

BookMySpace 1.0

SC – SE – MAR2024 / NAIMURIT1

|  |  |
| --- | --- |
| **Project Group** | |
| Wilson Ma | *Wilsonma320@gmail.com* |
| Junu Paneru | *\*Team member 2 Email Address\** |
| Yahya Muxumed | *\*Team member 3 Email Address\** |
| Wael Sherif | *\*Team member 4 Email Address\** |

|  |  |
| --- | --- |
| SUPERVISED BY | *Jennifer Vaz* |
| CLIENT’S DELEGATE FULL NAME | *Daniel Macbooth\** |
| CLIENT’S DELEGATE EMAIL ADDRESS | *\*Insert here\** |

|  |  |
| --- | --- |
| **Document Submission Date** *31/05/2024* | ***Final Document Wordcount***  xxxx |

Contents

[Abstract 2](#_Toc167360316)

[Introduction 3](#_Toc167360317)

[Methodology 6](#_Toc167360318)

[Database Design 7](#_Toc167360319)

[User Interface Design 11](#_Toc167360320)

[Technology Behind 16](#_Toc167360322)

[Conclusion 16](#_Toc167360323)

[Recommendations 16](#_Toc167360324)

[Bibliography 16](#_Toc167360325)

[Appendix 16](#_Toc167360326)

# Abstract

Over the past months, we have focused on developing and enhancing a web application using a combination of front-end technologies like React.js and backend technologies like Express.js.

**Key Activities:**

1. **Frontend Development with React.js:**

* We started by creating a React.js application to serve as the front end of our web application for its efficiency in building dynamic and interactive user interfaces.
* Components were designed and implemented to create the user interface elements, such as login forms, signup forms, and navigation menus.
* State management was handled using React's built-in state and hooks, allowing for efficient data handling and updates within the application.

1. **Backend Development with Express.js:**

* We developed an Express.js server to handle backend operations such as user authentication and database interactions.
* We defined routes to handle HTTP requests from the front end, including login requests, signup requests, and serving static files.
* Database connectivity was established using MySQL, allowing the server to interact with the database to perform user authentication and data retrieval operations.

1. **User Authentication and Authorization:**

* A key focus of our development was implementing user authentication and authorization mechanisms to ensure secure access to the web application.
* Login and signup functionalities were implemented, allowing users to create accounts, log in with their credentials, and access protected resources based on their user roles.
* User data was securely managed, with passwords hashed and stored in the database to protect against security vulnerabilities.

Overall, the past week has been productive in advancing the development of our web application. By leveraging the capabilities of React.js and Express.js, we have made significant progress towards creating a robust and user-friendly platform for our intended purposes.

# Introduction

**Our Team**

**Wilson Ma**

****

**Flinders University (Australia): Bachelor of Creative Arts (Digital Media), 2013**

* Digital Arts
* History of Arts
* 3D Design
* Web Development
* Cinematography
* Animation

Design experts, always focus on functionality and aesthetic. Have been creating content since 2010.

Team Leader of Team 1, Responsible for UI design, front-end development, and coordinating the entire project.

**Junu Paneru**



**University of Greenwich (UK): Master of Art**

* Web Design and Content Planning

**Saint Louis University (USA): BSIT**

* Data Structure, Web  Technologies, Networking

Interest in front-end, Database design UI and UX design

Team Member of Team 1, responsible for Database Design in this project.

**Yahya Muxumed**



**Middlesex University:** BSc Information Technology

* Computer Systems Architecture
* Computing and Digital Technology

Research interests in Database Management Systems, Backend Development Frameworks and RESTful API Design and Development

Team Member of Team 1, responsible for backend development using Express.js, handling database interactions with MySQL and managing user authentication and authorisation.

