences gained during this project will undoubtedly serve as a solid foundation for my future endeavors in the field of computer science and web development.

ABSTRACT

The NSU-Connect project presents a solution to the prevalent issue of fragmented student interactions during free periods at North South University (NSU). This digital platform employs a Matching Algorithm to pair students based on availability and interests, fostering spontaneous and meaningful connections. The User-Friendly Interface ensures convenient access across multiple devices. The website includes a detailed class schedule table that is updated semester wise and helps students know of the classroom availability. NSU-Connect facilitates academic collaborations and social engagements through its real-time chat room that extends to both a public group chat and direct messaging. The implementation of a Secure Login system guarantees data privacy and user authenticity. This project was created with the idea of addressing the existing challenges in student interactions in NSU and enhancing the overall university experience of our peers and juniors to come, paving the way for a more connected and engaged NSU community.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Introducing the Project

The NSU Connect project is an initiative that tackles the challenge of idle time gaps faced by NSU students between classes. During these intervals, students often find themselves disconnected from potential peers interested in similar activities. NSU Connect serves as a robust solution, creating a bridge between students with shared interests, allowing them to engage in collaborative academic endeavors, community projects, and social interactions. Unlike existing methods, NSU Connect offers a tailored platform designed exclusively for NSU students, ensuring a seamless experience for every user.

1.2 Motivation behind the Project

The motivation driving the NSU Connect project stems from the observed disconnect experienced by NSU students during their free periods. The absence of structured platforms for immediate and meaningful interactions prompted the development of NSU Connect. The project strives to eliminate the isolation often felt by students, promoting a sense of community and fostering an environment where intellectual discussions, group studies, and recreational activities can thrive. By addressing this fundamental need for connection, NSU Connect aims to enhance the overall university experience for every NSU student.

1.3 Initial Goals of the Project

The primary objective of our project was to address the limitations of existing methods, particularly Facebook Groups, which often lack spontaneity. Our focus was on developing a system to match students based on their interests and availability, ensuring accurate and immediate pairing. This precise matching process was integral to establishing connections that were not only swift but also meaningful. To facilitate meet-ups between paired students, we aimed to create a direct messaging feature. Additionally, we envisioned setting up a public chat room to enable announcements, event discussions, and community collaboration. These goals formed the foundation of our project, emphasizing the importance of prompt and substantial interactions among NSU students.

2. PROJECT OVERVIEW

2.1 Project Environment

The NSU Connect project employs a harmonious blend of front-end technologies, such as HTML, CSS, JavaScript, and the Bootstrap Framework, to craft an intuitive and visually captivating user interface optimized for diverse devices. Meanwhile, the back-end operations are driven by JavaScript and the Firebase Framework, ensuring a powerful backbone for robust functionality and real-time data processing. Operating within a website-based ecosystem, NSU Connect is effortlessly accessible through standard web browsers, promoting extensive usability and enhancing overall accessibility for users.

2.2 Scope of the Project

The scope of the NSU Connect project encompasses the development of a comprehensive platform tailored specifically for NSU students. This platform facilitates secure and personalized access through Student & Admin Login features, ensuring data privacy and user authenticity. The project includes an interactive chat system comprising public chat rooms and direct messaging functionalities, encouraging both group discussions and one-on-one interactions. Additionally, the project's scope incorporates the implementation of a method to pair up students based on their interest and availability, promoting meaningful connections such as group study sessions. Furthermore, the platform has a class database that allows users to check classroom availability, identify the teacher for a specific class, and access other relevant class details.

2.3 User Classes and Characteristics

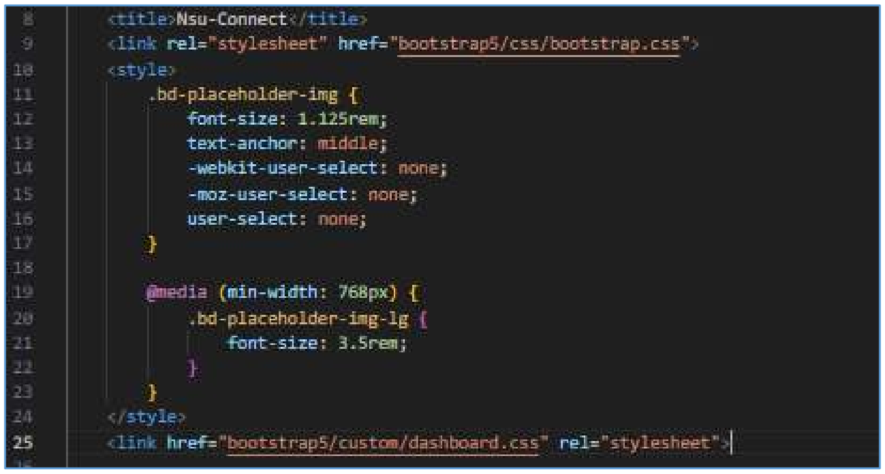
NSU Connect caters to two primary user classes: students and administrators. Students, the primary users, engage with the platform to connect with peers, participate in group activities, and collaborate on academic and community projects. Administrators, on the other hand, oversee the platform's functionality, ensuring a secure and user-friendly environment. Administrators exercise their abilities through the Firebase Terminal.

