Name Surname:		Name Surname:
Berk Önder	**CTIS 222 Object Oriented Analysis and Design**	Şevki Armağan Oğuz
Student ID:	Fall 2019-2020	Student ID:
21502378	**Homework # 3 (3rd Iteration of the Project)**	21503106
	(15% of the total contribution)	
	Due Date : 15 January 2020, Wednesday 23:55 (Last day of the Final Exams)	

A) Project Title: BSO Event Management System

B) Project Description:

Bilkent Symphony Orchestra is the orchestra of Bilkent University. It was founded it 1993. This orchestra organizes many events per year (piano, violin, movie musics etc.).

BSO Management Event System enables Bilkent students and teachers to buy ticket for events. This software is to be used by 2 types of users, students/teachers (user) and admin. BSO Management Event System has sign up and log in pages.

BSO Management Event System shall enable admins to create events, update events, delete events, manage events. Typical events consist of piano concert, violin concert, movie musics concert.

BSO Management Event System shall enable users to view events and buy tickets for events. Users may have the option to display celebrity information while viewing the event. Bilkent University gives 500 credits to users per year. There is endless chair in the concert hall. Every type of event has different credit. For instance, piano concert is 50 credits, violin concert is 30 credits, movie musics concert is 100 credits.

Each user has account. After they buy a ticket, the credit of that event will decrease from the user account's total credits. Neither users ran out of credit nor have extra credits, all credits will be deleted at the end of the year. If users ran out of credits, users can not able to buy ticket.

BSO Management Event System shall enables admin to sends e-mail to users who have purchased tickets if there is any update (time of event has changed etc.) or cancellation in the event. For instance, there is a piano concert and the event was canceled because our pianist got sick. E-mail is sent to users about this issue.

BSO Management Event System can be reachable from PCs, laptops, tablets, web.

Updated B) Project Description:

IMPORTANT NOTE: We didn't change project description

Bilkent Symphony Orchestra is the orchestra of Bilkent University. It was founded it 1993. This orchestra organizes many events per year (piano, violin, movie musics etc.).

BSO Management Event System enables Bilkent students and teachers to buy ticket for events. This software is to be used by 2 types of users, students/teachers (user) and admin. BSO Management Event System has sign up and log in pages.

BSO Management Event System shall enable admins to create events, update events, delete events, manage events. Typical events consist of piano concert, violin concert, movie musics concert.

BSO Management Event System shall enable users to view events and buy tickets for events. Users may have the option to display celebrity information while viewing the event. Bilkent University gives 500 credits to users and this operation repeats by timer per year. There is endless chair in the concert hall. Every type of event has different credit. For instance, piano concert is 50 credits, violin concert is 30 credits, movie musics concert is 100 credits.

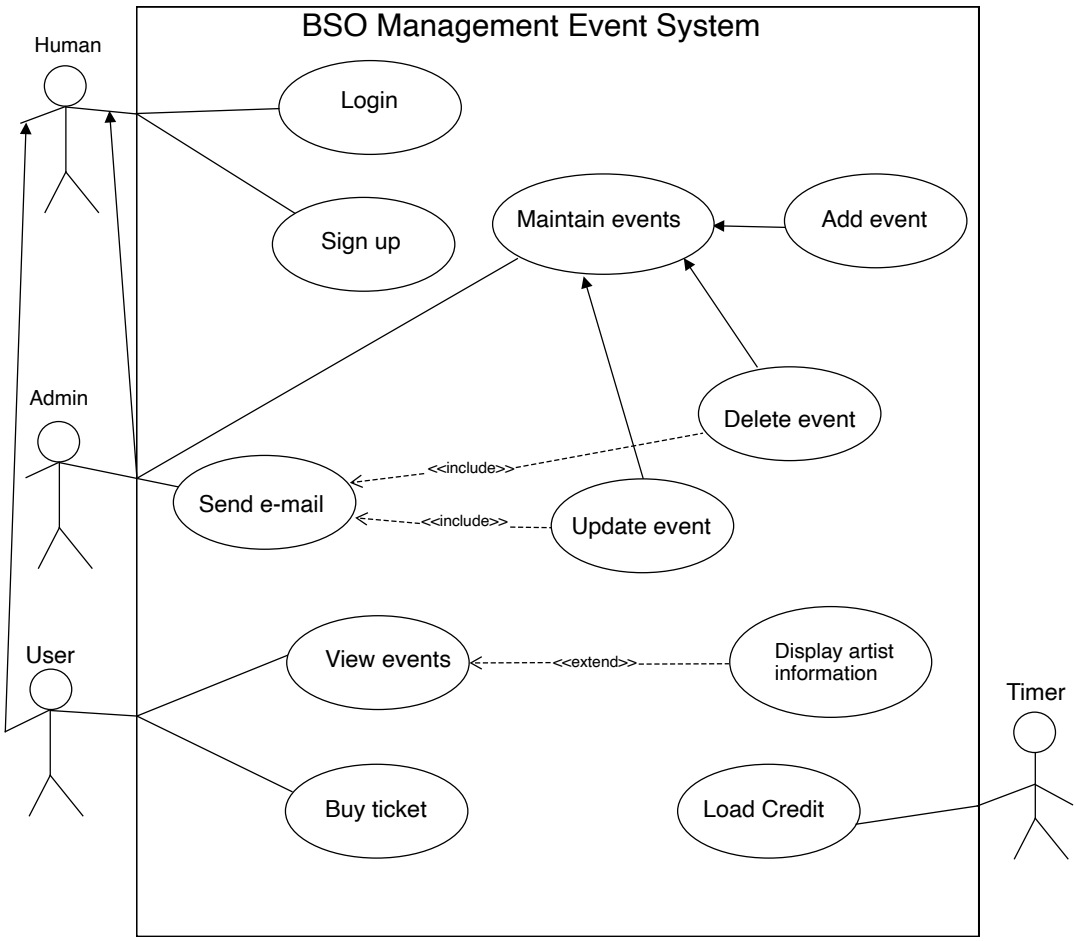
Each user has account. After they buy a ticket, the credit of that event will decrease from the user account's total credits. Neither users ran out of credit nor have extra credits, all credits will be deleted at the end of the year. If users ran out of credits, users can not able to buy ticket.

BSO Management Event System shall enables admin to sends e-mail to users who have purchased tickets if there is any update (time of event has changed etc.) or cancellation in the event. For instance, there is a piano concert and the event was canceled because our pianist got sick. E-mail is sent to users about this issue.

BSO Management Event System can be reachable from PCs, laptops, tablets, web.

C) Identify actors and use cases for your system and show them on a UML Use Case Diagram (System Boundary, use cases, actors, associations, <<include>>, <<extend>>, generalization).

