**4.** **Software Design Description**

**4.1.** **Purpose**

This chapter states what the system’s architecture is to the developer team and how they should be implemented. The chapter includes:

* Architecture overview
* Component diagram
* Database design
* Detailed design

**4.2.** **Architecture Overview**

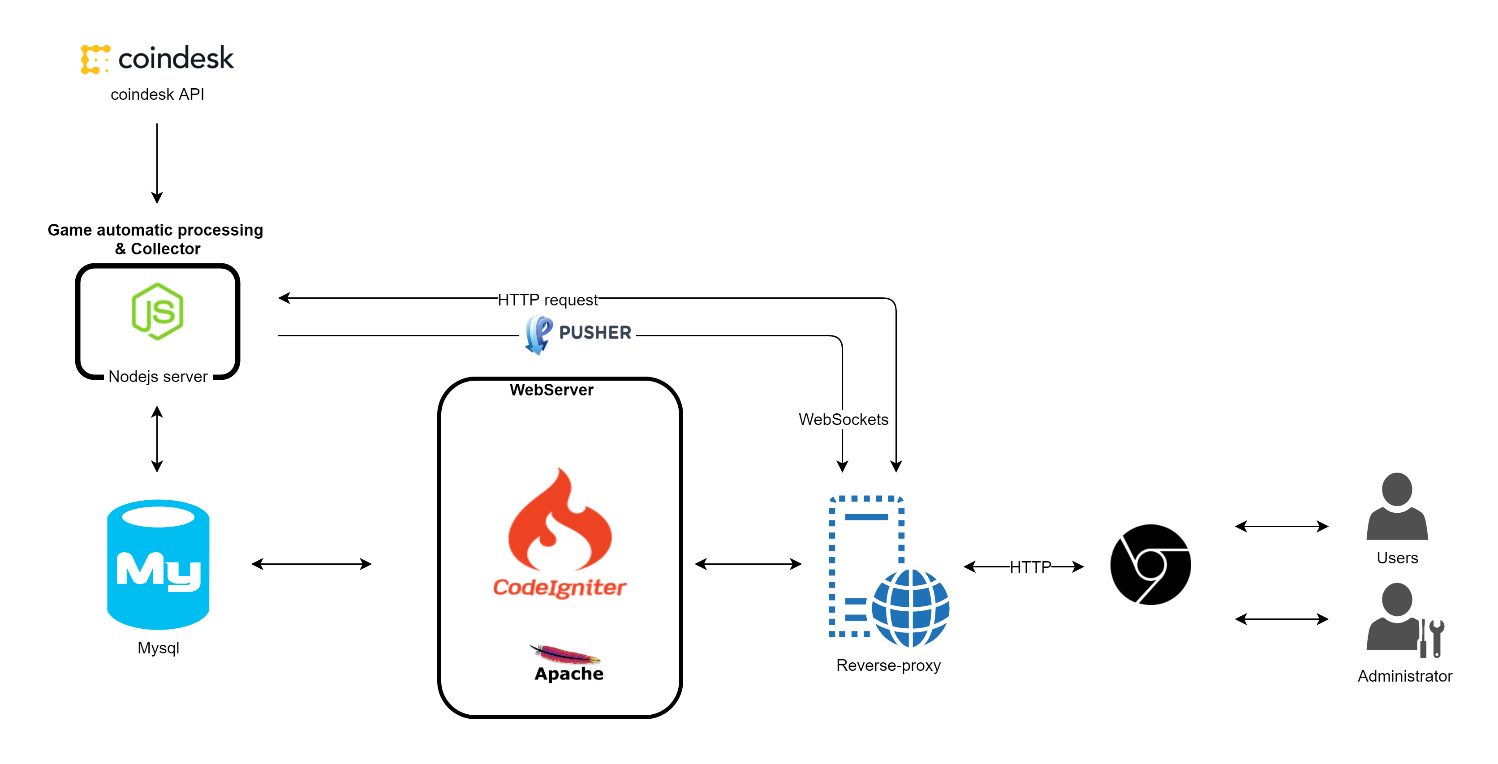


Figure 1: Lowhope system's architecture

**4.2.1.** **System Architecture Explanation**

Our system uses some 3-third party services to aim at delivering a secured, responsive and highly available system. In the following section, we will explain the function and mechanism of each unit used.

**4.2.1.1. CoinDesk**



Figure 2: CoinDesk

CoinDesk is the leading digital media, events and information services company for the digital asset and blockchain technology community. Its mandate is to inform, educate and connect the global community as the authoritative daily news provider dedicated to chronicling the space.

**4.2.1.2. Nodejs**



Figure 3: Nodejs

Node.js® is a JavaScript runtime built on **Chrome's V8 JavaScript engine**. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

**4.2.1.3. Pusher**

****

Figure 4: Pusher

Pusher is a hosted service that makes it super-easy to add **real-time data** and functionality to web and mobile applications.

Pusher sits as a **real-time layer** between your **servers** and **your clients**. Pusher maintains **persistent connections to the clients - over WebSocket** if possible and falling back to HTTP-based connectivity - so that as soon as your servers have new data that they want to push to the clients they can do, instantly via Pusher.

**4.2.1.4. Mysql**



Figure 5: Mysql

MySQL is the world's most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, Yahoo! and many more.

**4.2.1.5. CodeIgniter – PHP framework**

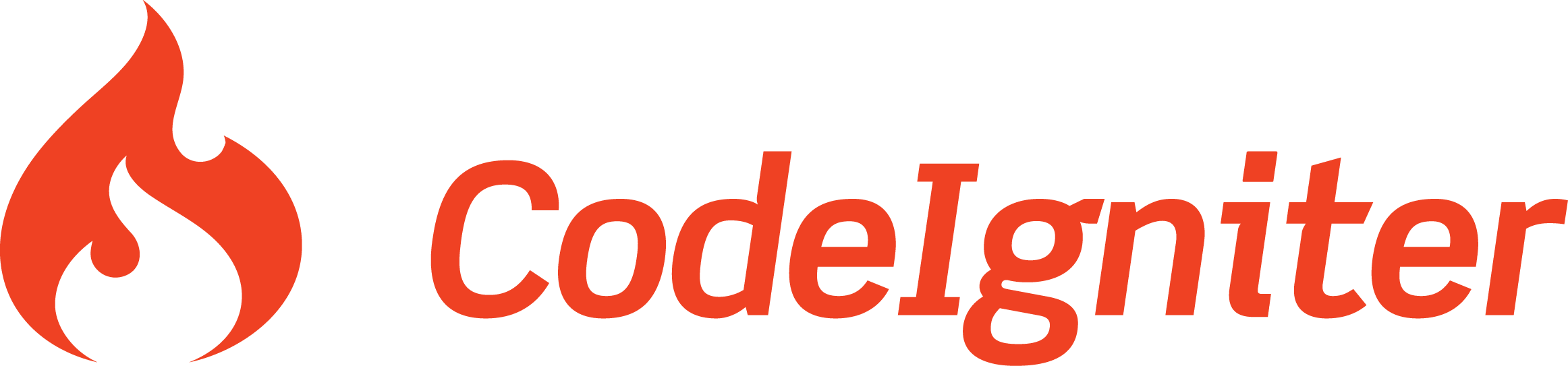


Figure 6: CodeIgniter

CodeIgniter is a powerful **PHP framework** with a very **small footprint**, built for developers who need a simple and elegant toolkit to **create full-featured web applications**.

