TicketVault

Design Documents

Group: TicketVault (Team 8)
Members: Chengkang Xu, Hanxun Huang, Kai Sun,
Longding Zhang, Yijie Wu,

Table of Contents

Purpose

Current Organization website, such as: BoilerLink.com does not provide tickets-reserving service for organizations and students. Our goal is to develop a platform (i.e. a website with well-organized UI) that provides a convenient way for students to reserve tickets, and organizations to promote their events.

Summary of list of functional requirements:

1) Sign-up & sign-in.
    a) Student users are allowed to sign up to gain access to personal profile pages and reserve tickets
    b) Organization user sign up is invitation only. It does not open to public.
    c) Both student users and organization users use login field one the top of the webpage to sign in.
2) Profile page
    a) Users will be able to view their personal information
    b) Users are also allowed to edit personal information, such as: contact information, address… etc.
3) Ticket management page
    a) There will be a tickets manage button on the top of homepage, which redirect users to tickets managing pages.
    b) On tickets managing page, users are able to view the list of events that they have created (organization users) or reserved (student users).
    c) Organization users are able to create new events by click on the button “create new events” in tickets managing page.
    d) Each item in the list contains a hyperlink that redirects users to detail info page of the events

e) In detail info page, organization users are able to modify events' information, such as: tickets price and remaining tickets. Student users are able to cancel the reservation.

4) Ticket reserving
   a) Student users are allowed to either add tickets to cart or reserve tickets in events detail page.
   b) Students will have two hours to contact events officials to make the purchase before the reservation is cancelled.
   c) Events' officials will receive emails that contain students' information
   d) During late night (i.e. 1AM), all reserving functions are suspended, unless events organizer enable late-night-reserving function. Reserving functions will be enabled automatically after 8 AM.
5) Category searching
   a) Users are allowed to search events by their categories
   b) The categories are ranged in alphabetic order to help users quickly find specific category.
6) Events bookmarking and updates (If the time allows)
   a) Users are allowed to bookmark events
   b) Users can view all bookmarked events in personal profile pages
   c) If a bookmarked event is about to close, a bubble that contains the closing events information will prompt each time the user is logged in.
7) Payment page (If the time allows)
   a) Replace "reserve" button with "purchase" button in events detail info page
   b) User will be able to make purchase through PayPal
8) Recommendations of popular events (If the time allows)
   a) At bottom of the homepage, there will be a rolling bar that shows popular events
   b) Popular events calculation is based on largest percentage sale during past hour.

GENERAL PRIORITIES

Usability: Our main priority is usability, since some student cannot handle a complex system due the fact that they do not use computer very often. Design style of our system will be very simple and clear. We will use big button and big size font in order to make the buttons very easy to click and viewable by all users. In addition, we may use a gallery to display all events' flyers, which may help user find the events they want to attend quickly.

Performance: Since ticket reserving is all about racing with the time --- first come first serve. Users need to quickly browse all events and find the events that they like; otherwise the tickets of the events may quickly be sold out. We need to ensure the data is real time reflected on the webpage, and make browsing speed as fast as possible. The information on webpage is updated every time user refreshes the page.

Extensibility: Our website may not only provide tickets reserving service, but also it may be extended to provide other service, such as: ticket selling, online chatting, and quick search. With such a short amount of time to build this website, we have limited many functions. Nevertheless, many of the functions are easy to be rebuilt. We could change the ticket reserving function to ticket selling function very easily, as long as we have enough time to build a secured database that stores user's payment information. In addition, we may extend the category search function to quick search function in order to provide user a better user experience.

Design Outline

Our system will use the Client-Server-Database architecture with a central server that communicates between client (i.e. UI) and database. It interacts with users by following steps: client sends user's inputs to server, server forwards requests to database and send the updated data that is retrieved from database back to client. Since we need to make sure that events information (i.e. remaining tickets, date) is real time, we will utilize AJAX throughout our design.

