Technical Design Document

It always seems impossible until it's done.

- Nelson Mandela

By: cd msc/

Nathan Moore | Gustavo Castillo | Peter DeNicola

Anthony Cloudy | Joe St. Angelo

Introduction:

Quizard Quest is a new approach to studying online and making flashcards. Rather than slaving away to prepare for an exam or review notes, Quizard Quest's goal is to turn studying into something fun and enjoyable. Using a combination of graphics, animations, and sound effects, we have turned the monotony of simply practicing flash cards into something enjoyable, interesting, and most importantly, fun! Quizard Quest puts excitement and motivation into the fundamentals of studying and learning.

Members and Roles:

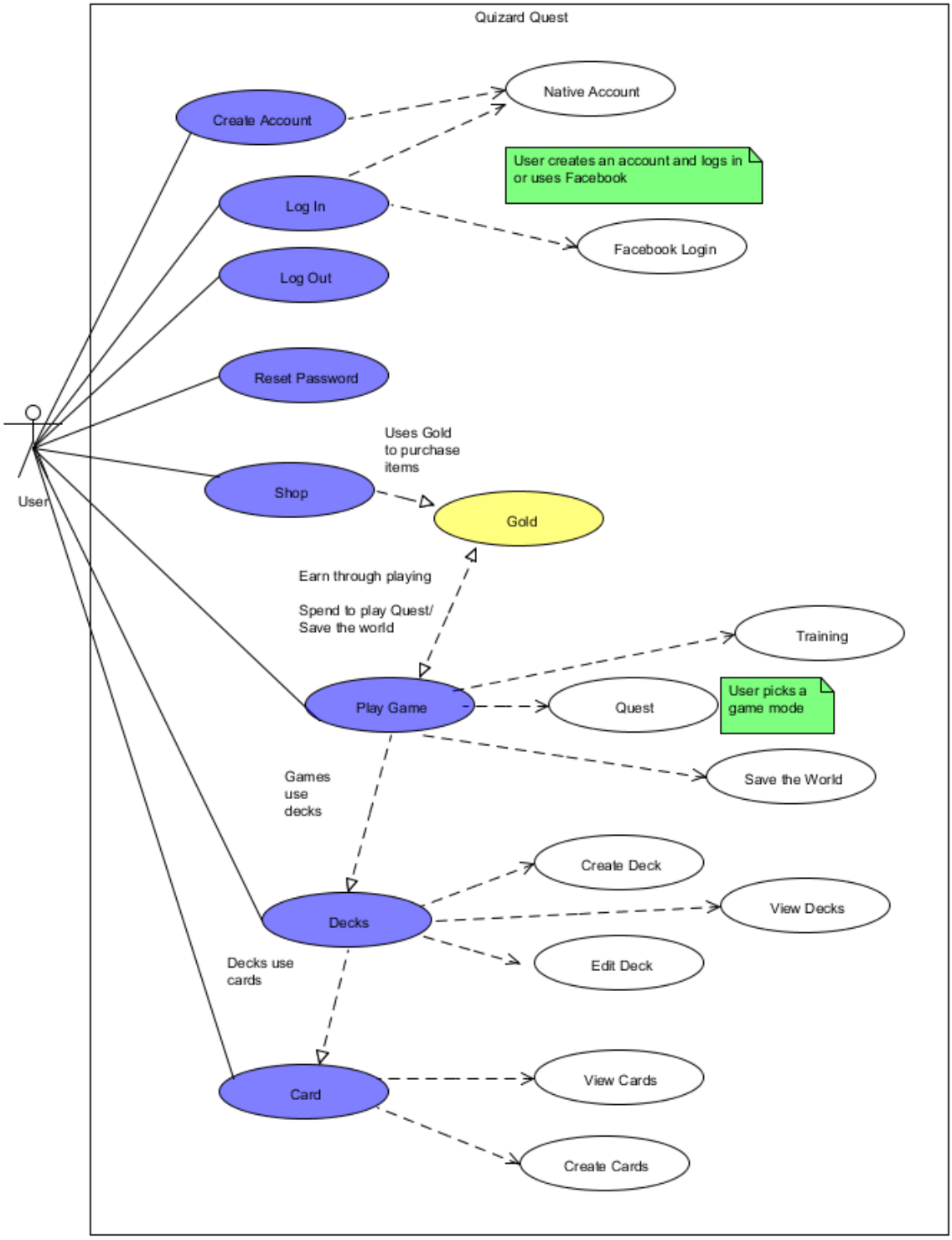
- Anthony: Assistant Project Manager, Build Master, and Game Designer
- Gus: Website Designer
- Joe: Database Designer and Database/Website Communication Manager
- Nathan: Project Manager, CSS Designer, and Jack-of-All-Trades
- Peter: Android Designer and Manager