**4.2.1.6. Apache**

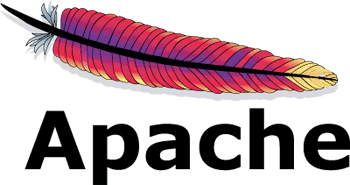


Figure 7: Apache

The Apache HTTP Server is free and open-source cross-platform web server software, released under the terms of Apache License 2.0. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation.

**4.2.1.7. Reverse proxy server – Ngix**

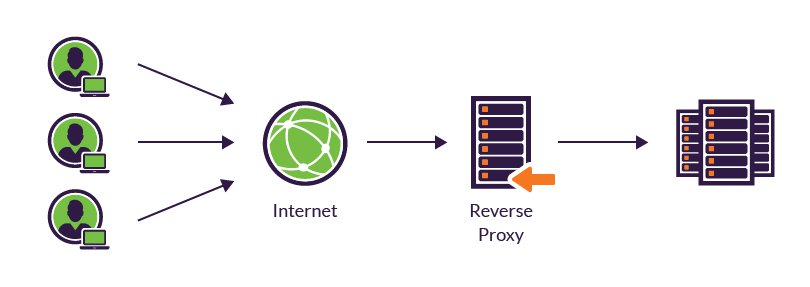


Figure 8: Reverse proxy

A **reverse proxy server** is a type of proxy server that typically sits behind the firewall in a private network and directs client requests to the appropriate backend server. A reverse proxy provides an additional level of abstraction and control to ensure the smooth flow of network traffic between clients and servers. **Common uses** for reverse proxy server include: **load balancing**, **Web acceleration**, **Securit and anomymity**.

**4.3. System Design**

**4.3.1. Component Diagram**

**N/A**

**4.3.2. Database Design**

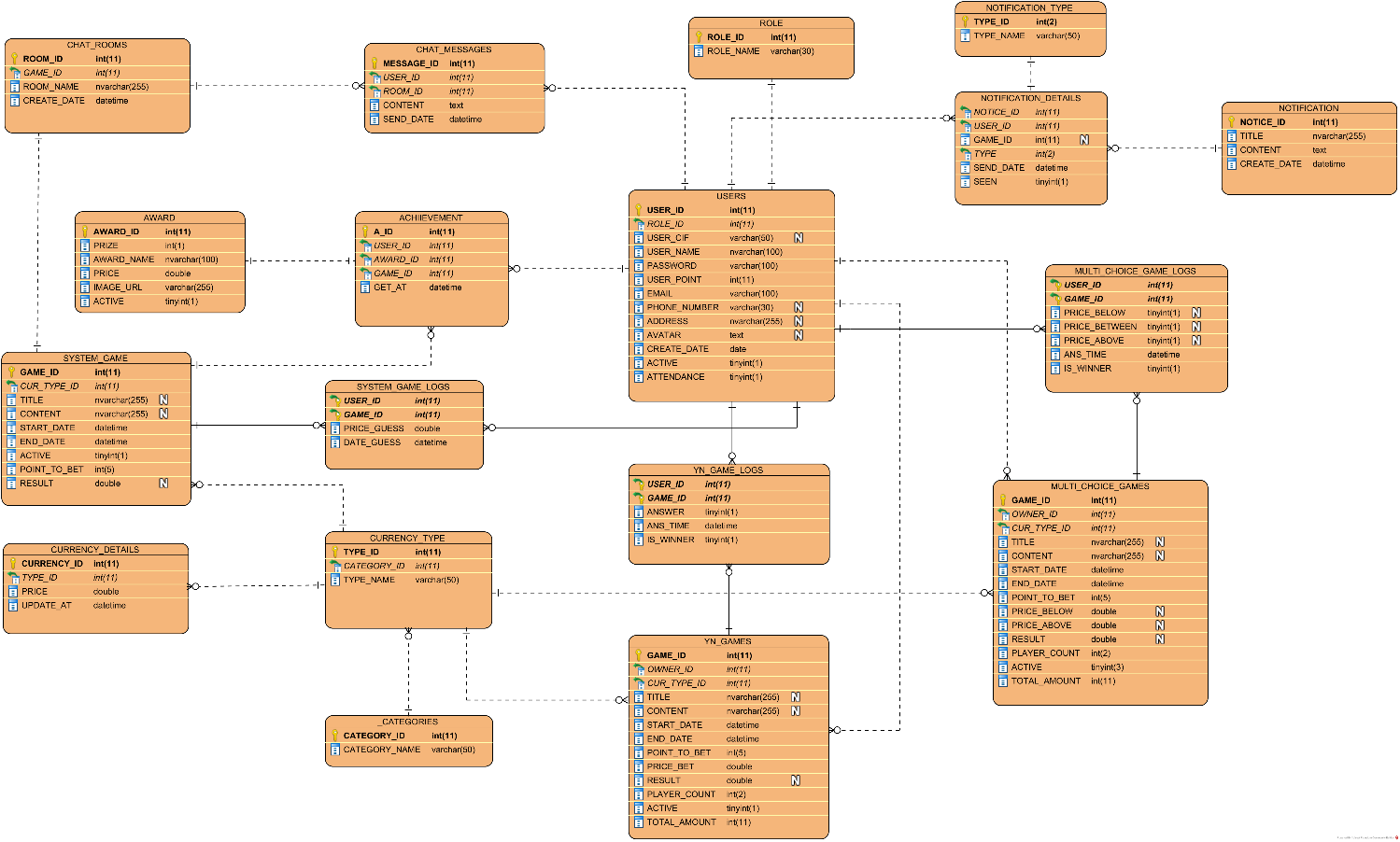


Figure 9: Database design

**4.3.2.1. Data Dictionary**

**4.3.2.1.1. Entities**

|  |  |  |
| --- | --- | --- |
| **No** | **Entities name** | **Description** |
| 1. | ROLE | ROLE table store users’ role information. |
| 2. | USERS | To store users’ information. |
| 3. | NOTIFICATION | To store the notice sent to user when necessary. |
| 4. | NOTIFICATION\_TYPE | Each notice have one type that it’s to be applied for. |
| 5. | NOTIFICATION\_DETAILS | To store logs when messages sent to users. |
| 6. | CHAT\_ROOMS | Each system game have a chat channel for users chatting. |
| 7. | CHAT\_MESSAGES | To store message that is commented by user to system game. |
| 8. | CATEGORIES | To store currency category information. |
| 9. | CURRENCY\_TYPE | Table stores currency type. |
| 10. | CURRENCY\_DETAILS | To store information about a specificed currency unit at a time. |
| 11. | SYSTEM\_GAMES | To store the game automatically created by system. |
| 12. | SYSTEM\_GAME\_LOGS | To store history of system game. |
| 13. | AWARD | To store gift for the 3 first prizes of system game. |
| 14. | ACHIEVEMENT | To store winners of system game. |
| 15. | YN\_GAMES | To store binary option game (true/faslse) created by users. |
| 16. | YN\_GAME\_LOGS | To store history of yn\_game. |
| 17. | MULTI\_CHOICE\_GAMES | To store multichoice game created by user. |
| 18. | MULTI\_CHOICE\_GAME\_LOGS | To store history of multi choice game. |