# Methodology

# 

# Database Design

**Database Schema**

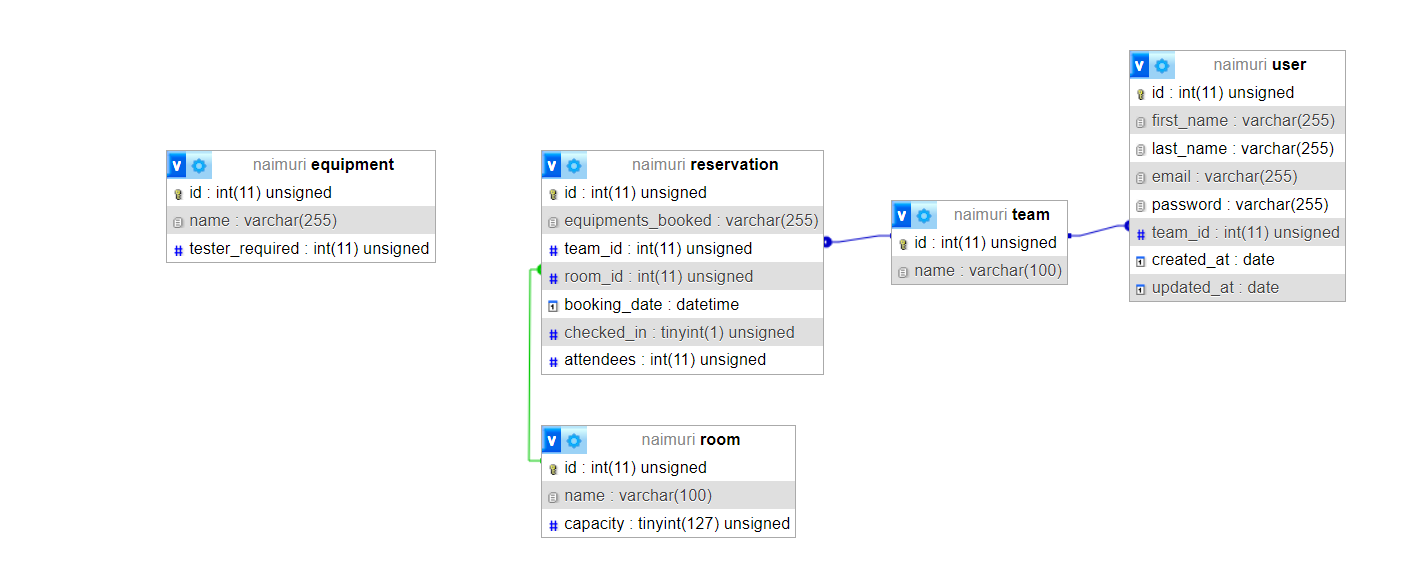


Figure 1 Schema Structure

* **User Table:**
  + This table stores user information.
    - **id:** A unique identifier for each user (primary key, auto-increment).
    - **first\_name:** The user's first name.
    - **last\_name:** The user's last name.
    - **email:** The user's email address.
    - **password:** The user's password.
    - **team\_id:** A foreign key referencing the ‘**team’** table.
    - **created\_at:** The date the user was created.
    - **updated\_at:** The date the user was last updated.

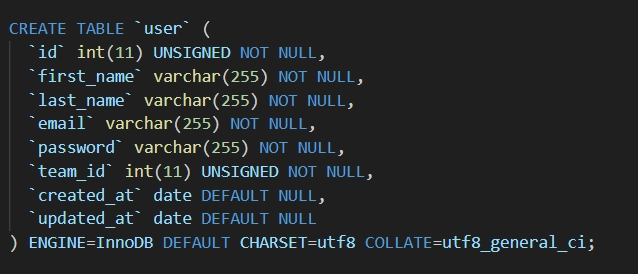


Figure 2 User Table Creation

* **Team Table:**
  + This table holds information about the teams.
    - **id:** A unique identifier for each team (primary key, auto-increment).
    - **name:** The name of the team (e.g., Team A).
* Each user is connected to their team through the ‘**team\_id’** column in the ‘**user’** table, simplifying organisation and team management in the system.

