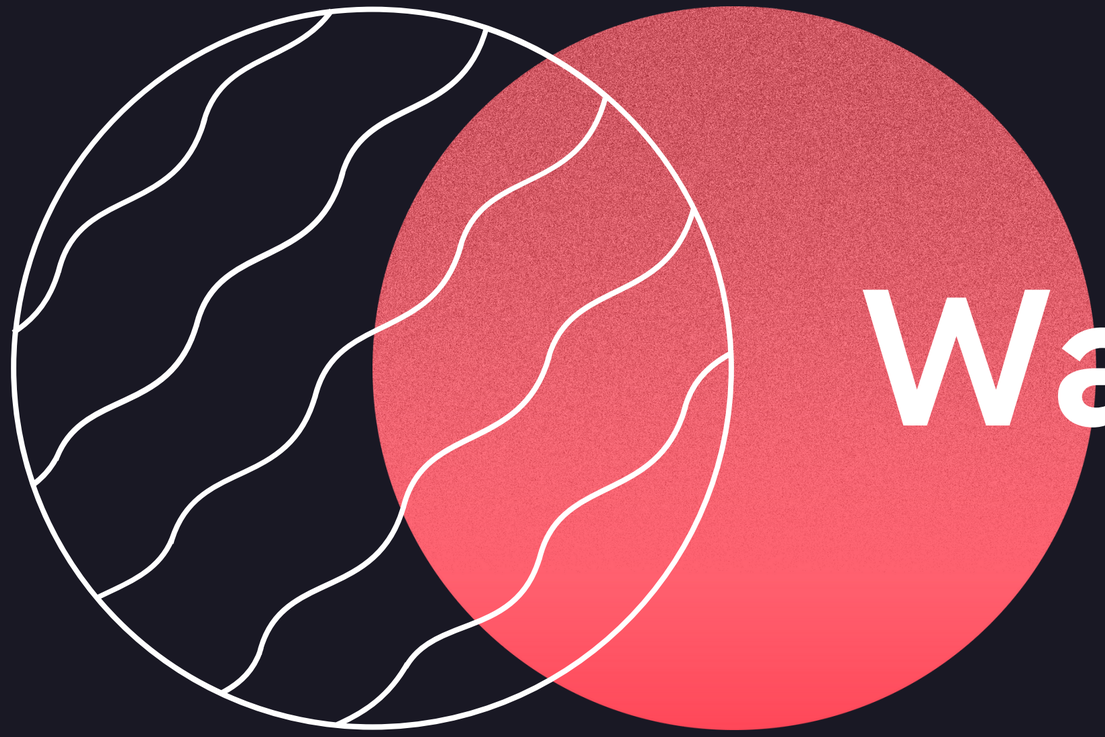
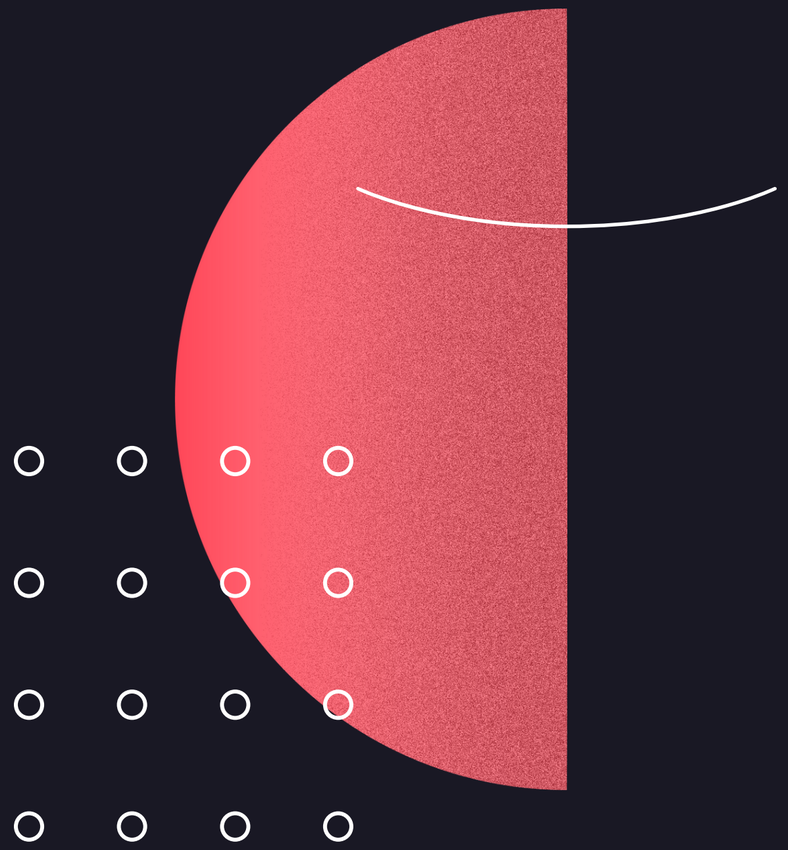
Plan your adventure, stress-free



WanderList

Sheng-Jung Chen 002981985
Yu-Fang Chang 001539627
Yifei Chen 002749660

Project Overview



WanderList is a travel planning and organization application designed to solve the often confusing and stressful problems of travel planning and preparation. The application is designed to provide a comprehensive and user-friendly solution for organizing travel schedules.

