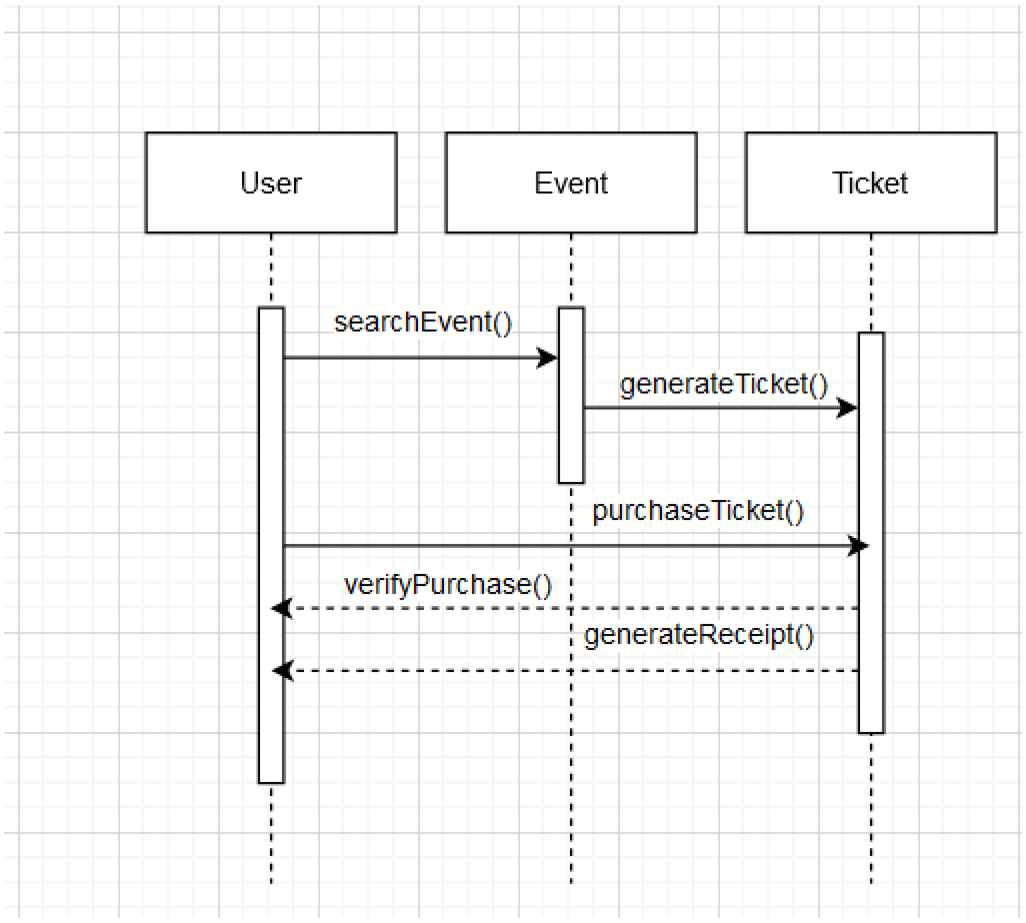
SSD


User
Event
Ticket
searchEvent()
generateTicket()
purchaseTicket()
verifyPurchase()
generateReceipt()

Contract CO1: purchaseTicket

Name	Details
Operation:	purchaseTicket(userID: integer, ticketID: unsigned integer, cardInfo: struct, eventID)
Cross References:	Use Cases: Purchasing a ticket.
Preconditions:	* Buyer must use the application to submit a request to purchase a ticket to the server * Buyer must have filled out the purchase form with credit card information. * Buyer must be logged in.
Postconditions	- eventID.atendees was incremented by 1 - userID.account.tickets was updated with the ticket information

Contract CO2: searchFor

Name	Details
Operation:	searchFor(query: String)
Cross References:	Use Cases: Searching for a ticket.
Preconditions:	* A user must type into the search bar
Postconditions:	- A query through the database was performed - event.information query results were displayed to the user in a presentable format.

Contract CO3: generateReceipt

Name	Details
Operation:	generateReceipt(userID: integer)
Cross References:	Use Cases: Purchasing a ticket.
Preconditions:	* Buyer must be logged in. * Buyer must have purchased a ticket.
Postconditions:	- ticket.information was generated and associated with the buyer.

Contract CO4: sendReceipt

Name	Details
Operation:	sendReceipt(receiptID: integer)
Cross References:	Use Cases: Purchasing a ticket.
Preconditions:	* Buyer must be logged in. * Buyer must have purchased a ticket.
Postconditions:	- ticket.information was sent to the email of the buyer.

Contract CO5: verifyPurchase()

Name	Details
Operation:	verifyPurchase()
Cross References:	Misuse Case: Declined card
Preconditions:	* Buyer must be logged in. * Buyer must attempt to purchase a ticket. * The card was declined when making a purchase.
Postconditions:	- user.verifyPurchase() was called. - ticket.validTicket was updated if needed.

Detailed Class Overview

```
class User:
     int userID;
     String name;
     // String passwordHash;
     // String passwordSalt;
     String password;
     String email;
     Ticket tickets[];

     constructor(string name,string email,string password)
     static validatePassword(string p)
     // static passwordHashing(string password)
     static searchForEvent(String userInput)
     static purchaseTicket(int cardNumber, Ticket ticket)
     Static verifyPurchase();

class Event:
     int eventID;
     int capacity;
     int remainingSeats;
     DateTime startTime;
```

```
    DateTime endTime;
    string eventName;
    string eventDetails

    constructor(
         String eventName,
         int capacity,
         DateTime startTime,
         DateTime endTime
    );

    generateTicket(String eventDetails, String eventName);


class Ticket:
    User attendee;
    Event event;
    DateTime purchaseTime;
    generateReceipt();
    constructor(User user, Event event)
    constructor(User user, Event event, DateTime PurchasedOn)
```