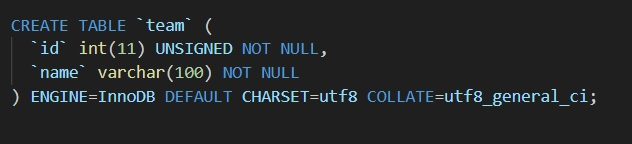


Figure 3 Team Table Creation

* **Reservation Table:**
  + This table keeps track of equipment reservations.
    - **id:** A unique identifier for each reservation (primary key, auto-increment).
    - **equipments\_booked:** A string listing the equipment booked.
    - **team\_id:** A foreign key referencing the ‘**team’** table.
    - **room\_id:** A foreign key referencing the ‘**room’** table.
    - **booking\_date:** The date and time of the booking.
    - **checked\_in:** A boolean indicating whether the reservation has been checked in (1 for true, 0 for false).
    - **attendees:** The number of attendees for the reservation.
  + It contains records of reservations made by teams for specific equipment in particular rooms at certain times.
  + These foreign keys establish relationships between reservations and the teams, and rooms involved in each reservation.

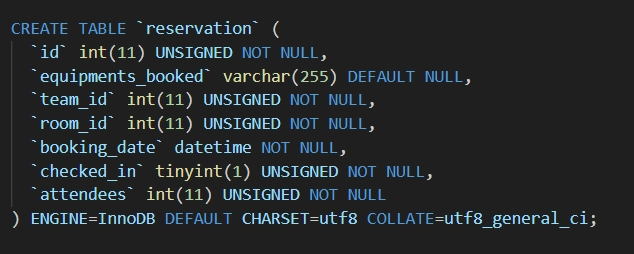


Figure 4 Reservation Table Creation

* **Room Table:**
  + This table contains information about the rooms available for booking.
    - **id:** A unique identifier for each room (primary key, auto-increment).
    - **name:** The name of the room (e.g., Room A).
    - **capacity:** The maximum capacity of the room.
  + It provides information about different rooms available for booking, such as their names and maximum capacities.

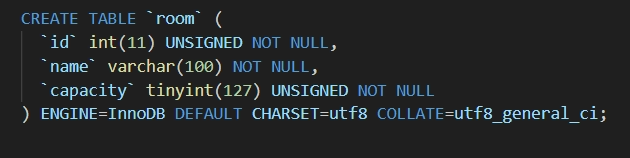


Figure 5 Room Table Creation

* **Equipment Table:**
  + This table stores information about different pieces of equipment available for booking.
    - **id:** A unique identifier for each equipment item (primary key, auto-increment).
    - **name:** The name of the equipment (e.g., Phone, Network).
    - **tester\_required:** An integer representing some requirement for the equipment, possibly the number of testers required.
  + It lists various types of equipment available in the system, along with unique IDs and names.
  + It also specifies the number of testers required for each equipment.

