CandyFactory: Cloud-Based Educational Game for Teaching Fractions

Tiancheng Ying

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

May 1, 2019

Blacksburg, Virginia

Keywords and phrases: Cloud software engineering, educational game, fractions, game-based learning, mathematics education

CandyFactory: Cloud-Based Educational Game for Teaching Fractions

Tiancheng Ying

ABSTRACT

Nowadays cross platform software development is more expensive than ever before in terms of time and effort. Meantime with increasing number of personal devices, it is harder for local applications to synchronize and connect to the Internet.

In terms of educational games, they can be divided into “local educational game” and “web educational game.” “Local game” indicates the ones either on tablets, mobile devices or PC, which is an application on the corresponding platform. This kind of game mostly does not have backend support nor cross platform features such as the iPad version of CandyFactory. For one specific game, if the developer wants it to run on iPad and Android tablets, they need to develop two applications based on corresponding development framework, which is time and effort consuming. “Web game” indicates the ones on websites, which support cross platforms, but do not have backend support. Usually they are pure JavaScript or flash games with no backend recording the performances and the achievements.

Software development for each individual platform is time and effort consuming. In order to achieve cross platform development, many programming languages and platforms like Java, Python, and JVM appear. Among all the cross platform approaches, cloud-based software development is the most universal solution to this problem. With web browsers built into every operating system, cloud software can be compatible with almost any device. Moreover, “Software-as-a-Service” (SaaS) is becoming a new software engineering paradigm and cloud-based software development is more popular because of its flexible scalability and cross platform features.

In this thesis, we create a cloud-based educational game, CandyFactory, based on an iPad version of CandyFactory, and add backend to it to record user performance as well as achievements. Firstly, we re-develop the whole game from the iOS platform to the cloud-based Java EE platform. Secondly, we add new features to improve the game play such as ruler functionality and achievements animation. Thirdly, we add backend support to CandyFactory, including user account creation, course creation and performance report generation. With this functionality, teachers can monitor their students’ performances and generate course reports. Moreover, teachers can view a specific student’s report in order to provide more specific and effective help to their students. Lastly, with the advantages of cloud-based software development, we can update the whole application at any time without forcing the user to reinstall the update or re-download the game. With the hot update, the cloud-based CandyFactory is highly maintainable.

The cloud-based CandyFactory runs on any computer that supports minimum 1024x768 screen resolution. The computer could be iPads, Android or Microsoft tablets, Windows or Mac laptops and desktops, and any other computer with a web browser. The advantages of cloud-based educational games over local educational games and web educational games are: firstly, they have cross platform features; secondly, they have backend data collection support; thirdly, they are consistent even if users log in with different computers, their game record and history will always be the same; lastly, the teacher can always keep track of his/her students' performance and provide more specific help and feedback.

CandyFactory: Cloud-Based Educational Game for Teaching Fractions

Tiancheng Ying

GENERAL AUDIENCE ABSTRACT

Providing services on the cloud has become universal. The term “Cloud-Based” indicates that the software application runs on a server computer and users access the application by using a web browser anywhere and anytime.

This thesis presents a cloud-based educational game called CandyFactory to teach fractions. The users can use CandyFactory under a web browser on an Internet-connected tablet, laptop, or desktop computer with minimum 1024x768 screen resolution. User’s game performance data is recorded on the server computer regardless of which tablet, laptop, or desktop computer the user uses to play the game.

Cloud-based CandyFactory has four kinds of users: Individual, Teacher, Student, Administrator. *Individual* users can play the game to learn fractions as well as generate performance reports. *Teachers* can create a course, automatically generate student accounts under a course, and generate performance reports for individual students or for the whole class. *Students* can play the game under the account provided by the teacher and view their performance reports. *Administrator* is a built-in account user for maintaining the cloud-based software application.

By developing the cloud-based CandyFactory educational game, we provide the users a cross-platform and cross-computers solution which helps the teachers and students learn fractions more efficiently and effectively.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Dr. Osman Balci for his professional and invaluable guidance. When I was having the hardest time, he gave me an opportunity to be funded and learn Cloud Software Development. Also his hardworking and attitude towards knowledge affects me a lot. When I was making decisions, it is him that analyzed the pros and cons for me from my perspective. He is also the one who tells me family has the top priority. It is my life time pleasure to have an advisor like him and to be his student, he is one of the most important person in my life. Without his help, I could not complete my study in Computer Science at Virginia Tech and could not be what I am now.

I am also very grateful to my committee members Dr. Anderson Norton and Dr. Denis Gracanin for their professional guidance in my thesis.

Lastly, I am deeply thankful to my wife Jing Luo and my parents Fengbo Ying and Yu Chen. It is my wife accompanied me through the darkest time in my life. It is my wife who gave up her graduate study and came to the U.S. to build our dreams. It is my wife who I make decision with. Along the way, it is my parents who did their best to support me studying abroad and give me backups. Thank you for your love, support, encouragement, understanding, and sacrifices. Your love and encouragement have been the most valuable thing in my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACROYNMS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CDI	Context-Dependency Injection
CSS	Cascading Style Sheets
DEG	Digital Educational Game
EJB	Enterprise Java Bean
EL	Expression Language
ERD	Entity Relationship Design
GB	Giga Bytes
HLA	High Level Architecture
HTML	HyperText Markup Language
IDE	Integrated Development Environment
iOS	Apple's mobile Operating System
Java EE	Java platform, Enterprise Edition
JDBC	Java Database Connectivity
JSF	JavaServer Faces
JSON	JavaScript Object Notation
JS	JavaScript
JQuery	JQuery JavaScript framework
OOD	Object Oriented Design
RAM	Random Access Memory
RDBMS	Relational Database Management System
SaaP	Software as a Product
SaaS	Software as a Service
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	Extensible Markup Language

Chapter 1: Problem Definition and Overview

1.1 Introduction

A *Digital Educational Game* (DEG) is a game created for the purpose of teaching a subject in the form of software that runs on a computer such as desktop, laptop, handheld, or game console [Aslan and Balci 2015]. Prensky [2007] indicates that DEG-based learning has four advantages: it fits better with today's and the future's generations of learners, it is fun and motivating, it is incredibly versatile and extremely effective.

CandyFactory Educational Game for iPad [Aslan, Norton, and Balci 2019; LTRG 2019] is an educational game that supports more powerful conceptions of fractions. Also as a *Balefire Labs TopRated* [Balefire Labs] application, CandyFactory leverages the actions of partitioning and iterating candy bars to teach students different fractions intuitively with interactive operations and animations. However, the CandyFactory iPad version can only be used on iPads. Those schools that do not have iPads are left out of using the CandyFactory iPad version.

"Software-as-a-Service" (SaaS) has become the software engineering paradigm of choice and cloud-based software development is more popular because of its flexible scalability and cross platform features. In the age of Internet, more software-based solutions are expected to be available in the cloud, which enables the use of the software under a web browser on any Internet-connected mobile device, laptop, or desktop computer.

In order to take advantage of both the CandyFactory iPad version and the features of cloud software, in this thesis we design, architect and implement a Cloud-based CandyFactory. For the CandyFactory game, we fully reproduce all the functionalities and features. Additionally, we redesign several interaction techniques as well as add new features to enhance the game. For the cloud end, we develop a robust system supporting account creation, class creation, student creation, performance recording and performance report generation. With the features of cloud software, Cloud-based CandyFactory can be played on any browser-supported platform and the user's data are synchronized through all the different devices. By just typing in the CandyFactory URL, users are able to get their historical data and play the game anytime and anywhere, without bothering installing the game in the Apple App Store [Apple App Store 2019].

1.2 Statement of the Problem

The *CandyFactory Educational Game for iPad* [Aslan, Norton, and Balci 2019; LTRG 2019] received the Top Rated award from Balefire Labs [Balefire Labs 2019]. However, teachers using the iPad game in teaching fractions to middle school students have complained that (a) they are unable to see how the students are performing in game-based learning, and (b) the students who do not have an iPad at home cannot use it outside of school. Some teachers showed great interest in using the app, but they could not because their schools do not have iPads for students to use.

1.3 Statement of the Objectives

The research described herein aims to accomplish the following objectives:

1. Redevelop the *CandyFactory Educational Game for iPad* as a cloud-based software application.
2. Enable anyone to play the CandyFactory educational game anywhere and anytime using a web browser on any kind of Internet-connected tablet (e.g., iPad, Android tablet, Microsoft Surface), laptop (e.g., Mac, Windows, Linux), or desktop (e.g., Mac, Windows, Linux) computer with minimum 1024x768 screen resolution.
3. Enable a teacher to create a class at a school, automatically generate student accounts in that class, and provide the usernames and passwords for the students to use to play the game.
4. Enable a teacher to automatically generate performance reports in game-based learning either individually for each student or for the entire class.

1.4 Overview of the Thesis

This thesis is organized as follows: Chapter 2 describes the CandyFactory iPad version as background and foundational work. Chapter 3 introduces Cloud-based CandyFactory's five tier architecture. Chapter 4 describes CandyFactory's functionality design as well as implementation design. All the Cloud-based CandyFactory functionalities and new features are introduced in Chapter 5. Chapter 6 self-evaluates CandyFactory based on 12 software quality indicators. Chapter 7 provides conclusions and plans for future work.

1.5 Summary of Contributions

The research contributions can be summarized as follows:

1. Redevelop the *CandyFactory Educational Game for iPad* as a cloud-based software application.
2. Enable anyone to play the CandyFactory educational game anywhere and anytime using a web browser on any kind of Internet-connected tablet, laptop, desktop computer with minimum 1024x768 screen resolution.
3. Enable a teacher to create a class at a school, automatically generate student accounts in that class, and provide the usernames and passwords for the students to use to play the game.
4. Enable a teacher to automatically generate performance reports in game-based learning either individually for each student or for the entire class.
5. Provide new features, such as achievements notification animation, to enhance the CandyFactory game. Redesign the interaction techniques.
6. Allow CandyFactory to be updated, tested and deployed quickly without reinstallation or any other operations done by the user and provide CandyFactory elastic scalability.

7. Protect user confidential information by using industry encryption algorithm.
8. Provide synchronization upon different devices.
9. Optimize the data source storage space and operation time complexity.
10. Provide students with an easy-to-memorize account as well as attractive randomly assigned profile image.
11. Enable an administrator to maintain the cloud-based software application.

Chapter 2: CandyFactory iPad Version

“Only about 10% of the apps we have reviewed so far (of a total more than 3,800) have achieved a grade of A or B. Congratulations on your app reaching this level of excellence!” [Balefire Labs], this is how Balefire Labs described the CandyFactory iPad version.

CandyFactory Educational Game for iPad [Aslan, Norton, and Balci 2019; LTRG 2019] is an educational game that provides an effective approach for teaching fractions. By performing operations such as selecting, slicing (partitioning), copying (iterating), and shipping candy bars, students are able to intuitively learn fractions by manufacturing a whole candy bar into a customer requested candy bar size.

2.1 Hardware and Software Requirements

CandyFactory iPad version runs on all generations of iPad (1, 2, 3G, 4G), iPad Air, and iPad Mini (1G, 2G) [Aslan, Norton, and Balci 2019; LTRG 2019].

2.2 Description of Levels

CandyFactory provides five levels of game play in increasing complexity. A tutorial mode teaches how to play each level step by step by an animated CandyFactory Manager. The player mode lets the students to explore and learn themselves by operating on different kinds of candy bar.

The description of the five levels are as follows:

- *Level 1:* In this level, candies are discrete and the fractions are always proper fraction. In this case, students can explore the proper fraction intuitively, and the discrete candies make this level easier to measure.
- *Level 2:* In this level, candies are continuous and the fractions are always unit fraction. Different from the previous level, this level increases the candy’s difficulty but decreases the fraction difficulty. By doing this, we try to let the students understand how to measure a continuous candy bar instead of a discrete one, meantime as the fraction is easier, we can keep a smooth learning curve when the students explore and learn.
- *Level 3:* This level is like level 2, except the fractions can be any proper fraction. Student should begin understanding from 1/m to n/m.
- *Level 4:* This level is like level 3, but the fractions can be any fractions including improper fraction. In this case, students can learn improper fraction intuitively by slicing and copying instead of pure theoretical concepts. Also based on the previous levels, CandyFactory provides a very smooth learning curve for the students.

- *Level 5:* This level does reverse operations to level 4. As a wrap up level, the student can learn fractions from a different aspect thus give and test their whole understanding of all kinds of fractions.

Through the five levels, students are able to gain a smooth, effective and efficient learning curve on fractions.

2.3 Game Mode Introduction

CandyFactory iPad version provides two game modes: *Tutorial Mode* and *Game Mode*. In *Tutorial Mode*, students are guided by a CandyFactory Manager to learn the game procedure and game operations. In *Game Mode*, students are able to play the game themselves in order to learn fractions efficiently and effectively.

2.3.1 Tutorial Mode

The *Tutorial Mode* contains five levels corresponding to each level in *Game Mode*. The "TUTORIAL MODE" on the top of the level selection page indicates that the player is in *Tutorial Mode* (Figure 1). Players can select any level to start the tutorial mode, or choose to go back to main menu.

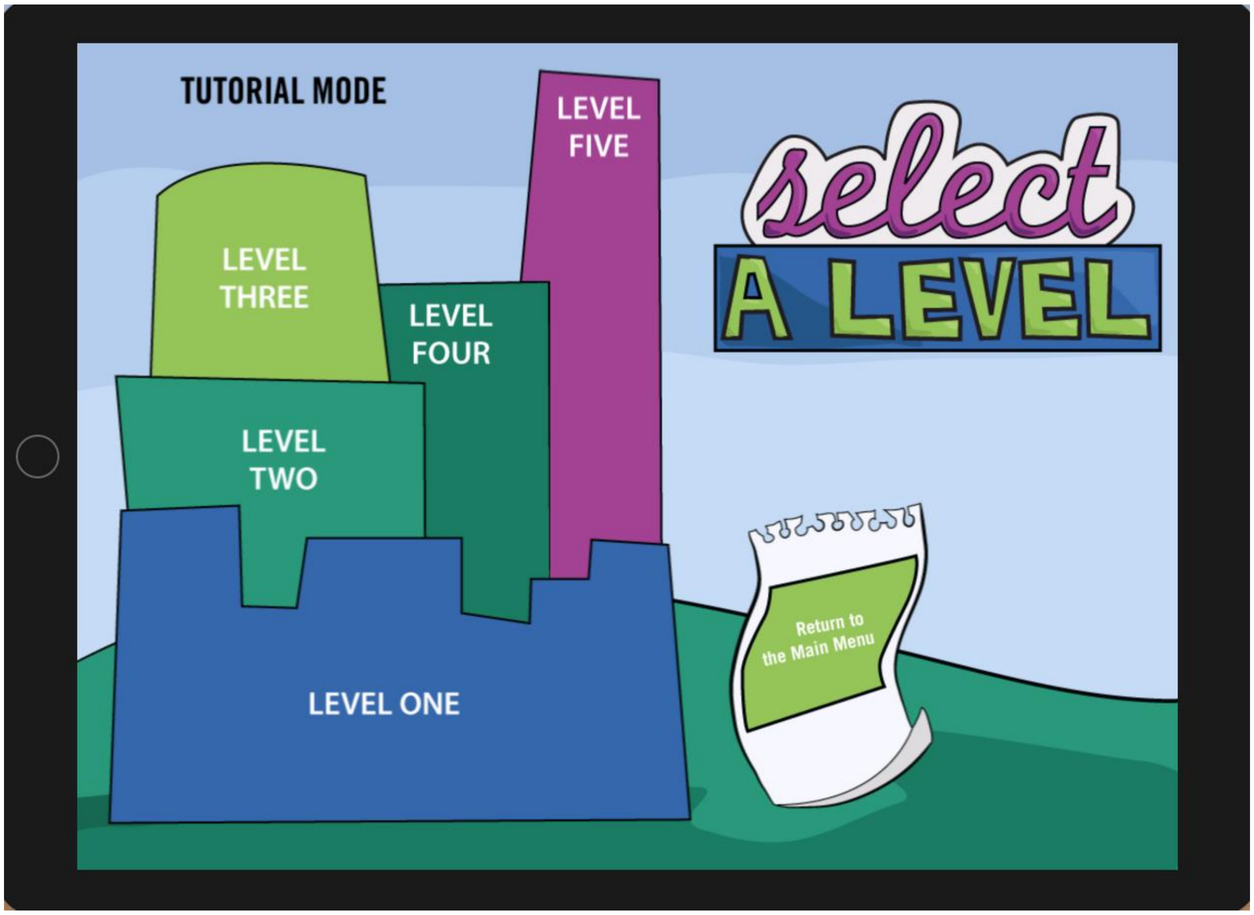


Figure 1. iPad CandyFactory Tutorial Mode Level Selection

Under each level, there is a CandyFactory Manager who explains how CandyFactory is played and guides the user throughout the gameplay (Figure 2). With the guided hands-on tutorial, users can learn how to play the game with selecting, slicing, copying and shipping the candy bars as well as learn the basic interaction techniques CandyFactory provides, such as slice and drag-and-drop.

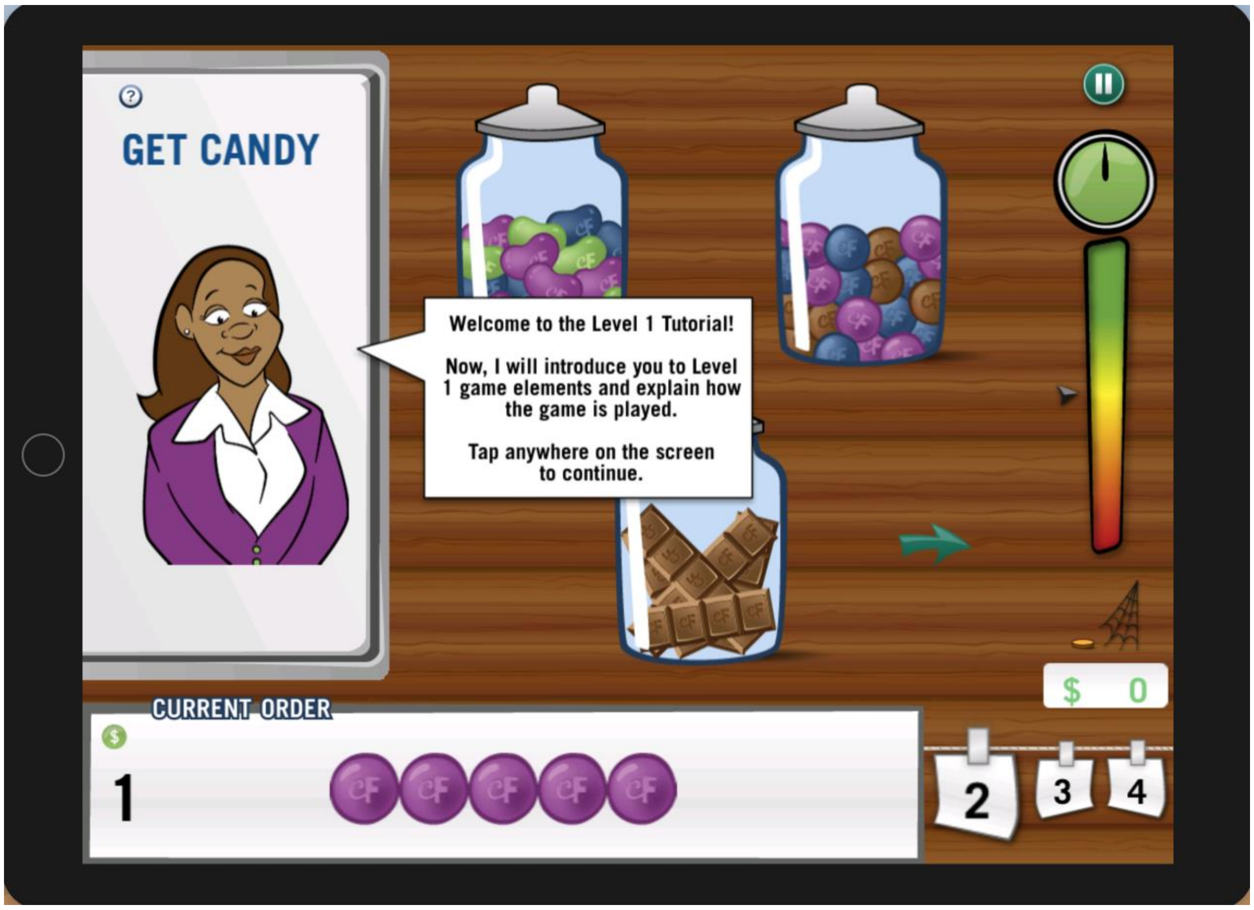

GET CANDY
Welcome to the Level 1 Tutorial!
Now, I will introduce you to Level 1 game elements and explain how the game is played.
Tap anywhere on the screen to continue.
CURRENT ORDER
1
$ 0
2
3
4

Figure 2. iPad CandyFactory Tutorial Mode Manager

During the *Tutorial Mode*, the timer is off so that users are not stressed under time constraint while learning how to play the game. Also users can pause or quit the tutorial mode at any time by clicking the pause button on the top right corner.

2.3.2 Game Mode

In the *Game Mode*, with the initial lock option in the "Option" menu, all the levels are locked except the first level. Only when users pass the previous level will the next level be unlocked (Figure 3).

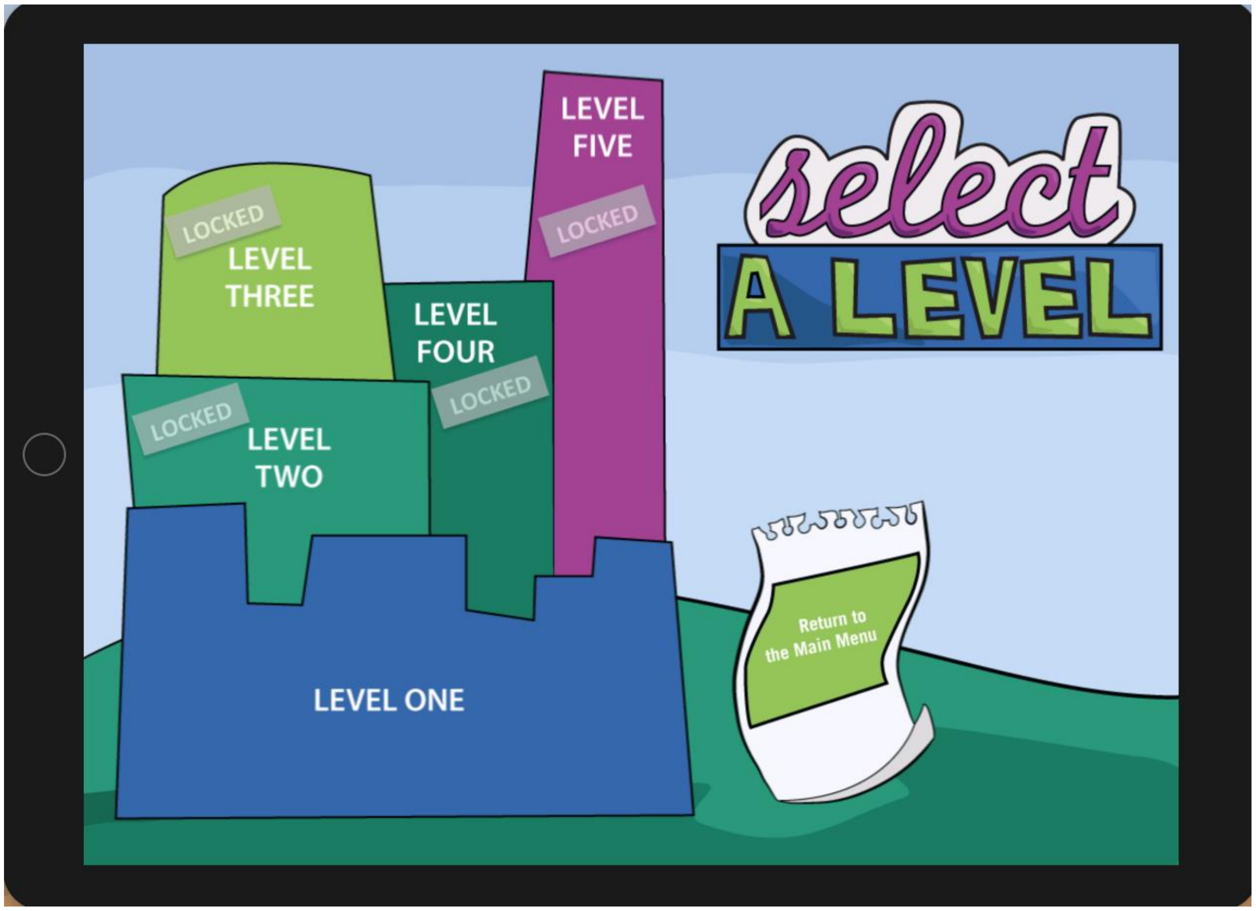


Figure 3. iPad CandyFactory Game Mode Level Selection

When users play each level for the first time, they are forced to pass the corresponding *Tutorial Mode* (Figure 2) for the first game round. Only by passing the *Tutorial Mode* for the first time, users will then be able to play by themselves.

CandyFactory iPad version allows the users to unlock all the levels under the “Option” menu, thus the users have access to any of the five levels without passing the previous level.

The timer for the CandyFactory iPad version is three minutes, users can turn on and off the timer, but unable to change the timer’s time. Also during *Game Mode*, some of the customer orders are bonus orders, in this case by correctly finishing the order, the users can get bonus cash in the game.

2.4 Level Details

In this section, we are going to introduce each level’s details with figures to give a better understanding of CandyFactory. In cloud-based CandyFactory, we reproduce all the functionalities as well as enhance the core features on the cloud. By doing this, cloud-based

CandyFactory not only inherits all the advantages of the iPad version CandyFactory, but also introduces new cloud features and paradigms to make the whole system more powerful.

2.4.1 Level 1 – Discrete Candy of Proper Fraction

In *Level 1*, all the candies are discrete and the fractions are proper fractions (Figure 4). With discrete candies, users can easily count the number of the customer candies and the whole candy bar. In this case, CandyFactory provides the user with a player friendly starting level, which can help the user to be familiar with the game and learn the most basic fraction – proper fraction.

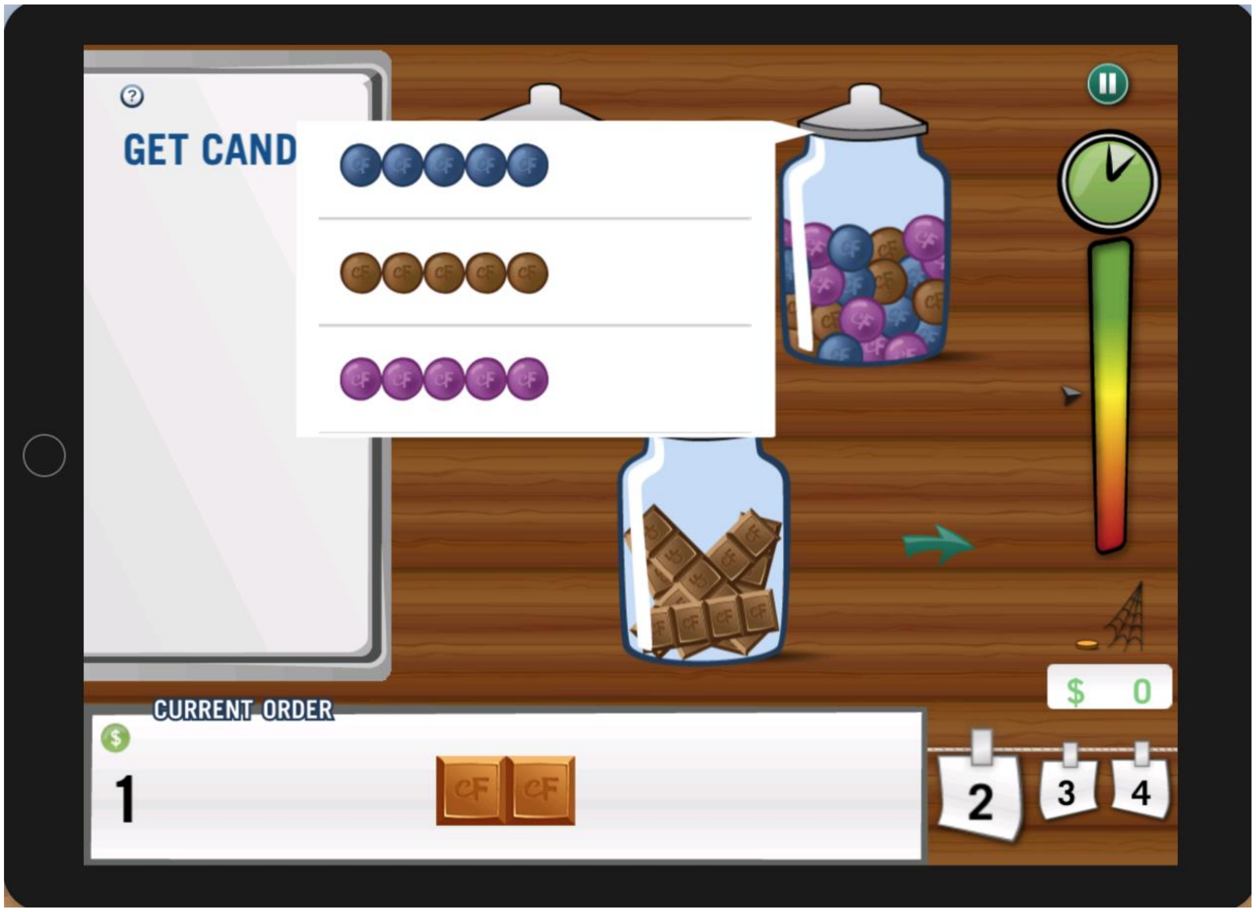


Figure 4. Level 1 Candy Selection

2.4.2 Level 2 – Continuous Candy of Unit Fraction

In *Level 2*, CandyFactory increases the candy bars' difficulty but decreases the fractions' difficulty in order to provide the user a smooth difficulty curve. In this level, all the customer candy bars are continuous but the fractions are unit fraction, which means the fractions are always *1/n* (Figure 5).

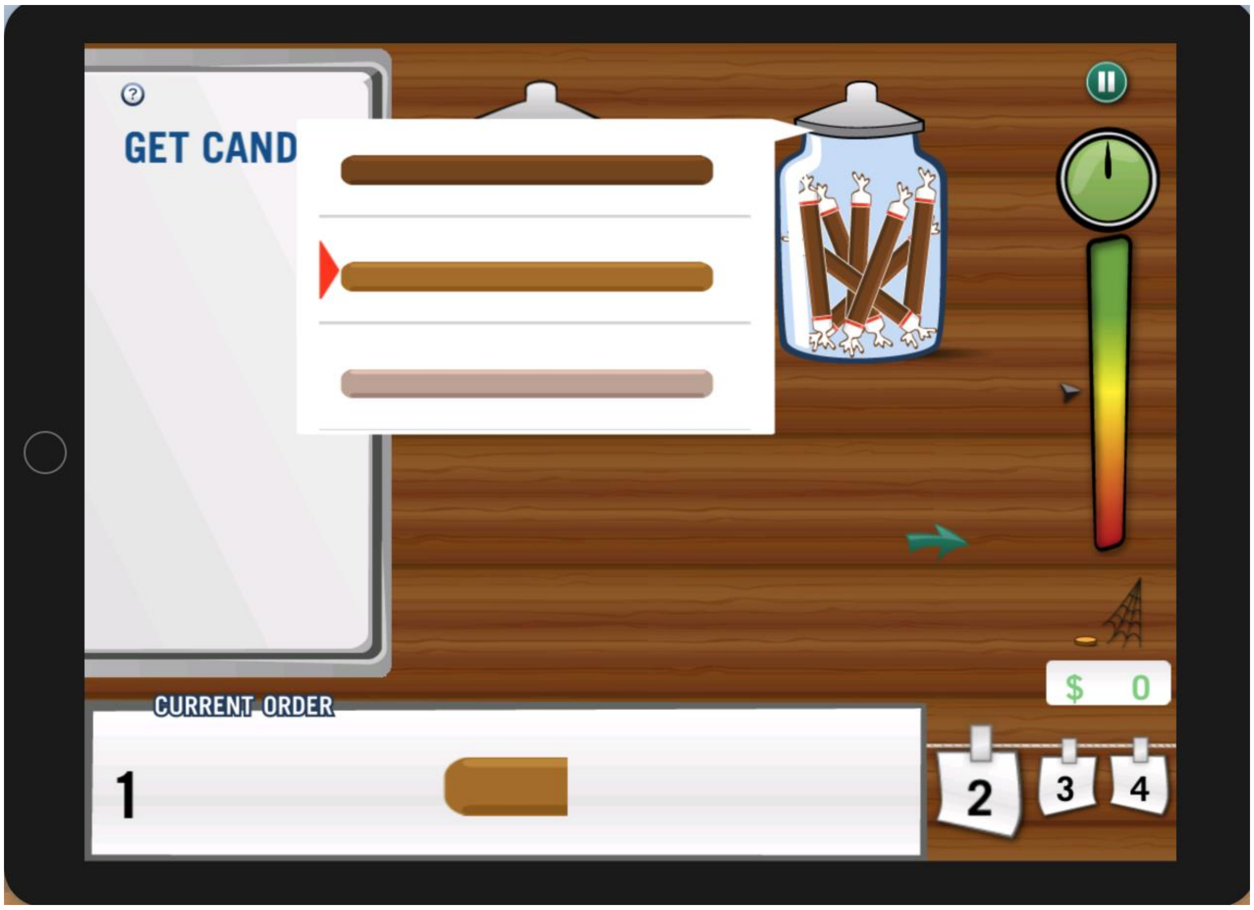


Figure 5. Level 2 Candy Selection

By offering this level, CandyFactory starts to introduce continuous concepts to the users, meantime paves the way for the more advanced levels. From this level, users will operate on continuous candy bars which are harder to count the length of. Thus users may need to use external measuring tool to compare and produce the required length and fractions.

2.4.3 Level 3 – Continuous Candy of Proper Fraction

After being familiar and comfortable with continuous candy bar, CandyFactory increases the fractions' difficulty by changing the unit fractions to proper fractions (Figure 6).

