Georgia Tech LMC Film Equipment Inventory System

JID 4121

Mengzhu Ou, Daniel Pan, Qihui Wang, Wesley Wilson, Xisheng Zhang

Client: Mr. John Thornton

Repository: **https://github.com/Qihuiwang3/JID-4121-LMCFilm**

Table of Contents

Table of Figures

Terminology

1. **API (Application Programming Interface):** A set of tools and protocols that allow different software applications to communicate with each other, enabling them to interact and share data or services.
2. **Backend:** The part of the system responsible for server-side logic, database interactions, and the overall management of data and business logic. It operates behind the scenes and is not visible to users.
3. **Base64 Encoding:** A method of encoding binary data, such as image files, into a text format for storage or transfer.
4. **EmailJS:** A third-party API enabling email functionality without needing a custom server.
5. **Express.js:** A web application framework for Node.js, designed to simplify the development of server-side applications. It provides a robust set of features for web and mobile applications.
6. **Frontend:** The client-side part of the system that users interact with directly. It involves everything the user experiences visually, including the design, layout, and functionality in a web browser.
7. **HTTPS:** A secure protocol for data transmission, encrypting data exchanged between the user and the website.
8. **HTTP-Only Cookies**: A type of cookie that can only be accessed by the server and is not available to client-side scripts like JavaScript. This makes them a secure option for storing sensitive information, such as session tokens, as they help protect against cross-site scripting (XSS) attacks.
9. **In-Memory Storage**: A type of data storage that keeps data in the system's RAM (random access memory) for fast access and temporary use. It is commonly used for caching or session management, as it provides quicker read/write speeds compared to traditional storage methods but is not persistent and loses data when the system is restarted.
10. **JWT (JSON Web Token)**: A compact and self-contained way of securely transmitting information between parties as a JSON object. It is commonly used for authentication and authorization in web applications, allowing users to log in and securely access resources.
11. **MERN**: A technology stack consisting of MongoDB, Express.js, React, and Node.js, used to develop full-stack web applications. Each component of the stack handles a different layer of the application.
12. **MFA (Multi-Factor Authentication)**: A security method that requires users to provide two or more verification factors to access a system. These factors typically include something the user knows (like a password), something they have (like a phone or security token), or something they are (like a fingerprint), enhancing protection against unauthorized access.

13. **MongoDB:** A NoSQL database that stores data in flexible, JSON-like documents. It is used for managing large volumes of unstructured or semi-structured data.
14. **Node.js:** A JavaScript runtime that allows developers to execute JavaScript on the server side, enabling backend development using the same language as frontend development.
15. **NoSQL:** A type of database that uses non-relational, schema-less data models, ideal for unstructured data.
16. **PayPal Integration:** A secure payment gateway used for handling transactions in the system.
17. **PWA (Progressive Web App):** A type of web application that uses modern web capabilities to deliver an app-like experience. PWAs can work offline, load quickly, and are installable on mobile devices like native apps
18. **React:** A JavaScript library used for building user interfaces, specifically for single-page applications. It allows developers to create reusable UI components.
19. **Redux:** A state management library for JavaScript applications, commonly used with React for managing application state.

Introduction

Background

The film equipment reservation system was developed for the School of Literature, Media, and Communication at Georgia Tech to serve two key user groups: students and administrators, which typically includes professors from the LMC department and occasionally assigned student assistants. The system offers students a seamless experience for reserving film equipment, while providing professors with an efficient platform to manage the entire inventory as administrators. Built as a progressive web application using the MERN stack (MongoDB, Express.js, React.js, and Node.js), it delivers a desktop-like experience. Additionally, it integrates emailJS which is a third-party open-source API that allows emails to be sent without having to implement a server for email provider ourselves. The application streamlines the reservation process, supports secure online payments via PayPal with a barcode generated after successful transactions, and includes a damage reporting feature for students. Barcode scanning for check-in/check-out, a notification system for sending announcements through Outlook, and tools to track and manage damaged equipment make it easier for administrators to manage inventory.

Document Summary

The **System Architecture** section provides a high-level overview of how the backend and frontend of our system work together, both statically and dynamically. This includes the interaction between the user interface, data fetching, local storage, and APIs. The static system architecture diagram shows how the application is organized into various components and how they interact, while the dynamic diagram walks through an example scenario, illustrating how the system performs its tasks.

The **Data Storage Design** section outlines the structure of our system's database using MongoDB, connecting to a remote NoSQL database to store equipment and user data. We carefully considered collection relationships to ensure model interactions are efficient and logical.

The **Component Design** section offers an overview of the system's components, presenting both static and dynamic diagrams. These diagrams are logically divided into three parts: Web View, Backend Logic, and Data.

The **UI Design** section showcases the user interface for our two-target user groups and describes the various screens in our application.

System Architecture

Introduction

The Film Equipment Inventory System is a web-based application developed for the School of Literature, Media, and Communication at Georgia Tech, designed to optimize the reservation and management of film equipment. This system facilitates seamless interaction between students and administrators by providing an efficient way to reserve, manage, and track equipment usage. The goal is to replace manual processes with an automated solution that ensures real-time availability updates, streamlined reservation management, and comprehensive inventory control.

The system empowers students to browse and reserve equipment for their class, select pick-up and return times, and make payments where applicable. Administrators, on the other hand, are equipped with tools to manage inventory, monitor equipment status, generate unique class codes, and oversee the reservation process. Additionally, the system provides detailed insights into equipment usage and damage reports to aid in effective resource management.

Our solution is built using the MERN stack (MongoDB, Express.js, React.js, and Node.js), a modern web development stack that ensures a scalable, dynamic, and high-performance application. The use of this stack enables seamless interaction between the front-end and back-end systems, ensuring real-time data synchronization, intuitive user interfaces, and an overall enhanced user experience. MongoDB provides a flexible and scalable database solution for storing equipment data, class codes, and reservations, while Express.js and Node.js form the backbone of the API, ensuring smooth server-side operations. React.js delivers a dynamic and responsive front-end, ensuring students and administrators can easily navigate and interact with the system.

This architecture guarantees a robust and maintainable system that can adapt to the evolving needs of the school while simplifying the reservation and management of film equipment for both students and professors.

Rationale

Starting off with the overall structure, for our technology stack choice we went for the MERN route (MongoDB, Express.js, React.js and Node.js). This was the best option for our group.

- MongoDB: MongoDB allows needed flexibility, which is helpful for utilizing the data we need in the LMC database.
- Express.js: Express.js allows for ease of use when it comes to APIs that help handle our inventory and equipment.
- React.js: A component-based interface makes the frontend of the project more easy to manage. The use of React.js allows for a dynamic and responsive web application that enhances user interactions and supports modular component design.

- Node.js: Node.js allows for server-side JavaScript execution which enables our codebase to easily unify across the entirety of the stack.

Architecture: We have a dedicated frontend, backend, and database. This separation makes maintenance very manageable depending on what type of changes need to occur.

Third Party APIs: The use of third-party APIs is needed for our project to operate. One of the APIs used is for automatic email sending from emailJS, which is a well-known API. APIs are needed for our student data collection and equipment listing within the LMC department. Precaution is taken when it comes to security concerns, such as incorporating encryption with or utilizing .env files.

The use of ENV files in our code allows us to securely store sensitive details avoiding coding sensitive information into source code. Encryption through means such as HTTPS in our project allows for secure data transmission to our database

Secure Payment: For secure payment when renting LMC equipment, we have chosen to use PayPal. PayPal is a secure payment gateway that can easily handle credit card transactions for our project, perfect for security when it comes to the payment info of the students and professors renting equipment.

Static Architecture

Diagram

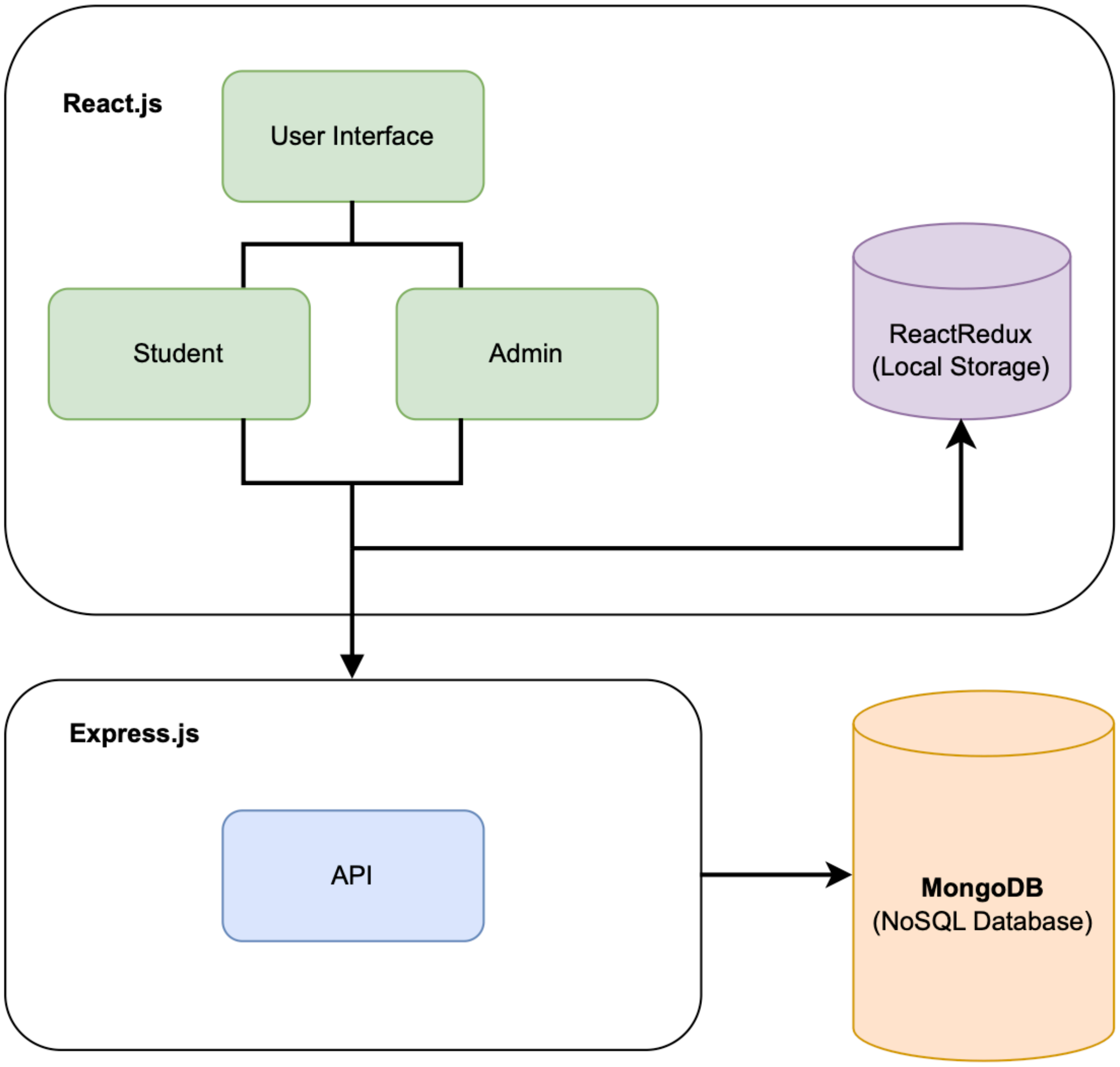


Figure 1: Static System Architecture Diagram

Description

The Film Equipment Inventory System is architected around three primary components: a **React.js Frontend**, an **Express.js Backend**, and a **MongoDB Database**. Each part plays a crucial role in ensuring a smooth, reliable, and user-friendly experience for students and administrators managing film equipment reservations.

The **React.js** serves as the user interface of the system, providing an intuitive and interactive experience for two primary user groups: students and administrators. The interface allows users to perform tasks related to film equipment reservation and management. ReactRedux is utilized for local state management within the frontend to ensure smooth data handling and quick access to user-specific information, storing essential information on the user's machine to reduce load on the API and enhance responsiveness.

The **Express.js** functions as the API layer of the system. It handles all incoming requests from the frontend, handles logic, and makes communication with the database easier. Express.js is selected due to its ease of use and its capacity to handle API requests effectively, which is crucial for managing multiple user interactions simultaneously. The backend is responsible for securely handling core functionalities required by the application.

The **MongoDB** acts as the database for the system, where storing all the necessary data, including user information, equipment details, and reservation records. MongoDB is a NoSQL database, is capable of handling a variety of data structures and has scalability for future expansion By connecting the backend and MongoDB, secure data retrieval and storage is ensured, and all interactions are logged and managed appropriately.

The architecture is made with the goal of modularity and scalability in mind. By clearly separating the frontend, backend, and database, each component can be developed and maintained independently. This separation of concerns not only enhances the system's flexibility but also allows for potential future changes, such as modifying the user interface, adjusting backend logic, or migrating to a different database technology without disrupting the overall application flow.

Dynamic Architecture

Diagram

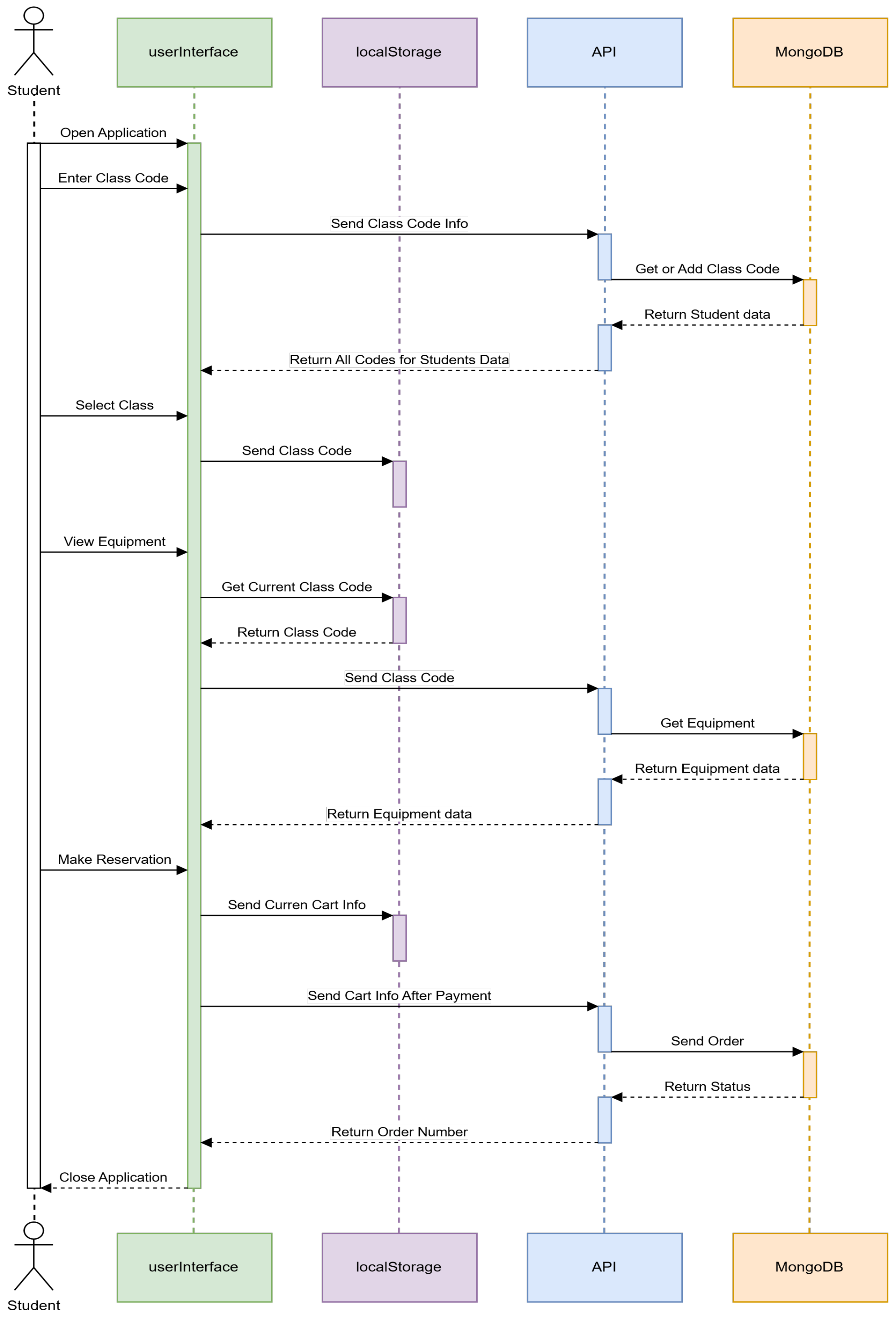


Figure 2: Dynamic System Architecture Diagram (Users: Students)