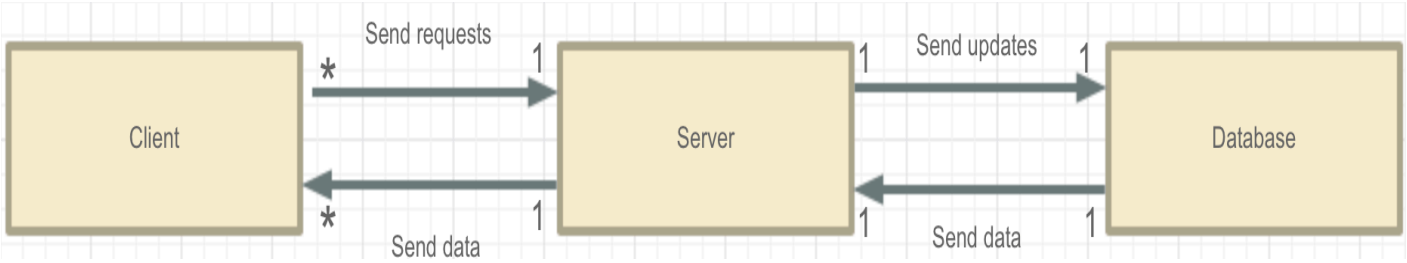

Figure1. UML diagram of high-level structure of our system

1. Client

Purpose: Client is the user interface (UI) that allows users interact with our system. Users may input data, such as: reserve tickets, update personal profile through client or gain access to data that was sent from database, such as: search events, and browse events' information.

Interactions: Client sends AJAX/HTML request to server, and receives AJAX/HTML response from server

2. Server

Purpose: Server is the bridge among clients and database. It provides a function or service to one or many clients, which initiate requests for such services and forward the updates to database. In order to prioritize incoming requests from clients, we will implement a schedule system based on queue in our server.

Interactions: Server receives AJAX/HTML requests from client, and sends AJAX/HTML responses to client with the updated data retrieved from database.

3. Database

Database is where the system stores all data, such as: users' profiles and events information. We will implement a cursor to lookup information that is stored in database. A MySQL database is desired in our design, which should provide enough memory to store user's data.

Interactions: Server retrieves data from database, and sends updates or new data to database.