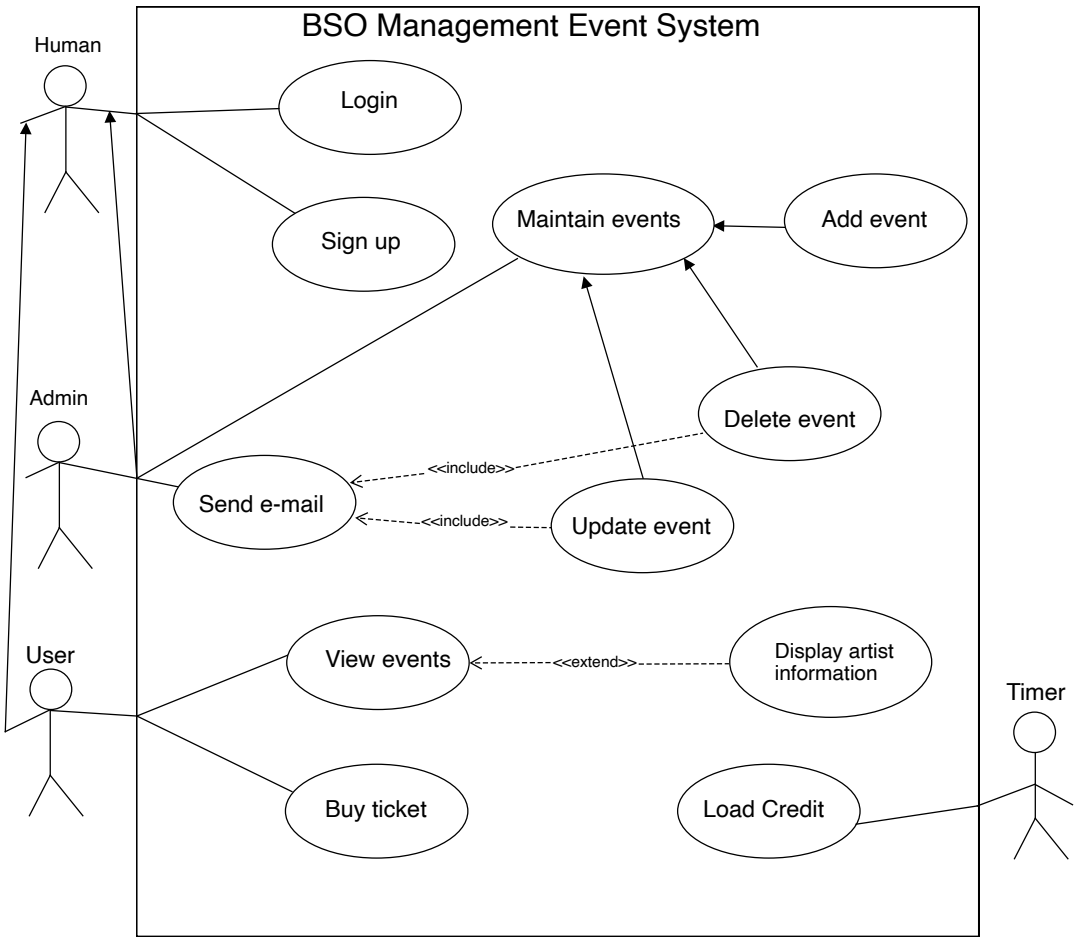
Draw a UML Use Case Diagram of your project.



Updated C) Identify actors and use cases for your system and show them on a UML Use Case Diagram (System Boundary, use cases, actors, associations, <<include>>, <<extend>>, generalization).

Draw a UML Use Case Diagram of your project.

IMPORTANT NOTE: We didn't change our use case diagram.



D) After finalizing your use case diagram, elaborate (describe) <u>ALL use cases</u> of your system by making use of the <u>Use Case Template</u> (UC Description) on Moodle.

E) ID:	UC1
Title:	Login
Description:	Defines how the human login to the system
Primary Actor:	Human
Preconditions:	Human fill the personal information
Post-conditions:	Human view the content of the events
Main Success Scenario: (Main Flow)	1. Human fill the personal information (username,password,email,etc…) 2. Human click the login button 3. Human view the content of the events
Extensions: (Alternative flow)	A1.1. Fields are wrong or invalid. A1.2 Continue with Step 1 in the basic flow
Frequency of Use:	Very Frequent
Priority:	P1

ID:	UC2
Title:	Sign up
Description:	Defines how the human sign up to the system
Primary Actor:	Human
Preconditions:	Human click the sign up to the system button
Post-conditions:	Human view the content of the events
Main Success Scenario: (Main Flow)	1. Human click the sign up button 2. Human fill the personal information (username,password,email,etc…) 3. Human's informations are added to the system 4. Human view the content of the events
Extensions: (Alternative flow)	A1.1. Fields are wrong or invalid. A1.2 Continue with Step 1 in the basic flow
Frequency of Use:	Very Frequent

Priority:	P1

ID:	UC3
Title:	Add event
Description:	Defines how the admin add an event
Primary Actor:	Admin
Preconditions:	UC1
Post-conditions:	A new activity is added that users can view and buy a ticket for.
Main Success Scenario: (Main Flow)	1. Admin login to the system with username and password 2. Admin click the add event button 3. Admin adds event title, event description, ticket price, event type. 4. Event added to the system
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 Continue with Step 1 in the basic flow A3.1 If necessary fields are empty, system ask again to fill the fields. A3.2 continue with Step 3
Frequency of Use:	Very Frequent
Priority:	P1

ID:	UC4
Title:	Delete event
Description:	Defines how the admin delete an event
Primary Actor:	Admin
Preconditions:	UC1
Post-conditions:	Existing activity is deleted from the system, users can no longer view and buy tickets

Main Success Scenario: (Main Flow)	1. Admin login to the system with username and password 2. Admin click the delete event button 3. Existing event deleted from the system
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 Continue with Step 1 in the basic flow
Frequency of Use:	Less Frequent
Priority:	P2

ID:	UC5
Title:	Update event
Description:	Defines how the admin update an event
Primary Actor:	Admin
Preconditions:	UC1
Post-conditions:	Existing activity is updated from the system, users can see new content and time of the event.
Main Success Scenario: (Main Flow)	1. Admin login to the system with username and password 2. Admin click the update event button 3. Admin adds new event title, new event description, new ticket price, new event type. 4. Existing event updated from the system
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 Continue with Step 1 in the basic flow A3.1 If necessary fields are empty, system ask again to fill the fields. A3.2 continue with Step 3
Frequency of Use:	Very Frequent
Priority:	P1

ID:	UC6
Title:	Send e-mail

Description:	Sends e-mail about updated or deleted events to users that bought a ticket for.
Primary Actor:	Admin
Preconditions:	UC1 and UC4(Delete event) or UC5 (Update event) should be executed.
Post-conditions:	Information mail goes to users
Main Success Scenario: (Main Flow)	1. Admin login to the system with username and password 2. Admin execute UC4 or UC5 3. If UC4 executed, the mail about the deletion of the event goes to users who bought a ticket for 4. If UC5 executed, the mail about the new content of the event goes to users who bought a ticket for
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 If user don't have an account, sign up to the system A1.3 Continue with Step 1 in the basic flow
Frequency of Use:	Less Frequent
Priority:	P2

ID:	UC7
Title:	View events
Description:	Defines how the user view the event content
Primary Actor:	User
Preconditions:	UC1
Post-conditions:	User see the event content.
Main Success Scenario: (Main Flow)	1. User login to the system with username and password 2. User click the event. 3. User see the content of the event
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 If user don't have an account, sign up to the system A1.3 Continue with Step 1 in the basic flow
Frequency of Use:	Very Frequent
Priority:	P1