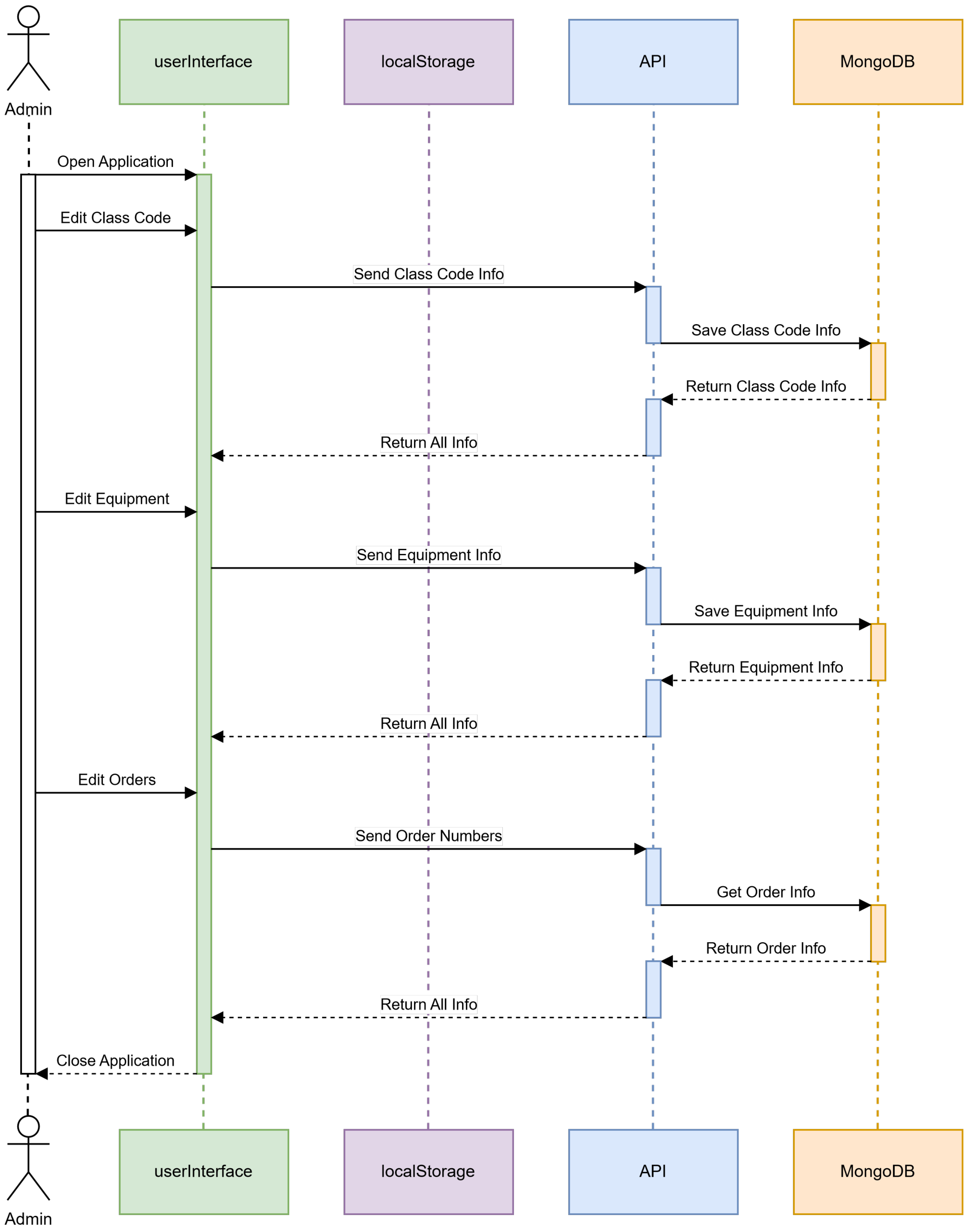


Figure 3: Dynamic System Architecture Diagram (Users: Administrators)

Description:

The system is designed to operate as a multi-tier architecture with distinct functionalities for different user roles. At the core of the system, a centralized database manages real-time information on available equipment, reservations, and checkouts. Students interact with the system through a user-friendly interface, where they can browse, reserve, and check out film equipment. The system dynamically updates the availability status based on these interactions.

For students, they begin by accessing the page and entering a class code. When the class code is entered on this page, a fetch request will be sent to the database through the fetch API to retrieve the equipment data related to the class. This data is returned and will be properly parsed through and displayed back to the user on the reservation page. On this reservation page, the equipment will be filtered based on their equipment type, such as cameras, lights, or bundle packages. When a student reserves a piece of equipment or package but does not complete the reservation and saves it for later, the equipment and its data will be temporarily stored in localStorage, which is uniquely stored on the student's local device. On the other hand, if the reservation is complete, then the reservation is stored into the system MongoDB database. The reservation will then display a message that it is confirmed, and the user has the option to close the application.

For admins, the process begins by accessing the application page. They have the ability to view current reservation orders that student users have created and submitted. Upon landing on the page to view all reservation orders, a fetch request is made to the database to retrieve the relevant data. This data is then returned, parsed, and properly displayed for the admin. If the admin applies any filter preferences while viewing the reservations, the current filter state is stored in localStorage. After completing the review, the admin can choose to close the application. In addition to managing student reservation orders, admins can view and modify the equipment database. When accessing the page for the equipment inventory, a fetch request retrieves all equipment data from the database, which is then parsed and displayed, showing relevant attributes for each piece of equipment. Admins can also view detailed information about any of the classes that require reservations for film equipment, including details like the professor assigned to each class and any equipment that is inclusive to a specific class. All this information is stored in the database, and any modifications made by admins are saved, ensuring that data is accurately retrieved the next time it is accessed.

Component Design

Introduction

This section provides an overview of the **static component design** and **dynamic component design** of our system architecture. These diagrams are color-coded synonymously with other diagrams throughout the document. The application consists of four main components: the web application interface, the API, local storage, and the backend. In the diagram, green represents the interface of our application, purple represents the local storage, blue represents the API, and orange represents the backend.

The **static component design diagram** is represented through a component diagram (Figure 4), which highlights the core components and their interactions, focusing on the relationships between the frontend and backend functionalities. It showcases distinct user roles - Student Role and Admin Role - along with their respective capabilities within the system. Each role interfaces with shared and specific components with a clear hierarchical flow that leads to payment processing and data management.

The **dynamic component design diagram** is represented through a sequence diagram (Figure 6), which captures the workflow for an admin's search within the system. This diagram illustrates how user interactions progress, from initiating a search for class codes to fetching data from the database. It details backend processing, including interactions with MongoDB and the display of associated class elements.

Static

Diagram

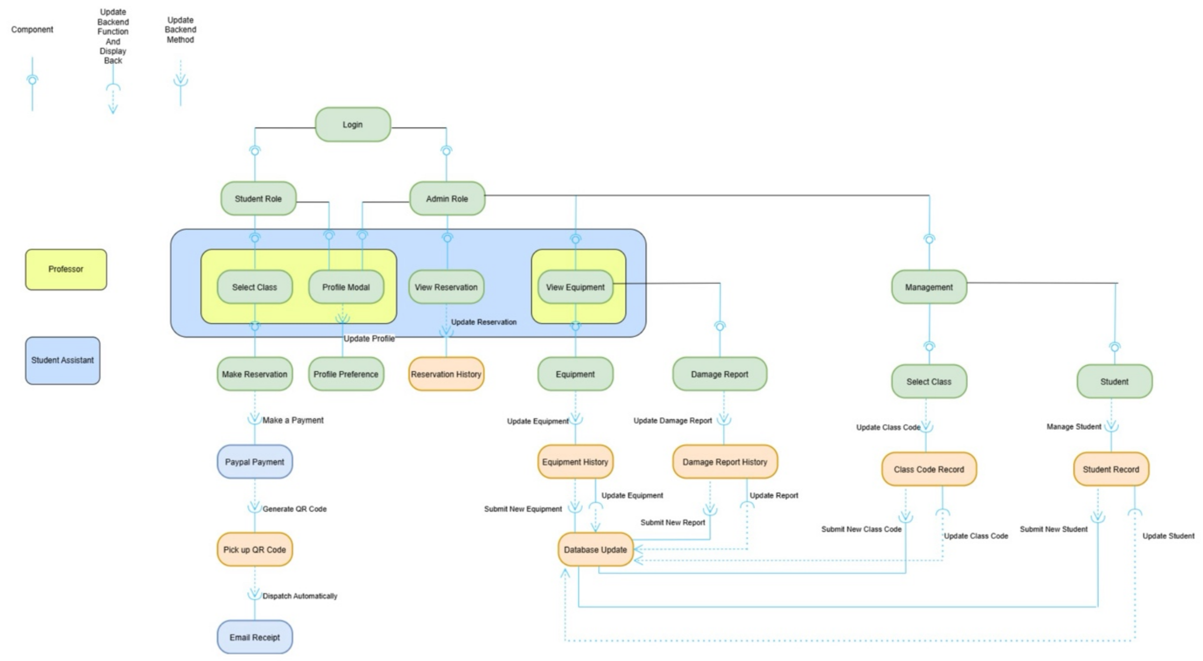


Figure 4 - Static Component Design Diagram

Description

The Figure 4 component diagram above illustrates the structure of our Equipment Reservation System, depicting the relationships between frontend and backend components. On the front end, there are distinct user roles, including Student Role and Admin Role, which manage user interactions.

The **Student Role** component allows students to select a class, make reservations, and process payments, while the **Admin Role** component enables admins to manage equipment, view damage reports, and update class codes. The **Profile Preference** component is shared between both roles for updating personal preferences. Back-end components like **Database Update** handle real-time updates of reservation data, equipment status, and class codes, ensuring data consistency. The system follows a hierarchical interaction, with **Select Class** and **Make Reservation** components facilitating user actions that eventually integrate with **PayPal Payment** or transaction handling. Admins manage equipment and damage reports through **View Equipment** and **View Damage Report**, both of which update the system through the **Database Update** component. Relationships between front-end components are represented by solid lines, showing aggregation and usage dependencies, while back-end components maintain key connections for updating and managing data.

Dynamic

Diagram

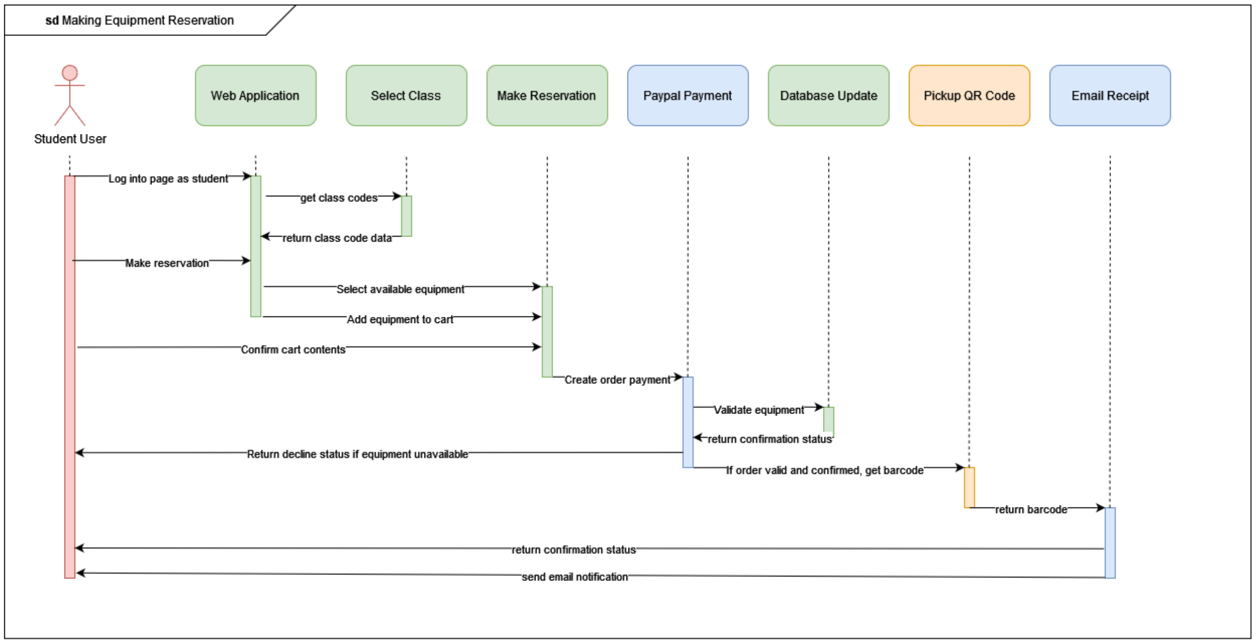


Figure 5 - Runtime Interactions Diagram

Description

For the Figure 5 Runtime Interactions Diagram above, it illustrates the flow of how the user will interact with the application, and how information will be transferred between the two. In this diagram, the user will perform the action of making a reservation.

They will begin by logging into the system, and the system will return available and registered class codes for the user, which the user will decide which to use. After confirming a class code and being directed to the reservation page, the system will request for equipment that are available to register from the database and return equipment back to the user that they can reserve for. Once the user has selected the equipment that they would like to reserve, an order will be validated after payment and updating the equipment database for the items, such as the equipment status, pick up and return date, and who currently has it reserved. For the final step, an order status will be sent out to the email of the user, and a barcode will be generated for the user to scan in person at pickup.

Diagram

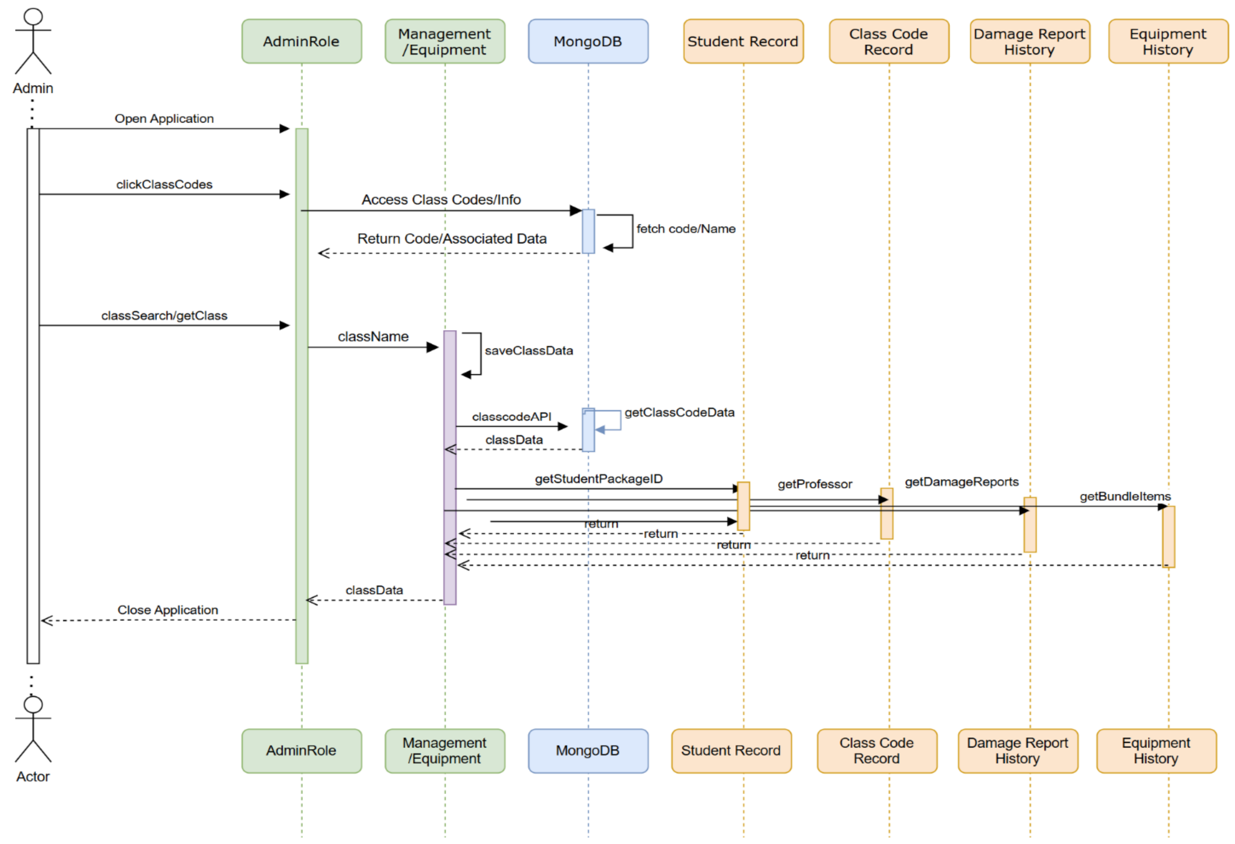


Figure 6 - Dynamic Component Design Diagram

Description

The Figure 6 sequence diagram above illustrates the flow for when an admin decides to search for a specific class in the database and its associated numbers/names/reports that are assigned to it. This specific use involved a user interface (adminRole), a search bar function(seen in management and equipment), student record, class code record, damage report history, and equipment history.

The user starts the interaction by accessing the class codes page. Here, is displayed all the classes within the database. The user can then select the search bar, and search for a specific class by class name. Once a name is searched, it is then fetched to the MongoDB server for certain specifics of the class.

Upon searching and selection of a class, multiple elements associated with that class pop up in a modal. Some of these include damage reports, professor of the class, and the equipment and students associated with the class if the add button is selected as well. The admin can search for a specific class by name and get the associated info.

Data Design

Introduction

We manage data storage in our application using a combination of MongoDB and browser memory through React Redux. We use a NoSQL database, MongoDB, which employs a Key-Value pair data schema to store permanent user data, such as order information, damage reports, and other critical interactions.

As the diagram shown below, it represents how we use localStorage. The purple filled-in square represents redux functions including selectors, which we use to store, update, or get data from Redux Store.

Additionally, certain data, such as cart items, userId and payment details are stored temporarily in the browser's memory using React Redux. This approach enables us to avoid props drilling and facilitates easy access to data across the application without requiring constant database calls.

