Design Document for Gamification for learning project

Group#26

Tukpetov Raiymbet: %marks

Ferdauissiuly Aibek: %marks

Sauyrkulov Serikadil: %marks

Akzhol Bakytzhan: % marks

Dosymbetov Alimkhan: % marks

Version	Date	Author	Change
0.1	21/03/15		Initial Document
0.2	03/04/15		Updated with Interface design

Table of Contents

1 Introduction

<Make sure you follow guidelines in the ASSIGNMENT – as well as in this document>

<DO NOT DELETE ANY SECTION. INSTEAD, if you have nothing to write in any section, then write [NONE] or NA (not applicable) >

<< Please delete ALL instructor comments before submission>>

1.1 PURPOSE

The purpose of this document is to explain the design and architecture of the Gamification for learning freshman

1.2 SCOPE

This document includes system decomposition, interfaces, dependencies and design architecture

1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS

Term	Description

1.4 DESIGN GOALS

<Put in order of your team's priorities. Also, explain what each item means and be very specific about each item you put in here.>

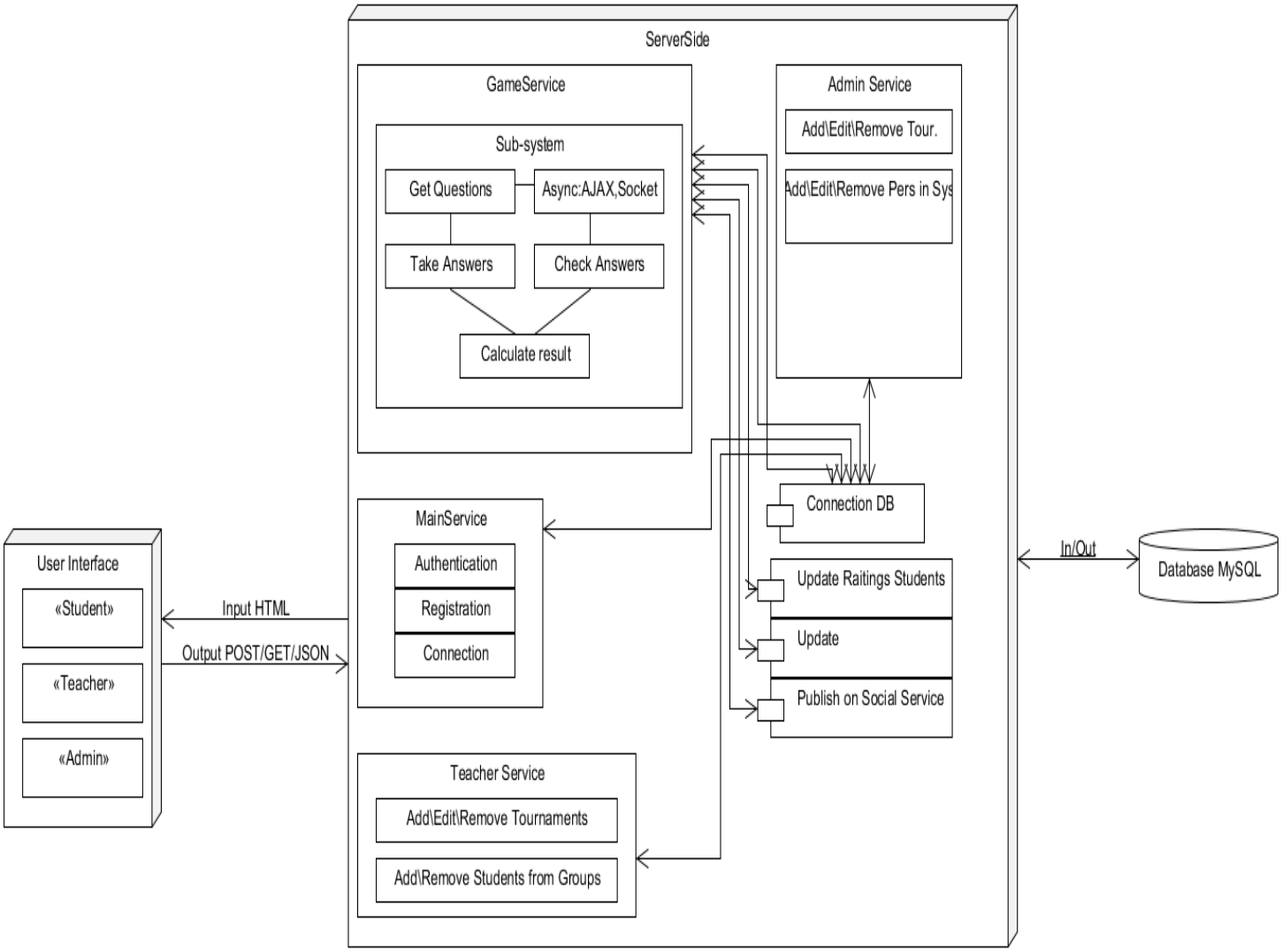
1. Reliability
2. Maintainability
3. Extensibility
4. Response Time
5. …..
6. ….
7. …..