ID:	UC8
Title:	Buy ticket
Description:	Defines how the user can buy a ticket for event
Primary Actor:	User
Preconditions:	UC1
Post-conditions:	User bought a ticket for related event and amount credit of the event decrease from the user's account
Main Success Scenario: (Main Flow)	1. User login to the system with username and password 2. User click the event 3. User see the content of the event 4. User click buy ticket button 5. System calculate the event price and account. 6. Amount credit of the event decrease from the user's account and user can go to event.
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 If user don't have an account, sign up to the system A1.3 Continue with Step 1 in the basic flow A5.1 System check the credit and if credit is not enough, user cannot buy a ticket A5.2 System sends a message to the user whether the credit is enough A5.3 Continue with Step 3 in the basic flow
Frequency of Use:	Very Frequent
Priority:	P1

ID:	UC9
Title:	Display celebrity information
Description:	Defines how the user view artist information
Primary Actor:	User
Preconditions:	UC1
Post-conditions:	User view the artist information
Main Success Scenario:	1. User login to the system with username and password 2. User click the event

(Main Flow)	3. User see the content of the event 4. User click the display artist information button 5. User see artist information of the event
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 If user don't have an account, sign up to the system A1.3 Continue with Step 1 in the basic flow
Frequency of Use:	Less Frequent
Priority:	P2

ID:	UC10
Title:	Load Credit
Description:	Automatically loads 500 credit to users per year
Primary Actor:	Timer
Preconditions:	-
Post-conditions:	Users get 500 credit
Main Success Scenario: (Main Flow)	1. New year begin 2. Load 500 credit to all users
Extensions: (Alternative flow)	
Frequency of Use:	Less Frequent
Priority:	P2

Updated D) After finalizing your use case diagram, elaborate (describe) ALL use cases of your system by making use of the Use Case Template (UC Description) on Moodle.

IMPORTANT NOTE: We didn't change our use case descriptions.

ID:	UC1
Title:	Login
Description:	Defines how the human login to the system
Primary Actor:	Human
Preconditions:	Human fill the personal information
Post-conditions:	Human view the content of the events
Main Success Scenario: (Main Flow)	1. Human fill the personal information (username,password,email,etc…) 2. Human click the login button 3. Human view the content of the events
Extensions: (Alternative flow)	A1.1. Fields are wrong or invalid. A1.2 Continue with Step 1 in the basic flow
Frequency of Use:	Very Frequent
Priority:	P1

ID:	UC2
Title:	Sign up
Description:	Defines how the human sign up to the system
Primary Actor:	Human
Preconditions:	Human click the sign up to the system button
Post-conditions:	Human view the content of the events
Main Success Scenario: (Main Flow)	1. Human click the sign up button 2. Human fill the personal information (username,password,email,etc…) 3. Human's informations are added to the system 4. Human view the content of the events
Extensions:	A1.1. Fields are wrong or invalid.

(Alternative flow)	A1.2 Continue with Step 1 in the basic flow
Frequency of Use:	Very Frequent
Priority:	P1

ID:	UC3
Title:	Add event
Description:	Defines how the admin add an event
Primary Actor:	Admin
Preconditions:	UC1
Post-conditions:	A new activity is added that users can view and buy a ticket for.
Main Success Scenario: (Main Flow)	1. Admin login to the system with username and password 2. Admin click the add event button 3. Admin adds event title, event description, ticket price, event type. 4. Event added to the system
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 Continue with Step 1 in the basic flow A3.1 If necessary fields are empty, system ask again to fill the fields. A3.2 continue with Step 3
Frequency of Use:	Very Frequent
Priority:	P1

ID:	UC4
Title:	Delete event
Description:	Defines how the admin delete an event
Primary Actor:	Admin
Preconditions:	UC1
Post-conditions:	Existing activity is deleted from the system, users can no longer view and buy tickets
Main	1. Admin login to the system with username and password

Success Scenario: (Main Flow)	2. Admin click the delete event button 3. Existing event deleted from the system
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 Continue with Step 1 in the basic flow
Frequency of Use:	Less Frequent
Priority:	P2

ID:	UC5
Title:	Update event
Description:	Defines how the admin update an event
Primary Actor:	Admin
Preconditions:	UC1
Post-conditions:	Existing activity is updated from the system, users can see new content and time of the event.
Main Success Scenario: (Main Flow)	1. Admin login to the system with username and password 2. Admin click the update event button 3. Admin adds new event title, new event description, new ticket price, new event type. 4. Existing event updated from the system
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 Continue with Step 1 in the basic flow A3.1 If necessary fields are empty, system ask again to fill the fields. A3.2 continue with Step 3
Frequency of Use:	Very Frequent
Priority:	P1

ID:	UC6
Title:	Send e-mail
Description:	Sends e-mail about updated or deleted events to users that bought a ticket

	for.
Primary Actor:	Admin
Preconditions:	UC1 and UC4(Delete event) or UC5 (Update event) should be executed.
Post-conditions:	Information mail goes to users
Main Success Scenario: (Main Flow)	1. Admin login to the system with username and password 2. Admin execute UC4 or UC5 3. If UC4 executed, the mail about the deletion of the event goes to users who bought a ticket for 4. If UC5 executed, the mail about the new content of the event goes to users who bought a ticket for
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 If user don't have an account, sign up to the system A1.3 Continue with Step 1 in the basic flow
Frequency of Use:	Less Frequent
Priority:	P2

ID:	UC7
Title:	View events
Description:	Defines how the user view the event content
Primary Actor:	User
Preconditions:	UC1
Post-conditions:	User see the event content.
Main Success Scenario: (Main Flow)	1. User login to the system with username and password 2. User click the event. 3. User see the content of the event
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 If user don't have an account, sign up to the system A1.3 Continue with Step 1 in the basic flow
Frequency of Use:	Very Frequent
Priority:	P1

ID:	UC8
Title:	Buy ticket
Description:	Defines how the user can buy a ticket for event
Primary Actor:	User
Preconditions:	UC1
Post-conditions:	User bought a ticket for related event and amount credit of the event decrease from the user's account
Main Success Scenario: (Main Flow)	1. User login to the system with username and password 2. User click the event 3. User see the content of the event 4. User click buy ticket button 5. System calculate the event price and account. 6. Amount credit of the event decrease from the user's account and user can go to event.
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 If user don't have an account, sign up to the system A1.3 Continue with Step 1 in the basic flow A5.1 System check the credit and if credit is not enough, user cannot buy a ticket A5.2 System sends a message to the user whether the credit is enough A5.3 Continue with Step 3 in the basic flow
Frequency of Use:	Very Frequent
Priority:	P1

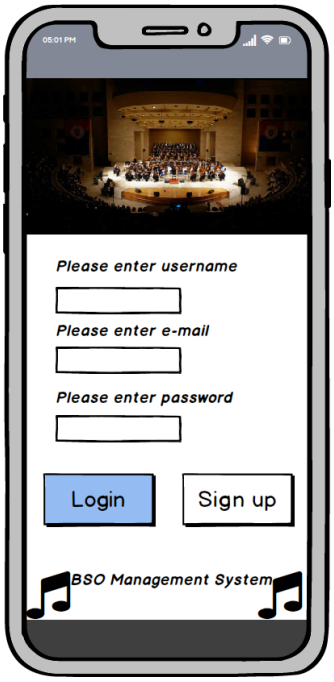
ID:	UC9
Title:	Display celebrity information
Description:	Defines how the user view artist information
Primary Actor:	User
Preconditions:	UC1
Post-conditions:	User view the artist information
Main Success Scenario:	1. User login to the system with username and password 2. User click the event

