**Product Design**

**Specification**

**for**

***Rocket Math***

Version 1.0

February 18, 2015

**Prepared by:** **Samuel Arseneault, Kelly Markaity, John Luu**

Table of Contents

[**1 Introduction** 1](#_Toc413451051)

[**1.1 Purpose of the product Design Specification** 1](#_Toc413451052)

[**1.2 Scope** 1](#_Toc413451053)

[**1.3 Definitions and Abbreviations** 1](#_Toc413451054)

[**2 General Overview and Design Guidelines/Approach** 2](#_Toc413451055)

[**2.1 Assumptions/Constraints/Standards** 2](#_Toc413451056)

[**2.1.1 Assumptions** 2](#_Toc413451057)

[**2.1.2 Business Constraints** 3](#_Toc413451058)

[**2.1.3 Technical Constraints** 3](#_Toc413451059)

[**3 Architecture Design** 3](#_Toc413451060)

[**3.1 Logical View** 3](#_Toc413451061)

[**3.2 Hardware Architecture** 3](#_Toc413451062)

[**3.3 Software Architecture** 4](#_Toc413451063)

[**3.3.1 iOS Application** 4](#_Toc413451064)

[**3.3.2 Web Portal** 5](#_Toc413451065)

[**3.3.3 iOS Application Sending and Receiving Data** 6](#_Toc413451066)

[**3.3.4 Web Portal Class Sending and Receiving Data** 7](#_Toc413451067)

[**3.4 Security Architecture** 7](#_Toc413451068)

[**3.4.1 SSL Encryption and Identification** 7](#_Toc413451069)

[**3.4.2 Storing Credentials** 8](#_Toc413451070)

[**3.5 Communication Architecture** 8](#_Toc413451071)

[**3.6 Performance Architecture** 8](#_Toc413451072)

[**4 System Design** 8](#_Toc413451073)

[**4.1 Use-Case** 8](#_Toc413451074)

[**4.1.1 Actors** 8](#_Toc413451075)

[**4.1.2 List of Researcher Use Cases** 8](#_Toc413451076)

[**4.1.3 Use Case Diagram** 9](#_Toc413451077)

[9](#_Toc413451078)

[**4.1.4 Use Cases** 11](#_Toc413451079)

[**4.2 Sequence Diagrams** 14](#_Toc413451080)

[**4.3 Data Flow Diagram** 15](#_Toc413451081)

[**4.3.1 Teacher Role Data Flow Diagram** 15](#_Toc413451082)

[**4.4 Database Design** 15](#_Toc413451083)

[**4.5 Class Diagram** 15](#_Toc413451084)

[**4.6 Application Program Interface** 15](#_Toc413451085)

[**4.7 User Interface Design** 15](#_Toc413451086)

[**5. Product Design Specification Approval** 16](#_Toc413451087)

# **1 Introduction**

## **1.1 Purpose of the product Design Specification**

The Product Design Specification document documents and tracks the necessary information required to effectively define architecture and system design in order to give the development team guidance on architecture of the system to be developed. The Product Design Specification document is created during the Planning Phase of the project. Its intended audience is the project manager, project team, and development team. Some portions of this document such as the user interface (UI) may on occasion be shared with the client/user, and other stakeholder whose input/approval into the UI is needed.

## **1.2 Scope**

This document is guideline for creating a first grade math learning iOS application and web view completely from scratch. The iOS application will be used by first grade students, which include features such as navigating through levels, and having the ability to select practice or test modes. The student will also be able to enter in answers to the solvable math questions. The web portal will be used by the teachers and administrators. The features available to the teacher will include creating a class, adding students, creating levels, and adding questions to these levels. They will also be able to view the students’ progress. The features available to the administrator will include deleting teacher accounts, clearing the entire system, and allowing or denying new teachers to create accounts. This document will contain all of the contracts between the system and student, teacher, or administrator interactions, as well as specific implementation details.

## **1.3 Definitions and Abbreviations**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **iOS** | **A mobile operating system developed by Apple Inc. which powers the iPad device.** |
| **iOS Application** | **The application that our team will be creating, which will run on iOS devices.** |
| **Web Portal** | **An internet application which allows a teacher to log to and access student records.** |
| **MB** | **MB stands for megabyte, which is a unit of computer memory or data storage.** |
| **MVC** | **Model View Controller - a software design patterned.** |
| **JSON** | **JavaScript Object Notation - a syntax for storing and exchanging data.** |
| **SSL** | **Secure Socket Layer** |
| **Student** | **User of the iOS application.** |
| **Teacher** | **User of the web portal.** |
| **Administrator** | **User of the web portal.** |
| **Xcode** | **An integrated development environment containing a suite of software development tools created by Apple Inc. Used to create the iOS application.** |
| **Objective C** | **An object-oriented programming language used within Xcode. Used to create the iOS application.** |
| **JavaScript** | **A programming language designed for client-side interactions.** |
| **D3.js** | **A JavaScript library that uses data to create graphical visuals.** |
| **PHP** | **A server-side scripting language designed for web development.** |

# **2 General Overview and Design Guidelines/Approach**

This section describes the principles and strategies to be used as guidelines when designing and implementing the system. It will provide a general overview of constraints involved with the system to provide a basic frame for the system.

## **2.1 Assumptions/Constraints/Standards**

### **2.1.1 Assumptions**

* Teachers who may have a limited technological skillset will use the web portal application of this system.
* First grade students who rely on patterns and visual cues will be using the iOS application of this system.
* Students of different learning levels will be using the iOS application of this system.