The application will allow users to easily add, edit and view their travel itineraries, facilitating users to keep track of activity scheduling and attraction bookings. The result of this project is a fully functional, user-friendly application that simplifies travel planning and organization, forgotten items, and overall anxiety.

CONTRIBUTIONS

Sheng-Jung Chen

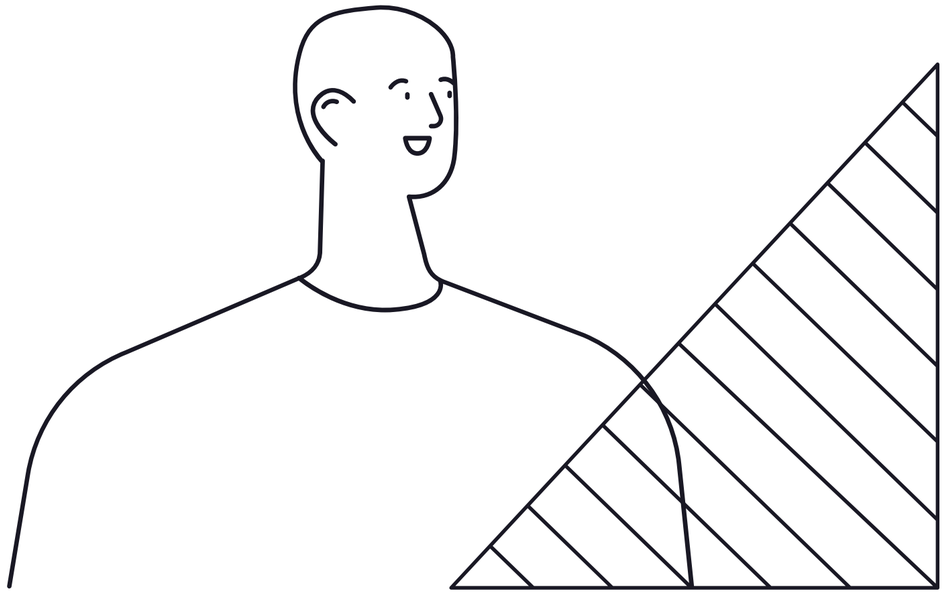
- Database connection
- Dashboard page
- Add and edit trip
- Wishlist item scheduleing

Yu-Fang Chang

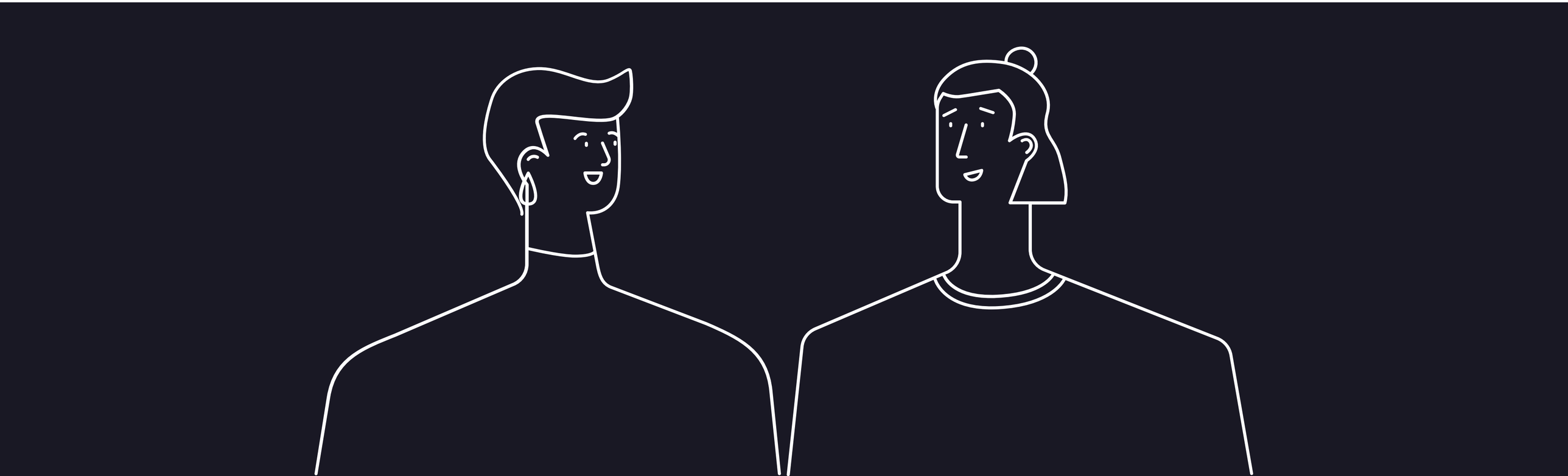
- Add and edit items page
- Wishlist item scheduling
- UI beautify