(Main Flow)	3. User see the content of the event 4. User click the display artist information button 5. User see artist information of the event
Extensions: (Alternative flow)	A1.1 System rejects username or password. A1.2 If user don't have an account, sign up to the system A1.3 Continue with Step 1 in the basic flow
Frequency of Use:	Less Frequent
Priority:	P2

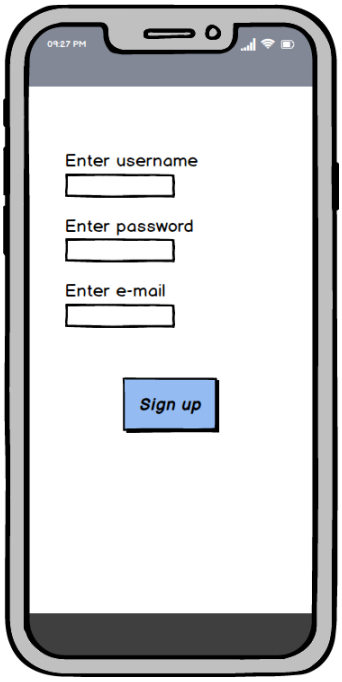
ID:	UC10
Title:	Load Credit
Description:	Automatically loads 500 credit to users per year
Primary Actor:	Timer
Preconditions:	-
Post-conditions:	Users get 500 credit
Main Success Scenario: (Main Flow)	1. New year begin 2. Load 500 credit to all users
Extensions: (Alternative flow)	
Frequency of Use:	Less Frequent
Priority:	P2

F) Draw a Mock-up for each use case. You may have more (or sometimes less) mock-ups than the number of use cases.

1. Login page

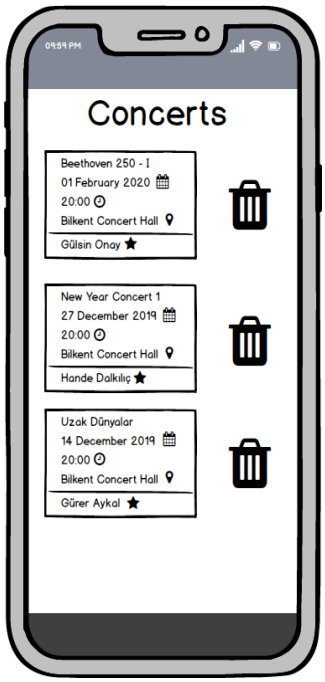


2. Sign up page



3. Add event page

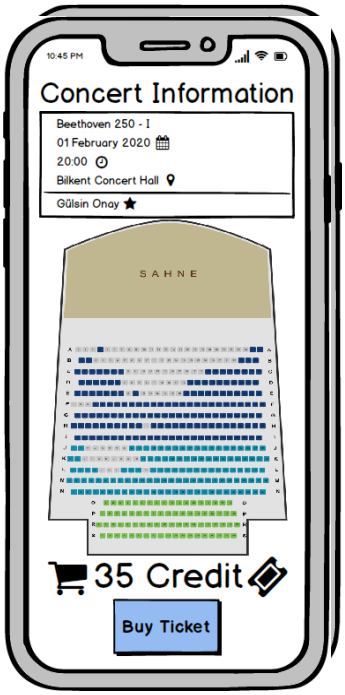
4. Delete event page



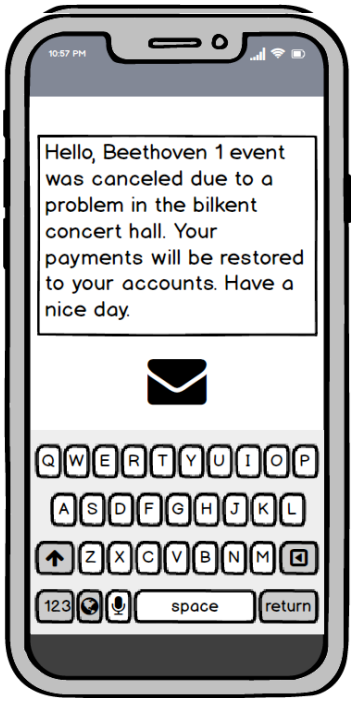
5. Update event page

6. View events

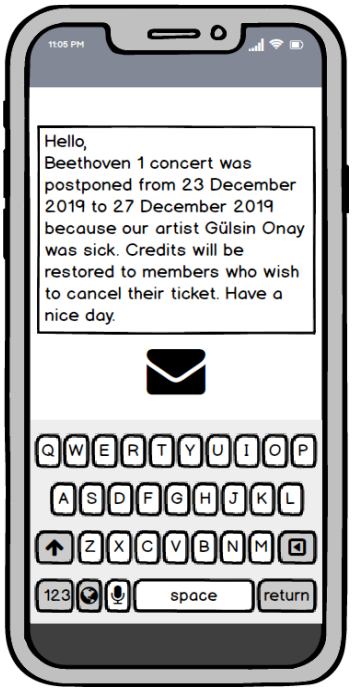
7. Buy ticket



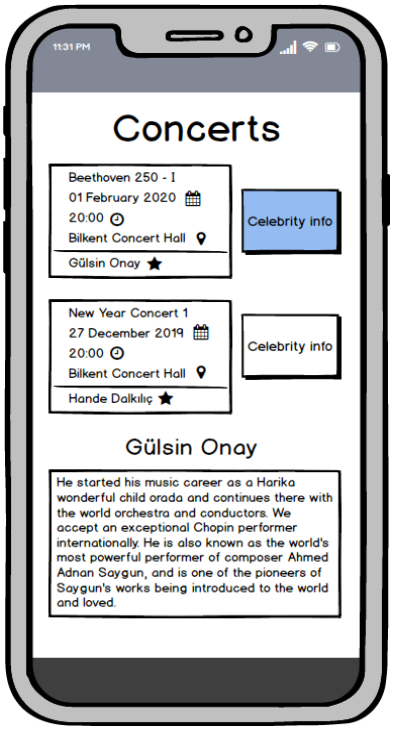
8.1. Send e-mail(for delete)



8.2. Send e-mail(for update)



9. Display celebrity info



G) List of Domain Classes

- Seat
- Human
- User
- Admin
- Payment
- Event
- Artist
- Item
- Timer

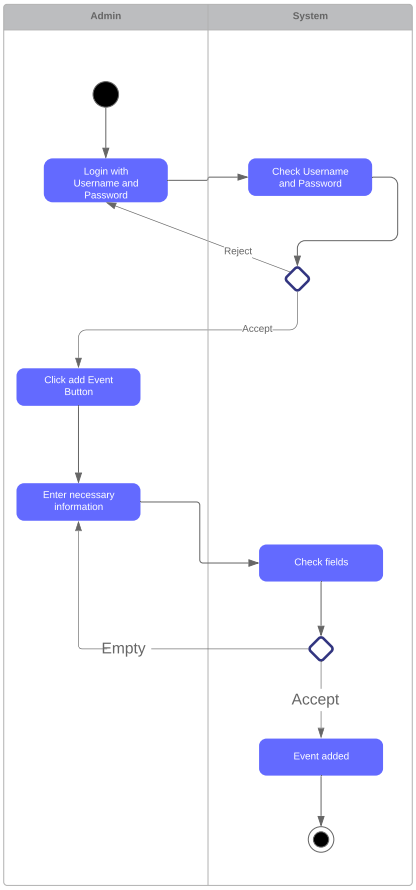
H) UML Activity Diagram

After *finalizing* your use case templates (descriptions), model 3 most important (core) UCs using UML Activity Diagram.