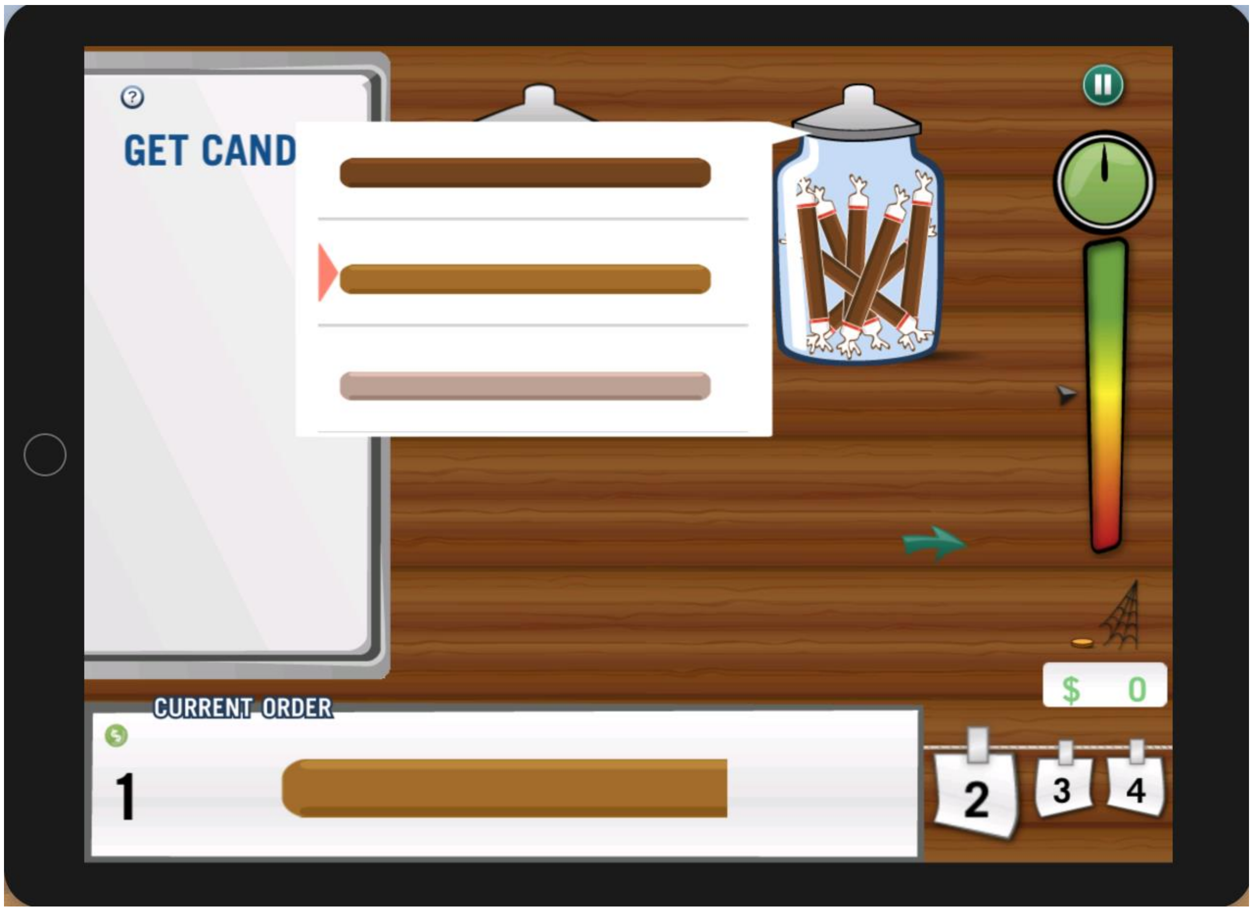


Figure 6. Level 3 Candy Selection

After playing *Level 2*, the users have already been familiar with the continuous candy bar. By playing *Level 3*, users can further learn proper fraction on continuous candy bar. With measuring, selecting, copying and shipping the candy bars, users are able to learn proper fractions intuitively and efficiently.

2.4.4 Level 4 – Continuous Candy of Improper Fraction

For most students, proper fraction is intuitive and easy to understand, for example cutting a pizza into 5 piece, 1 piece is *1/5*. But understanding improper fraction can be hard, for example cutting a pizza into 5 piece, then what is *6/5* of the pizza? CandyFactory, by providing *Level 4*, can help students understand improper fraction intuitively as well. With customer orders, students are going to slice on a shorter candy bar (Figure 7). By concatenating and iterating them one by one to form the customer candy bar, students can have a very vivid situation where an improper fraction will be used.

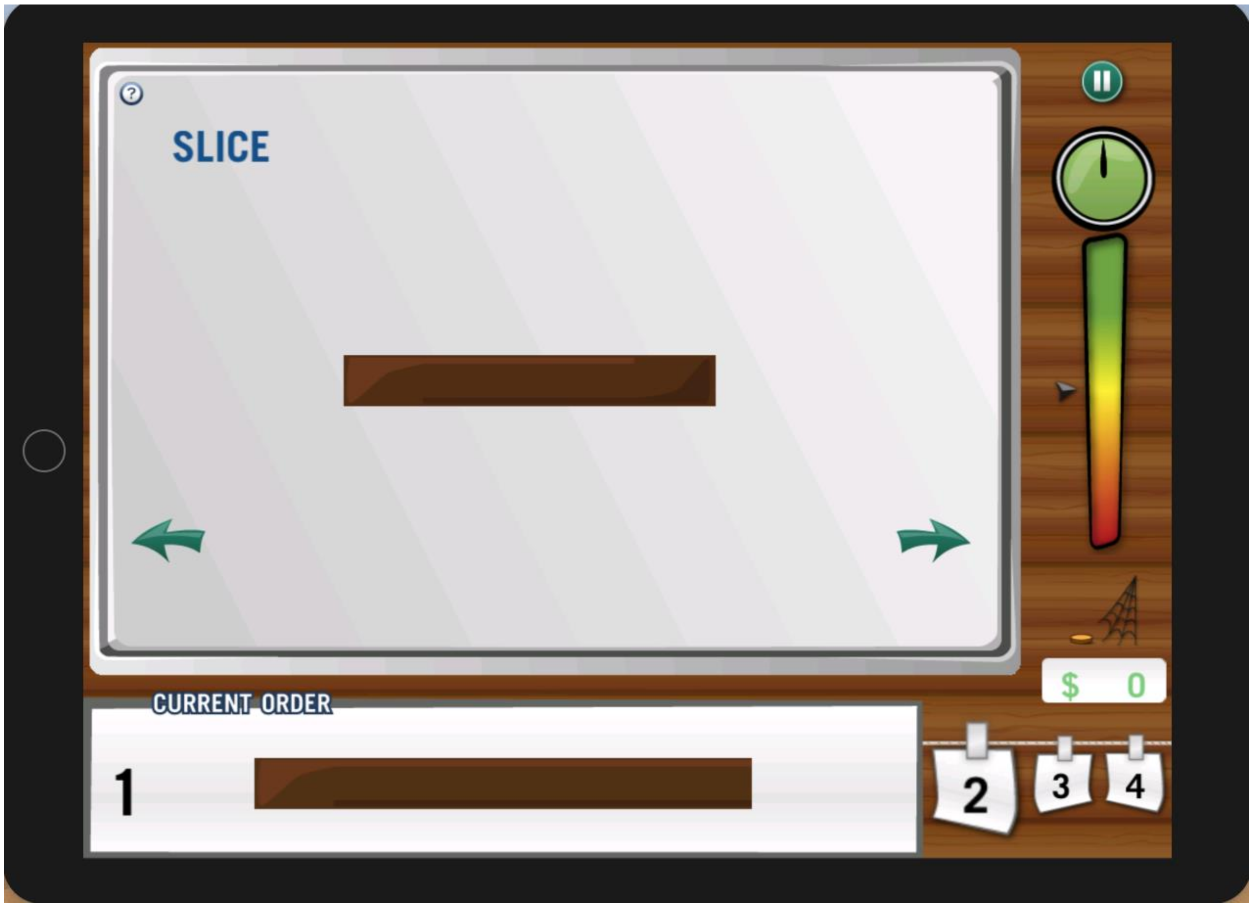


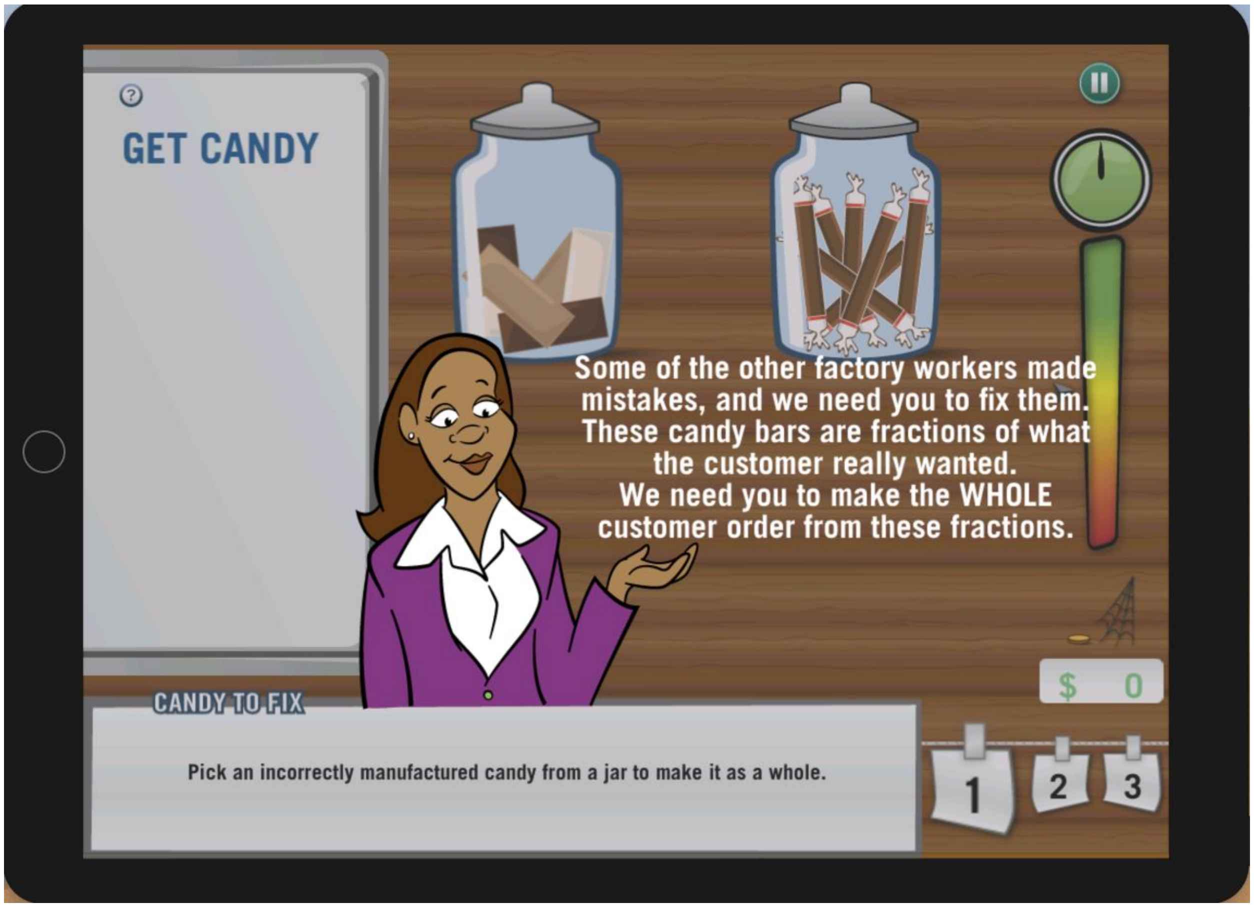
Figure 7. Level 4 Whole Candy Bar Shorter Than Customer Order

By finishing *Level 4*, students have already learned proper fractions as well as improper fractions. And through *Level 1* to *Level 4*, CandyFactory provides a smooth difficulty curve, an effective learning method and an efficient, intuitive way for teaching fractions.

2.4.5 Level 5 – Continuous Candy of Reverse Fraction

Level 5 is a reverse level of *Level 4*. Through *Level 1* to *Level 4*, students should have learned proper and improper fractions on continuous candy bars. In this level, CandyFactory intends to help students use fractions flexibly, thus introduces reverse fractions.

In this level, the student is given a wrong candy to recover it to a whole (Figure 8).



Some of the other factory workers made mistakes, and we need you to fix them. These candy bars are fractions of what the customer really wanted. We need you to make the WHOLE customer order from these fractions.

Pick an incorrectly manufactured candy from a jar to make it as a whole.

Figure 8. Level 5 Reverse Fraction

By finishing this level, students can understand fractions from a different aspect. Compared to previous levels, this level has a different play pattern: firstly, students can pick any of the candy bars in the jar; secondly, since the picked candy is made by the factory worker by mistake, students are required to correct it into a whole candy bar; thirdly, by partitioning and iterating, students can finally correct that candy bar and deliver it to the customer. According to the game process, this level makes CandyFactory game more fresh and attractive, therefore helps students to learn fractions through play and fun.

2.5 Scene Details

CandyFactory iPad version has five levels. Each level contains four scenes and within each scene, users can perform different actions to produce customer candy bars.

In this section, we are going to introduce each individual scene's details with an example of *Level 1*.

2.5.1 *Scene 1 – Candy Selection*

In this scene, there are three candy jars on the top and a customer order at the bottom (Figure 9). Users can click on any of the candy jars to select a candy bar in order to make the customer candy. In Figure 9, in order to make a correct customer candy, users are supposed to select the chocolate bar under the third candy jar.

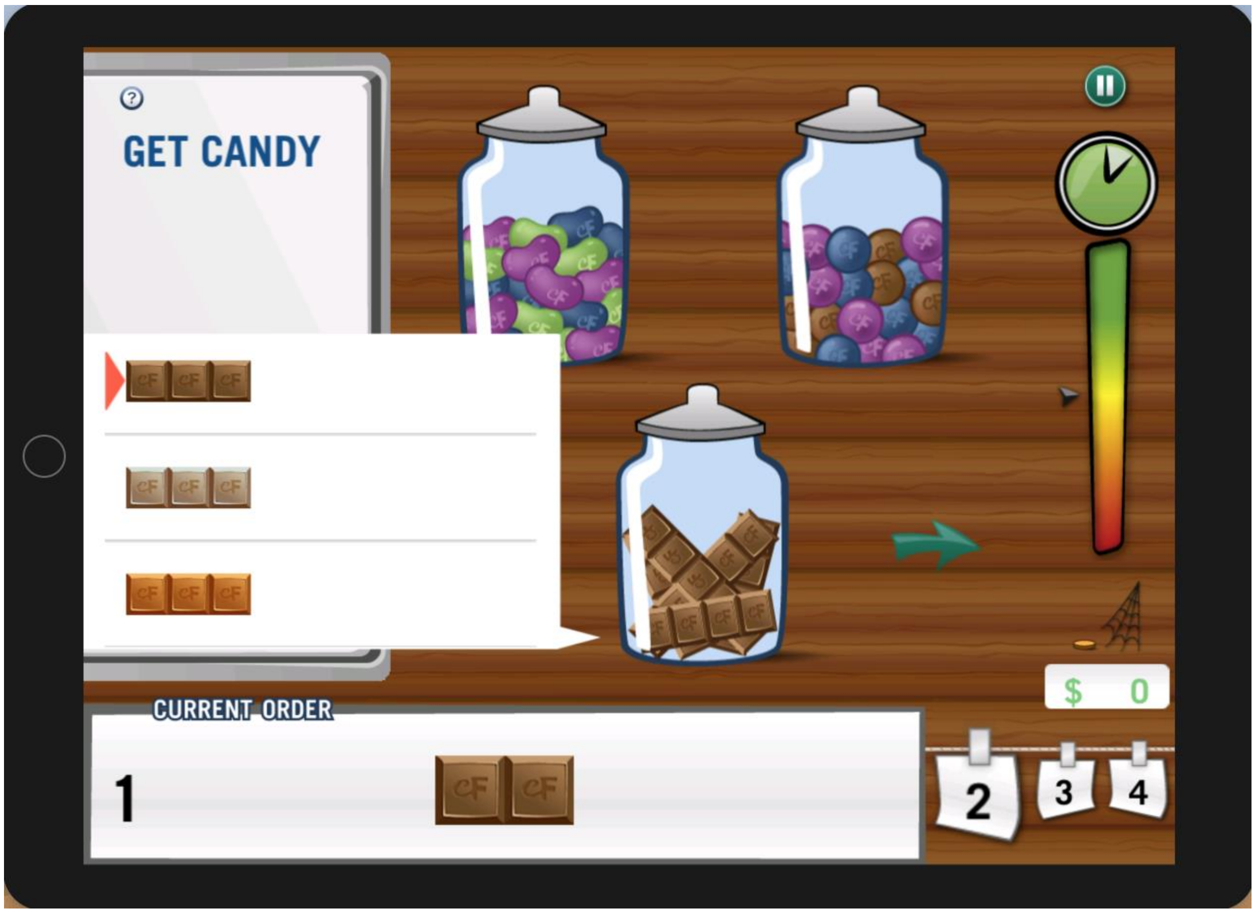


Figure 9. Scene 1 Candy Selection

After selecting the candy bar, users can click on the green arrow to go to the next scene with the selected candy bar.

2.5.2 *Scene 2 – Candy Partition*

In this scene, users are required to slice the selected candy bar into units. There are nine slicing numbers which users can choose from (Figure 10). By choosing a slicing number, there will be dash lines indicate the slices. In Figure 10, the user select to slice the candy bar into 3 pieces, so there are 2 dash lines on the candy bar.

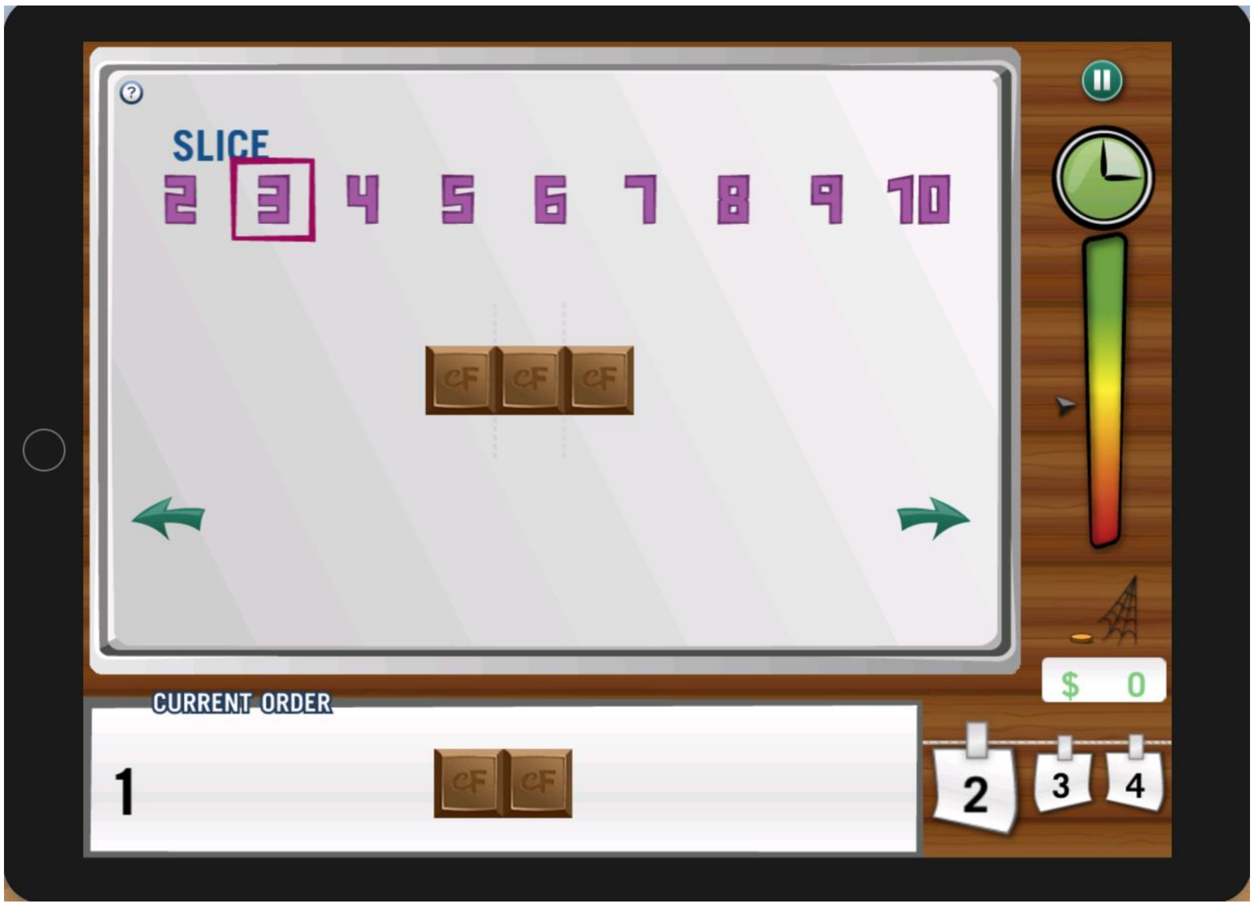


Figure 10. Scene 2 Candy Partition

After slicing the candy bar and getting the unit candy, users can click the right green arrow to go to the next scene with the unit candy, or click the left green arrow to go to previous scene to redo the candy selection.

2.5.3 Scene 3 – Candy Iteration

In this scene, users are required to copy the unit candy several times to make the customer order. Users can drag-and-drop the unit candy into the white box to concatenate them one by one (Figure 11). Meantime, the green fraction will keep updating when the user modifying the candy bar, to reflect the current fraction the manufactured candy represents.

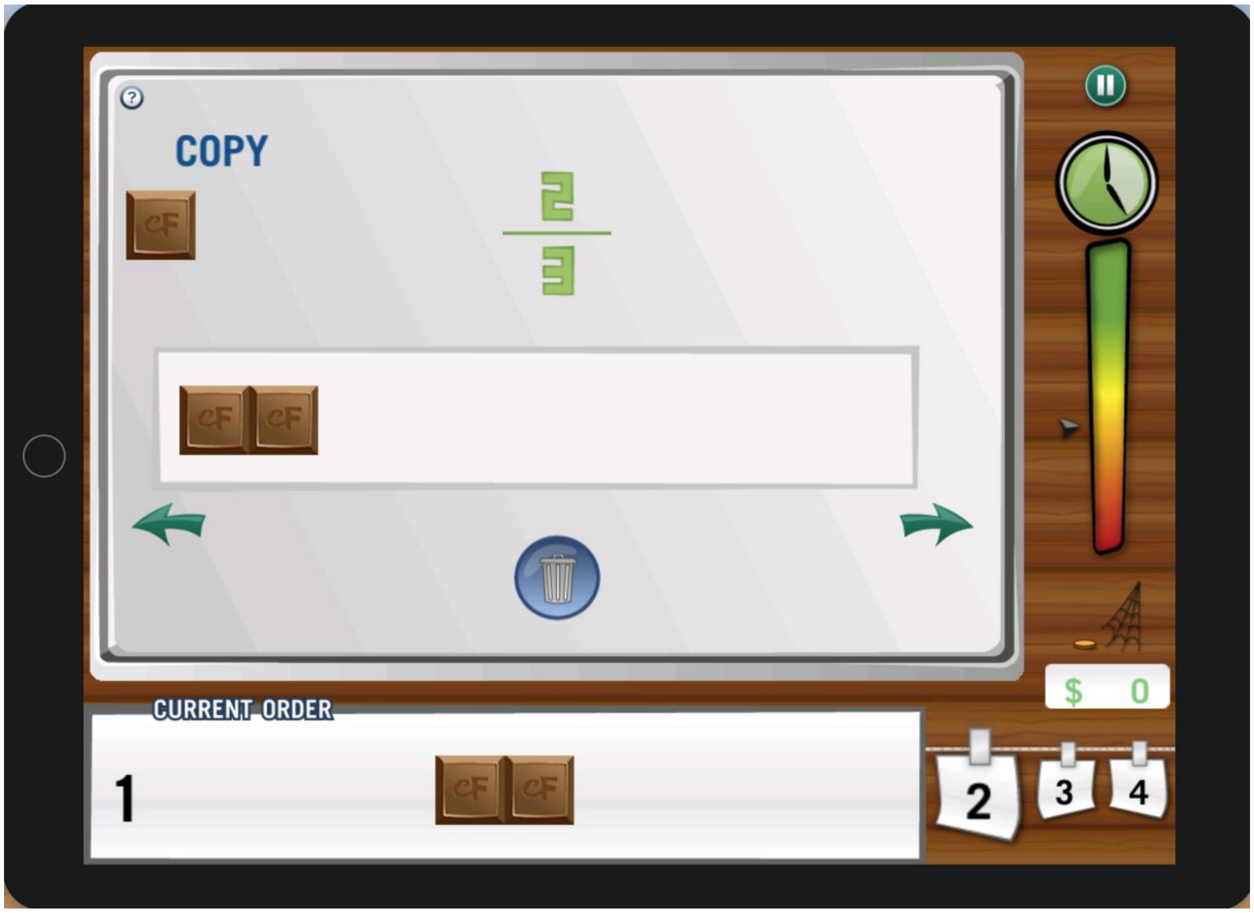


Figure 11. Scene 3 Candy Iteration

After copying the unit candy for several times, users can click the green arrow on the right to go to the next scene, or click the left green arrow to redo the candy slicing.

2.5.4 Scene 4 – Candy Shipment

In this scene, users are required to check whether the manufactured candy meets the customer order. If so, users can click “Ship” button and drag the manufactured candy to ship it (Figure 12). If the manufactured candy is not correct, users can click on the left green arrow to go back to previous scenes to correct the manufactured candy.

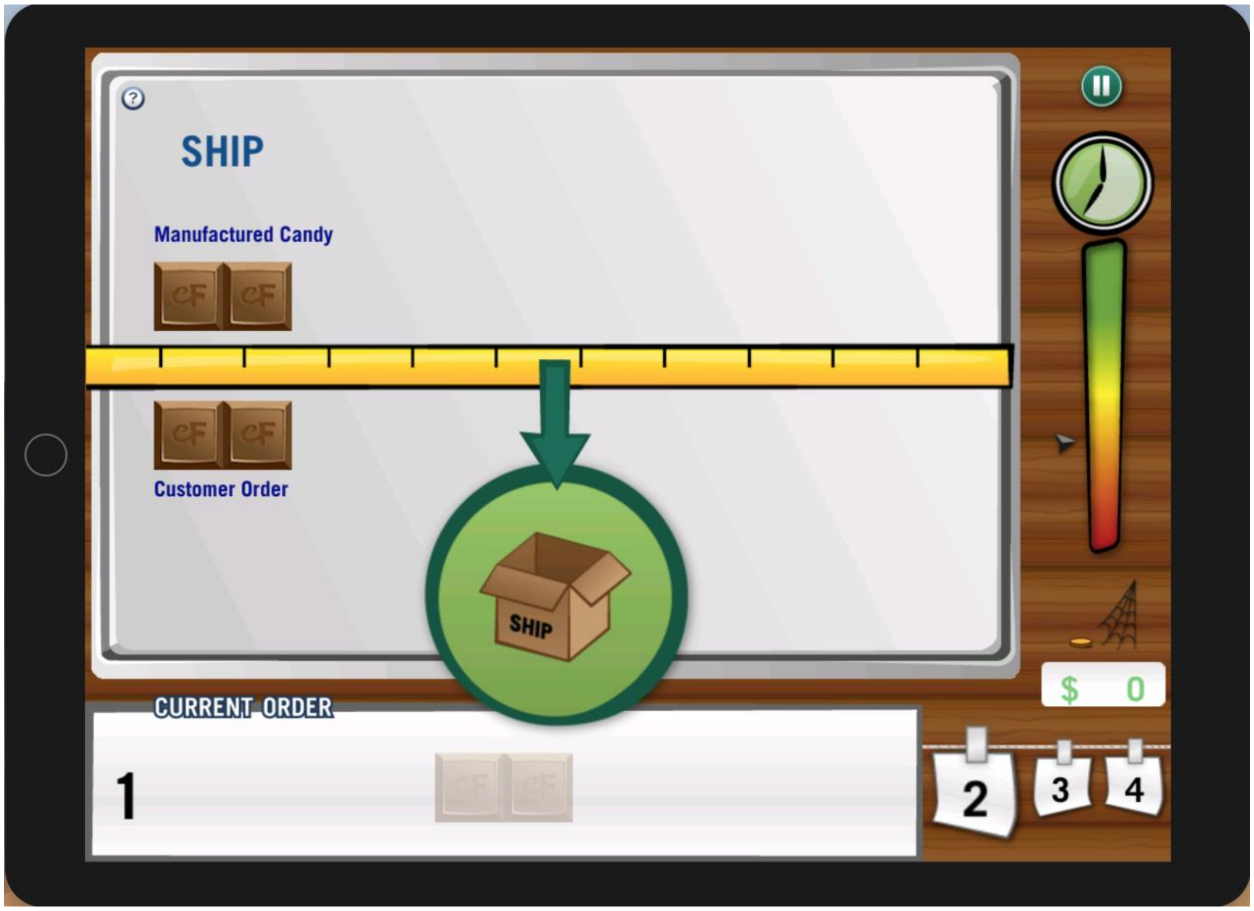


Figure 12. Scene 4 Candy Shipment

After shipping the manufactured candy, if time is not up, the user will start another round immediately. With finishing one round, the manufactured candy and customer candy will be recorded by the CandyFactory. Later when game is over, the stored gameplays will be displayed in “Shift Log”.

2.6 Shift Log

Shift Log of CandyFactory shows the order history of one gameplay, it contains completed orders, customer satisfaction, bonus earned, and performance summary (Figure 13).

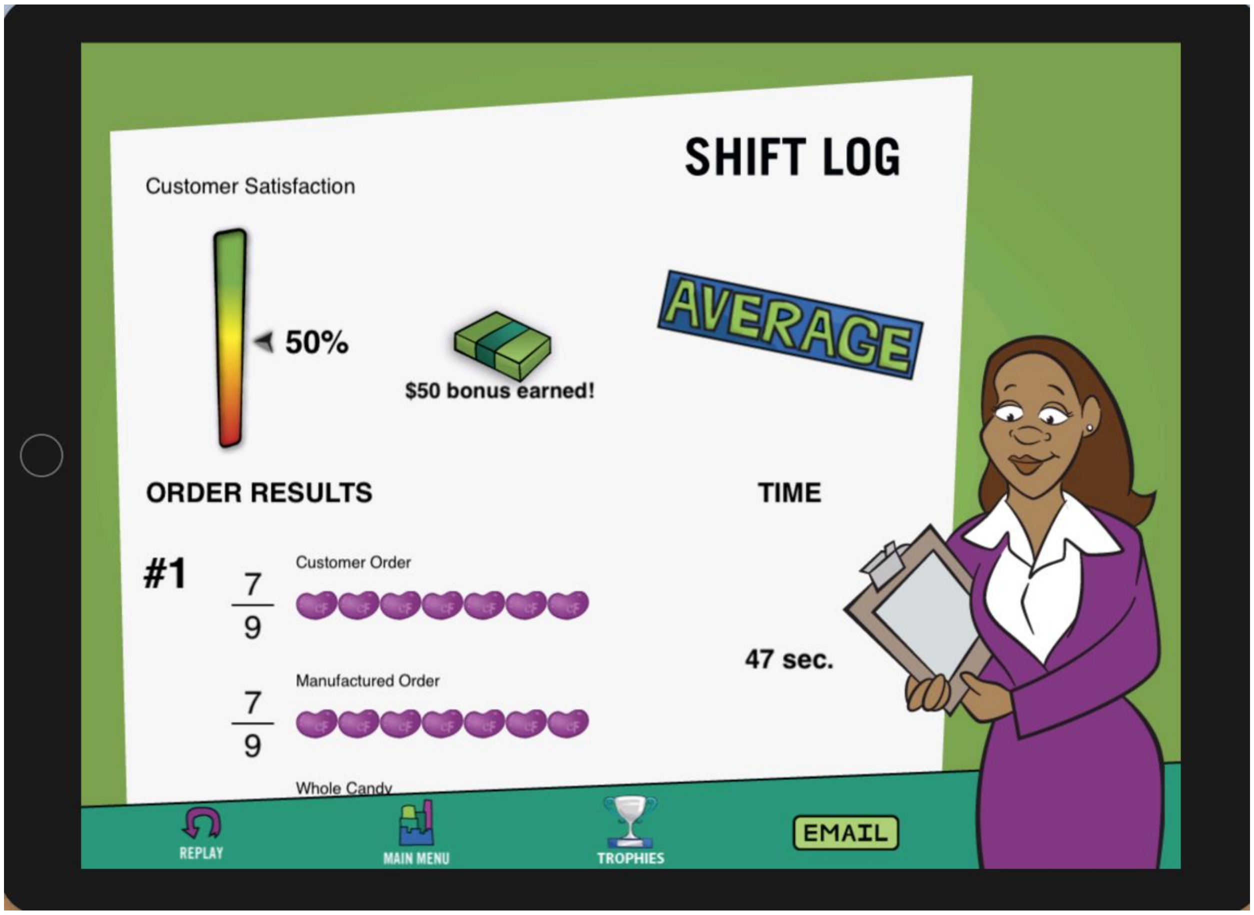


Figure 13. CandyFactory Shift Log

In the order results, *Shift Log* displays the customer order, the manufactured order and the whole candy. By viewing the order results, users are able to review the orders and the fractions they made. Also with displaying each orders time spent, users can compare their performance roughly.

CandyFactory iPad version supports emailing the *Shift Log* page to either user themselves or others. Besides this, there is no other way to store the gameplay history. As long as the user starts a new gameplay, all the previous history will be lost.

2.7 Achievements

The *Achievements* page, as a reward mechanism, can record user achievements based on their gameplay. There are fifteen achievement medals corresponding to fifteen rewards (Figure 13), such as “ship 1 correct order in 30 seconds.”