3. TECHNICAL IMPLEMENTATIONS

3.1 Setting up Bootstrap

Incorporating Bootstrap into NSU-Connect was essential to ensure a visually appealing and responsive user interface. Below are the Bootstrap5 setup configurations employed for NSU-Connect,

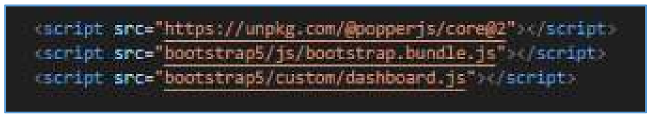
<u>Linking Bootstrap CSS files to our html:</u>



```
<title>Nsu-Connect</title>
<link rel="stylesheet" href="bootstrap5/css/bootstrap.css">
<style>
    .bd-placeholder-img {
        font-size: 1.125rem;
        text-anchor: middle;
        -webkit-user-select: none;
        -moz-user-select: none;
        user-select: none;
    }

    @media (min-width: 768px) {
        .bd-placeholder-img-lg {
            font-size: 3.5rem;
        }
    }
</style>
<link href="bootstrap5/custom/dashboard.css" rel="stylesheet">
```

The first one is the main Bootstrap stylesheet and the second is a custom stylesheet for project-specific styling. As indicated by 'href' path, bootstrap5 was downloaded and saved to the same folder as the source code of the project.

<u>Importing Bootstrap JavaScript libraries:</u>



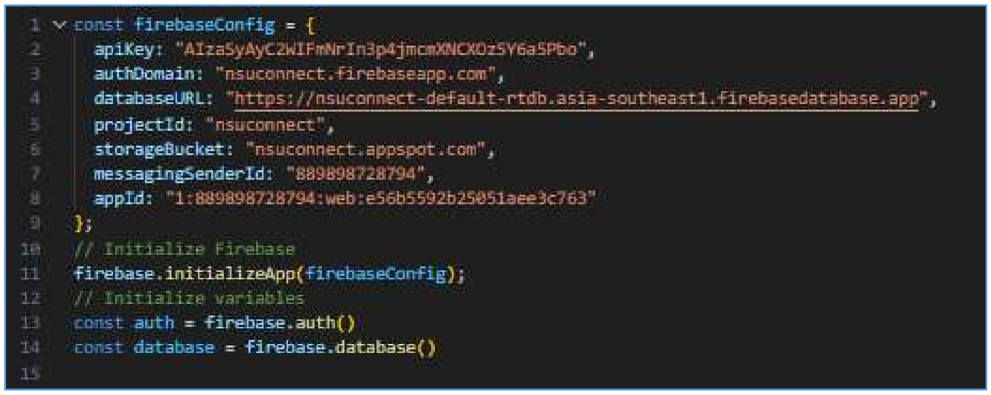
```
<script src="https://unpkg.com/@popperjs/core@2"></script>
<script src="bootstrap5/js/bootstrap.bundle.js"></script>
<script src="bootstrap5/custom/dashboard.js"></script>
```

These script tags import Bootstrap JavaScript libraries. The first line includes the Popper.js core, essential for Bootstrap's dropdowns and other interactive elements.

The second line imports the Bootstrap bundle, which includes Bootstrap's JavaScript plugins. Lastly, the third line links a custom JavaScript file, dashboard.js, tailored to provide specific functionalities within NSU Connect.

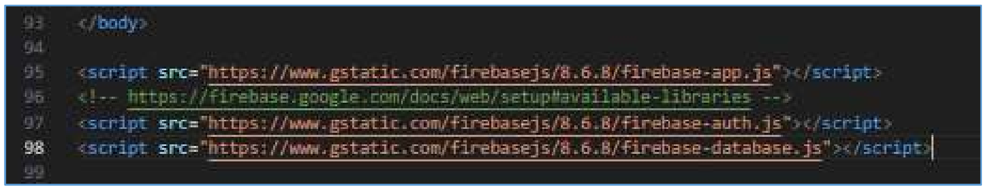
3.2 Setting up Firebase

Setting up Firebase for NSU Connect was a crucial step in ensuring seamless real-time interactions and data management. The first step in initializing firebase is to create a Project in the firebase console that is accessible with a google account.
URL - https://console.firebase.google.com/
After the project is setup, create an app for the project and setup the configuration in your local environment, in our case we loaded Firebase JavaScript SDK libraries from the CDN (content delivery network).
The project's Firebase configuration is outlined below, it forms the backbone of its database operations and user authentication:



```
const firebaseConfig = {
  apiKey: "AIzaSyAyC2WIFmNrIn3p4jmcmXNCXOz5Y6a5Pbo",
  authDomain: "nsuconnect.firebaseapp.com",
  databaseURL: "https://nsuconnect-default-rtdb.asia-southeast1.firebasedatabase.app",
  projectId: "nsuconnect",
  storageBucket: "nsuconnect.appspot.com",
  messagingSenderId: "889898728794",
  appId: "1:889898728794:web:e56b5592b25051aee3c763"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);
// Initialize variables
const auth = firebase.auth()
const database = firebase.database()
```