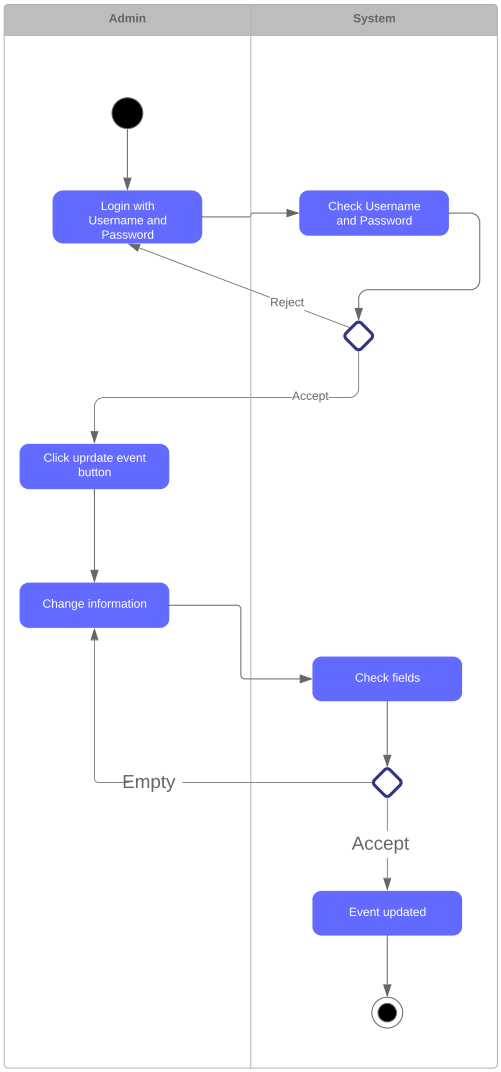
While filling in the UC templates in the Section D, you must have already identified the priority of each UC. Herein, you can use this priority value mentioned in these templates to determine which UC's activity diagram you should draw.

You ALSO need to model alternative (exceptional) flows of the chosen UCs.