### **2.1.2 Business Constraints**

* Slow connection from the back end of the system to the web portal and iOS application due to the server being hosted at Wayne State instead of locally at the school where the teachers and first graders are at.
* Lack of knowledge of children learning behavior. There are many different scenarios which confuse children such as distracting visuals or the formatting of math questions.
* The iOS application must be connected to the internet so it can successfully send information and data to the database.
* There may not be single administrator dedicated to maintaining this system.

### **2.1.3 Technical Constraints**

* The application used by the students must be functional on the iPad Air and the iPad 2 running software iOS 8.0
* The web portal used by the teachers and administrators must at least work on the Google Chrome web browser.
* The server will only be able to handle a small set of people. Even though this will be a public application

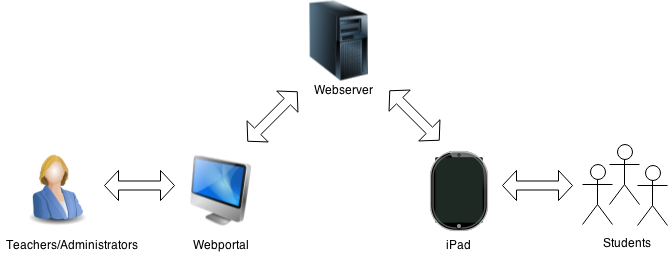
# **3 Architecture Design**

This section outlines the system and hardware architecture designs of system that is being built.

## **3.1 Logical View**

## **3.2 Hardware Architecture**

The Grayling Math system is setup on a public network with devices communicating wired and wirelessly with a web server.

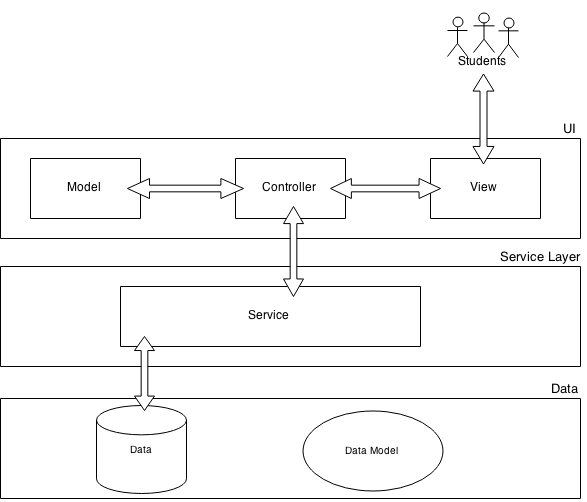


## **3.3 Software Architecture**

The Grayling Math system has two software architectures. One for the teachers’ and administrators’ web portal, and one for the students’ iOS application.

### **3.3.1 iOS Application**

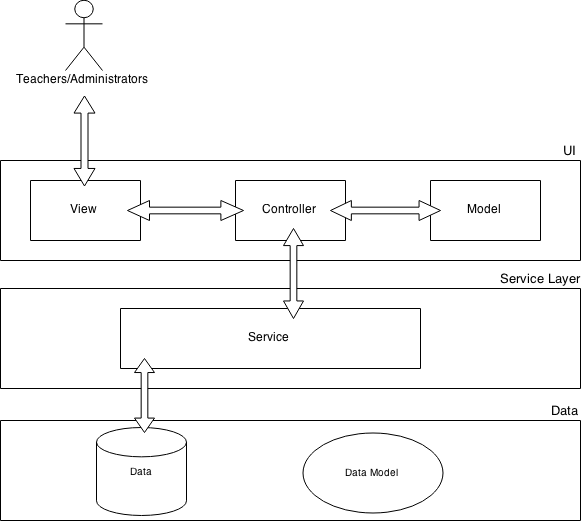
The student iOS application is build using Xcode and objective C, so by default it uses a MVC design pattern. The Controllers are responsible for communication between the model and the view layers of the system, as well as requesting data from the service layer. Model is responsible for storing data and providing methods for to work with the data. The view is responsible for the UI for the students.



### 

### **3.3.2 Web Portal**

The web portal will be following the same design MVC design pattern as the students’ iOS application. The Controllers are still responsible for communication between the model and the view layers of the system, as well as requesting data from the service layer. Model is responsible for storing data and providing methods for to work with the data. The view is responsible for the UI for the teachers.



### **3.3.3 iOS Application Sending and Receiving Data**

The iOS application will send and receive level, question, and student level progress information. This information will be sent to the web browser in the format of a JSON object.

The JSON objects follow the following format:

{"Table Name":[

{"Column 1":"Record 1 data", "Column 2":"Record 1 data"},

{"Column 1":"Record 2 data", "Column 2":"Record 2 data"},

{"Column 1":"Record 3 data", "Column 2":"Record 3 data"}

]}

The Grayling Math system will format the following tables in JSON format:

**Level Information:**

{"Level":[

{"level\_id":"0...n", "name":"ABC", "time\_limit":"min:sec",

       "description":"ABC","testq\_set":"T-F", "num\_tquestions":"0...n"}

]}

**Question Information:**

{"Question":[

{"question\_id":"0...n", "name":"ABC","level\_id":"0..n",

       "description":"ABC", "hint":"ABC", "operand1":"0..n"},

       "operand":"+-/\*"},"operand2":"0..n"}, "answer":"0..n"},

       "question\_type":"T-F"}

]}

**Progress Information:**

{"Progress":[

{"progress\_id":"0...n", "student\_id":"0..n","level\_id":"0..n",

       "question\_id":"0...n","status":"T-F"},"elapsed\_time":"min:sec"},

       "attempts":"0...n"}, "date":"MM/DD/YYYY"}

]}

### **3.3.4 Web Portal Class Sending and Receiving Data**

The web portal will also send and receive level, question, and student level progress information. However, the web portal will not be using JSON formatting, since querying the database will written in PHP, the web portal will eliminate the JSON formatting step and just store result sets in a string or an array.