Yifei Chen

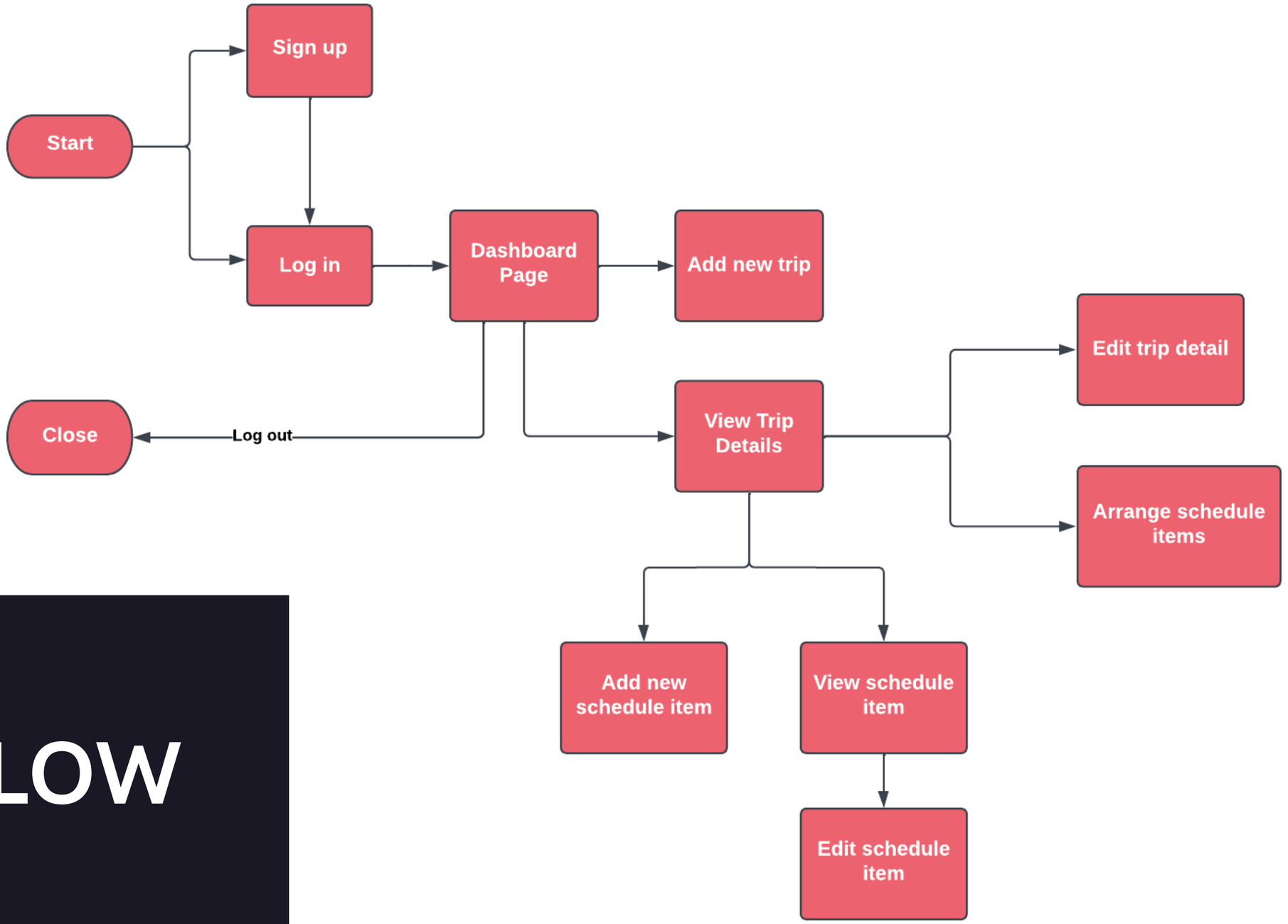
- Login / Signup page and login authentication
- Item types detail display pages
- Presentation slides



DESIGN AND IMPLEMENTATION



WORKFLOW


Sign up
Start
Log in
Dashboard Page
Add new trip
Close
Log out
View Trip Details
Edit trip detail
Arrange schedule items
Add new schedule item
View schedule item
Edit schedule item