Database Use Subsection

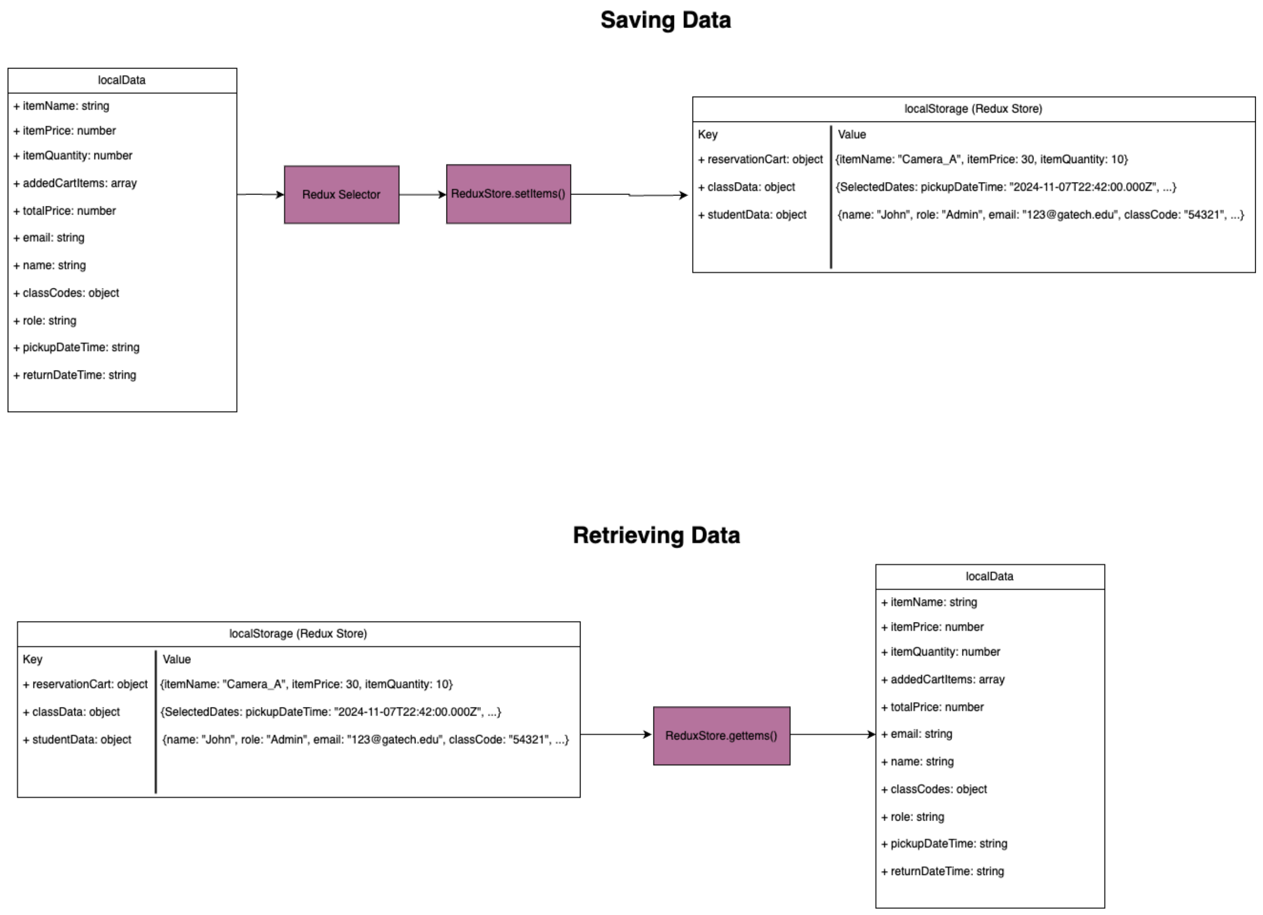


Figure 7 - Database Use Subsection Diagram

File Use Subsection

In our system, we handle the storage and management of image files associated with damage reports. These images provide a visual reference for the reports and are critical for the system's operation. The following outlines how these files are stored, formatted, and handled.

File Format:

- Type: Image files.
- Allowed Formats: We support standard image formats such as:
    - JPEG (image/jpeg)
    - PNG (image/png)
    - JPG (image/jpg)

The image files are uploaded by the user in the frontend and passed to the backend as part of a form submission. The images are stored as Base64-encoded strings within the database for simplicity and portability.

Data Exchange Subsection

The protocol we use for data transfer is HTTP/HTTPS between our web browser and our server/MongoDB database. HTTP/HTTPS allows us to send REST API requests such as GET, POST, PUT, and DELETE Requests. This means objects like order, items, students and more can get altered, created or deleted.Some examples of our requests are:

```
await axios.put(`${BACKEND_URL}/students/${email}/remove-classcode`
```

```
await axios.put(`${BACKEND_URL}/api/bundle-items/${updatedBundleItem.bundleId}`
```

```
const res = await axios.put(`${BACKEND_URL}/api/damage-reports/${id}`
```

In terms of data format to access and request from the server, we use JSON (JavaScript Object Notation) to help store variables for certain items, For example, an Order has its own OrderNumber, studentName, items, and more stored as variables to itself in JSON format. We also have images embedded as variables in our Damage report object.

Data Security

Because this is a webapp, security is important for users when inputting sensitive information. Even though this extends only as far as the PayPal system and their own secure service that we use, other details such as name and email are the next in line that need to be secured.

i. Data encryption is certainly needed for user login information, such as email, username, and password. As such, none of these should be stored directly into a database as plain text; common password encryption algorithms like PBKDF2 or PKCS#5. Furthermore, authenticating a user's current session while logged in is only stored on client side and not in plain text, so retrieving a session token or the like is only possible if there is physical access to the device.

ii. The data exchange channel should be secured and use https, as it encrypts all data exchanged between the user and the website.

iii. User login is required to access all features in the application, with user authentication being a top priority. We use a token-based authentication system (JWT) to manage sessions, securely storing tokens to prevent unauthorized access. For added security, tokens are only stored in secure locations, such as HTTP-only cookies or in-memory storage, reducing the risk of token theft. Our session management includes a timeout policy where tokens expire 8 hours after login, ensuring sessions do not remain active indefinitely. Additionally, our next step is to integrate the GT Login system to further enhance security, as it provides Multi-Factor Authentication (MFA).

iv. For transactions, we rely on the PayPal library, which provides a secure and established interface for processing payments. This library creates a direct, secure connection between the user and PayPal’s system, ensuring that sensitive financial data, such as bank or credit card details, is handled in a secure manner without direct access by our application.

v. Usernames, emails, and other sensitive details are solely for internal use within our application and are not shared or exposed externally. To address privacy, we implement a data retention policy that removes inactive user information after two years, helping maintain user privacy and compliance with data protection practices.

User Interface Design

Introduction

In this section, we will introduce and present the user interface of our website. We will specifically demonstrate how users can interact with the website's design and explain how we designed it to be more intuitive, following the ten usability heuristics for user interface design. This website is primarily built for two main user groups: administrators and students. Additionally, within the administrator group, there are three sub-groups: the owner of the website, student assistants, and professors. The relationship among these sub-groups is as follows: the website owner has full administrative access with the ability to see all features and functionalities, while student assistants and professors can only access certain features and functionalities based on their roles. For the sake of convenience, we will introduce this section based on the two main user groups, administrators and students. Finally, we will illustrate the major user interface screens specific to professors and student assistants.

Login System

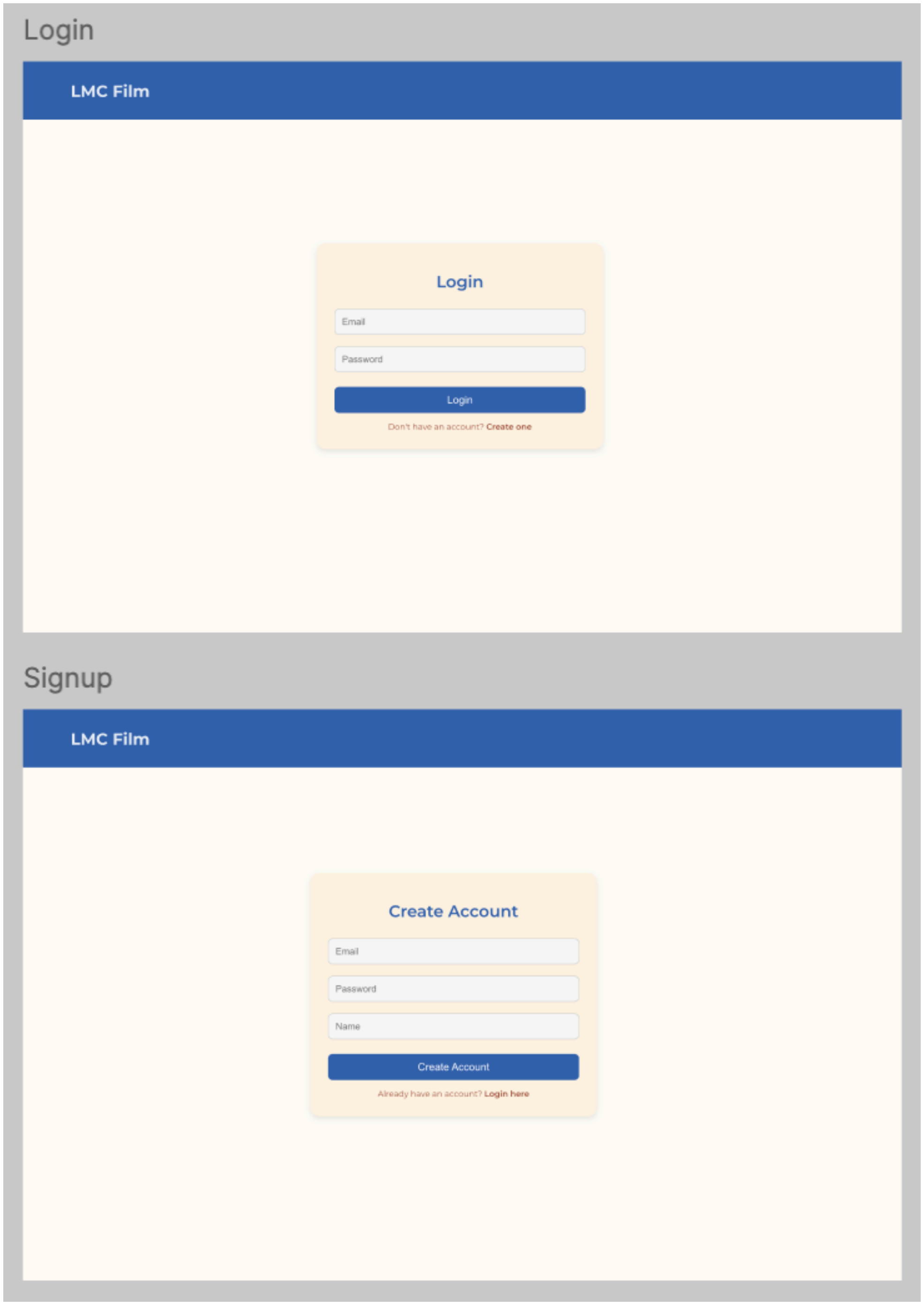
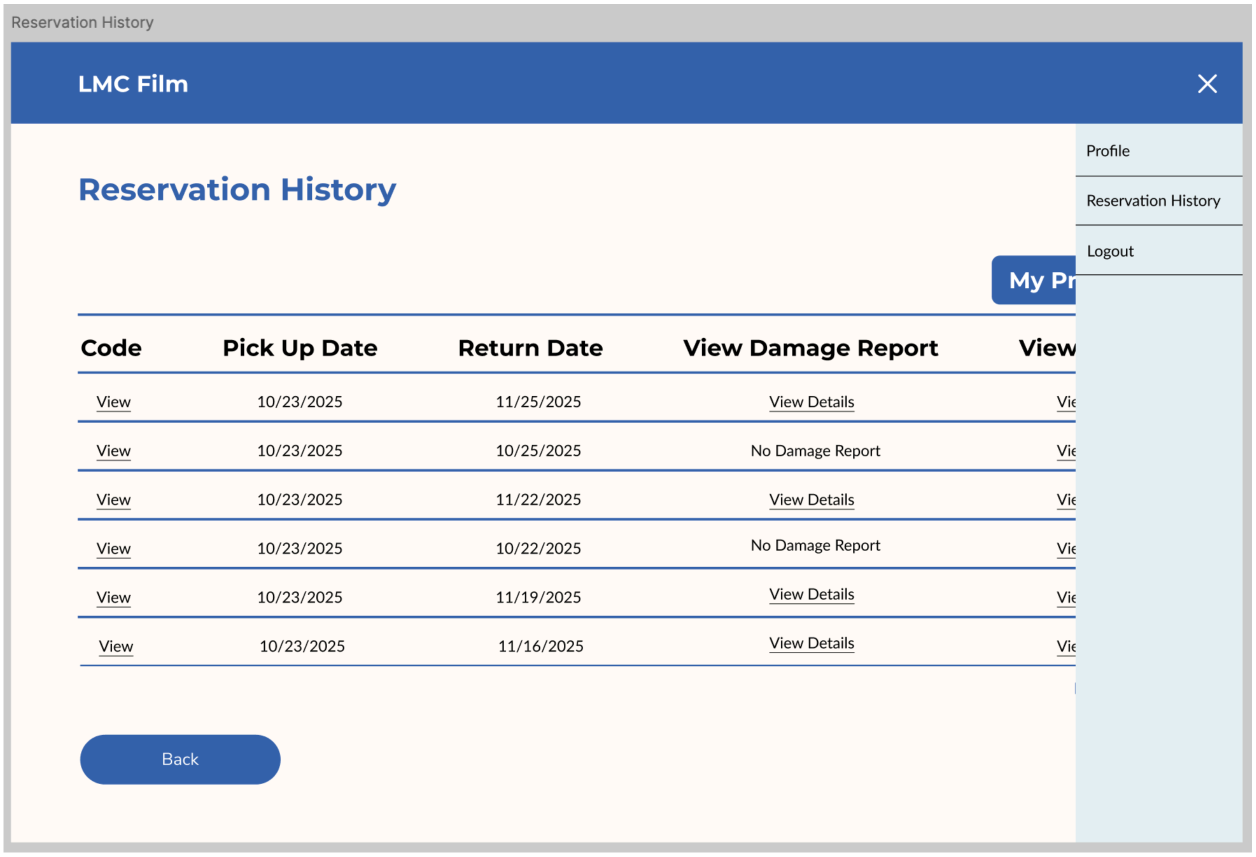


Figure 8 – Login & Signup Page

This is the login and signup page (Figure 8) for the system. First-time users can create an account by entering their email address, password, and name during signup. Returning users can log in using their email and password. Upon login, the system automatically assigns the default role of "student" to first-time users. Administrators can later update the role to "student assistant" or "professor" as needed. This setup ensures the system effectively distinguishes between student and non-student logins.

Navigation Bar

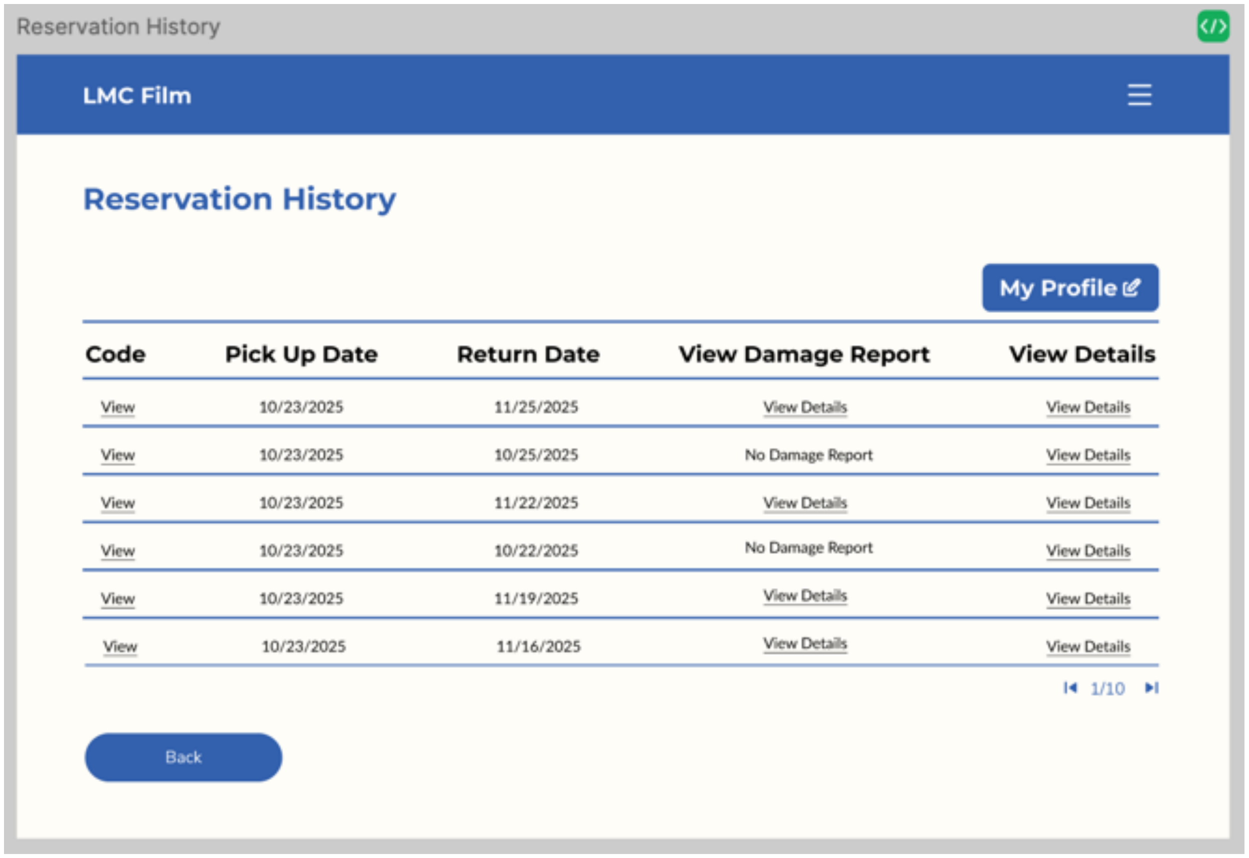


Reservation History

Code	Pick Up Date	Return Date	View Damage Report	View
View	10/23/2025	11/25/2025	View Details	Vie
View	10/23/2025	10/25/2025	No Damage Report	Vie
View	10/23/2025	11/22/2025	View Details	Vie
View	10/23/2025	10/22/2025	No Damage Report	Vie
View	10/23/2025	11/19/2025	View Details	Vie
View	10/23/2025	11/16/2025	View Details	Vie

Figure 9 - Screenshot of Navigation Bar

This is the navigation bar accessible to all users across the website (Figure 9). Users can open the navigation bar from any page by clicking the hamburger icon. Within the navigation bar, users can view their profile information, access their reservation history, and log out of the website. When users hover over any section, a visual indicator highlights the active section. This design adheres to the consistency and standards heuristic by maintaining a familiar and predictable interface element throughout the application. Additionally, the hover feature enhances usability and aligns with heuristic evaluation principles.


