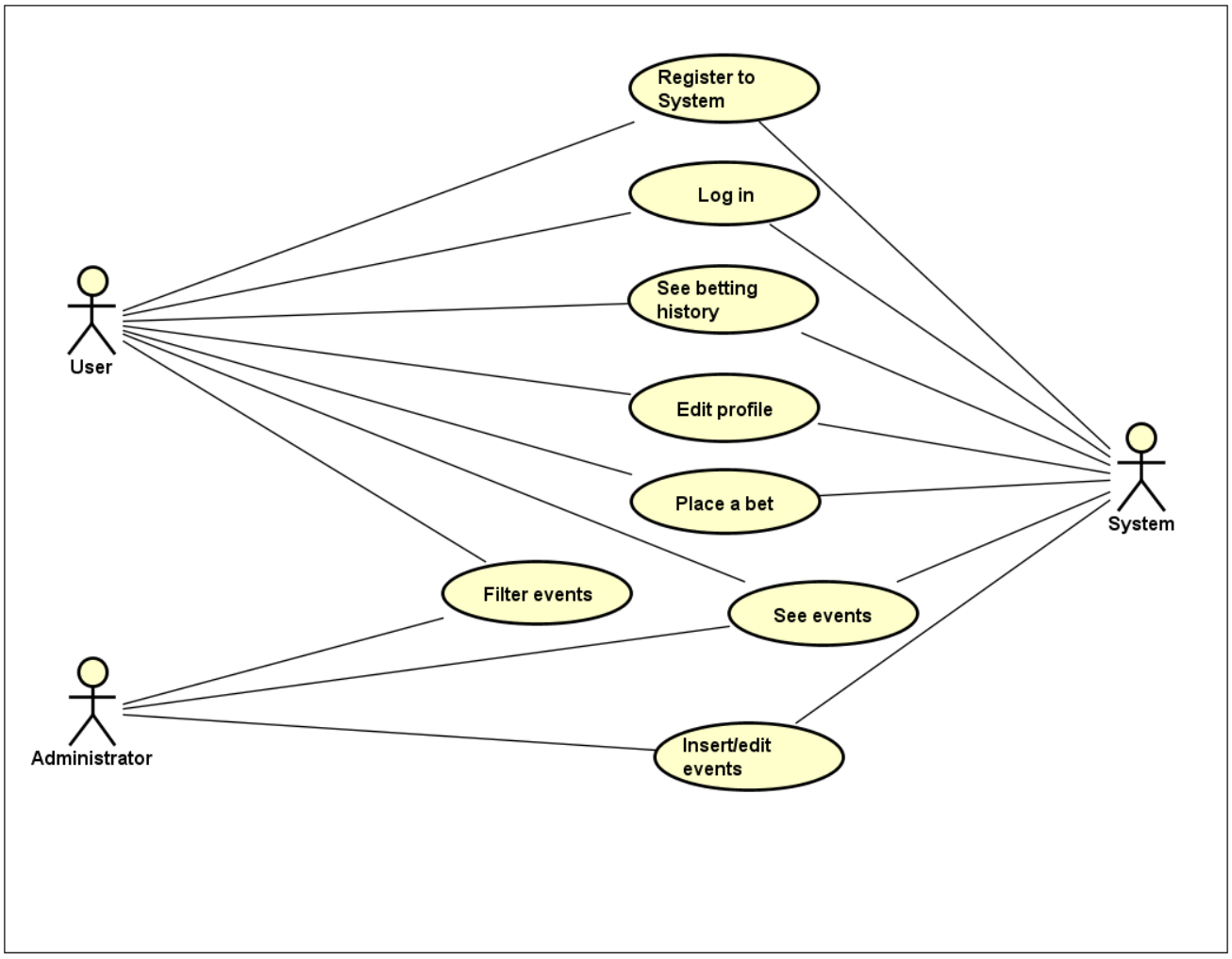
Appendices

Table of Contents

1. Use case

1.1 Use case model



1.2 Use case description

ITEM	VALUE
UseCase	Place a bet
Summary	New ticket is created and stored in the system's database
Actor	User System
Precondition	1. User is logged in 2. User's balance is greater than 0 3. User selected at least one match
Postcondition	1. New ticket is created 2. Money are withdrawn from the user's account 3. Ticket is added to the user's betting history
Base Sequence	1. User chooses one or more matches 2. User places bets on all selected matches 3. User writes the amount of money he/she wants to bet 4. User clicks on the "Place a bet" button 5. System checks if the user has enough money on his/her account 6. System creates a new Ticket 7. Ticket is stored in the system's database
Branch Sequence	
Exception Sequence	Not enough money on the user's account 1-4 as base sequence : 6. Not enough resources on the user's account Warning message appears Empty ticket 4. as base sequence Warning message appears
Sub UseCase	
Note	

ITEM	VALUE
UseCase	Log in
Summary	A user of the System logs in to the System.
Actor	User System
Precondition	1. Server is running 2. Server has access to the database 3. Client is successfully connected to the server 4. User is already registered
Postcondition	Success - the User is authenticated and the system displays the user's interfac e Failure - User typed a wrong username or password
Base Sequence	1. The System prompts the user for a username and password or to register a new account 2. The user enters his/her username and password. 3. System checks login information 4. User is logged in 5. The system displays the user's interface
Branch Sequence	
Exception Sequence	There is no user with the given username 1-2 as base sequence : 3. System declines user's request Warning message appears 2. Wrong password 1-2 as base sequence : 3. System declines the user's request Warning message appears
Sub UseCase	
Note	

ITEM	VALUE
UseCase	See events
Summary	User can look through all matches
Actor	User Administrator System
Precondition	1. User is connected to the server 2. User is logged in
Postcondition	1. Table displays a list of today's matches
Base Sequence	1. User logs in 2. Table with detailed information about today's matches appears
Branch Sequence	
Exception Sequence	There are no matches for today 1 as base sequence : 2. No matches for today in the database Empty event's table
Sub UseCase	
Note	