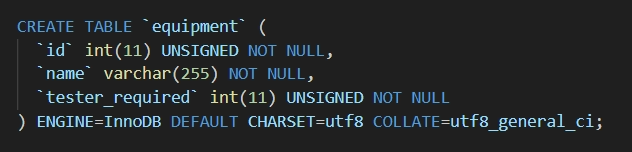


Figure 6 Equipment Table Creation

**ER Diagram**

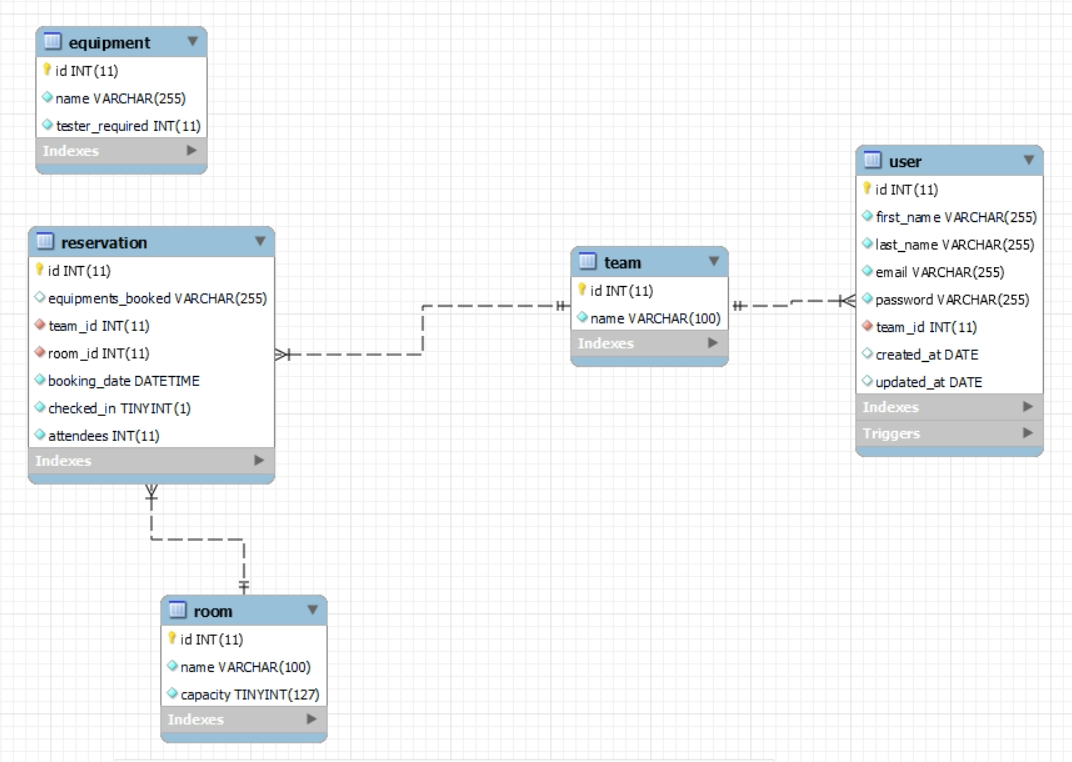
****

Figure 7 ER-Diagram

**One-to-Many Relationships:**

* **User to Team:**
  + Each user belongs to one team.
  + A team can have multiple users.
  + Represented by the **’team\_id’** foreign key in the user table.
* **Reservation to Team:**
  + Each reservation has one and only one team.
  + A team can have multiple reservations.
  + Represented by the **’team\_id’** foreign key in the reservation table.
* **Reservation to Room:**
  + Each reservation is made for one room.
  + A room can have multiple reservations.
  + Represented by the **‘room\_id’** foreign key in the reservation table.

# User Interface Design

# Introducing a user-friendly interface, specifically designed for internal team use. This platform enables seamless booking of equipment and rooms for users, while providing admins with comprehensive management capabilities for both equipment and rooms.

**Login Page**  
Upon logging in, users will enter their credentials on the Login page. The system will then identify their “team\_id” and direct them accordingly. If the user's “team\_id” is not “1”, they will be redirected to the “User Dashboard.” Conversely, if their “team\_id” is “1”, they will be sent to the “Admin Dashboard.”

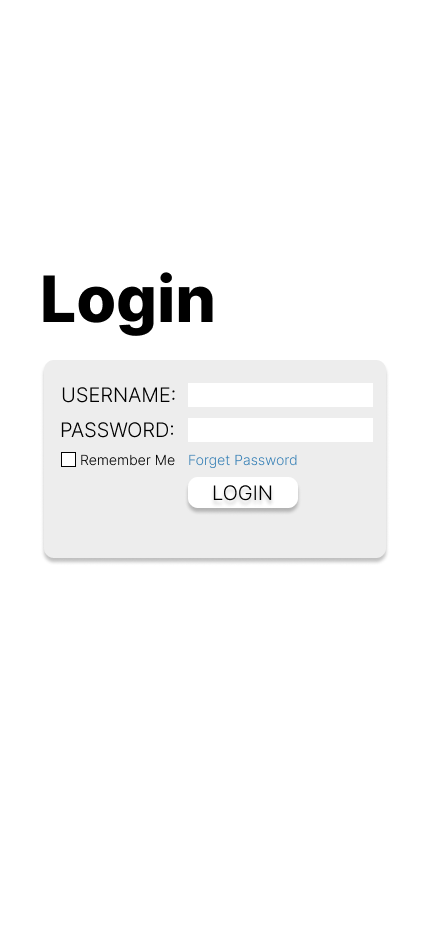


Figure 8 Login page, Routing Users into 2 different sessions

**Dashboard**

After logging in, users will be presented with a Dashboard that displays all bookings made by their corresponding team, as recorded in the database. If a user from Team A logs in, only reservations booked by Team A will be shown. Similarly, if a user from Team B logs in, bookings by Team A will not be displayed.