This configuration, encompassing API keys and database URLs, establishes a secure connection with Firebase services. Additionally, we import necessary software development kits (SDKs) into the html file directly from Firebase to utilize firebase reserved words/functions in our JS code:



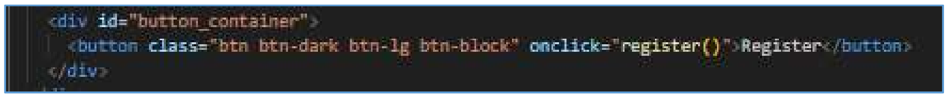
```
</body>

<script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-app.js"></script>
<!-- https://firebase.google.com/docs/web/setup#available-libraries -->
<script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-auth.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-database.js"></script>
```

Next we setup the real-time database, this was initialized in test mode – enabling anyone to write to the database. The rule for the database in test mode was later updated when we started to add nodes to the json file, more on this down the road.

3.3 Implementing User Registration

The first step to enable user registration was to create an appropriate form that takes input of email, username, and password from the user.

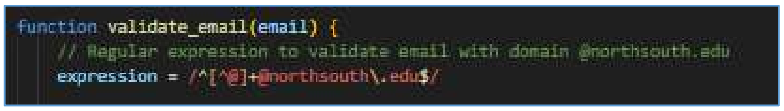
To handle the data received by the form, we created the register function, that is called by- onclick="register()"



```
<div id="button_container">
  <button class="btn btn-dark btn-lg btn-block" onclick="register()">Register</button>
</div>
```

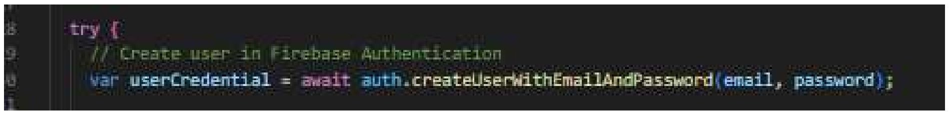
Steps involved in the register function to sign up a user:

- Input Retrieval: retrieve form data
- Input Validation: for example, if the email is not from the '@northsouth.edu' domain), an alert is shown, and the function stops execution.

  ```
  function validate_email(email) {
    // Regular expression to validate email with domain @northsouth.edu
    expression = /^[^@]+@northsouth\.edu$/
  ```

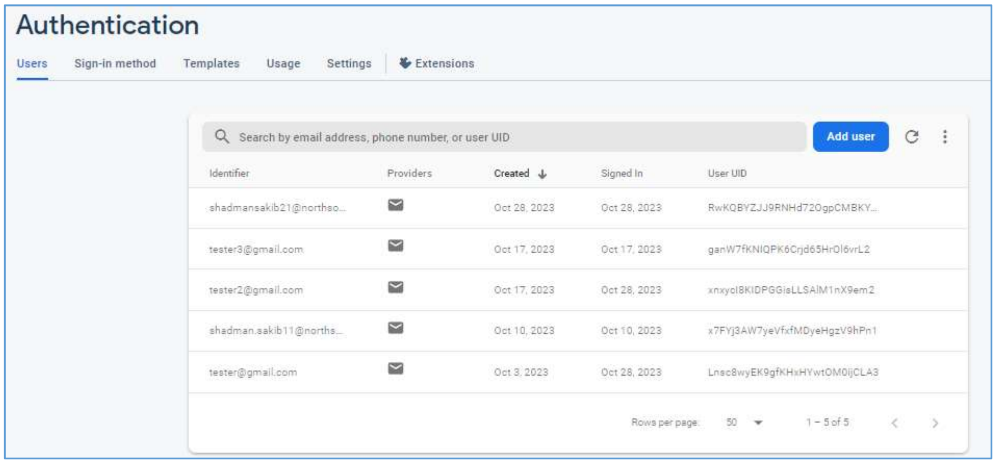
- User Creation in Firebase Authentication:

  ```
    try {
      // Create user in Firebase Authentication
      var userCredential = await auth.createUserWithEmailAndPassword(email, password);
  ```

  createUserWithEmailAndPassword is a firebase method.

  The await keyword ensures that the code waits for this operation to complete before proceeding, preventing errors.