## **3.4 Security Architecture**

### **3.4.1 SSL Encryption and Identification**

When sending information between the web portal and the server or the server and the iOS application, the Grayling Math application will follow an SSL encryption and identification process.

Below are steps describing the encryption and identification steps:

**Encryption:**

Step 1: Both devices agreeing on how to encrypt

Step 2: Server sends certificate

Step 3: User device requests the server to start encrypting

Step 4: The server requests the user device to start encrypting

Step 5: All messages are now encrypted

**Identification:**

Step 1: Company asks CA for a certificate

Step 2: CA creates certificate and signs it

Step 3: Certificate installed in server

Step 4: Browser issued with root certificates

Step 5: Browser trust correctly signed certs

### **3.4.2 Storing Credentials**

The administrator, teacher and student credentials will be stored into the database using salting and hashing on the passwords column. This will provide an extra layer of security along with the SSL encryption and identifications.

## **3.5 Communication Architecture**

## **3.6 Performance Architecture**

# 

# **4 System Design**

## **4.1 Use-Case**

### **4.1.1 Actors**

There are administrator, teacher, and student user types in the base of the Grayling Math system.

**Administrator** – the master user of the Grayling Math system. This user will be created by default and cannot be destroyed.

**Teachers** - any user who has been assigned the teacher role by creating a teacher account.

**Student** - any user who has been assigned the student role by a teacher creating this student user account.

### **4.1.2 List of Researcher Use Cases**

UseCaseID\_1 Teacher Creates a Class

UseCaseID\_2 Teacher Deletes a Class

UseCaseID\_3 Teacher Removes a Student

UseCaseID\_4 Teacher Adds a Student

UseCaseID\_5 Teacher Views Class Progress

UseCaseID\_6 Teacher Views Student Level Progress

UseCaseID\_7 Teacher Adds a Level to the Class

UseCaseID\_8 Teacher Removes a Level

UseCaseID\_9 Teacher Adds a Question to a Level

UseCaseID\_10 Teacher Removes a Question

UseCaseID\_11 Administrator Deletes a Teacher

UseCaseID\_12 Administrator Wipes the Database

UseCaseID\_13 Administrator Blocks New Teachers

UseCaseID\_14 Student Selects Practice

UseCaseID\_15 Student Selects Test

UseCaseID\_16 Student Enters Answer for Test

UseCaseID\_17 Student Enters Answers for Practice

### 

### **4.1.3 Use Case Diagram**

#### 4.1.3.1 Teacher Role Use Case Diagram



#### 4.1.3.2 Administrator Role Use Case Diagram

#### 

#### 4.1.3.3 Student Role Use Case Diagram



### 

### **4.1.4 Use Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UseCaseID\_1 | | | |
| **Use Case Name:** | Teacher Creates a Class | | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a first grade teacher, who after creating a new teacher account in the web portal, will be able to create a class for their students. The actor will be referred to as a teacher. | | |
| **Description:** | | This use case describes the creation of a class inside of the web portal. The teacher’s goal is to create a class to be able to add students, levels, and questions to it. | | |
| **Trigger:** | | The teacher logs into the web portal wishing to create a class, and selects the “Create class” button. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The teacher must have a valid session within the Grayling Math system. 3. The teacher must be using the Google Chrome web browser on a desktop/laptop. 4. The teacher must be connected to the internet. | | |
| **Postconditions:** | | Teacher successfully creates a class. | | |
| **Normal Flow:** | | 1. Teacher enters in the following class information:  * Teacher name * Class name * Class password  1. Teacher selects the “Create class” button 2. PHP sends query to the server to insert the new class information into the database. 3. JavaScript prompts with “Successfully created class message” | | |
| **Alternative Flows:**  **[Alternative Flow 1 – Class name already exists]** | | 2a. In step 2 of the normal flow, if the class name already exists   1. JavaScript informs the teacher that the class name has already been taken. 2. Teacher may return to step 1. | | |
| **Exceptions:** | | 2b. In step 2 of the normal flow, if a user enters invalid form data (e.g. missing class name, or incorrect password length)   1. JavaScript informs the teacher of the invalid input 2. Teacher may return to step1. | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UseCaseID\_2 | | | |
| **Use Case Name:** | Teacher Deletes a Class | | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a first grade teacher, who after creating a class in the web portal, will be able to delete that class. The actor will be referred to as a teacher. | | |
| **Description:** | | This use case describes the deletion of a class inside of the web portal. The teacher’s goal is to delete a previously created class. | | |
| **Trigger:** | | The teacher logs into the web portal wishing to delete a class, and selects the “Delete class” button. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The teacher must have previously created a class through the web portal. 3. The teacher must have a valid session within the Grayling Math system. 4. The teacher must be using the Google Chrome web browser on a desktop/laptop. 5. The teacher must be connected to the internet. | | |
| **Postconditions:** | | Teacher successfully deletes a class. | | |
| **Normal Flow:** | | 1. Teacher selects the “Delete class” button. 2. PHP sends query to the server to delete the class information from the database. | | |
| **Alternative Flows:** | |  | | |
| **Exceptions:** | | 1a. In step 1 of the normal flow, if the class has already been   1. JavaScript informs the teacher that the class has already been deleted. 2. Teacher is redirected to the teacher view menu. | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UseCaseID\_3 | | | |
| **Use Case Name:** | Teacher Removes a Student | | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a first grade teacher, who after creating a class and adding students in the web portal, will be able to remove a student from that class. The actor will be referred to as a teacher. | | |
| **Description:** | | This use case describes the removal of a student inside of the web portal. The teacher’s goal is to remove a previously created student. | | |
| **Trigger:** | | The teacher selects the “remove student” button. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The teacher must have previously created a class through the web portal. 3. The teacher must have previously added students to the class through the web portal. 4. Levels and questions have been created. 5. The teacher must have a valid session within the Grayling Math system. 6. The teacher must be using the Google Chrome web browser on a desktop/laptop. 7. The teacher must be connected to the internet. | | |
| **Postconditions:** | | Teacher successfully removes a student from a class. | | |
| **Normal Flow:** | | 1. The teacher selects the “Add/Remove student button” button. 2. JavaScript redirects teacher to the add remove student view. 3. The teacher selects a student to remove and selects the remove student button. 4. PHP sends query to the server to delete the student from the class. 5. JavaScript prompts user with a successful removal 6. The teacher selects the “OK” button 7. JavaScript redirects the user to the teacher view page | | |