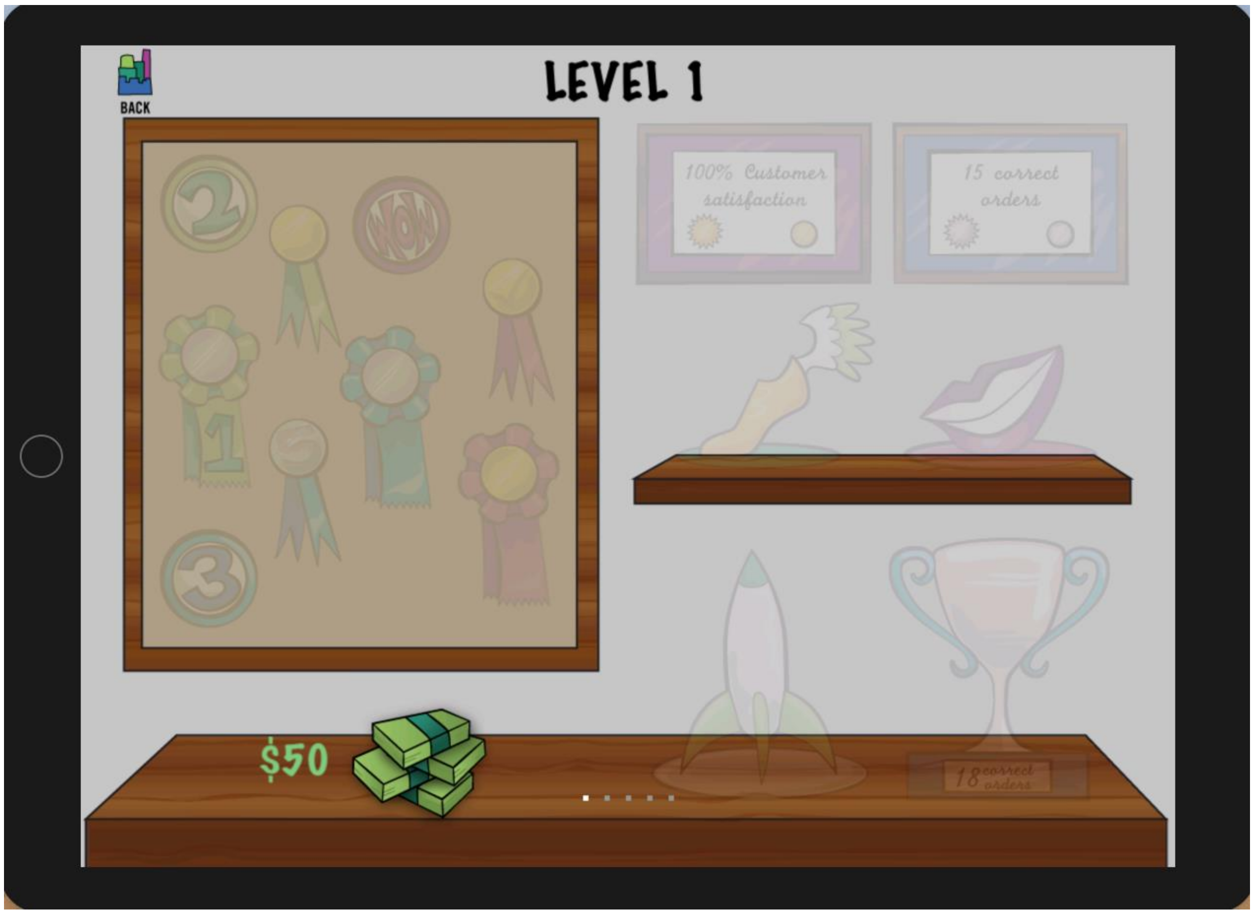


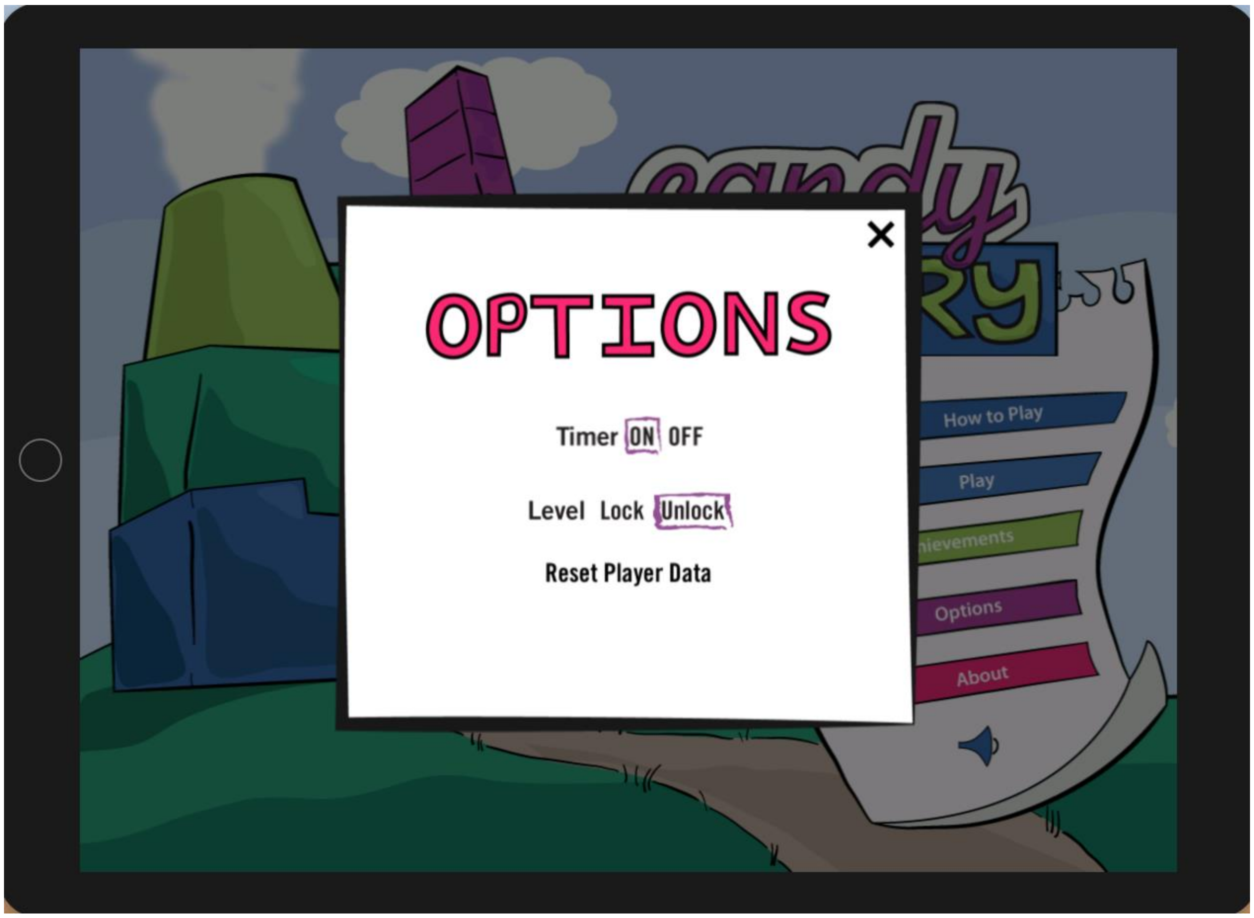
Figure 14. CandyFactory Achievement Scene

After each gameplay, the iPad CandyFactory checks the performances of the orders, if certain metrics are met, CandyFactory will light up and store the achievement medal in local storage.

One drawback of the iPad version CandyFactory is that -- the data is stored in local storage, as long as you change device or reinstall the application, all the previous achievements will be lost. This thesis will solve this problem by applying cloud software techniques to CandyFactory.

2.8 Options

The *Options* dialog allows users to change the game settings. There are three options the user can change (Figure 15): turn on/off of the timer, lock/unlock all the levels in *Game Mode*, and reset the player data. With these options, users can set the game as their needs.



OPTIONS

Timer [✓ ON] OFF

Level Lock [✓ Unlock]

Reset Player Data

Figure 15. CandyFactory Options Dialog

2.9 Interaction Techniques

The interaction techniques iPad CandyFactory uses are listed as follows:

1. One finger tapping.
2. Two finger tapping.
3. Dragging and Dropping.
4. Swiping.

Chapter 3: CandyFactory Architecture

Cloud-based CandyFactory is developed based on the Java EE client-server architecture. This architecture consists of five tiers: Client Tier, Web Tier, Business Tier, Data Mapping Tier and Data Source Tier [Balci 2019]. Each tier takes its own responsibility to guarantee that the cloud-based CandyFactory is able to run efficiently and robustly.

3.1 Hardware and Software Development Environment

Cloud-based CandyFactory is built on the Java EE platform. We leverage many Java EE APIs including JavaServer Faces (JSF), Expression Language (EL), Context and Dependency Injection (CDI), Enterprise JavaBeans (EJBs), CDI-managed Beans, Java Persistence API (JPA) and JPA Façade Beans. For the Client Tier, we build the game based on JQuery, XHTML and CSS.

To meet the above environment requirements, Cloud-based CandyFactory is deployed on a server computer which has the following hardware and software:

- Hardware Environment
  - PowerEdge T330 server computer
  - 64 GB RAM
  - 480 GB solid state hard drive

- Software Environment
  - CentOS Linus operating system
  - MySQL relational database management system
  - Java Development Kit 8
  - GlassFish 4.1 application server
  - JQuery JavaScript library (built inside CandyFactory)

With the hardware and software development environment on the server computer, Cloud-based CandyFactory can be deployed and run correctly and smoothly.

3.2 Architecture

"An architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution." [IEEE 2000] According to the five tiers of Java EE architecture, different tiers focus on different tasks. For instance, the client tier focuses on CandyFactory game while the business tier focuses on backend logic such as user account management. Figure 16 presents an architectural overview of Cloud-based CandyFactory. The responsibilities of each tier are discussed in the following sections.

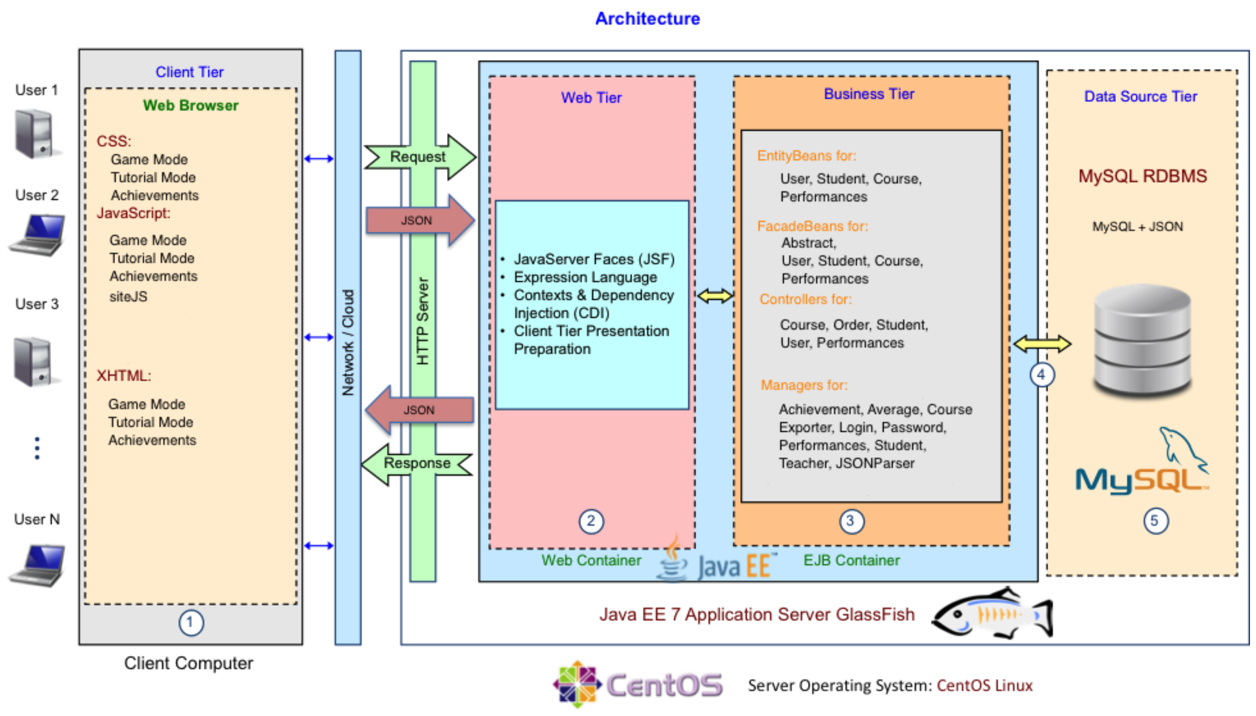


Figure 16: Cloud-Based CandyFactory Architecture

3.2.1 Tier 1: Client

The client tier consists of application clients that access a Java EE server remotely. Clients can be a web browser, a standalone application, or other servers, and they run on a different machine from the Java EE server [Oracle 2018]. Cloud-based CandyFactory's clients are web browsers on any computer or server.

The client tier of Cloud-based CandyFactory provides the interactions between users and CandyFactory. This tier has two main features: firstly, it provides interactions such as CandyFactory game and user account managements; secondly, it provides computational resources to relieve the remote server's workload.

3.2.1.1 Providing Interactions

In Cloud-based CandyFactory, there are two main interactions which the client tier provides: user account management and CandyFactory game interactions.

For user account management interactions, the client tier provides users with easy understandable user interfaces. With the navigation links and explanatory user interfaces, users can interact with the whole system with ease. Also with the client tier, we are able to hide all the complexity of the system behind it, and offer users with a self-explanatory and easy-to-operate way to interact with Cloud-based CandyFactory.

For CandyFactory game, it is fully built in this layer. The reasons we build CandyFactory game fully in this tier are as follows:

1. JQuery and JavaScript are in this tier, we can build a more animated and powerful game on top of JQuery and JavaScript.
2. Client tier is the nearest tier to the users. Thus by building CandyFactory game under this tier, we can minimize the reaction time and maximize the efficiency and the speed of the game. In this case, when users are playing the game, they will have zero latency since this tier is on the users' computers or servers.

According to above, we developed a zero latency, high speed, and well animated CandyFactory game on client layer.

3.2.1.2 Providing Computational Resources

Cloud-based CandyFactory can support thousands of users to play CandyFactory, record their performance, as well as generate performance reports. Thus the server's computational resources and workload need to be carefully designed and handled. In this case, we delicately considered how to balance the workload in order to keep from overloading the server.

Considering the five tiers and their responsibilities, we decided to distribute appropriate workload to the client tier to leverage more client tier's computational resources. In order to do so, we carefully designed the CandyFactory game: instead of frequently interacting with the server to get the next scene's data, we keep all the intermediate data within the client tier. After each game round, we assign achievements and performance calculation to the client tier and only send the final processed results to the remote server. By doing this, we largely decrease the server's workload and allow more users to play the game at the same time. Moreover, we minimize the communication and data transmission time between the client tier and the remote server. The only drawback of this approach is when the user closes the game in the middle of one game round, the intermediate data of this game round will be lost. This drawback is trivial in practice, if the user terminates the game round in the middle, the performance data will then be useless. Therefore, distributing workload to the client tier allows more users to play CandyFactory simultaneously and increases its scalability tremendously.

3.2.2 Tier 2: Web

The web tier contains JavaServer Faces (JSF), Expression Language (EL), and Contexts & Dependency Injection (CDI). With these Java EE technologies, the web tier becomes crucial in the whole architecture.

JavaServer Faces (JSF) plays an important role between the client tier and the business tier. It is a medium for the client tier to pass data to the business tier. In Cloud-based CandyFactory, JSF pages accept information from the user/client tier and then send them to the business tier through Expression Language (EL). With the help of EL, JSF pages can set or receive the business tier data in a very neat format.

Expression Language (EL) provides a communication mechanism between the web tier and the business tier. All the EL can be used within XHTML and JSF pages by annotating them with "#" or "$." With this technology, by hiding the complexity behind EL, JSF pages can "directly" invoke the business tier's methods and fields.

Context & Dependency Injection (CDI) is another very accessible technology frequently used in this tier. With this technology, developers are able to get different references of objects by annotating them with "@Inject." For example, in Cloud-based CandyFactory, a user bean is a session-scoped object which lives through the user's session. In order to get the user bean's reference in other session beans, we can use CDI to inject the user bean's reference to other session beans, thus obtaining the user bean's properties or invoking its methods. Because of CDI's existence, Enterprise JavaBeans are able to be used effectively and efficiently.

3.2.3 Tier 3: Business

The business tier, as the most crucial tier, provides the most important functionalities of a Java EE-based cloud software application. There are five main technologies in this tier: Enterprise JavaBeans (EJBs), CDI-Managed Beans, Java Persistence API (JPA) Entity Beans, JPA Façade Beans, and JAX-RS RESTful Services. In Cloud-based CandyFactory, we leverage the first four technologies to develop the application.

Enterprise JavaBeans and Java Persistence API (JPA) provide services related to the data source tier. With these two technologies, Cloud-based CandyFactory can store and retrieve data from the data source neatly and efficiently. Since the entity beans are generated from the database tables, Java EE introduces JPA Façade Beans in order to provide the developer with easier API. This technology is designed on top of Façade Design Pattern [Schmidt and Douglas 2013], by encapsulating all the complexity within the functions, the Façade Beans offer a very brief and intuitive API to the developers. For example, in order to create a new record in database, firstly developers need to connect to the database; secondly they need to get the persistence context and the reference of the entity manager; finally, with the correct SQL statement, the developers can create the new object in the database. Fortunately, these operations are all encapsulated within Façade Beans. By encapsulating them, the Façade Beans provide an API which takes only the to-be-created object. The above complex creation procedure is taken care of by the Façade Beans.

The CDI-Managed Beans offer the developers a new feature, with which they can inject a managed bean into another by adding the "@" annotation instead of recreating them. In Cloud-based CandyFactory, we use this technology to get the logged in user's information. By using this technology, developers can save tons of time and effort when dealing with different managed beans.

The business tier links the web tier and the data mapping tier. It receives data from the frontend, applies application logics, then stores it into the database, and vise versa. As a result, this layer plays a core role throughout the Java EE architecture and the whole Cloud-based CandyFactory.

3.2.4 Tier 4: Data Mapping

The data mapping tier works as a connection tool between the business tier and the data source tier. It includes Java Persistence API and Java Database Connectivity (JDBC). JDBC is a module which executes SQL queries based on Java requests. In order to keep the application running, the remote server must always have an active JDBC connection.

3.2.5 Tier 5: Data Source

The data source tier provides different representations of data. Java EE supports both relational (e.g. MySQL) and non-relational (e.g. MongoDB) databases.

Cloud-based CandyFactory uses a relational database management system – MySQL – to be the data source layer. For relational databases, they can efficiently store consistent and structured data. However, relational databases lack horizontal data scalability. In Cloud-based CandyFactory, the game performance data are highly mutable and unstructured while the other data are consistent and structured. In order to achieve efficient data storage and retrieval while maintaining the game performance data's scalability, we use a MySQL + JSON diagram to store all the user information as well as the game data.

For consistent and structured user data such as user account information, we store them in standard MySQL tables. Thus SQL queries can perform a secure and efficient way of modifying the data in MySQL relational databases. As long as these kinds of information are designed, they will not be changed on-the-go in the future. As a result, we use standard MySQL relational database tables to store the user information.

For mutable and flexible data such as game performance information, we store them as serialized JSON objects in the MySQL database. For example, for performance data such as "correctness" and "time," instead of creating a MySQL table with columns "correctness" and "time," we wrap them into a JSON object as "correctness: true" and "time: 10s." We then serialize the JSON object as a string to store it into MySQL database. The column of the performance in MySQL database can be the "VARCHAR" type. On top of this architecture, Cloud-based CandyFactory is able to leverage the efficiency of standard SQL queries as well as the flexibility of mutable JSON objects. Developers are always able to change the performance data attributes on-the-go without regenerating the whole database table. For example, with the existing attributes "correctness" and "time," we now want one more metric – "efficiency" – to measure the user's performance. We simply add the "efficiency" metric to the JSON object and modify the JSON parser class to correctly parse it. Developers never need to either redesign and regenerate the database nor delete all the old data to fit the new attributes.

By using MySQL relational database + JSON objects architecture, Cloud-based CandyFactory is able to achieve fast efficient database query, high scalability, and easy data source maintenance.

3.3 CandyFactory Components

Cloud-based CandyFactory contains several components in each tier, the components operate collaboratively to ensure the entire cloud software application will run robustly and efficiently. In this section, we introduce the components based on the CandyFactory tier architecture.

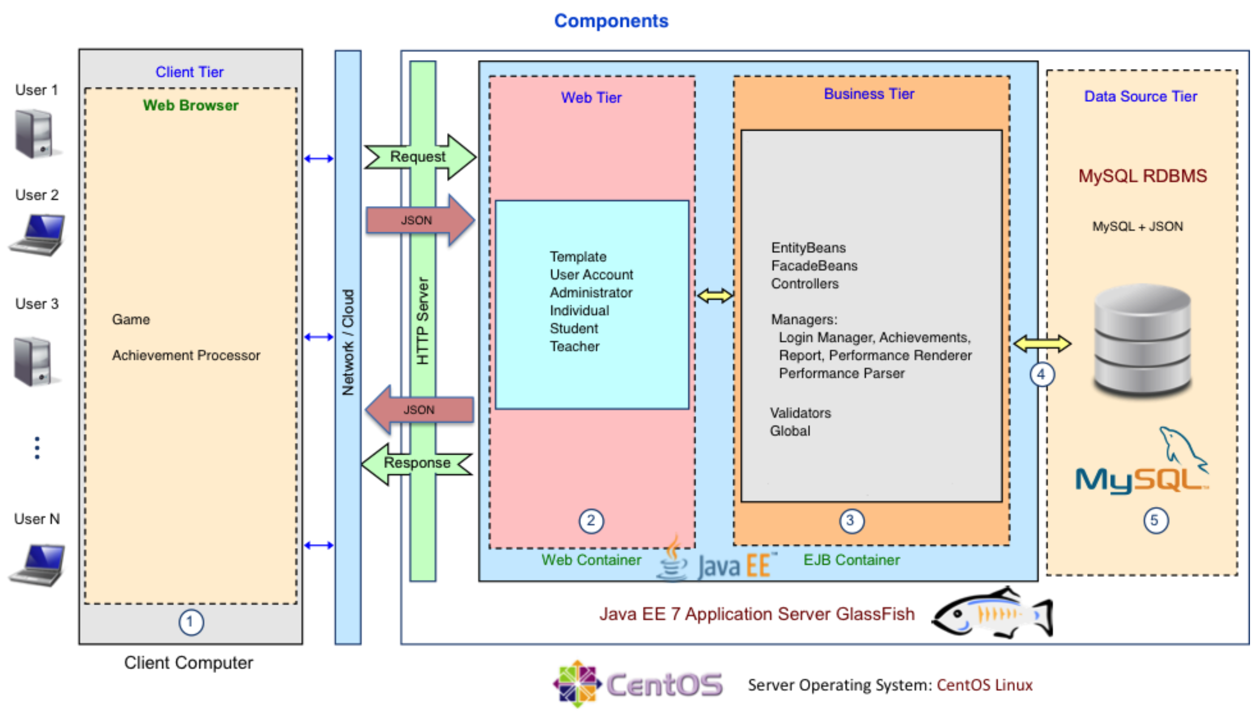


Figure 17: CandyFactory Components

3.3.1 Client Tier Components

The client tier contains all the game interactions as well as computational related components. As introduced in 3.2.1.2, components under the client tier are designed and integrated with game performance computational modules and an achievement system processing module.

3.3.1.1 Game

Game is the most important component throughout Cloud-based CandyFactory. This component provides the gaming animations and interactions with users, which reproduces all the functionalities and features of the iPad version of CandyFactory as well as delivers new features in Cloud-based CandyFactory.

Game also provides the game performance related computational resources. Each round’s performance data is calculated, passed, and processed within the component. By passing the parameters inside this component, Cloud-based CandyFactory can achieve faster performance data processing and better maintainability. As soon as the user finishes a game round, the on-the-go processed performance data are packed as JSON objects and sent to the backend through a hidden form.

3.3.1.2 Achievement Processing

Achievement Processing allows Cloud-based CandyFactory to handle binary representations of the achievement medal list. The optimized algorithm of achievement processing is introduced in 5.5.2.3. In order to process binary representations, this layer firstly acquires the pre-processed achievement data from the backend and stores it in the session map. After achievements are fully loaded and processed by the achievement processing unit, the stored achievements data will be read out from the session map and applied to the achievement web page.

3.3.2 Web Tier Components

The web Tier is composed of JSF pages, which are highly related to the components under this tier. In this section, we introduce the components based on their functionalities.

3.3.2.1 Template

The template holds all of the consistent parts of the whole cloud software, such as the application’s header and footer. It provides growl messages and session timeout functionality. With the site template, all of the consistent modules are composed and organized within it. This component greatly increases the application’s reusability.

3.3.2.2 User Account

The user account provides the account-related operations such as account creation and password modification. With this component, users can modify their account intuitively.

3.3.2.3 Administrator

The administrator component allows the administrator of the application to monitor the whole system’s status, such as registered user accounts and courses created by teachers.

3.3.2.4 Individual

The individual component contains several smaller components. With the individual component, users can view their performance data and generate individual game performance reports. As long as the user type is individual, all the game performance data will be delivered and processed by this component.

3.3.2.5 Student

The student component contains several smaller components. With the student component, students are able to view their performance details as well as generate students' game performance reports. When the user type is student, all the game performance data will be delivered to this component as well as the teacher's components, thus processed by corresponding components to get the final report.

3.3.2.6 Teacher

The teacher component allows teachers to create courses and students. Meanwhile, by accepting data from the student component, the teacher component can generate course reports and even redirect to an individual student's report. In this case, the teacher component allows the teachers to keep track of each student's performance and provides individual students with more targeted help.

3.3.3 Business Tier Components

Components in the business tier largely determine the main logic of the Cloud-based CandyFactory. They accept data from the frontend, process them, and store them into the database. With retrieving the data, they read out data from database, reformat them, and render them to the frontend. Therefore, Cloud-based CandyFactory largely depends on this tier's components to run correctly.

3.3.3.1 Entity Beans

Entity Beans are the most basic components of this tier. They are the representations of the objects corresponding to data tables in the database. According to Object Oriented Design (OOD), Entity Beans are the objects that are manipulated by upper layers' classes.

3.3.3.2 Facade Beans

Façade Beans provide an easy way to operate the database. According to Schmidt and Douglas's paper "Façade Design Pattern" [2013], façade beans offer easier-to-operate APIs such as the "create" function by encapsulating the database operation complexity inside the façade beans. By doing this, façade beans play a sub-bottom layer role throughout the business tier.

3.3.3.3 Controllers

Controllers are top level components in this layer. They leverage the façade beans with EJB technology as well as CDI Injections to implement the complex logic of the system. For example, the user controller is in charge of user creation, user type checking, user account password changing, etc. There are other controllers to take responsibility for manipulating the logic in different directions. By separating them into different controllers, Cloud-based CandyFactory is able to achieve lower coupling and higher cohesion [Eder and Johann 1994].

3.3.3.4 Managers

Managers group all of the components with more advanced and complex functionalities though they are not as strict as Controllers. Managers correspond to the Entity Beans but not as strictly as the Controllers as described in 3.3.3.3. By leveraging different Entity Beans, Façade Beans, and Controllers, Managers are able to implement very powerful logic and functionalities. This subsection introduces some Managers in more detail.

The login manager handles all the login logic. After receiving the login information from the frontend, the login manager first checks the user type and delivers the information to different branches. For students, the login manager obtains the course entity based on the input course ID and then checks whether the username and password are correct. For individuals and teachers, the login manager checks whether the username exists in the database. If yes, it checks whether the input user type is the same as the user type in the database. If yes, the login manager will hash the password using SHA-1 [Eastlake 2001] to check whether the password is correct. If all the information can be matched, the user will be logged in by the login manager.

The achievements manager handles the users’ achievements. When the user clicks on the “Achievements” button in the game, the achievements manager will first read out all of the performance data under the user. Secondly, the achievements manager uses the binary “or” to integrate all the records to get the final achievement integer. Finally, it will redirect the user to the achievement page while sending the processed achievement data to the frontend.

The report managers helps to generate the game performance report. Briefly, by retrieve all the records under users, teachers or courses, the report manager calculates the averages based on the hierarchical data, to generate the final report. In this component, complex data table connections are performed and handled to get the correct data rows. With the grouped data, the averages of the performance metrics can finally be correctly calculated.

The performance renderer renders the preformatted and calculated data.

The data exporter customizes the exportation format of the game report as well as performance data. For example, we can customize the output PDF’s font size and weight in the data exporter. Also we can add detailed information such as author to the output PDF by modifying this component.

The performance parser manager is a JSON parser tool class. It handles all the JSON parsing tasks. Since the performance data is stored in JSON format in MySQL database, it has very good flexibility and mutability. When developers want to add new metrics into the performance JSON format on-the-go, by modifying this component, we can read the old data as well as the new data without redesigning the database nor clean-washing the database. So this component handles the mutable JSON format and provide an object to other classes.

3.3.3.5 Validators

The validators provide functions which can check the users' inputs' correctness. There are four validators in this component – email validator, password validator, username validator, and zip code validator. When users create their accounts, these validators are applied to validate the user input, therefore to protect the whole application from wrong inputs or corruptions.

3.3.3.6 Global

The global contains all the tool components such as password encryption and growl message display components. Every class in the application has access to this component and is able to invoke the tool functions in the global component.

3.3.4 Data Source Tier Components

The two components in this tier is MySQL tables and JSON performance data. JSON performance data is a column of MySQL table, which increases the whole data source's mutability. Developers can add performance metrics at any time without redesigning or recreating the whole database.

3.4 Client-Server Communication Strategy

Cloud-based CandyFactory has two main client-server communication strategies: communication between JSF pages and backend beans and communication between JavaScript and backend beans.

For JSF pages and backend beans communication, Java EE provides Expression Language (EL) to set and get values or invoke functions of backend beans.

For JavaScript and backend beans communication, the strategy is more complex since Java EE doesn't provide technologies nor APIs to support this feature.

To send data from JavaScript to backend beans, JavaScript uses hidden forms and remote commands to submit the data package: firstly, JavaScript packs the data corresponding to the submit form's attributes; secondly, JavaScript invokes the remote command to send the data to backend beans. Within the hidden form, it sets the backend bean's property using EL, thus achieves communicating from JavaScript to backend beans.

To send data from backend beans to JavaScript, the communication strategy is as follows:

1. In JSF page, we add on-click listener to invoke a JavaScript function to accept a backend bean's property by using EL.
2. When the element is clicked, backend bean's properties are passed into the JavaScript function.

3. After JavaScript function accepted the backend bean’s properties, since the JavaScript is not guaranteed to be fully loaded, for safety reason, JavaScript has to store the backend bean’s properties in its session map.
4. As soon as the JavaScript is fully loaded, we retrieve the stored backend bean’s property from the session map.

Using this strategy, Cloud-based CandyFactory allows JavaScript to communicate with backend beans as well as to transmit data.

Chapter 4: CandyFactory Design

Cloud-based CandyFactory's design is based on Object Oriented Design (OOD), which is a paradigm based on the concept of encapsulation, inheritance, and polymorphism. In OOD, everything is object. With OOD paradigm, Cloud-based CandyFactory is able to be implemented in a more reusable, maintainable, and scalable way.

Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software system [Tutorials Point 2019]. It provides abundant notations for representing the design of software development.

Entity Relationship Diagram (ERD) is a type of chart that indicates how entities are related to each other. It is usually used for database table design.

In this chapter, we introduce the design of Cloud-based CandyFactory based on OOD, UML, and ERD, which includs functionality design and implementation design.

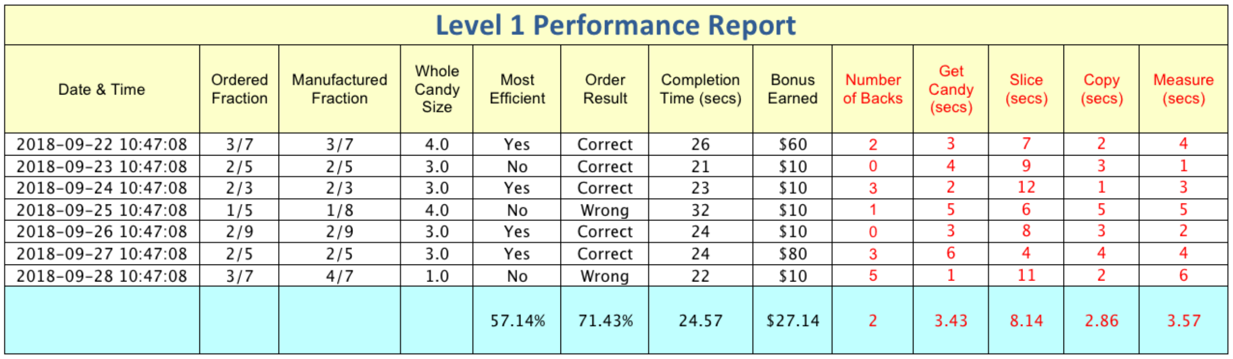
4.1 Functionality Design

Functionality design plays an important role before implementation. In this section, we introduce the design of performance metrics, database, user accounts, and achievement system.

4.1.1 Performance Metrics Design

Performance recording and analyzing are key features of Cloud-based CandyFactory. Considering all the performance data we can get from the frontend game, we decided to record and store the following metrics for further study (Figure 18):

- Date & Time: The date and time when is the game round started.
- Ordered Fraction: The fraction of customer order.
- Manufactured Fraction: The fraction of candy made by the user.
- Whole Candy Size: The whole candy size in the candy jar.
- Most Efficient: Can the user produce the right candy bar with the minimum operations.
- Order Result: Whether the customer order is correctly manufactured.
- Completion Time: The total completion time of the game round.
- Bonus Earned: How much bonus does the user earn.
- Number of Backs: How many times does the user click on the back arrow to go to the previous stage to make corrections.
- Get Candy: The time spent to select the right candy bar.
- Slice Candy: The time spent to slice the candy bar.
- Copy: The time spent to copy the unit candy.
- Measure: The time spent to check and measure the manufactured candy bar and customer order.



Level 1 Performance Report												
Date & Time	Ordered Fraction	Manufactured Fraction	Whole Candy Size	Most Efficient	Order Result	Completion Time (secs)	Bonus Earned	Number of Backs	Get Candy (secs)	Slice (secs)	Copy (secs)	Measure (secs)
2018-09-22 10:47:08	3/7	3/7	4.0	Yes	Correct	26	$60	2	3	7	2	4
2018-09-23 10:47:08	2/5	2/5	3.0	No	Correct	21	$10	0	4	9	3	1
2018-09-24 10:47:08	2/3	2/3	3.0	Yes	Correct	23	$10	3	2	12	1	3
2018-09-25 10:47:08	1/5	1/8	4.0	No	Wrong	32	$10	1	5	6	5	5
2018-09-26 10:47:08	2/9	2/9	3.0	Yes	Correct	24	$10	0	3	8	3	2
2018-09-27 10:47:08	2/5	2/5	3.0	Yes	Correct	24	$80	3	6	4	4	4
2018-09-28 10:47:08	3/7	4/7	1.0	No	Wrong	22	$10	5	1	11	2	6
				57.14%	71.43%	24.57	$27.14	2	3.43	8.14	2.86	3.57

Figure 18. Performance Metrics Design Table

By recording these metrics, we can know "which is the most difficult scene for the students", "whether the order is made most efficiently", "are the students doing better by playing more times", etc. Thus with these metrics, teachers are able to understand the students' learning process better and provide more targeted help to each individual student.

4.1.2 Database Design

Database tables are designed based on Entity Relationship Diagram (ERD). For user types, we designed two tables for different user types: individual, teachers, and administrators are stored under the User table; students are stored under the Student table. Because of student account's particularity, we separate them from the other user types. Considering of protecting students' privacy and decreasing operation complexity, student accounts only have an integer username and a 6 digits' password, which differs from other users. Hence, we separate it into another table.

Students and teachers are linked by the courses. To get a student under a teacher, firstly, we get all the courses created by the teacher; secondly, we get the student number under each course; finally, we sum up all the student numbers of each course to get the total student number of the teacher.

The performance data is separated into two tables: user performance data and student performance data. Each of them has a column to store the JSON string of the game performance. By using both relational database and JSON, Cloud-based CandyFactory allows developers to modify the performance data without redesigning or recreating the database tables.