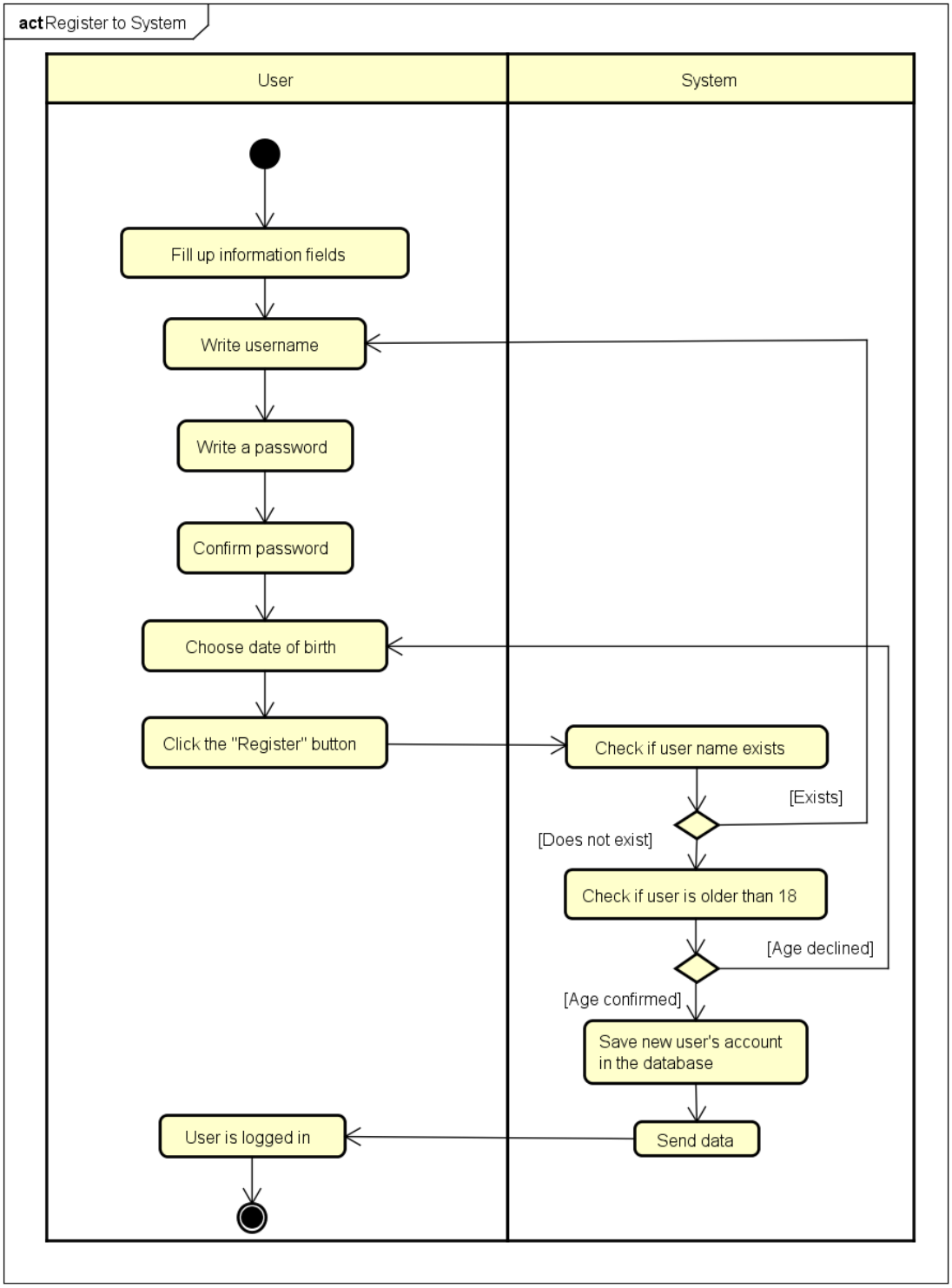
ITEM	VALUE
UseCase	Filter events
Summary	Filter events based on type/time period
Actor	User Administrator
Precondition	1. User is connected to the server 2. User is logged in
Postcondition	1. Users interface displays table of matches based on the filters conditions
Base Sequence	1. User chooses the type of the event 2. User chooses the start date from the calendar 4. User chooses the end date from the calendar 5. User clicks on the "Filter" button 6. Table of events is filled with required events
Branch Sequence	
Exception Sequence	There are no matches in the selected time period 1-5 as base sequence : 6. Event table is empty User case ends 2. End date is before the start date 1-5 as base sequence: 6. Event table is empty User case ends
Sub UseCase	
Note	

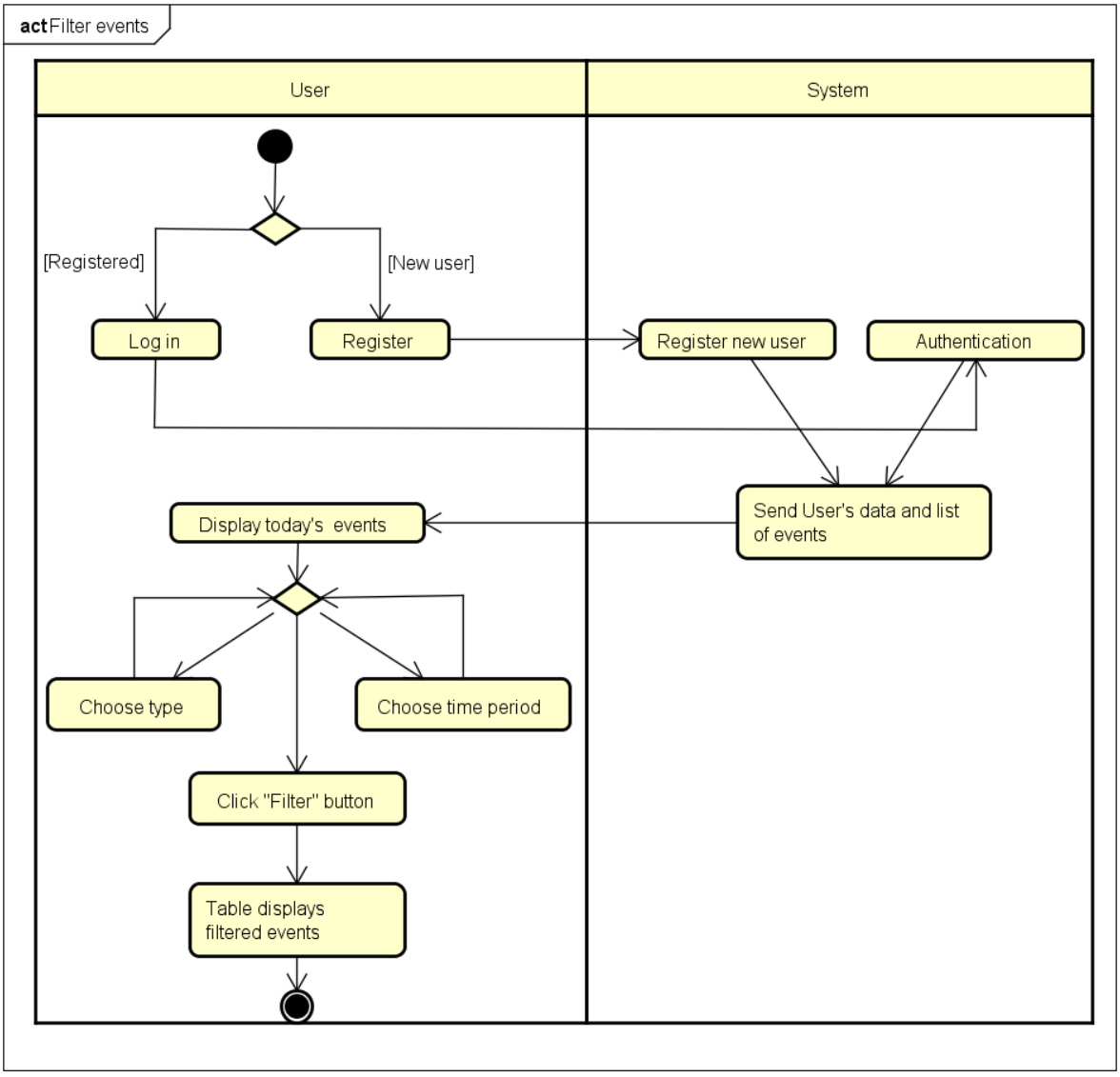
ITEM	VALUE
UseCase	Place a bet
Summary	New ticket is created and stored in the system's database
Actor	User System
Precondition	1. User is logged in 2. User's balance is greater than 0 3. User selected at least one match
Postcondition	1. New ticket is created 2. Money are withdrawn from the user's account 3. Ticket is added to the user's betting history
Base Sequence	1. User chooses one or more matches 2. User places bets on all selected matches 3. User writes the amount of money he/she wants to bet 4. User clicks on the "Place a bet" button 5. System checks if the user has enough money on his/her account 6. System creates a new Ticket 7. Ticket is stored in the system's database
Branch Sequence	
Exception Sequence	Not enough money on the user's account 1-4 as base sequence : 6. Not enough resources on the user's account Warning message appears Empty ticket 4. as base sequence Warning message appears
Sub UseCase	
Note	