Reservation History

LMC Film

Reservation History

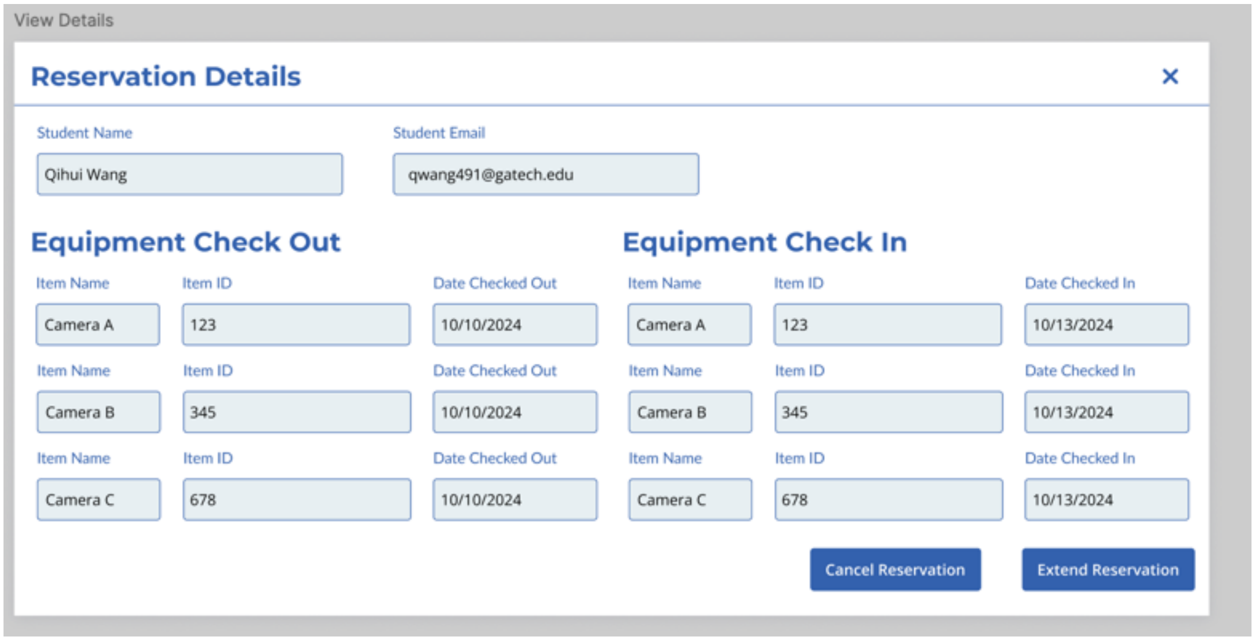
My Profile

Code	Pick Up Date	Return Date	View Damage Report	View Details
View	10/23/2025	11/25/2025	View Details	View Details
View	10/23/2025	10/25/2025	No Damage Report	View Details
View	10/23/2025	11/22/2025	View Details	View Details
View	10/23/2025	10/22/2025	No Damage Report	View Details
View	10/23/2025	11/19/2025	View Details	View Details
View	10/23/2025	11/16/2025	View Details	View Details

1/10

Back

Figure 10 - Reservation History Page


View Details

Reservation Details

Student Name: Qihui Wang

Student Email: qwang491@gatech.edu

Equipment Check Out

Item Name	Item ID	Date Checked Out
Camera A	123	10/10/2024
Camera B	345	10/10/2024
Camera C	678	10/10/2024

Equipment Check In

Item Name	Item ID	Date Checked In
Camera A	123	10/13/2024
Camera B	345	10/13/2024
Camera C	678	10/13/2024

Cancel Reservation

Extend Reservation

Figure 11 - Reservation Details Modal

The first screenshot (Figure 10) illustrates what users see after clicking on the **Reservation History** tab in the navigation bar. This section stores all reservation information, allowing users to view their past and current reservations. They can access barcodes, review damage reports, and see detailed reservation information for each order. The second screenshot (Figure 11) shows the interface that appears when users click the **View Details** button on a specific order. Here, users can view additional details, such as items reserved, corresponding item IDs, and pick-up and return dates. This design aligns with usability heuristics by ensuring that users do not need to remember information from other parts of the interface. All reservation-related details are conveniently stored here and easily accessible through the navigation bar.

Administrator

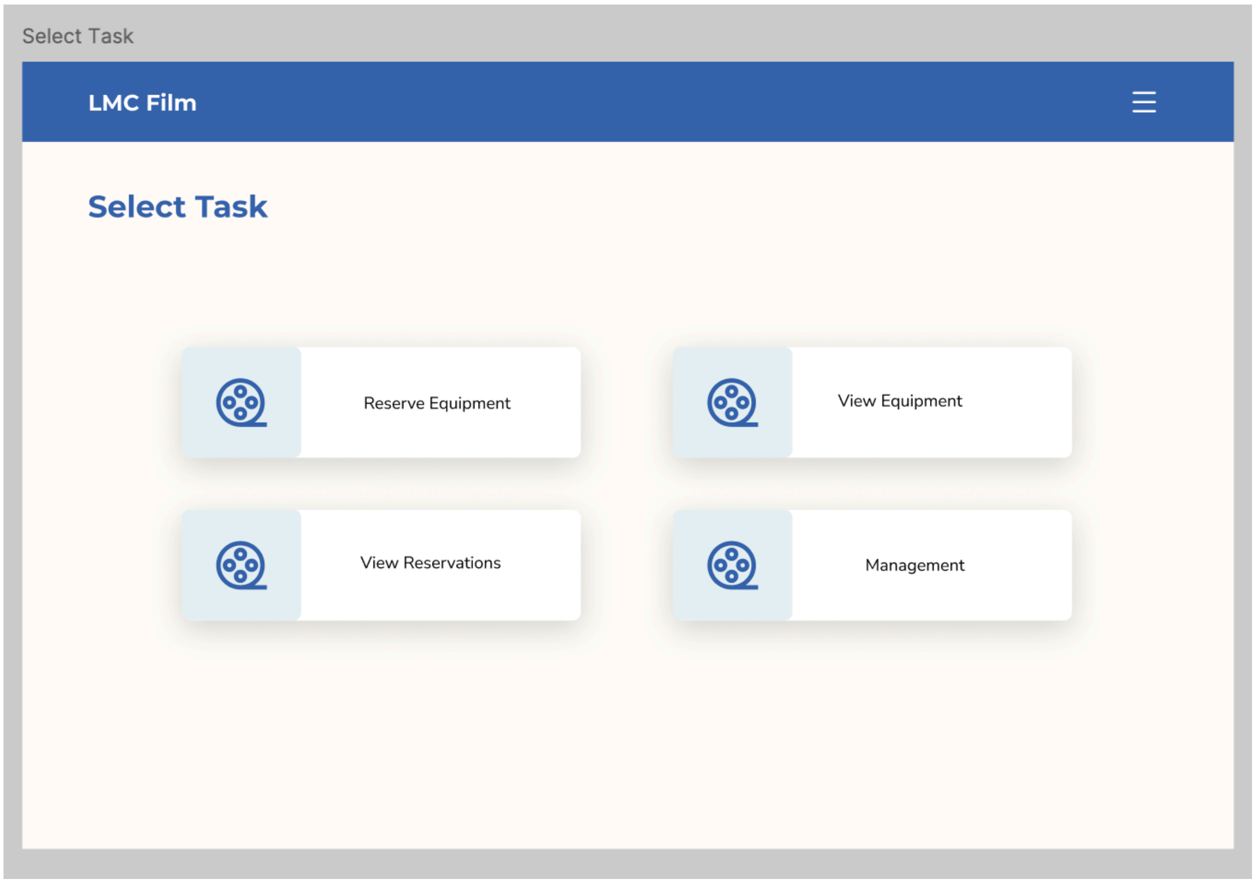


Figure 12 - Select Task Page

This is the home page for the administrator, where they can select various tasks. The **Reserve Equipment** tab allows users to make film reservations, while the **View Equipment** tab enables them to browse all equipment in the inventory. The **View Reservations** tab facilitates checking students in and out during equipment pick-up or return. The **Management** section handles system tasks, such as assigning user roles (e.g., student assistants or professors) and generating class codes for classes or clubs needing film equipment for projects. This home page aligns with a usability heuristic by guiding users in task completion through clear commands.

Throughout the website, a title at the top left indicates the current page, helping users stay informed about their location in the system. For instance, on this homepage, the title reads **Select Task**; while navigating to **View Equipment** changes the title accordingly. This design reinforces the heuristic of keeping users informed about their status in the application.

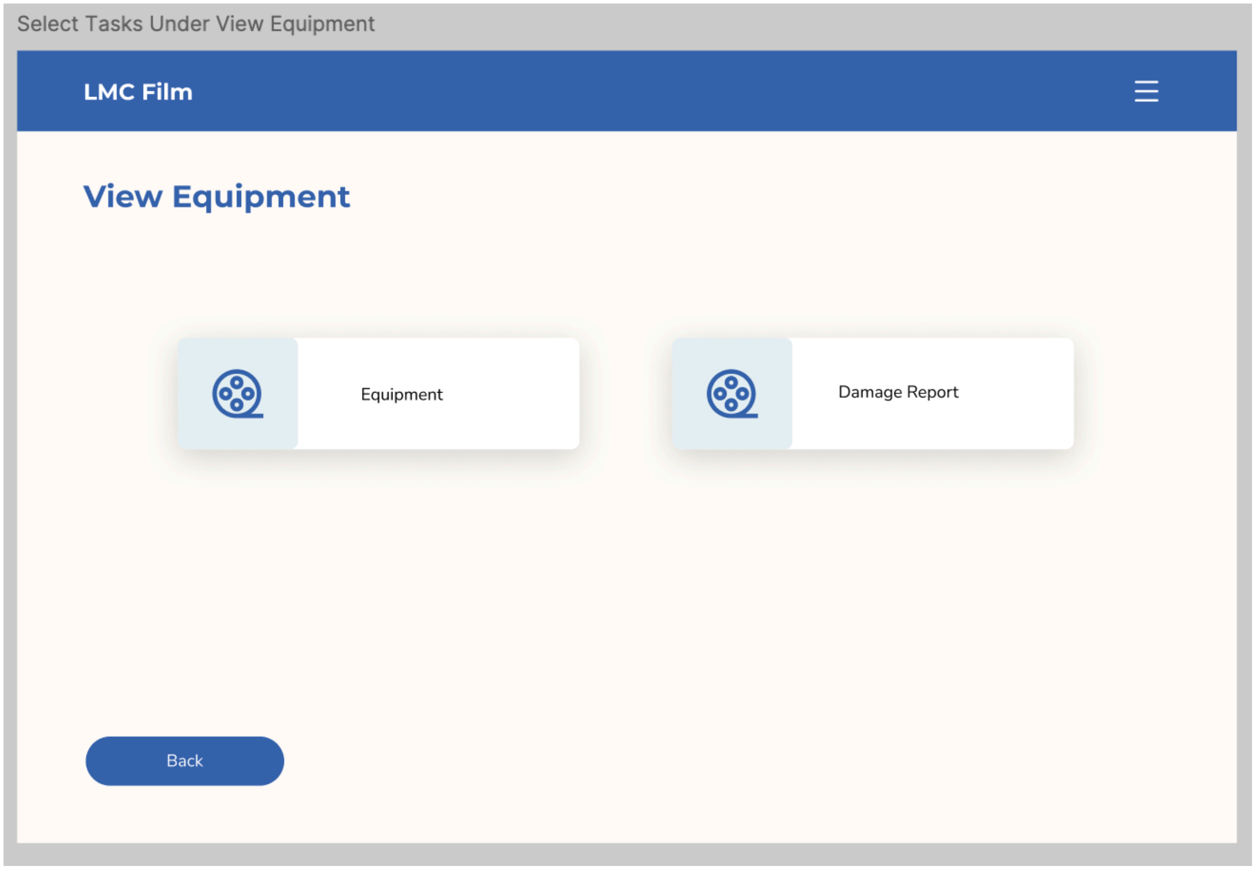
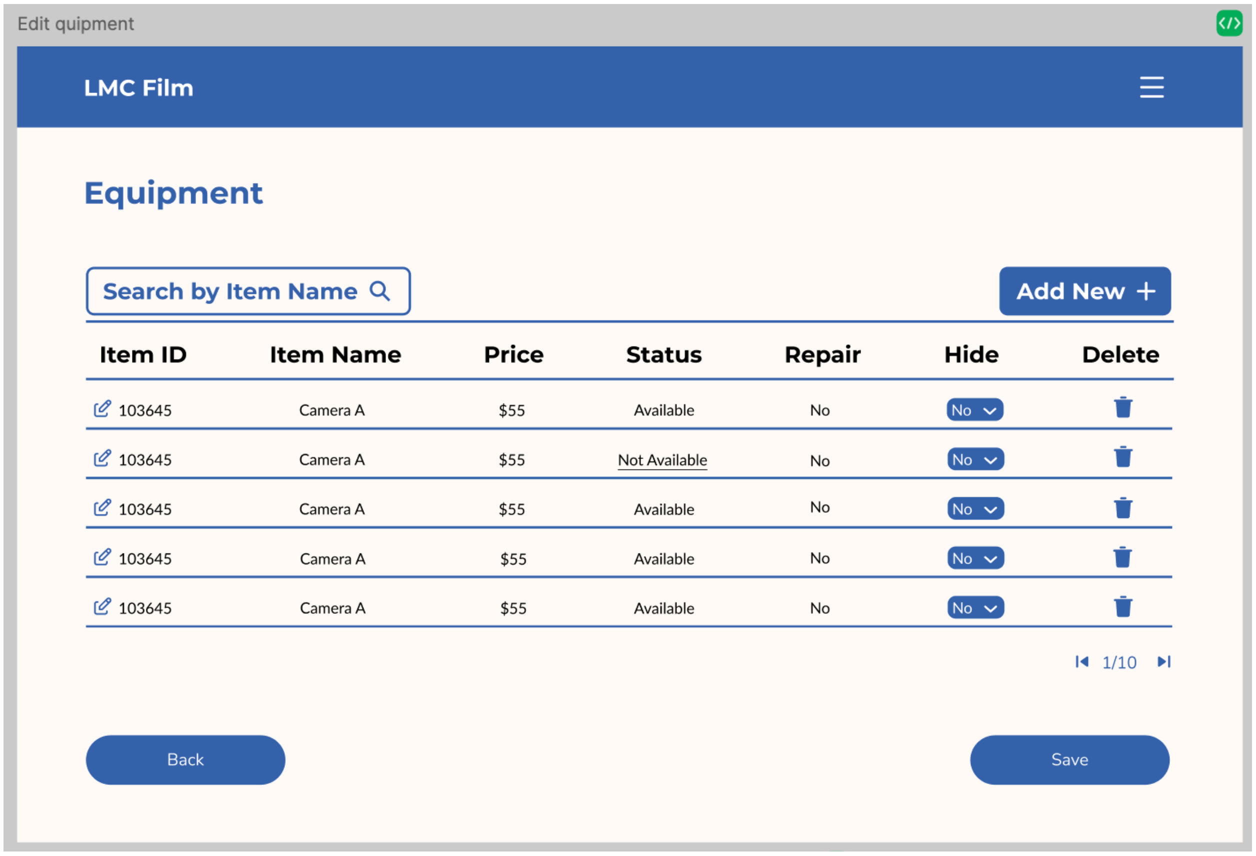


Figure 13 - View Equipment Page

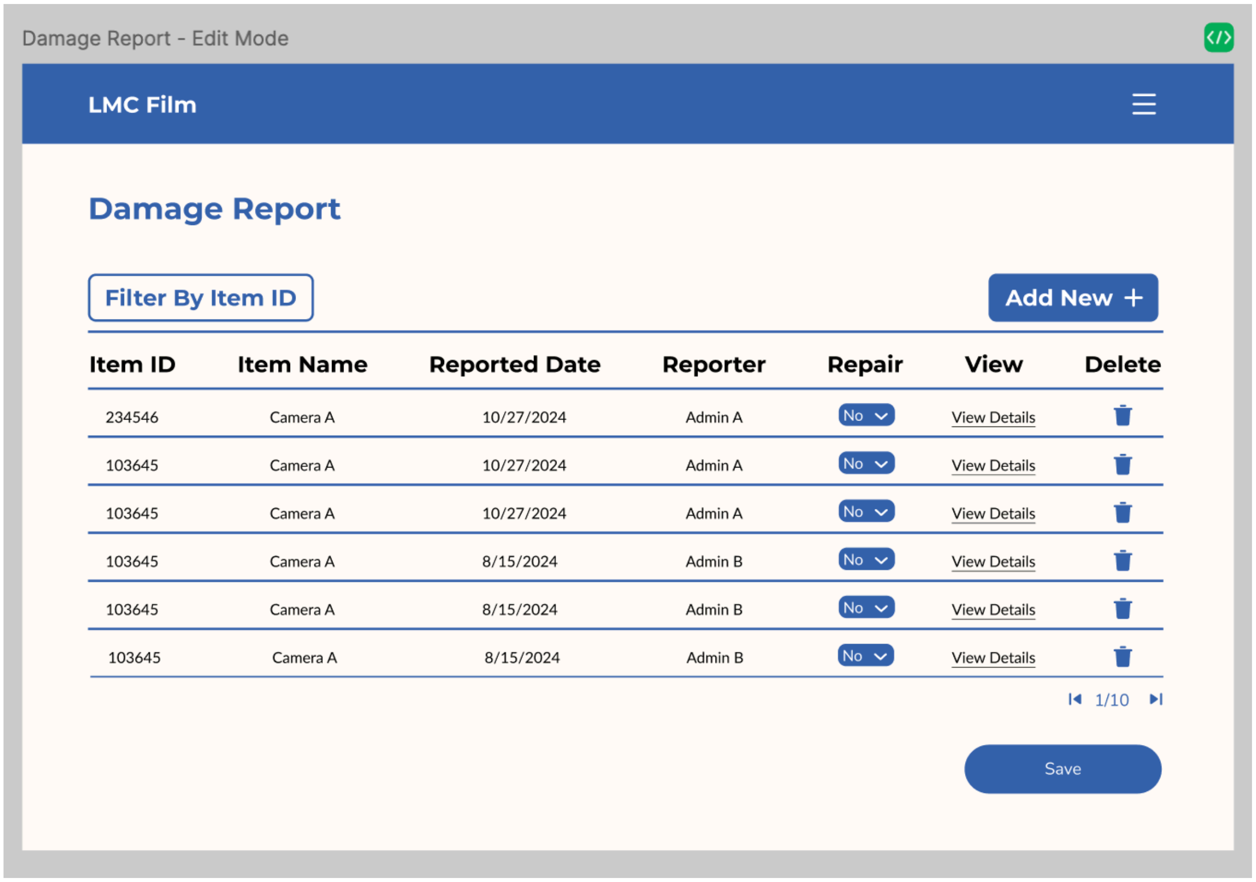
This is the page administrators see after selecting the **View Equipment** tab. Here, administrators can choose to view all equipment in the inventory by selecting the **Equipment** tab or create a damage report for any equipment by selecting the **Damage Report** tab.