**4.3.2.1.2. Attributes**

|  |  |  |
| --- | --- | --- |
| **ROLE** | | |
| **Field Name** | **Description** | **Data Type** |
| ROLE\_ID | Unique identifier | int(11) |
| ROLE\_NAME | Name of role |  |

|  |  |  |
| --- | --- | --- |
| **USERS** | | |
| **Field Name** | **Description** | **Data Type** |
| USER\_ID | Unique identifer | int(11) |
| ROLE\_ID | Reference to ROLE to classify user | int(11) |
| USER\_CIF | The user’s id is returned from facebook or google while login with these providers | varchar(50) |
| USER\_NAME | Name of user | nvarchar(100) |
| PASSWORD | Password to login | varchar(100) |
| USER\_POINT | Point to join game | int(1) |
| EMAIL | User’s email | varchar(100) |
| PHONE\_NUMBER | User’s phone number | varchar(30) |
| ADDRESS | User’s address | nvarchar(255) |
| AVATAR | User’s profile image | text |
| CREATE\_DATE | The date that user join the system | datetime |
| ACTIVE | To block or active user | bool |
| ATTENDANCE | To check if user login on the new day | bool |

|  |  |  |
| --- | --- | --- |
| **NOTIFICATION** | | |
| **Field Name** | **Description** | **Data Type** |
| NOTICE\_ID | Unique identifier | int(11) |
| TITLE | Notification title | nvarchar(255) |
| CONTENT | Notification content | text |
| CREATE\_DATE | The create date | datetime |

|  |  |  |
| --- | --- | --- |
| **NOTIFICATION\_TYPE** | | |
| **Field Name** | **Description** | **Data Type** |
| TYPE\_ID | Unique identifier | int(11) |
| TYPE\_NAME | Type’s name | varchar(50) |

|  |  |  |
| --- | --- | --- |
| **NOTIFICATION\_DETAILS** | | |
| **Field Name** | **Description** | **Data Type** |
| NOTICE\_ID | Reference to NOTIFICATION that notice is sent | int(11) |
| USER\_ID | Reference to USERS that receive the notice | int(11) |
| GAME\_ID | Reference to SYSTEM\_GAME that notice\_id belongs to | int(11) |
| TYPE\_ID | Reference to NOTIFICATION\_TYPE | int(11) |
| SEND\_DATE | The date that notice is sent | datetime |
| SEEN | The status of notice | bool |

|  |  |  |
| --- | --- | --- |
| **CHAT\_ROOMS** | | |
| **Field Name** | **Description** | **Data Type** |
| ROOM\_ID | Unique identifier | int(11) |
| GAME\_ID | Reference to SYSTEM\_GAME that have chat channel | int(11) |
| ROOM\_NAME | Room’s name | nvarchar(255) |
| CREATE\_DATE | The date that room is created | datetime |

|  |  |  |
| --- | --- | --- |
| **CHAT\_MESSAGES** | | |
| **Field Name** | **Description** | **Data Type** |
| MESSAGE\_ID | Unique identifer | int(11) |
| USER\_ID | Reference to USER that send the message | int(11) |
| ROOM\_ID | Reference to CHAT\_ROOM that contain the message | int(11) |
| CONTENT | Message’s content | text |
| SEND\_DATE | The date message is sent | datetime |

|  |  |  |
| --- | --- | --- |
| **CATEGORIES** | | |
| **Field Name** | **Description** | **Data Type** |
| CATEGORY\_ID | Unique identifer | int(11) |
| CATEGORY\_NAME | Category’s name | varchar(50) |

|  |  |  |
| --- | --- | --- |
| **CURRENCY\_TYPE** | | |
| **Field Name** | **Description** | **Data Type** |
| CURRENCY\_TYPE | Unique identifer | int(11) |
| CATEGORY\_ID | Reference to CATEFORIES | int(11) |
| TYPE\_NAME | Type name | varchar(50) |

|  |  |  |
| --- | --- | --- |
| **CURRENCY\_DETAILS** | | |
| **Field Name** | **Description** | **Data Type** |
| CURRENCY\_ID | Unique identifier | int(11) |
| TYPE\_ID | Refernce to CURRENCY\_TYPE | int(11) |
| PRICE | Price of currency unit at a time | double |
| UPDATE\_AT | The time that currency is to be value at | datetime |

|  |  |  |
| --- | --- | --- |
| **SYSTEM\_GAMES** | | |
| **Field Name** | **Description** | **Data Type** |
| GAME\_ID | Unique identifier | int(11) |
| CUR\_TYPE\_ID | Reference to CUR\_TYPE\_ID that game use to bet | int(11) |
| TITLE | Game title | nvarchar(255) |
| CONTENT | Game content | nvarchar(255) |
| START\_DATE | Start date | datetime |
| END\_DATE | End date | datetime |
| ACTIVE | Status of game(active/not) | bool |
| POINT\_TO\_BET | Minimum point to join game | int(5) |
| RESULT | Game result when the game finish | double |

|  |  |  |
| --- | --- | --- |
| **SYSTEM\_GAME\_LOG** | | |
| **Field Name** | **Description** | **Data Type** |
| USER\_ID | Reference to USERS who play game | int(11) |
| GAME\_ID | Reference to SYSTEM\_GAME | int(11) |
| PRICE\_GUESS | The price user takes for guessing | double |
| DATE\_GUESS | The date the user leave their answer | datetime |

|  |  |  |
| --- | --- | --- |
| **AWARD** | | |
| **Field Name** | **Description** | **Data Type** |
| AWARD\_ID | Unique identifier | int(11) |
| PRIZE | The order that players get | int(1) |
| AWARD\_NAME | Award name | nvarchar(100) |
| PRICE | Value of price | varchar(255) |
| IMAGE\_URL | Image of award | double |
| ACTIVE | Status of award | bool |

|  |  |  |
| --- | --- | --- |
| **ACHIVEMENT** | | |
| **Field Name** | **Description** | **Data Type** |
| A\_ID | Unique identifier | int(11) |
| USER\_ID | Reference to USERS that get achivement | int(11) |
| AWARD\_ID | Reference to AWARD for each achivement | int(11) |
| GAME\_ID | Reference to SYSTEM\_GAME that have winners | int(11) |
| GET\_AT | The date that system award users | datetime |

|  |  |  |
| --- | --- | --- |
| **YN\_GAMES** | | |
| **Field Name** | **Description** | **Data Type** |
| GAME\_ID | Unique identifier | int(11) |
| OWNER\_ID | Reference to USERS that create the game | int(11) |
| CUR\_TYPE\_ID | Reference to CUR\_TYPE\_ID that game use to bet | int(11) |
| TITLE | Game title | nvarchar(255) |
| CONTENT | Game content | nvarchar(255) |
| START\_DATE | Start date | datetime |
| END\_DATE | End date | datetime |
| POINT\_TO\_BET | Minimum point to join game | int(5) |
| PRICE\_BET | Price given by owner to bet | double |
| RESULT | Game result when the game finishs | double |
| PLAYER\_COUNT | Number of players join the game (<=40) | int(2) |
| ACTIVE | Status of game(active/not) | bool |
| TOTAL\_AMOUNT | Total point get from players  (PLAYER\_COUNT\*POINT\_TO\_BET) | int(11) |

|  |  |  |
| --- | --- | --- |
| **YN\_GAME\_LOGS** | | |
| **Field Name** | **Description** | **Data Type** |
| USER\_ID | Reference to USERS that user play game | int(11) |
| GAME\_ID | Reference to YN\_GAMES | int(11) |
| ANSWER | User’s answer | bool |
| ANS\_TIME | The date user leave their answer | datetime |
| IS\_WINNER | Status of user when game finishes(1/0) | bool |

