PROJECT

Bus Reservation System

PRESENTED TO

Sakkaravarthi Ramanathan

FROM

Ethan Tran, Christopher Soussa and Nikolas Polyhronopoulos

PRESENTED

Monday, May 15, 2023

Outline

Contents

1 Description

The Bus Reservation System is a software application designed to streamline the process of booking and managing bus tickets for both customers and administrators. It is a user-friendly and efficient software application developed to simplify the process of reserving bus seats, managing bookings, and maintaining user profiles. The system provides an intuitive graphical user interface (GUI) designed using JavaFX, offering a seamless experience for both administrators and clients. It utilizes the Jackson library for data storage in JSON files, ensuring reliable and secure data management.

The primary goal of the software is to offer a convenient and reliable platform where passengers can effortlessly search for available bus routes, check seat availability, and make secure reservations from the comfort of their homes or on the go. The system also provides bus operators with a comprehensive management tool to efficiently manage their buses, schedules, and bookings.

2 Features & Functionalities

The system offers a comprehensive set of features that cater to the diverse needs of users. Clients can easily register and create their profiles, providing essential personal information. The user profiles can be conveniently updated at any time, allowing clients to modify contact details, email addresses, names, and other relevant information as needed. Furthermore, the system allows clients to deposit money into their accounts, providing a streamlined payment process. This deposit feature offers clients the flexibility to manage their funds conveniently within the system. Additionally, clients can withdraw money from their accounts, adhering to withdrawal rules and limits, thus ensuring transparency and user control. Clients can effortlessly search for available buses based on the desired source, destination, and date. The system presents clients with an interactive interface displaying the availability of seats on selected buses, allowing them to choose their preferred seats conveniently. Real-time seat availability information enables clients to make informed decisions while making bookings. Moreover, clients can reserve multiple seats in a single booking, subject to the capacity of the selected bus, thus accommodating group bookings or families traveling together.

The system also empowers administrators with extensive management capabilities. Admins have full authority to add, edit, and remove buses from the system, ensuring that the database accurately reflects the available fleet. Each bus entry within the system encompasses vital details such as the bus ID, available seating and capacity, origin and destination locations, departure and arrival date and times. Admins can update bus information whenever necessary, ensuring that the system maintains real-time data integrity. Admins can view and manage all bookings made by clients, granting them the ability to modify booking details such as seat numbers or reschedule journeys, accommodating last-minute changes or unforeseen circumstances. Furthermore, admins can cancel bookings when necessary, adhering to cancellation policies and refund rules, ensuring a fair and transparent process for both parties involved.

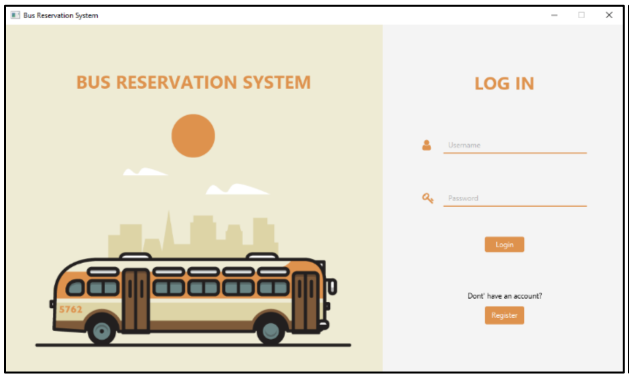
2.1 Login and Sign-Up

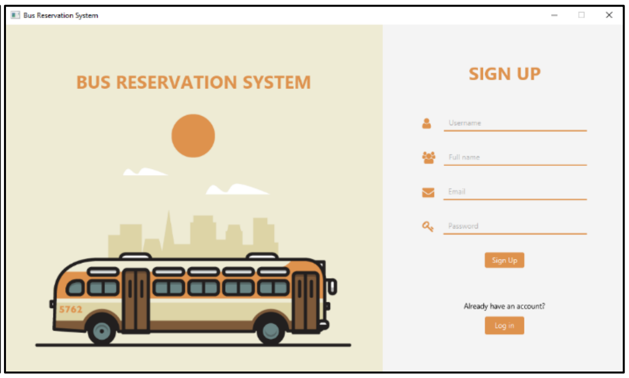
Clients and admins can create accounts and log in to the system to access booking functionality. To create an admin account, the admin can type @KEY_ADMIN after their username during registration. This ensures a personalized experience and allows users to manage their reservations effectively. Clients can register and create their profiles by providing necessary personal information. Clients can edit their profiles, including updating contact details, email addresses, name, password and other relevant information.

Controller Logic - AuthenticationController

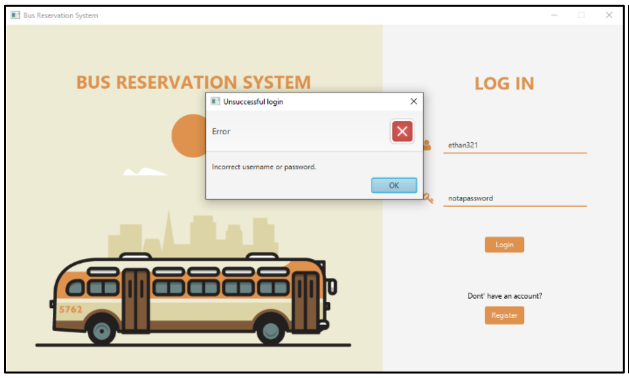
AuthenticationController in the bus reservation application is responsible for handling user authentication and account creation. It is linked to two FXML pages LoginForm.fxml and SignUpForm.fxml. It manages the login and sign-up forms, validating user input and performing the necessary operations. The login() function validates the username and password, sets the current user in the Database, and loads the appropriate scene based on the user's role (client or admin). The signup() function validates user input, creates a new client account, adds it to the Database, and loads the login form. The switchForm() function handles the switching between the login and sign-up forms based on user interaction.

2.1.1 Login and Sign-Up Pages





2.1.2 Login and Sign-Up User interactions

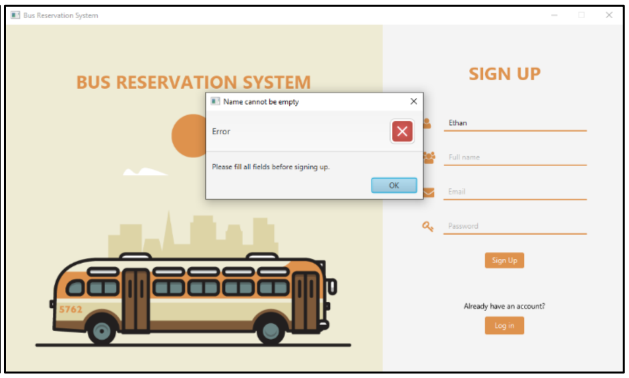


Unsuccessful login

Error

Incorrect username or password.

OK

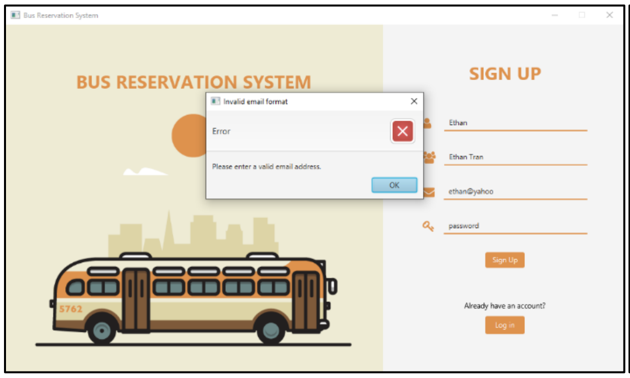


Name cannot be empty

Error

Please fill all fields before signing up.

OK



Invalid email format

Error

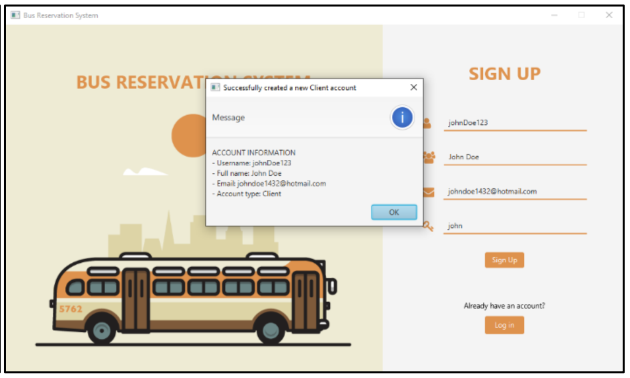
Please enter a valid email address.

Ethan

Ethan Tran

ethan@yahoo

password



Successfully created a new Client account

Message

ACCOUNT INFORMATION
- Username: johnDoe123
- Full name: John Doe
- Email: johndoe1432@hotmail.com
- Account type: Client

johnDoe123

John Doe

johndoe1432@hotmail.com

john

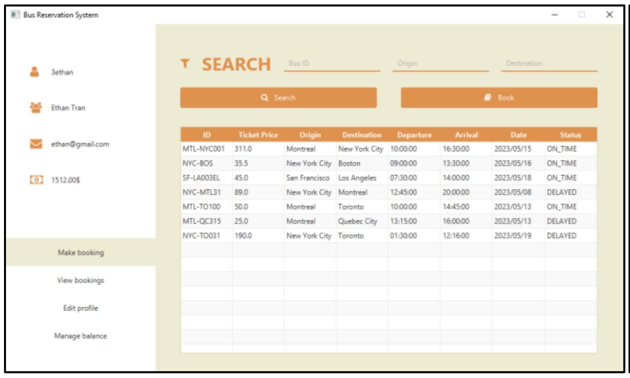
2.2 Client bus selection

This page serves as a dedicated interface for clients to search for and select their desired buses based on specific criteria. This page provides a range of functionalities to enhance the user experience and facilitate informed decision-making during the booking process. Clients can enter their preferred source and destination locations, along with the bus ID. The system performs a real-time search and displays a list of available buses that match the specified criteria.

Controller Logic - ClientMakeBookingsController

The ClientMakeBookingsController class is a controller responsible for managing the bus booking process for clients in the bus reservation system. It extends the ClientController class and initializes the UI elements, including a table view displaying bus information. Clients can search and filter buses based on criteria such as ID, origin, and destination. The controller retrieves data from the database and populates the table view accordingly. When the user selects a bus and clicks the "Book" button, the controller sets the selected bus as the current bus and loads the seat selection view.

2.2.1 Client bus selection page



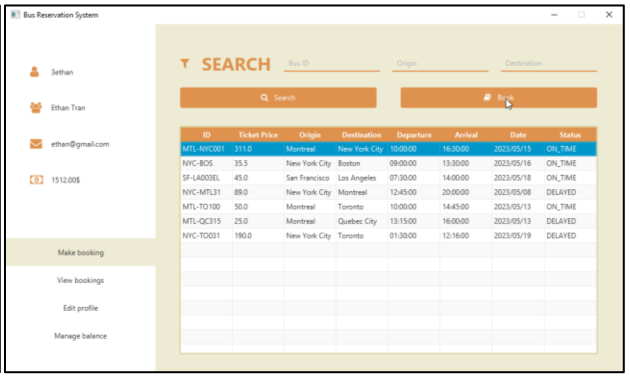
3ethan

Ethan Tran

ethan@gmail.com

1512.00$

ID	Ticket Price	Origin	Destination	Departure	Arrival	Date	Status
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED



3ethan

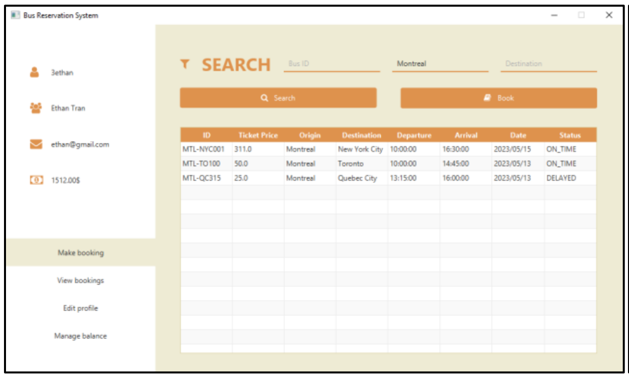
Ethan Tran

ethan@gmail.com

1512.00$

ID	Ticket Price	Origin	Destination	Departure	Arrival	Date	Status
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED

2.2.2 Client bus selection searching



3ethan

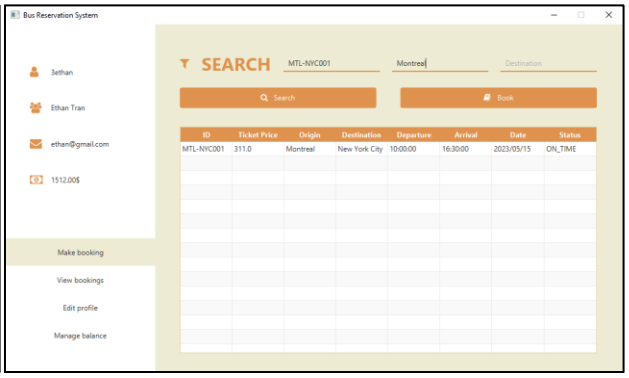
Ethan Tran

ethan@gmail.com

1512.00$

Montreal

ID	Ticket Price	Origin	Destination	Departure	Arrival	Date	Status
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED



3ethan

Ethan Tran

ethan@gmail.com

1512.00$

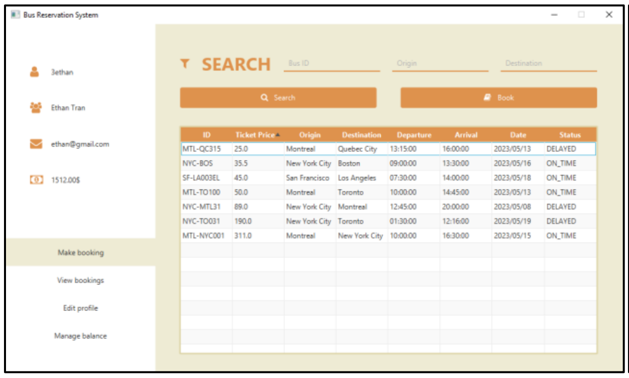
MTL-NYC001

Montreal

ID	Ticket Price	Origin	Destination	Departure	Arrival	Date	Status
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME

2.2.3 Client bus selection sorting

Sorting by ticket price – Ascending vs. Descending



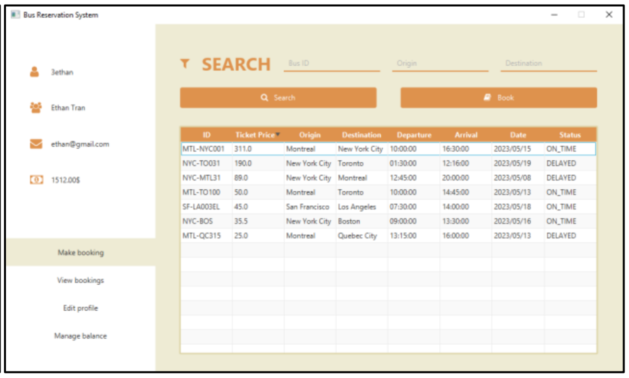
3ethan

Ethan Tran

ethan@gmail.com

1512.00$

ID	Ticket Price ▲	Origin	Destination	Departure	Arrival	Date	Status
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME



3ethan

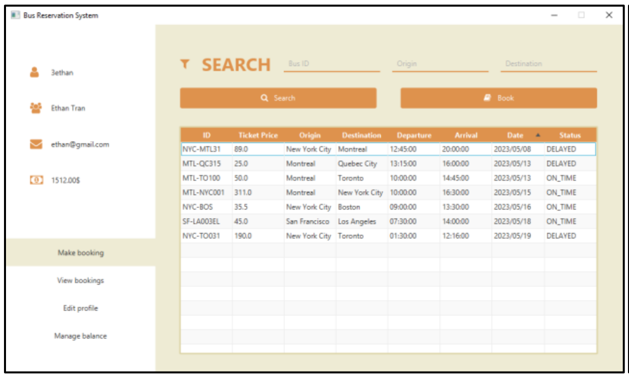
Ethan Tran

ethan@gmail.com

1512.00$

ID	Ticket Price ▼	Origin	Destination	Departure	Arrival	Date	Status
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED

Sorting by date – Ascending vs. Descending



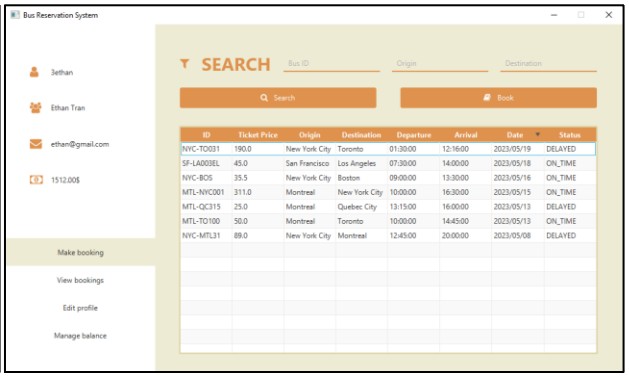
3ethan

Ethan Tran

ethan@gmail.com

1512.00$

ID	Ticket Price	Origin	Destination	Departure	Arrival	Date ▲	Status
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED



3ethan

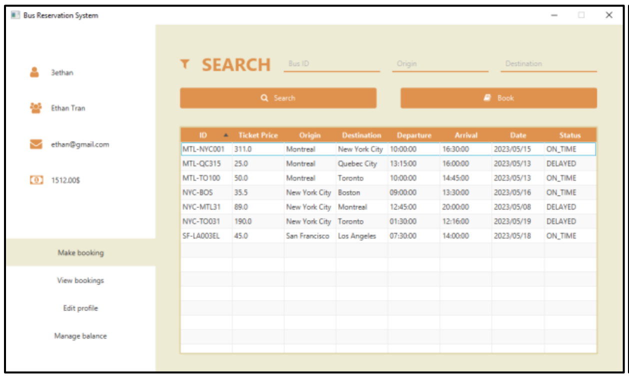
Ethan Tran

ethan@gmail.com

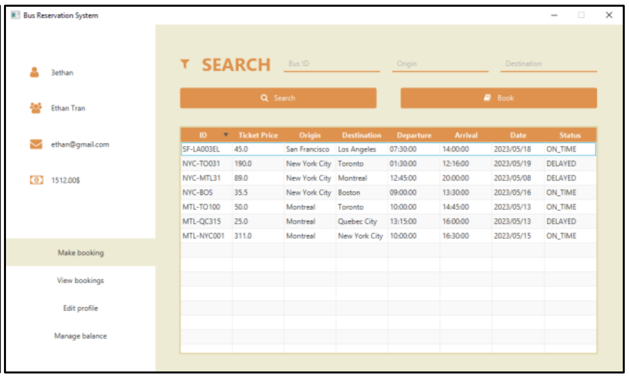
1512.00$

ID	Ticket Price	Origin	Destination	Departure	Arrival	Date ▼	Status
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED

Sorting by ID lexicographically – Ascending vs. Descending

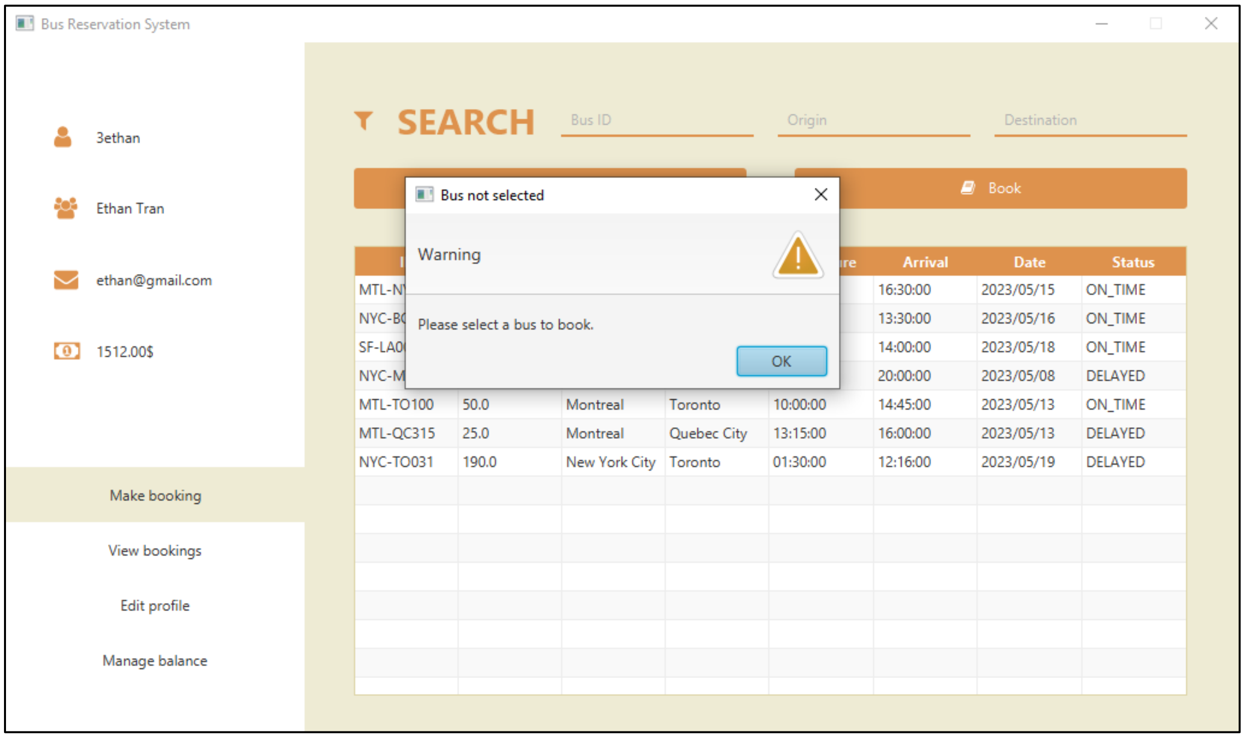


ID ▲	Ticket Price	Origin	Destination	Departure	Arrival	Date	Status
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME



ID ▼	Ticket Price	Origin	Destination	Departure	Arrival	Date	Status
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME

2.2.4 Client bus selection interactions



Bus not selected

Warning

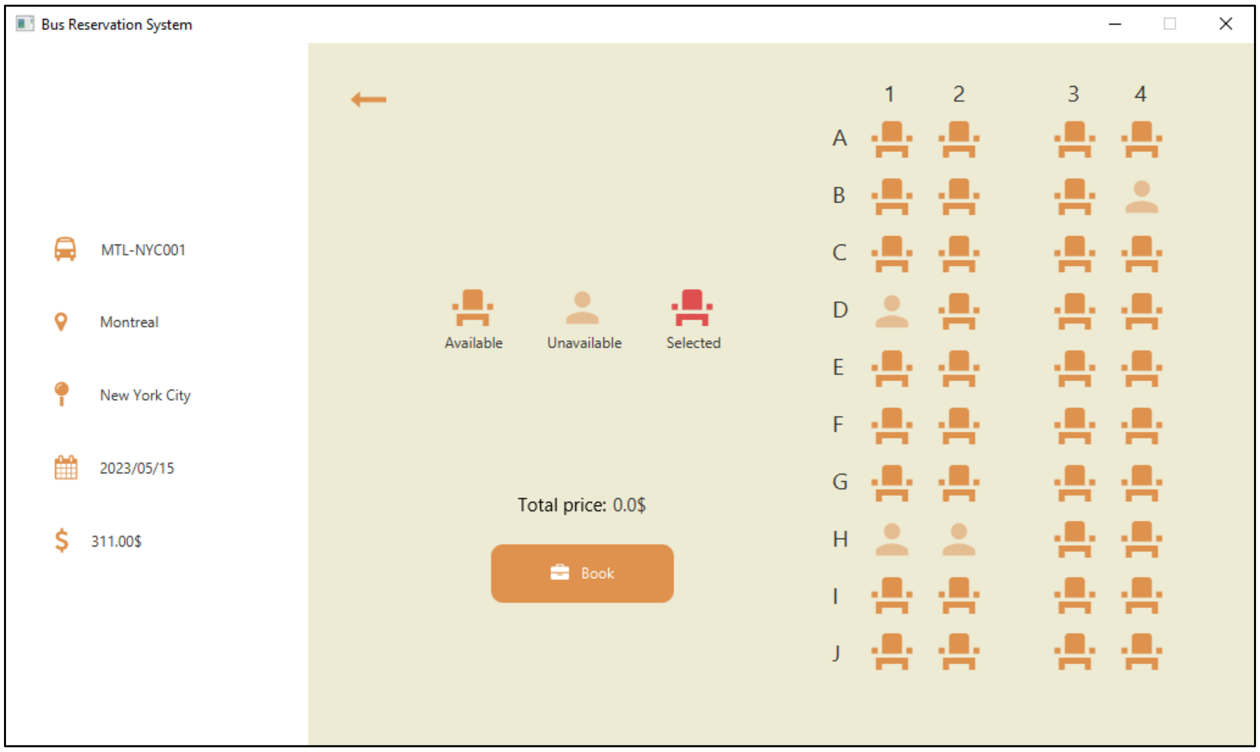
Please select a bus to book.

I				re	Arrival	Date	Status
MTL-N					16:30:00	2023/05/15	ON_TIME
NYC-B					13:30:00	2023/05/16	ON_TIME
SF-LA0					14:00:00	2023/05/18	ON_TIME
NYC-M					20:00:00	2023/05/08	DELAYED
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED

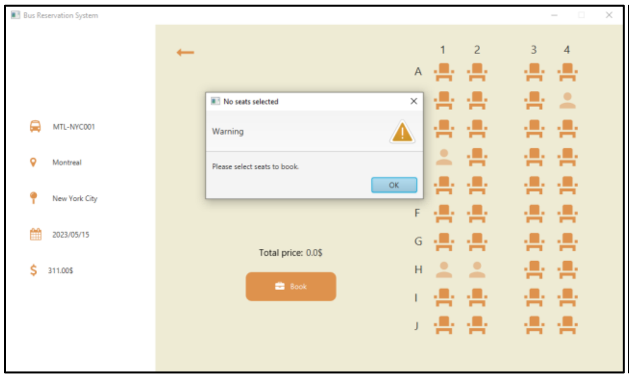
2.3 Client seat selection

Upon selecting a bus from the search results, clients can access detailed information about seat availability on that particular bus. The system presents an interactive seating layout, visually indicating the occupied and vacant seats. Clients can choose their preferred seats from the available options presented on the seating layout. The system allows clients to select multiple seats based on their requirements, subject to the bus's capacity. Seat selection is intuitive and user-friendly, ensuring a smooth booking experience. As clients make seat selections, the system dynamically calculates the total fare based on the chosen seats and any applicable pricing rules. Clients can view the calculated fare in real-time, providing transparency and helping them make informed decisions.

2.3.1 Client seat selection page



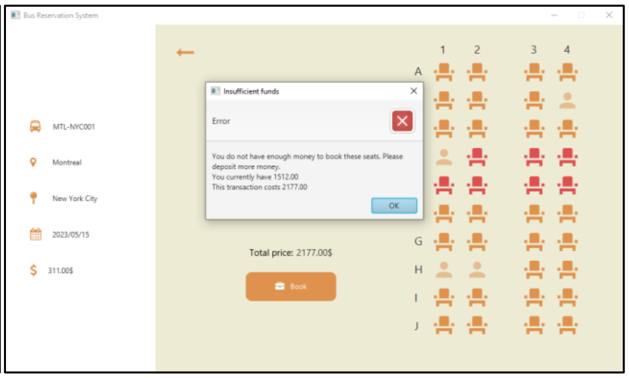
2.3.2 Client seat selection interaction


No seats selected

Warning

Please select seats to book.

OK


Insufficient funds

Error

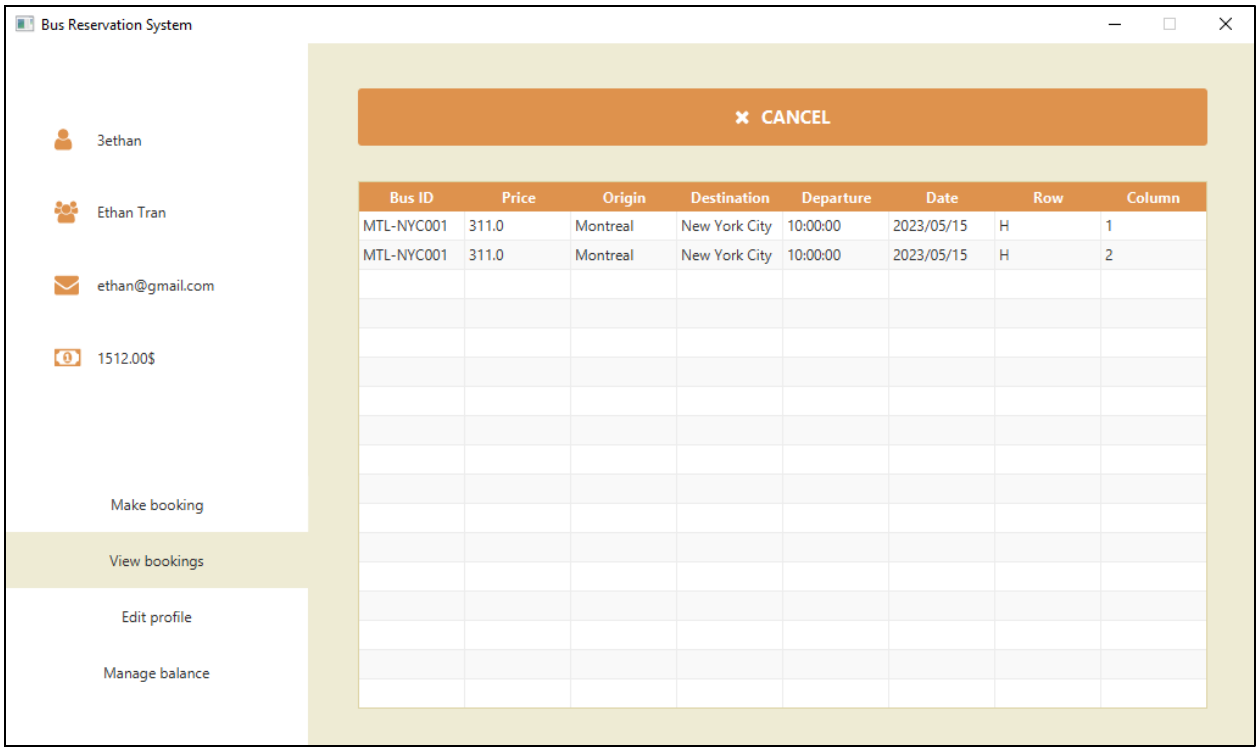
You do not have enough money to book these seats. Please deposit more money.
You currently have 1512.00
This transaction costs 2177.00

OK

2.4 Client view bookings

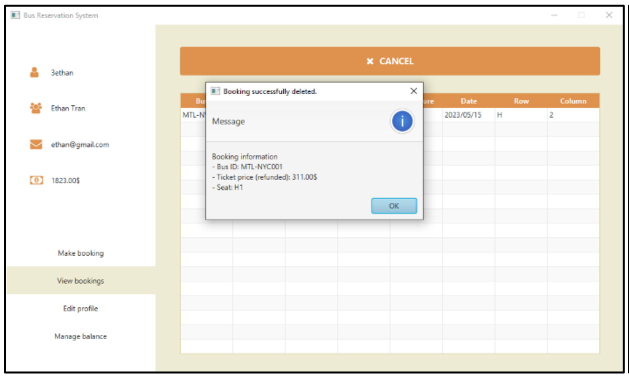
This page is a dedicated section within the Bus Reservation System that provides clients with comprehensive information and functionalities related to their bookings. Designed with user convenience in mind, this page offers a range of features that allow clients to manage their booked seats, view booking details, and cancel their booking. If they choose to do so, they will be refunded and the seat in the bus will be reopened.