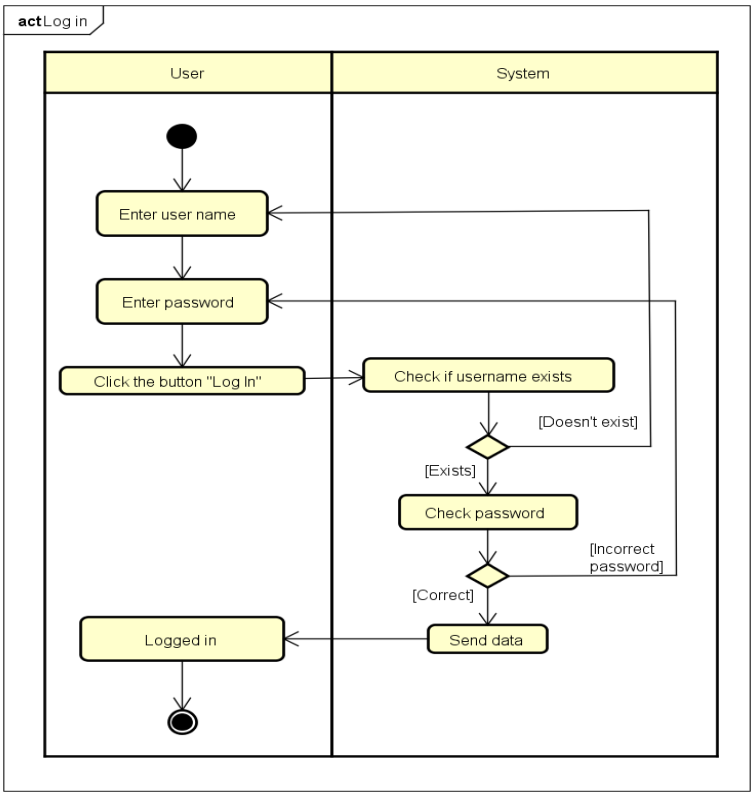
ITEM	VALUE
UseCase	See betting history
Summary	User can see all of his/her tickets
Actor	User System
Precondition	User has already placed at least one ticket
Postcondition	List of all tickets with detailed information about every ticket and match
Base Sequence	1. User clicks on the "History" tabbed pane 2. List of all the tickets appears 3. User selects one match from the list 4. User can see a detailed table of all matches from the selected ticket
Branch Sequence	
Exception Sequence	User has never placed a bet in this system 1 as base sequence: 2. No tickets in the user's history List shows a warning message
Sub UseCase	
Note	

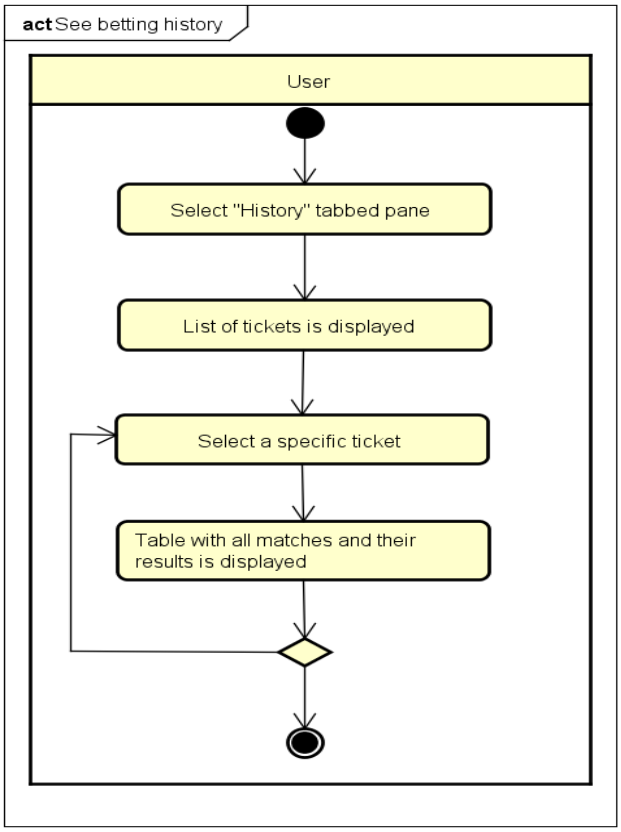
ITEM	VALUE
UseCase	Insert/edit events
Summary	Event is created/edited and stored in the system's database
Actor	Administrator System
Precondition	Administrator is connected to the database
Postcondition	Event is created/edited and stored in the system's database
Base Sequence	1. Administrator opens the program 2. Administrator fills in all the information about the new event 3. Admin hits the "Insert match" button 4. System checks the match ID 5. New match is created and stored in the system's database
Branch Sequence	
Exception Sequence	Match with given ID already exists 1 - 3 as base sequence : 4. Match with given ID already exists Warning message appears
Sub UseCase	
Note	