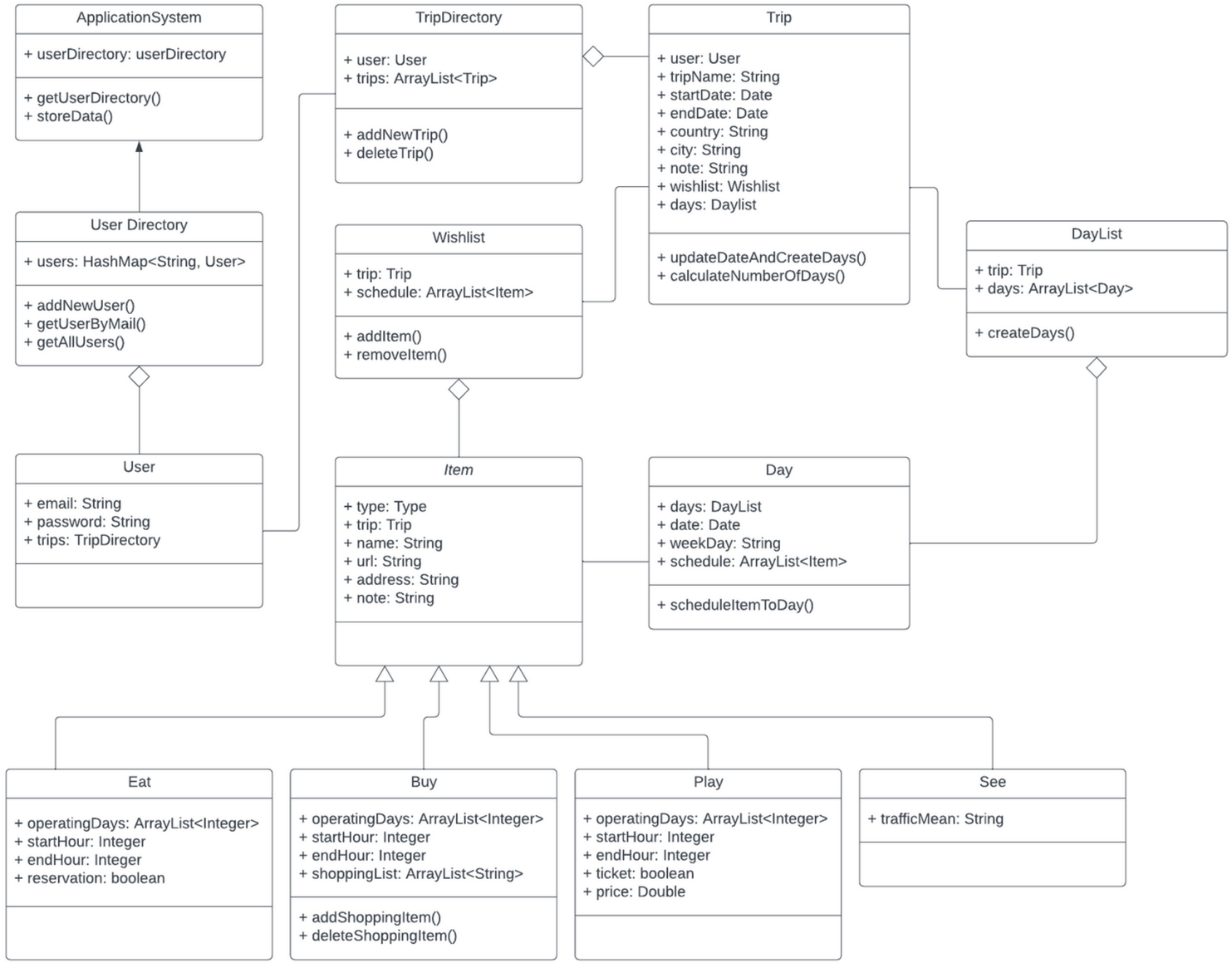
CONCEPTS COVERED IN THE PROJECT

- JavaFX
- Class Definition
- Inheritance/Polymorphism
- Abstract classes/Interfaces
- Lists
- Maps
- Iterator

TOOLS USED

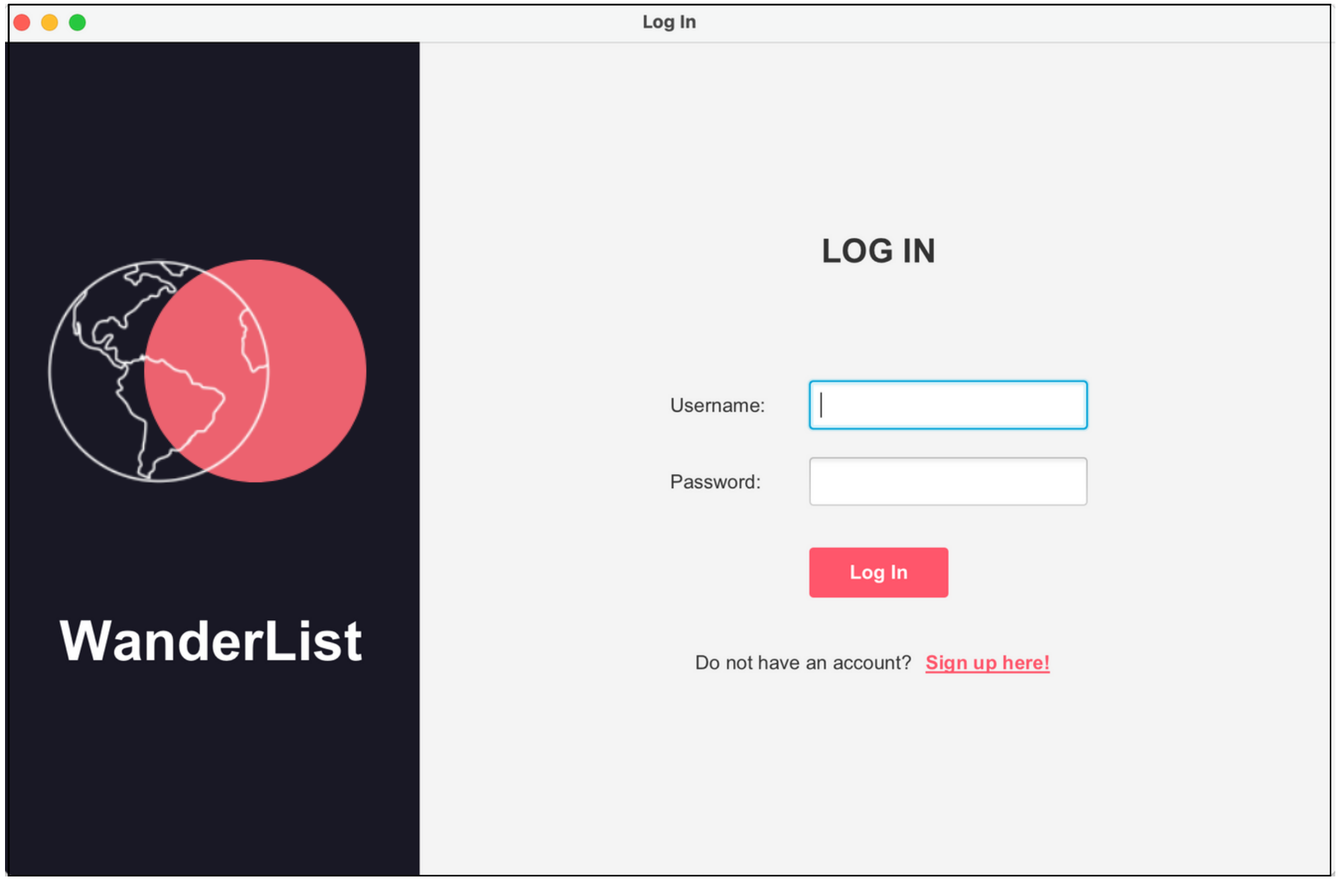
- Eclipse
- Git: version control
- Scene Builder: GUI
- Notion: project management
- LucidChart: UML, workflow