2 References

https://code.google.com/p/gamification-sdu-en3-04/source/browse/

3 Decomposition Description

3.1 MODULE DECOMPOSITION



Architectural Diagram for Gamification for learning freshman system

Our system has three main architecture systems such as server, client and database. The client part has three categories of users such as administrator, teacher and student. Depending on who is using system, they are given a variety services. Therefore, the architecture of the server is the more important aspect of architectural design. Architecture of the server system consists game service, admin service, main

service, teacher service and database service modules. Third main system is MySQL database which all data is stored.

3.1.1.1 Game Service

This module consists of all game functions which concerning game services of system. Like give answer, check answer etc. In this module has play page of system. All game services performed on this page.

3.1.1.2 Admin Service Description

This module consists of all services for admin. Such as delete, update user etc.

3.1.1.3 Main Service

This module includes main simple operations which performed on main page. Main operations is login, registration, logout, send message to admin, etc.

3.1.1.4 Teacher Service

This module includes all service which doing teacher. To this service include create tournament, add questions, delete questions, etc.

3.1.1.5 Database Services

This module includes SQL transactions, database operations. Like connection, select, insert, update, etc. The module is connectable service that integrates all other modules with the database.

3.1.1.6 Database

Database MySQL is used to store all the data. Specifically, all user data information, and all events creates with user will be stored here. It will be interact with database services which performed all transactions.

3.2 CONCURRENT PROCESS

<Give an overall description of the different processes, threads, instantiations>

3.2.1.1 Game Services Description

This services performed on student page when student click button play and playing tournament. For each student creates a new one thread that provides all game services for student. Thread includes following functions:

Get questions of tournament: When student select and click play tournament, this function allow take questions of tournament on the server and show question on play page.

Give answer: This function allow to user give answer of one question and receive answer

Check answer: Allow check answer of user and show result correct, incorrect

Calculate result: Allow compute result score of user.

See common result: Allow to user show common result all played students in this tournament

Asynchronous Javascript and XML is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page.

Socket JavaScript: Create TCP connection to server, and keep it as long as needed. Server or client can easily close it. Client goes through HTTP compatible handshake process, if it succeeds, then server and client can exchange data both directions at any time. It is very efficient if application requires frequent data exchange in both ways. WebSockets do have data framing that includes masking for each message sent from client to server so data is simply encrypted.

3.2.1.2 Admin Services Description

The services that allow to admin do some actions. All actions performed on admin page.

Delete user: remove user on server when admin select user and perform operation delete.

Update user: update user data on server when admin select user and perform operation update.

Delete tournament: removed tournament on server when admin select tournament and perform operation delete tournament

Create tournament: create new tournament on server when admin perform operation create tournament. After admin shall fill form create tournament. And if admin clicked save tournament creates and saved on server, but if admin clicked cancel operation will be stopped.

Update tournament: update tournament on server when admin perform operation update tournament. After admin shall fill form update tournament. Admin can perform delete, add some question. After if admin click save changes shall be saved, but if admin click cancel changes shall not be saved and return back to admin page.

Add news: Admin on news page can add, delete, update news. If admin perform add news operation then shall fill form of add news and saved it and can do cancel.

Delete news: Admin can remove news on server. When click delete new, this new shall deleted on server and shall not show on news of main page.

Update news: When admin click update new, user can change information, data of new and da save or cancel.

3.2.1.3 Main Services Description

Main service include operations which can do all user.

Registration: Registration performed on main page of system. When user click create new account then user shall fill form registration and can do save or cancel. If user click save then system shall firstly check user to exists on server and create new user and give access to system.

Authorization: Authorization performed on main page of system. When user click sign in, then shall fill form of sign in, if clicked sign in, then system checking email and password on server. If all correctly then system shall give access to user.