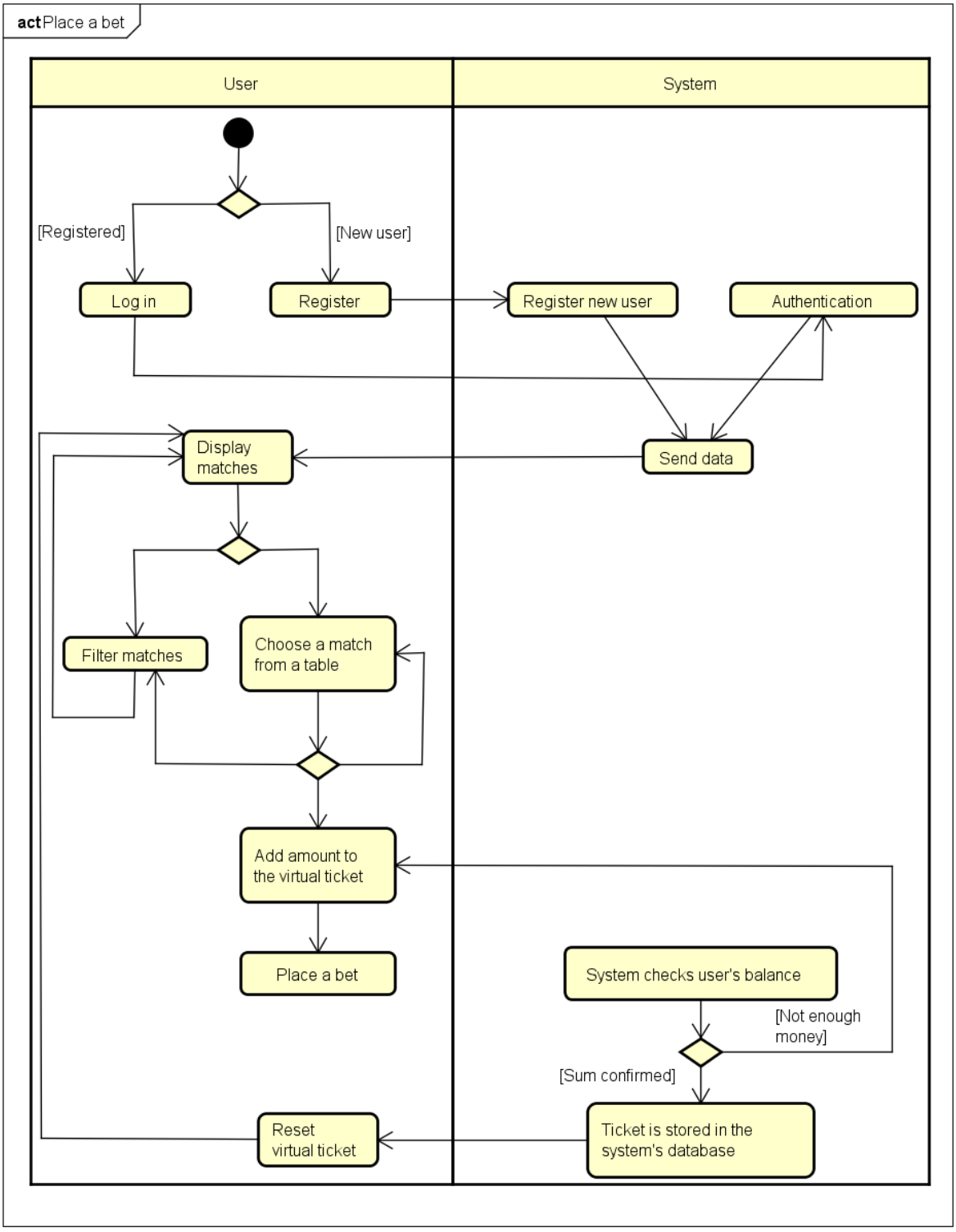
2. Activity diagrams

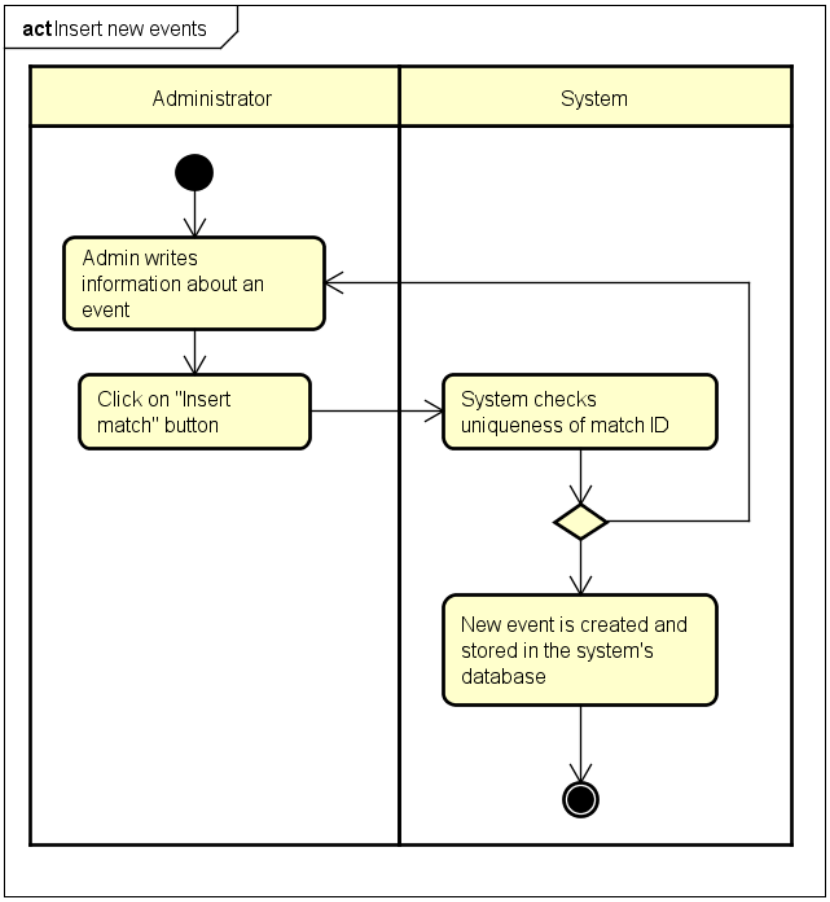


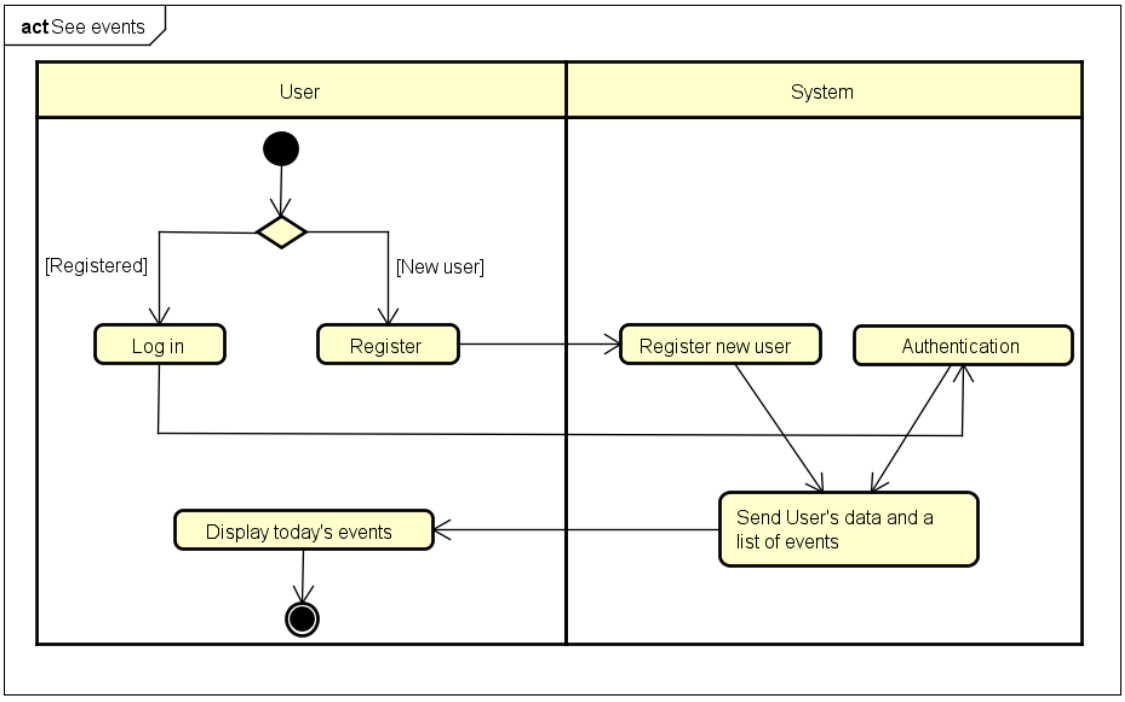

act Filter events
User
System
[Registered]
[New user]
Log in
Register
Register new user
Authentication
Send User's data and list of events
Display today's events
Choose type
Choose time period
Click "Filter" button
Table displays filtered events

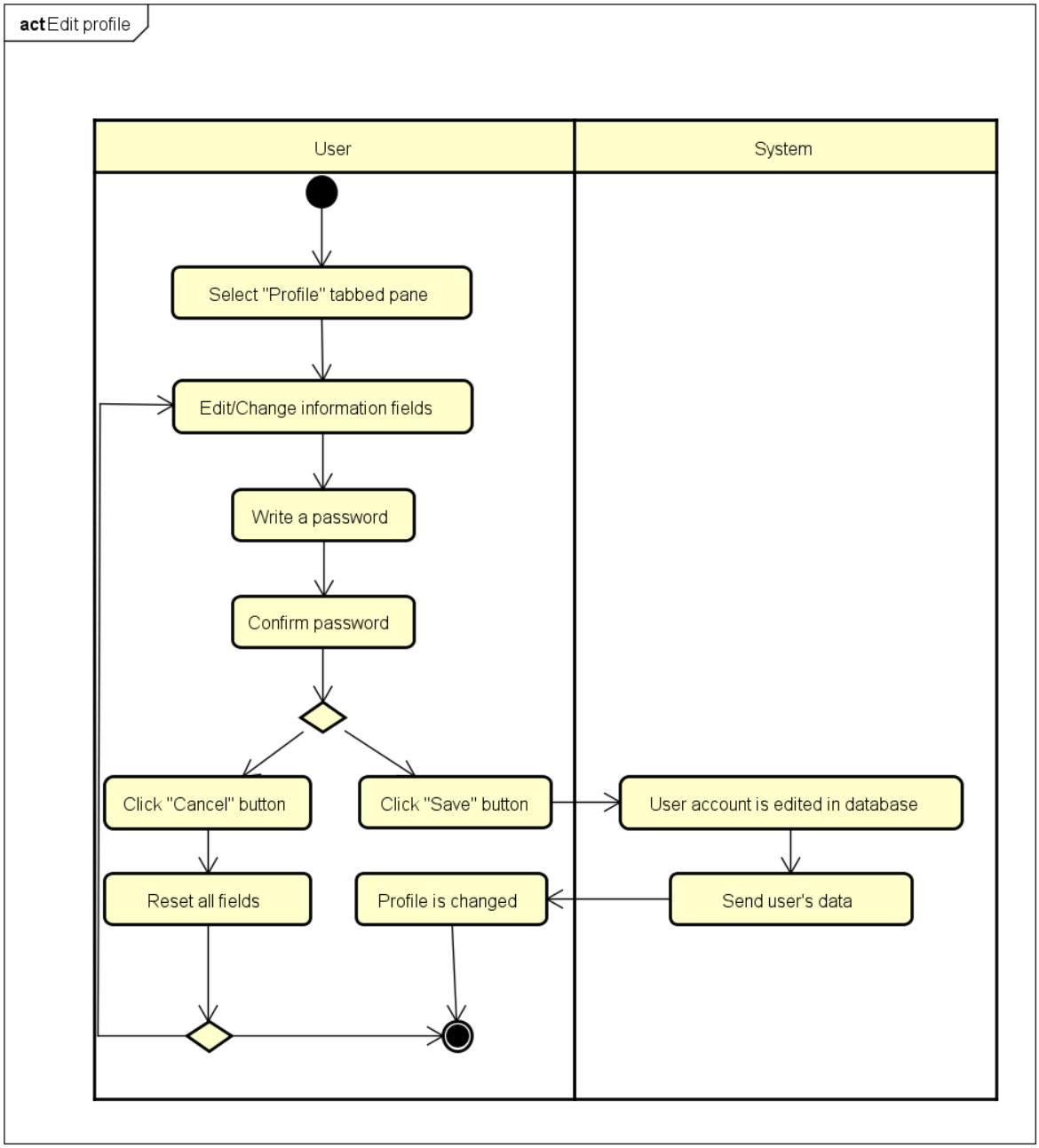

act Log in
User
System
Enter user name
Enter password
Click the button "Log In"
Check if username exists
[Doesn't exist]
[Exists]
Check password
[Incorrect password]
[Correct]
Logged in
Send data


act See betting history
User
Select "History" tabbed pane
List of tickets is displayed
Select a specific ticket
Table with all matches and their results is displayed


actPlace a bet
User
System
[Registered]
[New user]
Log in
Register
Register new user
Authentication
Display matches
Send data
Filter matches
Choose a match from a table
Add amount to the virtual ticket
Place a bet
System checks user's balance
[Not enough money]
[Sum confirmed]
Reset virtual ticket
Ticket is stored in the system's database