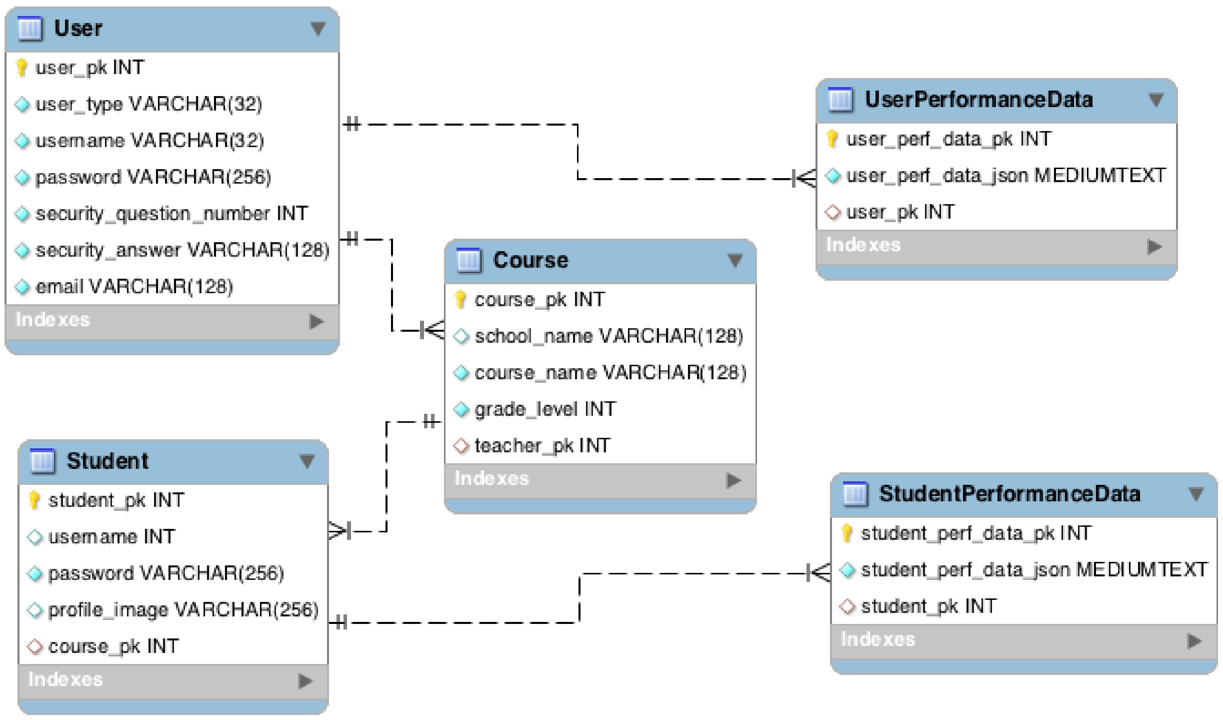


Figure 19. Database ER Diagram

4.1.3 User Account Design

Cloud-based CandyFactory has four types of user accounts: administrator, individual, teacher , and student. In this section we introduce each design based on their features.

Administrator account is a built-in reserved account. With hard-coded username and password, this account has access to the application's global data such as viewing created accounts' information. There are two reasons for hard-coding it instead of storing it in the database. First, hard-code within the application can prevent others from knowing its username. Second, developers won't need to create an administrator account every time the database is recreated. Considering of these two reasons, we decided to hard-code it within the application.

Individual accounts are designed for individual users. They only care about their own gameplay performance. Therefore, for this type of account, we require them to input more detailed information when registering.

Teacher accounts have more functionalities. First, they need to sign up with more detailed information. Second, they are able to create courses. Third, they can create student accounts under each course. Finally, they can generate reports for students and courses. Since the teacher-student interaction through Cloud-based CandyFactory is the core feature of this application, we designed the most functionalities under teacher accounts.

Student accounts have a different account creation mechanism. Considering of the students' privacy and account creation's complexity, we decided to let teachers create their student accounts. For simplicity, all the student accounts under certain course are randomly generated by the system. The account has one integer username and a six-digit password. The username is accumulated from one, thus we can keep the username's complexity as low as possible. Here we design a gap filling technique: if a teacher creates 100 students, the usernames are from 1 to 100; then the teacher deletes the students from 10-20; with new students' coming, the teacher creates another 20 students, the usernames will first fill all the gap between 10 and 20, then increase based on 100. By using the gap filling technique, we can save the usernames' space. Therefore, keep the username integer as small as possible. Cloud-based CandyFactory also offers a new feature to increase the attraction of the game – we assign each student with a random animal profile image to make their accounts special and intuitive. By using hash set in the backend, we are able to avoid duplicates to the max extent. With this new feature, teachers can easily remember which student's account belongs to whom according to the animal profile images.

4.1.4 Achievement System Design

Based on the iPad CandyFactory, cloud-based CandyFactory develops a more complicated but attractive achievement system. Since in the iPad version, CandyFactory only needs to store the current user's achievements locally, the mechanism is much easier. For cloud-based CandyFactory, we need to store achievements under each corresponding user. We are also required to identify the achievements after each play round and merge them with the specific user's existing achievements (Figure 39).

To do this efficiently, we came up with an optimization algorithm to store and retrieve the achievements efficiently.

Cloud-based CandyFactory has 5 levels, each level contains 15 achievement medals. The total medal number is 5 * 15 = 75. Storing the 75 medals intentionally can be storage consuming, since each user needs 75 medal field in the database. Thus, we came up with using *Binary String* to store the medals. For example, for level one, "100000000000000" indicates that this user achieved the first medal. Using *Binary String* can largely reduce the space when storing achievements under a certain user. In this case, after each gameplay, the frontend game can retrieve this user's previous achievement string, add new achievements and then store it back to the database. By using this approach, our cloud-based CandyFactory can store any achievements under their user accounts.

Although the above algorithm greatly saves the space, but three drawbacks still exist:

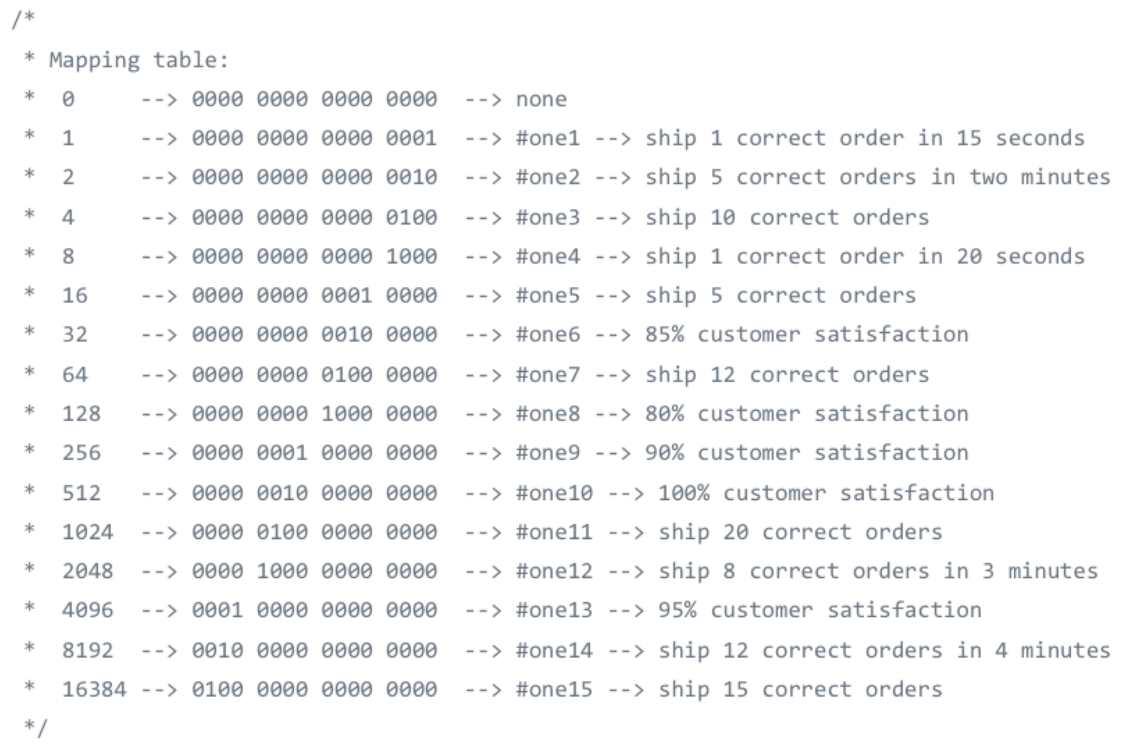
1. Although string saves space compared to individual fields, it is still space consuming: each user has 75 medals, which represented by 75 characters of a string, consuming 150-byte space. With user number increasing, this storage increases dramatically.
2. The read and write from the database are time consuming: frontend sends a request to backend asking for previous achievement string; backend does a query on the database to get the string; backend sends the string back to frontend; frontend adds new achievements to the string; frontend sends the new achievement string to backend; backend stores it into database; frontend starts a new game round. The whole communication between

frontend and backend is extremely time consuming, as it requires back-and-forth data retrieving and storing.

3. Adding new achievement strings to existing achievement string is time consuming. For example, the previous achievement string is "1000" and new achieved achievement string is "1011". In order to union them together, we need to compare every character in the string to union them. The time complexity is *O(n).*

Due to the above drawbacks, we optimized the algorithm even more to make it efficient, fast and small. The optimizations are as follows:

1. Instead of using *Binary String*, we use "binary bit" to store the medals. Since one integer has 4 bytes, which is 32 bits and we only have 15 medals per level. Therefore, one integer is enough for storing one level's achievements. For example, "medal 1" is represented by "0000 0001" which is integer "1"; "medal 2" is represented by "0000 0010" which is integer "2". In this case, each level's medals can be represented by a single integer, which is 4 bytes. The total five levels have 20 bytes. The final mapping table is shown in Figure 20.
2. For communication between the frontend and the backend, instead of reading and writing each time, we only write each round's achievements integer along with other performance data into the database. This is as fast as without this functionality. The only time we read these data is when the user clicks on "Achievement" button. In this case, we leverage "lazy retrieval" of the achievements to greatly increase the speed.
3. Adding new achievements to the previous records is a lot faster compared to the above *Binary String* approach. Since we can use "Bit Or" to union all the achievement integers. The time complexity for this is *O(1).*



```
/*
 * Mapping table:
 *  0     --> 0000 0000 0000 0000  --> none
 *  1     --> 0000 0000 0000 0001  --> #one1 --> ship 1 correct order in 15 seconds
 *  2     --> 0000 0000 0000 0010  --> #one2 --> ship 5 correct orders in two minutes
 *  4     --> 0000 0000 0000 0100  --> #one3 --> ship 10 correct orders
 *  8     --> 0000 0000 0000 1000  --> #one4 --> ship 1 correct order in 20 seconds
 *  16    --> 0000 0000 0001 0000  --> #one5 --> ship 5 correct orders
 *  32    --> 0000 0000 0010 0000  --> #one6 --> 85% customer satisfaction
 *  64    --> 0000 0000 0100 0000  --> #one7 --> ship 12 correct orders
 *  128   --> 0000 0000 1000 0000  --> #one8 --> 80% customer satisfaction
 *  256   --> 0000 0001 0000 0000  --> #one9 --> 90% customer satisfaction
 *  512   --> 0000 0010 0000 0000  --> #one10 --> 100% customer satisfaction
 *  1024  --> 0000 0100 0000 0000  --> #one11 --> ship 20 correct orders
 *  2048  --> 0000 1000 0000 0000  --> #one12 --> ship 8 correct orders in 3 minutes
 *  4096  --> 0001 0000 0000 0000  --> #one13 --> 95% customer satisfaction
 *  8192  --> 0010 0000 0000 0000  --> #one14 --> ship 12 correct orders in 4 minutes
 *  16384 --> 0100 0000 0000 0000  --> #one15 --> ship 15 correct orders
 */
```

Figure 20.Achievement Bit Mapping Table

The comparison of the original design and the optimized design shows as follows:

- Space:
  - Original – 150 bytes per user
  - Optimized – 20 bytes per user
- Time:
  - Original – O(n) union time complexity
  - Optimized – O(1) union time complexity
- Frontend and Backend Interaction:
  - Original – 6 one-way data transmission
  - Optimized – 2 one-way data transmission

4.2 Implementation Design

In this section, we introduce class design and package design based on UML language. The classes and packages are designed based on OOD paradigm such that they obey the three design principles: encapsulation, inheritance, and polymorphism.

4.2.1 Package Design

There are six main packages in the Cloud-based CandyFactory: EntityBeans, FaçadeBeans, Controllers, Managers, Globals, and Validators. Hierarchically, the calling chain is "EntityBeans ← FaçadeBeans ← Controllers ← Managers", while Globals and Validators are standalone packages.

EntityBeans package holds all the entities representing the database tables. FaçadeBeans holds all the facades which provide easier API to controllers. Controllers and Managers handle the main logic of the backend by leveraging the lower level packages. Globals and Validators provide global tool classes as well as validation classes.

By designing the package, all the classes are more organized. Hence, the application meets high cohesion and low coupling paradigm.

4.2.2 Class Design

Class design is a crucial part before implementation. With good design, modules can be more reusable and maintainable. In this section, we introduce the class design based on UML diagrams.

4.2.2.1 Entity Beans

The EntityBeans diagram (Figure 21) shows seven classes and their relationships. Each class has its data fields and methods. The relationships of the classes are represented as lines in the chart.

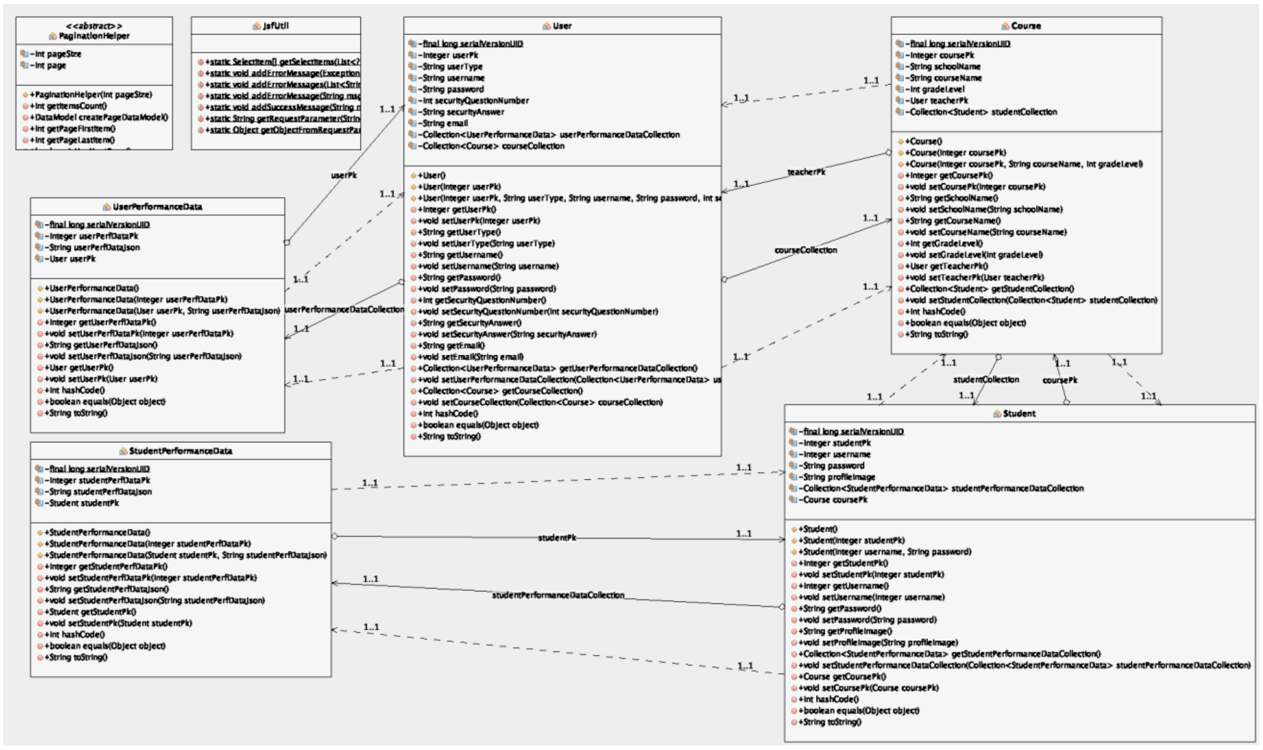

<<abstract>>
PaginationHelper
-int pageSize
-int page
+PaginationHelper(int pageSize)
+int getItemsCount()
+DataModel createPageDataModel()
+int getPageFirstItem()
+int getPageLastItem()
JsfUtil
+static SelectItem[] getSelectItems(List<?
+static void addErrorMessage(Exception
+static void addErrorMessages(List<Strin
+static void addErrorMessage(String msg
+static void addSuccessMessage(String m
+static String getRequestParameter(Strin
+static Object getObjectFromRequestPar
User
-final long serialVersionUID
-Integer userPk
-String userType
-String username
-String password
-int securityQuestionNumber
-String securityAnswer
-String email
-Collection<UserPerformanceData> userPerformanceDataCollection
-Collection<Course> courseCollection
+User()
+User(Integer userPk)
+User(Integer userPk, String userType, String username, String password, int se
+Integer getUserPk()
+void setUserPk(Integer userPk)
+String getUserType()
+void setUserType(String userType)
+String getUsername()
+void setUsername(String username)
+String getPassword()
+void setPassword(String password)
+int getSecurityQuestionNumber()
+void setSecurityQuestionNumber(int securityQuestionNumber)
+String getSecurityAnswer()
+void setSecurityAnswer(String securityAnswer)
+String getEmail()
+void setEmail(String email)
+Collection<UserPerformanceData> getUserPerformanceDataCollection()
+void setUserPerformanceDataCollection(Collection<UserPerformanceData> us
+Collection<Course> getCourseCollection()
+void setCourseCollection(Collection<Course> courseCollection)
+int hashCode()
+boolean equals(Object object)
+String toString()
Course
-final long serialVersionUID
-Integer coursePk
-String schoolName
-String courseName
-int gradeLevel
-User teacherPk
-Collection<Student> studentCollection
+Course()
+Course(Integer coursePk)
+Course(Integer coursePk, String courseName, int gradeLevel)
+Integer getCoursePk()
+void setCoursePk(Integer coursePk)
+String getSchoolName()
+void setSchoolName(String schoolName)
+String getCourseName()
+void setCourseName(String courseName)
+int getGradeLevel()
+void setGradeLevel(int gradeLevel)
+User getTeacherPk()
+void setTeacherPk(User teacherPk)
+Collection<Student> getStudentCollection()
+void setStudentCollection(Collection<Student> studentCollection)
+int hashCode()
+boolean equals(Object object)
+String toString()
UserPerformanceData
-final long serialVersionUID
-Integer userPerfDataPk
-String userPerfDataJson
-User userPk
+UserPerformanceData()
+UserPerformanceData(Integer userPerfDataPk)
+UserPerformanceData(User userPk, String userPerfDataJson)
+Integer getUserPerfDataPk()
+void setUserPerfDataPk(Integer userPerfDataPk)
+String getUserPerfDataJson()
+void setUserPerfDataJson(String userPerfDataJson)
+User getUserPk()
+void setUserPk(User userPk)
+int hashCode()
+boolean equals(Object object)
+String toString()
StudentPerformanceData
-final long serialVersionUID
-Integer studentPerfDataPk
-String studentPerfDataJson
-Student studentPk
+StudentPerformanceData()
+StudentPerformanceData(Integer studentPerfDataPk)
+StudentPerformanceData(Student studentPk, String studentPerfDataJson)
+Integer getStudentPerfDataPk()
+void setStudentPerfDataPk(Integer studentPerfDataPk)
+String getStudentPerfDataJson()
+void setStudentPerfDataJson(String studentPerfDataJson)
+Student getStudentPk()
+void setStudentPk(Student studentPk)
+int hashCode()
+boolean equals(Object object)
+String toString()
Student
-final long serialVersionUID
-Integer studentPk
-Integer username
-String password
-String profileImage
-Collection<StudentPerformanceData> studentPerformanceDataCollection
-Course coursePk
+Student()
+Student(Integer studentPk)
+Student(Integer username, String password)
+Integer getStudentPk()
+void setStudentPk(Integer studentPk)
+Integer getUsername()
+void setUsername(Integer username)
+String getPassword()
+void setPassword(String password)
+String getProfileImage()
+void setProfileImage(String profileImage)
+Collection<StudentPerformanceData> getStudentPerformanceDataCollection()
+void setStudentPerformanceDataCollection(Collection<StudentPerformanceData> studentPerformanceDataCollection)
+Course getCoursePk()
+void setCoursePk(Course coursePk)
+int hashCode()
+boolean equals(Object object)
+String toString()
userPk
userPerformanceDataCollection
teacherPk
courseCollection
studentCollection
coursePk
studentPk
studentPerformanceDataCollection
1..1

Figure 21. UML diagram of Entity Beans

User class have three types of users. For individuals, they are related to *UserPerformanceData* by *userPk*. For teachers, they are related to courses. Course is related to students. Each student has his/her own performance, thus *StudentPerformanceData* is related to Student.

4.2.2.2 Façade Beans

Façade beans' UML diagrams show in Figure 22. All the façade beans are derived from *Abstract Façade*, which provides basic database persistence operations.

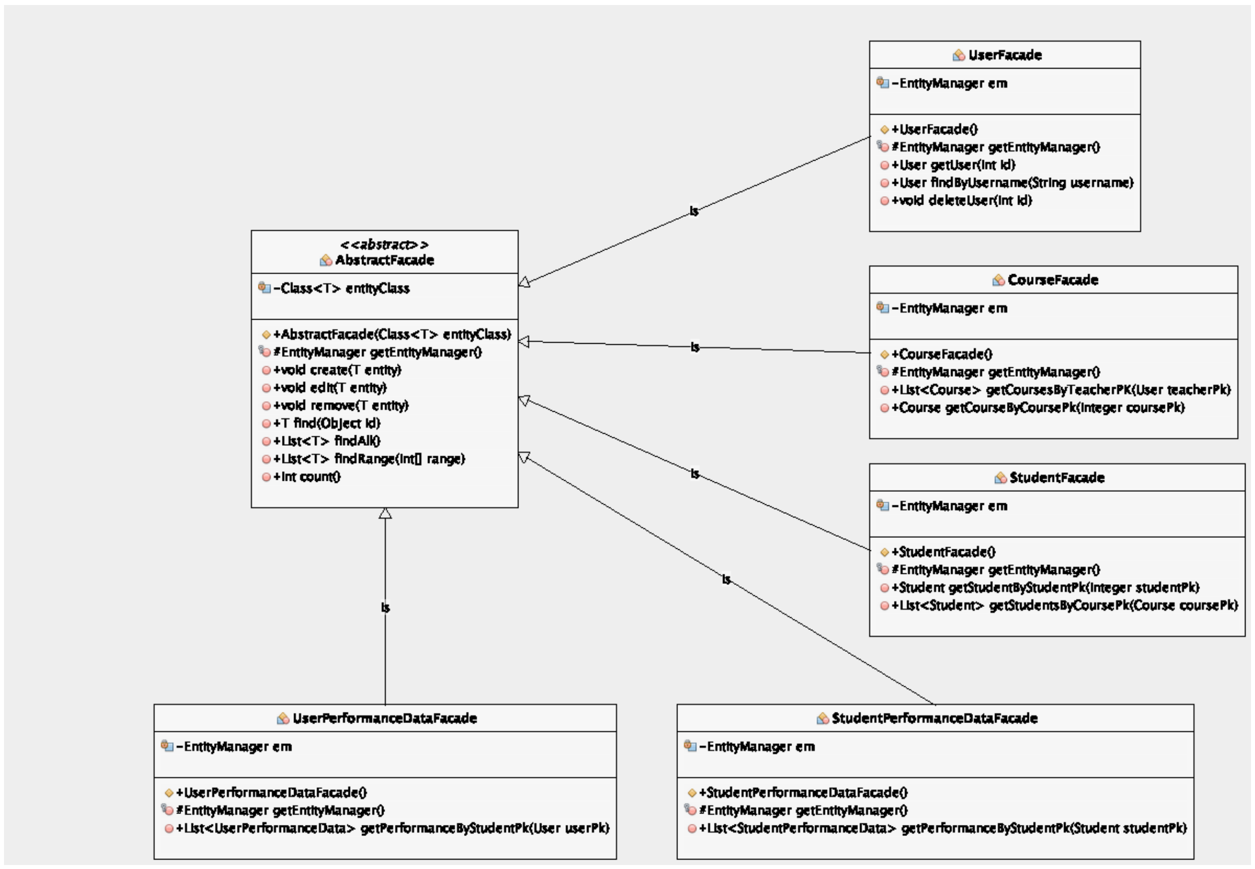

UserFacade
-EntityManager em
+UserFacade()
#EntityManager getEntityManager()
+User getUser(int id)
+User findByUsername(String username)
+void deleteUser(int id)
<<abstract>>
AbstractFacade
-Class<T> entityClass
+AbstractFacade(Class<T> entityClass)
#EntityManager getEntityManager()
+void create(T entity)
+void edit(T entity)
+void remove(T entity)
+T find(Object id)
+List<T> findAll()
+List<T> findRange(int[] range)
+int count()
CourseFacade
-EntityManager em
+CourseFacade()
#EntityManager getEntityManager()
+List<Course> getCoursesByTeacherPK(User teacherPk)
+Course getCourseByCoursePk(Integer coursePk)
StudentFacade
-EntityManager em
+StudentFacade()
#EntityManager getEntityManager()
+Student getStudentByStudentPk(Integer studentPk)
+List<Student> getStudentsByCoursePk(Course coursePk)
is
is
is
is
is
UserPerformanceDataFacade
-EntityManager em
+UserPerformanceDataFacade()
#EntityManager getEntityManager()
+List<UserPerformanceData> getPerformanceByStudentPk(User userPk)
StudentPerformanceDataFacade
-EntityManager em
+StudentPerformanceDataFacade()
#EntityManager getEntityManager()
+List<StudentPerformanceData> getPerformanceByStudentPk(Student studentPk)

Figure 22. UML Diagram of Facade Beans

4.2.2.3 Controllers

Controllers UML diagram shows in Figure 23.

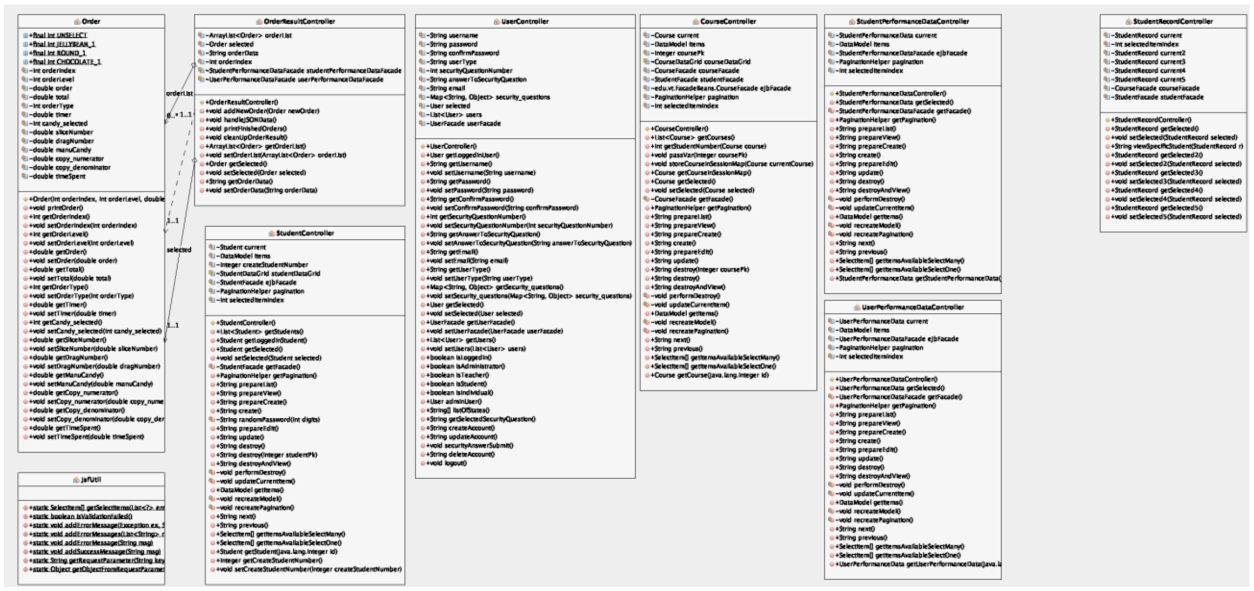


Figure 23. UML Diagram of Controllers

4.2.2.4 Managers

Managers UML diagram shows in Figure 24. Since managers handle most of the main logics, the classes inside it have more complex relationships. By handling the relationships properly, Cloud-based CandyFactory is able to run correctly and robustly.

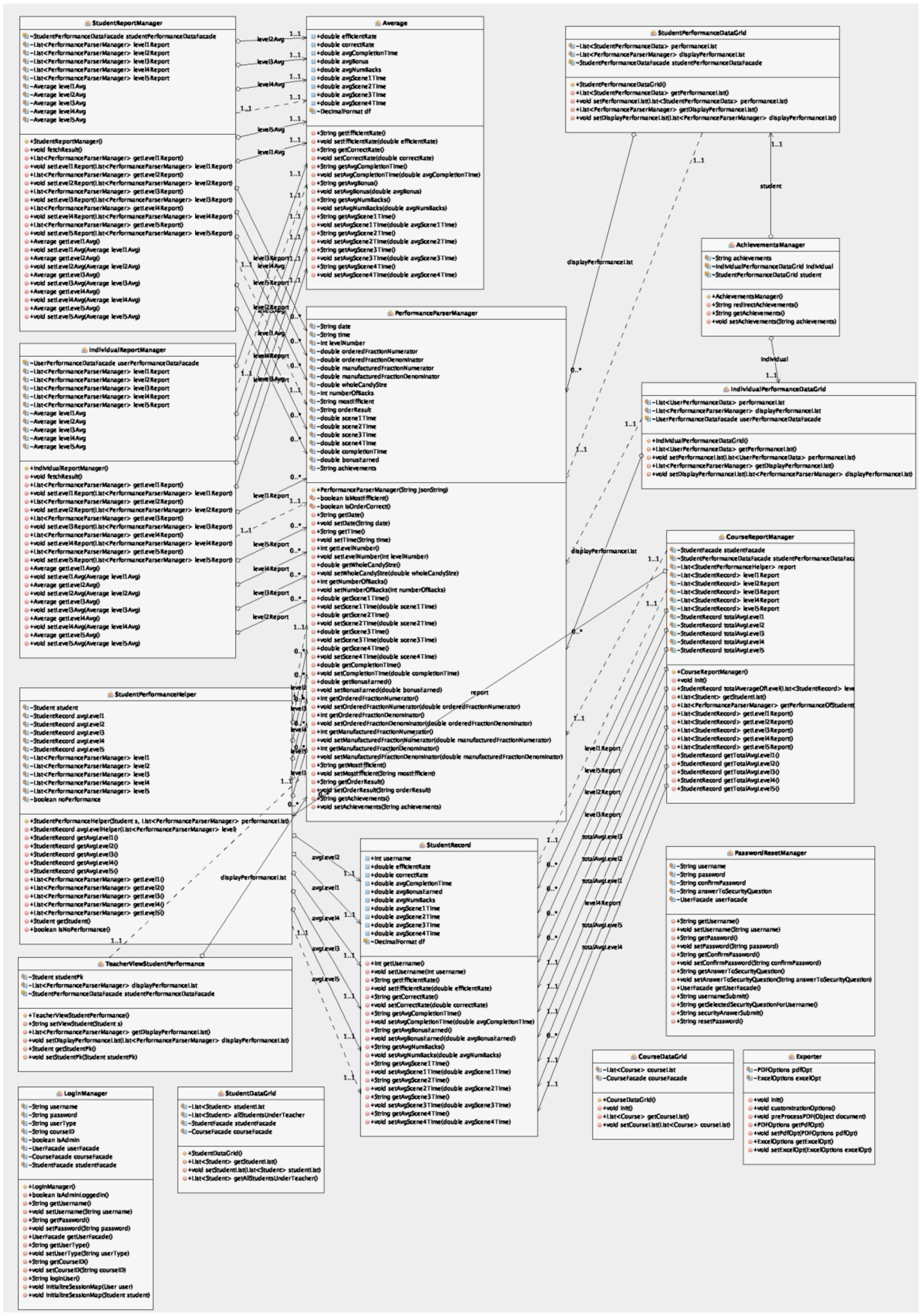


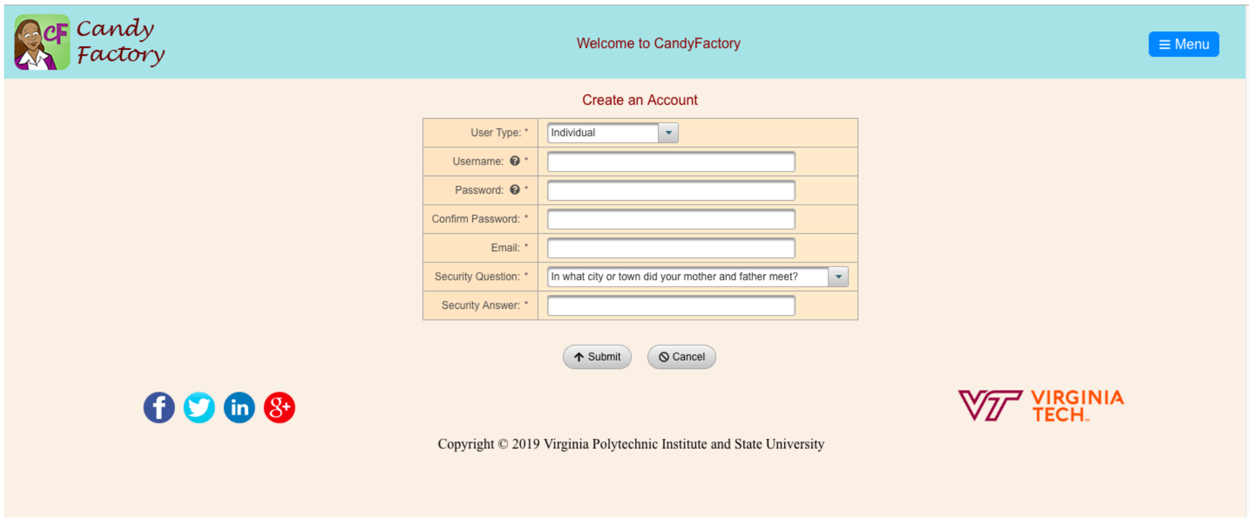
Figure 24. UML Diagram of Managers

Chapter 5: CandyFactory Functionality

This chapter describes the functionality of CandyFactory. There are three user types: teacher, student, and individual. Student and individual types perform almost the same functionalities; teacher type performs a different set of functionalities. The following sections presents functionalities for these user types.