Item ID	Item Name	Price	Status	Repair	Hide	Delete
103645	Camera A	$55	Available	No	No	
103645	Camera A	$55	Not Available	No	No	
103645	Camera A	$55	Available	No	No	
103645	Camera A	$55	Available	No	No	
103645	Camera A	$55	Available	No	No	

Figure 14 - Equipment Page

This is the **Equipment** page user interface. Here, administrators can add new equipment to the database by accessing the **Add New** button on the top right corner. They can view detailed information such as item name, price, and availability status. Additionally, they can edit any existing equipment and proceed with delete actions as needed.



Damage Report - Edit Mode

LMC Film

Damage Report

Filter By Item ID

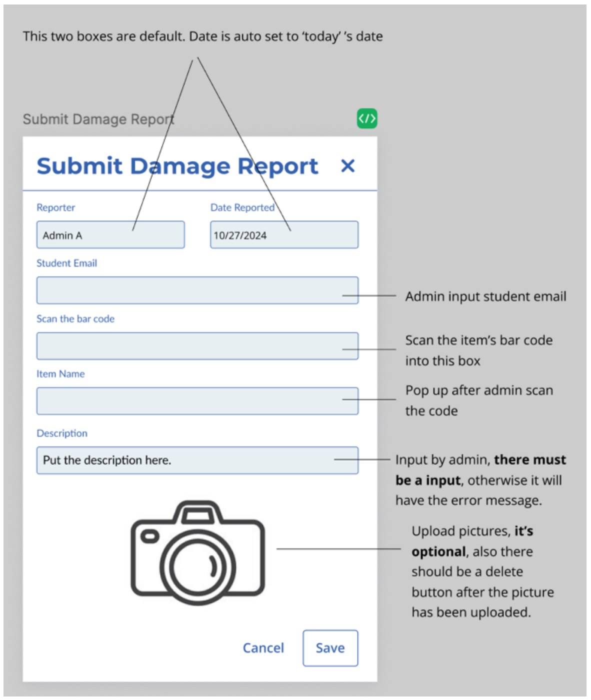
Add New +

Item ID	Item Name	Reported Date	Reporter	Repair	View	Delete
234546	Camera A	10/27/2024	Admin A	No	View Details	🗑
103645	Camera A	10/27/2024	Admin A	No	View Details	🗑
103645	Camera A	10/27/2024	Admin A	No	View Details	🗑
103645	Camera A	8/15/2024	Admin B	No	View Details	🗑
103645	Camera A	8/15/2024	Admin B	No	View Details	🗑
103645	Camera A	8/15/2024	Admin B	No	View Details	🗑

1/10

Save

Figure 15 - Damage Report Page



This two boxes are default. Date is auto set to 'today' 's date

Submit Damage Report

Submit Damage Report ×

Reporter: Admin A

Date Reported: 10/27/2024

Student Email

Scan the bar code

Item Name

Description: Put the description here.

Cancel Save

Admin input student email

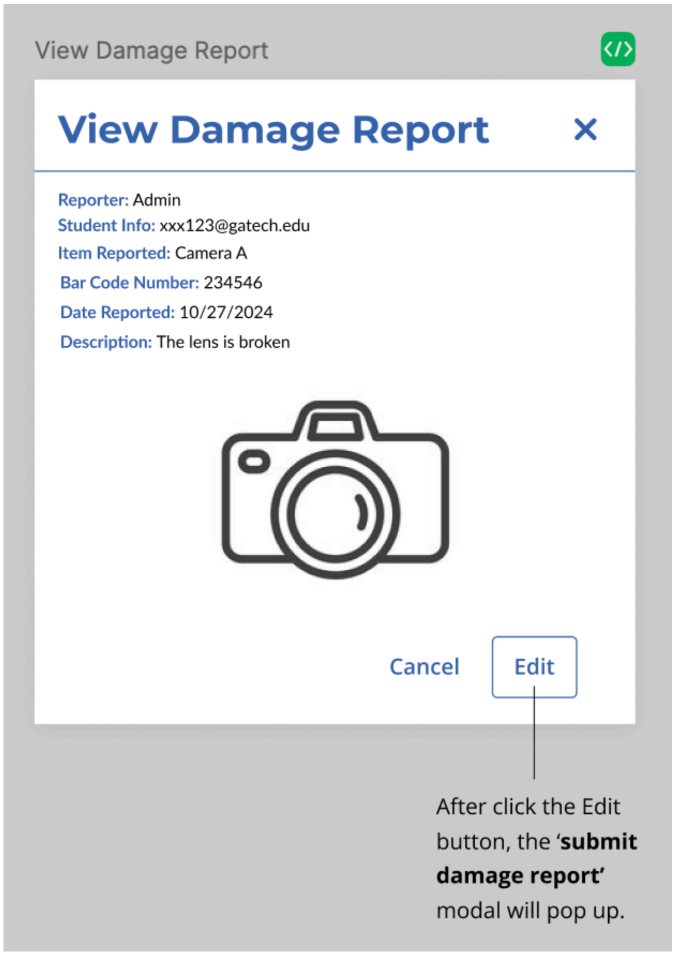
Scan the item's bar code into this box

Pop up after admin scan the code

Input by admin, **there must be a input**, otherwise it will have the error message.

Upload pictures, **it's optional**, also there should be a delete button after the picture has been uploaded.

Figure 16 - Submit Damage Report Modal



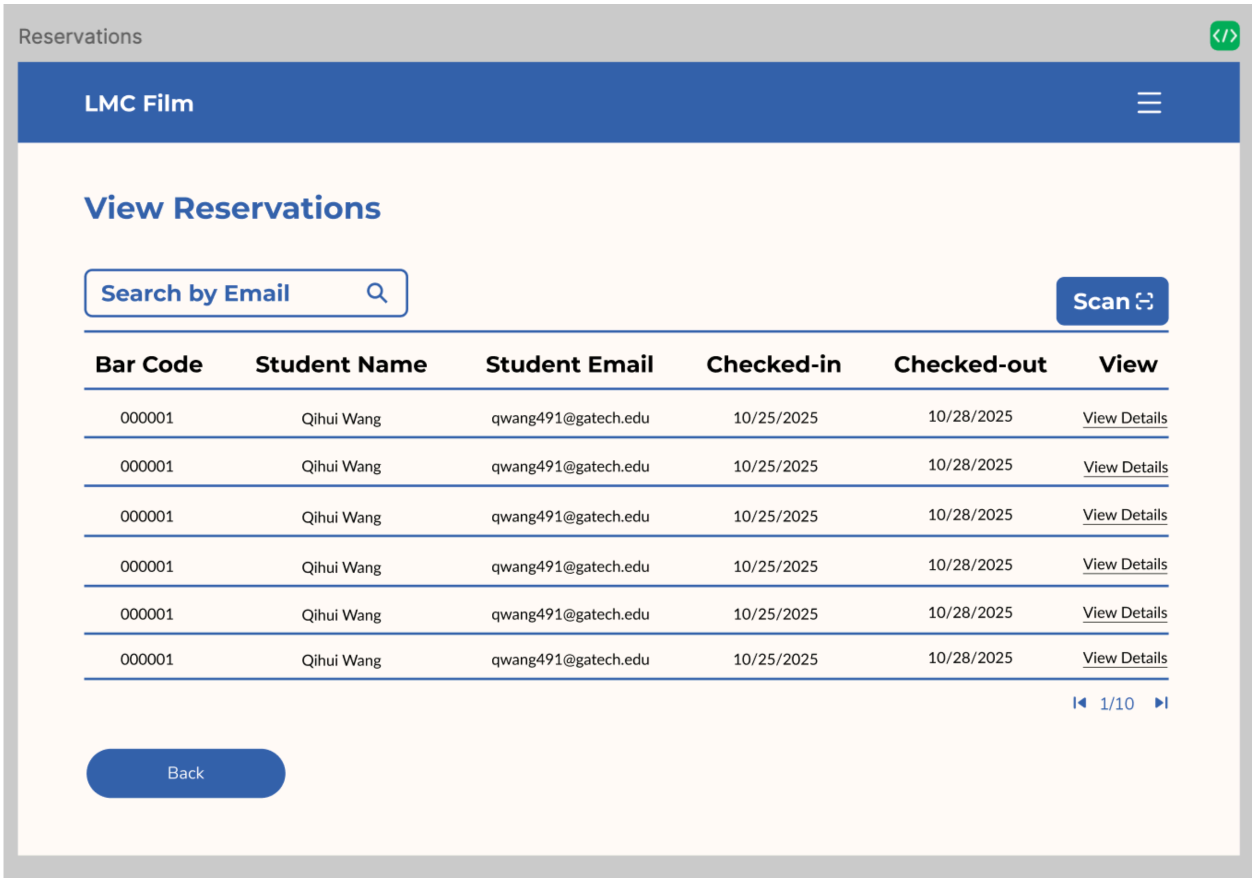
Reporter: Admin
Student Info: xxx123@gatech.edu
Item Reported: Camera A
Bar Code Number: 234546
Date Reported: 10/27/2024
Description: The lens is broken

After click the Edit button, the '**submit damage report'** modal will pop up.

Figure 17 - View Damage Report Modal

This is the user interface for the **Damage Report** page. Administrators can add new reports by clicking the **Add New** button in the top right corner, as shown in Figure 15. They also have the option to mark equipment as under repair, and the status will be simultaneously updated on the **Equipment** page. Additionally, administrators can view the details of damage reports in a popup modal, as shown in Figure 16, and edit them as needed (Figure 17). They can also delete reports or search for specific items by their Item ID.

The pop-up modal is designed to prevent interfaces from containing irrelevant information. For example, the **Damage Report** page only displays information specific to damage reports. If an admin needs to add a new damage report, a pop-up modal appears with fields for all relevant details about the new report. This approach aligns with usability heuristics, ensuring that interfaces do not include irrelevant or infrequently needed information. Each extra unit of information in an interface competes with the relevant content and reduces its visibility. Different pop-up modals are used for different tasks to maintain clarity and focus by displaying only pertinent information for each action.


Reservations

LMC Film

View Reservations

Search by Email

Scan

Bar Code	Student Name	Student Email	Checked-in	Checked-out	View
000001	Qihui Wang	qwang491@gatech.edu	10/25/2025	10/28/2025	View Details
000001	Qihui Wang	qwang491@gatech.edu	10/25/2025	10/28/2025	View Details
000001	Qihui Wang	qwang491@gatech.edu	10/25/2025	10/28/2025	View Details
000001	Qihui Wang	qwang491@gatech.edu	10/25/2025	10/28/2025	View Details
000001	Qihui Wang	qwang491@gatech.edu	10/25/2025	10/28/2025	View Details
000001	Qihui Wang	qwang491@gatech.edu	10/25/2025	10/28/2025	View Details

1/10

Back

Figure 18 - View Reservation Page

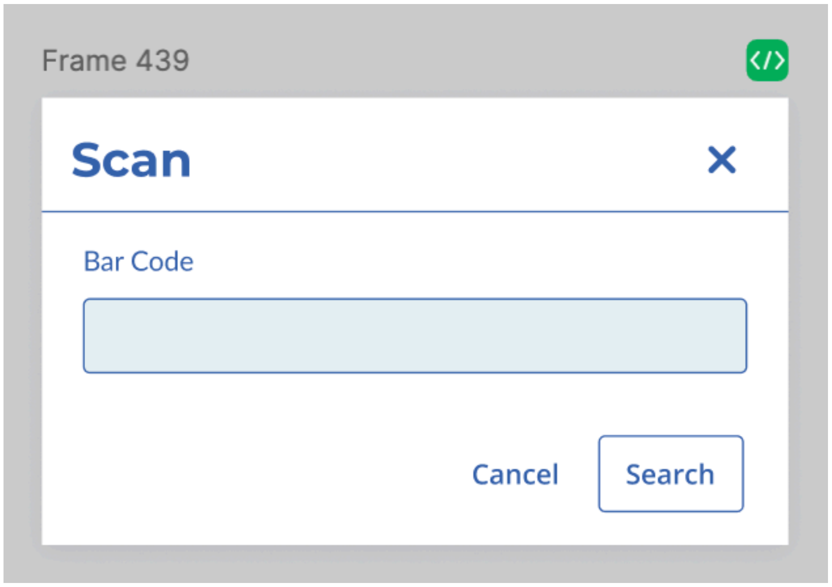


Figure 19 - Scan Modal

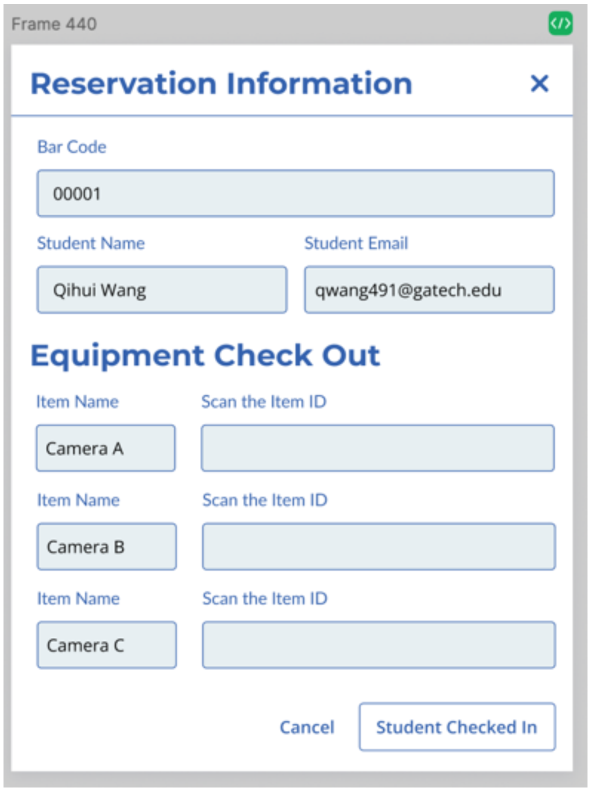


Figure 20 - Reservation Information (Pick Up Equipment) Modal

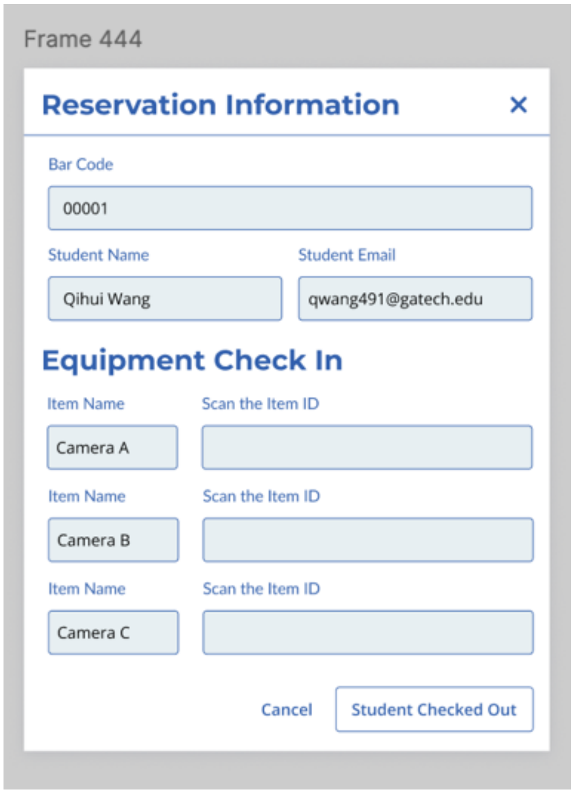
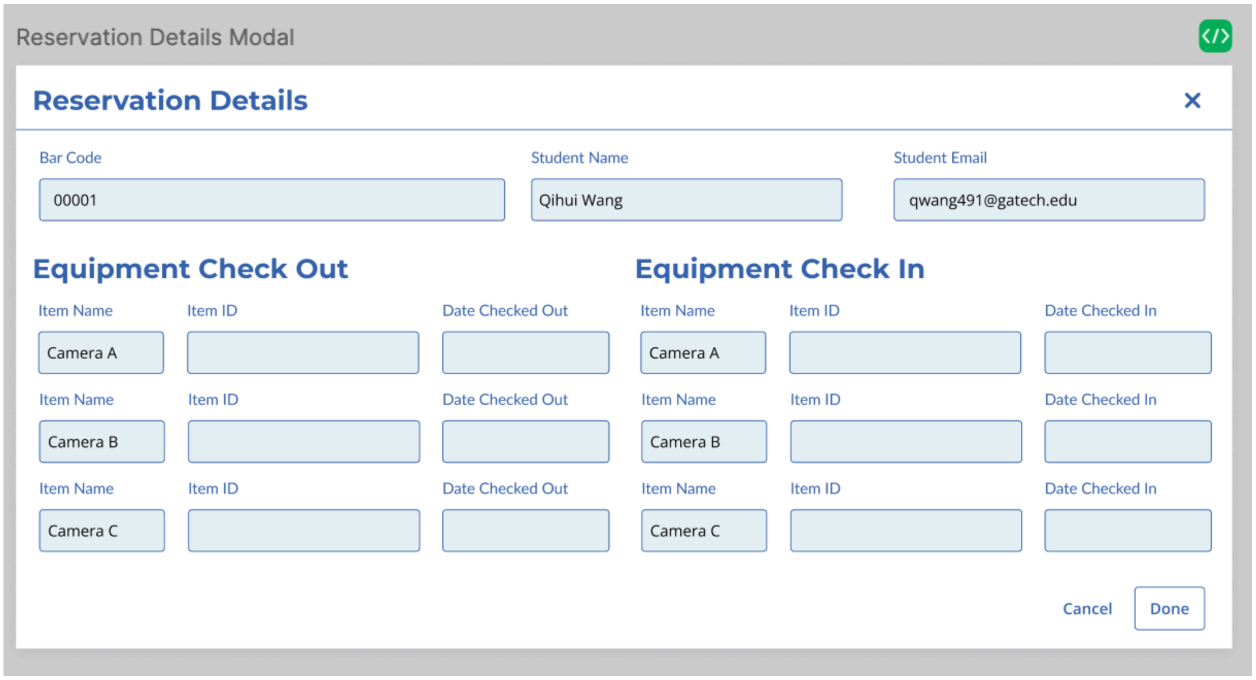


Figure 21 - Reservation Information (Return Equipment) Modal


Reservation Details Modal

Reservation Details

Bar Code	Student Name	Student Email
00001	Qihui Wang	qwang491@gatech.edu

Equipment Check Out

Item Name	Item ID	Date Checked Out
Camera A		
Camera B		
Camera C		

Equipment Check In

Item Name	Item ID	Date Checked In
Camera A		
Camera B		
Camera C		

Figure 22 - Reservation Details Modal

This is the user interface of **View Reservations** (Figure 18) after the administrator selects it from the home page. Here, administrators can check students in when they pick up equipment and check them out when they return it. The process is streamlined by clicking the **Scan** button in the top right. A barcode is generated for each reservation, which administrators scan when students arrive (Figure 19). Upon scanning, the reserved item's name appears on the screen, simplifying the process. Administrators need to scan the **barcode (Item ID)** attached to each piece of equipment to ensure accurate tracking, with today's date automatically recorded as **Date Checked Out** or **Date Checked In** (Figures 20 & 21).