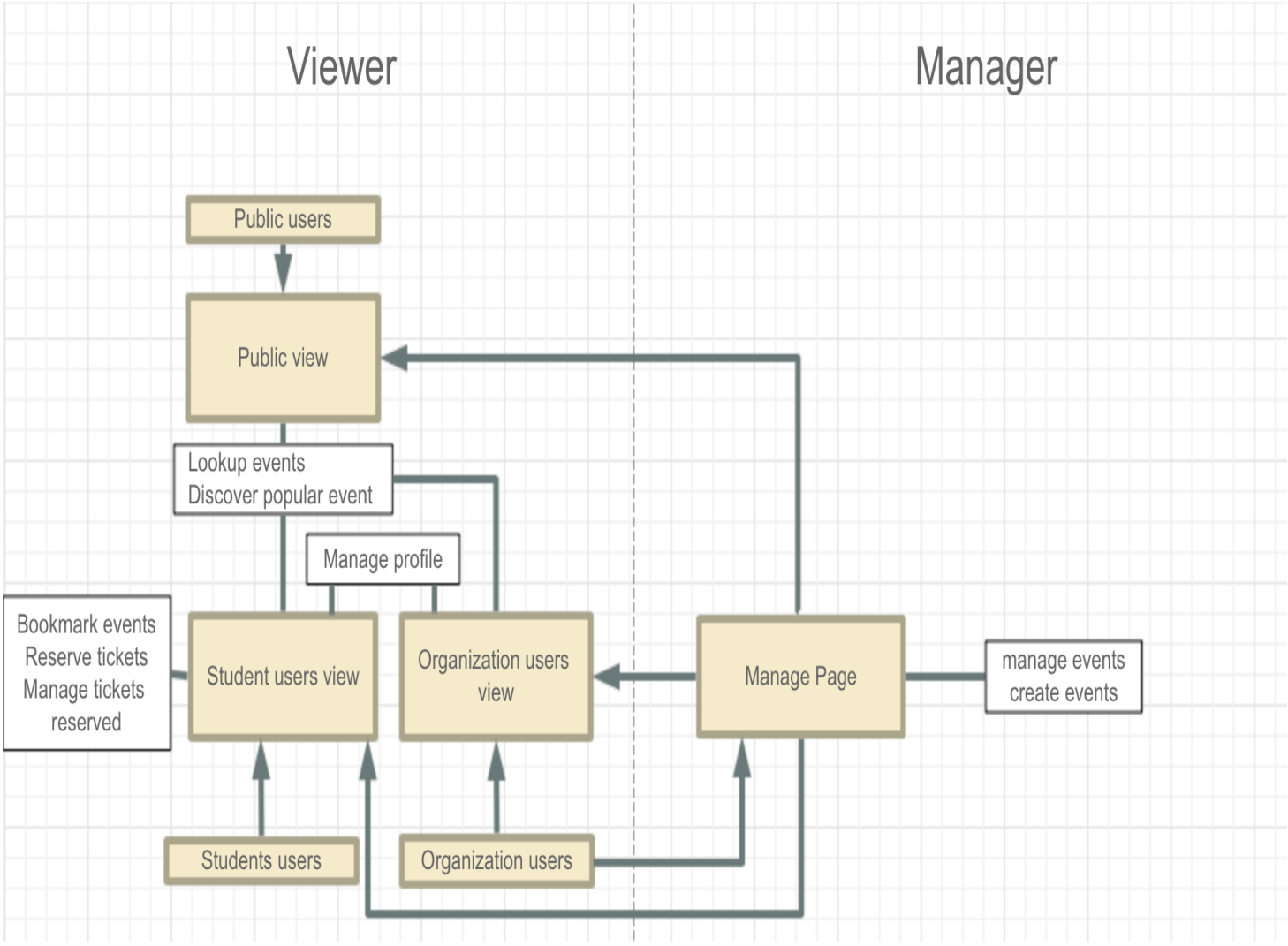

Figure 2. UML diagram of Conceptual modules of our system.

Our system can be divided in two parts: viewer and manager. Viewer part shows how users interact with functions, and manager part shows how organization user performs managing actions.

Users and Permissions:

Our system is designed for three types of users: student users (able to reserve and bookmark tickets), organization users (creating and managing events), and public users (Only able to view and search events). All users are allowed to search and view events; whereas student users and organization users have access to additional functions as listed below.

Student Users:

- Reserving tickets
- Managing reserved tickets
- Bookmark events
- Managing personal profile

Organization Users:

- Creating events
- Managing existed events
- Managing profiles

Design Issues

Issues 1: Login field VS login page

Option 1: Create a login field on the top of all pages.

Option 2: Create a single page for login

Decision: We are more likely to choose Option 1 fits our design better since it is simple and convenient. Redirecting users to a login page is unnecessary and time-consuming. In addition, we could create a base webpage with the login field and other page inherits from the base.

Issues 2: Category search VS quick search

Option 1: Build a category search system that allows user to search event by its category

Option 2: Build a search engine that allows user to search event by keywords. (Like Google, yahoo search)

Decision: We prefer option 1 since it is easier to accomplish. A search engine, like Google, is too complicated to build within a short time. Moreover, category search may help users find the events that they may be interested in, since it will display all events that are listed in a specific category at the same time.

Issue 3: How to inform events organizers when tickets of their have been reserved?

Option 1: Prompting a bubble that says how many tickets are reserved and who have reserved the tickets each time the events organizers login.

Option 2: Sending emails to events' organizers that contains the students' contact information after users reserved tickets.