5.1 User Registration

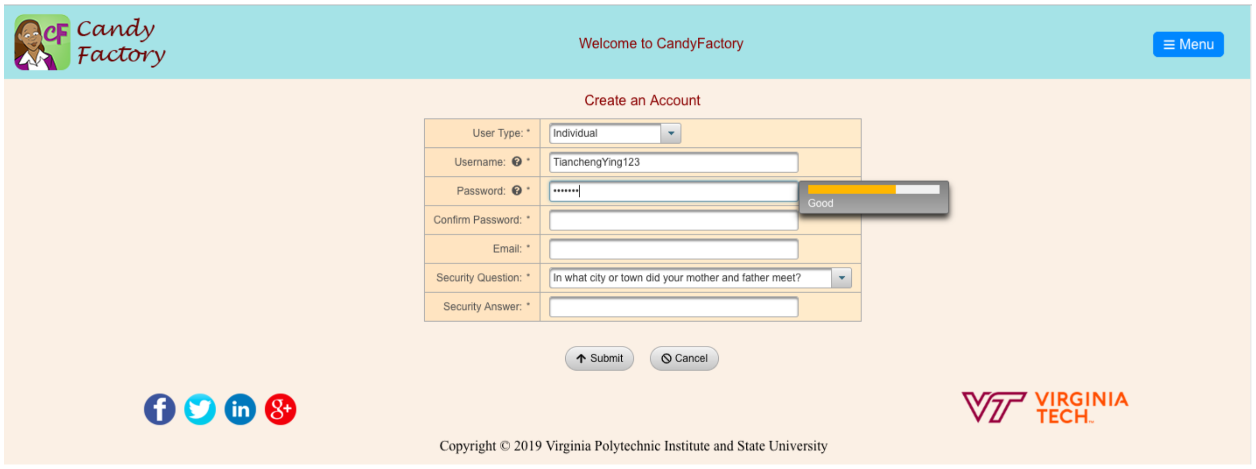
A regular user can register to be a teacher or an individual. An individual type of user can play game and generate report for each level. A teacher can create classes and students, and generate class reports and individual student reports. A user needs to provide the following required information: user type which includes "Individual" and "Teacher", username, password, confirm password, email, security question, and security answer (Figure 25). The username needs to contain 6 to 32 characters with capital letter, lowercase letter, number or special character. The password needs to contain 8 to 32 characters with at least 1 capital letter, 1 lowercase letter, 1 number, and 1 special characters. The email must meet a valid email format. If not, the registration page gives an email format error (Figure 26). The security question is chosen from a list, which is used for password modification.



User Type: *	Individual
Username: *	
Password: *	
Confirm Password: *	
Email: *	
Security Question: *	In what city or town did your mother and father meet?
Security Answer: *	

Copyright © 2019 Virginia Polytechnic Institute and State University

Figure 25. User Registration Interface



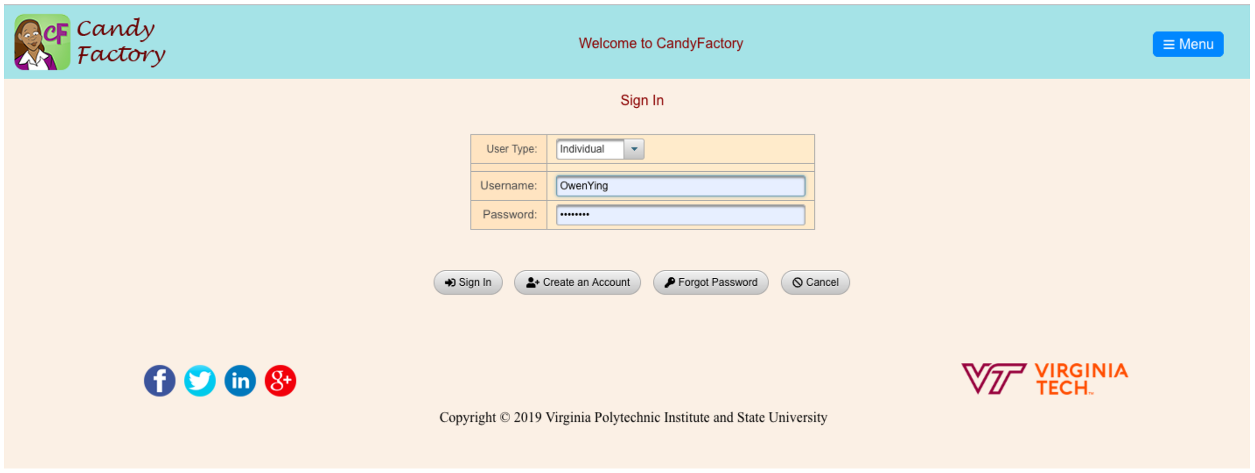
User Type: *	Individual
Username: *	TianchengYing123
Password: *	•••••••
Confirm Password: *	
Email: *	
Security Question: *	In what city or town did your mother and father meet?
Security Answer: *	

Good
Submit
Cancel
VIRGINIA TECH.
Copyright © 2019 Virginia Polytechnic Institute and State University

Figure 26. Password and Email Validator

5.2 User Login

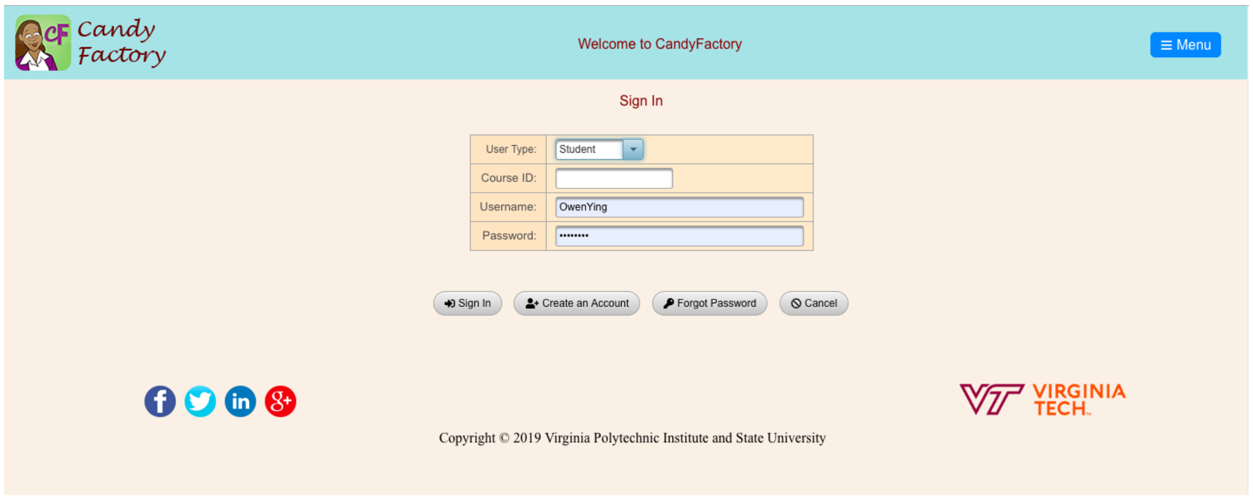
The user login functionality contains user type, username, and password. With different user types, the login page's UI is not the same. For student type, student needs to enter his/her course id in order to be located (Figure 28). For teacher and individual type, only the username and password are needed (Figure 27).



User Type:	Individual
Username:	OwenYing
Password:	••••••••

Sign In
Create an Account
Forgot Password
Cancel
VIRGINIA TECH.
Copyright © 2019 Virginia Polytechnic Institute and State University

Figure 27. Individual and Teacher Login Interface



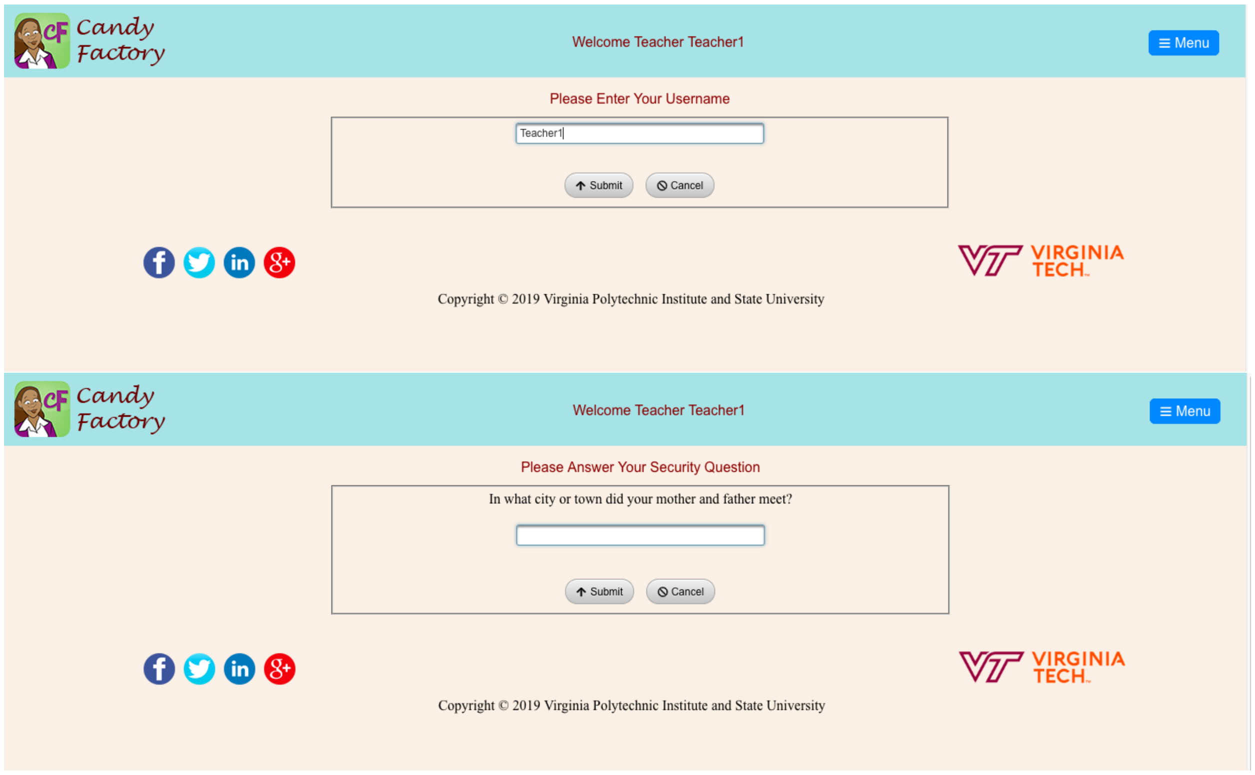
User Type:	Student
Course ID:	
Username:	OwenYing
Password:	••••••••

Copyright © 2019 Virginia Polytechnic Institute and State University

Figure 28. Student Login Interface

5.3 Password Modification

The password modification functionality contains three steps. Firstly, the user needs to input the logged in username. Secondly, the user needs to answer the security question of the logged in account. The security question is defined in user registration process. Lastly, the user needs to input the new password and confirm password. (Figure 29)



Teacher1

Copyright © 2019 Virginia Polytechnic Institute and State University

In what city or town did your mother and father meet?

Copyright © 2019 Virginia Polytechnic Institute and State University

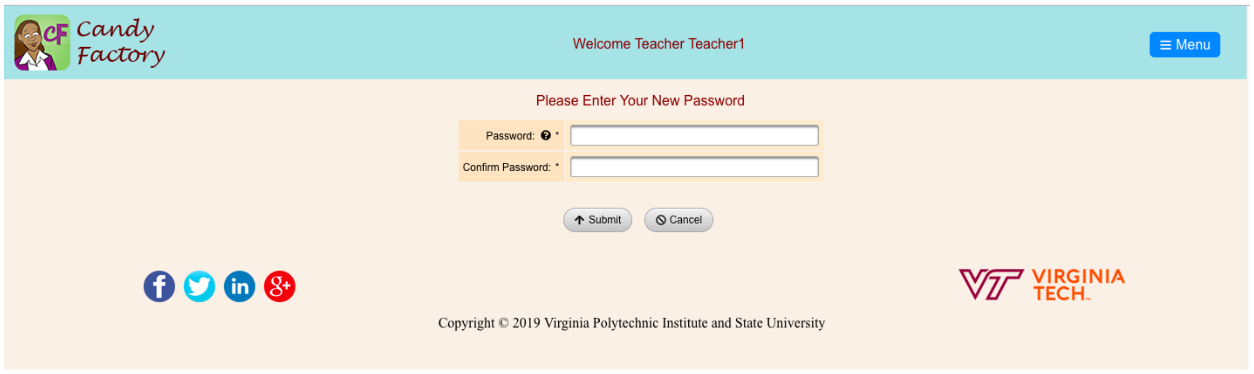
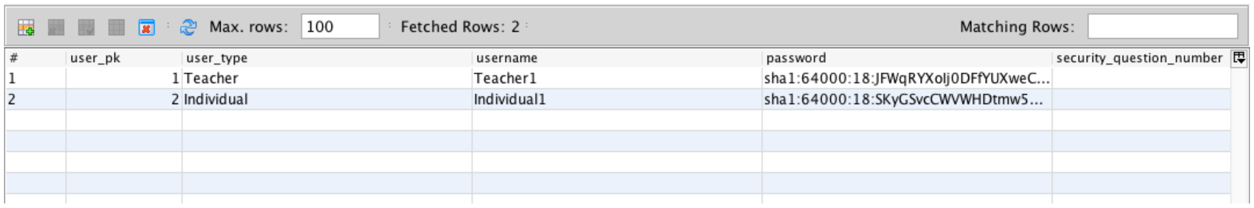


Figure 29. Change Password Interfaces

5.4 Password Encryption

CandyFactory has a very secure mechanism to protect the user's confidential information. For the user's password, in order to protect it from either malwares or CandyFactory developers and administrators, we use an industry standard encryption technique to encrypt all the passwords. Firstly, we generate a random 24-bits salt. Secondly, based on the salt and the user's password, we generate a binary hash using PBKDF2 [Wikipedia 2019] method. Lastly, we append "sha1:", the pre-defined iteration number, hash size, salt, and binary hash together to get the final hash string of the user's password. In this case, nobody can get the password based on the data stored in the MySQL database. Figure 30 shows the data stored in MySQL database. For the password field, we only store the encrypted password's hash instead of the real password. Therefore, we ensure the security of the user's confidential information.



#	user_pk	user_type	username	password	security_question_number
1	1	Teacher	Teacher1	sha1:64000:18:JFWqRYXolj0DFfYUXweC...	
2	2	Individual	Individual1	sha1:64000:18:SKyGSvcCWVWHDtmw5...	

Figure 30. User Account Information in Database Table

5.5 Game Play

Cloud-based CandyFactory strongly support different kinds of fractions. By selecting, measuring, slicing, and copying of the candies, students learn the fractions by intuitively produce the manufactured candy. With the achievement system, students can be encouraged to attain better performance, thus to learn faster with more fun. CandyFactory game levels are introduced in Description of Levels on the iPad version.

Cloud-based CandyFactory game, based on the iPad version, adds new features and animations. Also with the backend's support, cloud-based CandyFactory is more powerful from all aspects (Figure 31).

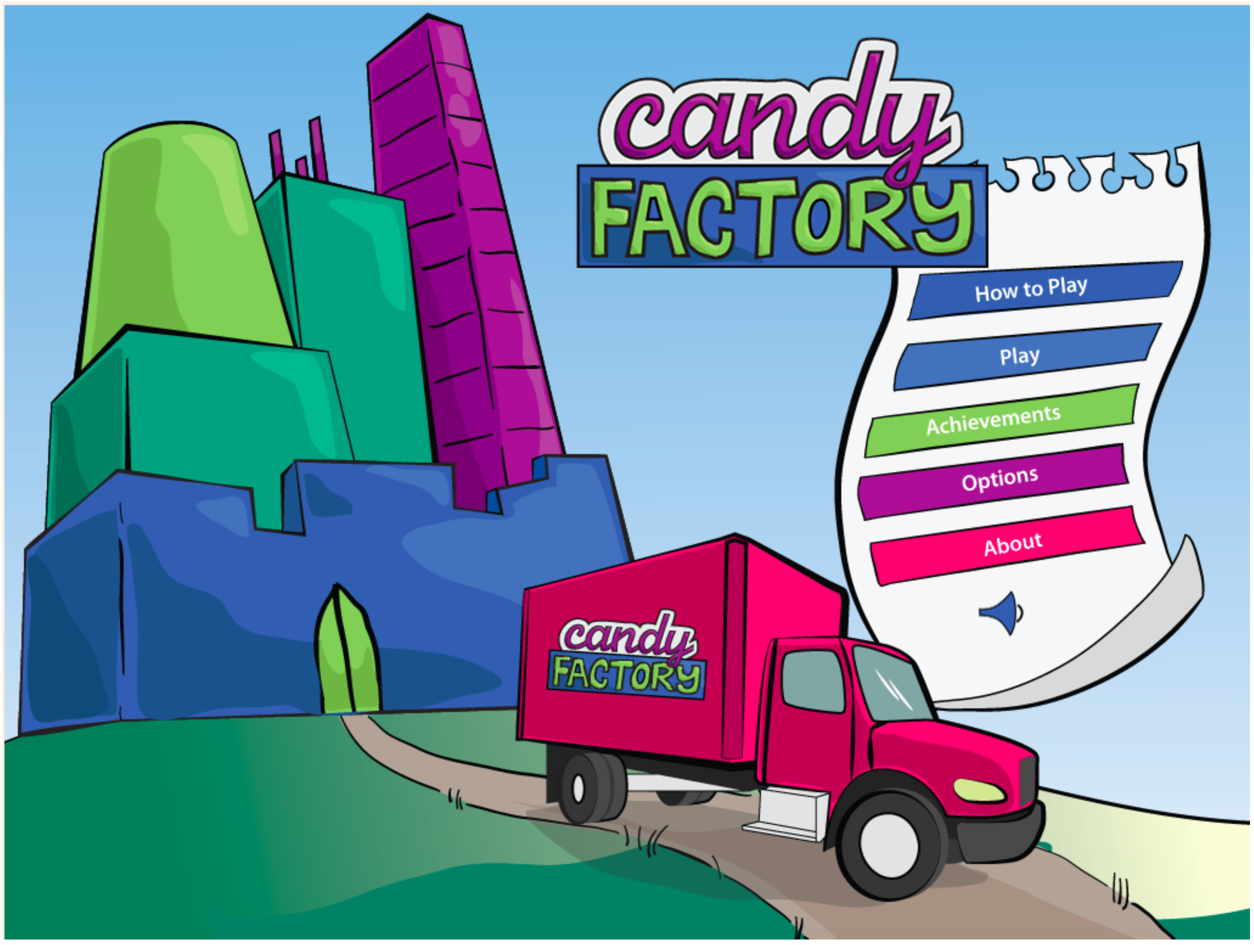


Figure 31. Cloud-based CandyFactory Welcome Page

5.5.1 Tutorial Mode

The tutorial mode contains five levels. Each level corresponds to a level in gameplay mode. In the level selection page (Figure 32), users can choose any level to learn how to play the game. The title "To Learn How To Play" indicates the tutorial mode. During tutorial mode, the timer is off, customer satisfaction is not calculated, and no data are collected. Consequently, this mode is a pure learning mode.

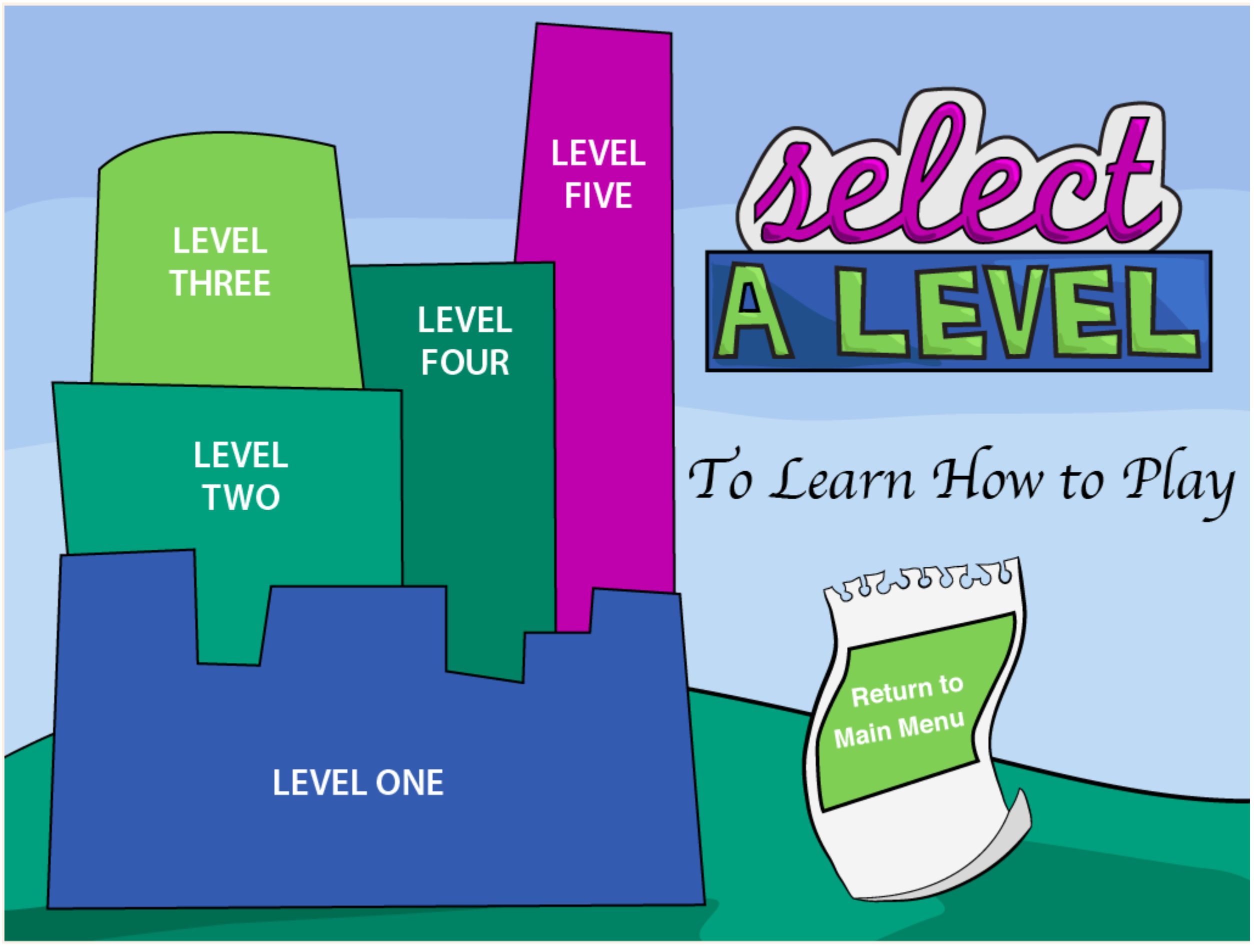
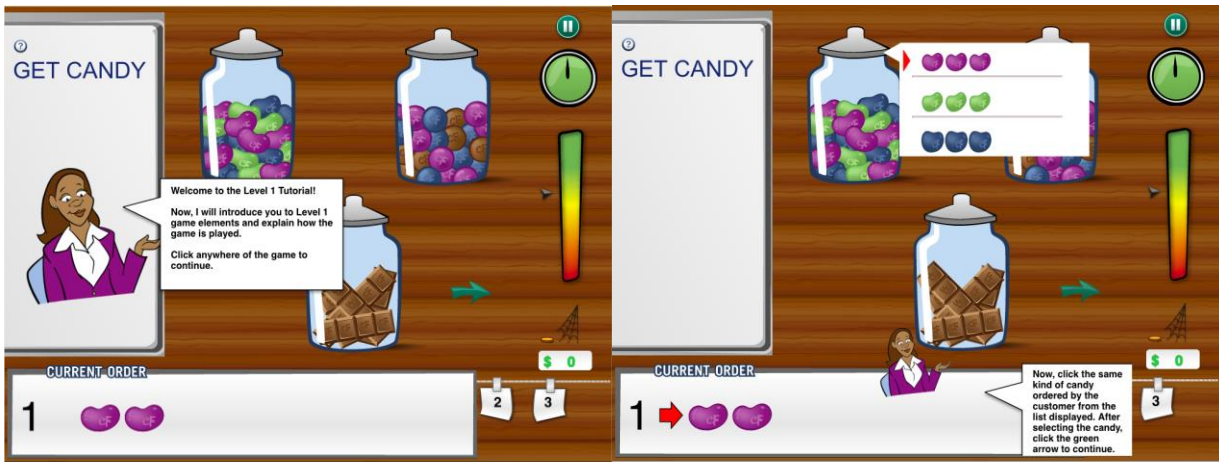


Figure 32. Tutorial Mode Level Selection

5.5.1.1 Scene 1

Tutorial mode Scene 1 teaches users to learn how to play CandyFactory with discrete candies (Figure 33). In this scene, the Candy Manager teaches users basic components' functionalities, such as clock, satisfaction bar, and pause button. Also in this scene, the Candy Manager guides users to choose proper candy type to manufacture on, in order to fulfill customer orders (Figure 33**).** The Candy Manager's guidance is implemented established on JQuery animation. With the animations, the guidance is more clear and intuitive.



Welcome to the Level 1 Tutorial!

Now, I will introduce you to Level 1 game elements and explain how the game is played.

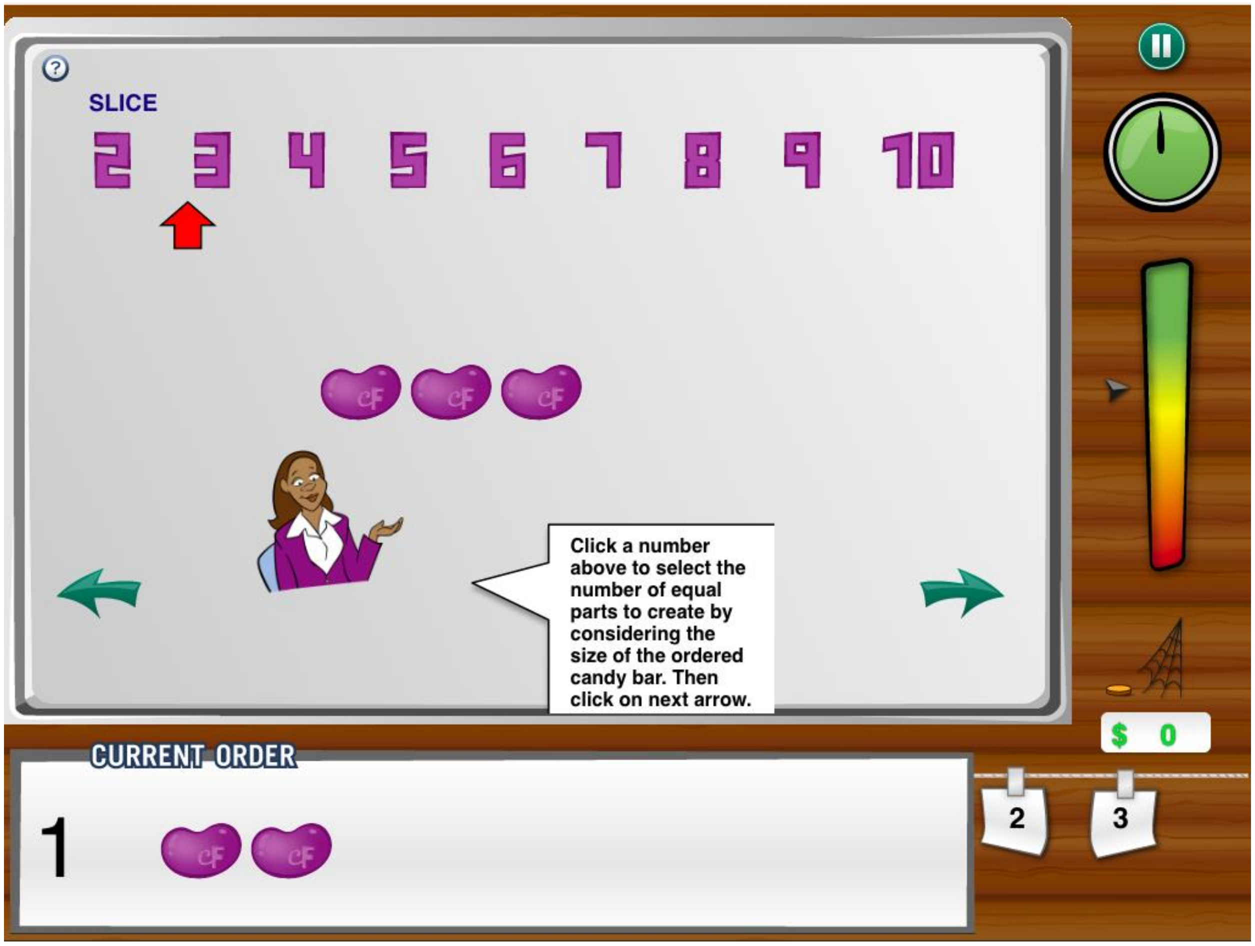
Click anywhere of the game to continue.

Now, click the same kind of candy ordered by the customer from the list displayed. After selecting the candy, click the green arrow to continue.

Figure 33. Tutorial Mode Scene 1

5.5.1.2 Scene 2

This scene is partition scene. Users are supposed to slice the whole candy into partitions in order to iterate and copy to make the customer candy. In this scene, an animated arrow floats back-and-forth to indicate the partition the user can make. Also with the Candy Manager's explanation dialog, the user can easily find out what he/she is supposed to do (Figure 34). After selecting a partition, the Candy Manager will guide the user to the next scene.

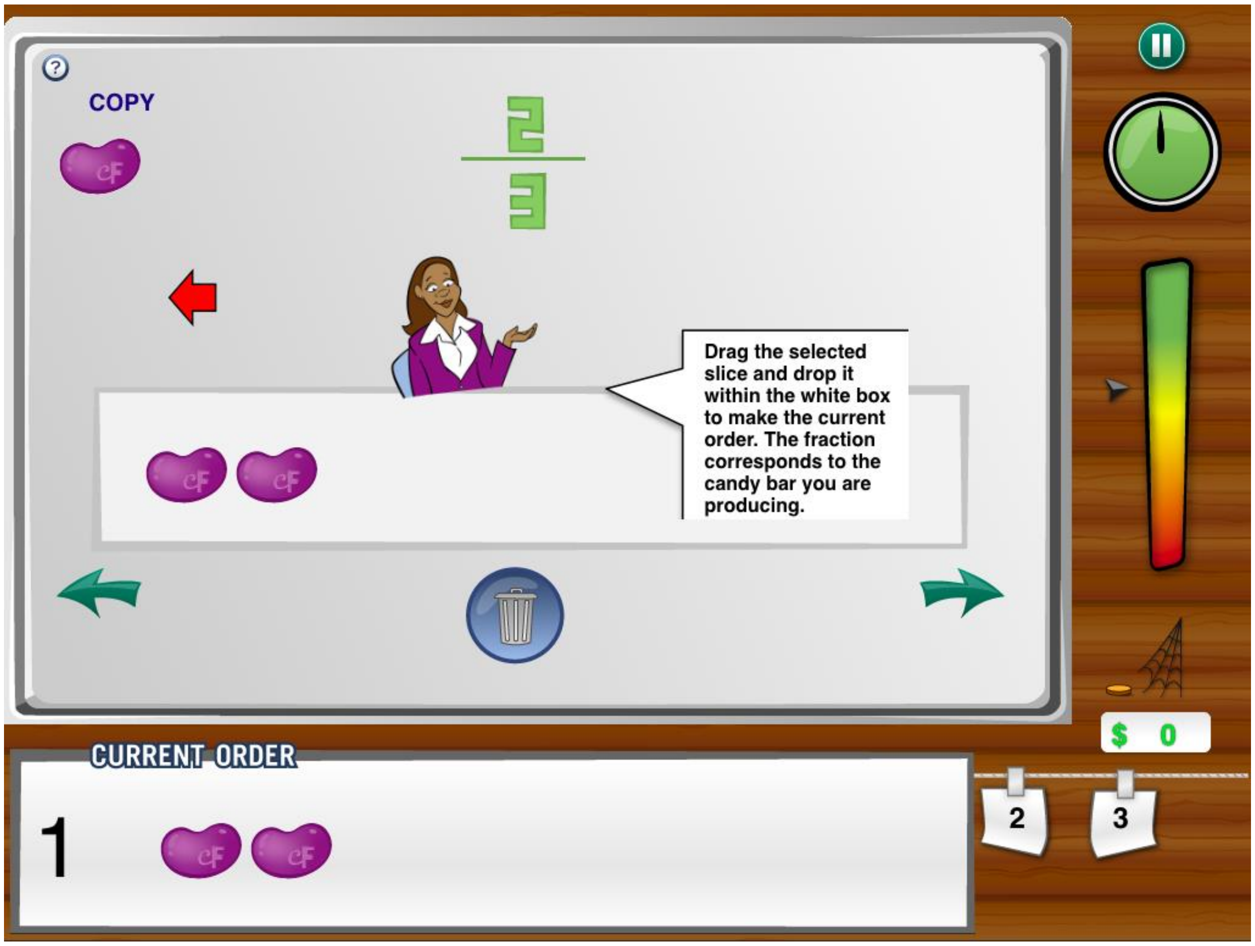


Click a number above to select the number of equal parts to create by considering the size of the ordered candy bar. Then click on next arrow.

Figure 34. Tutorial Mode Scene 2

5.5.1.3 Scene 3

Scene 3 is iteration scene. Users are supposed to drag-and-drop the unit candy to iterate *n* times in order to form customer candy. With the animated instruction arrow and Candy Manager, users can easily learn how to iteratively copy and delete the unit candy (Figure 35).



Drag the selected slice and drop it within the white box to make the current order. The fraction corresponds to the candy bar you are producing.

Figure 35. Tutorial Mode Scene 3

5.5.1.4 Scene 4

Scene 4 is measurement and shipment scene. With an animated ruler, users can measure the length of manufactured candy bar and customer order to compare whether they fulfill the customer's request. With back-up arrow, users can go to previous scenes anytime to modify the candy bar to meet the customer order. After measurement, users can ship the order to the customer and one game round is finished (Figure 36).