UML DIAGRAM


ApplicationSystem
+ userDirectory: userDirectory
+ getUserDirectory()
+ storeData()
TripDirectory
+ user: User
+ trips: ArrayList<Trip>
+ addNewTrip()
+ deleteTrip()
Trip
+ user: User
+ tripName: String
+ startDate: Date
+ endDate: Date
+ country: String
+ city: String
+ note: String
+ wishlist: Wishlist
+ days: Daylist
+ updateDateAndCreateDays()
+ calculateNumberOfDays()
User Directory
+ users: HashMap<String, User>
+ addNewUser()
+ getUserByMail()
+ getAllUsers()
Wishlist
+ trip: Trip
+ schedule: ArrayList<Item>
+ addItem()
+ removeItem()
DayList
+ trip: Trip
+ days: ArrayList<Day>
+ createDays()
User
+ email: String
+ password: String
+ trips: TripDirectory
Item
+ type: Type
+ trip: Trip
+ name: String
+ url: String
+ address: String
+ note: String
Day
+ days: DayList
+ date: Date
+ weekDay: String
+ schedule: ArrayList<Item>
+ scheduleItemToDay()
Eat
+ operatingDays: ArrayList<Integer>
+ startHour: Integer
+ endHour: Integer
+ reservation: boolean
Buy
+ operatingDays: ArrayList<Integer>
+ startHour: Integer
+ endHour: Integer
+ shoppingList: ArrayList<String>
+ addShoppingItem()
+ deleteShoppingItem()
Play
+ operatingDays: ArrayList<Integer>
+ startHour: Integer
+ endHour: Integer
+ ticket: boolean
+ price: Double
See
+ trafficMean: String