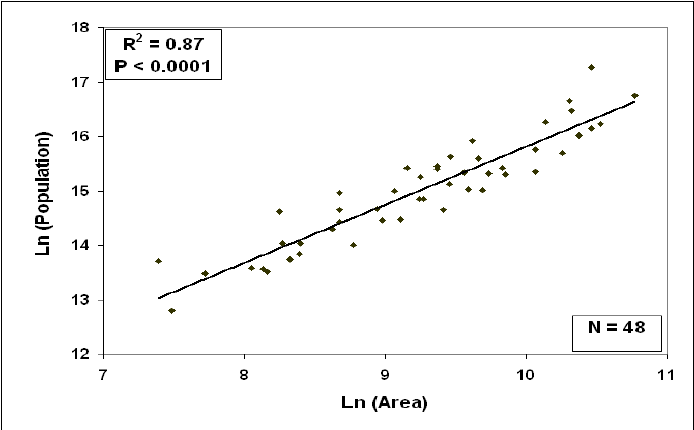
|  |  |  |
| --- | --- | --- |
| **MULTI\_CHOICE\_GAMES** | | |
| **Field Name** | **Description** | **Data Type** |
| GAME\_ID | Unique identifier | int(11) |
| OWNER\_ID | Reference to USERS that create the game | int(11) |
| CUR\_TYPE\_ID | Reference to CUR\_TYPE\_ID that game use to bet | int(11) |
| TITLE | Game title | nvarchar(255) |
| CONTENT | Game content | nvarchar(255) |
| START\_DATE | Start date | datetime |
| END\_DATE | End date | datetime |
| POINT\_TO\_BET | Minimum point to join game | int(5) |
| PRICE\_BELOW | Minimum value of currency unit that user raise to bet | double |
| PRICE\_ABOVE | Maximum value of currency unit that user raise to bet | double |
| RESULT | Game result when the game finishes | double |
| PLAYER\_COUNT | Number of players join the game (<=40) | int(2) |
| ACTIVE | Status of game(active/not) | bool |
| TOTAL\_AMOUNT | Total point get from players  (PLAYER\_COUNT\*POINT\_TO\_BET) | int(11) |
| **YN\_GAME\_LOGS** | | |
| **Field Name** | **Description** | **Data Type** |
| USER\_ID | Reference to USERS that user play game | int(11) |
| GAME\_ID | Reference to YN\_GAMES | int(11) |
| PRICE\_BELOW | Option that player choose. True if the real price < MULTI\_CHOICE\_GAMES (PRICE\_BELOW) | bool |
| PRICE\_BETWEEN | Option that player choose. True if the MULTI\_CHOICE\_GAMES (PRICE\_BELOW) <= real price <= MULTI\_CHOICE\_GAMES (PRICE\_ABOVE) | bool |
| PRICE\_ABOVE | Option that player choose. True if the real price > MULTI\_CHOICE\_GAMES (PRICE\_ABOVE) | bool |
| ANS\_TIME | The date user leave their answer | datetime |
| IS\_WINNER | Status of user when game finishes(1/0) | bool |

**4.3.3. Common Design**

**4.3.3.1. Bitcoin Price Prediction**

With the aim of support player before they leave their answer in guessing price, our system will indicate the price at the time that the system game ends as suggestion. We use Linear Regression model widely used in statistics for this approach.

**Linear Regression**



Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory (independent) variable, and the other is considered to be a dependent variable.

A linear regression line has an equation of the form

  is the explanatory variable

  is the dependent variable.

is the slope of the line and  is the intercept

We now have problem of using sample data to cpmpute estimates of and then build a linear equation that may be best fitting line for the observed data.

**Least-squares method**

The most common method for fitting a regression line is the method of least-squares. This method calculates the best-fitting line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line.

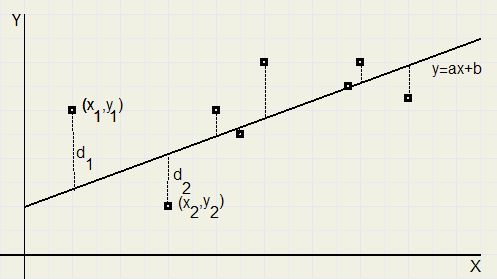


Figure 10: Least square method

Here we use least square method means the best model will have the smallest value for

…

**Correlation coefficient**

The strength of the linear association between two variables is quantified by the ***correlation coefficient.***

So we use this index to assess the model is fit our data set or not.

We have data set as bitcoin data updated every minute. The form of data like **[time, price]** and use the R programming language to estimate this index. If , the model fits our observed data. We use data set with about 5000 records as bitcoin data.

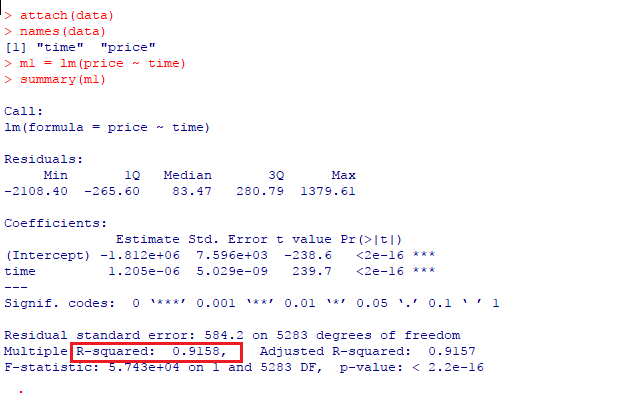


Figure 11: R programing language

Here we get the value is **0.9158**

**Implementation**

In nodejs, there is module for linear regression call **simple-statistics**

Code demo:



Figure 12: simple-statistics code demo

**4.3.4. Detail design**

**4.3.4.1. Admin Detail Design**

…

**4.3.4.2. User Detail Design**

…

**4.3.4.3. Game Automatic Processing Detail Design**

**4.3.4.3.1. System Game Processing**

**Screen Design**

N/A

**Class Diagram**

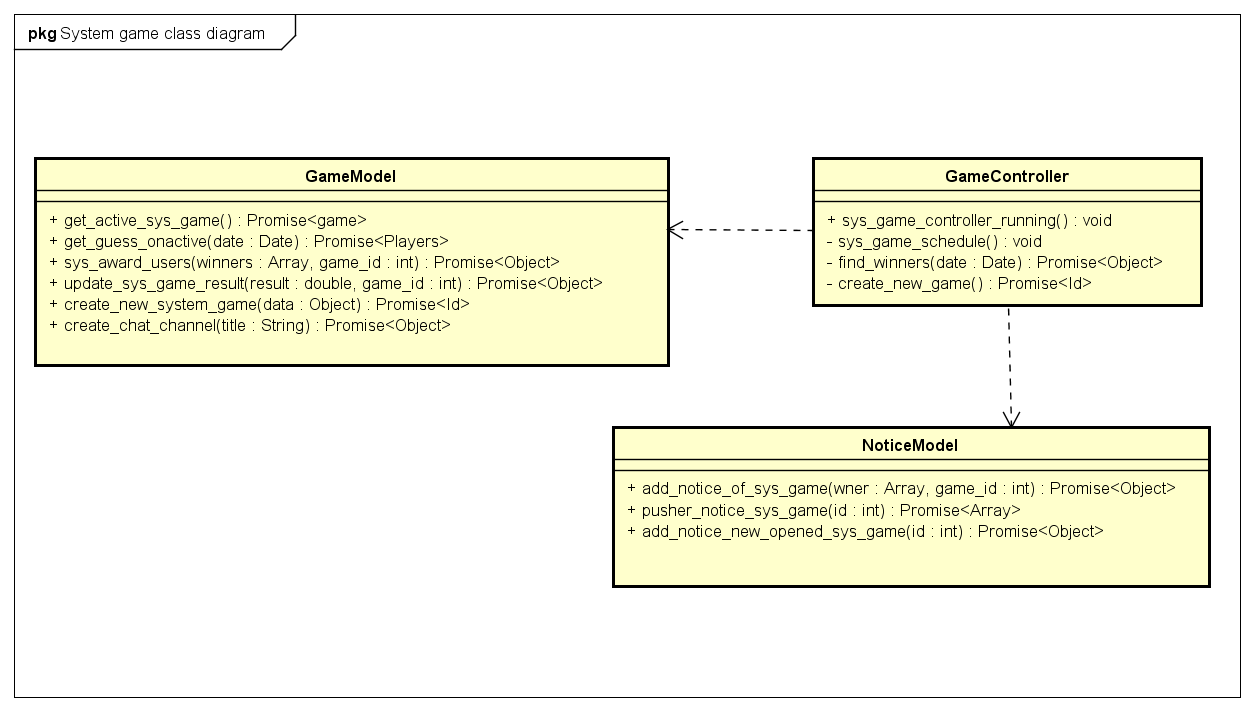


Figure 13: System game processing class diagram

**Class Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **GameController** | | | |
| **Physical address** | node-server/controller/gameController.js | | |
| **Base class** |  | | |
| **Attributes** | | | |
| N/A | | | |
| **Operations** | | | |
| sys\_game\_controller\_running | | | |
| **Return Type** | void | | |
| **Parameters** | N/A | | |
| sys\_game\_schedule | | | |
| **Return Type** | void | | |
| **Parameters** | N/A | | |
| find\_winners | | | |
| **Return Type** | Promise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | date | Date | The date that the system game ends |
| create\_new\_game | | | |
| **Return Type** | Promise<Id> | | |
| **Parameters** | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **GameModel** | | | |
| **Physical address** | node-server/models/gameModel.js | | |
| **Base class** |  | | |
| **Attributes** | | | |
| N/A | | | |
| **Operations** | | | |
| get\_active\_sys\_game | | | |
| **Return Type** | Promise<game> | | |
| **Parameters** | N/A | | |
| get\_guess\_onactive | | | |
| **Return type** | Promise<Players> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | date | Date | The date that the system game ends |
| sys\_award\_users | | | |
| **Return Type** | Promise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | winners | Array | 3 first prizes of system game |
|  | game\_id | Int | Id of system game |
| update\_sys\_game\_result | | | |
| **Return Type** | Promise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Result | Double | Bitcoin price when the system game ends |
|  | Game\_id | Int | Id of game |
| create\_new\_system\_game | | | |
| **Return type** | Promise<Id> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Data | Json object | Data for system game  data = { TITLE, CONTENT,  START\_DATE, END\_DATE,  ACTIVE,  POINT\_TO\_BET,  CUR\_TYPE\_ID  } |
| create\_chat\_channel | | | |
| **Return type** | Prmoise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Title | String | Name of chat room |

|  |  |  |  |
| --- | --- | --- | --- |
| **NoticeModel** | | | |
| **Physical address** | node-server/models/noticeModel.js | | |
| **Base class** |  | | |
| **Attributes** | | | |
| N/A | | | |
| **Operations** | | | |
| add\_notice\_of\_sys\_game | | | |
| **Return Type** | Promise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Winners | Array | Players won the system game |
|  | Game\_id | Int | The system game |
| pusher\_notice\_sys\_game | | | |
| **Return type** | Promise<Array> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Id | Int | The id of game to get notification for user |
| add\_notice\_new\_opened\_sys\_game | | | |
| **Return type** | Promise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Id | Int | Id of game that use notification |