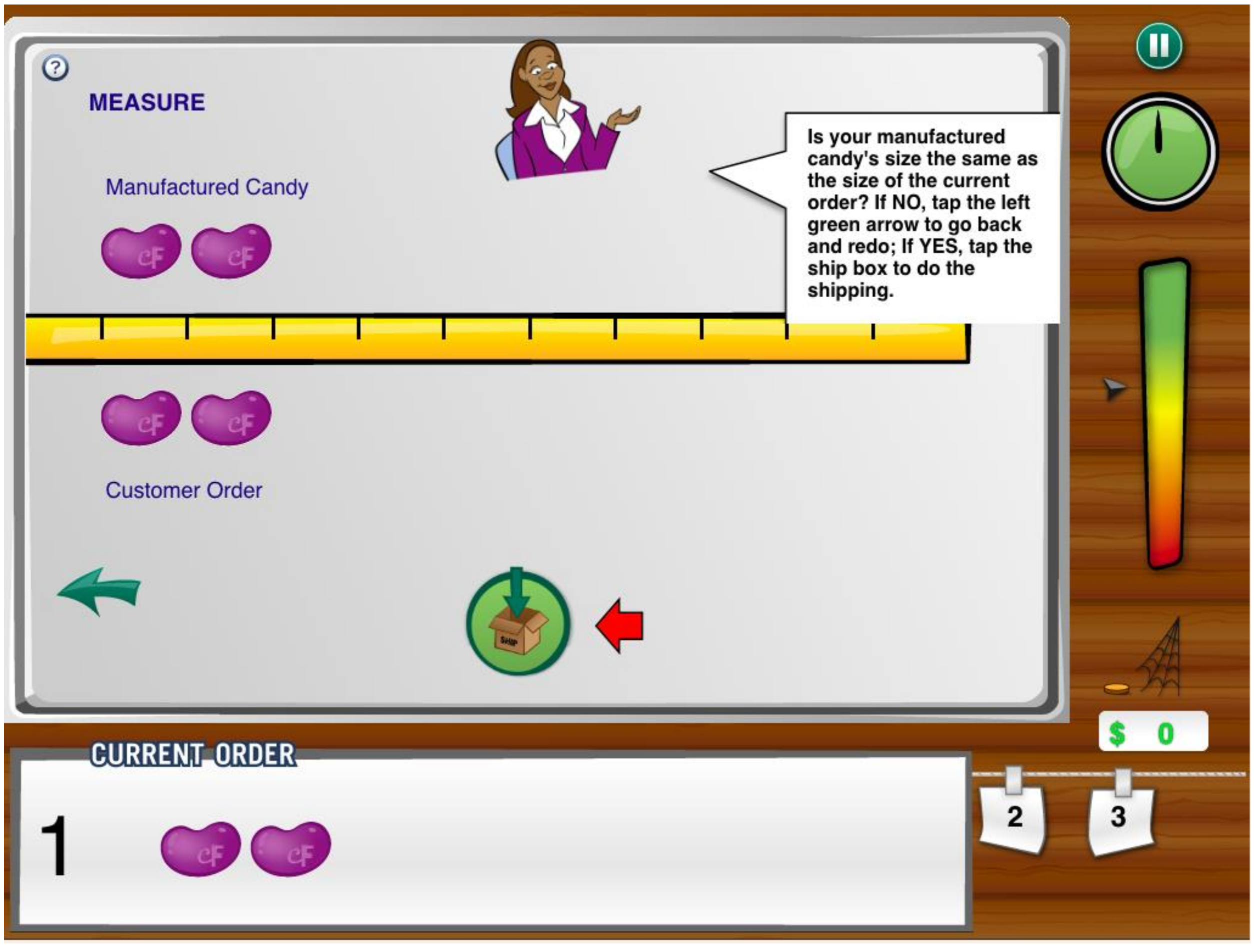

MEASURE
Manufactured Candy
Customer Order
Is your manufactured candy's size the same as the size of the current order? If NO, tap the left green arrow to go back and redo; If YES, tap the ship box to do the shipping.
CURRENT ORDER
1
2
3
$ 0

Figure 36. Tutorial Mode Scene 4

5.5.2 Game Mode

The game mode provides fun games to the users with attractive animations, timing, and achievement system. Based on the iPad version of CandyFactory, we add several new features to the game to provide better game experience and learning efficiency. The overall gaming process has been introduced in Tutorial Mode**,** thus in this section, we emphasize on the new features which the iPad version does not have.

5.5.2.1 Continuous Candy Ruler

In cloud-based CandyFactory, we add an adaptive ruler to the continuous candy bar to help the users identify the length of the candy bar. For different size of continuous candy bar, users may face difficulty measuring its length. Most users just eye balling the length, which makes it hard to fulfill the customer’s orders. Moreover, the candy measuring method is not a part of the factor to accelerate the learning rate. Consequently, we decide to add a ruler to help the users identify the length of the candy bar (Figure 37). Each small grid on the ruler represents a millimeter while a big grid represents a centimeter.

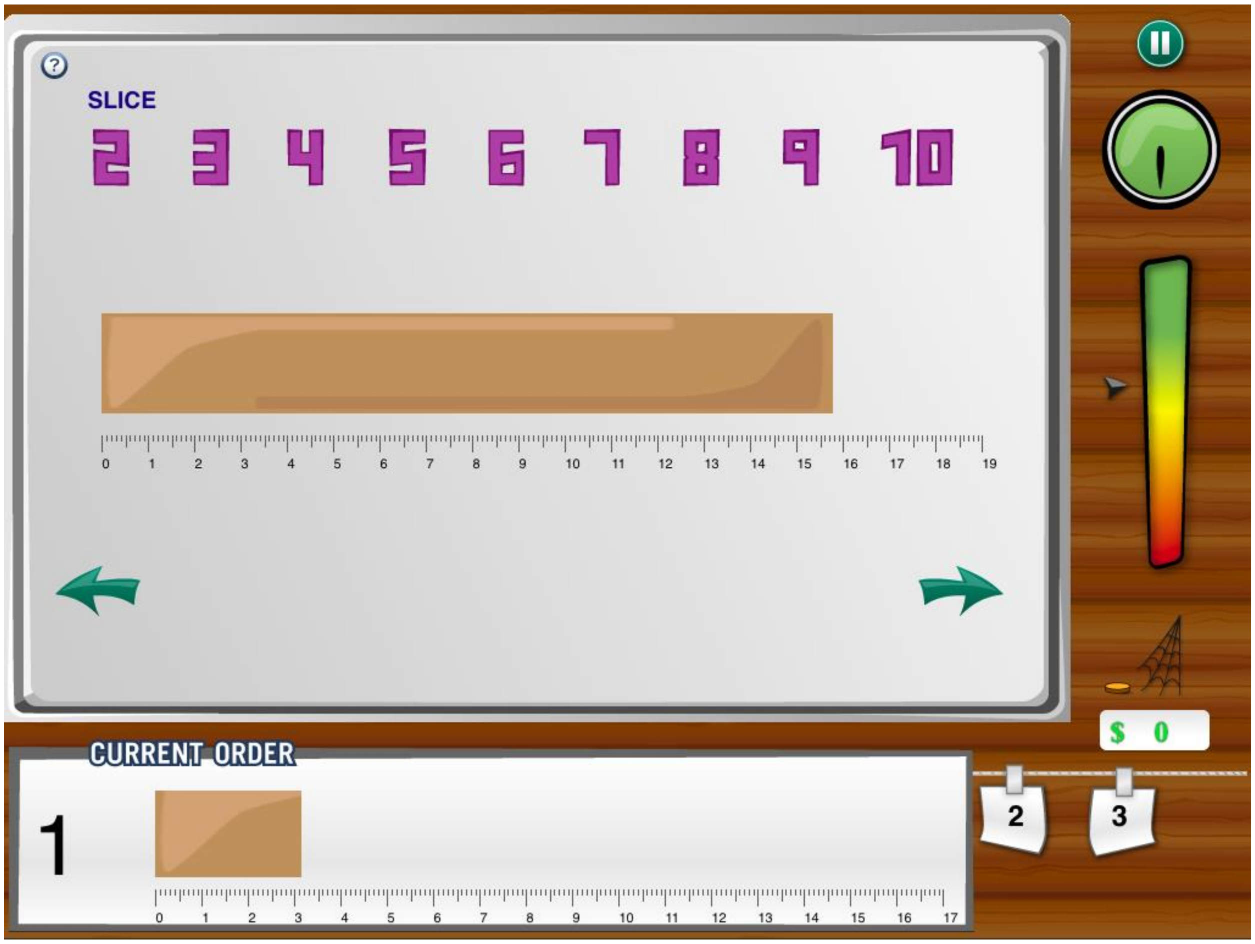


Figure 37. Ruler of Continuous Candy Bar

5.5.2.2 Animations

Based on the iPad CandyFactory, we add new animations to make this game more fun and attractive, such as achievement medal animation and shipment animation. As Figure 38 shown below, after each game round, users can get bonus and achievements. With the bonus and achievements animation, users are encouraged to achieve better performance. Also with reproducing the iPad CandyFactory's animations, we keep and improve the attractiveness to make users, especially students, to learn through fun better.

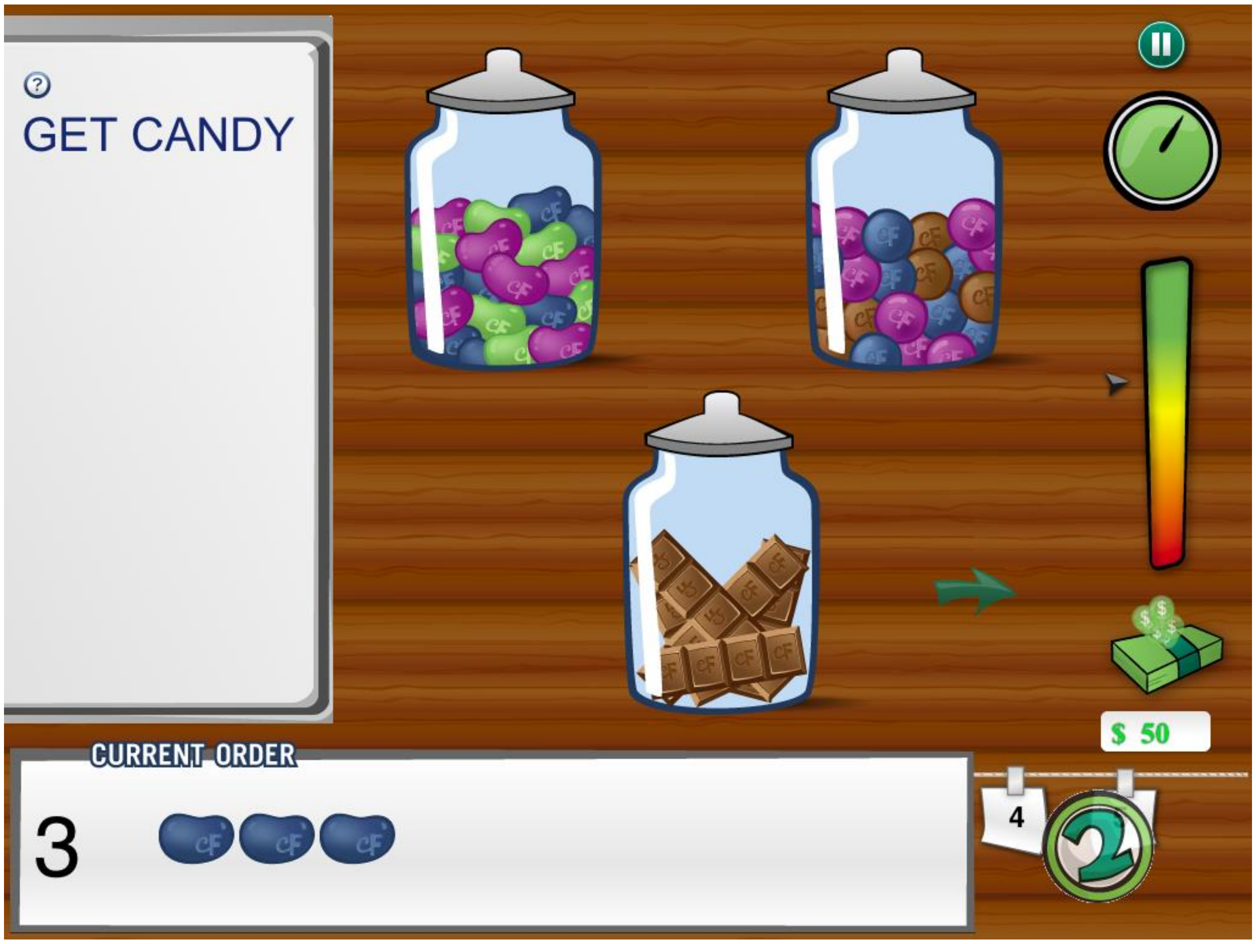


Figure 38. Animation Example

5.5.2.3 Achievement System

Based on the iPad CandyFactory, cloud-based CandyFactory develops a more complicated but attractive achievement system. Since in the iPad version, CandyFactory only needs to store the current user's achievements locally, thus the mechanism is much easier. For cloud-based CandyFactory, we need to store achievements under each corresponding user. We also need to identify the achievements after each play round and then merge with the specific user's existing achievements (Figure 39).

In order to do this efficiently, we come up with an optimization algorithm to store and retrieve the achievements introduces in Achievement System Design.

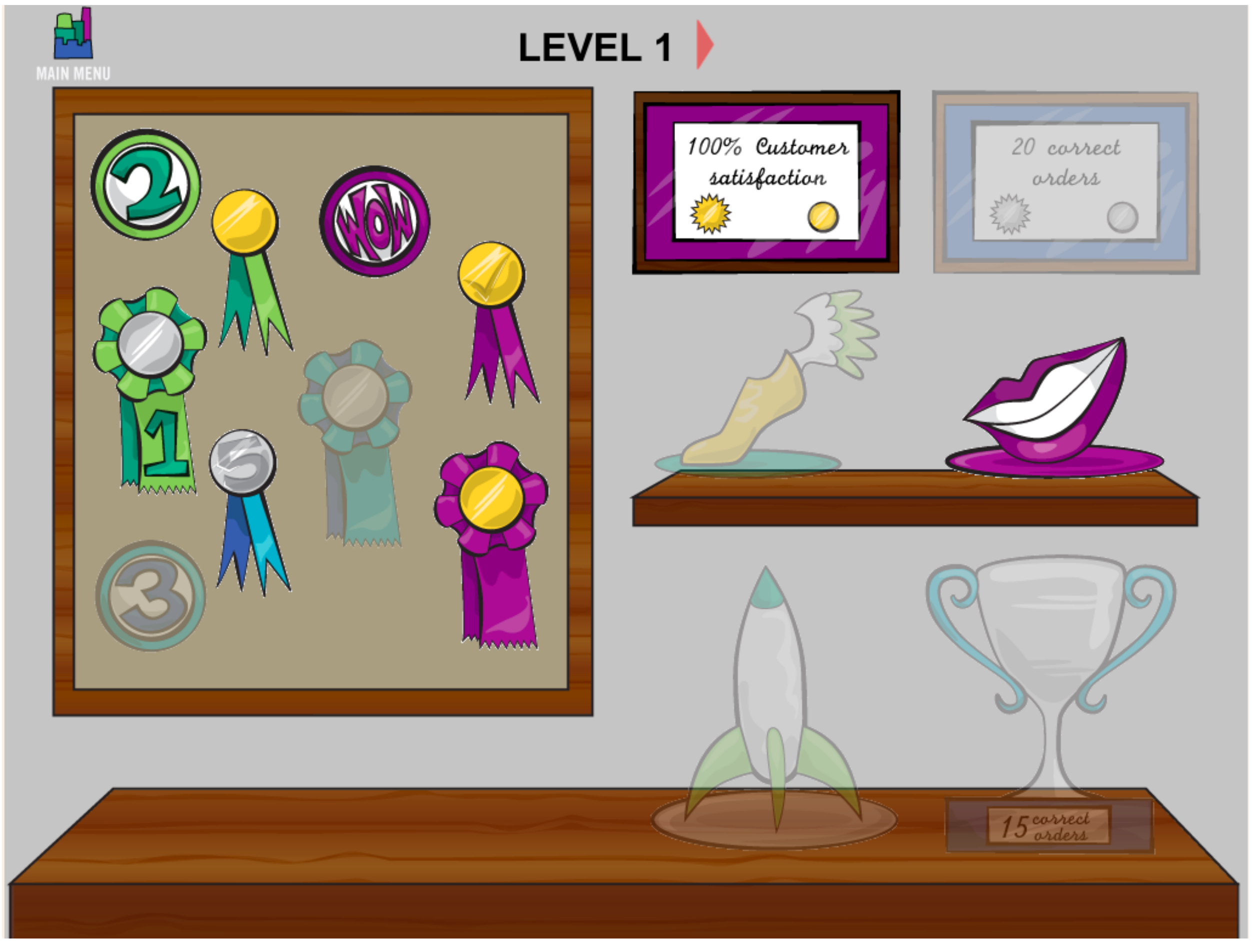


Figure 39. Achievement of Specific User

Also along with the achievement system, we provide the frontend with very attractive achievements animations. After each round, as long as the user has new achievement, the achievement medals will flow from left to right, to encourage the users to play better (Figure 40). Using the reward system, cloud based CandyFactory activates the users' learning passion and accelerates the users' learning process. Therefore, users are able to learn fractions through playing the game.

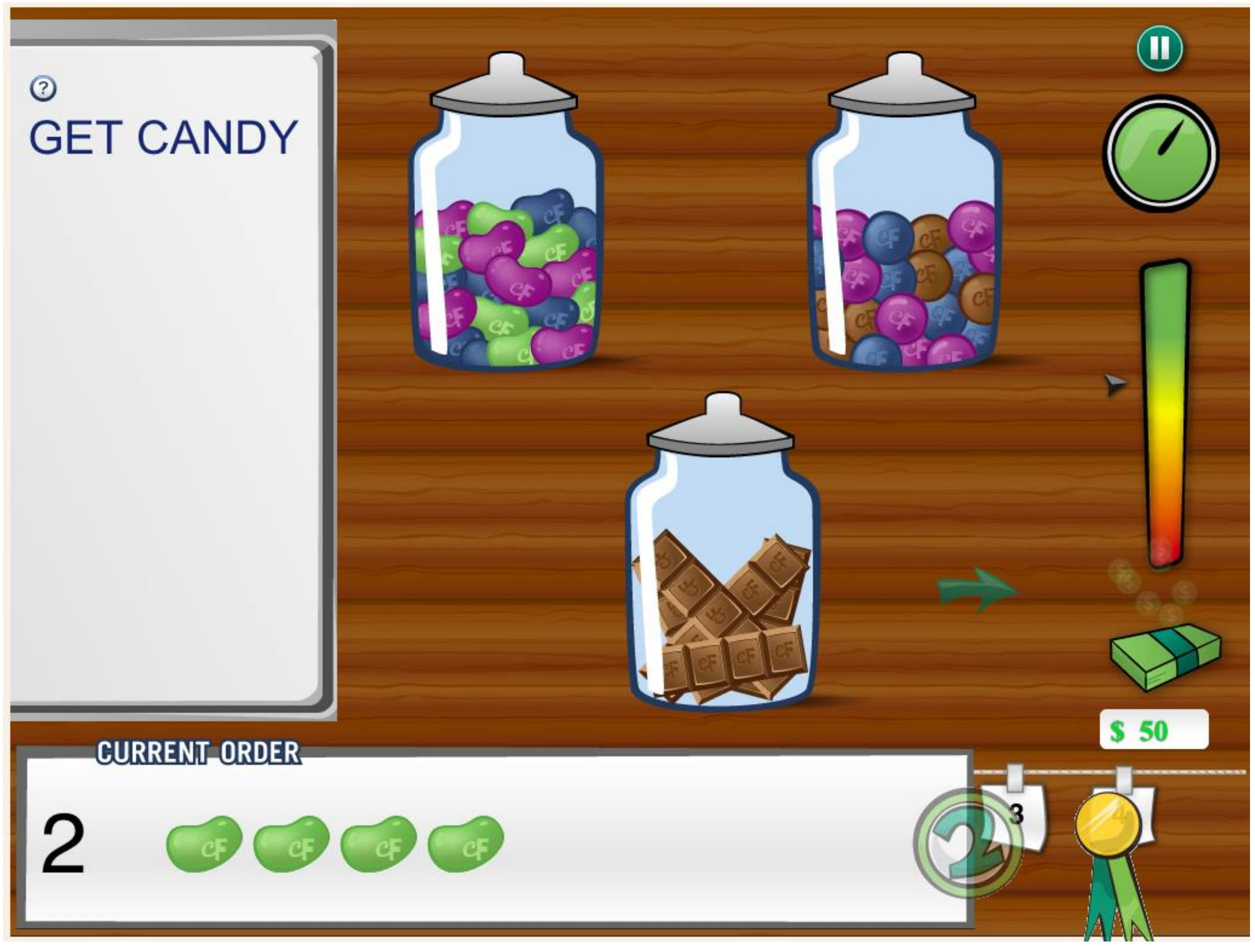
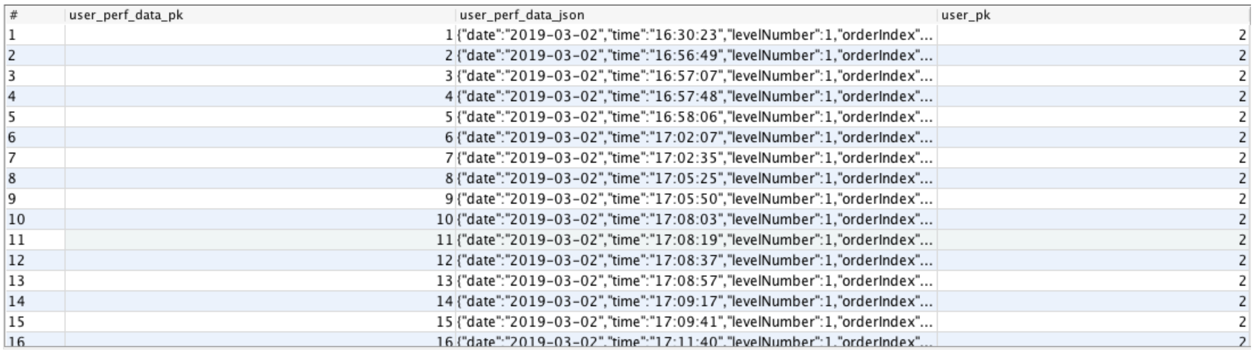


Figure 40. Achievement Animation Reward

5.5.2.4 Game Performance Data Transmission

Game performance transmission is a key feature of cloud-based CandyFactory. It is used to record user performance, generate user performance report, and generate course report. Thus, transmitting and storing them properly are very important.

Since the performance data format is flexible and tends to change, instead of using standard MySQL data table to store them, we use JSON to represent the performance as a flexible string to store in MySQL database. By using MySQL's standard storage plus JSON's flexible storage, cloud-based CandyFactory can retrieve and store standard data, such as account information, as well as mutable data, such as game performance data, efficiently. When requirement changes in the future, we can modify the performance JSON format to match the new requirement without redesigning all the MySQL database table (Figure 41).



#	user_perf_data_pk	user_perf_data_json	user_pk
1	1	{"date":"2019-03-02","time":"16:30:23","levelNumber":1,"orderIndex"...	2
2	2	{"date":"2019-03-02","time":"16:56:49","levelNumber":1,"orderIndex"...	2
3	3	{"date":"2019-03-02","time":"16:57:07","levelNumber":1,"orderIndex"...	2
4	4	{"date":"2019-03-02","time":"16:57:48","levelNumber":1,"orderIndex"...	2
5	5	{"date":"2019-03-02","time":"16:58:06","levelNumber":1,"orderIndex"...	2
6	6	{"date":"2019-03-02","time":"17:02:07","levelNumber":1,"orderIndex"...	2
7	7	{"date":"2019-03-02","time":"17:02:35","levelNumber":1,"orderIndex"...	2
8	8	{"date":"2019-03-02","time":"17:05:25","levelNumber":1,"orderIndex"...	2
9	9	{"date":"2019-03-02","time":"17:05:50","levelNumber":1,"orderIndex"...	2
10	10	{"date":"2019-03-02","time":"17:08:03","levelNumber":1,"orderIndex"...	2
11	11	{"date":"2019-03-02","time":"17:08:19","levelNumber":1,"orderIndex"...	2
12	12	{"date":"2019-03-02","time":"17:08:37","levelNumber":1,"orderIndex"...	2
13	13	{"date":"2019-03-02","time":"17:08:57","levelNumber":1,"orderIndex"...	2
14	14	{"date":"2019-03-02","time":"17:09:17","levelNumber":1,"orderIndex"...	2
15	15	{"date":"2019-03-02","time":"17:09:41","levelNumber":1,"orderIndex"...	2
16	16	{"date":"2019-03-02","time":"17:11:40","levelNumber":1,"orderIndex"...	2

Figure 41. Performance Data in Database Table

After each game round (Figure 42), when the user clicks on "Ship" button, the frontend JavaScript packs the performance data into a JSON object, then use a hidden form to send it to the backend. The backend, after receiving the JSON object, stores it as a JSON string into the database under the corresponding user. This process only contains JSON encoding.

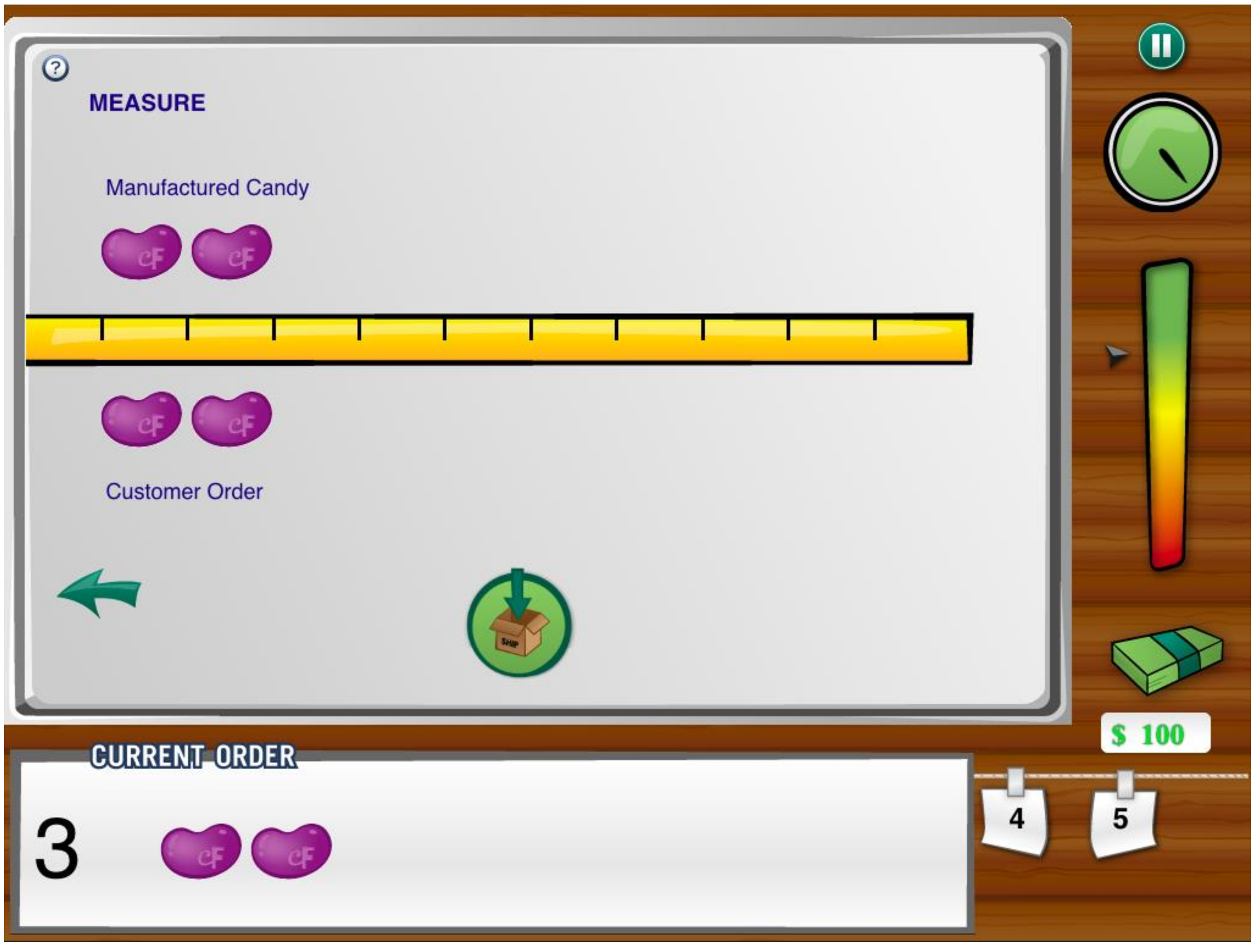


Figure 42. Ship Scene of CandyFactory Game

5.5.3 Shift Log

Shift log shows the final result of a whole gameplay of a user in a well-formatted animated way. In the shift log, users can know how are their customer satisfaction, bonus earned, performance overall, and each of the detailed orders. With the customer satisfaction's animation, users can see each game round's customer satisfaction's changes. Finally, after the animation is over, the result customer satisfaction is displayed (Figure 43).

The customer satisfaction is calculated by user points. When a user ships a correct order, the user points are increased by 50. Otherwise, the user points are decreased by 50. With a mapping algorithm inside the frontend, we calculate the final customer satisfaction of each game round and show the animation as well as store the satisfactions.

When game is over, the frontend JavaScript packs individual round of game data into JSON and store it into session map. After redirecting to shift log page, the shift log retrieves the JSON package from session map and parse the JSON array into readable data then transform it into more a intuitive animated format, which is shown in Figure 43.

Using this mechanism, for shift log display, the frontend doesn't have any interaction with the backend, which dramatically increases the display's speed and efficiency. Also by leveraging this mechanism, with the calculation and data transmission within the frontend, we largely save the server's calculation and storage resources. Thus we achieve large scale deployment of the whole system.

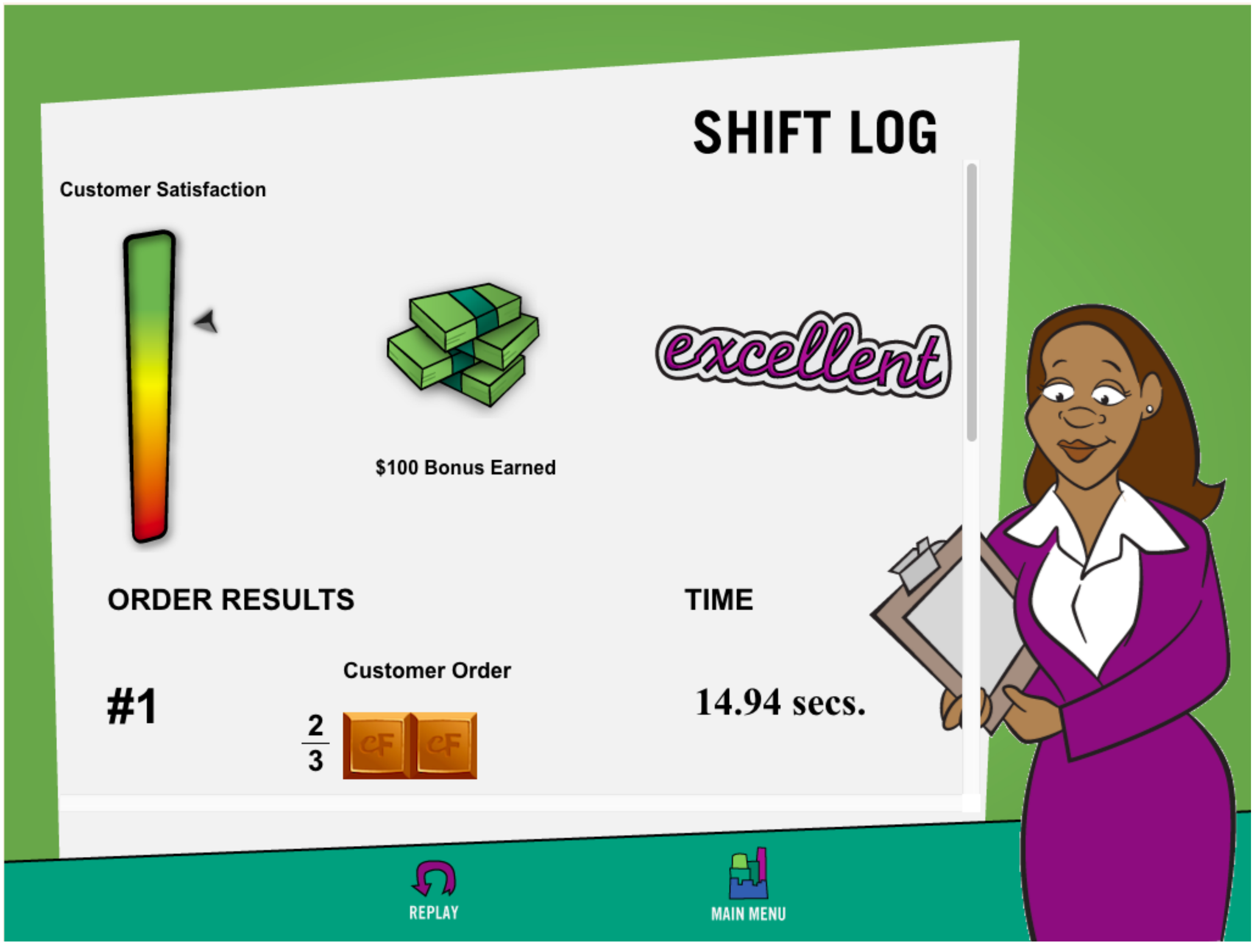


Figure 43. CandyFactory Game Shift Log

5.5.4 Options

Options provides users abilities to choose the timer from three to nine minutes (Figure 44). Also we allow users to turn off the timer while playing the game. With this functionality, the pace users want to play their game is up to them. This options can only be applied to Game Mode. The Tutorial Mode's timer is always off.