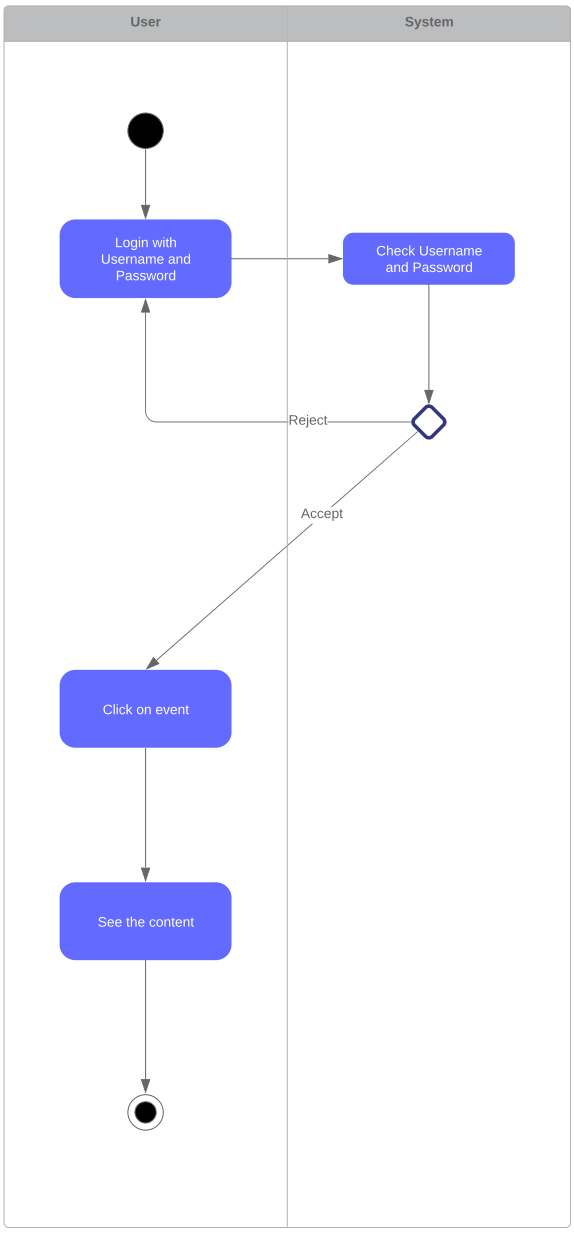
1. AddEvent-UC3



2. UpdateEvent-UC5

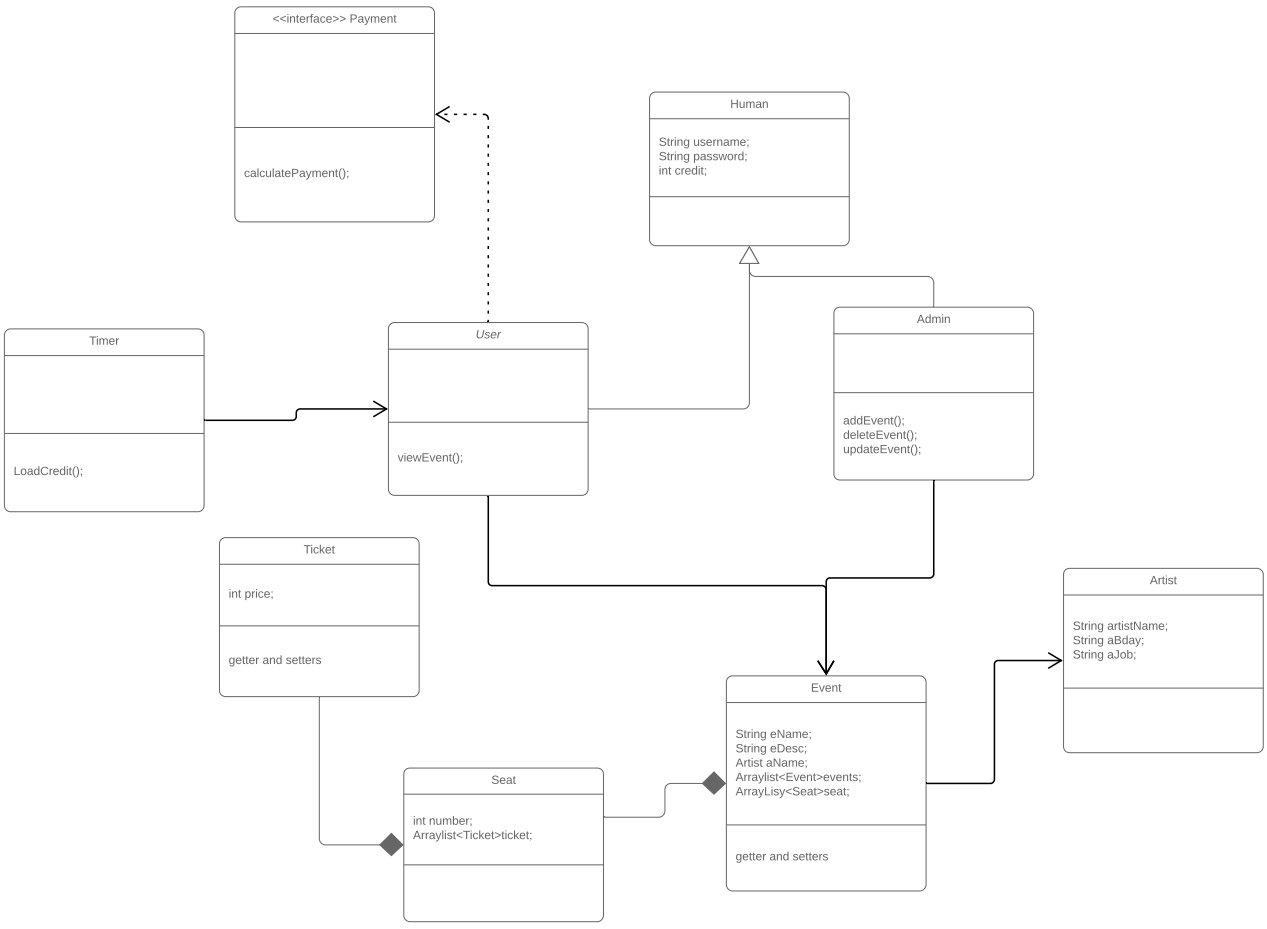


3. ViewEvent-UC7



İ) UML Class Diagram

Draw the UML Class diagram (class names, attributes, multiplicities, navigabilities, role names, method names (optional but it will ease your job during the design)) to model your domain classes in your project. Domain classes should NOT include any system, persistence storage, GUI, and design pattern classes.



Updated H) UML (Design) Class Diagram (New Section)

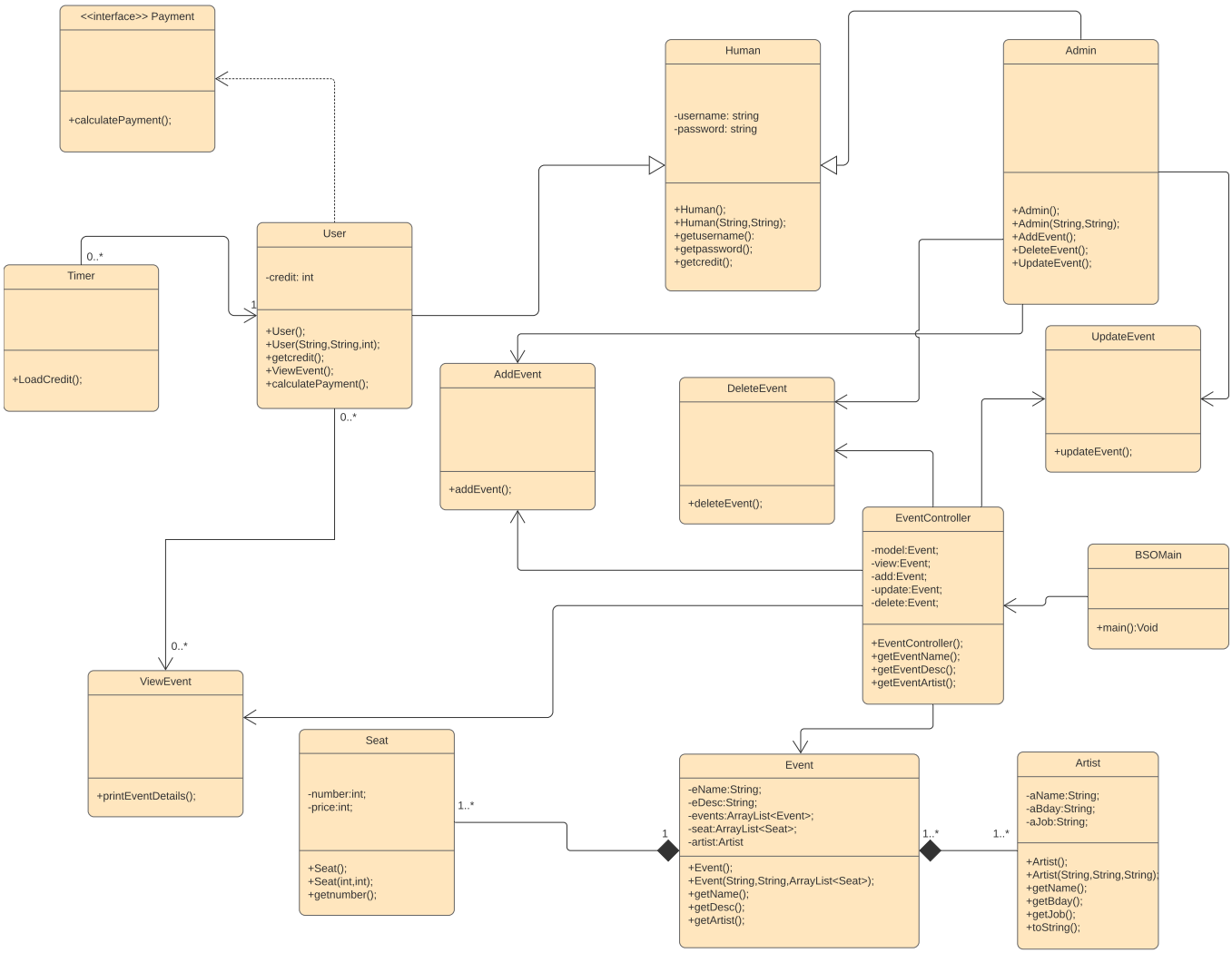
Model the detailed-design of your application by making use of UML Class Diagram.

<delete this part before sending the assignment>

Please also note that "software designs" are like the blueprints of a building that are sufficient for a contractor to build the required building. This means that it should include ALL details of your application.

The design classes you will model in this section will DEFINETELY show similarities with the domain classes you have previously modeled during the analysis phase.