Contact us: When user fill form contact us and click send then system send message to admin. And on admin page system shows information about it.

3.2.1.4 Database Services Description

On this module performed transactions and connection to database. For each user request creates new thread.

Connection DB: allow to server connect with database\

Update rating of students: automatically updated common result of students after each changes result of student

Update, Select, Insert: allow to server perform transactions faster, use clone of this functions

3.2.1.5 Teacher Services Description

On this module all services performed on teacher page and only use teacher.

Add/Edit/Remove tournaments: allow to user manage with tournament. And can perform operations add, edit, remove tournaments only yourself.

Add/Remove students from group: allow to user add and remove students on yourself groups. If added on group system give access to student to tournaments of user. If removed on group system give not access to student to tournaments of user.

3.3 DATA DECOMPOSITION

3.3.1.1 User-teacher

```
//Class for create each teacher with functions PHP code
Class Teacher {
        var $id;
        var $name;
        var $surname;
        var $password;
        var $birthday;
        var $gender;
        var $email;
        var $photo;
        var $telephone;
        function
}
```

3.3.1.2 User-student

```
Class Student{
}
```

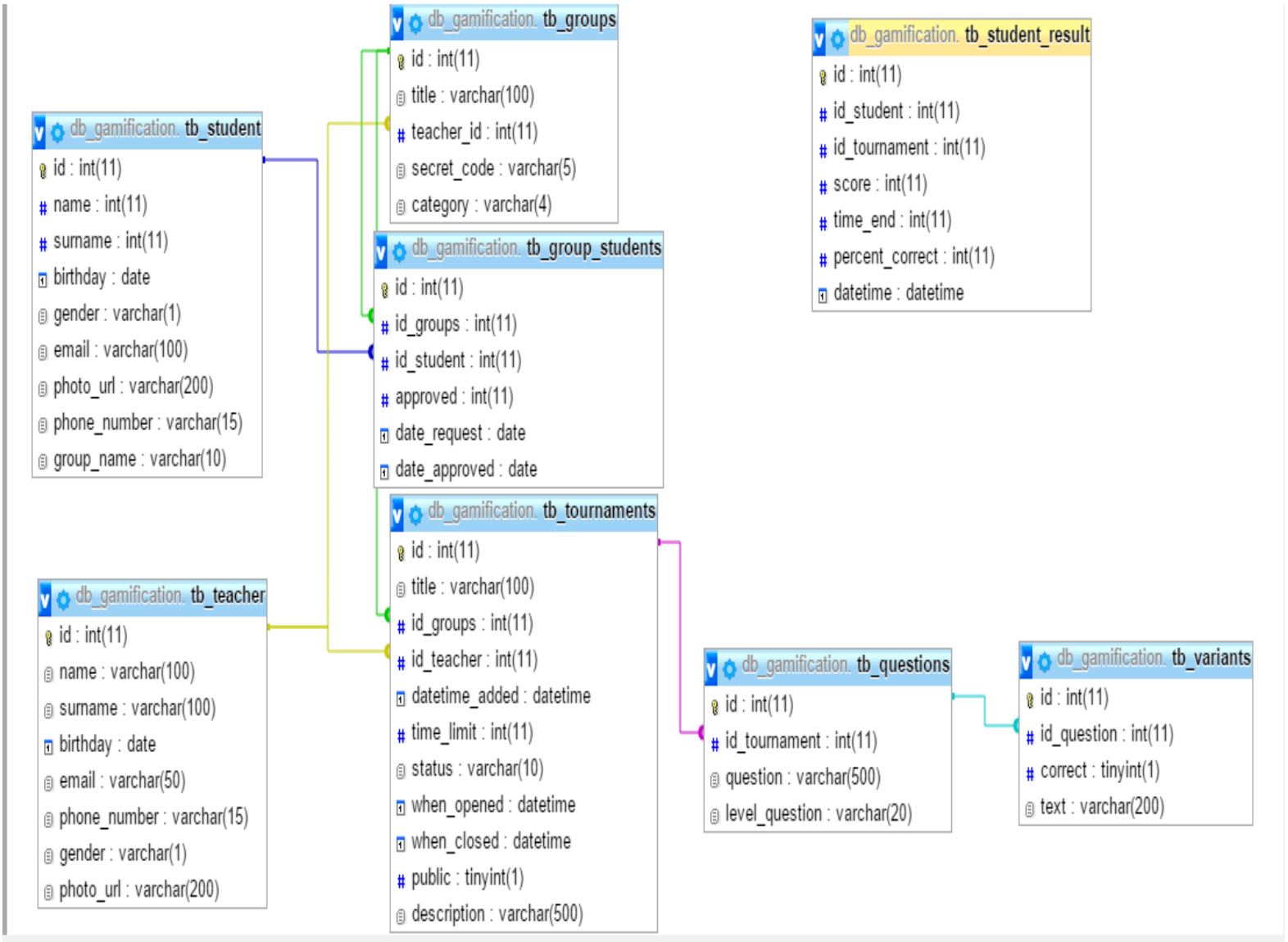
3.3.1.3 User-admin

```
Class Admin {
}
```