2.4.1 Client view bookings page



Bus ID	Price	Origin	Destination	Departure	Date	Row	Column
MTL-NYC001	311.0	Montreal	New York City	10:00:00	2023/05/15	H	1
MTL-NYC001	311.0	Montreal	New York City	10:00:00	2023/05/15	H	2

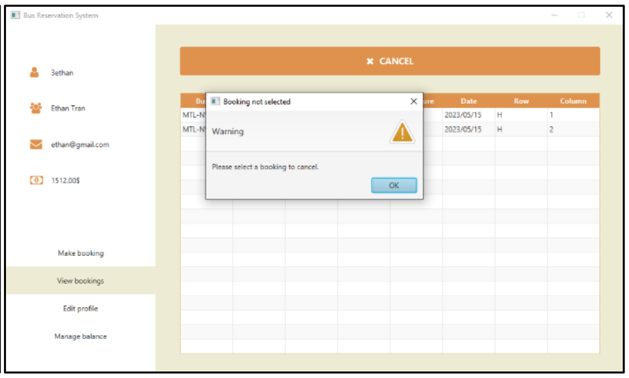
2.4.2 Client view bookings interactions



Booking successfully deleted.

Message

Booking information
- Bus ID: MTL-NYC001
- Ticket price (refunded): 311.00$
- Seat: H1



Booking not selected

Warning

Please select a booking to cancel.

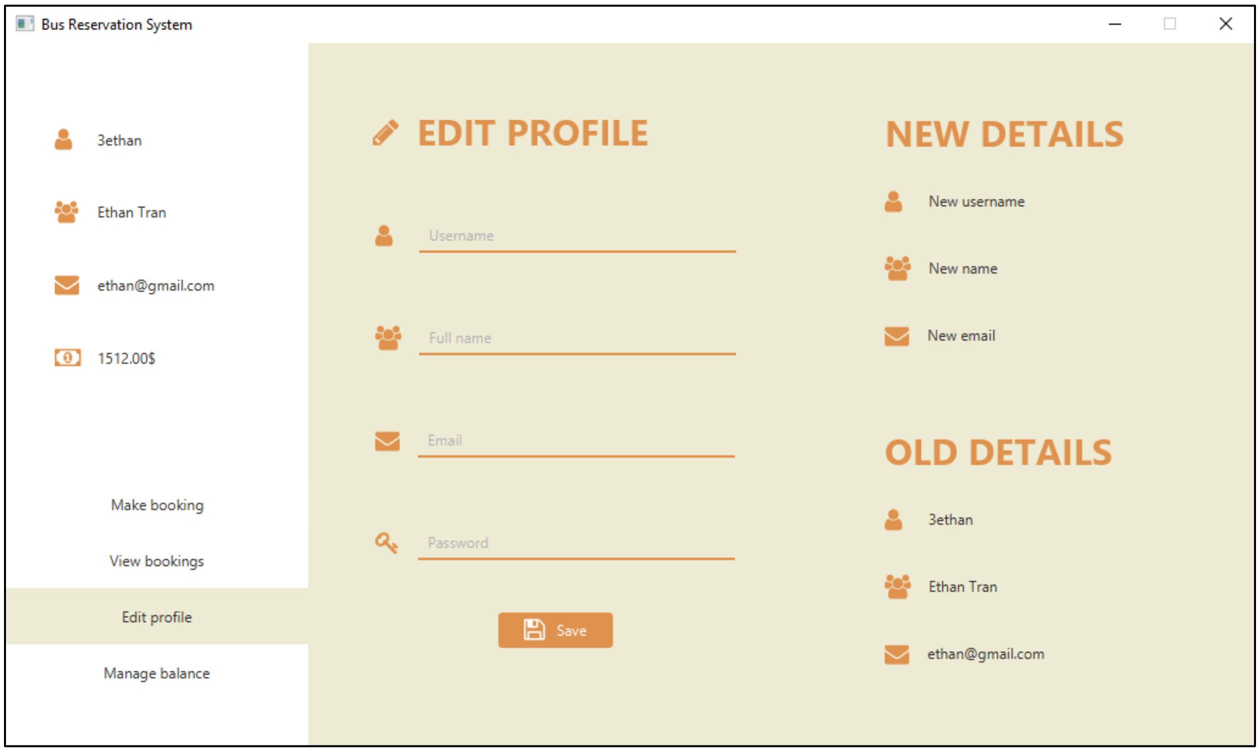
2.5 Client edit profile

This page provides clients with a convenient and user-friendly interface to modify their personal information and account settings within the Bus Reservation System. This page offers various functionalities, empowering clients to make changes to their username, name, password, and email address effortlessly.

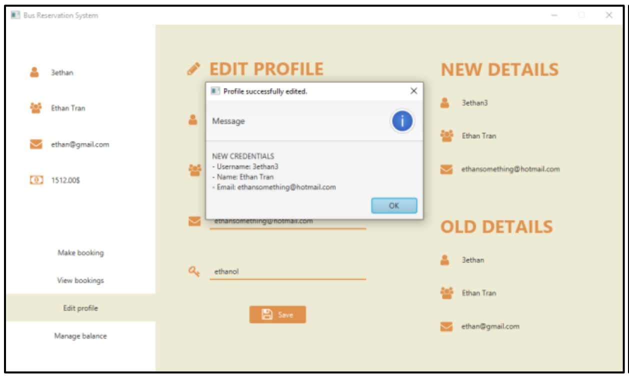
Controller Logic – ClientEditProfile

The ClientEditProfileController class is a controller responsible for managing the profile editing functionality for the client in the bus reservation system. It extends the ClientController class and implements the Initializable interface. The controller initializes the user interface elements and sets the current client's credentials in the appropriate labels. It handles the logic for saving the changes made to the client's profile when the "Save" button is clicked. The save() method retrieves the new credentials entered by the client and updates the client's information in the Database. It also updates all the bookings associated with the current client's username to reflect the changes. If there are any validation errors or exceptions, appropriate error messages are displayed using Alert. The newCredentialsChange() method is called whenever a character is inputted into the text fields. It updates the corresponding labels with the new values as the user types.

2.5.1 Client edit profile page

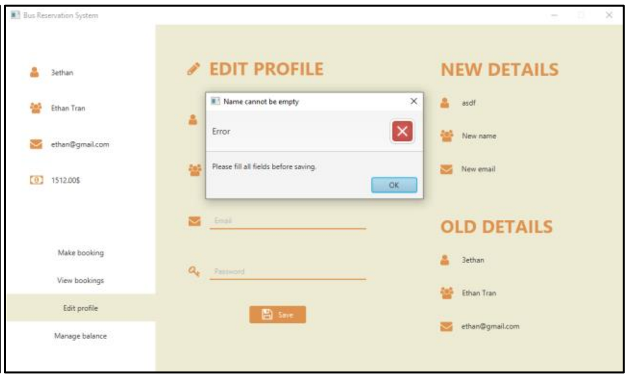


2.5.2 Client edit profile interactions


Profile successfully edited.

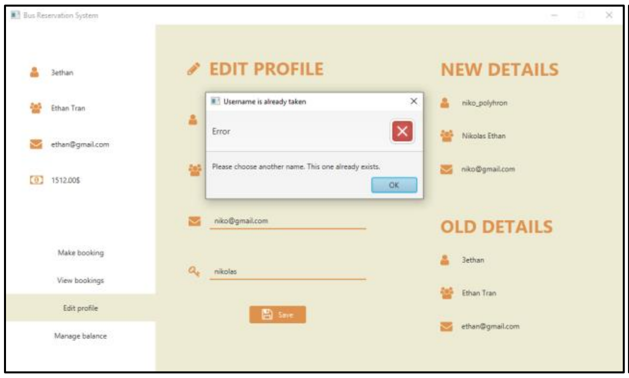
Message

NEW CREDENTIALS
- Username: 3ethan3
- Name: Ethan Tran
- Email: ethansomething@hotmail.com


Name cannot be empty

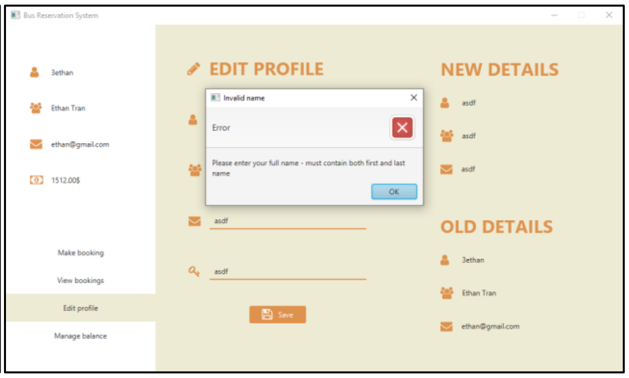
Error

Please fill all fields before saving.


Username is already taken

Error

Please choose another name. This one already exists.


Invalid name

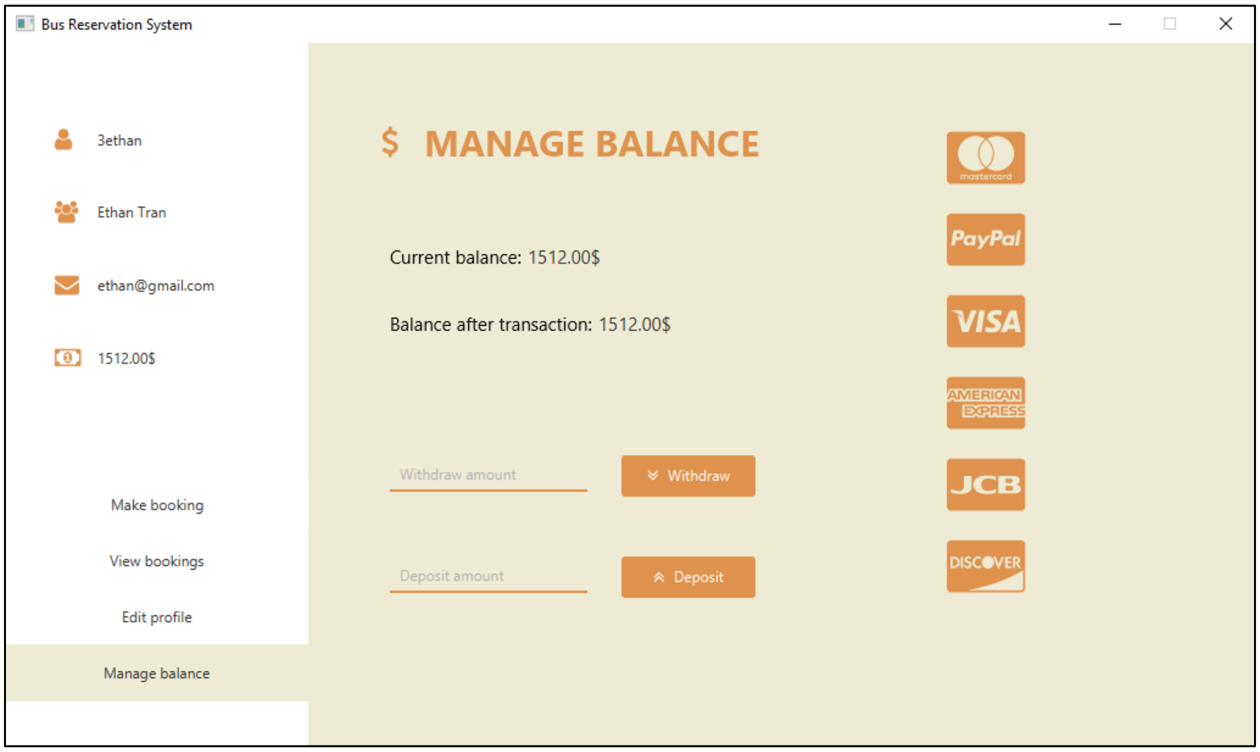
Error

Please enter your full name - must contain both first and last name

2.6 Client manage balance

The ClientManageBalanceController class is a controller responsible for managing the balance of a client in the bus reservation system. It initializes UI elements such as labels for displaying the current and updated balance, and text fields for deposit and withdrawal amounts. The controller updates the displayed balance based on the entered amount and performs transactions (deposit or withdrawal) on the client's balance. It handles errors related to invalid or negative amounts entered and displays a confirmation alert upon successful transactions. The controller allows clients to manage their balance by making deposits or withdrawals.

2.6.1 Client manage balance page


Bus Reservation System

3ethan

Ethan Tran

ethan@gmail.com

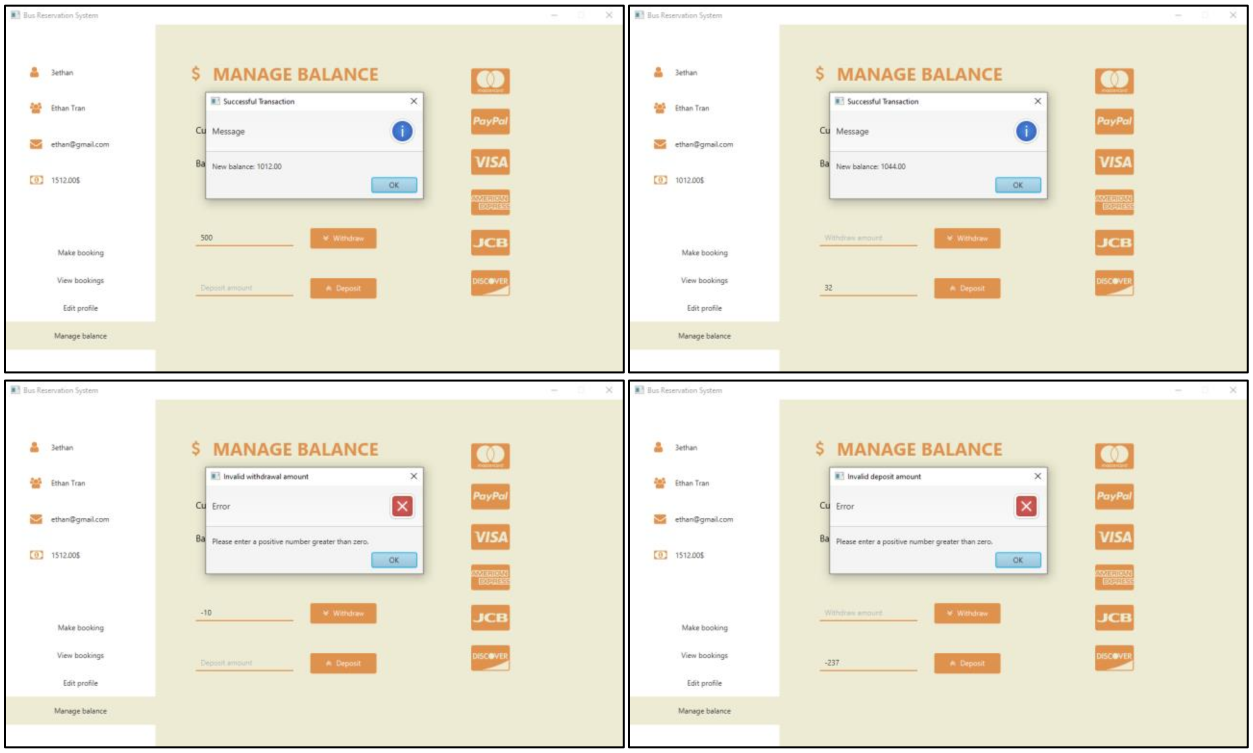
1512.00$

$ MANAGE BALANCE

Current balance: 1512.00$

Balance after transaction: 1512.00$

2.6.2 Client manage balance interaction


Successful Transaction

Message

New balance: 1012.00

Withdraw amount: 500

Successful Transaction

Message

New balance: 1044.00

1012.00$

Deposit amount: 32

Invalid withdrawal amount

Error

Please enter a positive number greater than zero.