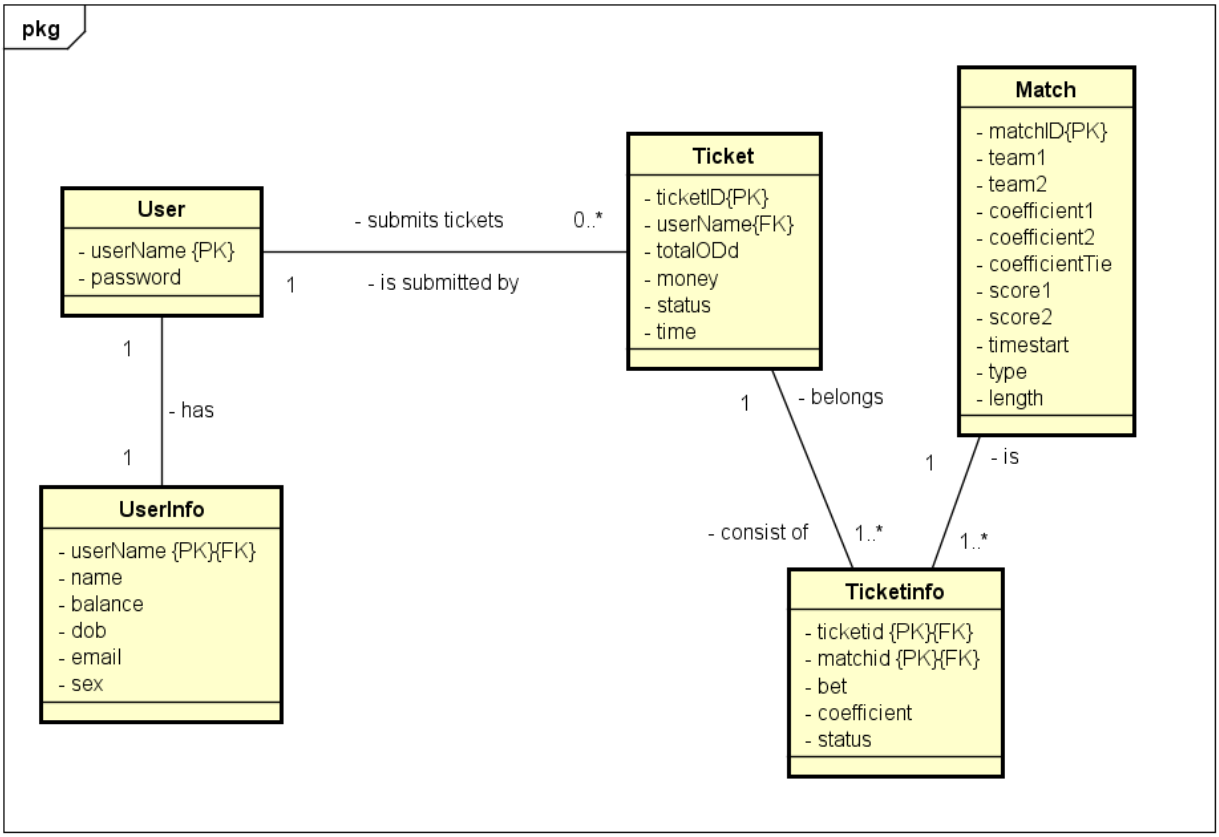

act Insert new events
Administrator
System
Admin writes information about an event
Click on "Insert match" button
System checks uniqueness of match ID
New event is created and stored in the system's database


act See events
User
System
[Registered]
[New user]
Log in
Register
Register new user
Authentication
Send User's data and a list of events
Display today's events


actEdit profile
User
System
Select "Profile" tabbed pane
Edit/Change information fields
Write a password
Confirm password
Click "Cancel" button
Click "Save" button
User account is edited in database
Reset all fields
Profile is changed
Send user's data

3. Database

3.1 EER diagram



3.2 SQL code

3.2.1 SQL code - NEW

```
CREATE SCHEMA bettingSystem;

CREATE TABLE login(
username    VARCHAR(255)    PRIMARY KEY NOT NULL,
password    VARCHAR(255)    NOT NULL);

CREATE TABLE userinfo(
username    VARCHAR(255)    PRIMARY KEY REFERENCES login(username),
name        VARCHAR(255)    NOT NULL,
balance     DECIMAL(8,2)    DEFAULT 100.00 CHECK(balance >= 0),
dob         DATE            NOT NULL,
email       VARCHAR(255),
sex         VARCHAR(10));

CREATE TABLE match(
matchid         SERIAL          PRIMARY KEY NOT NULL,
team1           VARCHAR(255),
team2           VARCHAR(255),
coefficient1    DECIMAL (4,2)   NOT NULL,
coefficient2    DECIMAL (4,2)   NOT NULL,
coefficienttie  DECIMAL (4,2)   NOT NULL,
score1          INTEGER         DEFAULT 0,
score2          INTEGER         DEFAULT 0,
timestart       TIMESTAMP       NOT NULL,
type            VARCHAR(255)    NOT NULL,
length          INTEGER);

CREATE TABLE ticket(
ticketid    SERIAL          PRIMARY KEY,
username    VARCHAR(255)    REFERENCES login(username),
totalodd    DECIMAL(8,2)    NOT NULL CHECK(totalodd > 0),
money       DECIMAL(6,2),
status      VARCHAR(15)     DEFAULT('Pending'),
timestart   TIMESTAMP);

CREATE TABLE ticketinfo(
ticketid      integer       REFERENCES ticket(ticketid),
matchid       integer       REFERENCES match(matchid),
bet           CHARACTER VARYING (1),
coefficient   numeric (6, 2),
status        CHARACTER VARYING (7),
PRIMARY KEY (ticketid,matchid));



--TRIGER FUNCTION FOR INSERT ACTUAL DATE WHEN NEW TICKET IS ADDED

CREATE OR REPLACE FUNCTION set_Ticket_Date() RETURNS
TRIGGER AS $$
BEGIN UPDATE ticket SET timestart = current_timestamp WHERE ticketid = new.ticketid;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER new_ticket
AFTER INSERT ON ticket FOR EACH ROW
EXECUTE PROCEDURE set_Ticket_Date();
```