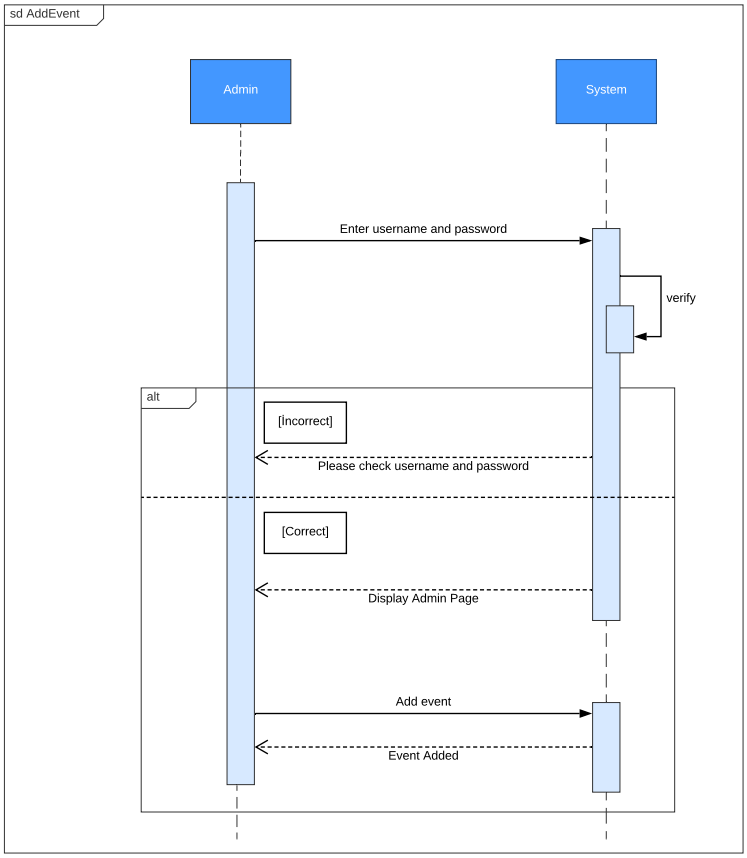
Draw the UML Class diagrams of your design classes.


<<interface>> Payment
+calculatePayment();
Human
-username: string
-password: string
+Human();
+Human(String,String);
+getusername():
+getpassword();
+getcredit();
Admin
+Admin();
+Admin(String,String);
+AddEvent();
+DeleteEvent();
+UpdateEvent();
User
-credit: int
+User();
+User(String,String,int);
+getcredit();
+ViewEvent();
+calculatePayment();
Timer
+LoadCredit();
0..*
1
0..*
AddEvent
+addEvent();
DeleteEvent
+deleteEvent();
UpdateEvent
+updateEvent();
EventController
-model:Event;
-view:Event;
-add:Event;
-update:Event;
-delete:Event;
+EventController();
+getEventName();
+getEventDesc();
+getEventArtist();
BSOMain
+main():Void
0..*
ViewEvent
+printEventDetails();
Seat
-number:int;
-price:int;
+Seat();
+Seat(int,int);
+getnumber();
1..*
1
Event
-eName:String;
-eDesc:String;
-events:ArrayList<Event>;
-seat:ArrayList<Seat>;
-artist:Artist
+Event();
+Event(String,String,ArrayList<Seat>);
+getName();
+getDesc();
+getArtist();
1..*
1..*
Artist
-aName:String;
-aBday:String;
-aJob:String;
+Artist();
+Artist(String,String,String);
+getName();
+getBday();
+getJob();
+toString();

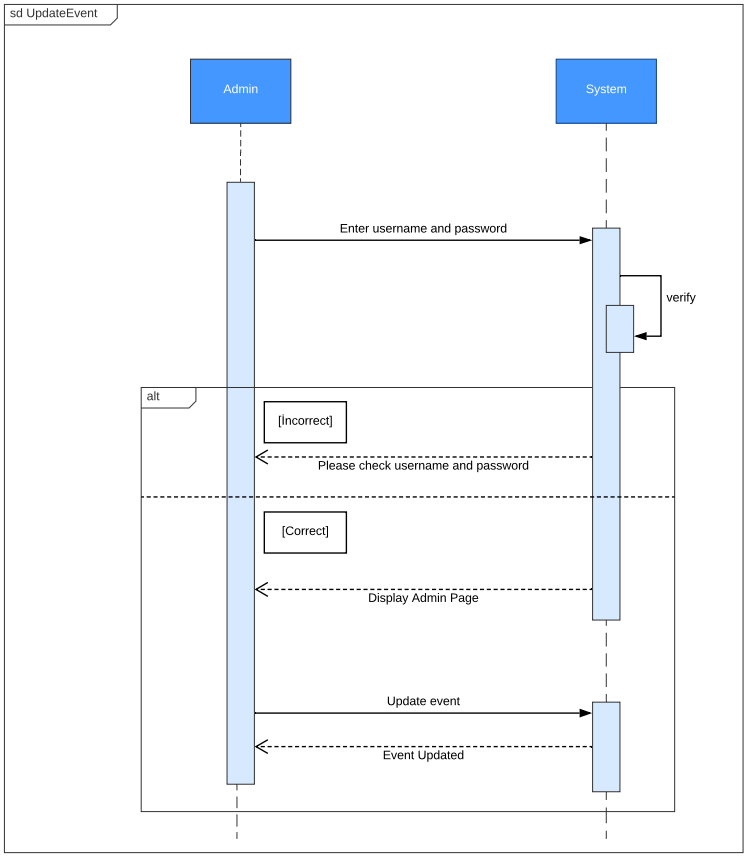
J) UML Sequence Diagram

Model the interaction of 3 most important (core) UCs with UML Sequence diagrams (SD). (You need to have at least 1 SD for each UC.)