Features List:

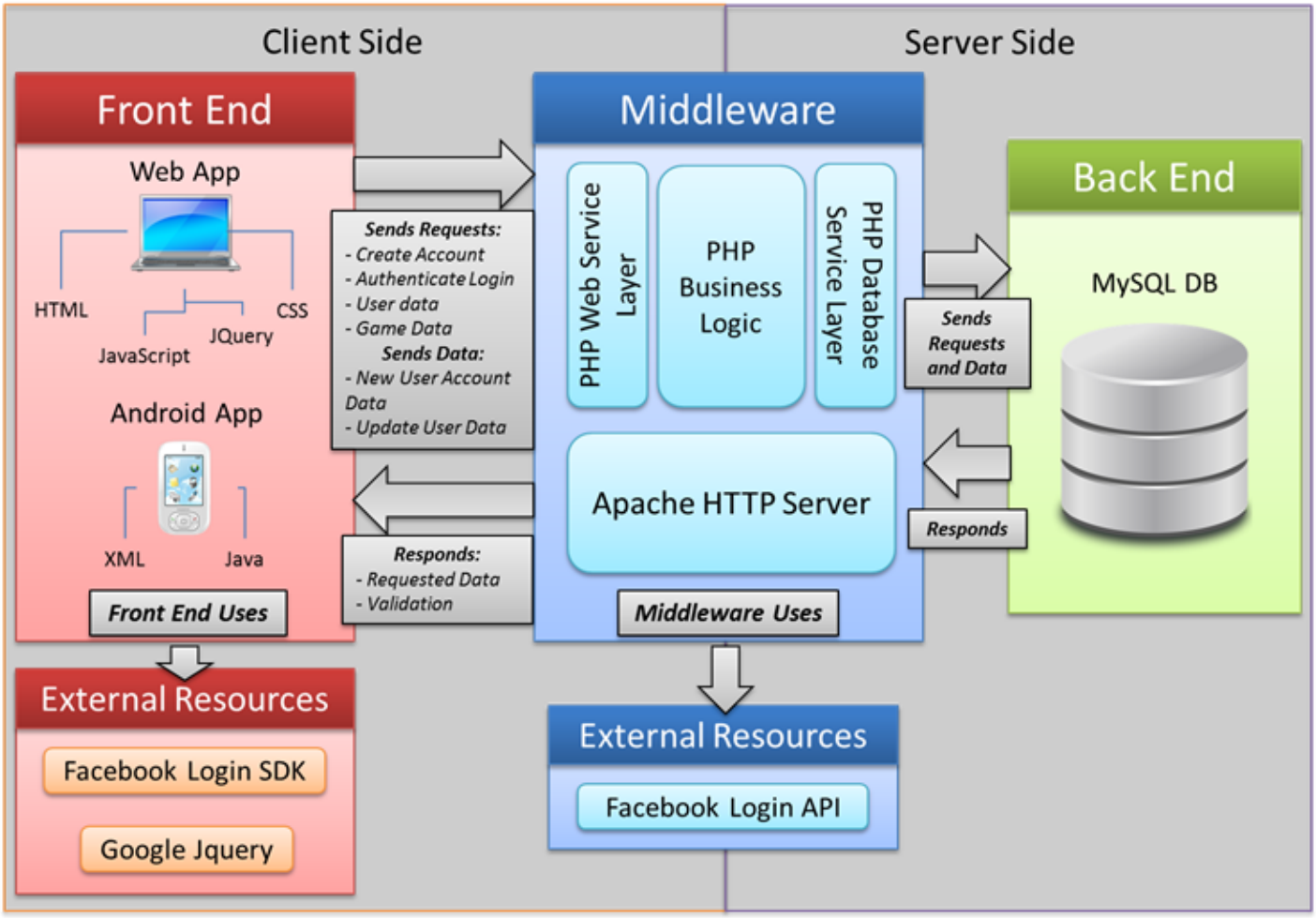
- Create a new account
  - Has a username
  - Has a password
  - Has an email
  - Has a security question and answer
  - Optional gender
  - Optional grade
- Connect through facebook
- Create flash cards
  - Has a category
  - Optional subcategory
  - Has a question
  - Has an answer
  - Has a difficulty
- Create decks
  - Use cards already created
  - Has a name