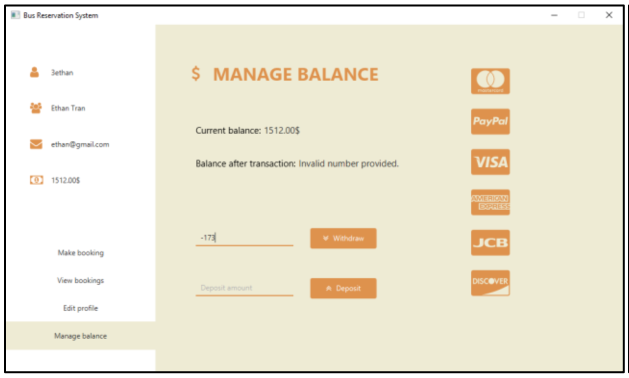
Withdraw amount: -10

Invalid deposit amount

Error

Please enter a positive number greater than zero.

Deposit amount: -237



Bus Reservation System

3ethan

Ethan Tran

ethan@gmail.com

1512.00$

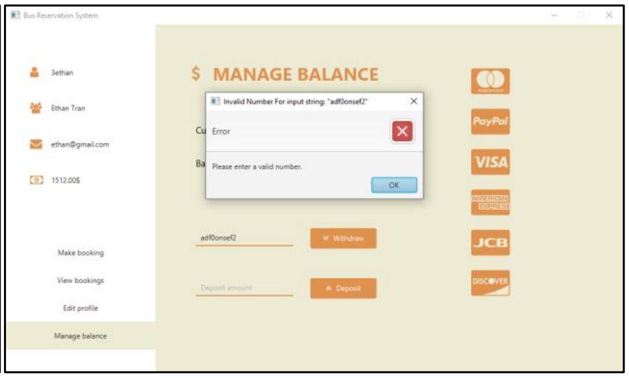
MANAGE BALANCE

Current balance: 1512.00$

Balance after transaction: Invalid number provided.

-173

Deposit amount



Bus Reservation System

3ethan

Ethan Tran

ethan@gmail.com

1512.00$

MANAGE BALANCE

Invalid Number For input string: "adf0onsef2"

Error

Please enter a valid number.

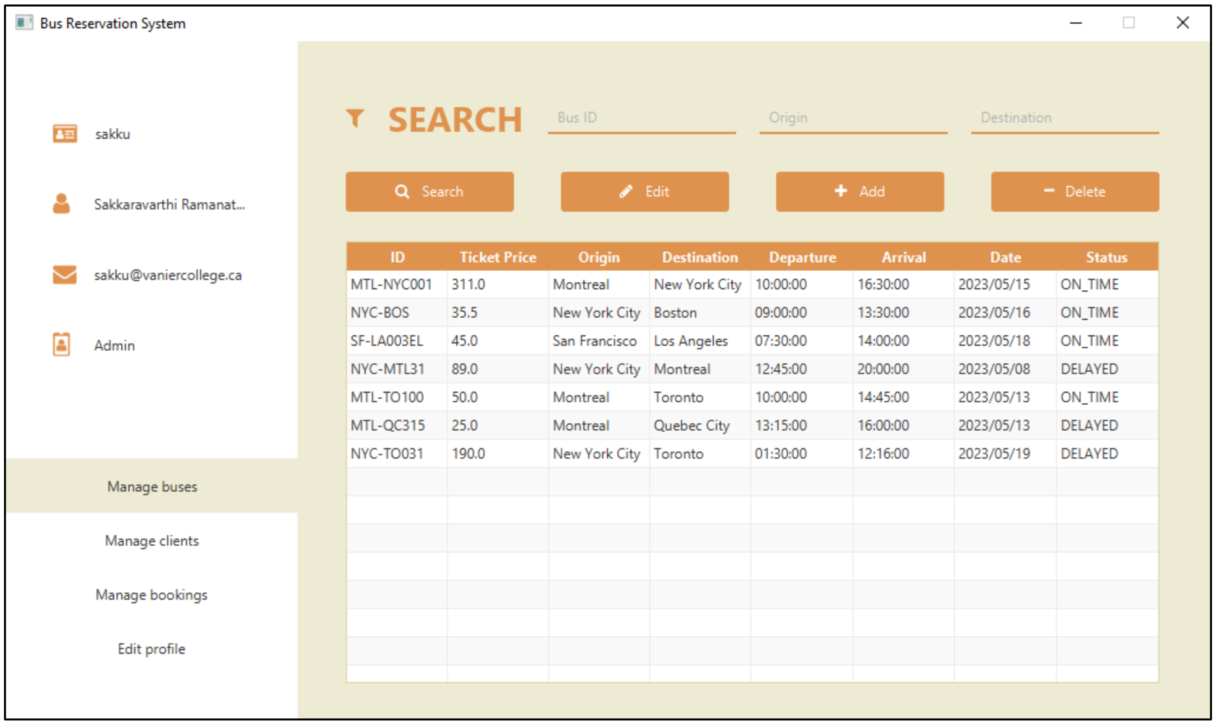
adf0onsef2

Deposit amount

2.7 Admin manage buses

The AdminManageBusesController is a controller class in the Bus Reservation System application. It is responsible for managing and displaying the buses available in the system. The controller initializes the table view and sets up the columns to display bus information, such as the bus ID, ticket price, origin, destination, departure time, arrival time, departure date, and status. It also formats the time and date values using DateTimeFormatter to ensure they are displayed correctly. The table view is populated with the buses retrieved from the database using the Database.getBuses() method. The buses are wrapped in an ObservableList and set as the items of the table view. If a bus is not selected when the admin tries to edit, an alert is displayed to prompt the admin to select a bus first. The AdminManageBusesController interacts with the Database class to retrieve and update the bus information and uses the Bus class to represent individual buses in the system.

2.7.1 Admin manage buses page



Bus Reservation System

sakku

Sakkaravarthi Ramanat...

sakku@vaniercollege.ca

Admin

SEARCH

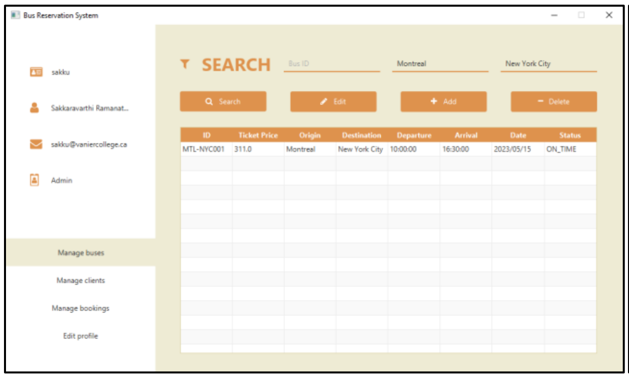
Bus ID

Origin

Destination

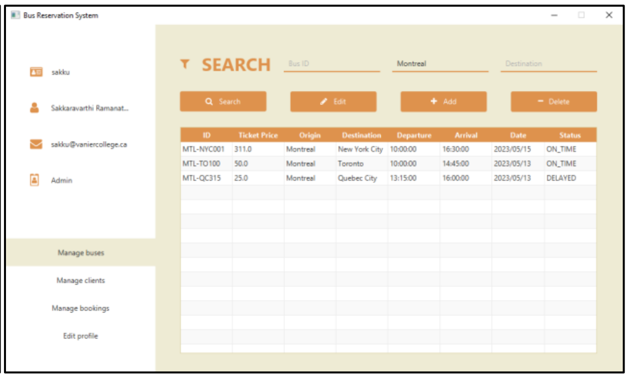
ID	Ticket Price	Origin	Destination	Departure	Arrival	Date	Status
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED

2.7.2 Admin manage buses searching



Montreal
New York City

ID	Ticket Price	Origin	Destination	Departure	Arrival	Date	Status
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME

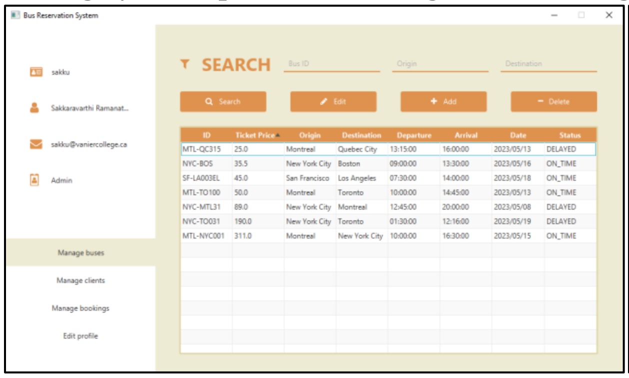


Montreal

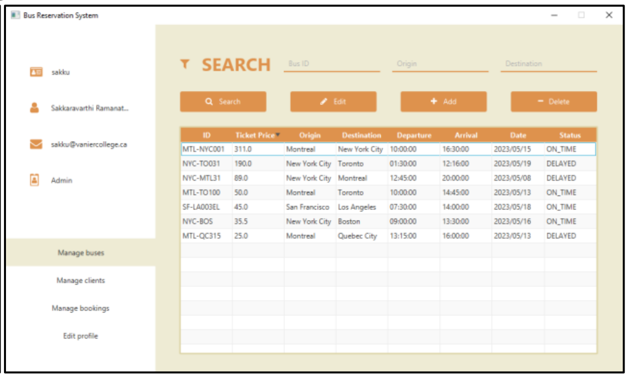
ID	Ticket Price	Origin	Destination	Departure	Arrival	Date	Status
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED

2.7.3 Admin manage buses sorting

Sorting by ticket price – Ascending vs. Descending

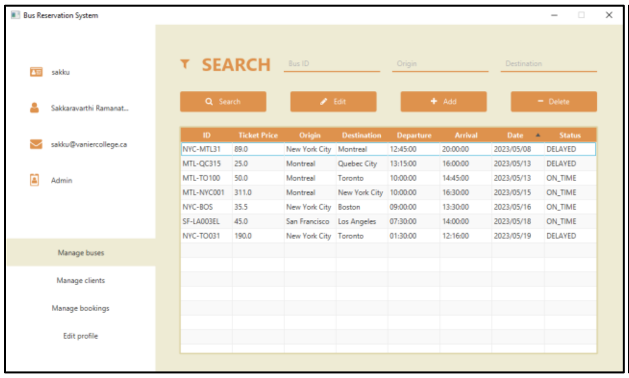


ID	Ticket Price ▲	Origin	Destination	Departure	Arrival	Date	Status
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME

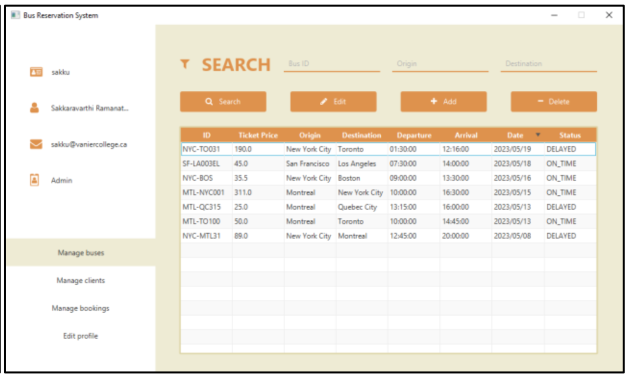


ID	Ticket Price ▼	Origin	Destination	Departure	Arrival	Date	Status
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED

Sorting by date – Ascending vs. Descending



ID	Ticket Price	Origin	Destination	Departure	Arrival	Date ▲	Status
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED



ID	Ticket Price	Origin	Destination	Departure	Arrival	Date ▼	Status
NYC-TO031	190.0	New York City	Toronto	01:30:00	12:16:00	2023/05/19	DELAYED
SF-LA003EL	45.0	San Francisco	Los Angeles	07:30:00	14:00:00	2023/05/18	ON_TIME
NYC-BOS	35.5	New York City	Boston	09:00:00	13:30:00	2023/05/16	ON_TIME
MTL-NYC001	311.0	Montreal	New York City	10:00:00	16:30:00	2023/05/15	ON_TIME
MTL-QC315	25.0	Montreal	Quebec City	13:15:00	16:00:00	2023/05/13	DELAYED
MTL-TO100	50.0	Montreal	Toronto	10:00:00	14:45:00	2023/05/13	ON_TIME
NYC-MTL31	89.0	New York City	Montreal	12:45:00	20:00:00	2023/05/08	DELAYED

2.8 Admin edit & create bus

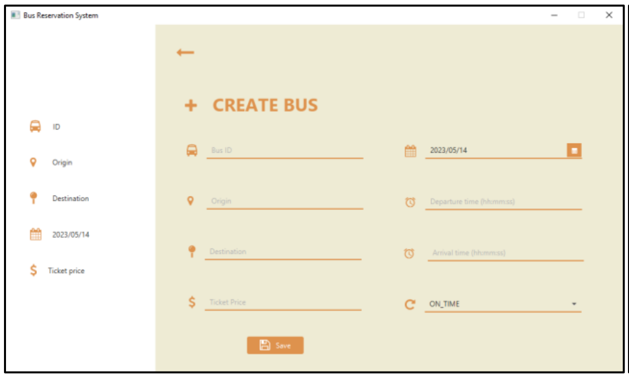
Controller logic: AdminEditBusController

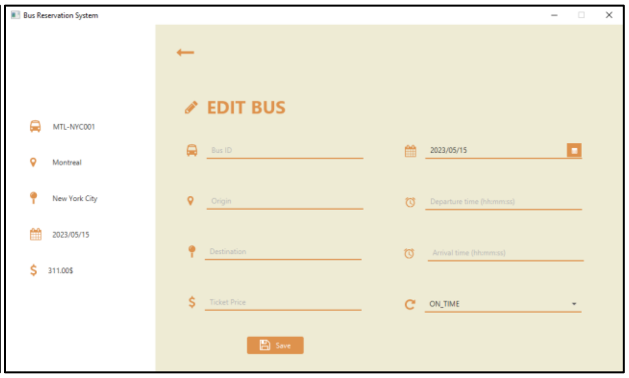
The AdminEditBusController is a controller class in the Bus Reservation System application. It allows the admin to edit bus details such as bus ID, origin, destination, ticket price, departure date, departure time, arrival time, and status. The controller retrieves the current bus object from the database and displays its details in the form fields. The admin can make changes to the bus information and save them, which updates the bus in the database. The controller also provides a navigation option to go back to the admin's buses management page.

Controller logic: AdminCreateBusController

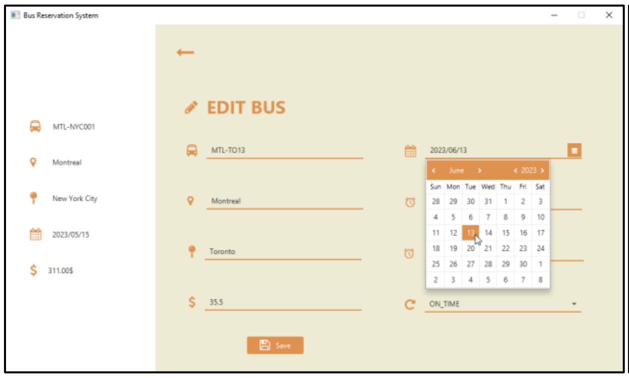
The AdminCreateBusController is responsible for handling the creation of new buses in the bus reservation system. It captures user input, validates the data, creates a Bus object with the entered details, stores it in the database, and provides feedback to the user. Any time a change is made to a bus, corresponding updates will be run through the Database in order to ensure consistency with all of the client's booking tickets.