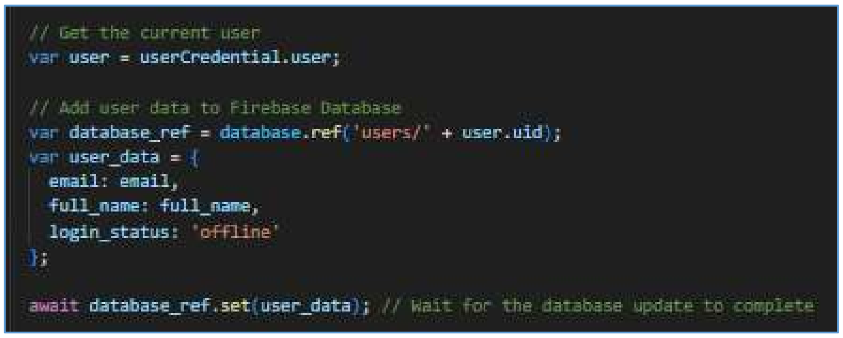
  Authentication

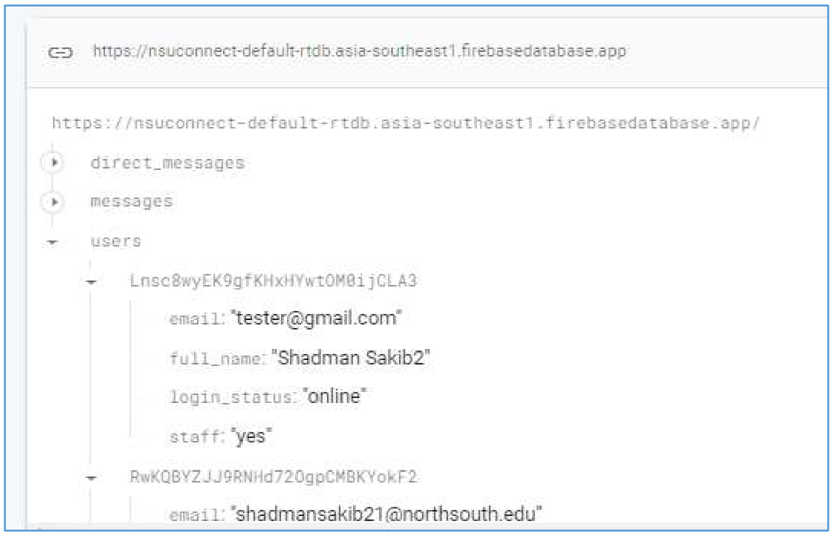
Identifier	Providers	Created	Signed In	User UID
shadmansakib21@northso...	✉	Oct 28, 2023	Oct 28, 2023	RwKQBYZJJ9RNHd72OgpCMBKY...
tester3@gmail.com	✉	Oct 17, 2023	Oct 17, 2023	ganW7fKNIQPK6Crjd65HrOl6vrL2
tester2@gmail.com	✉	Oct 17, 2023	Oct 28, 2023	xnxycl8KIDPGGisLLSAlM1nX9em2
shadman.sakib11@norths...	✉	Oct 10, 2023	Oct 10, 2023	x7FYj3AW7yeVfxfMDyeHgzV9hPn1
tester@gmail.com	✉	Oct 3, 2023	Oct 28, 2023	Lnsc8wyEK9gfKHxHYwtOM0ijCLA3

  Rows per page: 50 1 – 5 of 5

  The user auth data is saved in firebase cloud server securely. User password will never be visible to adminstrators.

- Adding User Data to Firebase Database: Once the user is created, their data (email, full name, and login status) is organized into an object (user_data) and stored in the Firebase Realtime Database under the 'users' node with the user's unique ID as the key. This is done because we do not have a way to manipulate user data on the cloud server.


```
// Get the current user
var user = userCredential.user;

// Add user data to Firebase Database
var database_ref = database.ref('users/' + user.uid);
var user_data = {
  email: email,
  full_name: full_name,
  login_status: 'offline'
};

await database_ref.set(user_data); // Wait for the database update to complete
```


```
https://nsuconnect-default-rtdb.asia-southeast1.firebasedatabase.app

https://nsuconnect-default-rtdb.asia-southeast1.firebasedatabase.app/
  direct_messages
  messages
  users
    Lnsc8wyEK9gfKHxHYwtOM0ijCLA3
      email: "tester@gmail.com"
      full_name: "Shadman Sakib2"
      login_status: "online"
      staff: "yes"
    RwKQBYZJJ9RNHd72OgpCMBKYokF2
      email: "shadmansakib21@northsouth.edu"
```

Staff/admin users are manually added to the database, JSON.

- Success Handling and Redirection.
- Error Handling.

3.4 Implementing User Login

Similar to the user registration process, the user login functionality in NSU Connect is implemented through a form that captures the user's email and password. Upon input validation, the login() function is triggered by an onclick event.

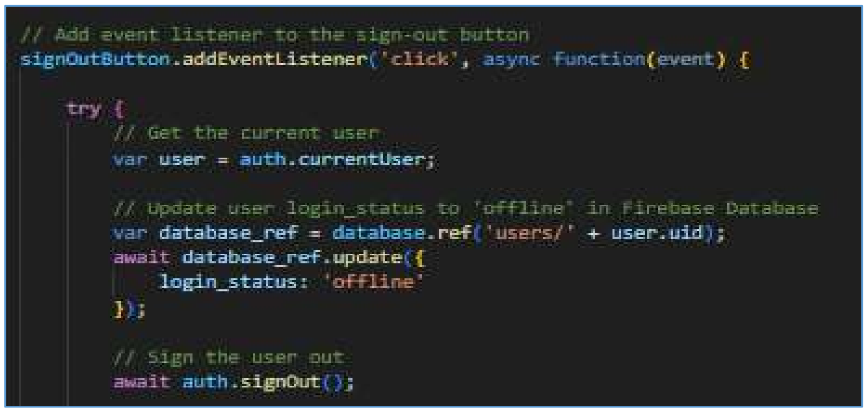
- User Authentication with Firebase: The auth.signInWithEmailAndPassword (email, password) method authenticates the user with the provided email and password. This operation is asynchronous and is awaited to ensure the user is signed in before proceeding. It is also a firebase reserved method imported from the SDK mentioned earlier.