If any issues arise with the scanner, administrators can locate reservations by searching the student's email. Detailed information is accessible via the **View Details** button (Figure 18), which shows the specific items reserved, exact equipment checked out, and timestamps for each pick-up and return (Figure 22).

The terms **Equipment Checked Out** and **Student Checked In** appear in the pop-up modal when students pick up equipment, and **Equipment Checked In** and **Student Checked Out** appear when students return it, aligning with usability heuristics by using language intuitive to administrators. When students check in to pick up equipment, it means the equipment is being checked out from the admin's perspective. This phrasing matches the terminology administrators would expect, enhancing clarity and ease of use in managing equipment and student interactions.

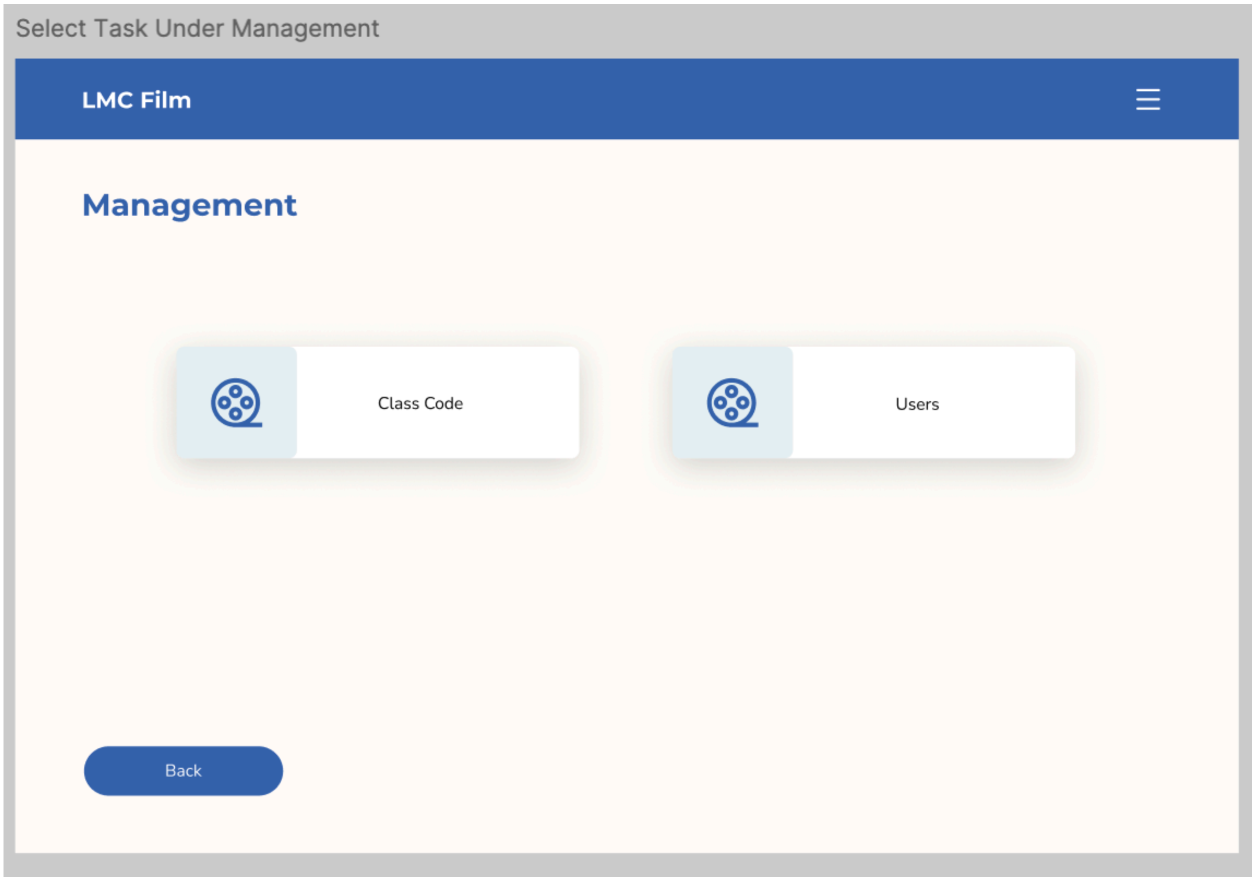
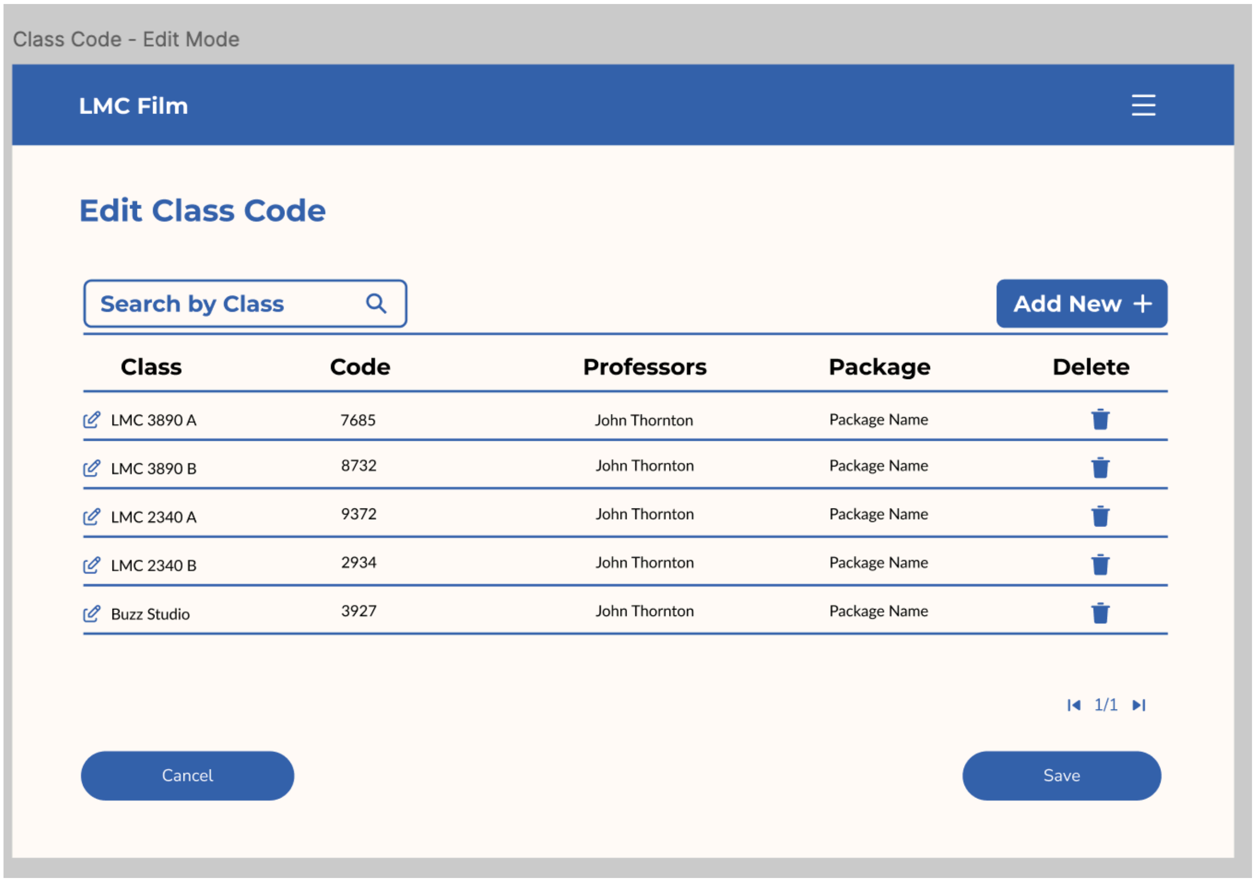


Figure 23 - Management Page

This is the page administrators see after selecting the **Management** tab. Here, administrators can generate new class codes for students by selecting the **Class Code** tab, allowing them to input a specific code that corresponds to their class. This ensures that students can only view the equipment required by their professors. Additionally, administrators can assign users to different roles, such as professors or students, once they have logged in using their GT email.



Class Code - Edit Mode

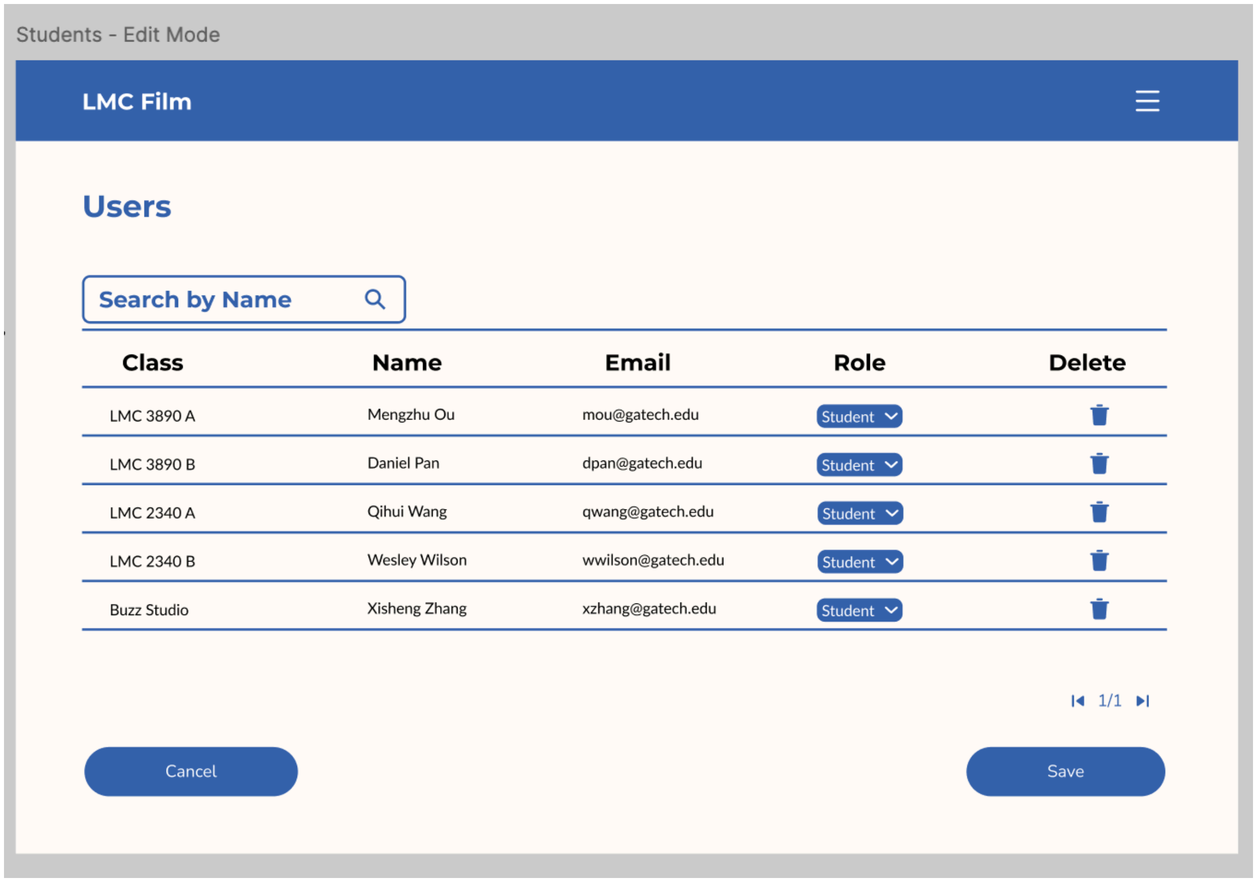
LMC Film

Edit Class Code

Class	Code	Professors	Package	Delete
LMC 3890 A	7685	John Thornton	Package Name	🗑
LMC 3890 B	8732	John Thornton	Package Name	🗑
LMC 2340 A	9372	John Thornton	Package Name	🗑
LMC 2340 B	2934	John Thornton	Package Name	🗑
Buzz Studio	3927	John Thornton	Package Name	🗑

Figure 24 - Class Code Page

This is the **Class Code** page, where admins can add new classes. Each class will have a randomly generated code from the system. Within this feature, admins can select specific equipment for students to access and create convenient packages that include all the equipment needed for their project. This allows students to reserve the entire package with a single click instead of selecting items individually.

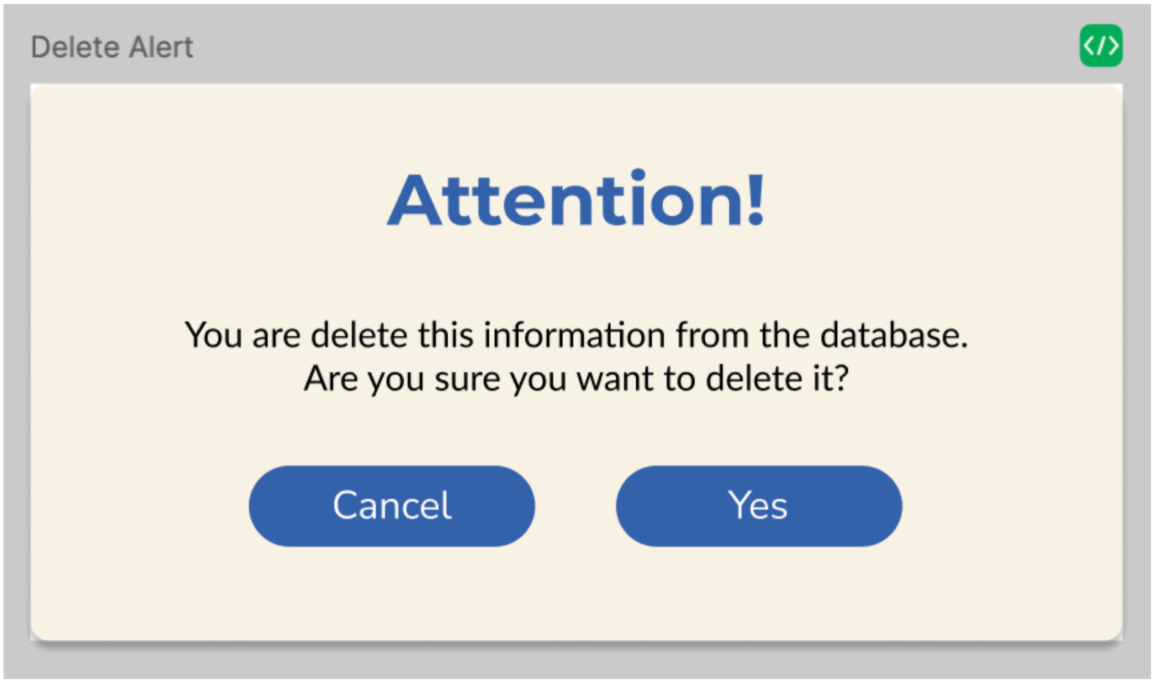

Students - Edit Mode

Class	Name	Email	Role	Delete
LMC 3890 A	Mengzhu Ou	mou@gatech.edu	Student	
LMC 3890 B	Daniel Pan	dpan@gatech.edu	Student	
LMC 2340 A	Qihui Wang	qwang@gatech.edu	Student	
LMC 2340 B	Wesley Wilson	wwilson@gatech.edu	Student	
Buzz Studio	Xisheng Zhang	xzhang@gatech.edu	Student	

Figure 25 - Users Page

This is the **Users** page, where admins can assign roles to users, particularly for students who will be film lab assistants and professors. When users first log in, their roles are set to "student" by default, and admins can update their roles as needed.

All of our interface designs containing tables and pop-up modals are consistently styled throughout the website. Additionally, elements like buttons, search bars, and colors are standardized across the site, which aligns with one of the usability heuristics for a cohesive user experience.



You are delete this information from the database.
Are you sure you want to delete it?

Figure 26 - Pop-up Delete Alert

This is a pop-up alert linked to all delete buttons on pages containing tables across the website. When an administrator clicks the delete button, this alert appears to confirm if they indeed intend to delete the specific item. If not, they can click **Cancel** to return to the previous screen. This acts as an “emergency exit”, aligning with a usability heuristic to prevent unintended actions by the user.