| **Alternative Flows:** | |  | | |
| **Exceptions:** | | 4a. In step 4 of the normal flow, if the student has already been removed   1. JavaScript informs the teacher that the student has already been removed. 2. Teacher can resume with step 3 of the normal flow. | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UseCaseID\_4 | | | |
| **Use Case Name:** | Teacher Adds a Student | | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a first grade teacher, who after creating a class in the web portal, will be able to add a student to that class. The actor will be referred to as a teacher. | | |
| **Description:** | | This use case describes the addition of a student to a class inside of the web portal. The teacher’s goal is to add a student to a previously created class. | | |
| **Trigger:** | | The teacher selects the add student button. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The teacher must have previously created a class through the web portal. 3. Levels and questions have been created. 4. The teacher must have a valid session within the Grayling Math system. 5. The teacher must be using the Google Chrome web browser on a desktop/laptop. 6. The teacher must be connected to the internet. | | |
| **Postconditions:** | | Teacher successfully adds a student to a class. | | |
| **Normal Flow:** | | 1. The teacher selects the “Add/Remove student button” button. 2. JavaScript redirects teacher to the add remove student view. 3. The teacher enters the following student information into the fields:  * First Name * Last Name * User Name * Password  1. The teacher selects the add student button. 2. PHP sends query to the server to add the student from the class. 3. JavaScript prompts user with a success message 4. The teacher selects the “OK” button 5. JavaScript redirects the user to the teacher view page | | |
| **Alternative Flows:** | |  | | |
| **Exceptions:** | | 5a. In step 5 of the normal flow, if the student username has already been taken.   1. JavaScript informs the teacher that username has already been used. 2. The teacher selects the “OK” button. 3. The teacher is redirected to the teacher view menu. | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UseCaseID\_5 | | | |
| **Use Case Name:** | Teacher Views Class Progress | | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a first grade teacher, who after creating a class in the web portal with active students, will be able to view their students’ progress. | | |
| **Description:** | | This use case describes the ability for a teacher to view the progress of their class as a whole. | | |
| **Trigger:** | | The teacher enters the teacher view menu. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The teacher must have previously created a class through the web portal. 3. There must be active students assigned to the class. 4. The teacher must have a valid session within the Grayling Math system. 5. The teacher must be using the Google Chrome web browser on a desktop/laptop. 6. The teacher must be connected to the internet. | | |
| **Postconditions:** | | Teacher successfully views their class progress. | | |
| **Normal Flow:** | | 1. The teacher selects the teacher view tab. 2. JavaScript redirects teacher to the teacher view menu. 3. The controller queries the data model for the students and levels. 4. The controller passes the result to JavaScript. 5. JavaScript passes the result to the D3.js API. 6. D3.js generates a table to display each of the students, and the status of the levels. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – No students in the class]**  **[Alternative Flow 2 – No levels in the class]** | | 5a. In step 5 of the normal flow, if there are no students in the class   1. JavaScript displays that the class is empty. 2. Teacher may select the “add/remove” student button to fill the class.   5b. In step 5 of the normal flow, if there are no levels in the class   1. JavaScript displays that there are no levels. 2. Teacher may select the “add levels” button to fill the class. | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UseCaseID\_6 | | | |
| **Use Case Name:** | Teacher Views Student Level Progress | | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a first grade teacher, who after creating a class in the web portal with active students, will be able to view their individual student’s progress. | | |
| **Description:** | | This use case describes the ability for a teacher to view the progress of each individual student. | | |
| **Trigger:** | | The teacher enters the teacher view menu. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The teacher must have previously created a class through the web portal. 3. There must be active students assigned to the class. 4. The teacher must have a valid session within the Grayling Math system. 5. The teacher must be using the Google Chrome web browser on a desktop/laptop. 6. The teacher must be connected to the internet. | | |
| **Postconditions:** | | Teacher successfully views an individual student’s progress. | | |
| **Normal Flow:** | | 1. The teacher selects the teacher view tab. 2. JavaScript redirects teacher to the teacher view menu. 3. The controller queries the data model for the students and levels. 4. The controller passes the result to JavaScript. 5. JavaScript passes the result to the D3.js API. 6. D3.js generates a table to display each of the students, and the status of the levels. 7. Teacher selects a specific completed level next to the desired student’s name. 8. JavaScript redirects the teacher to the level progress view page. 9. The controller queries the data model for the student and level progress. 10. The controller passes the result to JavaScript 11. JavaScript passes the result to the D3.js API. 12. DS.js generates a double bar graph, which displays the time in minutes and the number of retries. As well as a bar graph for the practice, which displays just the number of retires. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – Only practice has been completed]**  **[Alternative Flow 1 – Only practice has been completed]** | | 12a. In step 12 of the normal flow, if the student has only completed the practice for this level.   1. JavaScript displays “the test has not been attempted” on the test side.   12b. In step 12 of the normal flow, if the student has only completed the test for this level.  1. JavaScript displays “the practice has not been attempted” on the practice side. | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UseCaseID\_7 | | | |
| **Use Case Name:** | Teacher Adds a Level to the Class | | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a first grade teacher, who after creating a class in the web portal with active students, will be able to add levels to this class. The actor will be referred to as a teacher. | | |
| **Description:** | | This use case describes the addition of a level to a class inside of the web portal. The teacher’s goal is to add a level to a previously created class. | | |