3.3.1.4 User-common

```
Class user_common {
}
```

3.3.1.5 Database tables



3.4 STATES

3.4.1.1 <State/System 1 > Description

3.4.1.2 <State/System 2> Description

4 Dependency Description

4.1 INTERMODULE DEPENDENCIES

4.2 INTERPROCESS DEPENDENCIES

4.3 DATA DEPENDENCIES

5 Interface Description

5.1 MODULE INTERFACE

5.1.1.1 <Main Service> Interface

Main Service sub-system provides service registration, authorization and connection with server and it is interact with User Interface and Database modules. There are we use PHP code file for each service.

On User Interface we give data with $_POST

On Registration:

5.1.1.1.1$user , $name, $surname, $birthday, $gender, $email, $tel, $group, $password =passed with $_POST

5.1.1.1.2Received password we encrypt twice with MD5

5.1.1.1.3For insert to database we use our connection php and mysqli_query;

5.1.1.1.4Returned result is if has not error return message such as “Registration successfully” else return array of errors. Return result is passed through AJAX,JSON.

On Authorization:

5.1.1.1.5$password, $e_mail=email and password of user; passed through $_POST

5.1.1.1.6Then we do checking user on database uses stmt prepare through email of user

5.1.1.1.7Return: message through AJAX, JSON. Result will be “Авторизация прошла успешно” or “Неправильный логин или пароль”

User connected with Server through WebSocket

5.1.1.2 Database Interface

There are we have connection PHP code which provide connect server with Database. And have functions INSERT, SELECT, UPDATE

5.1.1.2.1Connection php

Parameters: $url, $user, $password, $DB

$con = mysqli_connect($url, $user, $password, $DB) or die("Connection error");

Return: $con;

5.1.1.2.2 **function insert_result_student($array[$id_student, $id_tour, $score,$time_end, $percent])**

Parameters: $id_student – id of student who play tournament

$id_tour – id tournament which tour student play

$score – result of student point

$time_end – time second when student is finished this tournament

$percent – percentage of correct answers

Return: message about action result.

5.1.1.2.3 **function update_common_scorestudent($id_student, $score)**

Parameters: $id_student – id of student

$score – student result of tournament

$common_score = ($sum_scores_student)/($ sum_score_tournament)

Return: none;

5.1.1.2.4 **function update_ratings()**

Updated rating of students. This sorted by common scores of students

Parameter: none;

Return: none;

5.1.1.3 Game Service Interface

This interface interacts with User Interface and Database. Interface provides in the main functions of games. Functions will be on php and javascript codes

5.1.1.3.1 **function getQuestion** (Integer :: id_question)

- Integer :: id_question - ID number issue
- The function returns a value as struktirovannogo (JSON): Integer :: id_question - ID number issue, String :: title - question, Integer :: id_tour - ID number of the tour, String [] :: variants - answers to the question, Boolean [] :: the correct answer (True - correct, False - incorrect)

5.1.1.3.2 **function calculateResult** (Integer :: id_question, Long :: time_expired, Integer :: id_student, Integer :: id_tour, Integet [] :: list_question, Boolean [] :: correct_array)

- Integer :: id_question - ID number issue
- Long :: time_expired - the amount of time spent during the game
- Integer :: id_student - student ID number
- Integer :: id_tour - ID number of the tour
- Integer [] :: list_question - an array with a value of integer, in all questions to ask a question identifiers
- Boolean [] :: correct_array - array c value boolean, within the record user responses

5.1.1.3.3 **function updateStatusUser** (Integer :: id_socket, Socket :: socket, String [] data)

- Integer :: id_student - student ID number
- Socket :: socket - established connection
- String [] data - struktirovanny array data.
- During the game with two players the server will request during a game player status, answers to questions. On create a new event, the server is obliged to inform the new event to another member of the tournament