- Edit decks
    - Add cards to a deck
    - Remove cards from a deck
- Training Mode
    - Free to play
    - Answer flashcards in a deck
    - Earn 10 gold per card correct
    - Earn 10 experience per card correct
- Quest Mode
    - Costs 150 gold to attempt
    - Answer flashcards in a deck
    - Earn 30 gold per card correct
    - Earn 10 experience per card correct
    - Game over if over 30% of cards are incorrect
- Save the World
    - Costs 300 gold to attempt
    - Answer flashcards in a deck
    - Earn 50 gold per card correct
    - Earn 10 experience per card correct
    - Game over if over 30% of cards are incorrect
    - Game over if timer hits zero, based on difficulty of cards
        - ➢ Easy: +5 seconds
        - ➢ Medium: +10 seconds
        - ➢ Hard: +15 seconds
        - ➢ Nigh-impossible: +20 seconds
- Shop
    - Purchase different avatars with gold
- Android
    - Allow user to login with created account
    - Allow user to select created deck
    - Go through the deck, answering questions

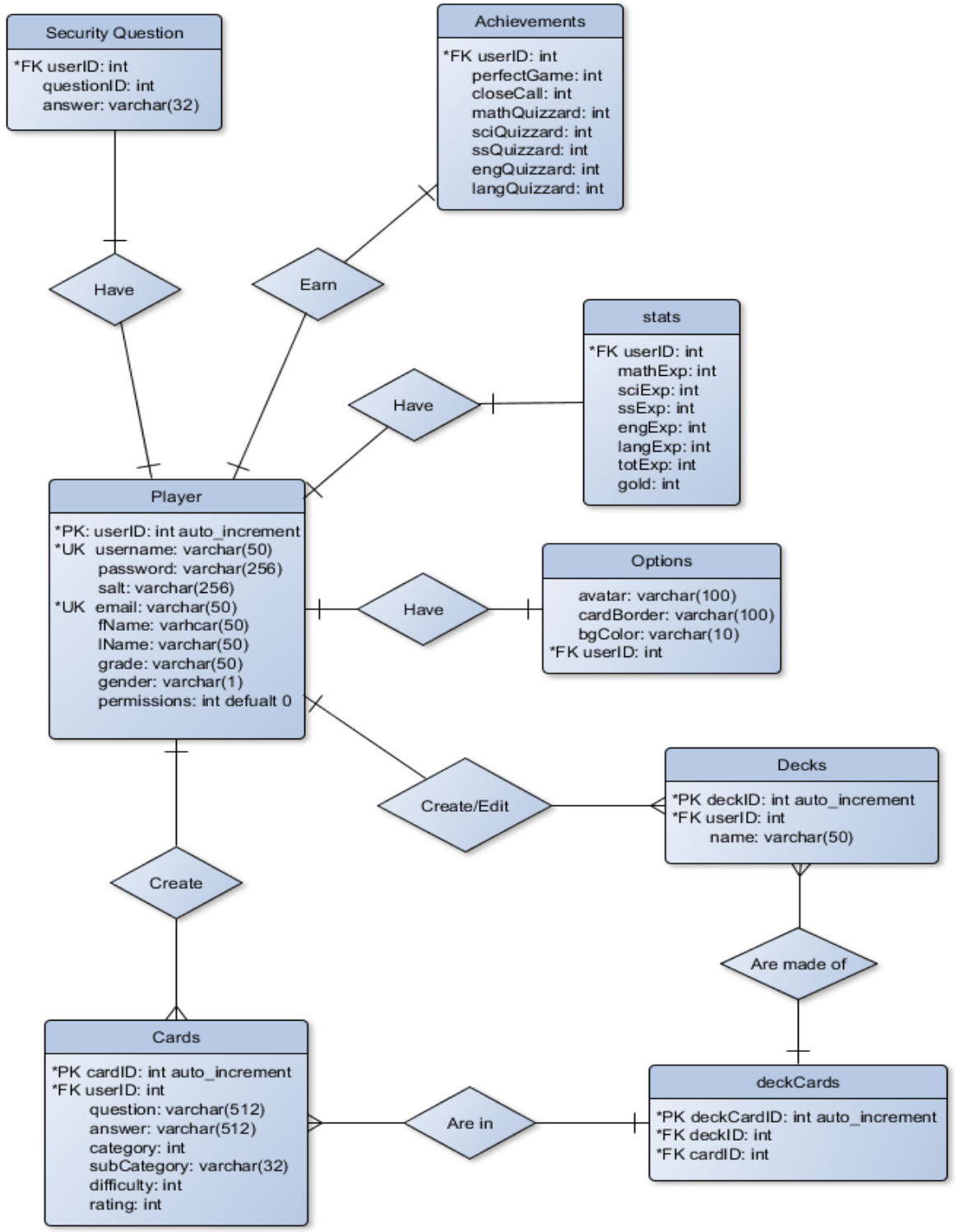
Use-Case Diagram:



Software Architecture:

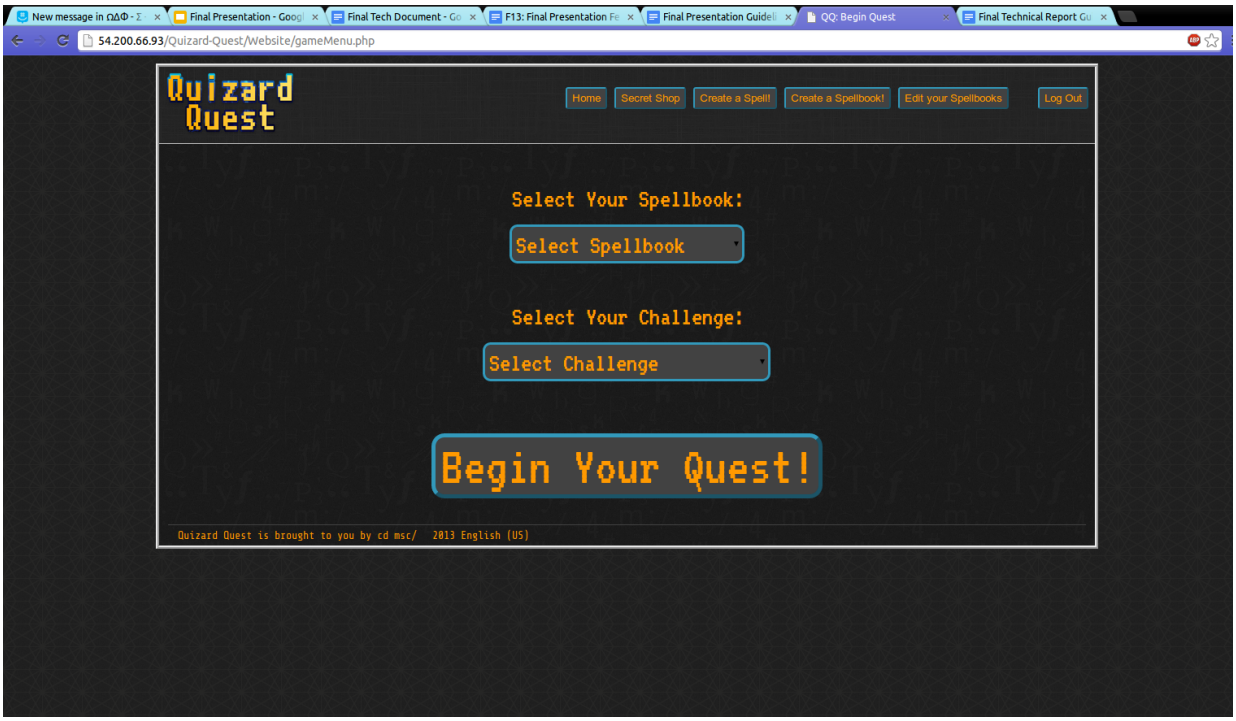


The Data Base (see appendix