| **Trigger:** | | The teacher selects the add level button. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The teacher must have previously created a class through the web portal. 3. The teacher must have a valid session within the Grayling Math system. 4. The teacher must be using the Google Chrome web browser on a desktop/laptop. 5. The teacher must be connected to the internet. | | |
| **Postconditions:** | | Teacher successfully adds a level to a class. | | |
| **Normal Flow:** | | 1. The teacher selects the teacher view tab. 2. JavaScript redirects teacher to the teacher view menu. 3. The controller queries the data model for the students and levels. 4. The controller passes the result to JavaScript. 5. JavaScript passes the result to the D3.js API. 6. D3.js generates a table to display each of the students, and the status of the levels. 7. Teacher selects the add levels button. 8. JavaScript redirects the teacher to the add level menu. 9. Teacher fills in the level name, description, level time field information and selects the add level button. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – Max levels already added]** | | 7a. In step 5 of the normal flow, maximum of 20 levels are already added   1. JavaScript displays that the class is empty. 2. Teacher selects “OK” 3. JavaScript redirects the teacher to the teacher view page. | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |
| **Use Case ID:** | | UseCaseID\_7 | | |
| **Use Case Name:** | | Teacher Adds a Level to the Class | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a first grade teacher, who after creating a class in the web portal, will be able to add levels to this class. The actor will be referred to as a teacher. | | |
| **Description:** | | This use case describes the addition of a level to a class inside of the web portal. The teacher’s goal is to add a level to a previously created class. | | |
| **Trigger:** | | The teacher selects the add level button. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The teacher must have previously created a class through the web portal. 3. Students have not been added to the class yet. 4. The teacher must have a valid session within the Grayling Math system. 5. The teacher must be using the Google Chrome web browser on a desktop/laptop. 6. The teacher must be connected to the internet. | | |
| **Postconditions:** | | Teacher successfully adds a level to a class. | | |
| **Normal Flow:** | | 1. The teacher selects the teacher view tab. 2. JavaScript redirects teacher to the teacher view menu. 3. The controller queries the data model for the students and levels. 4. The controller passes the result to JavaScript. 5. JavaScript passes the result to the D3.js API. 6. D3.js generates a table to display each of the students, and the status of the levels. 7. Teacher selects the add levels button. 8. JavaScript redirects the teacher to the add level menu. 9. Teacher fills in the level name, description, level time field information and selects the add level button. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – Max levels already added]**  **[Alternative Flow 2 – Students have been added to the class]** | | 7a. In step 7 of the normal flow, maximum of 20 levels are already added  JavaScript displays that the class is empty.   1. Teacher selects “OK” 2. JavaScript redirects the teacher to the teacher view page.   7a. In step 7 of the normal flow, students have already been added to the class. Once there are students added to a class a teacher will NOT be able to modify level information, until the students are removed.   1. The add level button is grayed out and unavailable for the teacher to use. | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | | UseCaseID\_8 | | |
| **Use Case Name:** | | Teacher Removes a Level | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a first grade teacher, who after add levels to a class, will be able to remove levels from this class. The actor will be referred to as a teacher. | | |
| **Description:** | | This use case describes the removal of a level from a class inside of the web portal. The teacher’s goal is to remove a level from a previously created class. | | |
| **Trigger:** | | The teacher selects the remove level button. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The teacher must have previously created a class through the web portal. 3. The teacher must have previously created a level through the web portal. 4. Students have not been added to the class yet. 5. The teacher must have a valid session within the Grayling Math system. 6. The teacher must be using the Google Chrome web browser on a desktop/laptop. 7. The teacher must be connected to the internet. | | |
| **Postconditions:** | | Teacher successfully removes a level from a class. | | |
| **Normal Flow:** | | 1. The teacher selects the teacher view tab. 2. Teacher selects the desired level to remove. 3. JavaScript redirects the teacher to the edit level menu. 4. Teacher selects remove level. 5. The controller queries the data model to remove the level. 6. The data model returns success to the controller. 7. The controller requests JavaScript to display the success message 8. JavaScript displays success 9. Teacher selects the “OK” button 10. JavaScript redirects the teacher to the teacher view page. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – Min levels exist]**  **[Alternative Flow 2 – Students have been added to the class]** | | 9a. In step 9 of the normal flow, minimum of 15 levels exists in the database.   1. JavaScript displays that there is a minimum of 15 levels and is unable to delete these levels.   7a. In step 7 of the normal flow, students have already been added to the class. Once there are students added to a class a teacher will NOT be able to modify level information, until the students are removed.   1. The edit level button is grayed out and unavailable for the teacher to use. | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | | UseCaseID\_9 | | |
| **Use Case Name:** | | Teacher Adds a Question to a Level | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a first grade teacher, who after creating a class and level in the web portal, will be able add a question to that level. | | |
| **Description:** | | This use case describes the addition of a question to a level inside of the web portal. The teacher’s goal is to add a question to a previously created level. | | |
| **Trigger:** | | The teacher selects the add question button. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The teacher must have previously created a class through the web portal. 3. The teacher must have previously created a level through the web portal. 4. Students have not been added to the class yet. 5. The teacher must have a valid session within the Grayling Math system. 6. The teacher must be using the Google Chrome web browser on a desktop/laptop. 7. The teacher must be connected to the internet. | | |
| **Postconditions:** | | Teacher successfully adds a question to a level. | | |