**Sequence diagram**

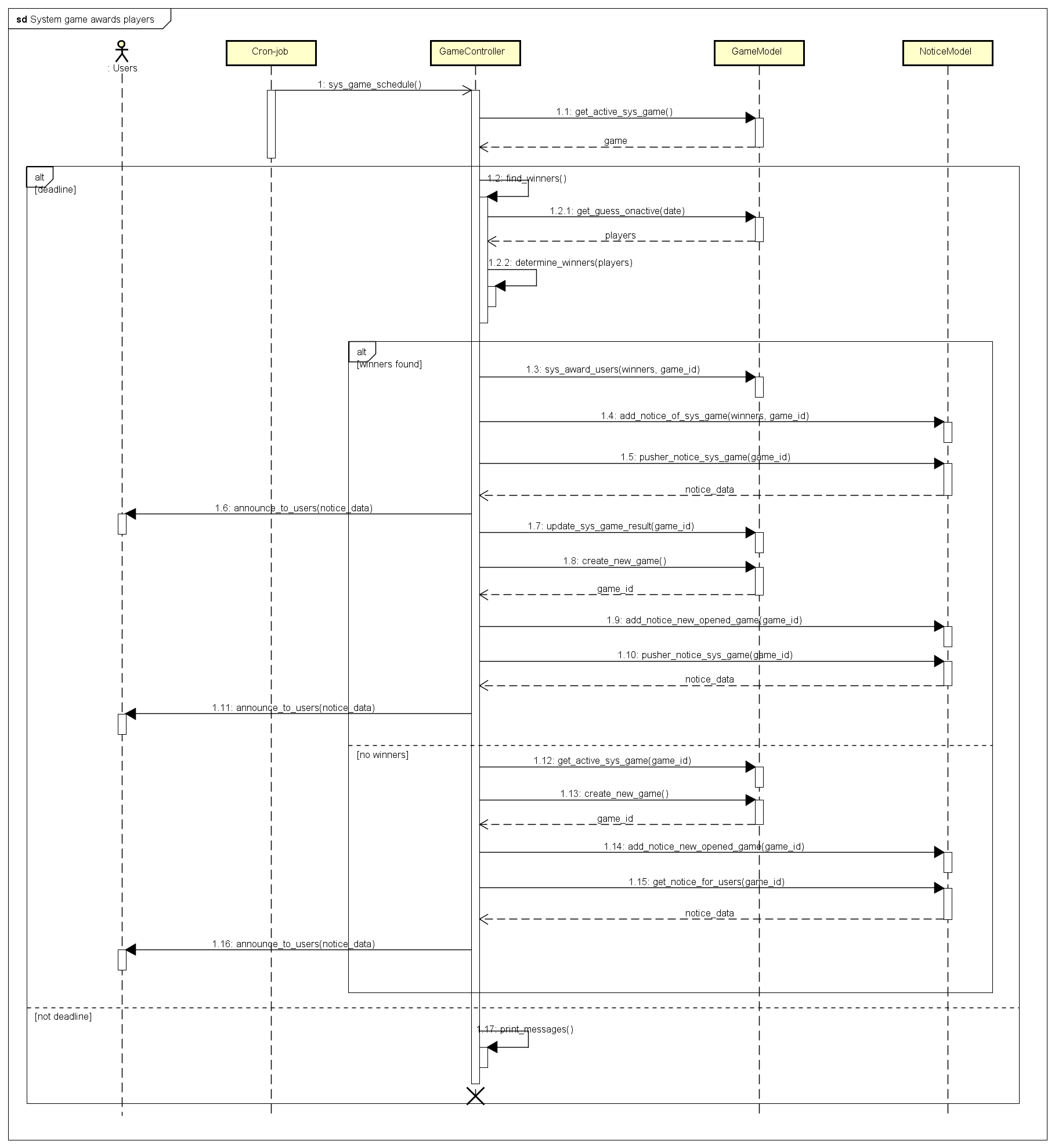
****

Figure 14: System game processing sequence diagram

**4.3.4.3.2. Yes-no games processing**

**Screen Design**

N/A

**Class Diagram**

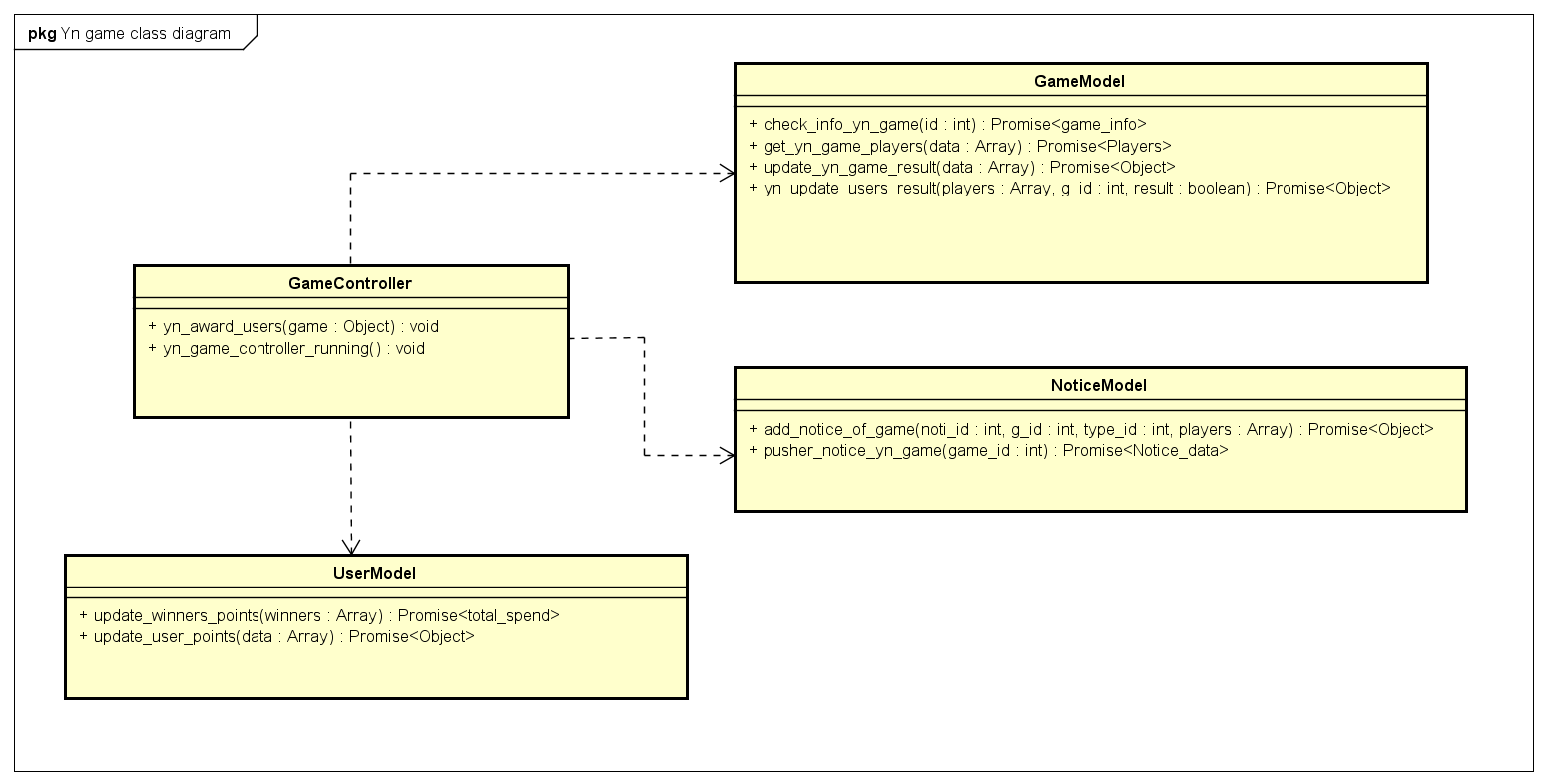
****

Figure 15: Yes-no game processing class diagram

**Class Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **GameController** | | | |
| **Physical address** | node-server/controller/gameController.js | | |
| **Base class** |  | | |
| **Attributes** | | | |
| N/A | | | |
| **Operations** | | | |
| yn\_award\_users | | | |
| **Return type** | void | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Game | Json object | Game data {game\_id, end\_date, owner\_id} |
| yn\_game\_controller\_running | | | |
| **Return type** | void | | |
| **Parameters** | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **UserModel** | | | |
| **Physical address** | node-server/models/userModel.js | | |
| **Base class** |  | | |
| **Attributes** | | | |
| N/A | | | |
| **Operations** | | | |
| update\_winners\_points | | | |
| **Return type** | Promise<total\_spend> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Winners | Array | Winners who gets points |
| update\_user\_points | | | |
| **Return type** | Promise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Data | Array | Data include points and id of user  form of data: [points, id] |

|  |  |  |  |
| --- | --- | --- | --- |
| **GameModel** | | | |
| **Physical address** | node-server/models/gameModel.js | | |
| **Base class** |  | | |
| **Attributes** | | | |
| N/A | | | |
| **Operations** | | | |
| check\_info\_yn\_game | | | |
| **Return type** | Promise<game\_info> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Id | Int | Id of game |
| get\_yn\_game\_players | | | |
| **Return type** | Promise<Players> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Data | Array | Form of data: [end\_date, game\_id]  (End\_date: time when game ends) |
| update\_yn\_game\_result | | | |
| **Return type** | Promise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Data | Array | Form of data:  [result, game\_id] |
| **yn\_update\_users\_result** | | | |
| **Return type** | Promise<Object> |  |  |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Players | Array |  |
|  | Game\_id | Int |  |
|  | Result | Bool | result states that user is winner or looser |

|  |  |  |  |
| --- | --- | --- | --- |
| **NoticeModel** | | | |
| **Physical address** | node-server/models/noticeModel.js | | |
| **Base class** |  | | |
| **Attributes** | | | |
| N/A | | | |
| **Operations** | | | |
| add\_notice\_of\_game | | | |
| **Return type** | Promise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Noti\_id | Int | Id of notice that will be add |
|  | G\_id | Int |  |
|  | Type\_id | Int | Type of notice |
|  | Players | Array |  |
| pusher\_notice\_yn\_game | | | |
| **Return type** | Promise<notice\_data> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Game\_id | int | Game\_id for select notice |