Decision: We prefer option 2. Since many organizers of events are also students, they may not be able to frequently login to our website and check tickets reservations status. Sending emails have better chances to reach the organizers, and have them to contact the students as soon as possible.

Issue 4: When student user reserved a ticket and is disconnected from our website,

Option 1: cancel the reservation right away.

Option 2: keep the reservation, and place a timer in order details page.

Decision: Option 2 is chosen. Since our webpage does not provide tickets-selling service, users are only able to reserve the tickets and contact events officials to make the purchase in person. Our system will hold the reservation for 2 hours to give events officials enough time to contact the students. There is no point for users to staying on the webpage for that long while they are waiting to be contacted.

Issue 5: How to properly update remaining tickets information?

Option 1: Adding a button that says: "Purchase complete!" in the emails that are sent to the organizers when new reservations are made. Once the button is clicked, events organizers will be redirected to a new page that asks for confirmation of the completion of the purchase. If organizers confirmed the purchase, and a message will be sent to our server to reduce the number of remain tickets by 1.

Option 2: Organizers manually adjust numbers of remaining tickets.

Decision: We are very likely to choose option 2, since some organizers may sell some of the tickets to their friends, the real numbers of remaining tickets are uncertain. Manually counting how many tickets are left and adjusting numbers of remaining tickets on the website is more accurate.

Issue 6: How to notify when an event that user has bookmarked is about to close?

Option 1: Sending an email to inform user about the closing events.

Option 2: Prompt a bubble that tells user the event is closing.

Decision: We choose option 1 for the same reason in Issue 3. Students may check their email more frequently than checking our websites. Email is a better way to inform students regarding the status of the events.

Issue 7: How should we obtain events information?

Option 1: Pulling events information directly from organizations' official websites, and set the remaining tickets of the events to "no tickets to sell". After verifying with the events organizer, set the remaining tickets to proper numbers.

Option 2: Let events organizer manually add events

Decision: We are more likely to choose option 2, since option 1 does not guarantee that the events we pulled from website have tickets for sale. In addition, manually verifying with events organizers is really time consuming.

Issue 8: How should users edit their profile?

Option 1: Adding edit buttons that only enables editing function for single item, next to each item in profile page

Option 2: Adding a big edit button that enables editing functions for all items, at the bottom of the profile page.

Decision: We prefer option 1. Since with option 1, user can only edit the item that they needs to, for example: telephone number. Enable editing function for all items at the same time may slow down the webpage loading speed, since server needs to check all items to see if there are any changes. Single item editing will reduce the processing time.

Issue 9: How to create an organization user account?

Option 1: Sending invitation to verified people who are leader of certain organizations

Option 2: Anyone can sign up as organization users as long as he/she chooses the "register as organization user" and submit a verification of his/her organization in sign up page.

Decision: We choose option 1. If the organization sign up is open to everyone, we may encounter many scams. In addition, the sign up process for organization users may be very long if we choose option 2, since we need to manually verify each application.