| **Normal Flow:** | | 1. The teacher selects the teacher view tab. 2. Teacher selects the desired level to edit. 3. JavaScript redirects the teacher to the edit level menu. 4. The teacher selects the add new question button. 5. JavaScript redirects the teacher to the add new question menu. 6. The teacher fills in the question name, description, frequency, type, operators, operand, and the answer fields. 7. Teacher selects the add question button. 8. JavaScript requests the controller to add the question to the level. 9. The controller queries the data model to add the question to the level. 10. The data model returns success to the controller. 11. The controller requests JavaScript to display the success message 12. JavaScript displays success 13. Teacher selects the “OK” button 14. JavaScript redirects the teacher to the teacher view page. | | |
| **Alternative Flows:** | |  | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | | UseCaseID\_10 | | |
| **Use Case Name:** | | Teacher Removes a Question | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a first grade teacher, who after adding questions to a level, will be able to remove a question. | | |
| **Description:** | | This use case describes the removal of a question to a level inside of the web portal. The teacher’s goal is to remove a question from a previously created level. | | |
| **Trigger:** | | The teacher selects the remove question button. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The teacher must have previously created a class through the web portal. 3. The teacher must have previously created a level through the web portal. 4. The teacher must have previously added a question to a level. 5. Students have not been added to the class yet. 6. The teacher must have a valid session within the Grayling Math system. 7. The teacher must be using the Google Chrome web browser on a desktop/laptop. 8. The teacher must be connected to the internet. | | |
| **Postconditions:** | | Teacher successfully adds a question to a level. | | |
| **Normal Flow:** | | 1. The teacher selects the teacher view tab. 2. Teacher selects the desired level to edit. 3. JavaScript redirects the teacher to the edit level menu. 4. The teacher selects the desired question to remove. 5. The teacher selects the remove question button. 6. JavaScript requests the controller to remove the question from the level. 7. The controller queries the data model to remove the question from the level. 8. The data model returns success to the controller. 9. The controller requests JavaScript to display the success message 10. JavaScript displays success 11. Teacher selects the “OK” button 12. JavaScript redirects the teacher to the teacher view page. | | |
| **Alternative Flows:** | |  | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | | UseCaseID\_11 | | |
| **Use Case Name:** | | Administrator Deletes a Teacher | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a web portal administrator, who will be able to delete a teacher from the Grayling Math system. | | |
| **Description:** | | This use case describes the deletion of a teacher from the Grayling Math system. The administrator’s goal is to remove a teacher. | | |
| **Trigger:** | | The administrator selects the remove teacher button. | | |
| **Preconditions:** | | 1. The teacher must have previously created a teacher account through the web portal. 2. The administrator must have a valid session within the Grayling Math system. 3. The administrator must be using the Google Chrome web browser on a desktop/laptop. 4. The administrator must be connected to the internet. | | |
| **Postconditions:** | | Administrator successfully deletes a teacher. | | |
| **Normal Flow:** | | 1. The administrator selects the view teacher menu. 2. JavaScript redirects the administrator to the view teacher menu. 3. JavaScript requests the controller to return the teacher list. 4. The controller queries the data model for the teachers in the database. 5. The data model returns the results to the controller. 6. The controller returns the results to JavaScript. 7. JavaScript passes the result the D3.js API. 8. D3.js API creates a table which displays the list of teachers. 9. The administrator selects the desired teacher and selects delete. 10. The controller queries the data model to remove the selected teacher. 11. The data model returns success 12. The controller returns success to JavaScript 13. JavaScript displays success to the administrator | | |
| **Alternative Flows:** | |  | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | | UseCaseID\_12 | | |
| **Use Case Name:** | | Administrator Wipes the Database | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a web portal administrator, who will be able to wipe the entire Grayling Math Database. | | |
| **Description:** | | This use case describes the deletion of a teacher from the Grayling Math system. The administrator’s goal is to remove a teacher. | | |
| **Trigger:** | | The administrator selects the remove teacher button. | | |
| **Preconditions:** | | 1. The administrator must have a valid session within the Grayling Math system. 2. The administrator must be using the Google Chrome web browser on a desktop/laptop. 3. The administrator must be connected to the internet. | | |
| **Postconditions:** | | Administrator successfully deletes a teacher. | | |
| **Normal Flow:** | | 1. The administrator selects the wipe entire data base button. 2. JavaScript prompts with a confirmation message. 3. The administrator selects the confirm button. 4. JavaScript request the controller to wipe the database. 5. The controller queries the data model to clear all the tables besides the administrator table. 6. The data model returns success. 7. The controller returns success to JavaScript. 8. JavaScript prompts user with a success message. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – Database is empty]** | | 6a. In step 6 of the normal flow, database is already empty   1. The data model returns false to the controller. 2. The controller returns false to JavaScript. 3. JavaScript prompts user with a failure message. | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | | UseCaseID\_13 | | |
| **Use Case Name:** | | Administrator Blocks New Teachers | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a web portal administrator, who will be able to block new teachers from creating accounts. | | |
| **Description:** | | This use case describes the restriction of a new teacher to create an account on the web portal. | | |
| **Trigger:** | | The administrator selects the “block new teachers” option. | | |
| **Preconditions:** | | 1. The administrator must have a valid session within the Grayling Math system. 2. The administrator must be using the Google Chrome web browser on a desktop/laptop. 3. The administrator must be connected to the internet. | | |
| **Postconditions:** | | Administrator successfully blocks new teachers from registering an account. | | |
| **Normal Flow:** | | 1. The administrator selects the “block new teachers” option. 2. JavaScript requests the controller to refuse new teachers. 3. The controller requests the data model to refuse new teachers. 4. The data model sets the “refuseNewTeacher” variable to true. 5. The data model returns true to the controller. 6. The controller returns true to JavaScript. 7. JavaScript prompts the administrator with successfully set to true message. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – The new teachers are already blocked]** | | 4a. In step 4 of the normal flow, “refuseNewTeacher” variable is already set to true.   1. The data model set the “refuseNewTeacher” variable to false. 2. The data model returns false to the controller. 3. The controller returns false to JavaScript. 4. JavaScript prompts the administrator with successfully set to false message. | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The teacher understands English. 2. The teacher is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | | UseCaseID\_14 | | |