**Sequence Diagram**

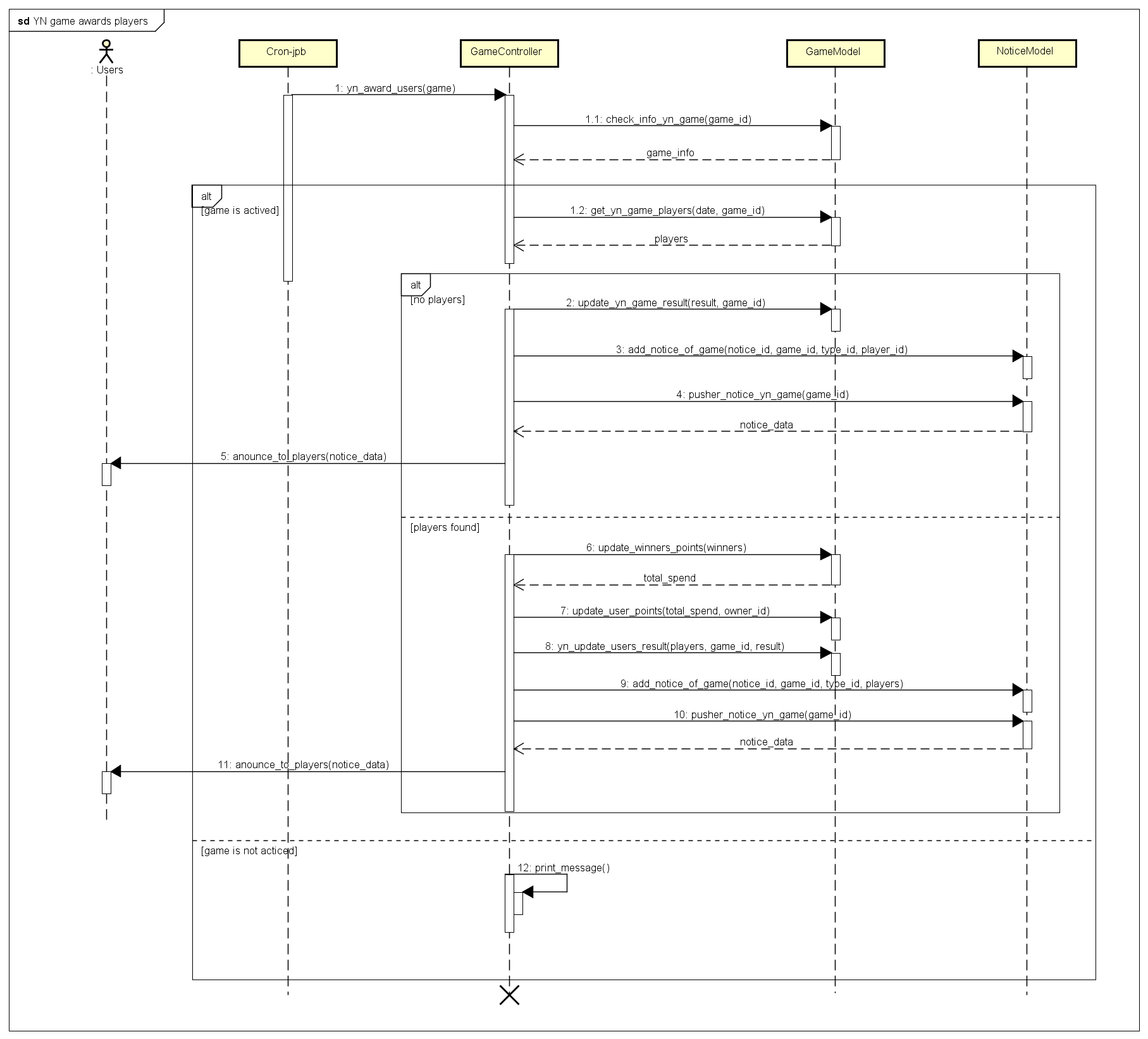
****

Figure 16: Yes-no game processing sequence diagram

**4.3.4.3.3. Multichoice game processing**

**Screen Design**

N/A

**Class Diagram**

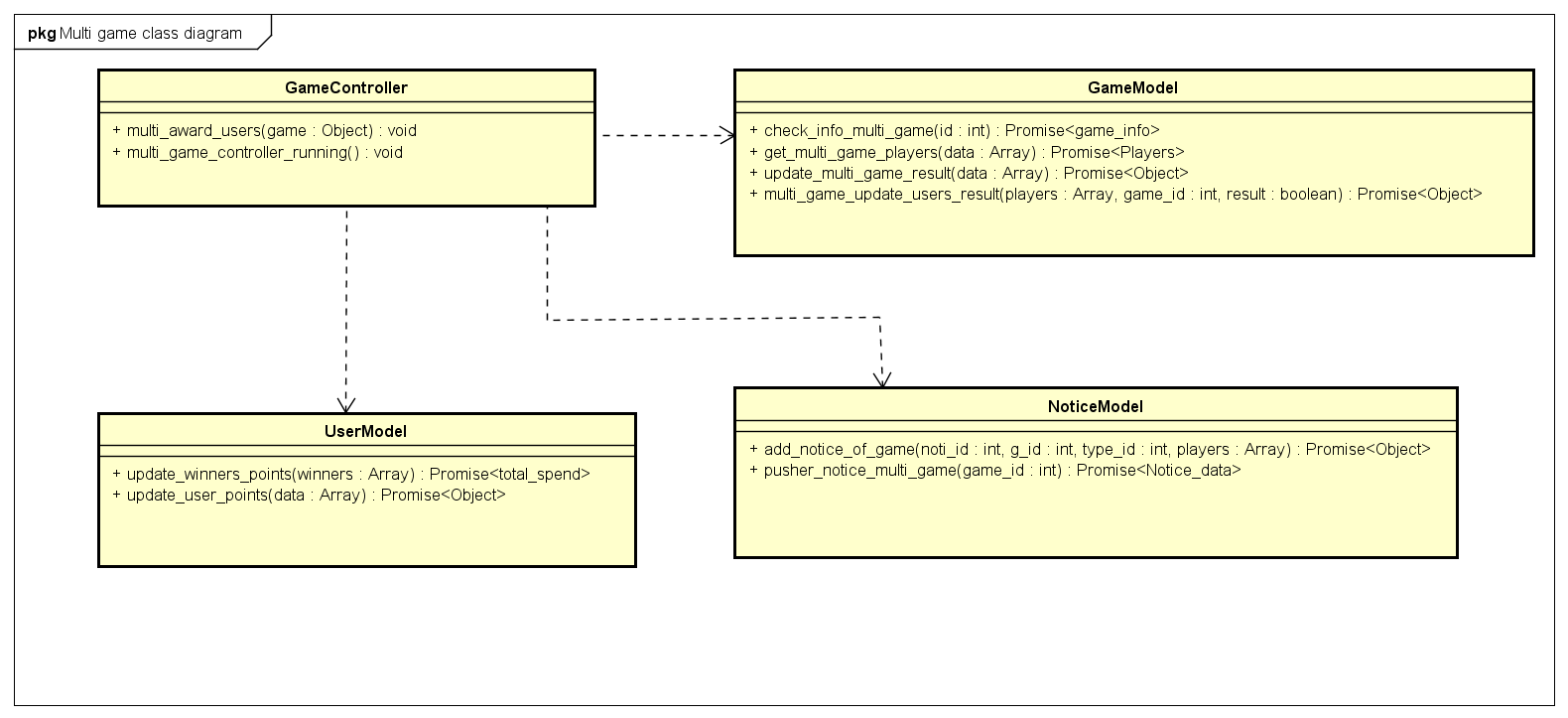
****

Figure 17: Multichoie game processing class diagram

**Class Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **GameController** | | | |
| **Physical address** | node-server/controller/gameController.js | | |
| **Base class** |  | | |
| **Attributes** | | | |
| N/A | | | |
| **Operations** | | | |
| multi\_award\_users | | | |
| **Return type** | void | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Game | Json object | Game data {game\_id, end\_date, owner\_id} |
| multi\_game\_controller\_running | | | |
| **Return type** | void | | |
| **Parameters** | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **UserModel** | | | |
| **Physical address** | node-server/models/userModel.js | | |
| **Base class** |  | | |
| **Attributes** | | | |
| N/A | | | |
| **Operations** | | | |
| update\_winners\_points | | | |
| **Return type** | Promise<total\_spend> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Winners | Array | Winners who gets points |
| update\_user\_points | | | |
| **Return type** | Promise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Data | Array | Data include points and id of user  form of data: [points, id] |

|  |  |  |  |
| --- | --- | --- | --- |
| **GameModel** | | | |
| **Physical address** | node-server/models/gameModel.js | | |
| **Base class** |  | | |
| **Attributes** | | | |
| N/A | | | |
| **Operations** | | | |