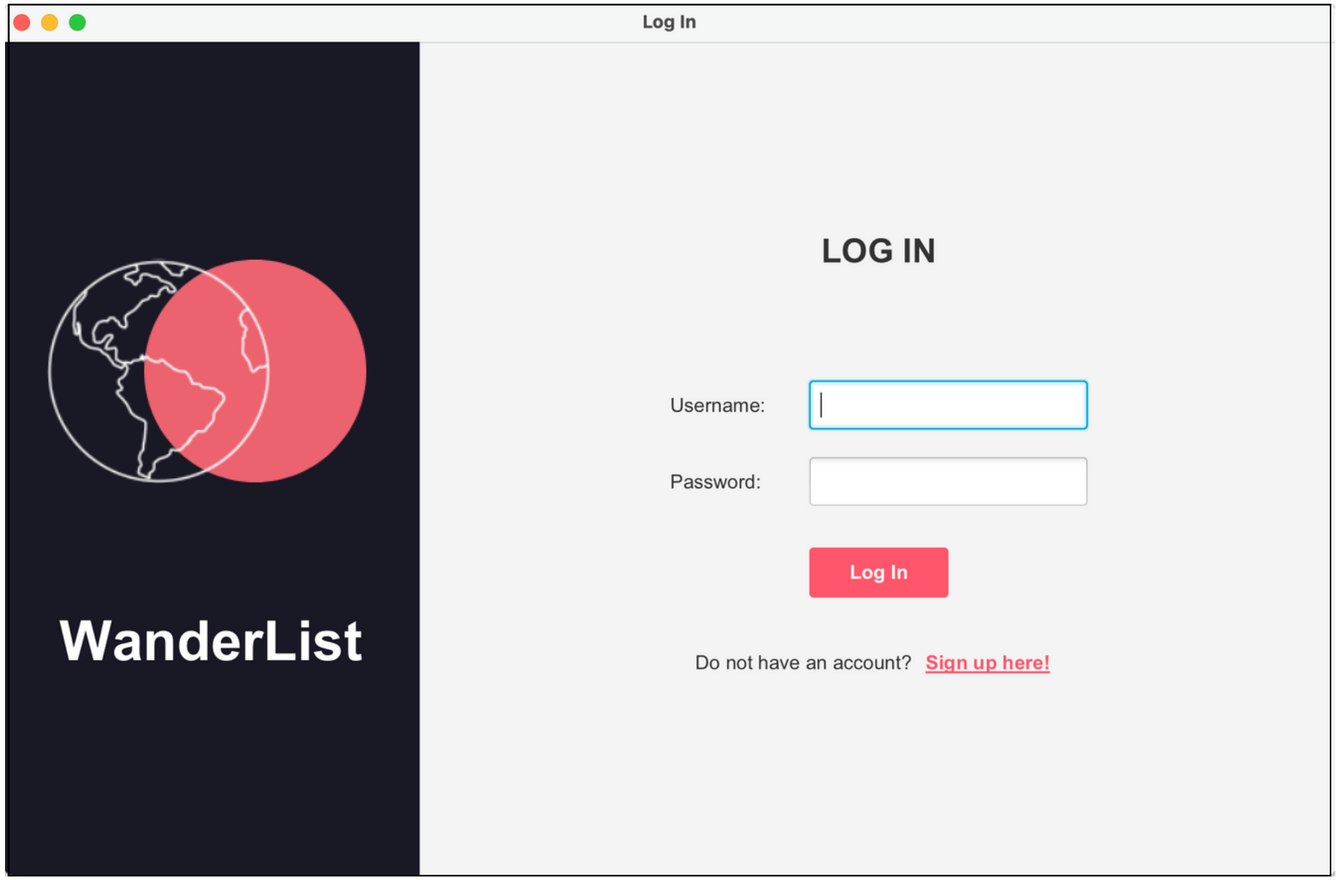
LOG IN

- Developed the login interface using JavaFX and FXML for GUI design
- Managed user actions and application logic with LoginController class
- Leveraged the ApplicationSystem singleton class for user data and authentication
- Validated user credentials during login by comparing the input with stored user data
- Implemented error handling and user feedback with JavaFX Alert dialogs for various scenarios



SIGN UP

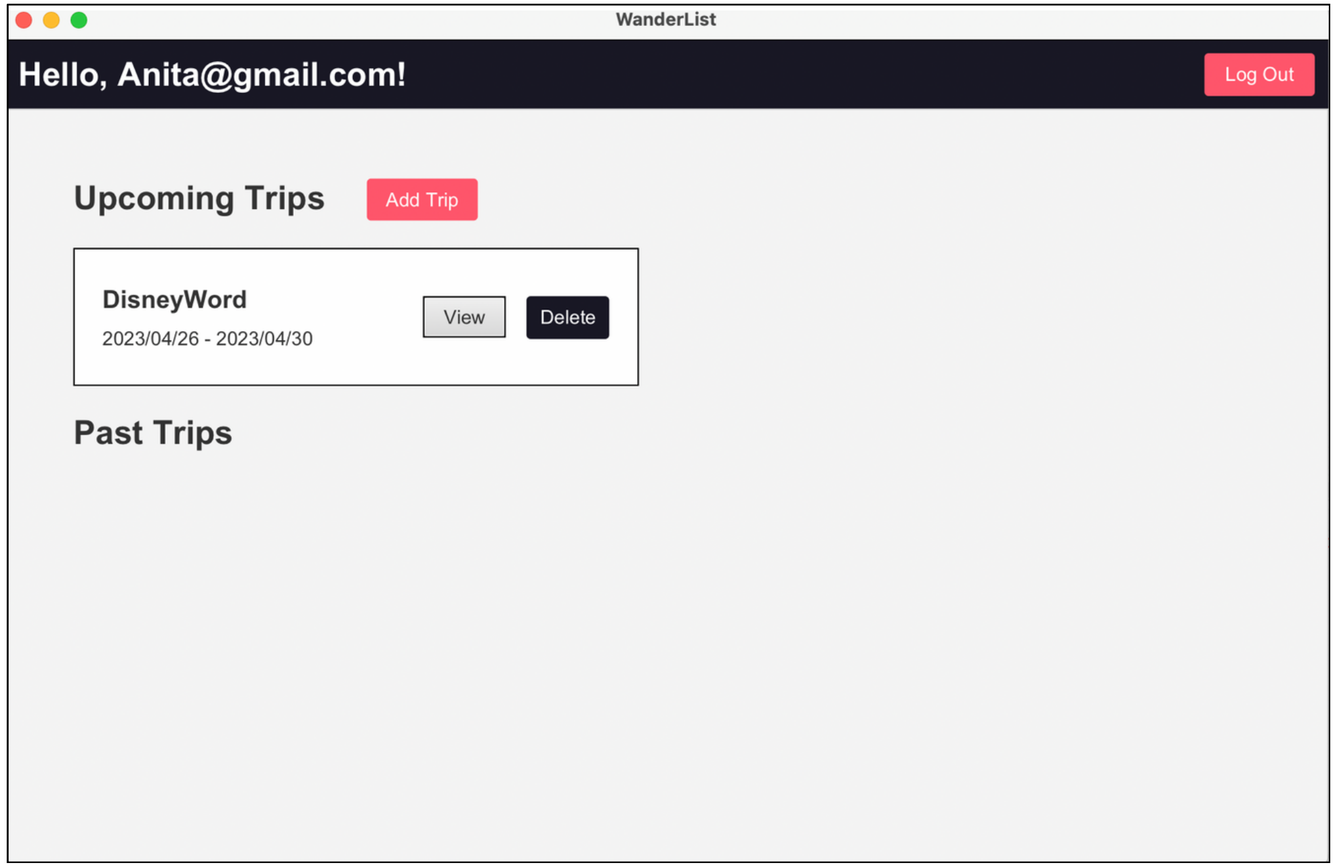
- SignUpController class: Validate user input for registration, and interacts with UserDirectory to create or check for existing users.
- Adding new users: SignUpController calls addNewUser method in UserDirectory to add a new user; if the email already exists, returns null, otherwise adds to HashMap
- UserDirectory class: Manages users in a HashMap (email as key, User as value) and offers methods to add and retrieve
- Provided informative alerts and feedback to the user with JavaFX Alert dialogs