- Updating User Data in Firebase Database: If the login is successful, the user's login status is updated to 'online' in the Firebase Realtime Database under the user's node. This information is crucial for tracking user status and interactions within the application.

3.5 Implementing User Logout & Login Status

In the NSU Connect project, the user logout functionality is implemented through the signOutButton element. Explanation and steps involved:

- Event Listener: The code adds an event listener to the signOutButton element. This listener triggers when the user clicks the sign-out button.
- User Status Update in Firebase Database: Before signing the user out, the code updates the user's login_status to 'offline' in the Firebase Realtime Database.
- User Sign-Out: The auth.signOut() method is called to sign the user out of their Firebase Authentication session.
- Redirection After Logout.
- Error Handling.



```
// Add event listener to the sign-out button
signOutButton.addEventListener('click', async function(event) {

    try {
        // Get the current user
        var user = auth.currentUser;

        // Update user login_status to 'offline' in Firebase Database
        var database_ref = database.ref('users/' + user.uid);
        await database_ref.update({
            login_status: 'offline'
        });

        // Sign the user out
        await auth.signOut();
```

Checking if the user is logged out was crucial for not only our website features, but also in testing if our logout function was working correctly. Our solution:

- Firebase Authentication Listener: The auth.onAuthStateChanged function is used to monitor the authentication state of the user. This listener is called whenever the authentication state changes, such as when a user logs in or out.
- Checking the User's Authentication State: Within the onAuthStateChanged function, it checks whether there is a user object. If there is no user object, it means the user is not logged in or authenticated.
- Redirecting to Login Page if user is not signed in.

3.6 Implementing Public Chat room & Online Users

This was one of the major functionality of the project. Below is a rundown of the steps involved in the design phase.

Step 1: Designing the User Interface

- Create a user interface with sections for online users, chat messages display, input field for typing messages, and a send button. Online users are displayed by a list of user's emails that have 'online' property on users node.
- Structure the UI elements using HTML and CSS & Bootstrap properties.

Step 2: Authentication and User Identification

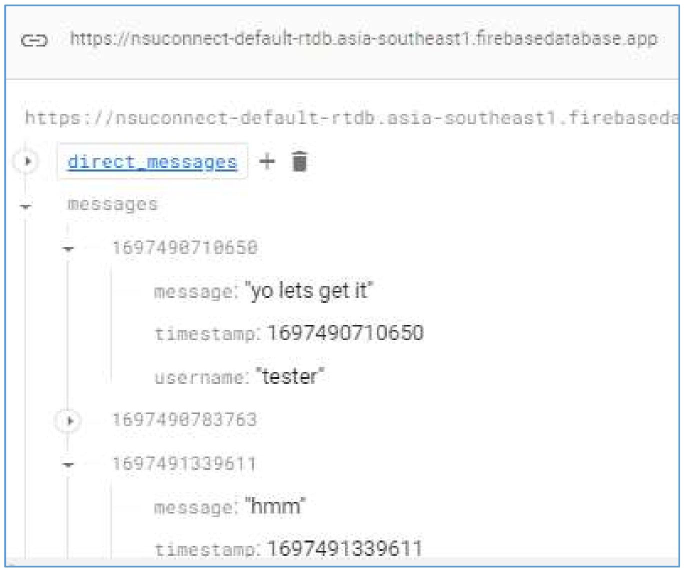
- Implement Firebase authentication to ensure users are signed in before accessing the chat feature.
- Extract the user's email and derive a username from it for identification purposes in the chat room.

Step 3: Sending Messages

- Set up an event listener for the message input form's submit event.
- Capture the message input, along with the sender's username and current timestamp.
- Clear the input field after sending the message.

Step 4: Real-time Database Integration

- Utilize Firebase Realtime Database to store and retrieve chat messages.
- Create a new database node for messages, where each message object includes fields for username, message content, and timestamp.



```
https://nsuconnect-default-rtdb.asia-southeast1.firebasedatabase.app

https://nsuconnect-default-rtdb.asia-southeast1.firebaseda
  direct_messages
  messages
    1697490710650
      message: "yo lets get it"
      timestamp: 1697490710650
      username: "tester"
    1697490783763
    1697491339611
      message: "hmm"
      timestamp: 1697491339611
```