Students

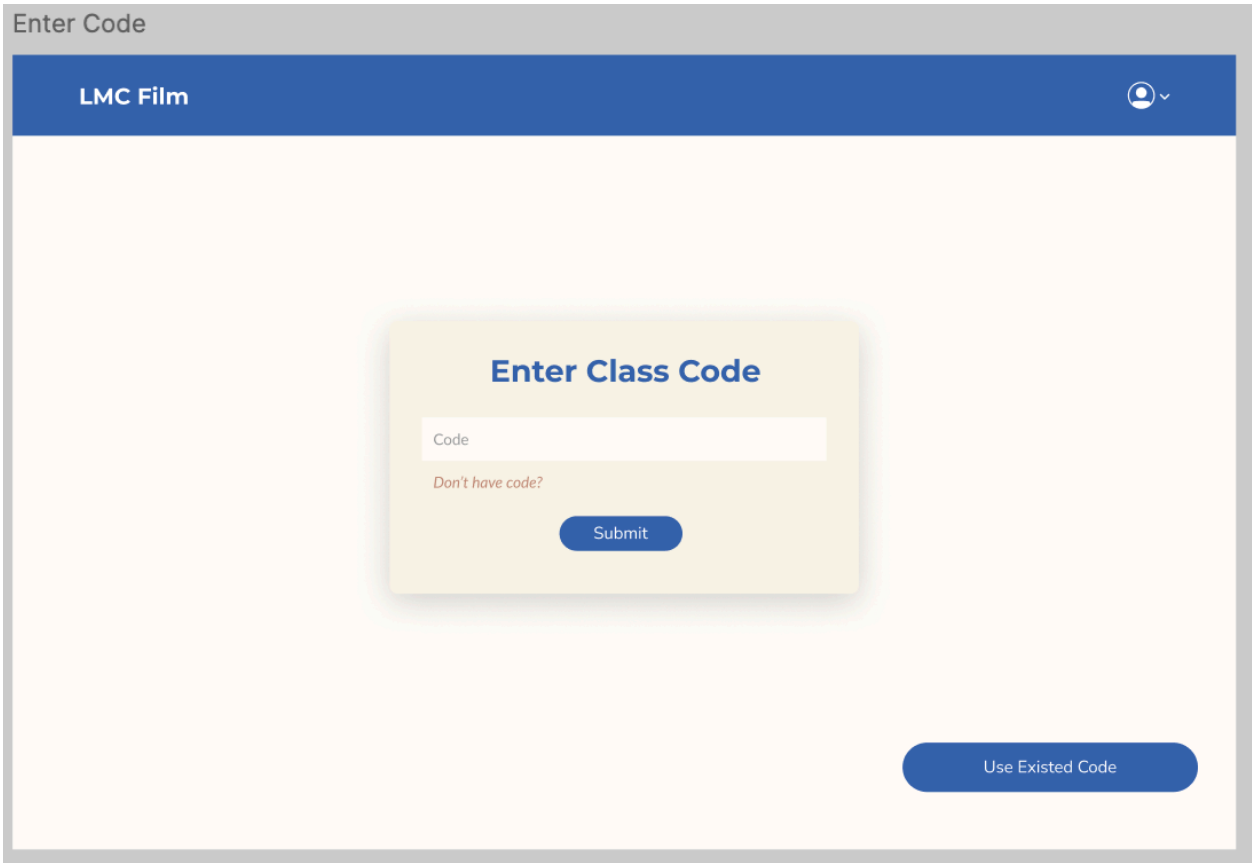
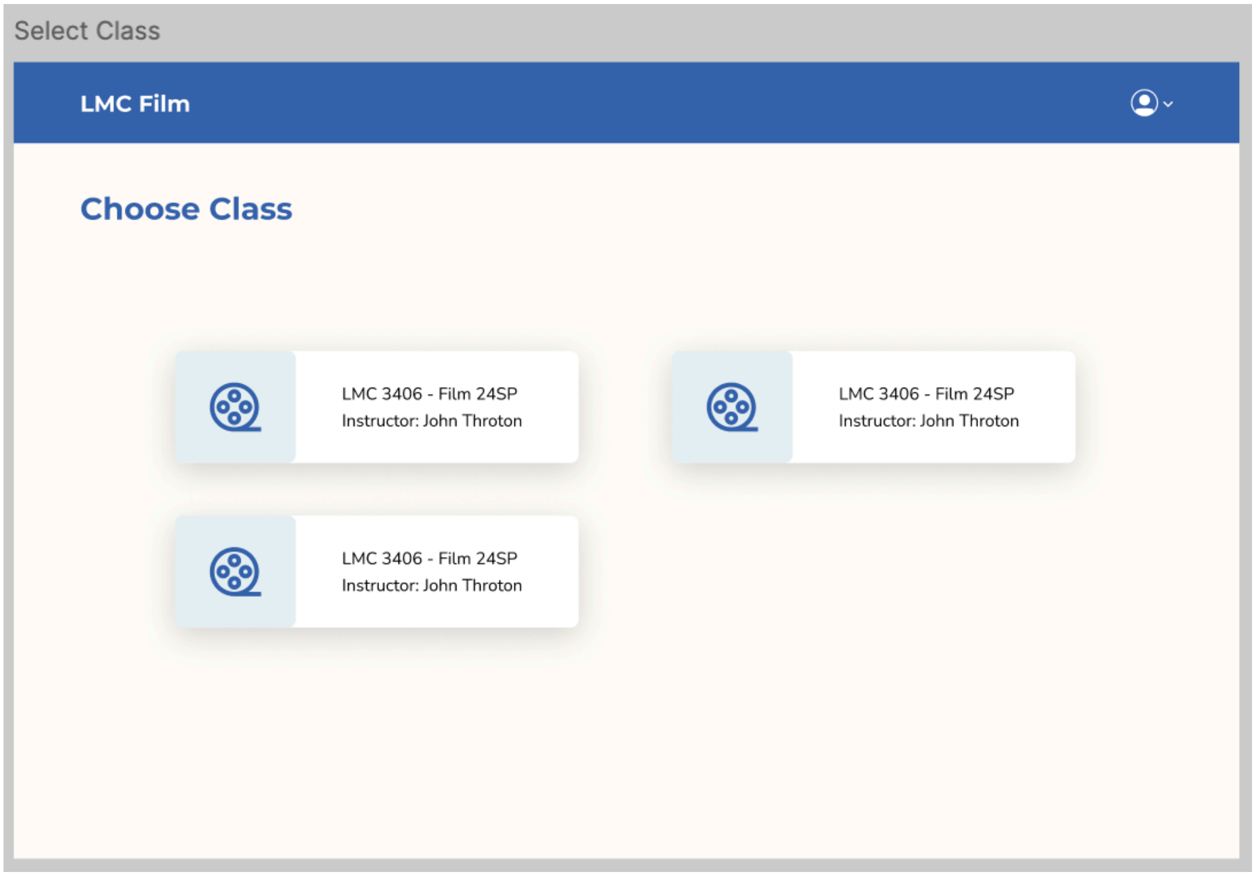


Figure 27 - Enter Code Page

After logging in with their GT email, students will see this page if the system identifies them as a student. Here, they can enter the code provided by their professor if they haven't entered one already. Otherwise, they can proceed to the next step by clicking the **Use Existed Code** button. This page will always be the first one students see upon entering the website, ensuring they can enter codes for multiple classes requiring film equipment reservations. If they've already entered a code, they can skip this step by moving to the next page using the button.



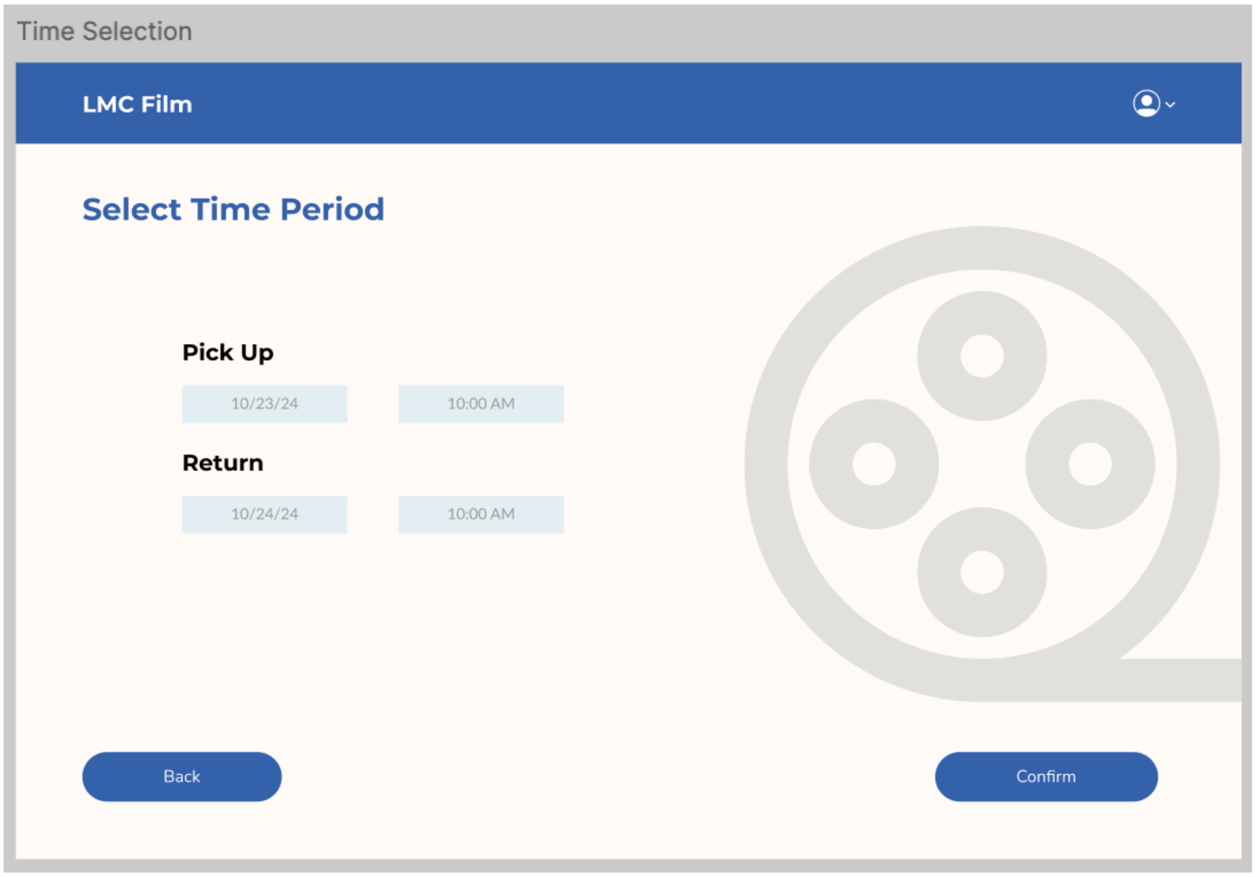
LMC 3406 - Film 24SP
Instructor: John Throton

LMC 3406 - Film 24SP
Instructor: John Throton

LMC 3406 - Film 24SP
Instructor: John Throton

Figure 28 - Select Class Page

This is the page students see after entering their class codes. The displayed number of classes corresponds to the number of codes they've entered. For example, in the figure above, the student has entered three class codes, as they're registered in three different classes. The maximum number of class codes a student can enter is four, as it's uncommon for a student to enroll in more than four LMC classes that require film equipment reservations.



Select Time Period

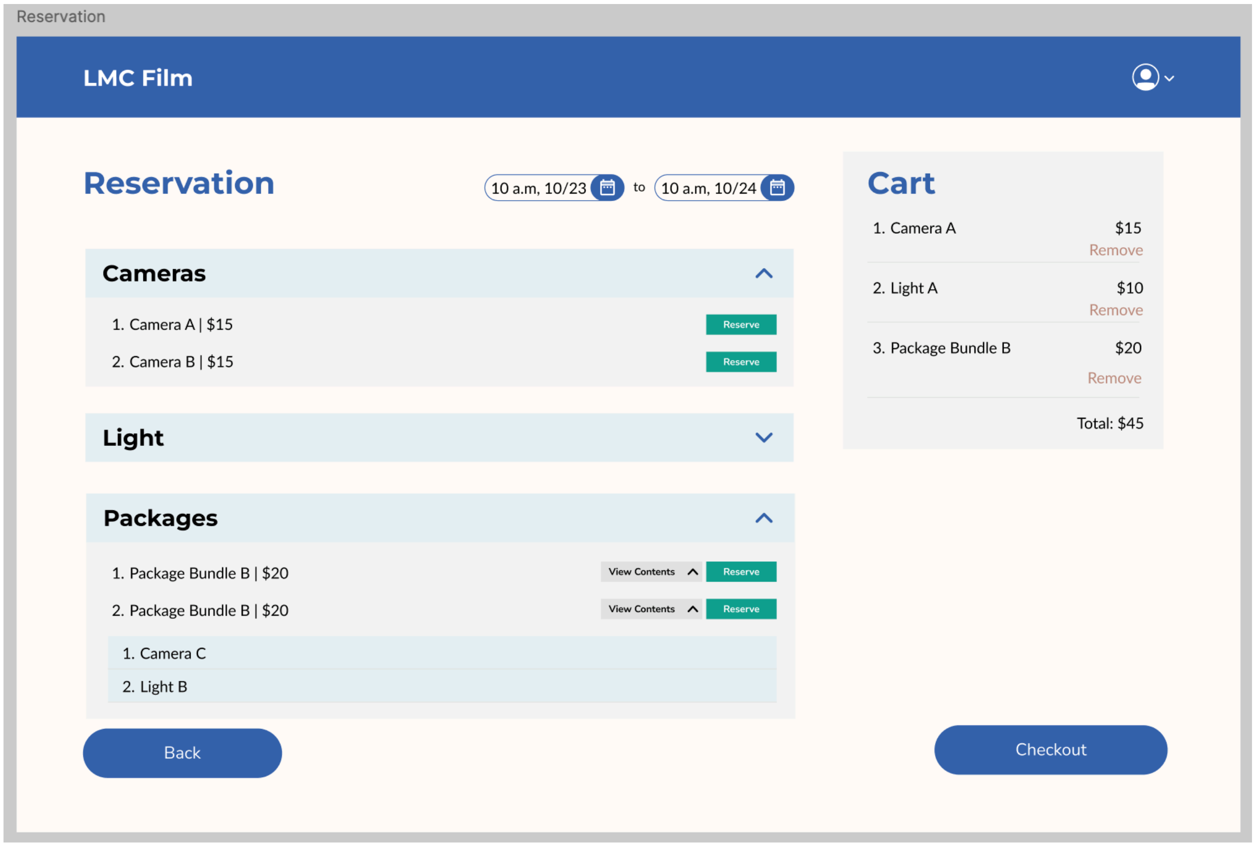
Pick Up

10/23/24 10:00 AM

Return

10/24/24 10:00 AM

Figure 29 - Select Time Page


Reservation

LMC Film

Reservation

10 a.m, 10/23 to 10 a.m, 10/24

Cameras

1. Camera A | $15 Reserve
2. Camera B | $15 Reserve

Light

Packages

1. Package Bundle B | $20 View Contents Reserve
2. Package Bundle B | $20 View Contents Reserve

1. Camera C
2. Light B

Back

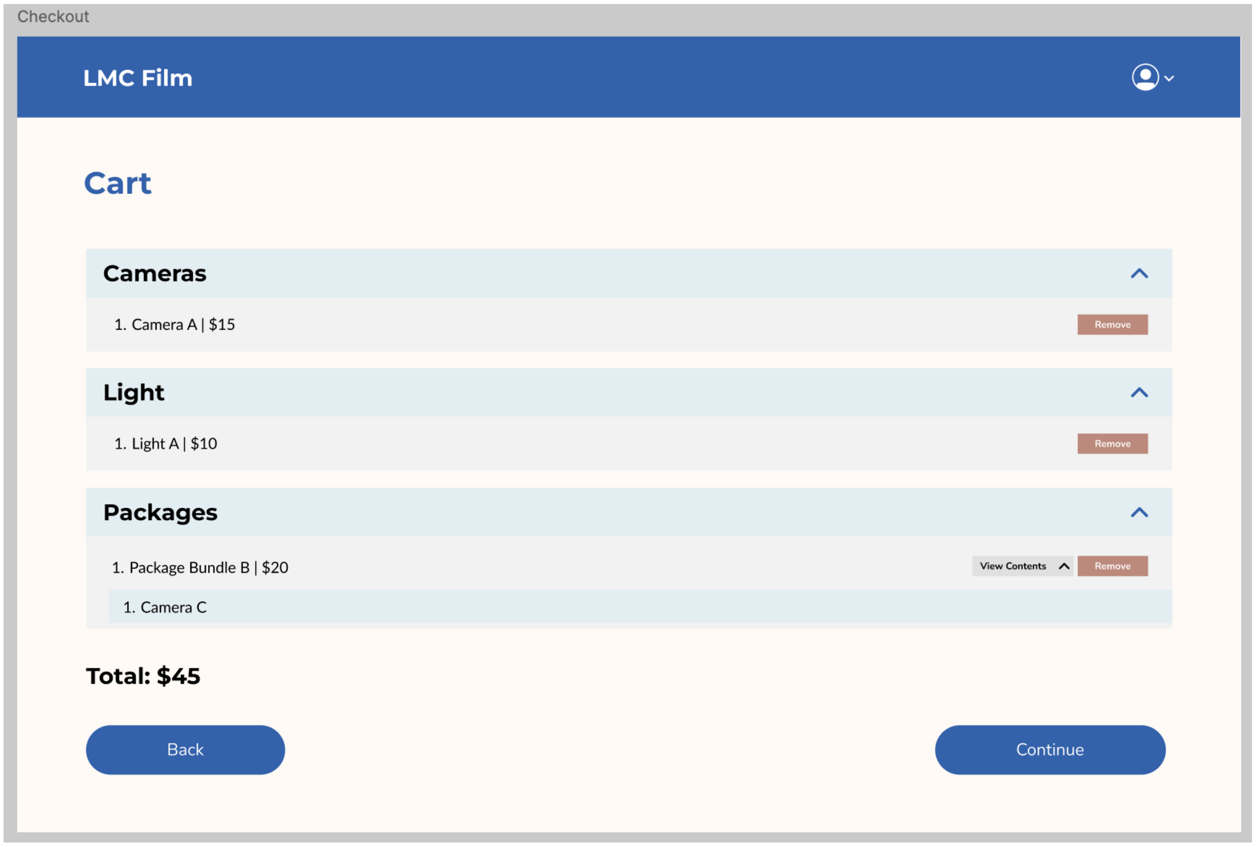
Cart

1. Camera A	$15 Remove
2. Light A	$10 Remove
3. Package Bundle B	$20 Remove
	Total: $45

Checkout

Figure 30 - Reservation Page

Figure 29 shows the next step after students select the class they want to make a film reservation for: time selection. Here, they can choose their pick-up and return dates and times. Once this is completed, they proceed to the page shown in Figure 30, where they can select specific equipment or packages created by the professor for the chosen class. As students make selections, the items appear in the **Cart** table on the right side of the page. If they wish to remove an item, they can click the **Remove** button next to it in the **Cart** section. Additionally, if they want to adjust the pick-up or return date and time, they can easily do so using the two tabs at the top without having to navigate back to the previous page. Once ready, students can proceed by clicking the **Checkout** button in the bottom right corner.