Design Details

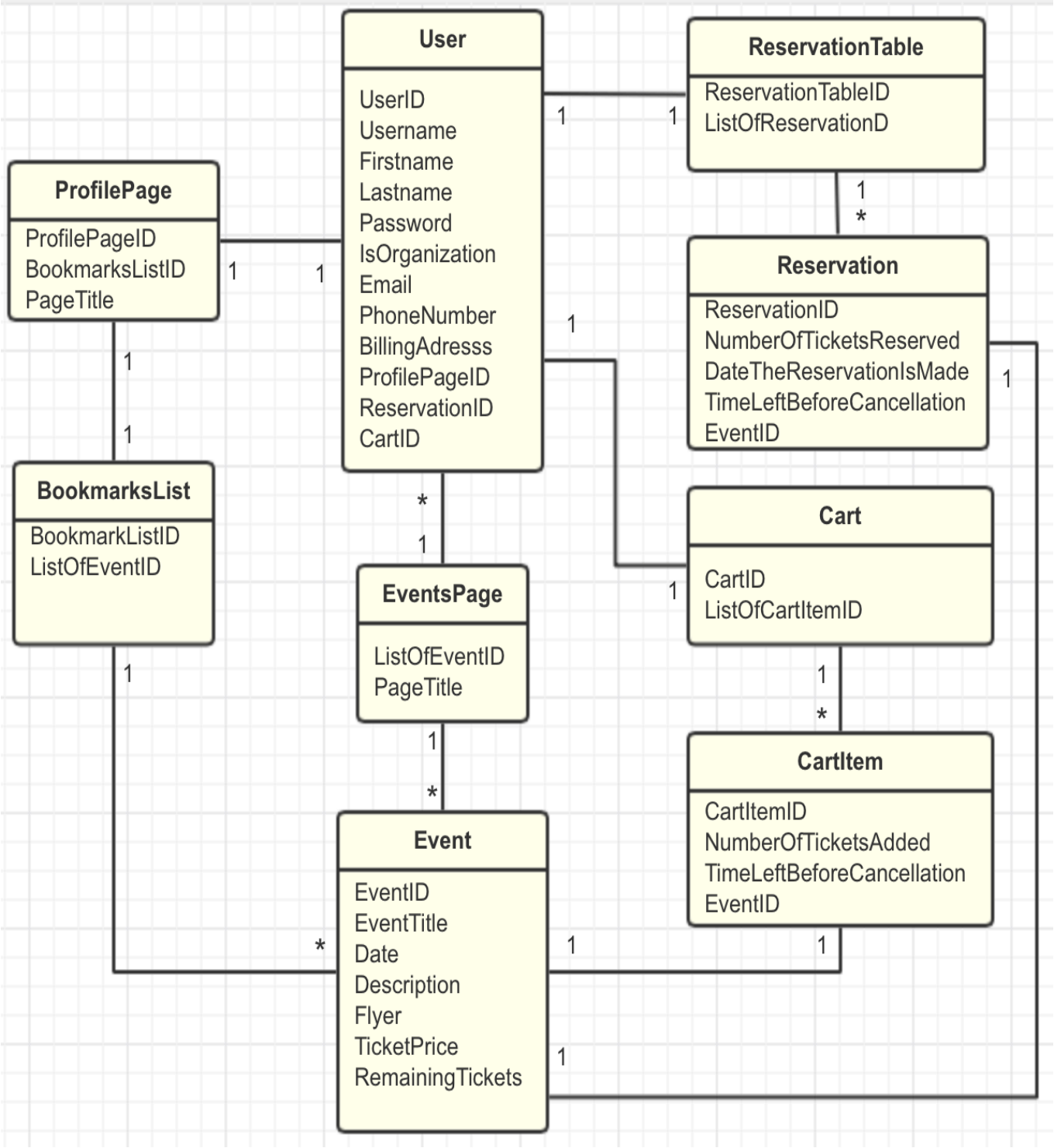

Figure 3. Class diagram of entities-relationships of classes with attributes that will be stored in database.

Description & Interactions:

User:

Attributes: Username，Firstname，Lastname，BillingAddress，Password，Email, PhoneNumber are manually set by users. isOrganization is set by the system and can not be changed by users. All users who sign up through webpage are non-organization users. We will create organization account manually. Server assigns each user with unique userID, ProfilePage ID, ReservationID and CartID to identity user and their data.

Purpose: Class that encapsulates all of the user's information, including both organization users and student users.

Interaction: User's ProfilePage ID, ReservationID and CartID are stored in this class. Whenever user needs to browse their profile, manage reservations and cart, server will utilize these three IDs as symbols to search the data of an individual user in database.

EventsPage:

Attributes: PageTitle is the title of the page. ListOfEvents stores all events.

Purpose: Display all the events to let users find the events they like.

Interaction: EventsPage class and User class have one to many relationships. All users have access to the same events page. The listOfEventsID stores ID of all events. System can retrieve the data stored in Event class by looking up the events ID in database.