Step 5: Displaying Messages

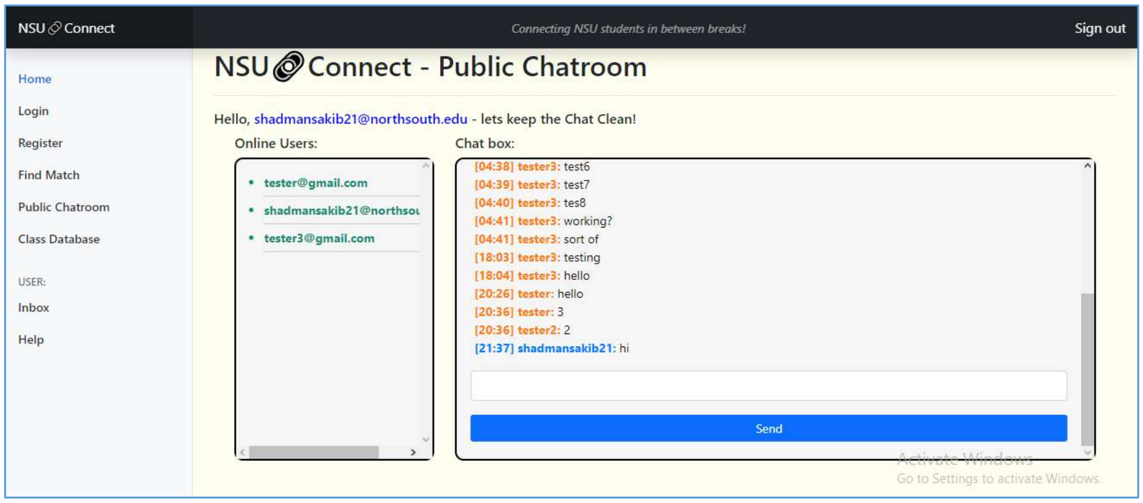
- Implement a listener for the database node using the child_added event.
- Retrieve new messages from the database and dynamically create HTML elements for displaying them.
- Style the messages differently based on whether they were sent by the current user or other users.

Step 6: Timestamp Formatting

- Implement a function to format timestamps into a human-readable format (e.g., HH:mm) for better readability in the chat.

Step 7: User Experience Enhancements

- Provide visual cues, such as different message styles, to distinguish between messages sent by the current user and other users.
- Implement automatic scrolling to enhance the user experience



Hello, shadmansakib21@northsouth.edu - lets keep the Chat Clean!

Online Users:

- tester@gmail.com
- shadmansakib21@northsou
- tester3@gmail.com

Chat box:

```
[04:38] tester3: test6
[04:39] tester3: test7
[04:40] tester3: tes8
[04:41] tester3: working?
[04:41] tester3: sort of
[18:03] tester3: testing
[18:04] tester3: hello
[20:26] tester: hello
[20:36] tester: 3
[20:36] tester2: 2
[21:37] shadmansakib21: hi
```

3.7 Implementing Direct Messaging

The direct messaging feature enables users to send private messages to specific recipients. Below is a summary of the steps involved in its implementation:

Step 1: Designing the User Interface

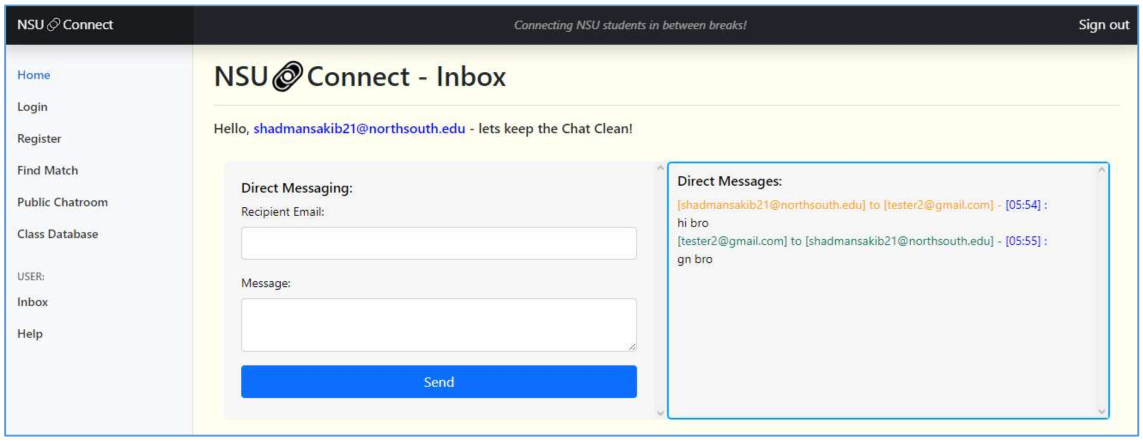
- Create a user interface with sections for entering recipient email, message input, and a send button.
- Set up a separate section to display real-time direct messages.
- Utilize HTML and CSS & bootstrap property to structure and style the direct messaging components.

Step 2: Form Submission and Firebase Integration

- Implement an event listener for the direct message form submission.
- Capture the recipient's email, message content, and the sender's email (retrieved from Firebase authentication).
- Push the direct message data (sender, recipient, message, timestamp) to the Firebase Realtime Database under the 'direct_messages' node.

Step 3: Displaying Direct Messages

- Set up a database listener using the **child_added** event to retrieve new direct messages in real-time.
- Differentiate messages sent by the current user (display in orange) and messages received by the current user (display in green).
- Format the message timestamp into a human-readable format (HH:mm) for improved readability.
- Dynamically create HTML elements for displaying direct messages, including sender, recipient, timestamp, and message content.
- Append the message elements to the direct message view, ensuring smooth scrolling behavior to display new messages.