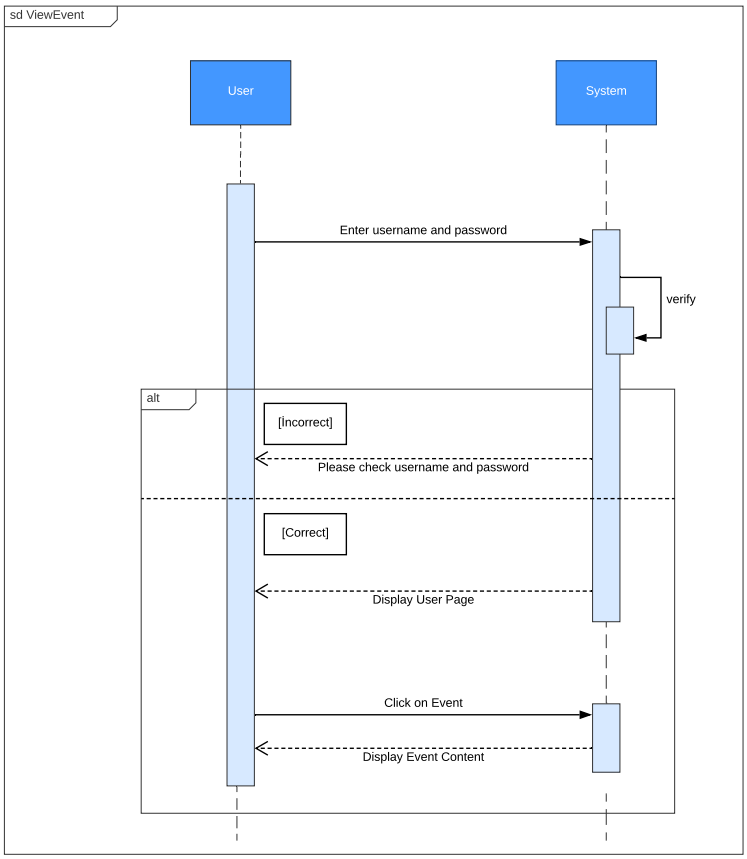
1. AddEvent-UC3



2. UpdateEvent-UC5



3. ViewEvent-UC7

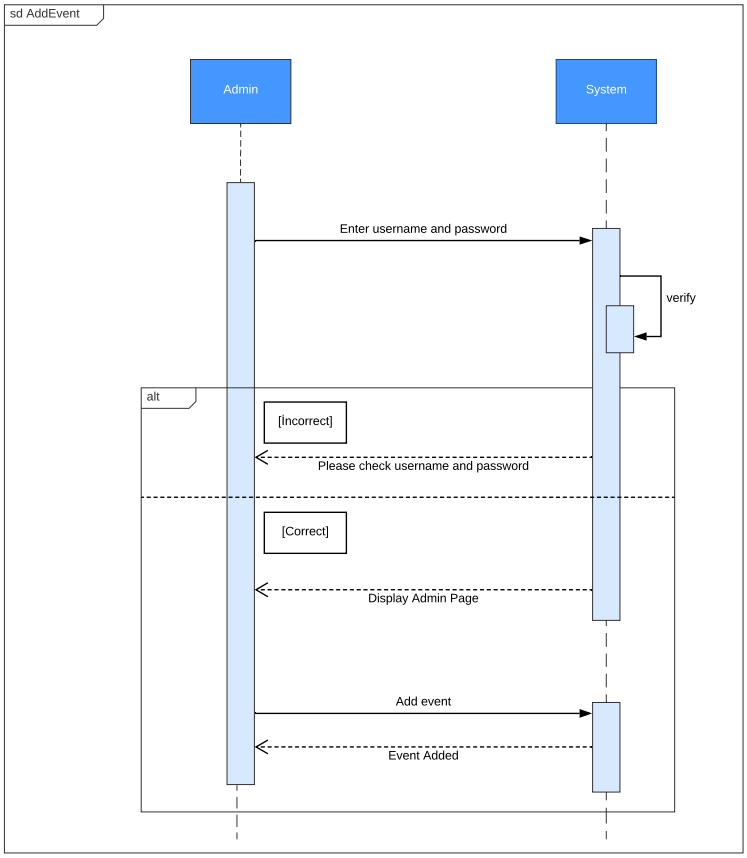


Updated I) UML (Design) Sequence Diagram (New Section)

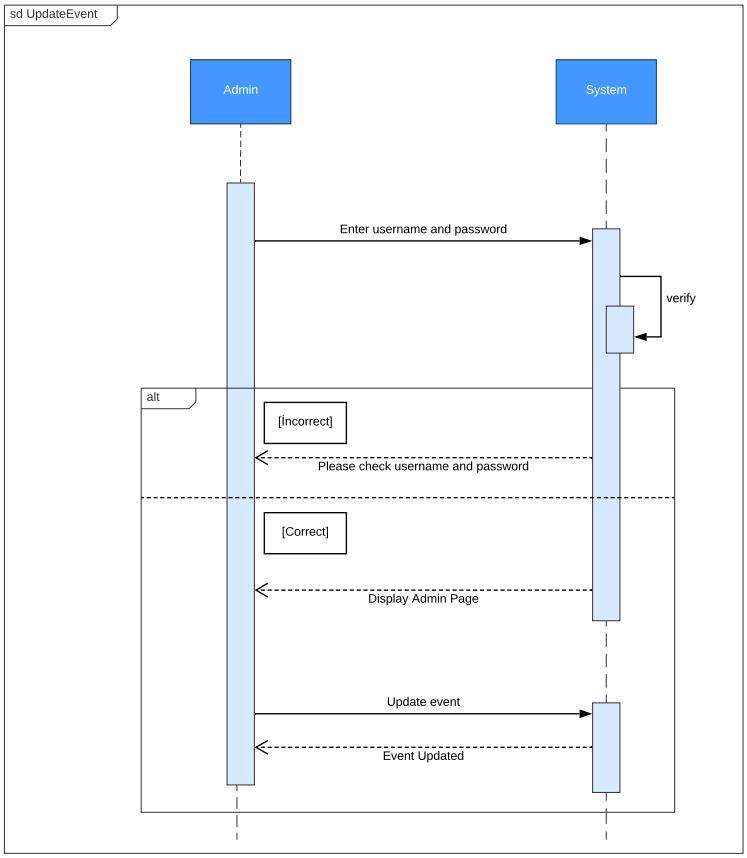
Update the selected UML Sequence diagrams (the same SDs) based on your detailed UML Class Diagram.

IMPORTANT NOTE: We didn't change our sequence diagram

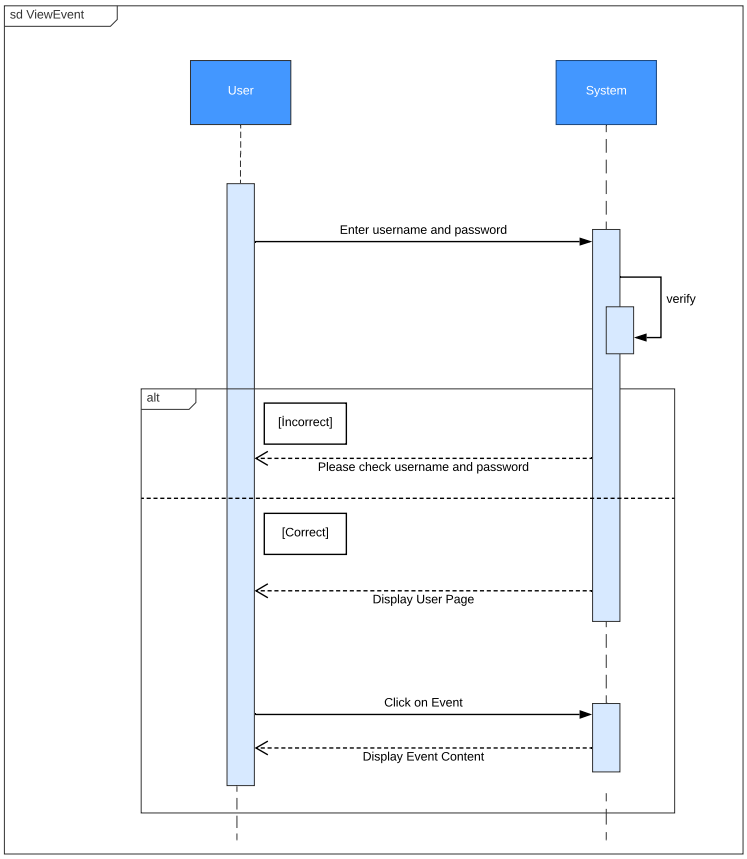
1. **AddEvent-UC3**



2. UpdateEvent-UC5



3. ViewEvent-UC7



J) Write down the names of the OOD principles, guidelines, and patterns (SOLID, GRASP, MVC etc.,) you used while modeling the detailed design of the project. You need to justify your design decision. (New Section)

	Name of OOD principles, guidelines, and patterns	Which class(es)/ method(s) you applied? Refer to the detailed (design) class diagrams. Did you have this class in the domain class diagram?	Why you used this OOD principles, guidelines, and patterns? What you have improved? What you have changed?
1	Single Responsibility Principle	Admin, User	Because, the methods written in these classes can only be used by the classes in which they exist.
2	MVC	Add Event, Update Event, Delete Event, View Event	With this way, we decrease the coupling and if we change one of these classes' inside, only this class will be affected.