DASHBOARD

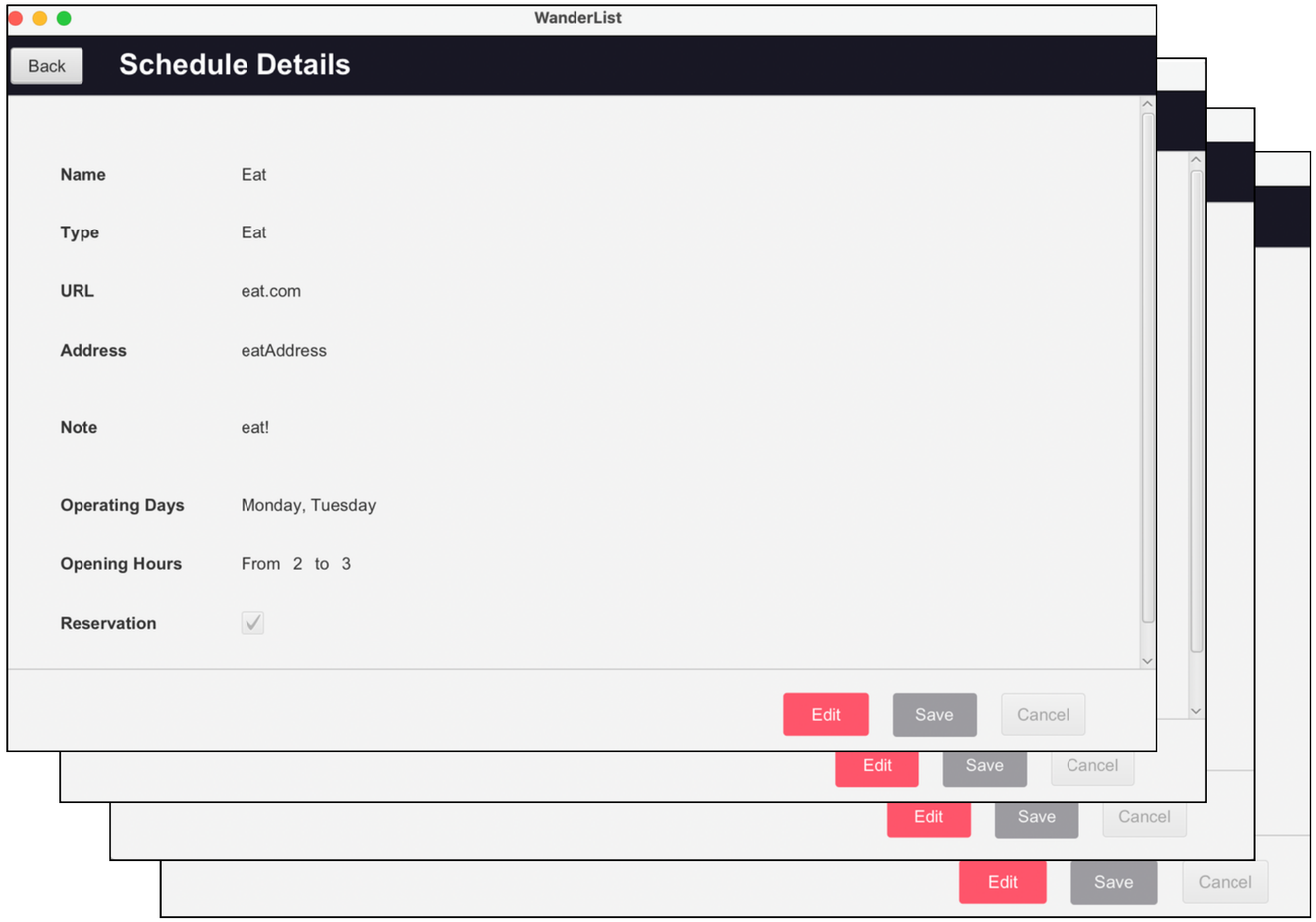
- DashboardController manages the dashboard page, including displaying user information and trips

- In the DashboardController, the place where all trips are loaded is within the loadPage method. This method utilizes an iterator to traverse through the user's trips, splits them into upcoming and past trips with the current date, and displays them in the dashboard.