Event:

Attributes: Except for EventID, organization users are able to manually set all other attributes. The EventID is assigned by system to identify individual event.

Purpose: Encapsulate data of the event, such as date, description, ticket price… etc.

Interaction: Information that is stored in event class can be retrieved through Reservation, CartItem, BookmarkList and EventsPage classes. All of these classes contain an EventID that will be used to retrieve event's data from database. Only organization users are allowed to edit data stored in event class.

ReservationTable:

Attributes: ListOfReservations stores all reservations that the user has made. ReservationTableID is assigned by system to identify individual reservation table for different users.

Purpose: Store a list of reservations that were made by users

Interaction: When user tries to look up a reservation, the reservationID that is stored in the ListOfReservations will be retrieved and will be used as a symbol to search in database.

Reservation:

Attributes: NumberOfTicketsReserved stores how many tickets the user want to reserve for the event. DateTheReservationIsMade stores the date when reservation was made. TimeBeforeCancellation shows how much time is left before the reservation is cancelled. The ReservationID is assigned by system to identify individual reservation.

Purpose: Store information of a reservation that is made by user.

Interaction: Reservation class contains an eventID that we be used to retrieve event data from database if user sends requests.

Cart:

Attributes: ListOfItemID stores the ID of all items that have been added to cart. The CartID is assigned by system to identify individual cart for different users.

Purpose: Store the list of items that user have been added to cart

Interaction: Cart Class also has a list that stores all CartItems. System will retrieve information of each item by looking up the CartItemID, which is stored in ListOfItemID, in database.

CartItem:

Attributes: NumberOfTicketsAdded stores how many tickets of the same events have been added to cart. TimeBeforeCancellation shows how much time is left before the item is removed from cart. EventID indicates which event this item is. The CartItemID is assigned by system to identify individual item in cart.

Purpose: Contains the information of the event that user has added to cart

Interaction: CartItem contains an EventID, which can be used to retrieve event information from Event Class. Also, since CartItemID is stored in the list in Cart Class. Cart Class also has access to information that is stored in Event Class

BookmarkList:

Attributes: ListOfEventsID stores the ID of all events that have been bookmarked. The BookmarkListID is assigned by system to identify individual BookmarkList for different users.

Purpose: Stores the list of the events that user has bookmarked

Interaction: BookmarkList Class contains a list that stores all EventID that have been bookmarked. System can retrieve information stored in Event class through the BookmarlList Class by looking up the EventID that stores in the ListOfEventID.

ProfilePage:

Attributes: BookmarkListID is the bookmark that is associated wit user's account. PageTitle is the title of the page. ProfilePageID is assigned by system to identify individual event for different users.

Purpose: Stores viewable contents of user's personal information, such as: bookmark list.

Interaction: As users edit their personal profile and bookmark list, the viewable content will also be updated.