Hello, shadmansakib21@northsouth.edu - lets keep the Chat Clean!

Direct Messages:

[shadmansakib21@northsouth.edu] to [tester2@gmail.com] - [05:54] :
hi bro
[tester2@gmail.com] to [shadmansakib21@northsouth.edu] - [05:55] :
gn bro

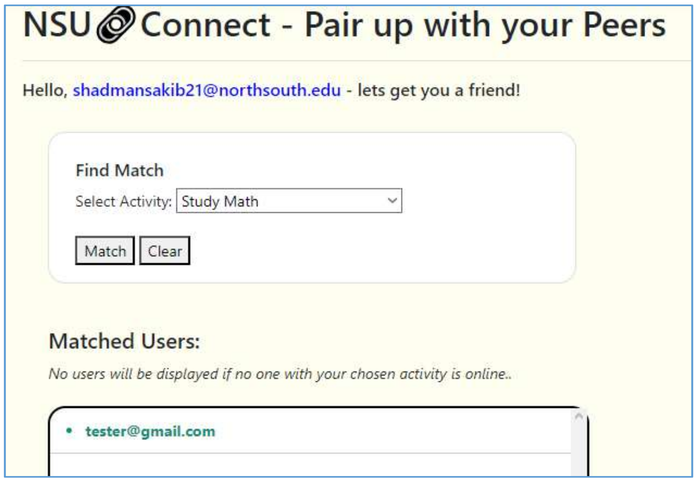
3.8 Implementing User-Pairing

The matching system implemented in this project allows users to find and pair with others who share similar interests. The system utilizes Firebase Realtime Database to store user data and interests.

Implementation steps:

- Create an HTML form that includes a dropdown menu for selecting activities, an input field for entering email, and buttons for pairing and clearing interests.

- Implement a JavaScript function pair() to capture the selected activity from the dropdown menu.
- Access the current user's data from Firebase Authentication.
- Update the user's data in the Firebase Realtime Database by adding the selected activity to the interests field.
- Query the Firebase Realtime Database to find users with the specified activity in their interests field and a login_status set to 'online'.
- Iterate through the query results and display the matched users' email addresses.
- Display the results in list with CSS, JavaScript, and bootstrap utilities.
- Implement a JavaScript function clearInterests() to remove interests from the current user's data in the Firebase Realtime Database. Update the user's data in the Firebase Realtime Database by setting the interests field to null.



NSU Connect - Pair up with your Peers

Hello, shadmansakib21@northsouth.edu - lets get you a friend!

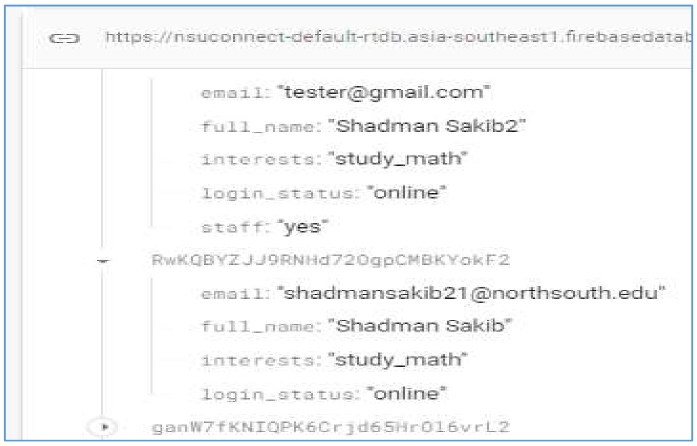
Find Match

Select Activity: Study Math

Matched Users:

No users will be displayed if no one with your chosen activity is online..

- tester@gmail.com



```
https://nsuconnect-default-rtdb.asia-southeast1.firebasedatat
      email: "tester@gmail.com"
      full_name: "Shadman Sakib2"
      interests: "study_math"
      login_status: "online"
      staff: "yes"
  RwKQBYZJJ9RNHd72OgpCMBKYokF2
      email: "shadmansakib21@northsouth.edu"
      full_name: "Shadman Sakib"
      interests: "study_math"
      login_status: "online"
  ganW7fKNIQPK6Crjd65HrO16vrL2
```

3.9 Firebase Rules

Firebase Rules serve as the backbone of security and access control within the Firebase Realtime Database. In our implementation, we have crafted specific rules to regulate data read and write operations, ensuring a secure and reliable messaging system.