| **Use Case Name:** | | Student Selects Practice | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a student iOS application user, who will be able to select the practice level option. The actor will be referred to as the student. | | |
| **Description:** | | This use case describes the ability of a student to select the practice level option. | | |
| **Trigger:** | | The student selects the practice button. | | |
| **Preconditions:** | | 1. The teacher must have created a student account for this student inside of the web portal. 2. The student must have a valid session within the Grayling Math system. 3. The student must be using the iOS application on the iPad Air or the iPad 2 4. The student must be connected to the internet. | | |
| **Postconditions:** | | Student enters the practice mode. | | |
| **Normal Flow:** | | 1. The student selects the desired level. 2. The controller requests the view to switch to the level type screen 3. View switches to the level type screen 4. The student selects the practice button 5. The controller requests the view to switch to the practice mode screen. 6. The view switches to the practice mode screen | | |
| **Alternative Flows:** | |  | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The student understands English. 2. The student is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | | UseCaseID\_15 | | |
| **Use Case Name:** | | Student Selects Test | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a student iOS application user, who will be able to select the test level option. The actor will be referred to as the student. | | |
| **Description:** | | This use case describes the ability of a student to select the test level option. | | |
| **Trigger:** | | The student selects the test button. | | |
| **Preconditions:** | | 1. The teacher must have created a student account for this student inside of the web portal. 2. The student must have a valid session within the Grayling Math system. 3. The student must be using the iOS application on the iPad Air or the iPad 2 4. The student must be connected to the internet. | | |
| **Postconditions:** | | Student enters the test mode. | | |
| **Normal Flow:** | | 1. The student selects the desired level. 2. The controller requests the view to switch to the level type screen 3. View switches to the level type screen 4. The student selects the test button 5. The controller requests the view to switch to the test mode screen. 6. The view switches to the test mode screen | | |
| **Alternative Flows:** | |  | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The student understands English. 2. The student is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | | UseCaseID\_16 | | |
| **Use Case Name:** | | Student Enters Answers for Test | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a student iOS application user, who after selecting the level type, will be able to enter answers for the provided questions. | | |
| **Description:** | | This use case describes the ability of a student to enter answers for provided questions. | | |
| **Trigger:** | | The student selects the submit button. | | |
| **Preconditions:** | | 1. The teacher must have created a student account for this student inside of the web portal. 2. The student must have a valid session within the Grayling Math system. 3. The student must have previously selected the test mode. 4. The student must be using the iOS application on the iPad Air or the iPad 2 5. The student must be connected to the internet. | | |
| **Postconditions:** | | Student successfully enters answers into the fields. | | |
| **Normal Flow:** | | 1. The view prompts the student with a math question. 2. The student drags in tiles to into the answer field to formulate an answer. 3. The student selects submit. 4. The view updates the controller with the formulated answer information. 5. The controller validates if the answer is correct. 6. The controller stores the answer 7. The controller request the view to change to the next question. 8. The view switches to the next question | | |
| **Alternative Flows:**  **[Alternative Flow 1 – Student answers all the questions]** | | 7a. In step 7 of the normal flow, the student answers all the provided questions.   1. The controller requests the view to switch to the level overview screen. 2. The view switches to the level overview screen | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The student understands English. 2. The student is able to enter in form data into the provided fields. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | | UseCaseID\_17 | | |
| **Use Case Name:** | | Student Enters Answers for Practice | | |
| **Created By:** | Samuel Arseneault | | **Last Updated By:** |  |
| **Date Created:** | 3/27/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be a student iOS application user, who after selecting the level type, will be able to enter answers for the provided questions. | | |
| **Description:** | | This use case describes the ability of a student to enter answers for provided questions. | | |
| **Trigger:** | | The student selects the submit button. | | |
| **Preconditions:** | | 1. The teacher must have created a student account for this student inside of the web portal. 2. The student must have a valid session within the Grayling Math system. 3. The student must have previously selected the practice mode. 4. The student must be using the iOS application on the iPad Air or the iPad 2 5. The student must be connected to the internet. | | |
| **Postconditions:** | | Student successfully enters answers into the fields. | | |
| **Normal Flow:** | | 1. The view prompts the student with a math question. 2. The student drags in tiles to into the answer field to formulate an answer. 3. The student selects submit. 4. The view updates the controller with the formulated answer information. 5. The controller validates if the answer is correct. 6. The controller request the view to prompt with the correct answer message. 7. The view prompts the user with the correct answer message. 8. The student selects the next question button. 9. The controller request the view to change to the next question. 10. The view switches to the next question | | |
| **Alternative Flows:**  **[Alternative Flow 1 – Student answers with an incorrect answer]** | | 5a. In step 5 of the normal flow, the student answers with an incorrect answer   1. The controller requests the view the view to prompt with the incorrect answer screen. 2. The view prompts the user with the incorrect answer screen. 3. The student drags in the operands and the correct answer. 4. The student make return to step 3 of the normal flow. | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case may be used hundreds of times in a day, but usually on-demand. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The student understands English. 2. The student is able to enter in form data into the provided fields. | | |