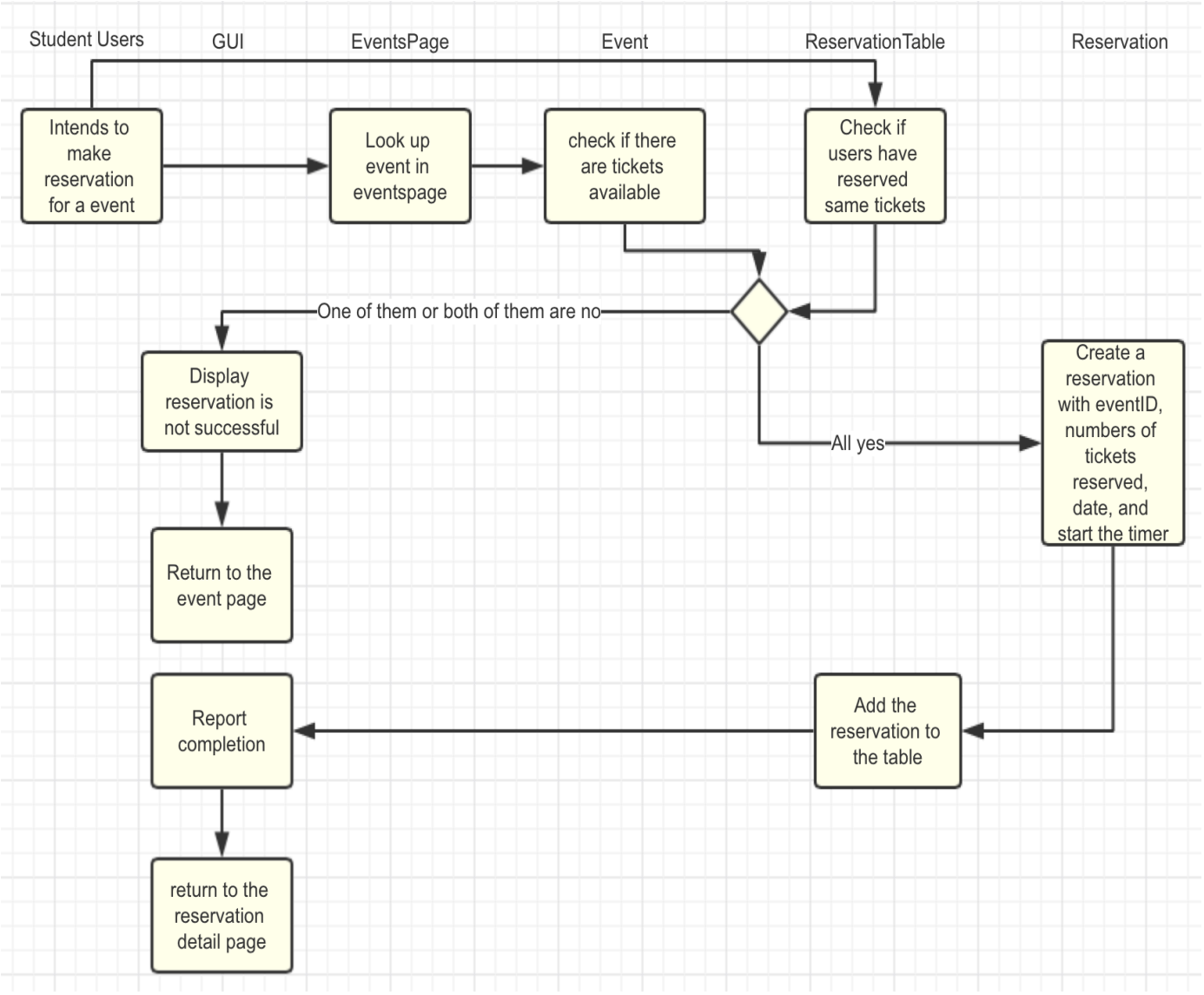

Figure 4. Sequence diagram of make a new reservation

When users try to make a reservation, they first need to look up the event in events page. After selecting the event, system check if the event has tickets available to sell. In the meanwhile, it will check if users have reserved the duplicated tickets before. If any of the check results is “No”, the reserving process will abort. Then it will display a message “Reservation is not successful” and, user will be redirected back to even page. If it passes all checks, an instance of Reservation class will be created. Its eventsID, number of tickets reserved and date will be set to appropriate value. Then the reservation is added to the list in ReservationTable class. User will see a completion message and be redirected to reservation detail page.