Checkout

LMC Film

Cart

Cameras

1. Camera A | $15 Remove

Light

1. Light A | $10 Remove

Packages

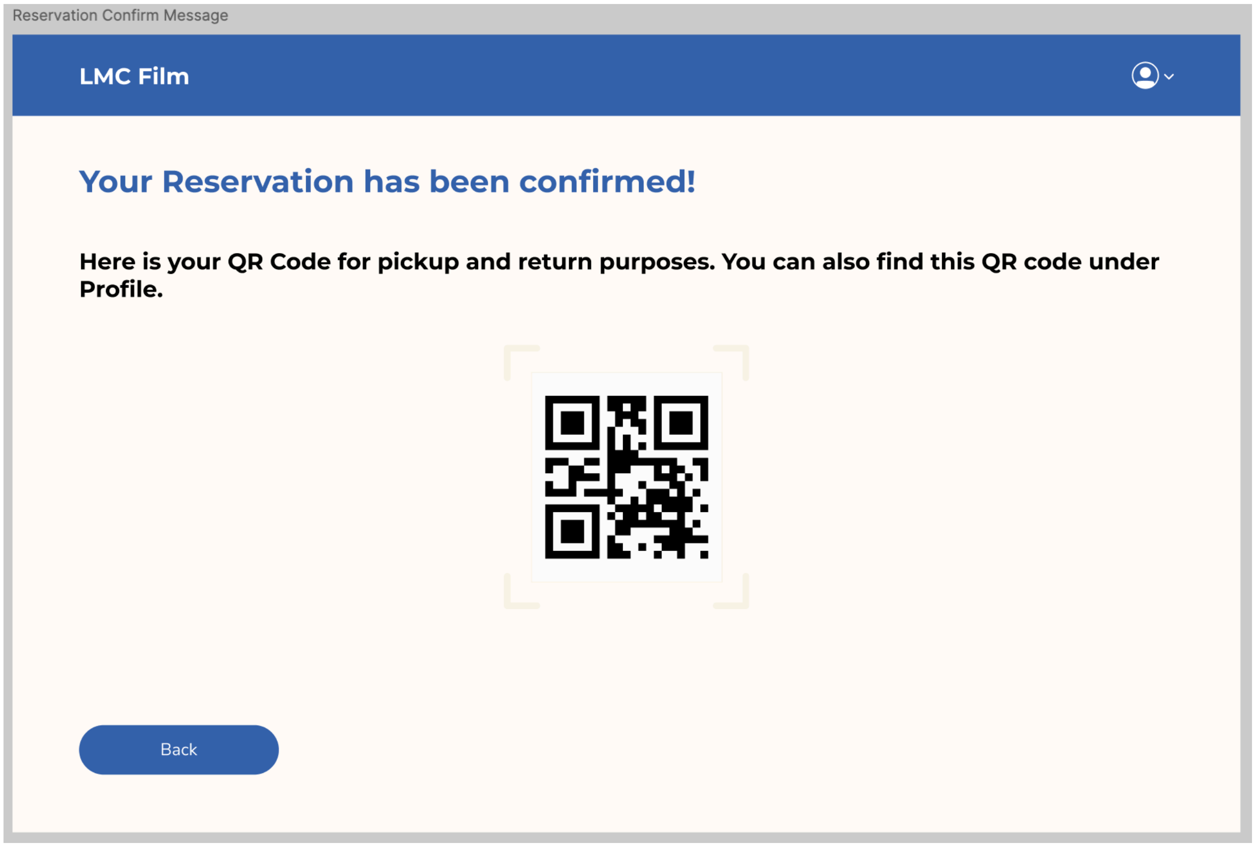
1. Package Bundle B | $20 View Contents Remove

1. Camera C

Total: $45

Figure 31 - Cart Page

This is the next page, where students can review all items in their cart and finalize their selections before proceeding to the payment step.

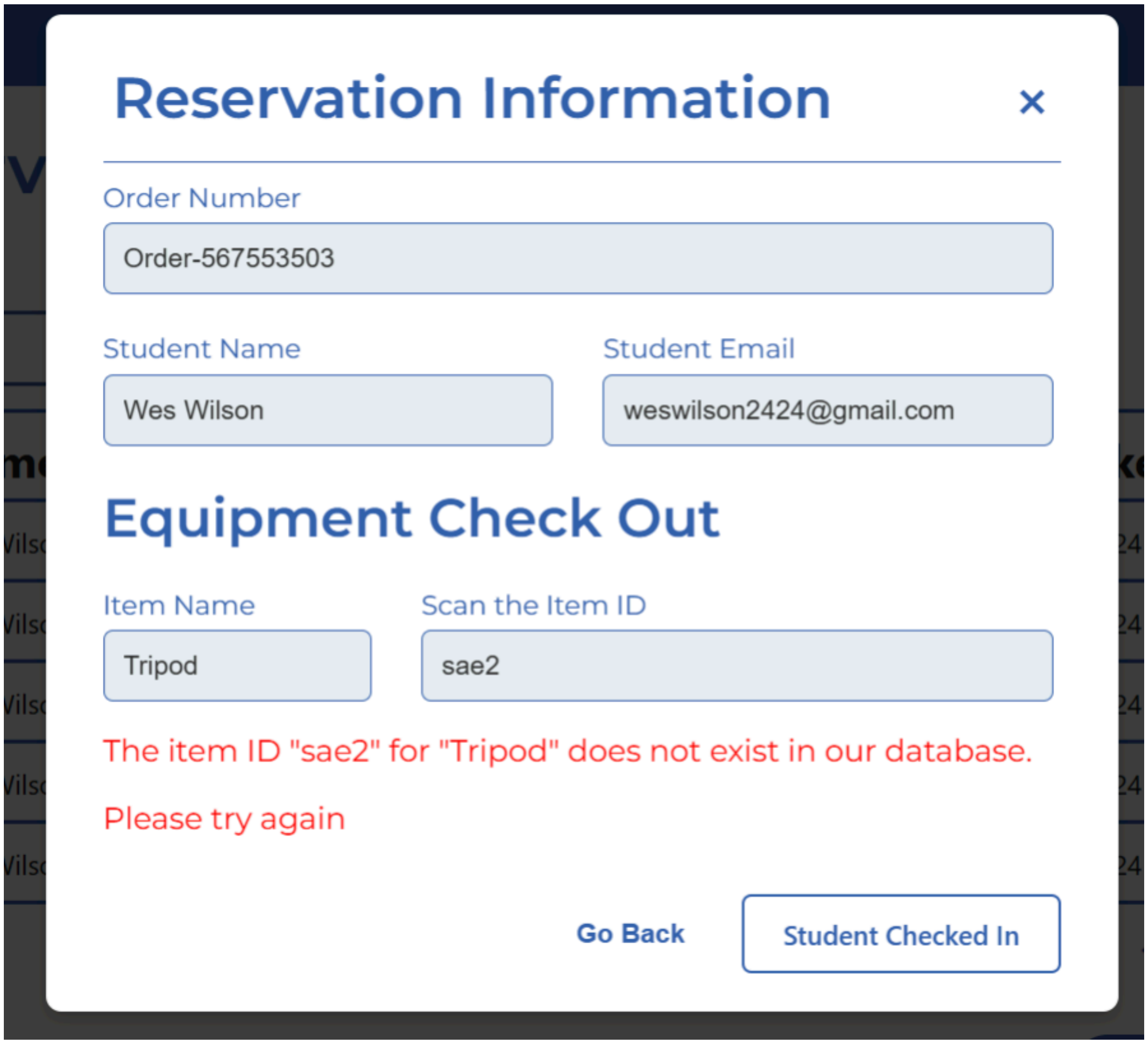

Reservation Confirm Message

Your Reservation has been confirmed!

Here is your QR Code for pickup and return purposes. You can also find this QR code under Profile.

Figure 32 - Reservation Confirmation Page

This is the page displayed to students after a reservation is successfully made. A barcode is generated for each order, and the same confirmation message is also sent to the student's email.



Reservation Information

Order Number: Order-567553503

Student Name: Wes Wilson

Student Email: weswilson2424@gmail.com

Equipment Check Out

Item Name: Tripod

Scan the Item ID: sae2

The item ID "sae2" for "Tripod" does not exist in our database.

Please try again

Figure 33 - Reservation Information (Pick Up Equipment) Modal

This is a pop-up modal displayed when a student arrives to pick up equipment and the administrator attempts to check the equipment out. If an incorrect barcode is scanned, one that does not exist in the database (meaning the Item ID is not listed in the Equipment page), an error message appears to indicate the equipment does not exist in the database. This error message design aligns effectively with usability heuristics for error prevention and user support in recognizing, diagnosing, and recovering from errors. Upon scanning a student's equipment barcode, the system instantly checks its validity against the database to prevent accidental checkouts of items not in the system. If the barcode isn't found, a prominent, clear error message appears, stating "**Equipment Does Not Exist in Our Database**," with guidance to "**Please Try Again.**" This message is in plain language to ensure clarity, and the "**Please Try Again**" prompt allows users to quickly resolve the error without disrupting their workflow. This design assists users in avoiding errors and provides a straightforward path for error recovery, supporting both error prevention and recovery principles.

Professors

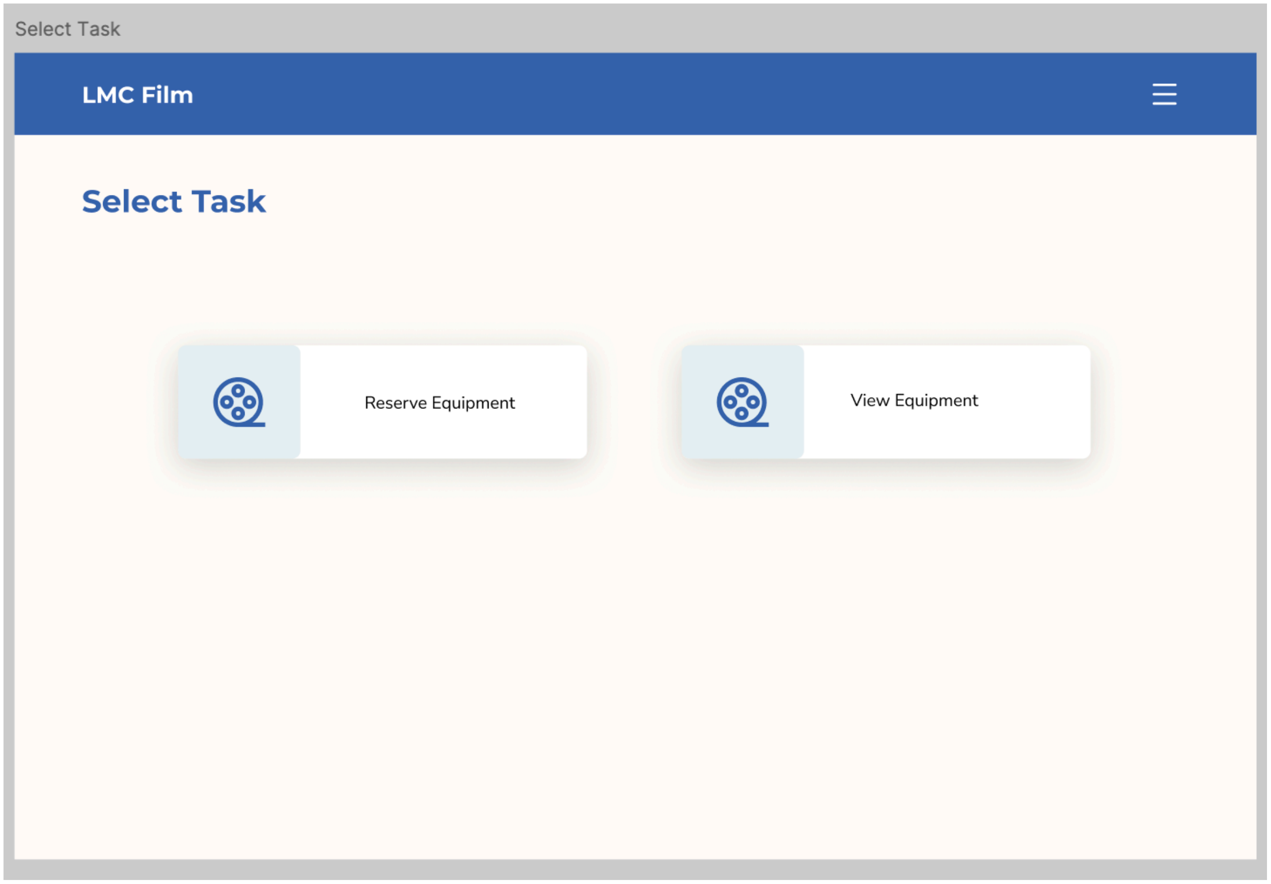


Figure 34 - Select Task Page for Professors

Student Assistants

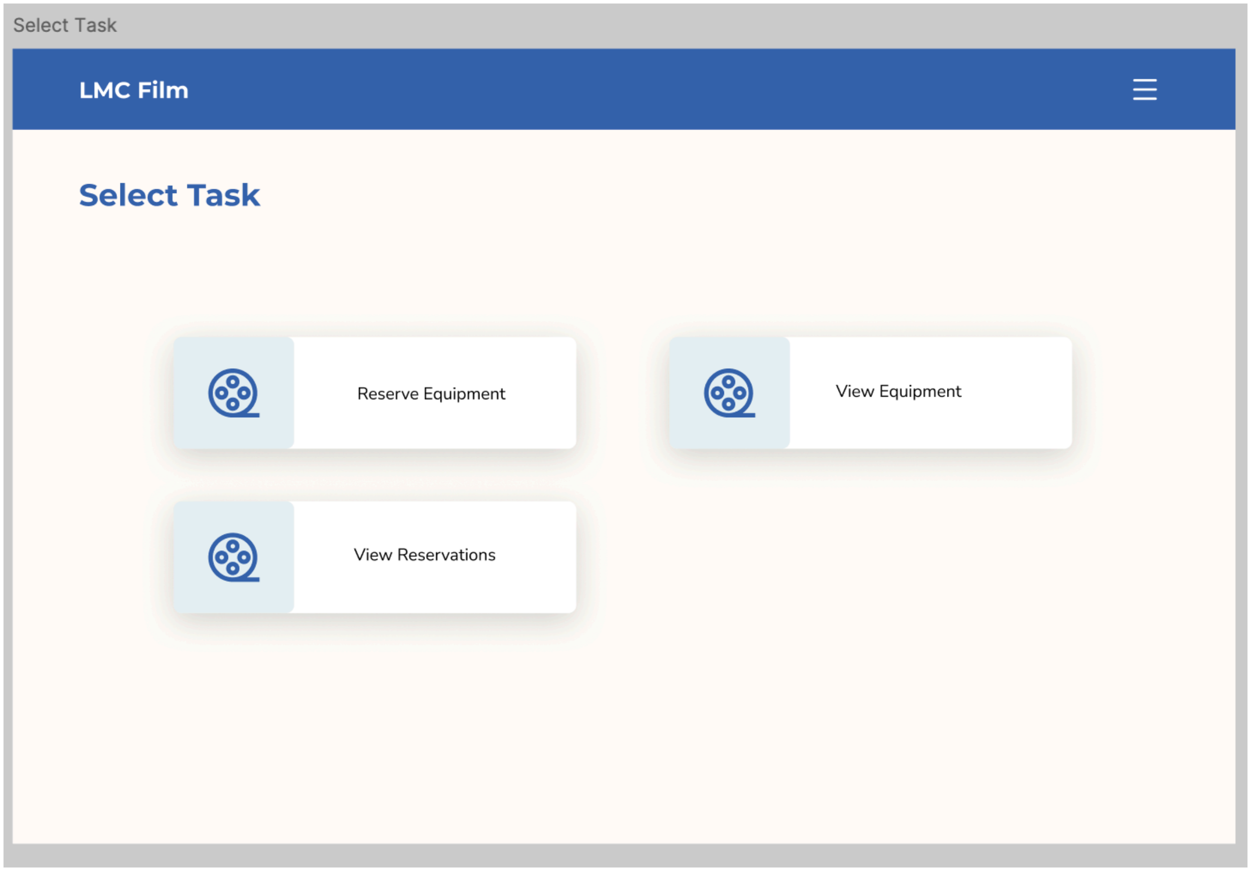


Figure 35 - Select Task Page for Student Assistants

Figures 34 and 35 illustrate the different tasks available to Professors and Student Assistants. As mentioned earlier, Professors and Student Assistants have access to a subset of the functionalities available to an Admin (the website owner). Since we categorized all admin tasks into four main areas, we can selectively restrict certain abilities based on the user's role, as requested by our client. This approach personalizes the experience by tailoring content and functionality to specific roles.