## **4.2 Sequence Diagrams**

## 

## 

## 

## 

## **4.3 Data Flow Diagram**

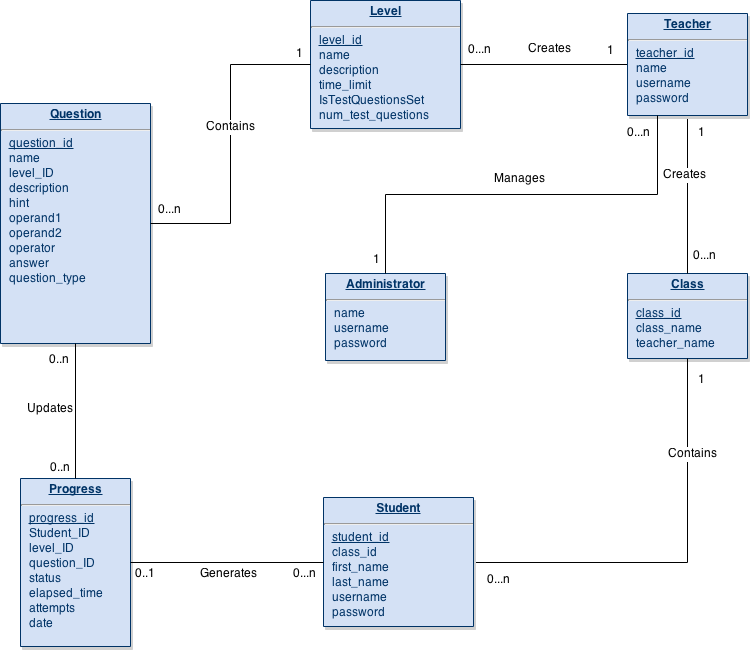
### **4.3.1 Teacher Role Data Flow Diagram**

## 

### **4.3.2 Administrator Role Data Flow Diagram**

### **4.3.3 Student Role Data Flow Diagram**

## **4.4 Database Design**

****

## **4.5 Class Diagram**

## **4.6 Application Program Interface**

The D3.js library will be used to create and display graphs on the. In the web portal of the Grayling Math system, the D3.js library will be used to convert the students’ test and practice information into bar graphs. It will also be used to create a table for the class level progress in the teacher view menu of the web portal. The queried data from the progress table will be fed into the D3.js to generate the graphs and tables.

The official API reference can be found at:

<http://d3js.org/>

The web portal will also be using the built-in APIs within PHP. PHP offers a built in JSON encoder and can be used by calling “json\_encode ($jsonObject);”

## **4.7 User Interface Design**

The User Interface will resize itself to fit within the browser window at 3 different pixel width breakpoints. These are based off of key breakpoints in the Twitter Bootstrap.

1. 768px and up
2. 992px and up
3. 1200px and up

At each breakpoint, the graph container div will resize to fill its parent div and JavaScript will trigger re-drawing the graph.

# **5. Product Design Specification Approval**

The undersigned acknowledge they have reviewed the Grayling Math Design Specification document and agree with the approach it presents. Any changes to this requirements definition will be coordinated with and approved by the undersigned or their designated representatives.

|  |  |  |  |
| --- | --- | --- | --- |
| Signature: |  | Date: |  |
| Print Name: |  |  |  |
| Title: |  |  |  |
| Role: |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Signature: |  | Date: |  |
| Print Name: |  |  |  |
| Title: |  |  |  |
| Role: |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Signature: |  | Date: |  |
| Print Name: |  |  |  |
| Title: |  |  |  |
| Role: |  |  |  |