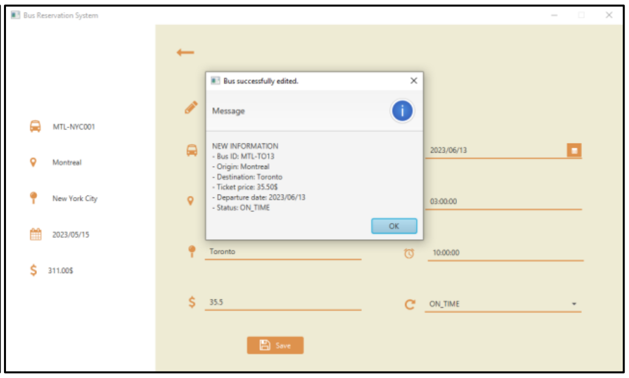
2.8.1 Admin edit and create bus pages





2.8.2 Admin edit bus interactions



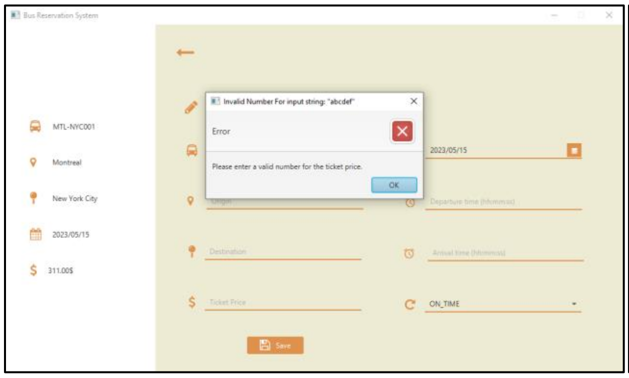


Bus successfully edited.

Message

NEW INFORMATION
- Bus ID: MTL-TO13
- Origin: Montreal
- Destination: Toronto
- Ticket price: 35.50$
- Departure date: 2023/06/13
- Status: ON_TIME

OK

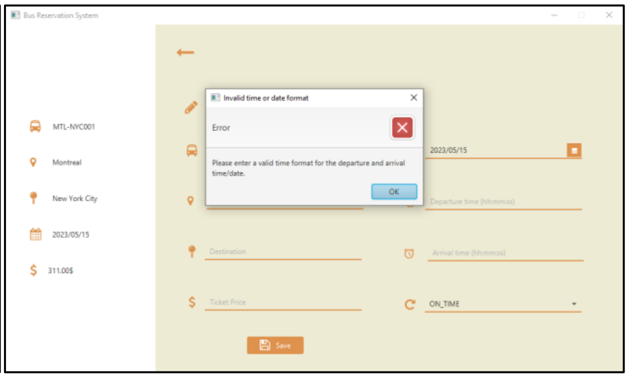


Invalid Number For input string: "abcdef"

Error

Please enter a valid number for the ticket price.

OK



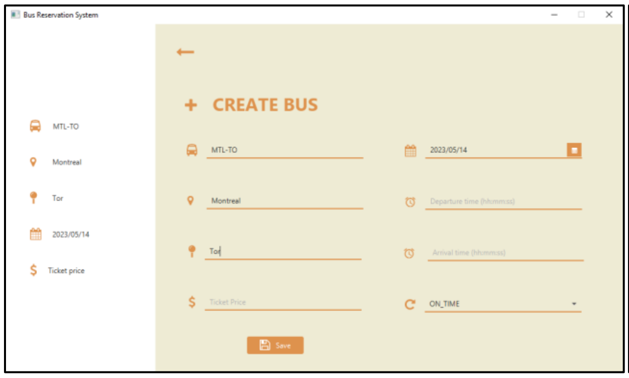
Invalid time or date format

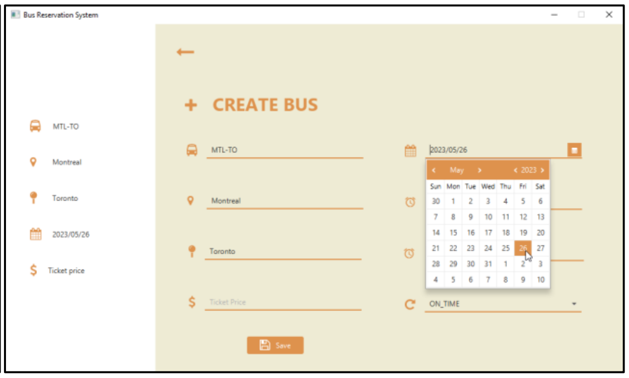
Error

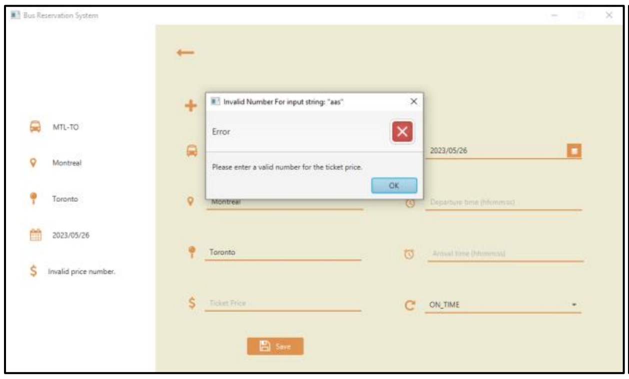
Please enter a valid time format for the departure and arrival time/date.

OK

2.8.3 Admin create bus interactions





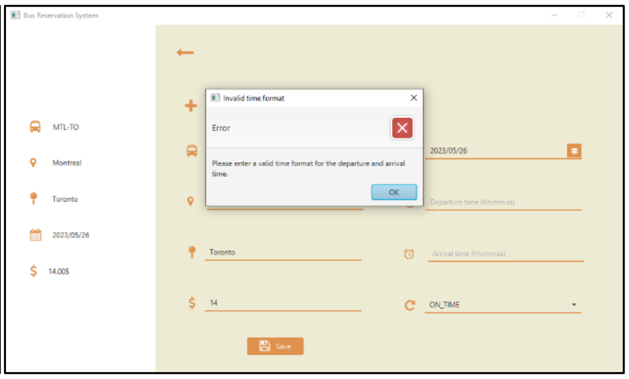


Invalid Number For input string: "aas"

Error

Please enter a valid number for the ticket price.

OK



Invalid time format

Error

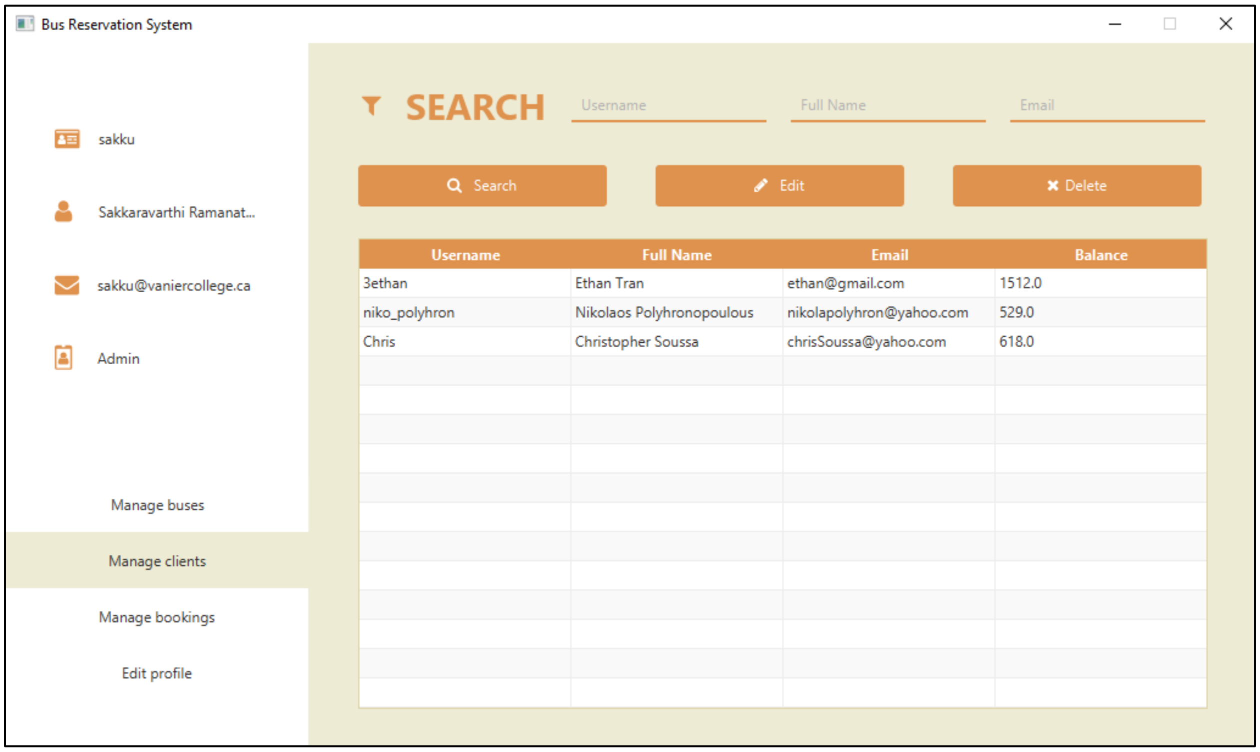
Please enter a valid time format for the departure and arrival time.

OK

2.9 Admin manage clients

The AdminManageClientsController is another controller class in the Bus Reservation System application. It is responsible for managing and displaying the clients registered in the system. The controller initializes the table view and sets up the columns to display client information, such as the username, full name, email, and balance. It uses the PropertyValueFactory to specify which properties of the Client class should be used to populate each column. The table view is populated with the clients retrieved from the database using the Database.getClients() method. The clients are wrapped in an ObservableList and set as the items of the table view. The controller provides functionality for searching clients based on the provided attributes: username, full name, and email. If a client is not selected when the admin tries to delete or edit, an alert is displayed to prompt the admin to select a client first. The AdminManageClientsController interacts with the Database class to retrieve and update the client information and uses the Client class to represent individual clients in the system.

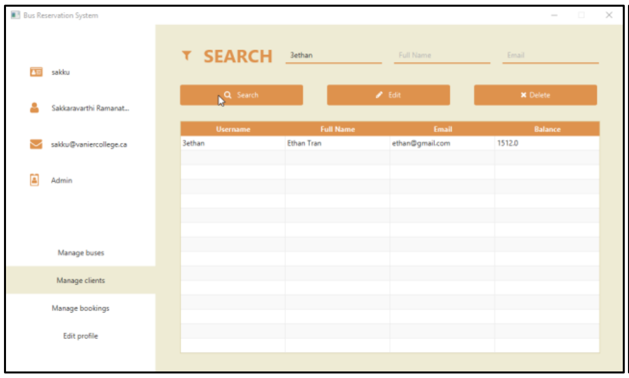
2.9.1 Admin manage clients page



sakku
Sakkaravarthi Ramanat...
sakku@vaniercollege.ca
Admin

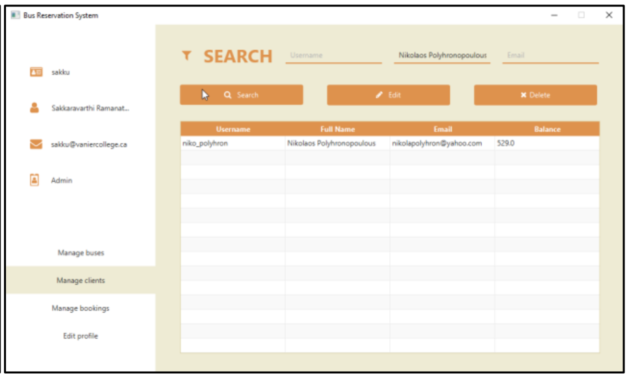
Username	Full Name	Email	Balance
3ethan	Ethan Tran	ethan@gmail.com	1512.0
niko_polyhron	Nikolaos Polyhronopoulous	nikolapolyhron@yahoo.com	529.0
Chris	Christopher Soussa	chrisSoussa@yahoo.com	618.0

2.9.2 Admin manage clients searching



Username: 3ethan

Username	Full Name	Email	Balance
3ethan	Ethan Tran	ethan@gmail.com	1512.0

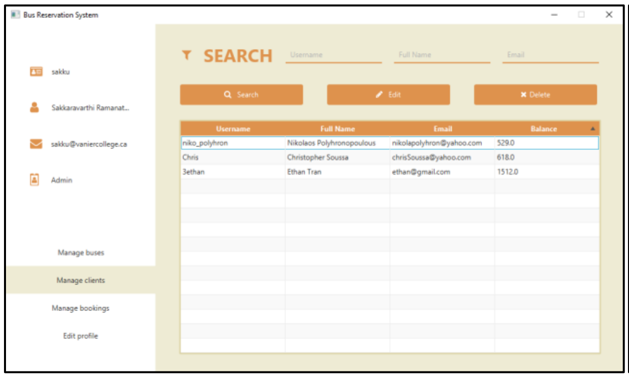


Full Name: Nikolaos Polyhronopoulous

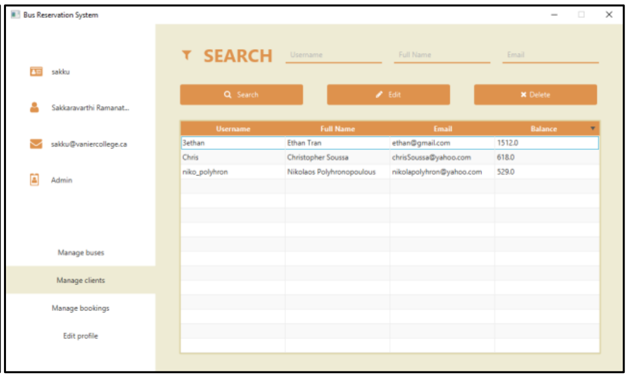
Username	Full Name	Email	Balance
niko_polyhron	Nikolaos Polyhronopoulous	nikolapolyhron@yahoo.com	529.0

2.9.3 Admin manage clients sorting

Sorting by balance – Ascending vs. Descending

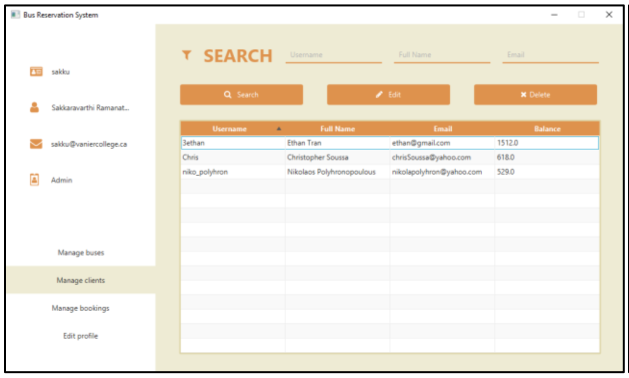


Username	Full Name	Email	Balance ▲
niko_polyhron	Nikolaos Polyhronopoulous	nikolapolyhron@yahoo.com	529.0
Chris	Christopher Soussa	chrisSoussa@yahoo.com	618.0
3ethan	Ethan Tran	ethan@gmail.com	1512.0