The booking information will be displayed in a container, which includes the following details:

- Booking Date

- Testing Equipment

- Team’s Name

- Room Booked

Functionality available on the dashboard includes:

1. \*\*Check-in\*\*: Users can mark their bookings as checked-in directly from the dashboard.

2. \*\*Modify Booking\*\*: Users have the option to change the booking date and room.

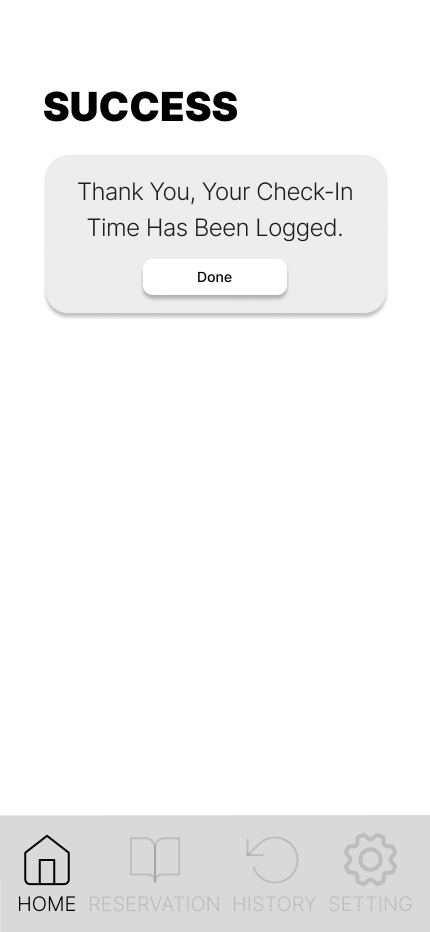
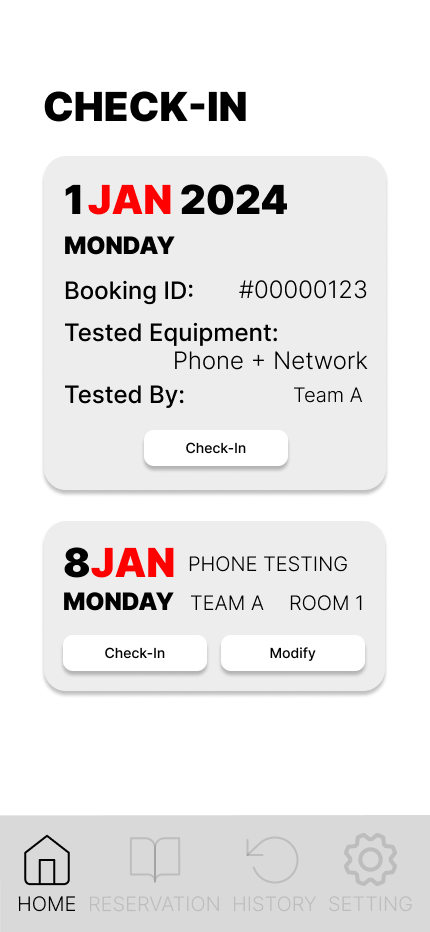
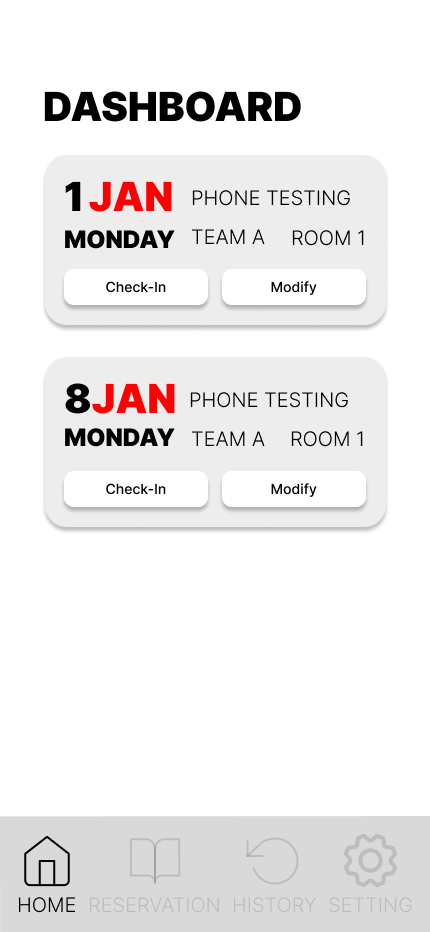