3.2.2 SQL code – OLD

```
CREATE TABLE logIn(
userName    VARCHAR(255)    PRIMARY KEY NOT NULL,
password    VARCHAR(255)    NOT NULL);

CREATE TABLE userInfo(
userName    VARCHAR(255)    PRIMARY KEY REFERENCES logIn(userName),
name        VARCHAR(255)    NOT NULL,
balance     INTEGER         DEFAULT 100.00 CHECK(balance>=0),
dOB         DATE            NOT NULL,
email       VARCHAR(255));

CREATE TABLE match(
matchId         INTEGER         PRIMARY KEY NOT NULL,
team1           VARCHAR(255),
team2           VARCHAR(255),
coeficient1     real            NOT NULL,
coeficient2     real            NOT NULL,
coeficientTie   real            NOT NULL,
score1          INTEGER         DEFAULT 0,
score2          INTEGER         DEFAULT 0,
timee           TIMESTAMP       NOT NULL,
type            VARCHAR(255)    NOT NULL,
length          INTEGER);

CREATE TABLE ticket(
ID          INTEGER         PRIMARY KEY,
userName    VARCHAR(255)    REFERENCES logIn(userName),
matches     BYTEA,
totalOdd    REAL            NOT NULL CHECK(totalOdd>0),
money       REAL,
status      VARCHAR(15)     DEFAULT('Pending'),
time        TIMESTAMP);
```

3.3 Third normal form

LogIn

Username	Password
dano	password123
naxxo	atanas96
iamstefan	123456

UserInfo

Username	Name	Balance	DOB	Email
dano	Daniel Hamarik	125	1996-02-12	daniel.hamarik@gmail.com
naxxo	Atanas Latinov	72	1996-05-17	naxxo.king@gmail.com
iamstefan	Stefan Aleksiev	89	1996-09-15	iamstefanaleksiev@gmail.com

Match

MatchID	Team1	Team2	Coefficient1	Coefficient2	CoefficientTie	Score1	Score2	Time	Type
101	Barcelona	Real	2.0	2.1	5.5	0	3	15:00:00	Football
105	Liverpool	Bayern	1.8	2.5	6.1	2	1	16:00:00	Football
102	Manchester	Juventus	1.1	3.0	8.8	0	0	19:00:00	Football
103	Horsens	Vejle	1.2	2.8	4.6	4	1	15:30:00	Football

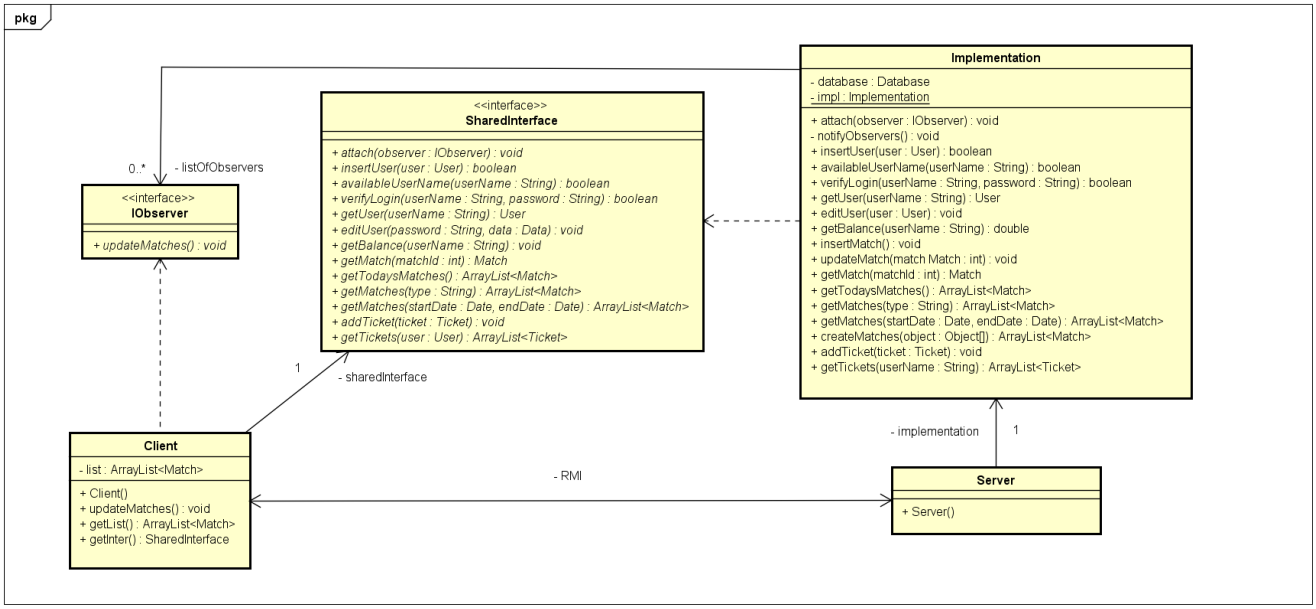
TicketInfo

TicketNo	MatchID	Bet	Coefficient	Status
1	103	1	1.2	Win
1	105	2	2.5	Lose
2	101	X	5.5	Lose
2	103	1	1.2	Win
2	105	1	1.8	Win