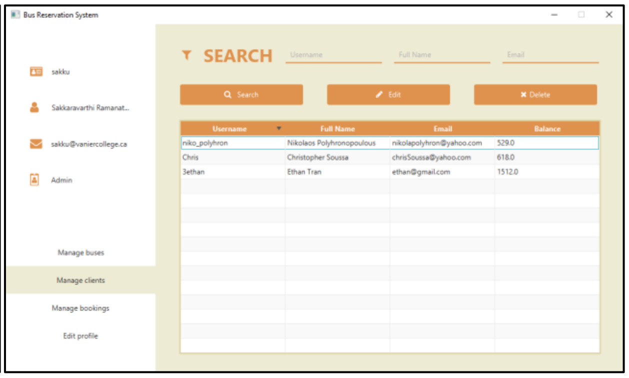


Username	Full Name	Email	Balance ▼
3ethan	Ethan Tran	ethan@gmail.com	1512.0
Chris	Christopher Soussa	chrisSoussa@yahoo.com	618.0
niko_polyhron	Nikolaos Polyhronopoulous	nikolapolyhron@yahoo.com	529.0

Sorting by username lexicographically – Ascending vs. Descending

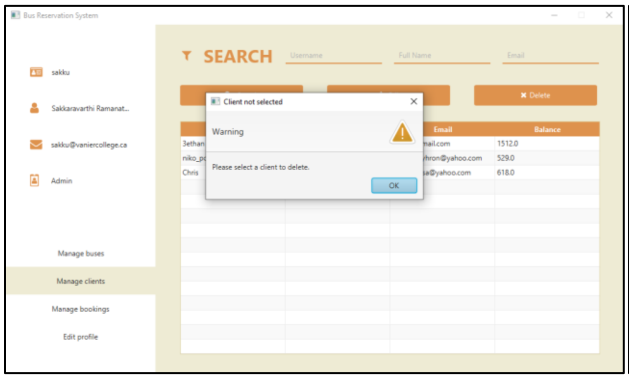


Username ▲	Full Name	Email	Balance
3ethan	Ethan Tran	ethan@gmail.com	1512.0
Chris	Christopher Soussa	chrisSoussa@yahoo.com	618.0
niko_polyhron	Nikolaos Polyhronopoulous	nikolapolyhron@yahoo.com	529.0



Username ▼	Full Name	Email	Balance
niko_polyhron	Nikolaos Polyhronopoulous	nikolapolyhron@yahoo.com	529.0
Chris	Christopher Soussa	chrisSoussa@yahoo.com	618.0
3ethan	Ethan Tran	ethan@gmail.com	1512.0

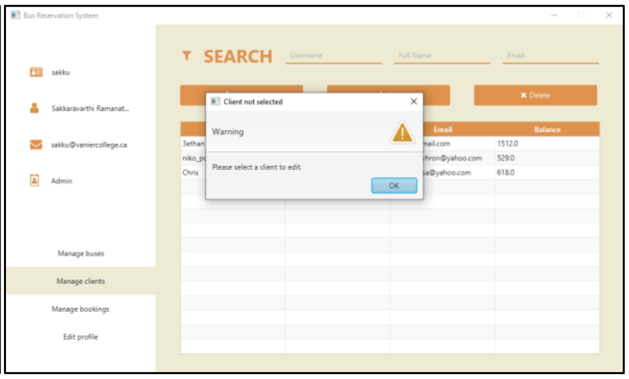
2.9.4 Admin manage clients interactions



Client not selected

Warning

Please select a client to delete.



Client not selected

Warning

Please select a client to edit.

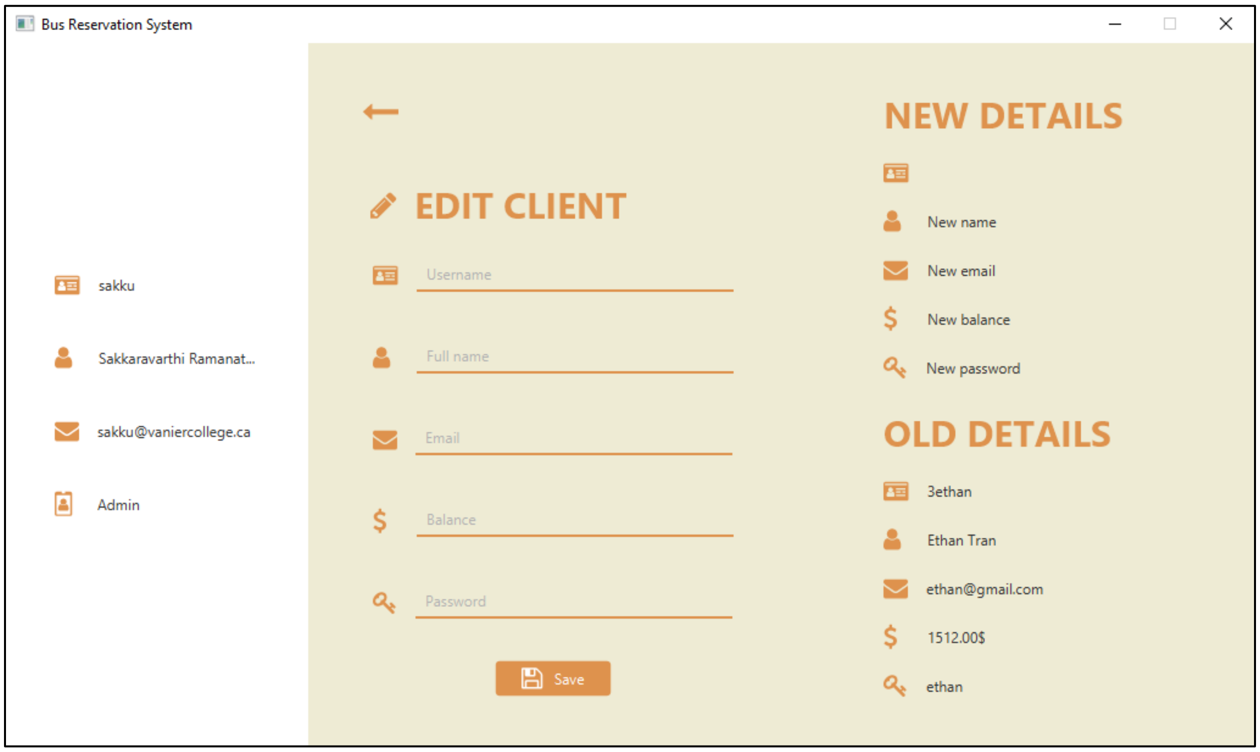
2.10 Admin edit client

This page is a crucial component of the Bus Reservation System, designed specifically for administrators to efficiently manage and update client information. This page empowers admins with the necessary tools to modify essential client details, including username, name, password, and email address. By providing these functionalities, the system ensures accurate and up-to-date client information within the database.

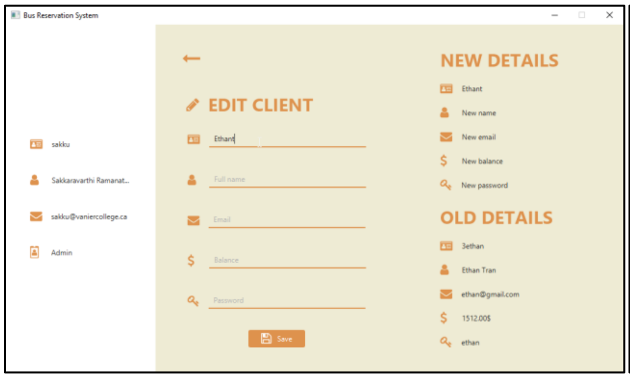
Controller Logic - AdminEditClientController

The AdminEditClientController is a controller class in the Bus Reservation System application. It allows the admin to edit client details such as username, name, email, balance, and password. The controller retrieves the current client object from the database and displays its details in the form fields. The admin can make changes to the client information and save them, which updates the client in the database. Any time a change to the client is made, corresponding updates will be run through the database in order to ensure consistency with this client's bookings. The controller also provides a navigation option to go back to the admin's clients management page.

2.10.1 Admin edit client page



2.10.2 Admin edit client interactions

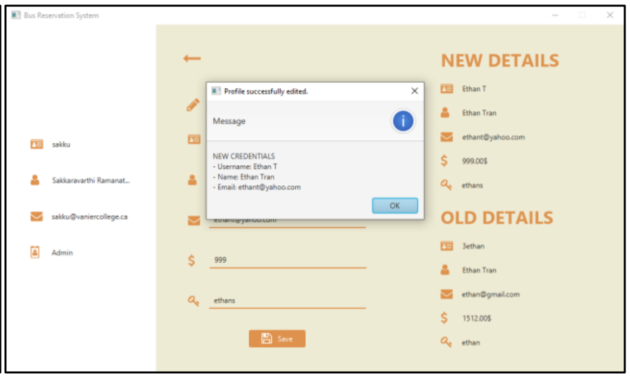


sakku
Sakkaravarthi Ramanat...
sakku@vaniercollege.ca
Admin

Ethant
Full name
Email
Balance
Password

NEW DETAILS
Ethant
New name
New email
New balance
New password

OLD DETAILS
3ethan
Ethan Tran
ethan@gmail.com
1512.00$
ethan



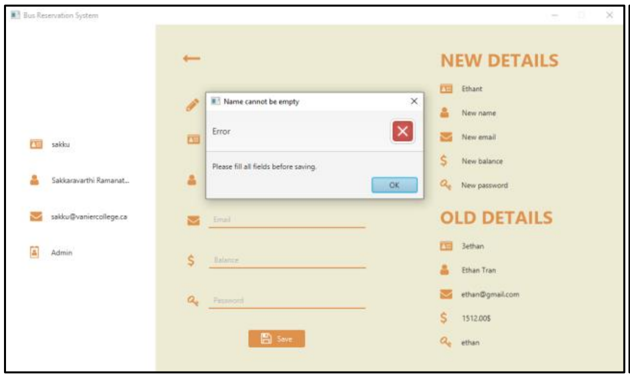
Profile successfully edited.
Message
NEW CREDENTIALS
- Username: Ethan T
- Name: Ethan Tran
- Email: ethant@yahoo.com

sakku
Sakkaravarthi Ramanat...
sakku@vaniercollege.ca
Admin

999
ethans

NEW DETAILS
Ethan T
Ethan Tran
ethant@yahoo.com
999.00$
ethans

OLD DETAILS
3ethan
Ethan Tran
ethan@gmail.com
1512.00$
ethan



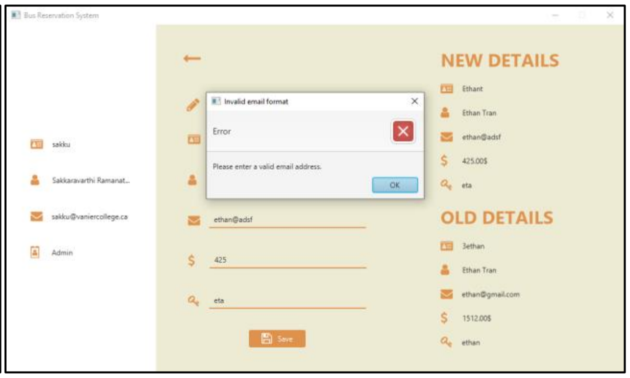
Name cannot be empty
Error
Please fill all fields before saving.

sakku
Sakkaravarthi Ramanat...
sakku@vaniercollege.ca
Admin

Email
Balance
Password

NEW DETAILS
Ethant
New name
New email
New balance
New password

OLD DETAILS
3ethan
Ethan Tran
ethan@gmail.com
1512.00$
ethan



Invalid email format
Error
Please enter a valid email address.

sakku
Sakkaravarthi Ramanat...
sakku@vaniercollege.ca
Admin

ethan@adsf
425
eta

NEW DETAILS
Ethant
Ethan Tran
ethan@adsf
425.00$
eta

OLD DETAILS
3ethan
Ethan Tran
ethan@gmail.com
1512.00$
ethan

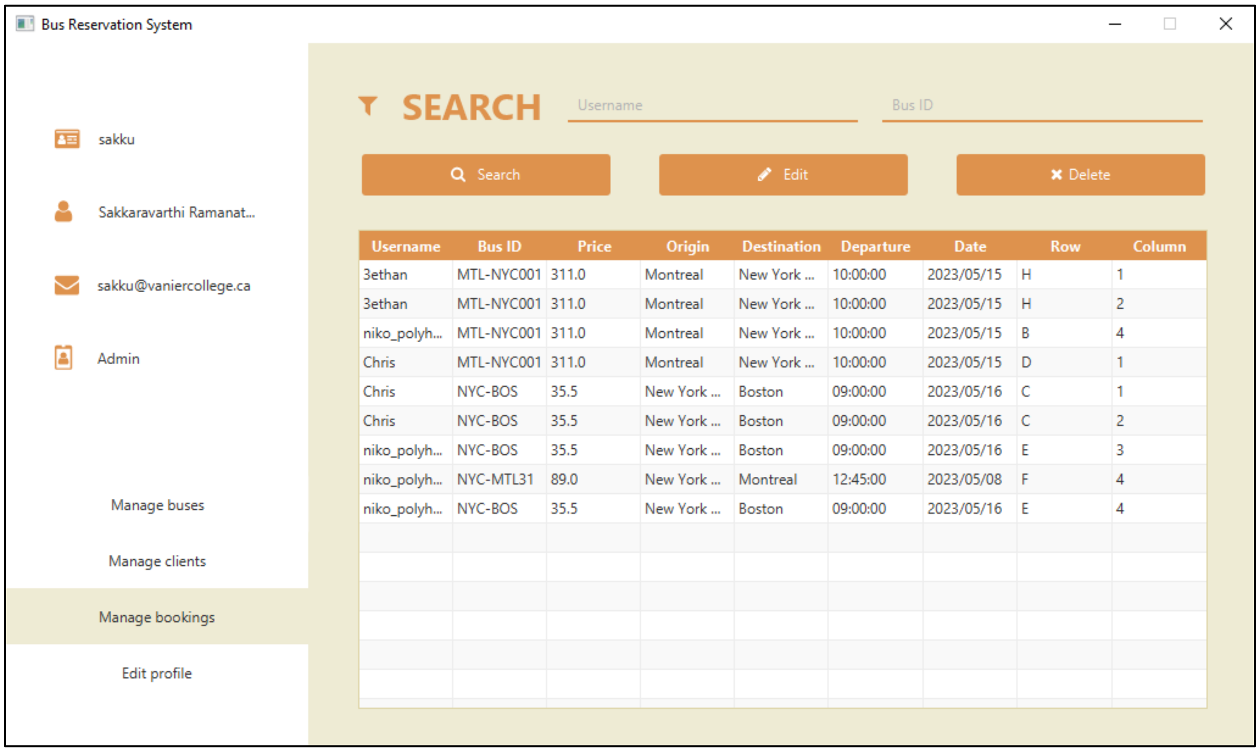
2.11 Admin manage bookings

This page gives administrators comprehensive control over client bookings. This page enables administrators to efficiently manage, update and view booking information, including usernames, bus ID, price, origin, destination, departure time, date, row, and column. With these functionalities, administrators can ensure accurate and up-to-date user data, enhancing system security and providing a seamless experience for both administrators and clients.

Controller Logic - AdminManageBookingsController

The AdminManageBookingsController is a controller class in the Bus Reservation System application. It is responsible for managing and displaying the bookings made by clients. The controller initializes the table view and sets up the columns to display the booking information, such as the client's username, bus ID, price, origin, destination, departure date, departure time, row, and column. It also formats the date and time values using DateTimeFormatter to ensure they are displayed correctly. If a booking is not selected when the admin tries to edit or cancel, an alert is displayed to prompt the admin to select a booking first. The AdminManageBookingsController interacts with the Database class to retrieve and update the bookings information, as well as the Admin and Booking classes to perform various operations.