Figure 9 User Dashboard

**Reservation**

Following the booking flow, users will first select the required date and equipment. Multiple pieces of equipment can be booked in a single reservation. Each piece of equipment will display the number of testers required.

Next, users will proceed to book rooms based on the equipment booked. They must select a room with a capacity larger than the total number of testers required. The available capacity of the selected room will be updated accordingly, and the database will reflect the number of users occupying the room.

1. \*\*Equipment Booking:\*\*

- Select the booking date.

- Choose one or multiple pieces of equipment from the list.

- The table will display equipment along with the number of testers required.

2. \*\*Room Booking:\*\*

- After selecting the equipment, proceed to book a room.

- Select a room with a capacity that is larger than the number of testers required.

- The system will update the room's available capacity based on the selection.

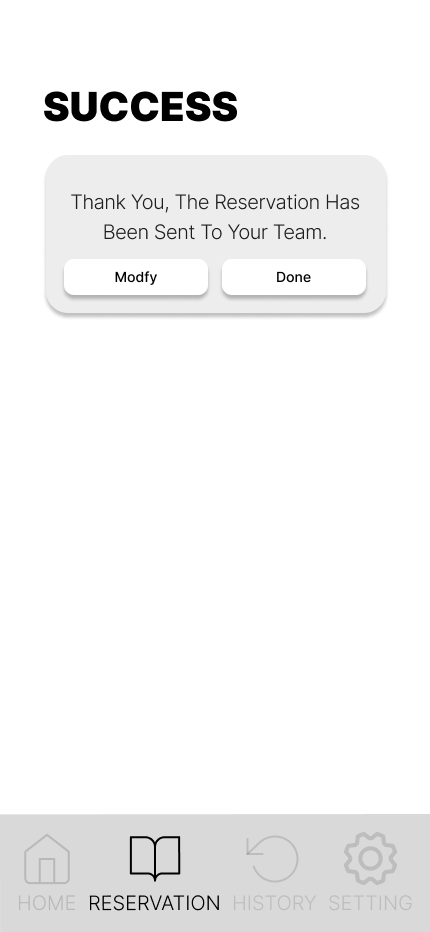
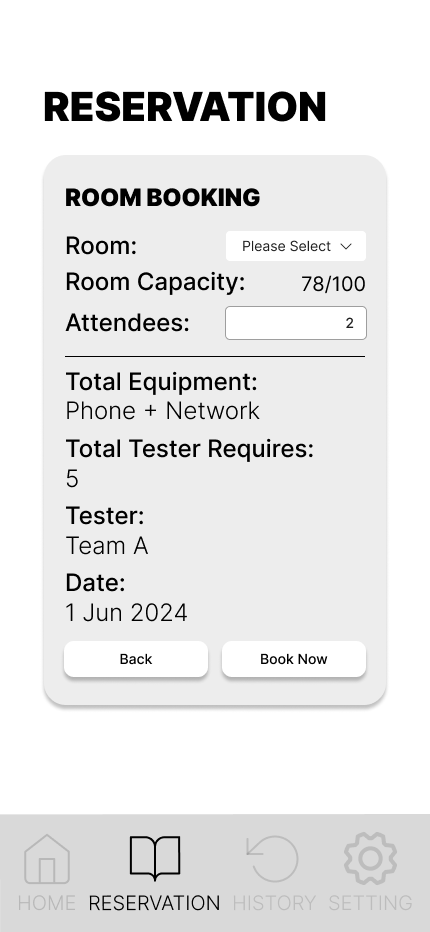
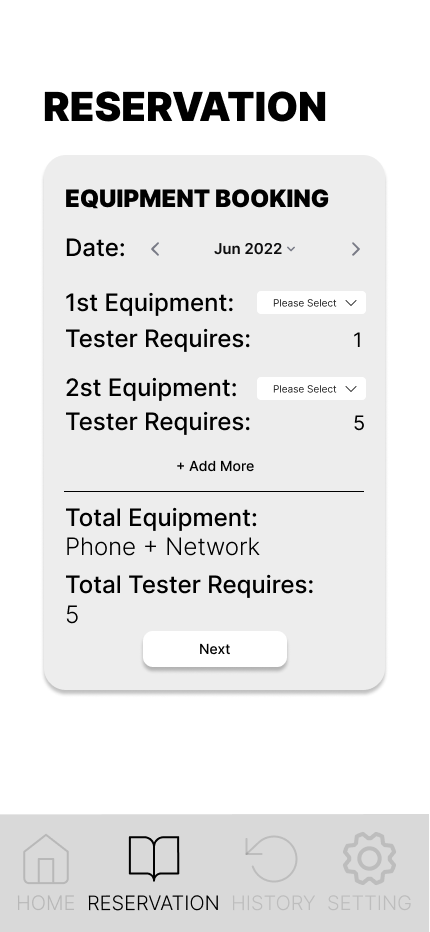