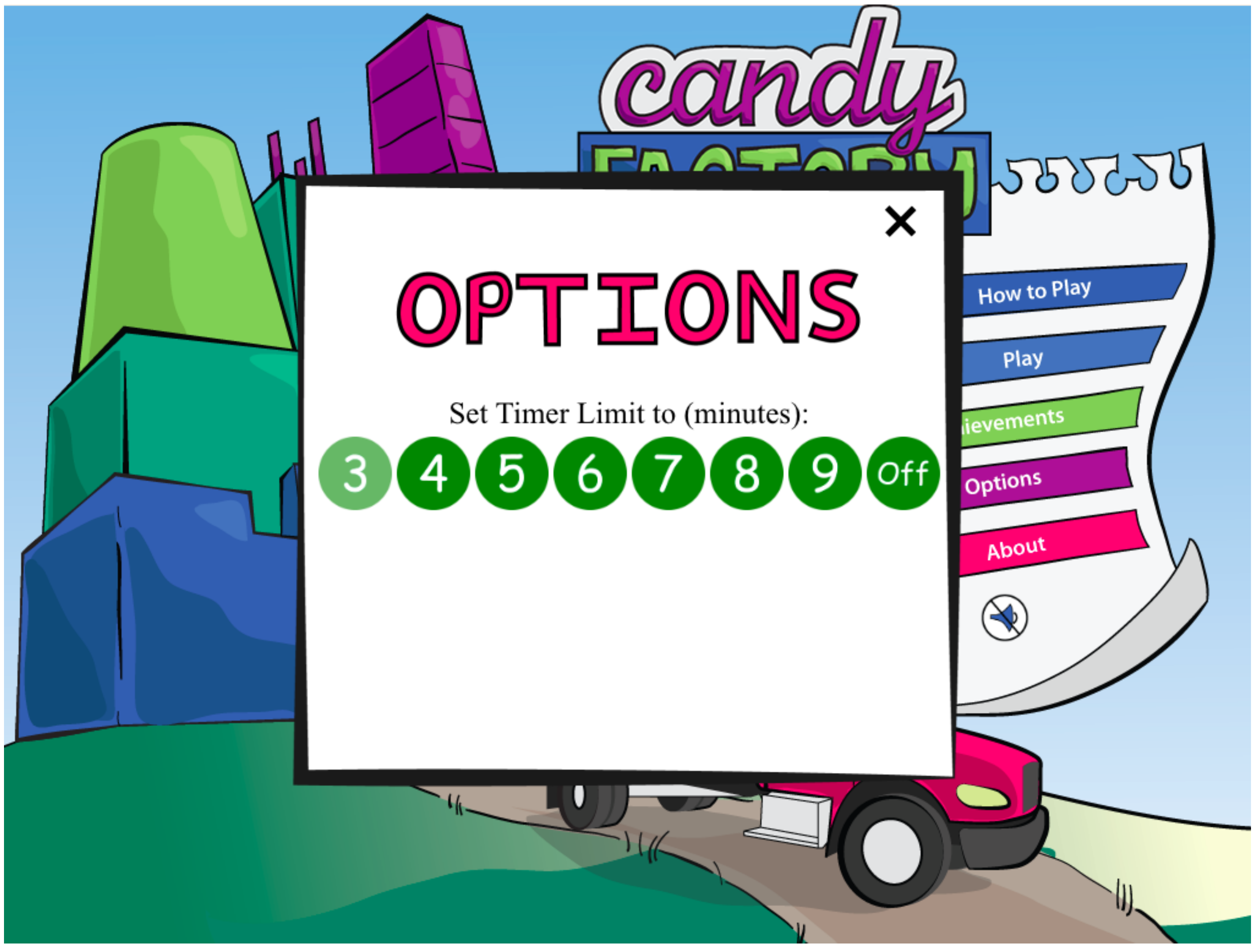
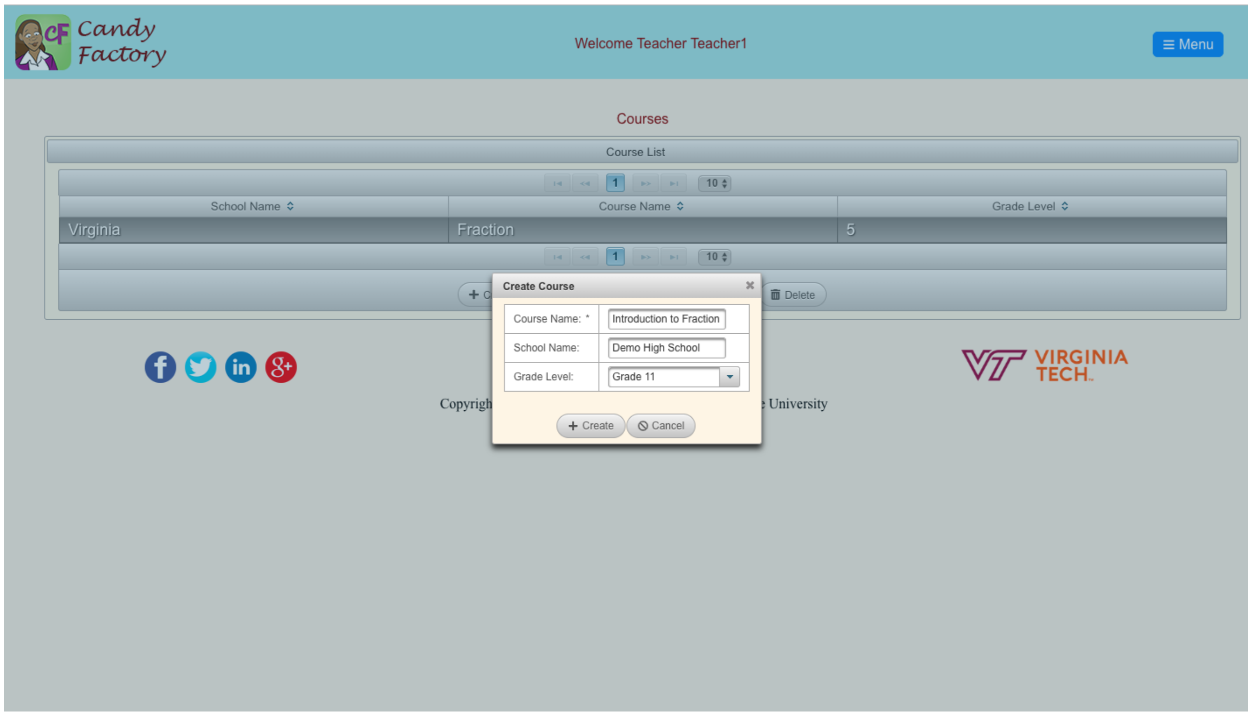


Figure 44. Option Menu of CandyFactory Game

5.6 Course Creation and Deletion

Course creation feature is provided to teachers, which allows them to create courses of different classes. In the course panel, after clicking on the create course button, a creation dialog shows up. By filling “Course Name”, “School Name”, and “Grade Level”, teachers are able to create courses for their classes (Figure 45). “Course Name” cannot be empty, since it is a vital information of the course. “School Name” is optional, which can protect the privacy if the teacher does not want to fill this field. “Grade Level” is a slide menu contains Grad one to Grad twelve.

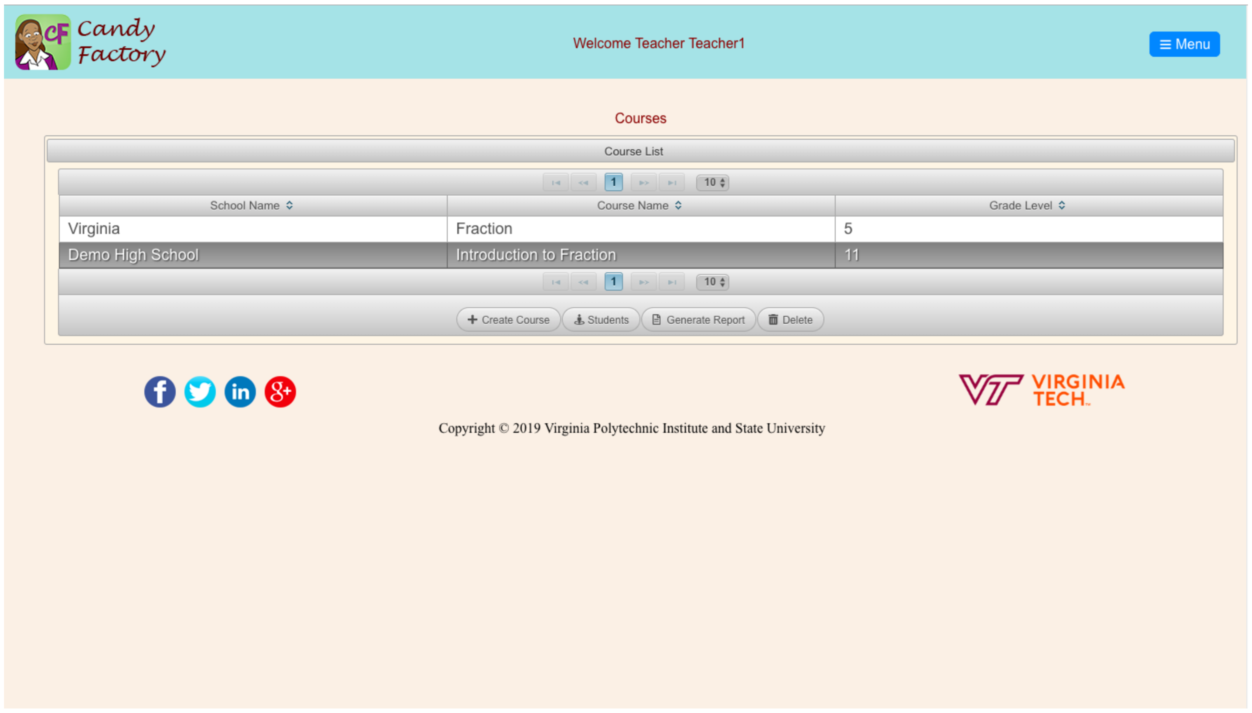


School Name	Course Name	Grade Level
Virginia	Fraction	5

Course Name: *	Introduction to Fraction
School Name:	Demo High School
Grade Level:	Grade 11

Figure 45. Course Creation Information Dialog

After clicking on the "Create" button, a course is created with the given information (Figure 46).

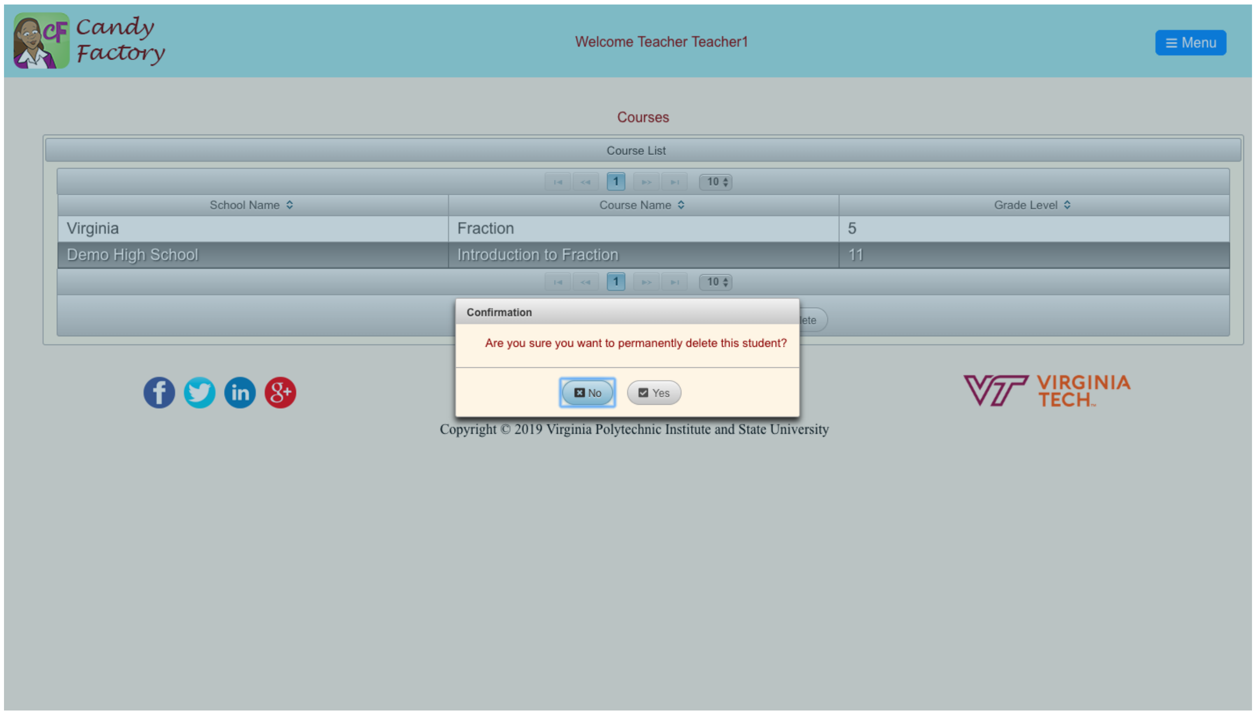


School Name	Course Name	Grade Level
Virginia	Fraction	5
Demo High School	Introduction to Fraction	11

Copyright © 2019 Virginia Polytechnic Institute and State University

Figure 46. Course Successfully Created with Given Information

By selecting a course and clicking on the "Delete" button, a course is deleted and all of the students and students' performance records under this course are deleted as well (Figure 47).


Welcome Teacher Teacher1

Courses

Course List

School Name	Course Name	Grade Level
Virginia	Fraction	5
Demo High School	Introduction to Fraction	11

Confirmation

Are you sure you want to permanently delete this student?

Copyright © 2019 Virginia Polytechnic Institute and State University

Figure 47. Course Deletion Confirmation Dialog

5.7 Student Account Creation and Deletion

Student account creation and deletion provide teachers ability to create accounts on the students' behalf. Considering of the students' age and the system's complexity, we decide to let the teachers create the students' accounts and distribute the accounts to the students. Due to privacy policies FERPA [U.S. Department of Education], students' performance should not be related to students' information. Thus, by letting teachers create students accounts, we can satisfy all the requirements.

Cloud-based CandyFactory has automated student account generation algorithm, which can assist teachers to create student accounts. With this algorithm, student creation is greatly simplified for the teachers. The only thing teachers need to do is to input the student number under that course (Figure 48). After clicking on the "Create" button, the backend system automatically creates students' accounts with very easy usernames and random 6-digit passwords. Meantime, in order to make the students accounts easier to remember and more attractive, we randomly assign animal images to the accounts as their profile images.

This functionality also provides student list exportation as Excel or PDF to let the teacher print out the student list and assign the student with a student account.

The process of creating the student accounts is as follows:

1. Usernames start from 1 and increase by 1 for each new account. We decide to use the auto-increased number as usernames because it is easy to be memorized and does not conflict with the privacy policy since people cannot tell who does this account belong to only by knowing the username. Thus, we largely protect the students' privacy.

2. The password is a randomly generated 6-digit number. We decide to use a 6-digit number to simplify the memorization of the password, which is student friendly.

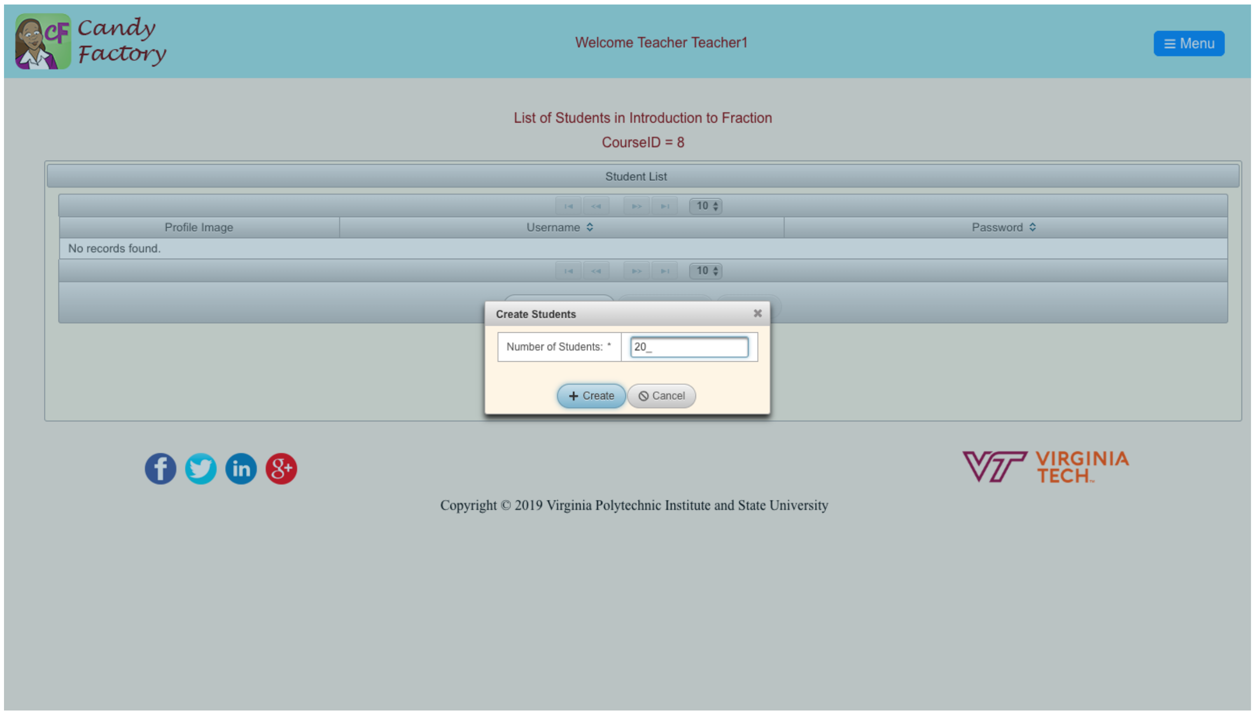
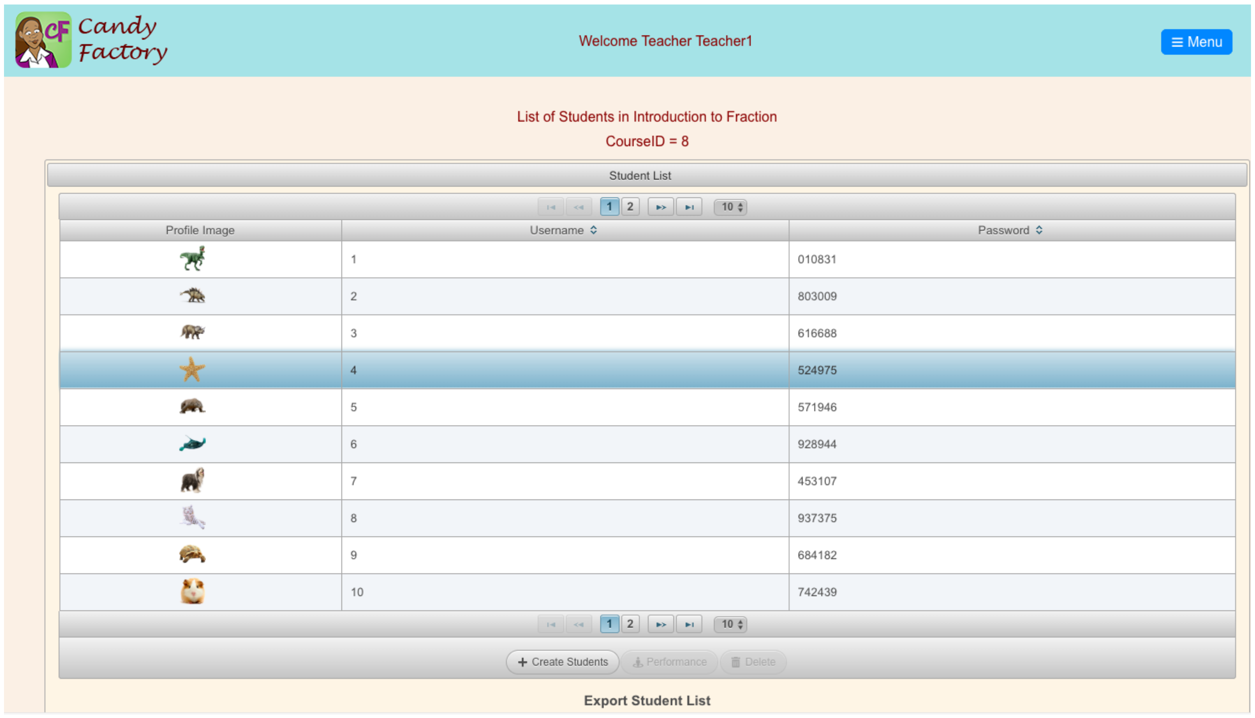


Figure 48. Input Student Number in the Course

For each account, the backend also randomly generates an animal's image as the student account's profile image. There are 100 pre-defined animal pictures which can be randomly assigned to each account (Figure 49). With the use of "HashSet", we prevent duplicates of the images to our best.

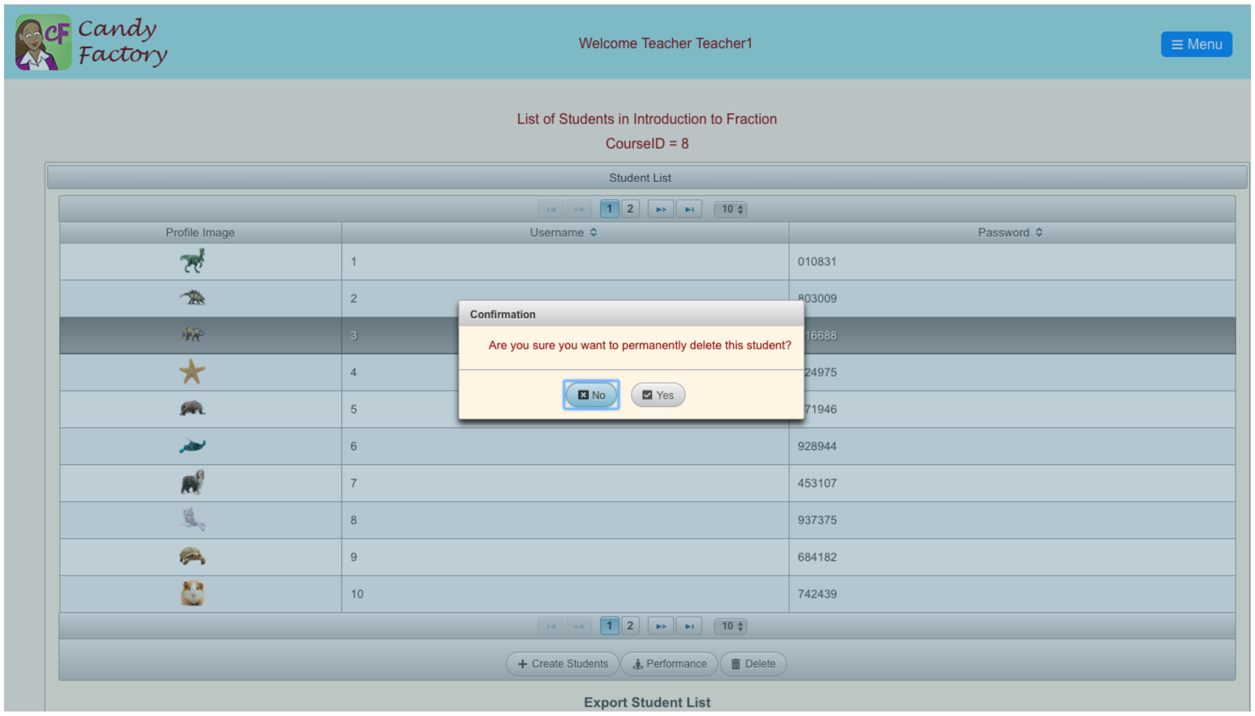


List of Students in Introduction to Fraction
CourseID = 8

Profile Image	Username	Password
	1	010831
	2	803009
	3	616688
	4	524975
	5	571946
	6	928944
	7	453107
	8	937375
	9	684182
	10	742439

Figure 49. Randomly Assigned Animal Profile Images

For student account deletion, the teacher can select a student then click on the "Delete" button, which deletes the student account and all the performance records under the student (Figure 50).



List of Students in Introduction to Fraction
CourseID = 8

Are you sure you want to permanently delete this student?

Figure 50. Student Deletion Dialog

5.8 Report Generation

Report generation is a key feature of cloud-based CandyFactory. This functionality takes use of the recorded user performance to generate a performance report to let the users understand better of their own performance as well as how well they have learned fractions.

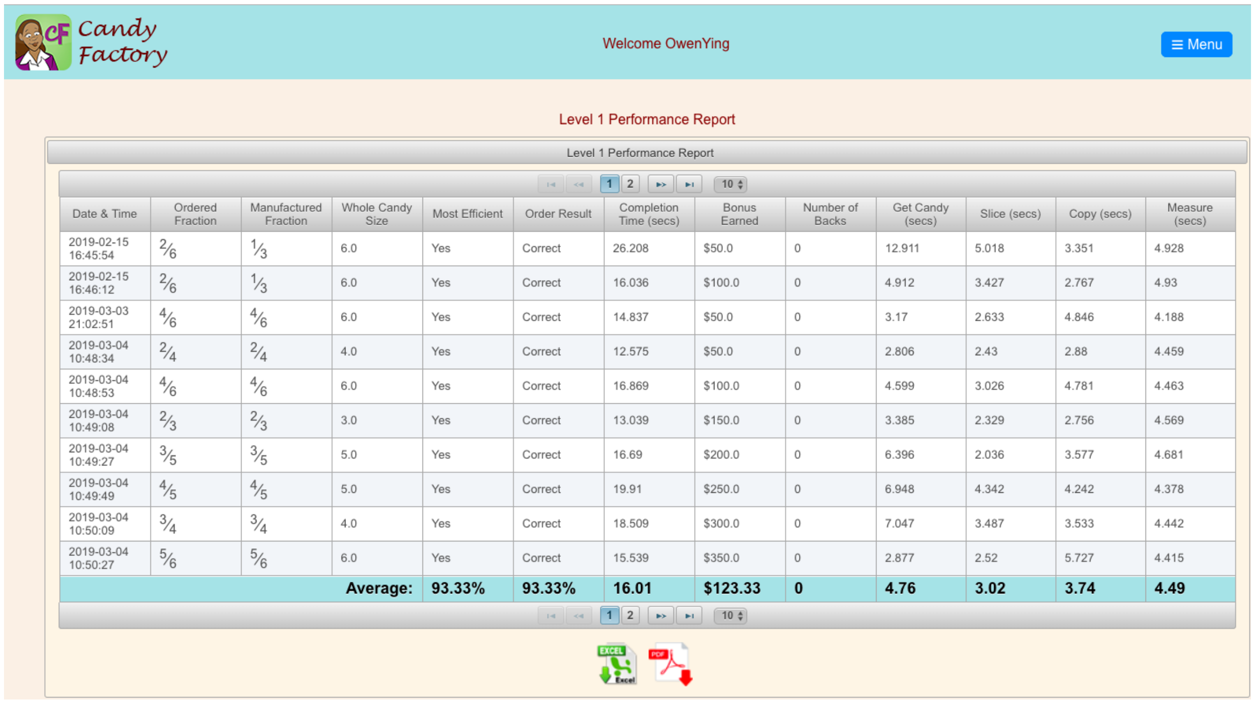
In this section we introduce three kinds of reports in cloud-based CandyFactory: Individual Report, Student Report, and Teacher Course Report.

5.8.1 *Individual Report Generation*

For individual users, cloud-based CandyFactory has two parts: performance report for each level and performance of all levels.

5.8.1.1 *Performance Report for Each Level*

For performance report of each level, when the user plays the CandyFactory game, after each game round, the performance are packed as JSON objects and sent to the backend to be stored in the database. When users want to generate their performance reports, firstly, the backend retrieves all the performance records of this user. Then by leveraging a PerformanceParser tool class in the backend, the backend parses the JSON performance strings into objects. As soon as the backend finished parsing, the result is passed into a grouping class which groups the performance by level number. After grouping, the grouped performance objects will then be processed by a calculation class to get the report. In the report, we display the recorded performance metrics as well as the averages among the metrics (Figure 51).



Level 1 Performance Report

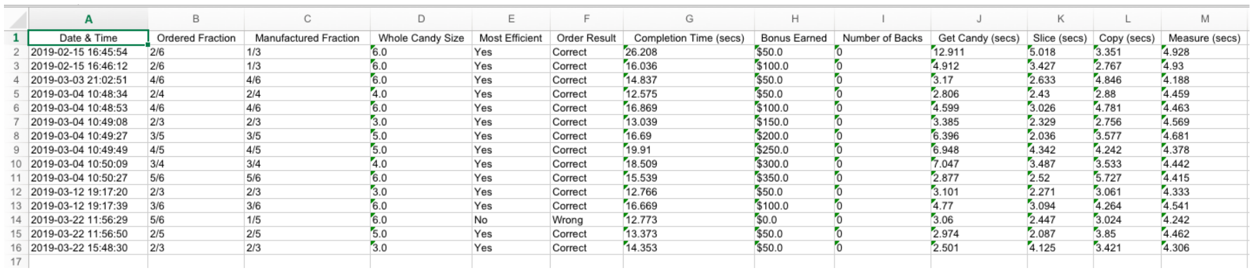
Level 1 Performance Report

Date & Time	Ordered Fraction	Manufactured Fraction	Whole Candy Size	Most Efficient	Order Result	Completion Time (secs)	Bonus Earned	Number of Backs	Get Candy (secs)	Slice (secs)	Copy (secs)	Measure (secs)
2019-02-15 16:45:54	2⁄6	1⁄3	6.0	Yes	Correct	26.208	$50.0	0	12.911	5.018	3.351	4.928
2019-02-15 16:46:12	2⁄6	1⁄3	6.0	Yes	Correct	16.036	$100.0	0	4.912	3.427	2.767	4.93
2019-03-03 21:02:51	4⁄6	4⁄6	6.0	Yes	Correct	14.837	$50.0	0	3.17	2.633	4.846	4.188
2019-03-04 10:48:34	2⁄4	2⁄4	4.0	Yes	Correct	12.575	$50.0	0	2.806	2.43	2.88	4.459
2019-03-04 10:48:53	4⁄6	4⁄6	6.0	Yes	Correct	16.869	$100.0	0	4.599	3.026	4.781	4.463
2019-03-04 10:49:08	2⁄3	2⁄3	3.0	Yes	Correct	13.039	$150.0	0	3.385	2.329	2.756	4.569
2019-03-04 10:49:27	3⁄5	3⁄5	5.0	Yes	Correct	16.69	$200.0	0	6.396	2.036	3.577	4.681
2019-03-04 10:49:49	4⁄5	4⁄5	5.0	Yes	Correct	19.91	$250.0	0	6.948	4.342	4.242	4.378
2019-03-04 10:50:09	3⁄4	3⁄4	4.0	Yes	Correct	18.509	$300.0	0	7.047	3.487	3.533	4.442
2019-03-04 10:50:27	5⁄6	5⁄6	6.0	Yes	Correct	15.539	$350.0	0	2.877	2.52	5.727	4.415
			Average:	**93.33%**	**93.33%**	**16.01**	**$123.33**	**0**	**4.76**	**3.02**	**3.74**	**4.49**

Figure 51. Individual Report Interface

After viewing the report, users can either export the results as Excel or as PDF to do further study on their own performance.

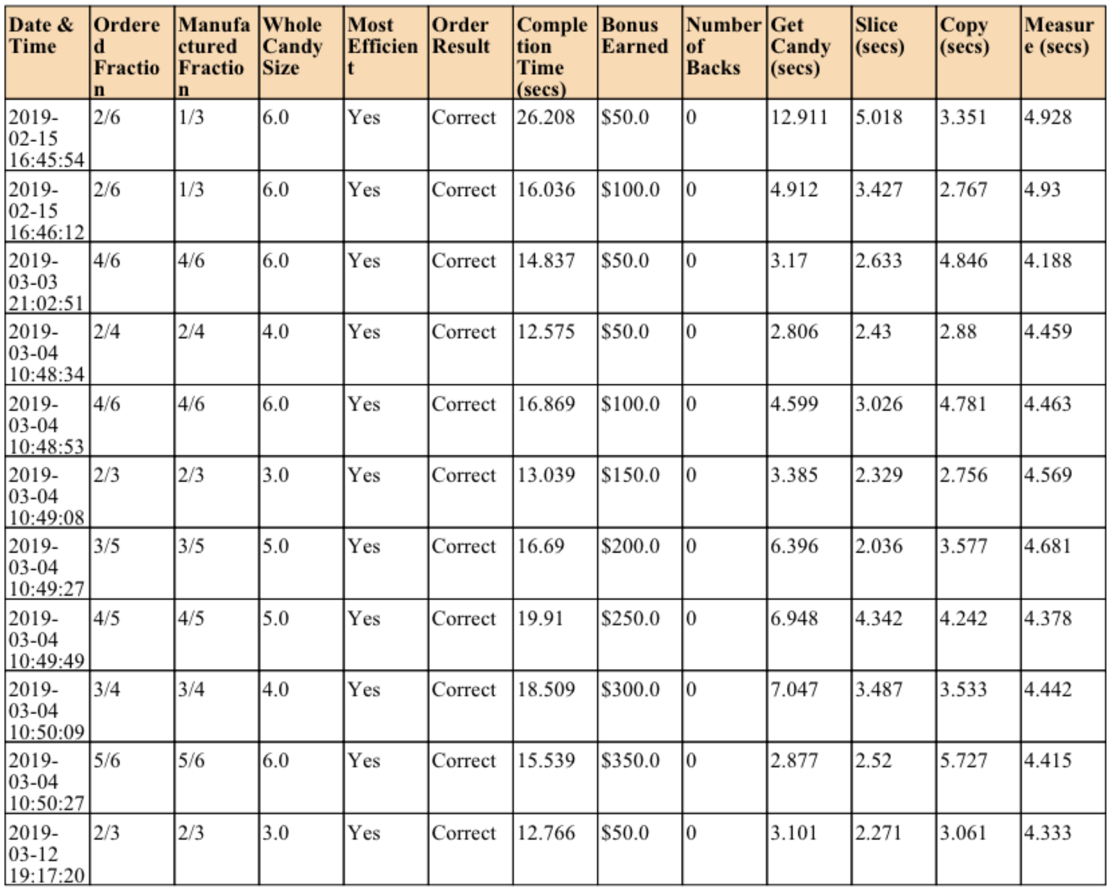
As of exporting as Excel (Figure 52), it is convenient for users to modify it, or further calculate other metrics based on the existing ones.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date & Time	Ordered Fraction	Manufactured Fraction	Whole Candy Size	Most Efficient	Order Result	Completion Time (secs)	Bonus Earned	Number of Backs	Get Candy (secs)	Slice (secs)	Copy (secs)	Measure (secs)
2	2019-02-15 16:45:54	2/6	1/3	6.0	Yes	Correct	26.208	$50.0	0	12.911	5.018	3.351	4.928
3	2019-02-15 16:46:12	2/6	1/3	6.0	Yes	Correct	16.036	$100.0	0	4.912	3.427	2.767	4.93
4	2019-03-03 21:02:51	4/6	4/6	6.0	Yes	Correct	14.837	$50.0	0	3.17	2.633	4.846	4.188
5	2019-03-04 10:48:34	2/4	2/4	4.0	Yes	Correct	12.575	$50.0	0	2.806	2.43	2.88	4.459
6	2019-03-04 10:48:53	4/6	4/6	6.0	Yes	Correct	16.869	$100.0	0	4.599	3.026	4.781	4.463
7	2019-03-04 10:49:08	2/3	2/3	3.0	Yes	Correct	13.039	$150.0	0	3.385	2.329	2.756	4.569
8	2019-03-04 10:49:27	3/5	3/5	5.0	Yes	Correct	16.69	$200.0	0	6.396	2.036	3.577	4.681
9	2019-03-04 10:49:49	4/5	4/5	5.0	Yes	Correct	19.91	$250.0	0	6.948	4.342	4.242	4.378
10	2019-03-04 10:50:09	3/4	3/4	4.0	Yes	Correct	18.509	$300.0	0	7.047	3.487	3.533	4.442
11	2019-03-04 10:50:27	5/6	5/6	6.0	Yes	Correct	15.539	$350.0	0	2.877	2.52	5.727	4.415
12	2019-03-12 19:17:20	2/3	2/3	3.0	Yes	Correct	12.766	$50.0	0	3.101	2.271	3.061	4.333
13	2019-03-12 19:17:39	3/6	3/6	6.0	Yes	Correct	16.669	$100.0	0	4.77	3.094	4.264	4.541
14	2019-03-22 11:56:29	5/6	1/5	6.0	No	Wrong	12.773	$0.0	0	3.06	2.447	3.024	4.242
15	2019-03-22 11:56:50	2/5	2/5	5.0	Yes	Correct	13.373	$50.0	0	2.974	2.087	3.85	4.462
16	2019-03-22 15:48:30	2/3	2/3	3.0	Yes	Correct	14.353	$50.0	0	2.501	4.125	3.421	4.306
17													

Figure 52. Export as Excel

As of PDF (Figure 53), it is convenient for users to print or email it to others in order to show their performance in CandyFactory.