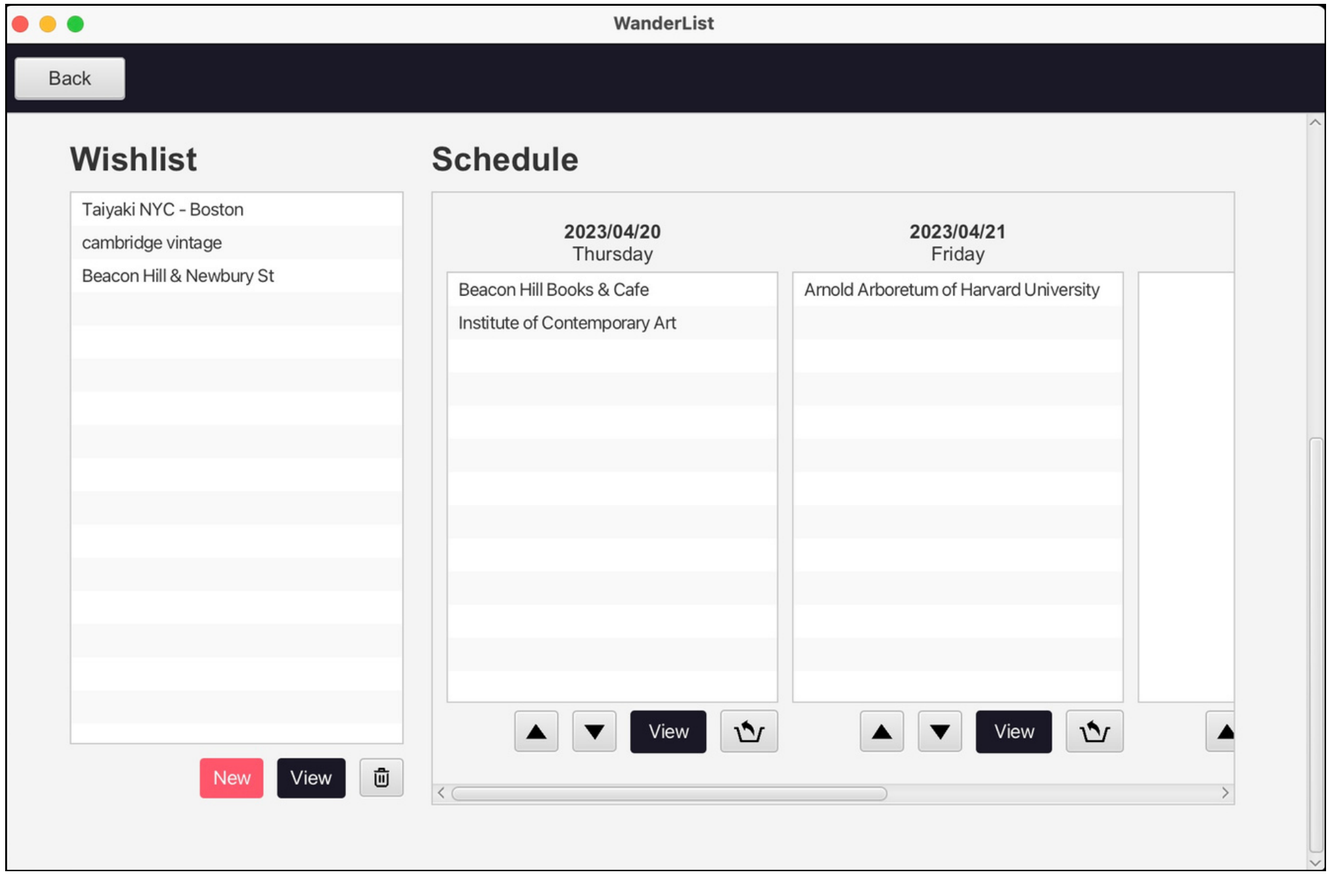
ITEM

- **Abstract class:** Item is an abstract class, providing common properties like type, trip, itemName, and a set of common methods.
- **Inheritance:** Eat, Play, See, and Buy classes extend from the Item abstract class, each class represents a different activity type with specific properties and additional methods.
- **Polymorphism:** Methods defined with Item as parameters can take Eat, Play, See, and Buy as arguments



WISHLIST & SCHEDULES

- Handling drag-and-drop events to schedule items into a day
- Using ArrayList<Day> dayList to store a list of Day objects
- Reordering items in a day with up and down buttons



db4o

```java
import java.nio.file.Paths;

public class DB4OUtils {

    // path to the data storage
    private static final String FILENAME = Paths.get("Database.db4o").toAbsolutePath().toString();
    private static DB4OUtils dB4OUtil;

    public synchronized static DB4OUtils getInstance() {
        if (dB4OUtil == null) {
            dB4OUtil = new DB4OUtils();
        }
        return dB4OUtil;
    }

    protected synchronized static void shutdown(ObjectContainer conn) {
        if (conn != null) {
            conn.close();
        }
    }

    private ObjectContainer createConnection() {
        try {
            EmbeddedConfiguration config = Db4oEmbedded.newConfiguration();
            config.common().add(new TransparentPersistenceSupport());
            // Controls the number of objects in memory
            config.common().activationDepth(Integer.MAX_VALUE);
            // Controls the depth/level of updation of Object
            config.common().updateDepth(Integer.MAX_VALUE);
            // Register your top most Class here
```

```java
            config.common().objectClass(ApplicationSystem.class).cascadeOnUpdate(true);
            // Change to the object you want to save
            ObjectContainer db = Db4oEmbedded.openFile(config, FILENAME);
            return db;
        } catch (Exception ex) {
            System.out.print(ex.getMessage());
        }
        return null;
    }

    public synchronized void storeSystem(ApplicationSystem system) {
        ObjectContainer conn = createConnection();
        conn.store(system);
        conn.commit();
        conn.close();
    }

    public ApplicationSystem retrieveSystem() {

        ObjectContainer conn = createConnection();
        // Change to the object you want to save
        ObjectSet<ApplicationSystem> systems = conn.query(ApplicationSystem.class);
        ApplicationSystem system;
        if (systems.size() == 0) {
            // If there's no System in the record, create a new one
            system = new ApplicationSystem();
        } else {
            system = systems.get(systems.size() - 1);
        }
        conn.close();
        return system;
    }
```

FUTURE IMPROVEMENT

1. Travel-sharing feature
   - Easily share travel itinerary with friends and family
   - Facilitate coordination between traveling companion

2. Google Maps functionality
   - Access real-time location data
   - Get directions
   - Explore nearby points of interest

Thank you!