| **check\_info\_multi\_game** | | | |
| **Return type** | Promise<game\_info> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Id | Int | Id of game |
| **get\_multi\_game\_players** | | | |
| **Return type** | Promise<Players> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Data | Array | Form of data: [end\_date, game\_id]  (End\_date: time when game ends) |
| **update\_multi\_game\_result** | | | |
| **Return type** | Promise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Data | Array | Form of data:  [result, game\_id] |
| **multi\_game\_update\_users\_result** | | | |
| **Return type** | Promise<Object> |  |  |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Players | Array |  |
|  | Game\_id | Int |  |
|  | Result | Bool | result states that user is winner or looser |

|  |  |  |  |
| --- | --- | --- | --- |
| **NoticeModel** | | | |
| **Physical address** | node-server/models/noticeModel.js | | |
| **Base class** |  | | |
| **Attributes** | | | |
| N/A | | | |
| **Operations** | | | |
| **add\_notice\_of\_game** | | | |
| **Return type** | Promise<Object> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Noti\_id | Int | Id of notice that will be add |
|  | G\_id | Int | Game id |
|  | Type\_id | Int | Type of notice |
|  | Players | Array |  |
| **pusher\_notice\_multi\_game** | | | |
| **Return type** | Promise<notice\_data> | | |
| **Parameters** | **Name** | **Type** | **Description** |
|  | Game\_id | int | Game\_id for select notice |

**Sequence Diagram**

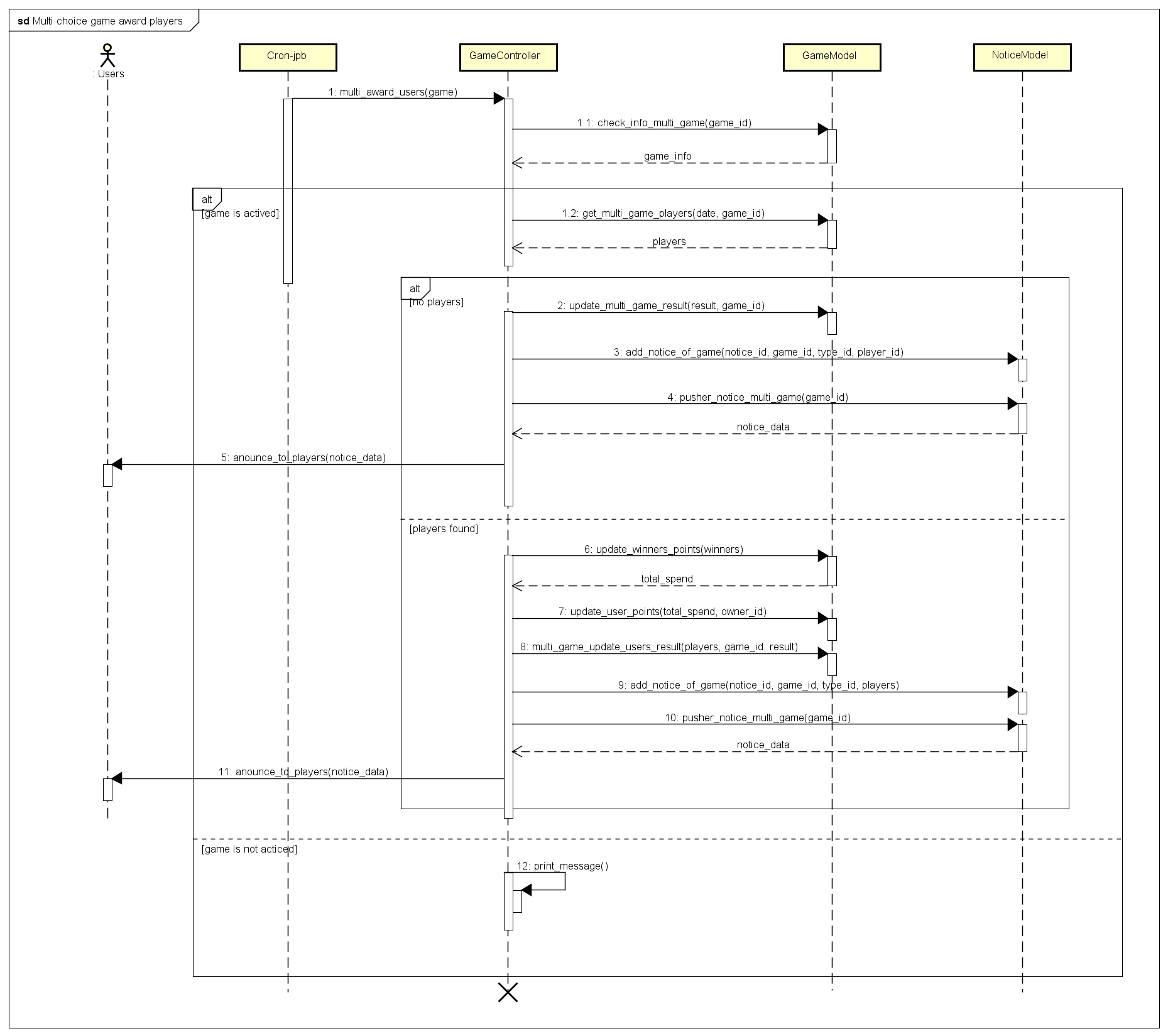
****

Figure 18: Multichoice game processing sequence diagram