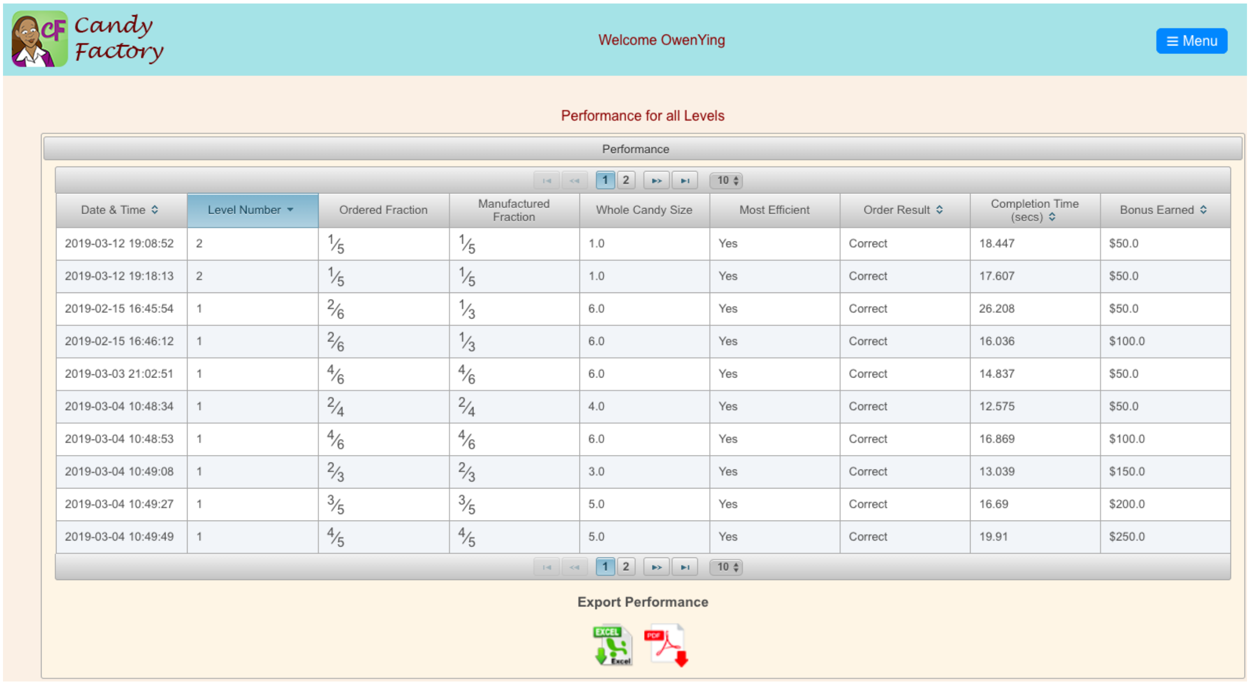


Date & Time	Ordered Fraction	Manufactured Fraction	Whole Candy Size	Most Efficient	Order Result	Completion Time (secs)	Bonus Earned	Number of Backs	Get Candy (secs)	Slice (secs)	Copy (secs)	Measure (secs)
2019-02-15 16:45:54	2/6	1/3	6.0	Yes	Correct	26.208	$50.0	0	12.911	5.018	3.351	4.928
2019-02-15 16:46:12	2/6	1/3	6.0	Yes	Correct	16.036	$100.0	0	4.912	3.427	2.767	4.93
2019-03-03 21:02:51	4/6	4/6	6.0	Yes	Correct	14.837	$50.0	0	3.17	2.633	4.846	4.188
2019-03-04 10:48:34	2/4	2/4	4.0	Yes	Correct	12.575	$50.0	0	2.806	2.43	2.88	4.459
2019-03-04 10:48:53	4/6	4/6	6.0	Yes	Correct	16.869	$100.0	0	4.599	3.026	4.781	4.463
2019-03-04 10:49:08	2/3	2/3	3.0	Yes	Correct	13.039	$150.0	0	3.385	2.329	2.756	4.569
2019-03-04 10:49:27	3/5	3/5	5.0	Yes	Correct	16.69	$200.0	0	6.396	2.036	3.577	4.681
2019-03-04 10:49:49	4/5	4/5	5.0	Yes	Correct	19.91	$250.0	0	6.948	4.342	4.242	4.378
2019-03-04 10:50:09	3/4	3/4	4.0	Yes	Correct	18.509	$300.0	0	7.047	3.487	3.533	4.442
2019-03-04 10:50:27	5/6	5/6	6.0	Yes	Correct	15.539	$350.0	0	2.877	2.52	5.727	4.415
2019-03-12 19:17:20	2/3	2/3	3.0	Yes	Correct	12.766	$50.0	0	3.101	2.271	3.061	4.333

Figure 53. Export as PDF

5.8.1.2 Performance of All Levels

Performance of all levels (Figure 54) shows the performance details of all the levels with sortable columns. With this functionality, users can view their performance details of each game round. Also with sortable columns, users can sort the performance records by any individual metrics, such as Date & Time, to get more organized view of performance records. This functionality also supports exportation as Excel or PDF.



Performance for all Levels

Performance

Date & Time	Level Number	Ordered Fraction	Manufactured Fraction	Whole Candy Size	Most Efficient	Order Result	Completion Time (secs)	Bonus Earned
2019-03-12 19:08:52	2	1/5	1/5	1.0	Yes	Correct	18.447	$50.0
2019-03-12 19:18:13	2	1/5	1/5	1.0	Yes	Correct	17.607	$50.0
2019-02-15 16:45:54	1	2/6	1/3	6.0	Yes	Correct	26.208	$50.0
2019-02-15 16:46:12	1	2/6	1/3	6.0	Yes	Correct	16.036	$100.0
2019-03-03 21:02:51	1	4/6	4/6	6.0	Yes	Correct	14.837	$50.0
2019-03-04 10:48:34	1	2/4	2/4	4.0	Yes	Correct	12.575	$50.0
2019-03-04 10:48:53	1	4/6	4/6	6.0	Yes	Correct	16.869	$100.0
2019-03-04 10:49:08	1	2/3	2/3	3.0	Yes	Correct	13.039	$150.0
2019-03-04 10:49:27	1	3/5	3/5	5.0	Yes	Correct	16.69	$200.0
2019-03-04 10:49:49	1	4/5	4/5	5.0	Yes	Correct	19.91	$250.0

Export Performance

Figure 54. Performance of All Levels

5.8.2 Student Report Generation

Student report generation is almost the same as individual report generation. The only feature that individual report generation does not have is that when collecting student performance, the performance records are attached with the course primary key, meantime the course primary key is attached to the teacher who created this course. In this case, all the performance records are recorded under corresponding teacher. With this feature, the teacher, who created the course and the students, has access to the students' performance reports as well as performance details.

With this feature, not only students can view their performance reports and details, but their teachers have access to their students' performance records as well.

5.8.3 Course Report Generation

Course report generation under teacher account is another core feature of cloud-based CandyFactory. This feature allows teachers to generate their course reports according to the students' performance under the course.

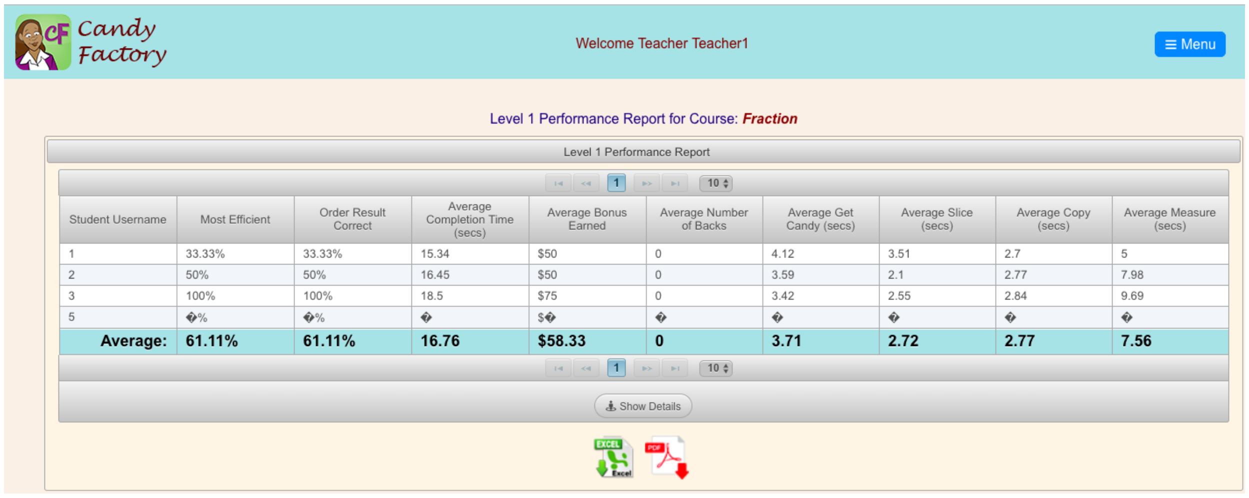
Under the teacher account, in the course panel, teachers can select a course and click on generate course report to get the course report of the selected course. The course report includes ten performance metrics which measures the corresponding student's performance.

Each row in the course report represents a student's current averages of all the performance records. The record is updated in real-time. As along as a corresponding student plays the game, the record row will then be updated after the page is refreshed. The rows with "?" means the student has not played the game at all. Thus rows with question marks are not counted when calculating the course report. The reason we don't display un-played rows as zeroes is zeroes can lead to misunderstanding of worst performance. Thus, we use question mark to represent un-played student performance.

In order to generate course report, cloud-based CandyFactory's backend does the following processing:

1. Get the current selected course's primary key.
2. Retrieve all the students under the course.
3. For each individual student, retrieve all the performance records under this student. Group the performance records by level number. Calculate the total averages of the student under certain level based on each level's performance records.
4. After calculating each student's averages under all levels, based on the whole averages, we calculate another average among all the previous calculated averages to get the final averages of the course.

For students currently have no performance records (have not played), we skip them when doing the calculation (Figure 55).



Level 1 Performance Report for Course: ***Fraction***

Level 1 Performance Report

Student Username	Most Efficient	Order Result Correct	Average Completion Time (secs)	Average Bonus Earned	Average Number of Backs	Average Get Candy (secs)	Average Slice (secs)	Average Copy (secs)	Average Measure (secs)
1	33.33%	33.33%	15.34	$50	0	4.12	3.51	2.7	5
2	50%	50%	16.45	$50	0	3.59	2.1	2.77	7.98
3	100%	100%	18.5	$75	0	3.42	2.55	2.84	9.69
5	�%	�%	�	$�	�	�	�	�	�
Average:	**61.11%**	**61.11%**	**16.76**	**$58.33**	**0**	**3.71**	**2.72**	**2.77**	**7.56**

Figure 55. Course Report Interface

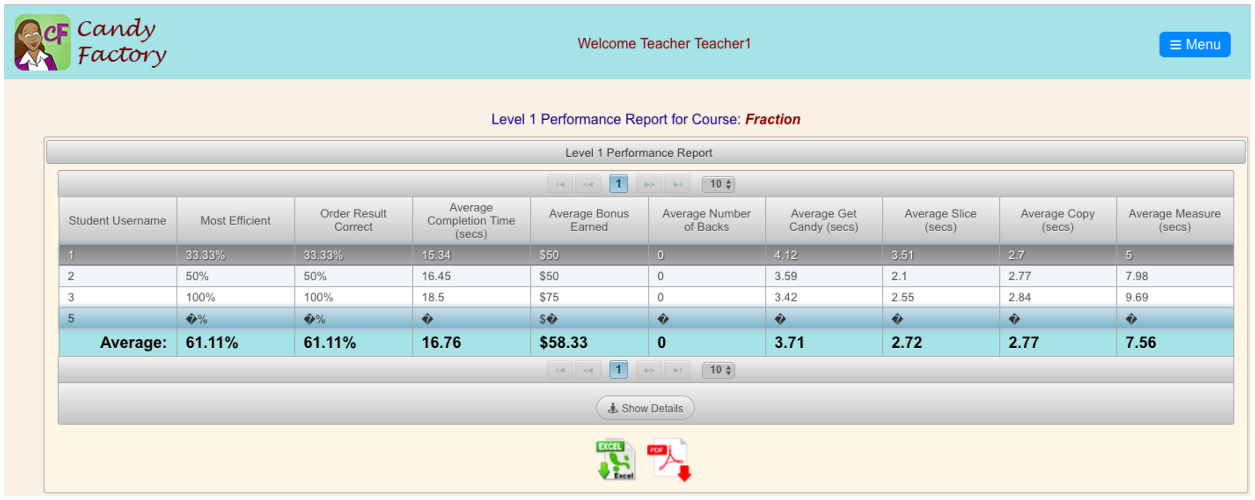
With the course report, the teacher can provide more targeted help to their students. For example, student1's order correctness of level one is very low, which can reflect that student1 currently has not understand or not understand well of discrete proper fractions. Therefore, the teacher can help student1 with discrete proper fractions.

As as result, with the help of course report generation, students' learning process can be smooth: teachers teach fractions; students learn and practice fractions while playing CandyFactory; teachers view students' performance reports to keep track of the students' learning process; teachers help individual student with more targeted method; students play the game again. Consequently, cloud-based CandyFactory provides students and teachers with an incremental and smooth learning progress, which helps students and teachers to learn better.

Course report generation also support exportation as Excel and PDF as previous sections.

5.8.4 View Specific Student's Report Under Course Report

This functionality provides the teacher access to individual student's performance reports through course report. After viewing a student's reports, if the teacher wants to view details of that student, can click on the specific student row and the "Show Details" button (Figure 56). Then the teacher will be redirected to the student's reports to view more detailed student's performance (Figure 57).

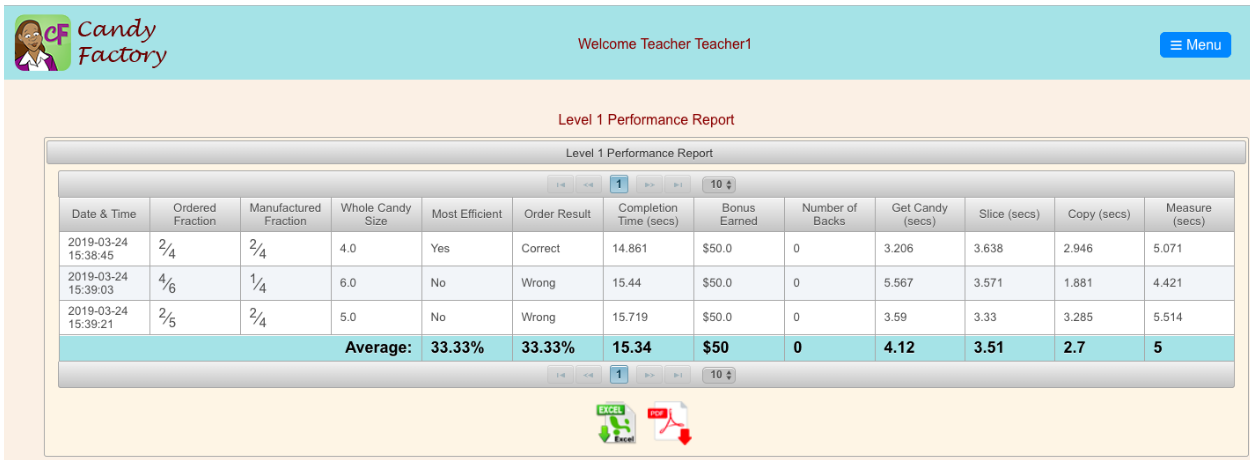


Level 1 Performance Report for Course: *Fraction*

Level 1 Performance Report

Student Username	Most Efficient	Order Result Correct	Average Completion Time (secs)	Average Bonus Earned	Average Number of Backs	Average Get Candy (secs)	Average Slice (secs)	Average Copy (secs)	Average Measure (secs)
1	33.33%	33.33%	15.34	$50	0	4.12	3.51	2.7	5
2	50%	50%	16.45	$50	0	3.59	2.1	2.77	7.98
3	100%	100%	18.5	$75	0	3.42	2.55	2.84	9.69
5	�%	�%	�	$�	�	�	�	�	�
Average:	**61.11%**	**61.11%**	**16.76**	**$58.33**	**0**	**3.71**	**2.72**	**2.77**	**7.56**

Figure 56. Select Student under Course Report



Level 1 Performance Report

Level 1 Performance Report

Date & Time	Ordered Fraction	Manufactured Fraction	Whole Candy Size	Most Efficient	Order Result	Completion Time (secs)	Bonus Earned	Number of Backs	Get Candy (secs)	Slice (secs)	Copy (secs)	Measure (secs)
2019-03-24 15:38:45	2/4	2/4	4.0	Yes	Correct	14.861	$50.0	0	3.206	3.638	2.946	5.071
2019-03-24 15:39:03	4/6	1/4	6.0	No	Wrong	15.44	$50.0	0	5.567	3.571	1.881	4.421
2019-03-24 15:39:21	2/5	2/4	5.0	No	Wrong	15.719	$50.0	0	3.59	3.33	3.285	5.514
			Average:	**33.33%**	**33.33%**	**15.34**	**$50**	**0**	**4.12**	**3.51**	**2.7**	**5**

Figure 57. Show Selected Student's Performance Report

5.9 Administrator Account

Cloud-based CandyFactory has a built-in administrator account, which is used by the system administrator to view the whole system's information, such as account information and course information.

After logging in as administrator, there are two buttons to redirect the administrator to either user account information or course information (Figure 58).

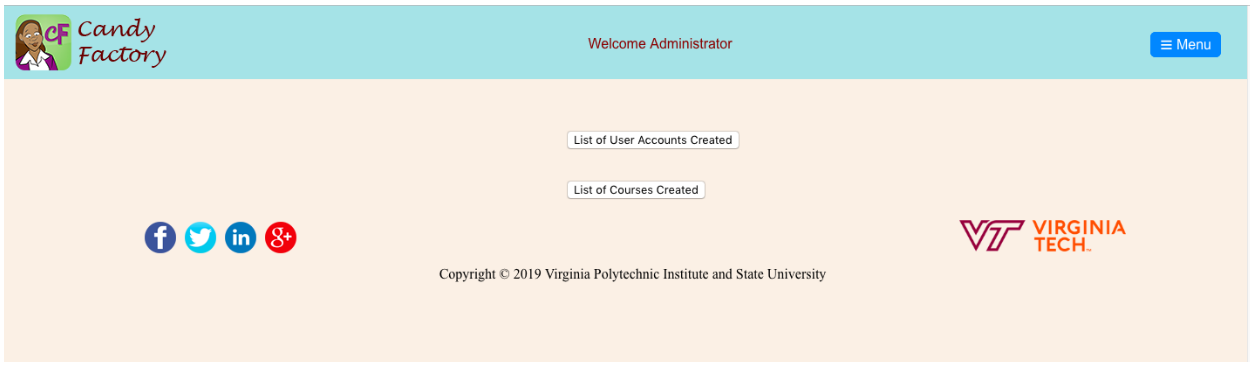
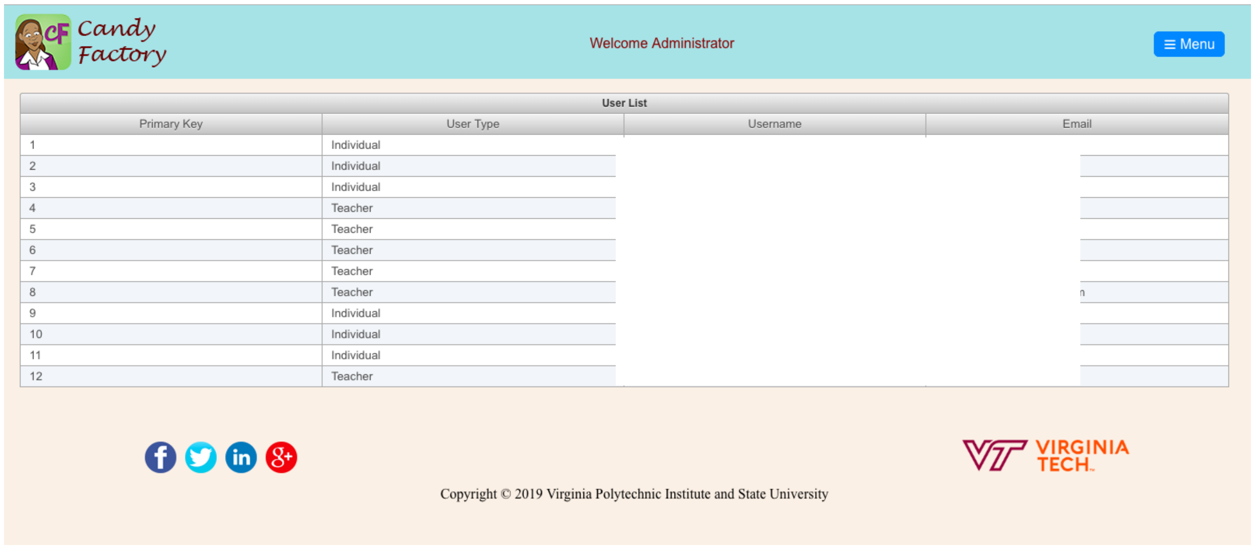

Candy Factory
Welcome Administrator
≡ Menu
List of User Accounts Created
List of Courses Created
VIRGINIA TECH
Copyright © 2019 Virginia Polytechnic Institute and State University

Figure 58. Administrator Account Page

The "List of User Accounts Created" redirects the administrator to currently created user accounts. The list includes the user's primary key, the account's user type, username, and email information (Figure 59).



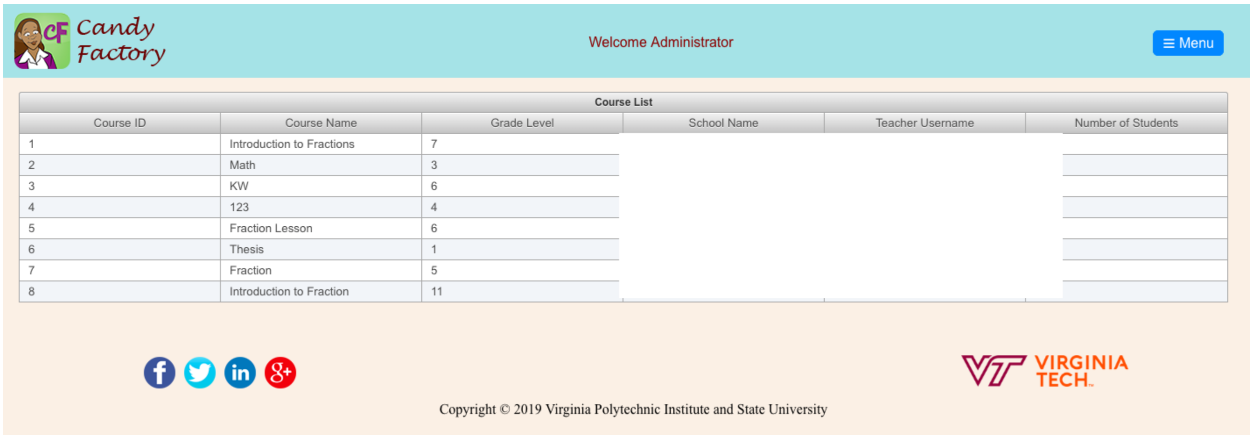
User List

Primary Key	User Type	Username	Email
1	Individual	[redacted]	[redacted]
2	Individual	[redacted]	[redacted]
3	Individual	[redacted]	[redacted]
4	Teacher	[redacted]	[redacted]
5	Teacher	[redacted]	[redacted]
6	Teacher	[redacted]	[redacted]
7	Teacher	[redacted]	[redacted]
8	Teacher	[redacted]	[redacted] n
9	Individual	[redacted]	[redacted]
10	Individual	[redacted]	[redacted]
11	Individual	[redacted]	[redacted]
12	Teacher	[redacted]	[redacted]

Copyright © 2019 Virginia Polytechnic Institute and State University

Figure 59. List of User Accounts Created Interface

The "List of Courses Created" redirects the administrator to currently created courses. The list includes course id, course name, grade level, school name, teacher who created this course, and number of students under this course (Figure 60).



Course List

Course ID	Course Name	Grade Level	School Name	Teacher Username	Number of Students
1	Introduction to Fractions	7	[redacted]	[redacted]	[redacted]
2	Math	3	[redacted]	[redacted]	[redacted]
3	KW	6	[redacted]	[redacted]	[redacted]
4	123	4	[redacted]	[redacted]	[redacted]
5	Fraction Lesson	6	[redacted]	[redacted]	[redacted]
6	Thesis	1	[redacted]	[redacted]	[redacted]
7	Fraction	5	[redacted]	[redacted]	[redacted]
8	Introduction to Fraction	11	[redacted]	[redacted]	[redacted]

Copyright © 2019 Virginia Polytechnic Institute and State University

Figure 60. List of Courses Created Interface

Chapter 6: CandyFactory Self-Evaluation

A Digital Educational Game (DEG) is a game created for the purpose of teaching a subject in the form of software that runs on a computer such as desktop, laptop, handheld, or game console [Aslan and Balci 2015]. Cloud-based CandyFactory is a DEG which teaches children and teenagers fractions by intuitively interacting with the game. In order to evaluate CandyFactory, we are going to use a set of quality indicators introduced in GAMED [Aslan and Balci 2015] including *Acceptability*, *Challengeability*, *Clarity*, *Effectiveness*, *Engageability*, *Enjoyability*, *Interactivity*, *Localizability*, *Rewardability*, *Simplicity*, *Transformativeness* and *Usability*.

In this chapter, we are going to apply self-evaluation to Cloud-based CandyFactory based on the metrics mentioned above.

6.1 Acceptability

Acceptability is the degree to which the DEG fulfills its requirements and learning objectives [Aslan and Balci 2015].

As introduced in Chapter 2:, the CandyFactory iPad version is an award winning application. Balefire Labs TopRated says “only about 10% of the apps we have reviewed so far (of a total of more than 3800) have achieved a grade of A or B” [Balefire Labs]. Meantime the CandyFactory iPad version is well designed and tested, the award can fully reflect its *Acceptability*.

Based on the CandyFactory iPad version, we reprogram all the functionalities and features on the cloud and add new features to further enhance the game. Thus Cloud-based CandyFactory can fulfill its requirements and learning objectives.

6.2 Challengeability

Challengeability is the degree to which the user finds the DEG to be exciting, stimulating, and inspiring to play [Aslan and Balci 2015].

CandyFactory has five levels, each level delivers more challenges than the previous level as we introduced in 2.4. The difficulty distributed smoothly throughout the five levels. Level 1 introduces discrete candy and proper fractions. Level 2 changes to continuous candy and unit fractions, which increases the candy bar’s difficulty but decreases the fraction’s difficulty. Level 3 upgrades to continuous candy and all proper fractions. Level 4 teaches continuous candy and improper fractions. Level 5 reverses level 4’s process, to teach the students in a different aspect.

From the smooth difficulty curve, students can find CandyFactory to be exciting, stimulating and inspiring to play.

6.3 Clarity

Clarity is the degree to which the DEG is unambiguous and understandable [Aslan and Balci 2015].

CandyFactory has two features to improve *Clarity*. First, CandyFactory has a tutorial mode corresponding to each level in game mode. In tutorial mode, students are guided by an animated CandyFactory Manager step by step to complete a customer order. With the animation and explanation of the Manager, students can easily understand the task and the gameplay operations. Second, in game mode, each scene has a "help" button on the top left corner. Clicking on it will display an explanation of the current scene, thus guiding the students on how to play CandyFactory.

With tutorial mode and help button, students are able to understand CandyFactory clearly.

6.4 Effectiveness

Effectiveness is the degree to which the DEG improves the effectiveness of learning the subject in a significantly better way in comparison to other pedagogies [Aslan and Balci 2015].

According to GAMED [Aslan and Balci 2015], the theory of learning in good DEGs fits better with the modern, high-tech, global world today's children and teenagers live in. In the CandyFactory iPad version, "students learn to conceive of fractions as sizes relative to the whole…by coordinating actions of partitioning and iterating candy bars" [Aslan, Norton, and Balci 2019; LTRG 2019]. Based on the iPad version CandyFactory, Cloud-based CandyFactory introduces cloud software's new features such as performance report generation, which greatly helps the teachers to track the students' learning progress and to offer more targeted help to the students.

With the foundation of the CandyFactory iPad version and the powerful cloud software's new features, Cloud-based CandyFactory is able to improve the effectiveness of learning the subject in a significantly better way in comparison to other pedagogies.

6.5 Engageability

Engageability is the degree to which the user is captivated and addicted by DEG playing [Aslan and Balci 2015].

CandyFactory iPad version is an award winning application. Cloud-based CandyFactory fully replicates the functionalities and features as well as adds new features to enhance the game. Thus Cloud-based CandyFactory has very good *Engageability*.

6.6 Enjoyability

Enjoyability is the degree to which the user finds the DEG playing to be fun [Aslan and Balci 2015].

Based on the strong foundation of the CandyFactory iPad version, Cloud-based CandyFactory has the same attractive animations and reward system. Moreover, Cloud-based CandyFactory leverages cloud software's features to enhance CandyFactory from different aspects such as multi-device synchronization. Therefore, users can find Cloud-based CandyFactory is fun to play, their game status is stored in the cloud thus can be accessed anywhere with Internet and web browsers.

6.7 Interactivity

Interactivity is the degree to which the user actively interacts with the DEG during playing [Aslan and Balci 2015].

By mapping the iPad CandyFactory's interaction techniques to Cloud-based CandyFactory, we provide more flexible interactions such as drag-and-drop deletion and shipment animation. When a user has new achievements, badges fly across the bottom with faded effects to celebrate their success.

Users can actively interact with Cloud-based CandyFactory. Additionally, Cloud-based CandyFactory encourages users to perform further interactions through the reward system.

6.8 Localizability

Localizability is the degree to which the DEG can easily be adopted, preferably via preferences or options, (a) to satisfy the needs of languages other than English, and (b) to local standards such as decimal separator, currency symbol, time zone, calendar, etc. [Aslan and Balci 2015].

Cloud-based CandyFactory offers more timing options than the iPad version of CandyFactory (Figure 44). Thus providing the user with more choices when playing the game. In terms of languages and other local standards, Cloud-based CandyFactory only supports English and does not support different decimal separators and currency symbols. The date and time in performance reports are localized based on where the user is.

Cloud-based CandyFactory does not satisfy the needs of languages other than English nor decimal separators or currency symbols. However, the game satisfies the time zone and calendar local standards.

6.9 Rewardability

Rewardability is the degree to which the DEG gives rewards (e.g., points, money, trophies, certificates) to the user so that the user feels a sense of accomplishment [Aslan and Balci 2015].

Compared to the iPad CandyFactory, Cloud-based CandyFactory largely enhances the reward system by adding synchronization of different devices and reward notifications, introduced in 5.5.2.3. After each game round, as long as the user gets new achievements, the trophies will fly from the right to the left in a shaded animation. When the user clicks on the achievement button, all the achievements will be displayed with a clickable label under each trophy. The user is able to view their achievements anywhere at any time as long as they are using the same account.

With the cloud software synchronization features and intuitive animations, Cloud-based CandyFactory offers a decent achievement system to give the user a sense of accomplishment.

6.10 Simplicity

Simplicity is the degree to which the DEG can be understood without difficulty [Aslan and Balci 2015].

Cloud-based CandyFactory has very simple interaction techniques: click and drag-and-drop. With the simple actions, the user can easily understand the operations in each scene. Moreover, with the tutorial mode and help button in each scene, Cloud-based CandyFactory can be easily understood without difficulty.

6.11 Transformativeness

Transformativeness is the degree to which the DEG transforms the subject learning in a significantly better way in comparison to other pedagogies [Aslan and Balci 2015].

Based on strong foundation of the iPad CandyFactory, Cloud-based CandyFactory as introduced in Chapter 5: has the ability to transform the subject learning in a better way.

6.12 Usability

Usability is the degree to which the DEG can easily be employed for its intended use [Aslan and Balci 2015].

For CandyFactory game, Cloud-based CandyFactory is based on JavaScript with JQuery package supported, thus has universal portability. Also in the client tier, Cloud-based CandyFactory leverages PrimeFaces [PrimeFaces 2019], a lightweight Java EE user interface framework regarded as an excellent UI library for creating a variety of easy-to-use user interfaces. It also provides different components and showcases for users to use.

By building the game on JQuery and the system on PrimeFaces, Cloud-based CandyFactory can easily be employed for its intended use.

Chapter 7: Conclusions and Future Research

7.1 Conclusions

This thesis presents a cloud-based educational game for teaching fractions. Based on the award winning iPad version of CandyFactory, our Cloud-based CandyFactory, taking the advantages of cloud software's characteristics, provides a scientific and attractive game with cross platform features and synchronization through different devices. Moreover, our Cloud-based CandyFactory provides a new paradigm for the teachers who can keep track of the students' learning progress and provide more targeted help through the game performance report functionality. Currently, local educational games lack connection between students and teachers, while few existing online educational games provide cloud software features such as report generation. Cloud-based CandyFactory innovatively offers easy course creation, convenient student accounts creation, performance report generation, and data exportation functionalities. Its technology also protects the students' performance privacy, user account privacy (by SHA-1 encryption), and simplifies the operations to its best extent.

Cloud-based CandyFactory is created based on the software development life cycle [Balci 2012]: problem formulation, requirements engineering, architecture design, software components design, and development. Verification, validation, and quality testing are integrated in this process. By following the life cycle, our Cloud-based CandyFactory is well designed, implemented, and tested.

With Cloud-based CandyFactory, students can learn fractions through a fun and smooth process; teachers can generate reports and provide more targeted help to their students; individual users can learn fractions through the game and then view their own performance report to analyze their learning progress. All the game data is synchronized upon all devices, users can access them anywhere at any time.

7.2 Future Research

This study focuses on a cloud-based educational game system for teaching fractions. Future research may include:

1. Provide a shop in the game for users to spend their bonus money.
2. Apply information visualization techniques to visualize the game performance report and provide exportations of the visualizations.
3. Apply data analysis techniques, such as machine learning, to generate more intelligent reports to teachers.
4. Refine the whole system's user interface.
5. Do user studies on Cloud-based CandyFactory.

REFERENCES

Aslan, S. and O. Balci (2015), "GAMED: Digital Educational Game Development Methodology," *Simulation: Transactions of the Society for Modeling and Simulation International 91*, 4 (Apr.), 307-319, doi: 10.1177/0037549715572673.

Aslan, S., A. Norton, and O. Balci (2019), "CandyFactory Educational Game," Apple App Store, https://itunes.apple.com/us/app/candyfactory-educational-game-for-ipad/id533213891?ls=1&mt=8

Apple (2019), "App Store," https://www.apple.com/ios/app-store/

Balci, O. (2012), "A Life Cycle for Modeling and Simulation," *Simulation: Transactions of the Society for Modeling and Simulation International 88*, 7, 870–883.

Balci, O. (2019), "CS5704 Software Engineering," https://manta.cs.vt.edu/cs5704/Course.html#Description

Balefire Labs (2019), "Apps," http://www.balefirelabs.com/apps/

Eastlake 3rd, D., & Jones, P. (2001). *US secure hash algorithm 1 (SHA1)* (No. RFC 3174).

Eder, Johann, Gerti Kappel, and Michael Schrefl (1994), "Coupling and cohesion in object-oriented Systems," Technical Report, University of Klagenfurt.

IEEE (2000), "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems," *IEEE STD* 1471-2000, New York, NY.

LTRG (2019), "CandyFactory Educational Game for iPad," http://ltrg.centers.vt.edu/projects/games/apps/candyFactoryiPad.htm

Oracle (2018), "The Client Tier," https://docs.oracle.com/cd/E19226-01/820-7759/gcrla/index.html

Prensky, M. (2007), "Digital Game-Based Learning, Paragon House," St. Paul, Minnesota.

PrimeFaces (2019), "PrimeFaces for JSF," https://www.primefaces.org/#primefaces

Schmidt, Douglas C., et al (2013), "Pattern-Oriented Software Architecture," *Patterns for Concurrent and Networked Objects*. Vol. 2. John Wiley & Sons.

Tutorials Point (2019), "UML Tutorial," https://www.tutorialspoint.com/uml/

U.S. Department of Education (2019), "Family Educational Rights and Privacy Act (FERPA)," https://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html

Wikipedia (2019), "PBKDF2," https://en.wikipedia.org/wiki/PBKDF2