2.11.1 Admin manage bookings page



Bus Reservation System

SEARCH Username Bus ID

Search Edit Delete

sakku

Sakkaravarthi Ramanat...

sakku@vaniercollege.ca

Admin

Manage buses

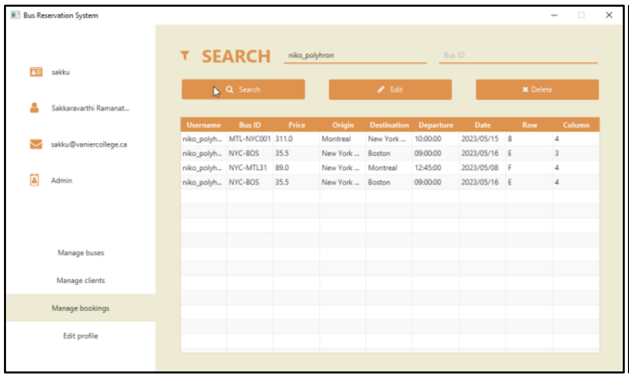
Manage clients

Manage bookings

Edit profile

Username	Bus ID	Price	Origin	Destination	Departure	Date	Row	Column
3ethan	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	H	1
3ethan	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	H	2
niko_polyh...	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	B	4
Chris	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	D	1
Chris	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	C	1
Chris	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	C	2
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	3
niko_polyh...	NYC-MTL31	89.0	New York ...	Montreal	12:45:00	2023/05/08	F	4
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	4

2.11.2 Admin manage bookings searching

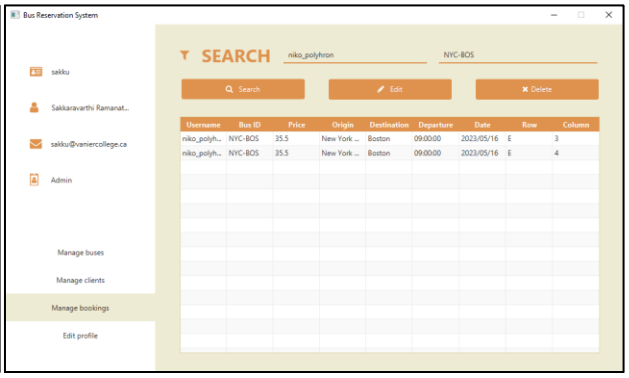


Bus Reservation System

SEARCH niko_polyhron Bus ID

Search Edit Delete

Username	Bus ID	Price	Origin	Destination	Departure	Date	Row	Column
niko_polyh...	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	B	4
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	3
niko_polyh...	NYC-MTL31	89.0	New York ...	Montreal	12:45:00	2023/05/08	F	4
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	4



Bus Reservation System

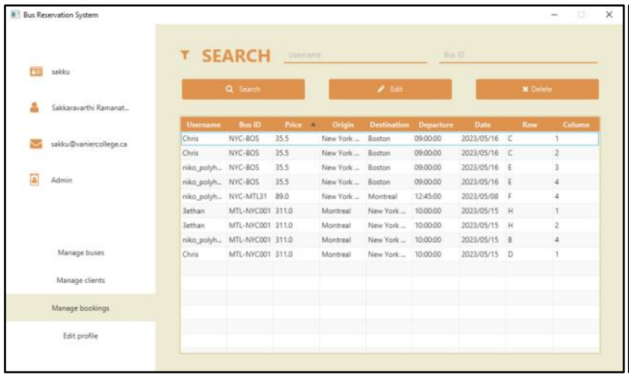
SEARCH niko_polyhron NYC-BOS

Search Edit Delete

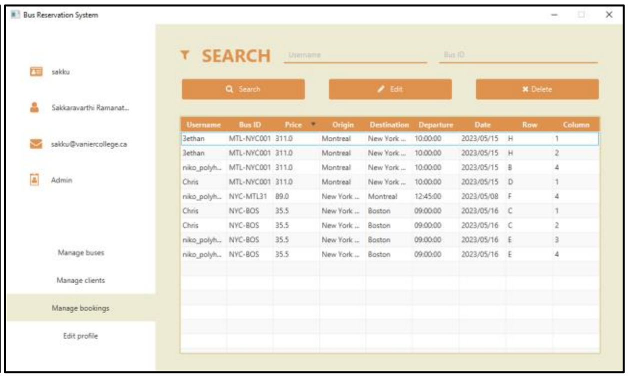
Username	Bus ID	Price	Origin	Destination	Departure	Date	Row	Column
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	3
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	4

2.11.3 Admin manage bookings sorting

Sorting by ticket price – Ascending vs. Descending

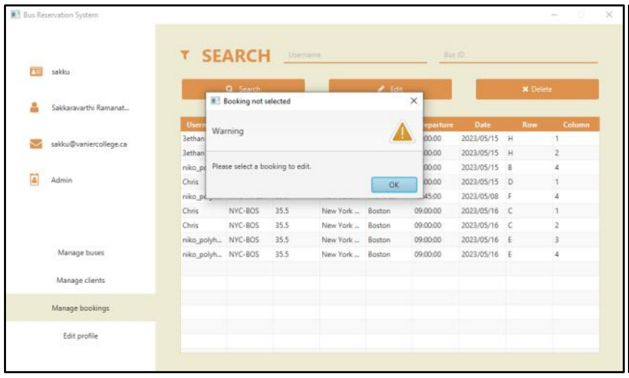


Username	Bus ID	Price	Origin	Destination	Departure	Date	Row	Column
Chris	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	C	1
Chris	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	C	2
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	3
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	4
niko_polyh...	NYC-MTL31	89.0	New York ...	Montreal	12:45:00	2023/05/08	F	4
3ethan	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	H	1
3ethan	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	H	2
niko_polyh...	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	B	4
Chris	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	D	1



Username	Bus ID	Price	Origin	Destination	Departure	Date	Row	Column
3ethan	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	H	1
3ethan	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	H	2
niko_polyh...	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	B	4
Chris	MTL-NYC001	311.0	Montreal	New York ...	10:00:00	2023/05/15	D	1
niko_polyh...	NYC-MTL31	89.0	New York ...	Montreal	12:45:00	2023/05/08	F	4
Chris	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	C	1
Chris	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	C	2
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	3
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	4

2.11.4 Admin manage bookings interactions

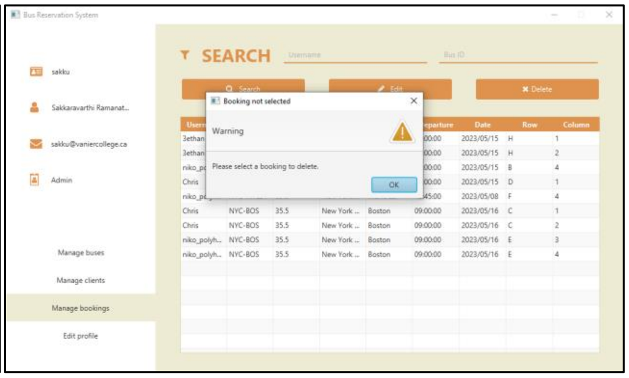


Booking not selected

Warning

Please select a booking to edit.

Usern					eparture	Date	Row	Column
3ethan					00:00	2023/05/15	H	1
3ethan					00:00	2023/05/15	H	2
niko_pc					00:00	2023/05/15	B	4
Chris					00:00	2023/05/15	D	1
niko_pc					45:00	2023/05/08	F	4
Chris	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	C	1
Chris	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	C	2
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	3
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	4



Booking not selected

Warning

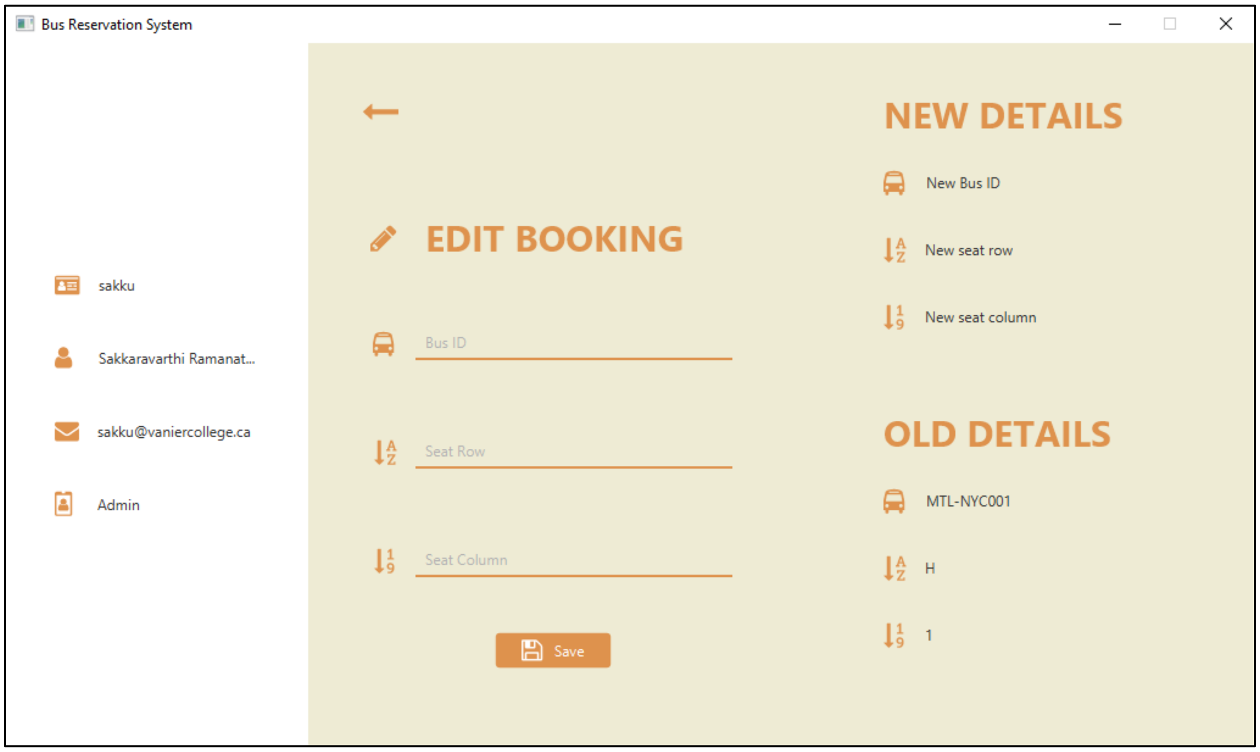
Please select a booking to delete.

Usern					eparture	Date	Row	Column
3ethan					00:00	2023/05/15	H	1
3ethan					00:00	2023/05/15	H	2
niko_pc					00:00	2023/05/15	B	4
Chris					00:00	2023/05/15	D	1
niko_pc					45:00	2023/05/08	F	4
Chris	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	C	1
Chris	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	C	2
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	3
niko_polyh...	NYC-BOS	35.5	New York ...	Boston	09:00:00	2023/05/16	E	4

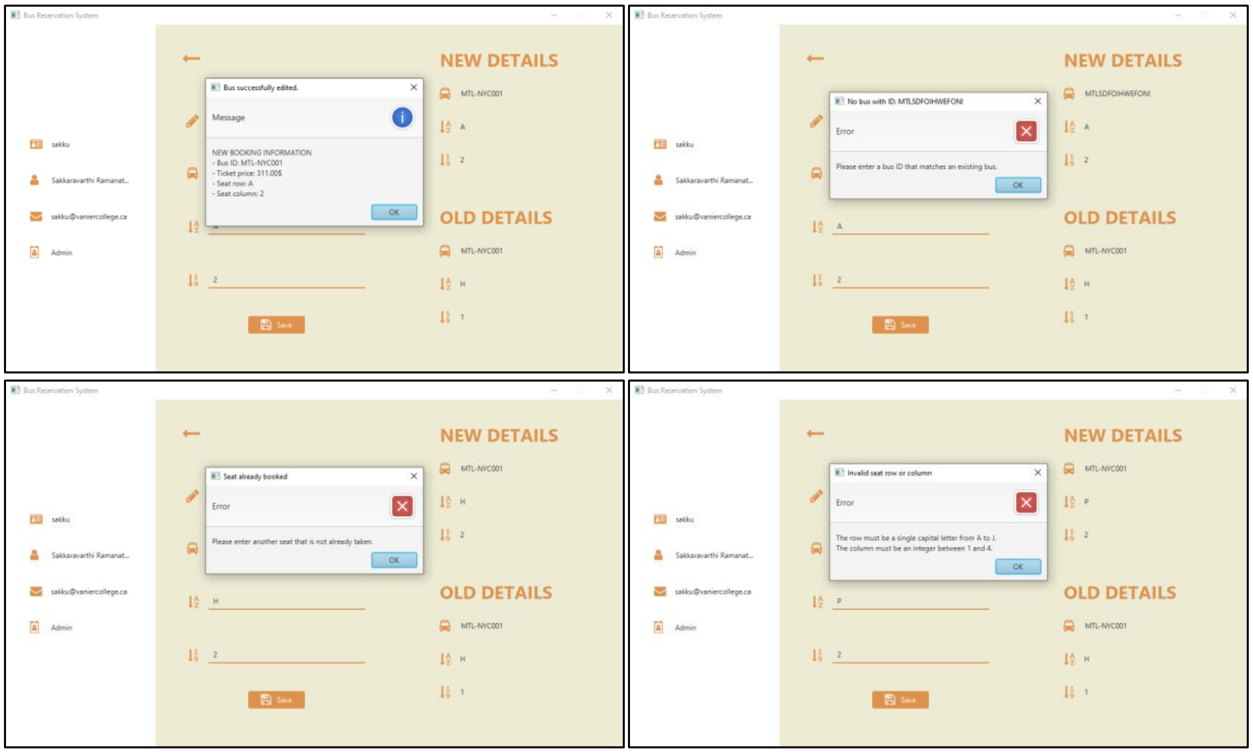
2.12 Admin edit booking

The AdminEditBookingController is a controller class in the Bus Reservation System application. The controller retrieves the existing booking information from the database and displays it. The admin can make changes to the booking and save them, which updates the booking in the database. The controller also provides navigation options to go back to the admin's bookings management page.

2.12.1 Admin edit booking page



2.12.2 Admin edit booking interactions



Bus successfully edited.

Message

NEW BOOKING INFORMATION
- Bus ID: MTL-NYC001
- Ticket price: 311.00$
- Seat row: A
- Seat column: 2

OK

No bus with ID: MTLSDFOIHWEFONI

Error

Please enter a bus ID that matches an existing bus.

OK

Seat already booked

Error

Please enter another seat that is not already taken.