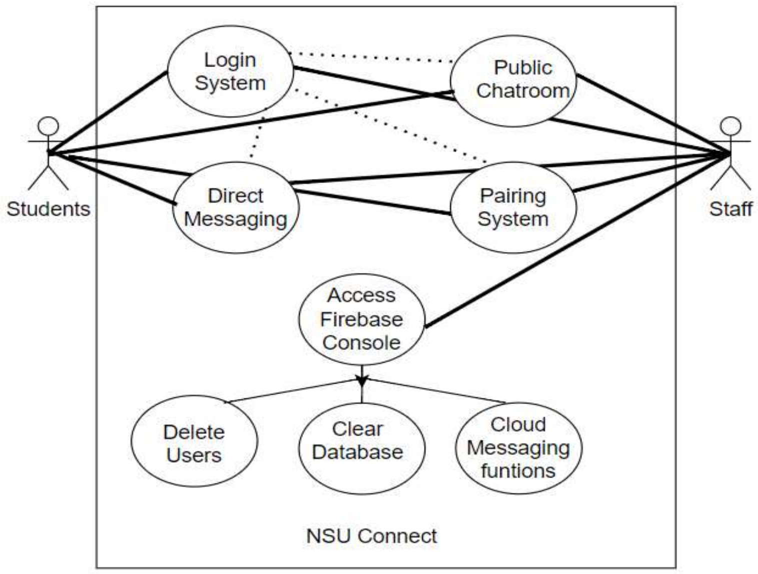
- The top-level rules ".read": true and ".write": true are set to grant public read and write access to all data in the database, allowing data retrieval and modification.
- For the direct_messages node, a more intricate set of rules is defined. Within the direct_messages node, we have restricted read and write access to authenticated users only (".read": "auth != null" and ".write": "auth != null"). This ensures that only logged-in users can access and interact with direct messages data.
- At the level of individual messages ("$messageId"), we have applied validation rules using the .validate attribute. These rules dictate that any new message being written must have its sender or recipient field matching the authenticated user's email address ("newData.child('sender').val() === auth.token.email || newData.child('recipient').val() === auth.token.email"). By enforcing this condition, we guarantee that users can only view messages where they are either the sender or the intended recipient, preventing unauthorized access to private messages.

3.10 Complimentary HTML Pages including Class Schedule.

Created some associative html pages with simple bootstrap and CSS for links. Example: Help page, Terms of Use etc. Class Schedule html is the Class Database tab on the sidenav. It has a search function implemented by JavaScript that can query any data in the table. Originally the plan was to only showcase Class rooms that were not in use, but due to time constraints an appropriate database for it was not created. It is however still useful in the instance where a student may need to check if a particular room like ‘Lib607’ is in use or not. The CSV file of database was converted to an html table and loaded up on the html file.

3.11 Use Case Diagram

There are 2 actors in the use case diagram, Students, and Staff. Staff has all features that students have and are also responsible for handling operations in the firebase console. Our project did not setup any cloud functions like email verification in this version but is intent on doing so before it goes live. Users of NSU connect has to be logged in for all important operations as indicated in the diagram. Class Database is however open to all visiting the website.



4. Results & Discussion

4.1 Summary of Implemented Features

NSU Connect successfully integrated essential features to enhance student interactions. A matching method that pairs up students promoting meaningful connections based on availability and interests. The platform offers an intuitive interface, accessible across devices, facilitating effortless navigation. Public chat rooms and direct messaging foster real-time communication. Robust authentication ensures user security and data privacy. Utilizing Firebase Realtime Database, NSU Connect enables instant and up-to-date interactions, enhancing overall student experience.

4.2 Constraints of the Project

Despite the successful implementation of key features, the project faced certain constraints:

- Limited Features: Due to time constraints and project scope, some additional features that could enhance user experience, such as multimedia sharing and file uploads, were not implemented in the current version.
- Platform Dependency: NSU Connect primarily operates as a web-based application, limiting its accessibility to users who have internet access and a compatible web browser. Native mobile applications for wider platform coverage could be a future consideration.
- Firebase real-time database: Has a maximum storage usage limit. Need to Clear data from database regularly in case of high usage.

4.3 Future Plans for the Project

To address the identified constraints and further enhance NSU Connect, several future plans are envisioned:

- Feature Expansion: Implement additional features such as multimedia sharing, file uploads, and discussion forums to enrich user interaction and engagement.
- Mobile Application Development: Develop native mobile applications for both Android and iOS platforms, ensuring broader accessibility and better user experience across devices.
- Email verification is not setup, Fire base cloud messaging is able to do this very easily. Modifications are to be made before the web project goes live.
- Integration of AI: Explore the integration of AI for intelligent matchmaking, providing more accurate and personalized student connections based on academic interests and extracurricular activities. Also for moderating chat room.
- User Feedback and Iterative Development: Actively gather user feedback through surveys and analytics to identify areas for improvement and develop iteratively.

5. Conclusion

5.1 Concluding Remarks

NSU Connect represents a significant advancement in fostering student connections within the North South University community. By effectively addressing the challenge of fragmented interactions, this platform provides an ideal environment for collaborative learning, idea exchange, and the formation of meaningful relationships. Looking ahead, our commitment to continuous enhancement and innovation will further elevate NSU Connect, ensuring its pivotal role in enhancing the overall student experience at North South University.

5.2 References

- Firebase Documentation. [Online]. Available: https://firebase.google.com/docs
- Bootstrap Documentation. [Online]. Available: https://getbootstrap.com/docs

Project Source Code: https://github.com/ShadmanSakib22/CSE299/