Figure 10 Reservation tab

**History**

In the History tab, all reservation details are displayed, including “booking\_date”, “booking\_id”, “checked\_in”, “booked\_equipment”, “attendees”, and “team\_ID” from the “Reservation” table. The status will be determined based on the “checked\_in” value from the reservation table:

- If the “checked\_in” value is “1”, the status will display as “Checked-In”.

- If the “checked\_in” value is “2”, the reservation will not be displayed.

- If the “checked\_in” value is “3”, the status will display as “Cancelled”.

**History Tab Display:**

- \*\*Dates:\*\*: The date(s) of the reservation.

- \*\*Booking ID:\*\*: The unique identifier for the reservation.

- \*\*Status:\*\*: Refer to the above table

- \*\*Booked Equipment:\*\*: The list of equipment booked for the reservation.

- \*\*Attendees:\*\*: The number of testers or participants for the reservation.

- \*\*Team ID:\*\*: The team associated with the reservation.

This layout ensures that users can easily track and review their past reservations.

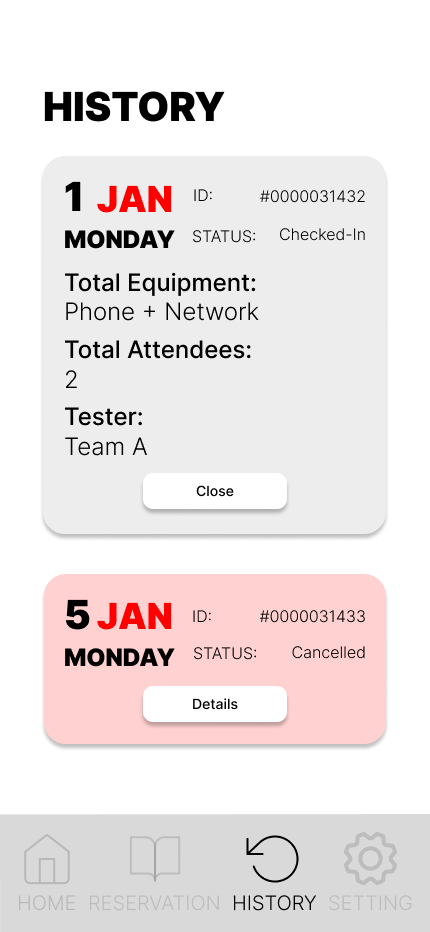


Figure 11 History Tab

**Setting**

2 Simple setting functions will be included here, they are reset password and logout.



Figure 12 Setting Menu

# Technology Behind

# Conclusion

# Recommendations

# Bibliography

# Appendix