OK

Invalid seat row or column

Error

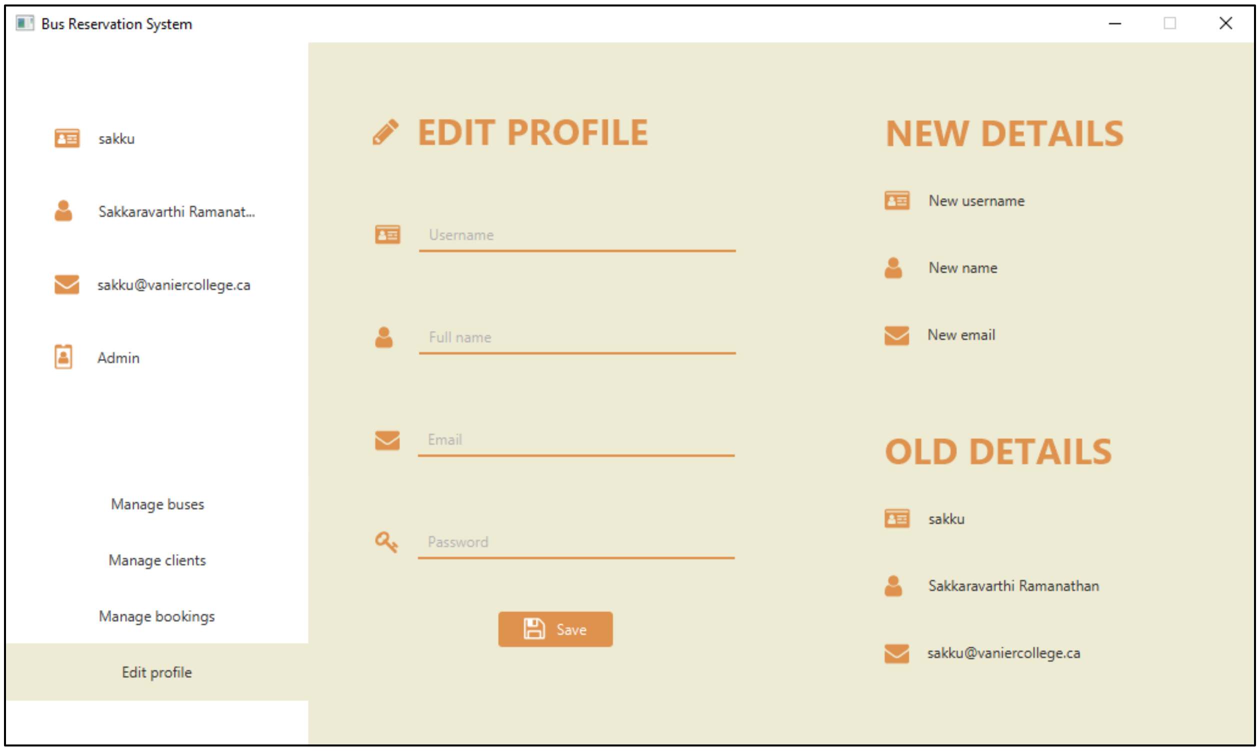
The row must be a single capital letter from A to J.
The column must be an integer between 1 and 4.

OK

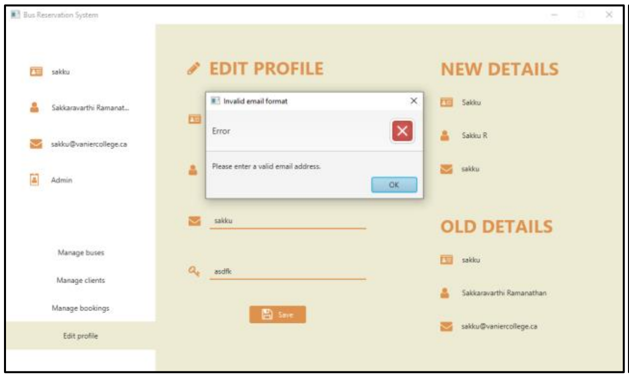
2.13 Admin edit profile

The AdminEditProfileController is a controller class in the Bus Reservation System application. It is responsible for handling the editing of the admin's profile information. The controller retrieves the current admin object from the database and displays its details in the corresponding labels. The admin can make changes to the username, full name, email, and password fields. When the admin clicks the "Save" button, the controller validates the input and updates the admin's profile information in the database. If any validation errors occur, such as an invalid email format or a username that is already taken, an appropriate error message is displayed using the Alert class. The controller also handles the text change events for the text fields, updating the corresponding labels based on the field that has been changed. After saving the changes or encountering an error, the controller reloads the page to update the left panel with the updated profile information.

2.13.2 Admin edit profile page



2.13.2 Admin edit profile interactions



Invalid email format

Error

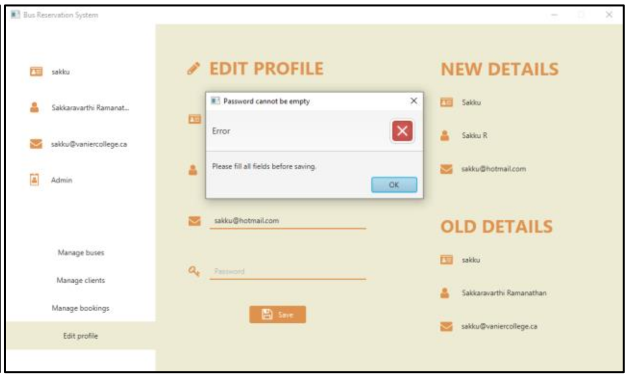
Please enter a valid email address.

Email: sakku

Password: asdfk

New details: Sakku / Sakku R / sakku

Old details: sakku / Sakkaravarthi Ramanathan / sakku@vaniercollege.ca



Password cannot be empty

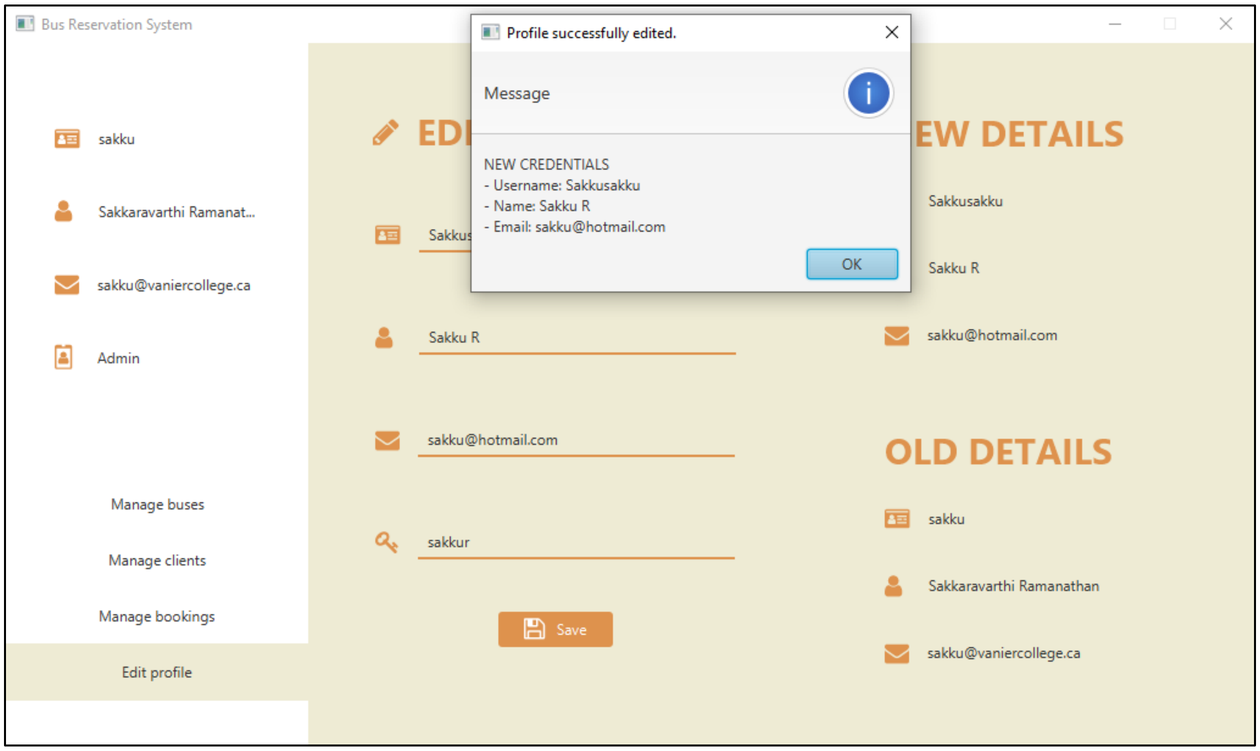
Error

Please fill all fields before saving.

Email: sakku@hotmail.com

New details: Sakku / Sakku R / sakku@hotmail.com

Old details: sakku / Sakkaravarthi Ramanathan / sakku@vaniercollege.ca


Bus Reservation System

Profile successfully edited.

Message

NEW CREDENTIALS
- Username: Sakkusakku
- Name: Sakku R
- Email: sakku@hotmail.com

3 Meeting Project Requirements

Classes - there are more than 4 classes, including, but not limited to:

- User: Abstract class representing a user in the system. (Abstract class)
- Admin: Class representing an administrator user in the system.
- Client: Class representing a customer user in the system.
- Bus: Class representing a bus entity in the system.
- Booking: Class representing a booking made by a client.

Variables - Each class has its own set of variables, including primitive data types and String data type. They also include static, instance, and final variables as applicable. Examples include, but not limited to:

- User class has variables fullName, username, password, email.
- Bus class has variables busNumber, capacity, driverName.
- Booking class has variables bookingId, client, bus, status.

Methods - Each class has multiple methods with different signatures, including constructors, getters, setters, and other utility methods. They have different parameters and return types, and can be static or instance methods. Examples include, but are not limited to:

- User class has getters and setters for fullName, username, password, email.
- Bus class has a constructor, getters and setters for busNumber, capacity, driverName.
- Booking class has constructors, getters, setters, and other utility methods like cancelBooking().

Interfaces - Transactional interface in the com.busreservationsystem.system package defines the methods for deposit, withdraw, and getBalance in the User class.

OOP related concepts:

- Classes have no-arg and parameterized constructors, and usage of getters/setters where applicable.
- At least one class has a static final variable, such as the Database class.
- Different access modifiers are used for class members (public, private, protected).
- Hierarchical inheritance: User class is inherited by Admin and Client classes. ClientController is inherited by ClientEditProfileController, ClientMakeBookingsController, ClientManageBalanceController, ClientSeatSelectionController, ClientViewBookingsController.
- Multiple inheritance, e.g.: Client class extends User class and implements Transactional interface, ClientMakeBookingsController class extends ClientController and implements Initializable.
- Polymorphism: Method overloading and overriding are demonstrated in various classes, like Client, which inherits from User and implements the getType() method.
- Abstraction: User, AdminController, ClientController classes are abstract classes, and Transactional is an interface.
- super keyword is used in subclasses to access variables or methods from the super class.
- super() keyword is used in constructors to invoke no-argument or parameterized constructors from the super class.
- At least one final variable or method is used in the Database class.

Data structures:

- ArrayLists are used in the Database class to store and process program data for clients, admins, buses, and bookings.
- JavaFX ObservableLists are used in JavaFX TableView to display data in the GUI.

Exception Handling - Exceptions are handled where applicable throughout the project, such as handling NoSuchElementExceptions when searching for a Bus with a particular ID, when reading numbers from the GUI (NumberFormatException), or when reading a LocalTime data entry for a Bus (DateTimeParseException). Error handling can be found in methods like setFullName() and setUsername() in the User class.

File I/O - File I/O is used in the Database class to load and write data from/to JSON files.

Sorting Algorithms:

- JavaFX uses the TimSort algorithm for sorting the TableView when you click on a column header to sort the data. TimSort is a hybrid sorting algorithm derived from merge sort and insertion sort. It is designed to perform well on many kinds of real-world data and provides stable sorting.
- Best-case and average-case time complexity: The best-case time complexity of TimSort is $O(n)$, where n is the number of elements to be sorted. This occurs when the input data is already partially sorted or when the data has a lot of presorted subsequences. The average-case time complexity is also $O(n \log n)$, making it efficient for a wide range of scenarios.
- This algorithm is used every single time a TableView is displayed in the GUI. Clients and Admins will be able to sort the data according to any field by clicking the column headers.

Searching Algorithms:

- Controllers such as ClientMakeBookingsController and AdminManageBusesController contain a method to search (filter) for a bus based on destination and other factors. In this case, since the data is in an unsorted list (there is no guarantee that the user has sorted the data), we use linear search to search, and filter elements from the ArrayList. This algorithm is used every time a TableView is displayed along with searching fields.
- In the worst-case scenario, linear search has a time complexity of O(n), where n is the number of elements in the list. This means that the time taken to perform the search grows linearly with the size of the list. In the average and best-case scenarios, linear search still has a linear time complexity. This makes linear search inefficient for large lists or datasets, as the search time increases proportionally with the number of elements; however, in the case of this project it proved to be largely sufficient.

4 Challenges

4.1 Difficulties & obstacles

JavaFX is a comprehensive and feature-rich framework for building desktop and rich internet applications, which means there is a lot to learn in addition to the steep learning curve. Unfortunately, we were unable to find any good reliable source of information regarding how to do specific things. Even for simple things like switching between scenes would prove to be quite complicated given the project structure. No tutorials or books were thorough in their explanation – we had to search through many different sources of information at every step of the development process, even for seemingly basic things. In addition, the framework encompasses various concepts, including UI controls, layouts, event handling, animations, CSS styling, and more. Mastering all these components requires time and effort, and given the short time given for this project, we had to spend many hours consecutively to learn the framework. All three of us were completely unfamiliar with Java graphical development. As such, understanding the hierarchy of classes, their relationships, and how to effectively use them was overwhelming initially. JavaFX relies heavily on scene graphs, which represent the structure and hierarchy of the UI components in the application. Understanding how scene graphs work, managing layouts, and positioning elements within the scene was quite challenging.

With the utilization of JavaFX for the GUI and Jackson for data storage, meticulous attention must be given to effectively manage dependencies and ensure compatibility between different libraries and frameworks. This requires careful configuration and version management to avoid conflicts and ensure smooth integration of components. This was one of the major obstacles in integrating JavaFX TableView to display data. For example, when trying to display a Bus in the table, one would have to wrap its LocalDate, LocalTime, or Status (custom Enum) attributes inside a JavaFX ObjectProperty. This caused conflicts with reading and writing these objects into JSON files, as writing to those files would cause it to write the ObjectProperty, not the actual value. A lot of time was spent resolving conflicts like these.

Another significant challenge arises from the complex logic involved in handling data modifications. For instance, when editing one bus or data entry, it may be necessary to update related information or modify other interconnected records. Managing such intricate relationships and ensuring data consistency throughout the system requires thoughtful design, meticulous testing, and robust error handling. For example, when modifying a single Bus attribute, one would have to change multiple attributes involved in every booking for every client, or when modifying a Client attribute, one would have to change all of the booking tickets for that client.

4.2 Objectives that we were unable to implement

Originally, our idea for the project was to fully implement a game that utilized JavaFX to display the game board. However, we quickly came to the realization that this was impossible due to the large complexity of the GUI display and the interactions between different classes. When analyzing the project requirements, we quickly realized that meeting them by using a game would be impossible.

Due to various constraints and considerations, the implementation of a user transaction system with real credit cards was not feasible within the Bus Reservation System, although that is the purpose of the Client Manage Balance page[1]. Several factors contributed to this decision. First and foremost, integrating a real credit card transaction system requires complying with stringent security standards and regulations to ensure the protection of sensitive financial information. Meeting these requirements would have necessitated significant resources, including robust security measures, which exceeded the scope and resources available for this project. Secondly, we would need to get access to an actual API that can process monetary transactions, and obtaining the necessary certifications and agreements with financial institutions and payment processors would be impossible.

5 Learning outcomes

The project provided a valuable learning experience for the team, encompassing various aspects of software development. We gained practical knowledge in Object-Oriented Programming (OOP) by implementing multiple classes with different functionalities. The team also developed skills in Graphical User Interface (GUI) development using JavaFX, designing and implementing layouts, handling user input, and displaying data in a GUI. As a team, we had to learn an entire graphics library very quickly in order to be able to proceed with anything regarding the project. As such, we learned the value behind proper time management and dedicating tasks between team members.

Additionally, the team gained experience in error handling and exception handling, ensuring the efficiency and stability of the application. We learned about dependency management and version control, effectively managing dependencies using Maven, learning the Maven build lifecycle process, and using tools like Git for collaboration. Brief, this was a great learning experience that allowed advance our knowledge in the process of software development.

[1] Client manage balance – p.10-11