Ticket

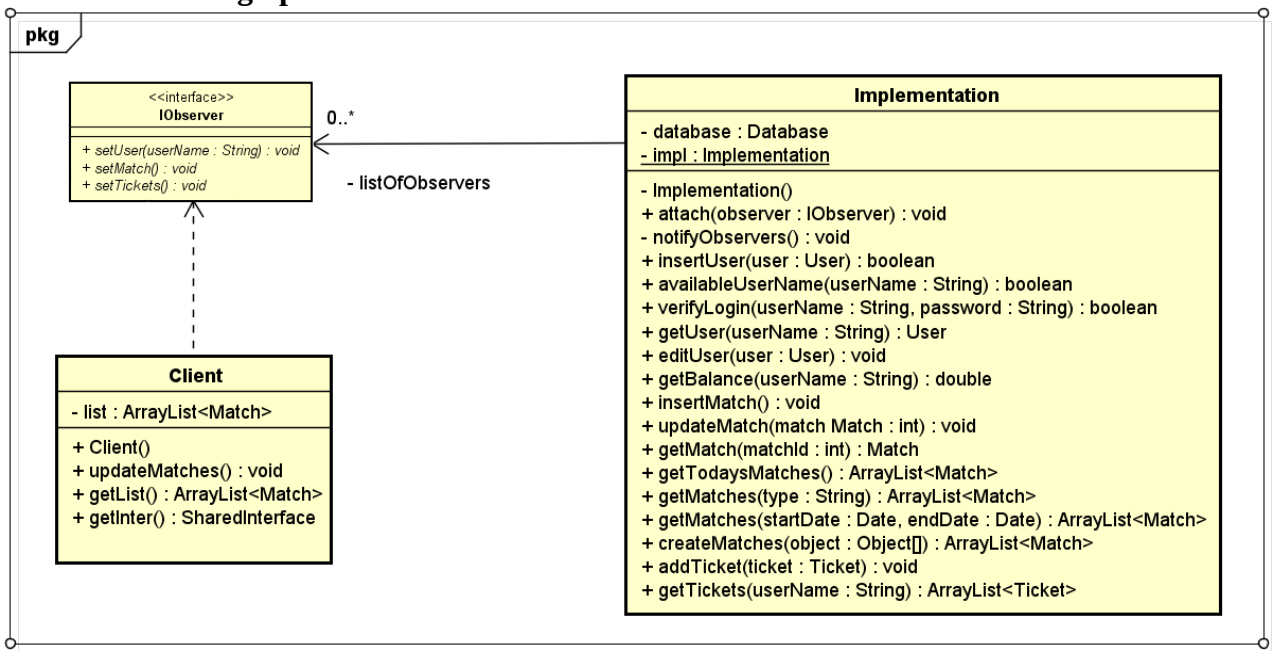
TicketNo	Username	totalOdd	Money	Status	Time
1	dano	3.0	10.0	Lose	2016-05-12 15:28:15
2	naxxo	7.1	5.0	LOSE	2016-05-12 18:42:38

4. Java remote method invocation (RMI)

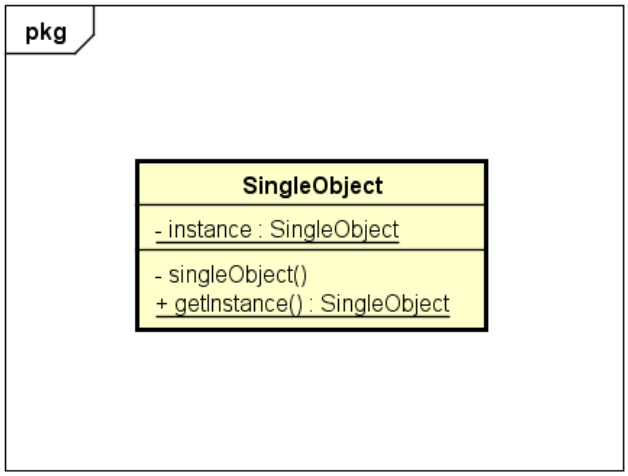


5. Design pattern

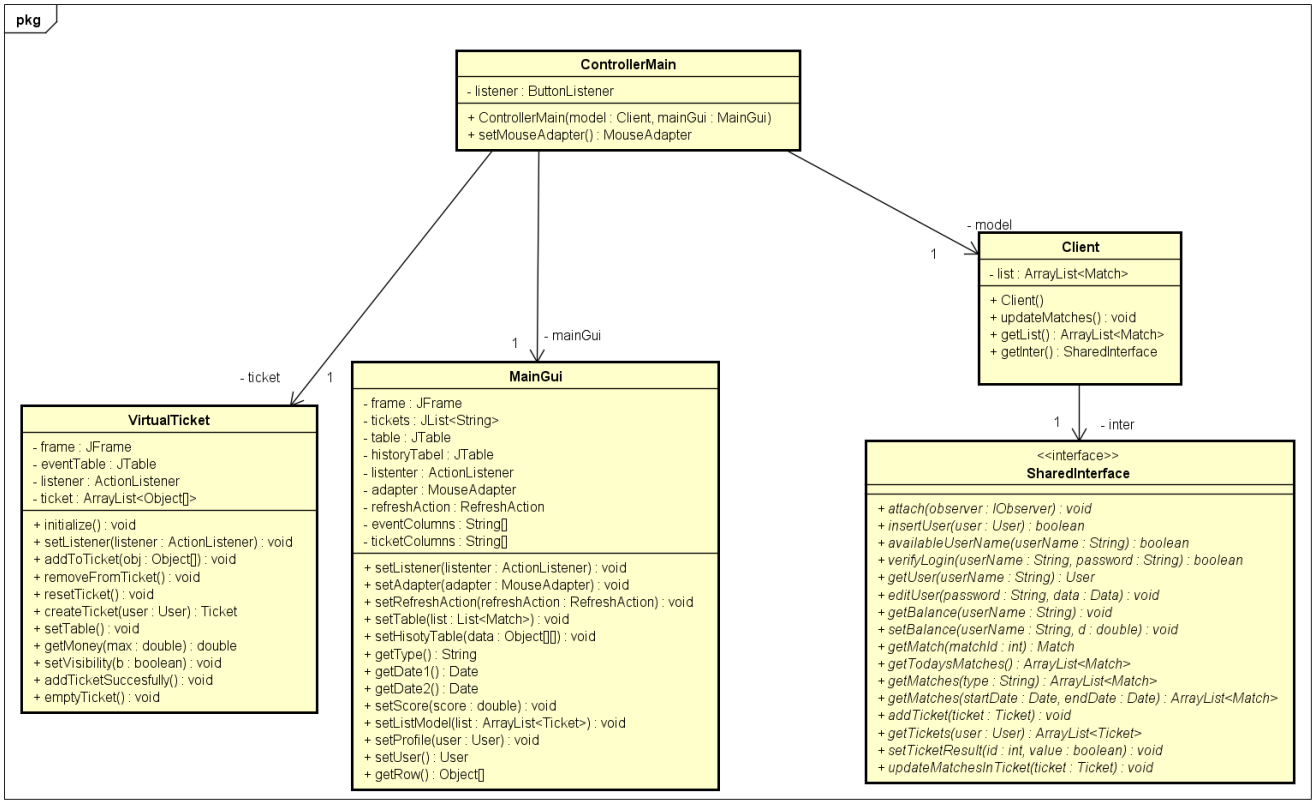
5.1 Observer design pattern



5.2 Singleton design pattern



5.3 Model–view–controller



6. Class diagram