5.1.1.3.4 **function startGame** (Integer :: id_tour, Integer :: id_student)

- Integer :: id_question - ID number issue
- Long :: time_expired - the amount of time spent during the game
- Integer :: id_student - student ID number
- Integer :: id_tour - ID number of the tour
- Integer [] :: list_question - an array with a value of integer, in all questions to ask a question identifiers

Creates an entry in the database Table :: student_result and adds value id_student, id_tour, time_expired, start_time, _expired_time and changes the value of the status on the game began. At the end of the tournament in the record appends earn point and change the status of the tournament passed. Returns a reference to the questions and gives access to them.

JAVASCRIPT code

5.1.1.3.5 **function ready_question_field** (String json)

String :: json - structured array data (1.1.1)

Cleans temporary variables and build a new window for the question.

5.1.1.3.6 **function send_answer** (Integer :: id_question)

- Integer :: id_question - ID number issue
- The function checks the answers to the questions and to make a note of the time the memory response. Asynchronous makes a request to the server to get the next issue (1.1.1). The response from the server to function transshipped training field for the question (1.2.1).

5.1.1.4 Teacher Services Interface

This module interacts with Database and User Interface modules.

5.1.1.4.1 **function addStudentToGroup** (Integer :: id_student, Integer :: id_teacher, Integer :: id_group, Integer :: status)

- Integer :: id_student - student ID number
- Integer :: id_teacher - ID number of the teacher
- Integer :: id_group - group ID number
- Integer :: status - status. The student is accepted into the group (1) or not prinmal group (-1) or still waiting for the decision (0)
- This function changes the current row in the table :: tb_groups_student.
- Returns the String message
  - 1 - OK
  - 2 - Fail (Message with clarifications)

5.1.1.4.2 **function createGroup** (Integer :: id_teacher, String :: title, String :: category)

- Integer :: id_teacher - ID number of the teacher
- String :: title - the name of the group
- String :: category - the category group
- Function contribute to this database by a new group and genereuet UNIQUE secret code for entry into the group of students.
- Returns the String message
  - 1 - OK
  - 2 - Fail (Message with clarifications)

5.1.1.4.3 **function deleteGroup** (Integer :: id_teacher, Integer :: id_group, String :: message)

- Integer :: id_teacher - group ID number
- Integer :: id_group - group ID number
- String :: message - a message about the reason for the removal of

• The record is then removed from the group database. All students registrirovavshis a leaving group will be notified via email.

5.1.1.4.4 **function createTournaments** (Integer :: id_teacher, integer :: id_group, Integer :: title,)

Creates new tournament.

5.1.1.5 Admin Services Interface

This interface interacts with User Interface and Databases. This service can use only user admin and he get direct access to database.

5.1.1.5.1**function delete_user($user_type, $user_email)**

Removes user and all data on batabase;

5.1.1.5.2**function add_user($user_type, $array[])**

Parameters:$user_type – student or teacher

$array – consists information about user

Return: string – message about action successfully or unsuccessfully

5.1.1.5.3**function edit_user($user_type, $user_id, $array[])**

Changes information of user

Return: string – message about action successfully or unsuccessfully

5.1.1.5.4**function add_tournament()**

Create new tournament

5.1.1.5.5**function delete_tournament()**

Remove existing tournaments

5.1.1.5.6**function edit_tournament()**

Change tournament information or data

5.2 PROCESS INTERFACE

5.2.1.1 <Game Service> Interface

5.2.1.2 <Main Service> Interface

5.2.1.3 <Admin Service> Interface

5.2.1.4 <Teacher Service> Interface

5.2.1.5 <Database Service> Interface

6 Detailed Design

NOT REQUIRED

7 Design Rationale

7.1 DESIGN ISSUES

7.2 <ISSUE 1>

7.2.1.1 Description

7.2.1.2 Factors affecting Issue

7.2.1.3 Alternatives and their pros and cons

7.2.1.4 Resolution of Issue

7.3 <ISSUE 1>

7.3.1.1 Description

7.3.1.2 Factors affecting Issue

7.3.1.3 Alternatives and their pros and cons

7.3.1.4 Resolution of Issue

8 Traceability

No	Use Case/ Non-functional Description	Subsystem/Module/classes that handles it
1		
2		

FEEL FREE TO ADD APPENDICES AS NEEDED. UPDATE TOC BEFORE SUBMITTING