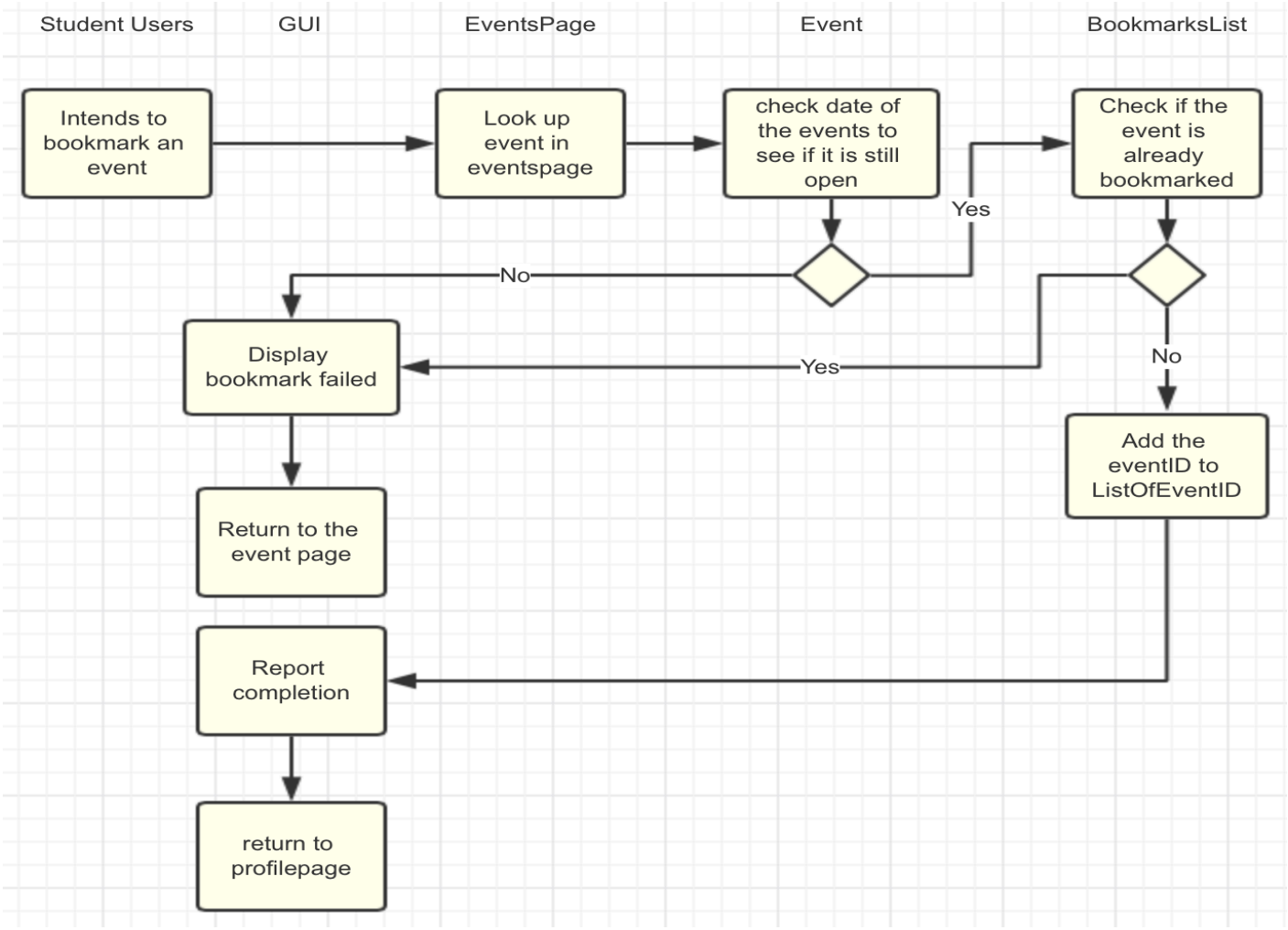

Figure 5. Sequence diagram of bookmarking an event

Searching procedure is the same as it is in making new reservation. After selecting the event, system will compare the date that the event is held with current date closes the event to check whether the event has been closed. If event is open and users click on the bookmark button, system will check if user has already bookmarked the event. If either users have bookmarked the event or the event is closed, the bookmarked process will fail and display “bookmark failed” message, then return to the event page. If it passes all checks, EventID of the event will be added to the list in BookmarkList Class, and users will be redirected back to the profilepage.