for dictionary):

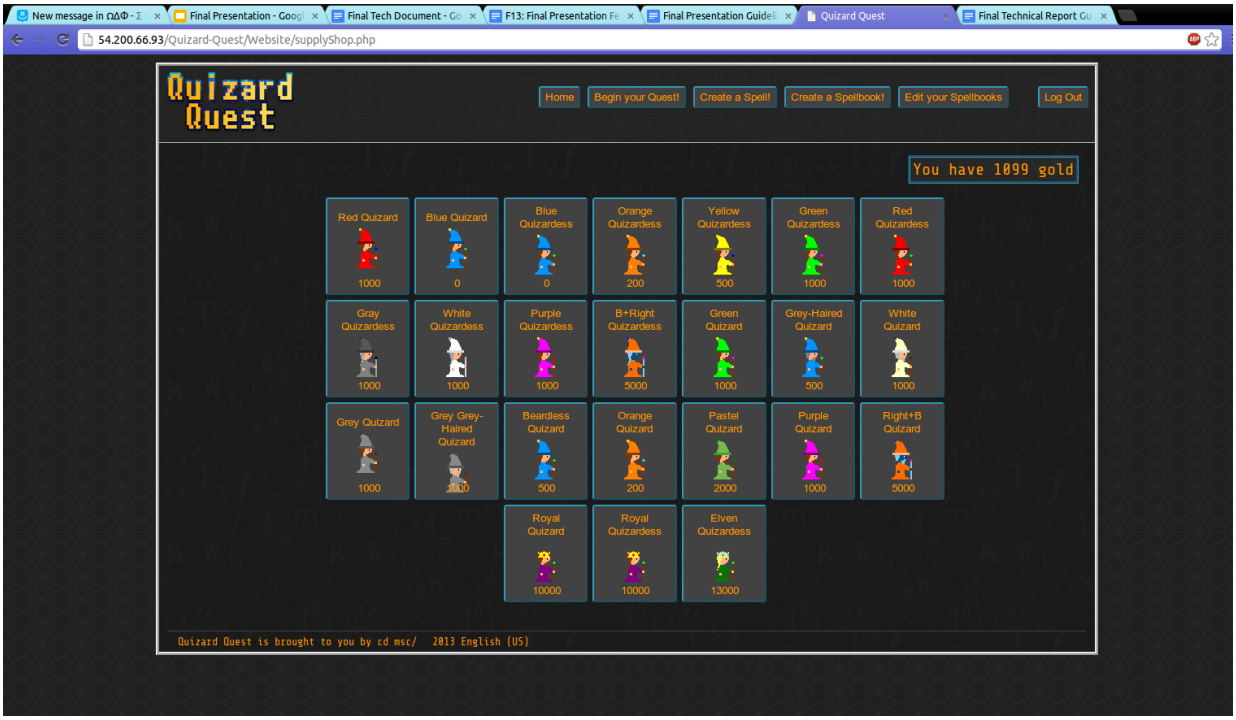


UI:

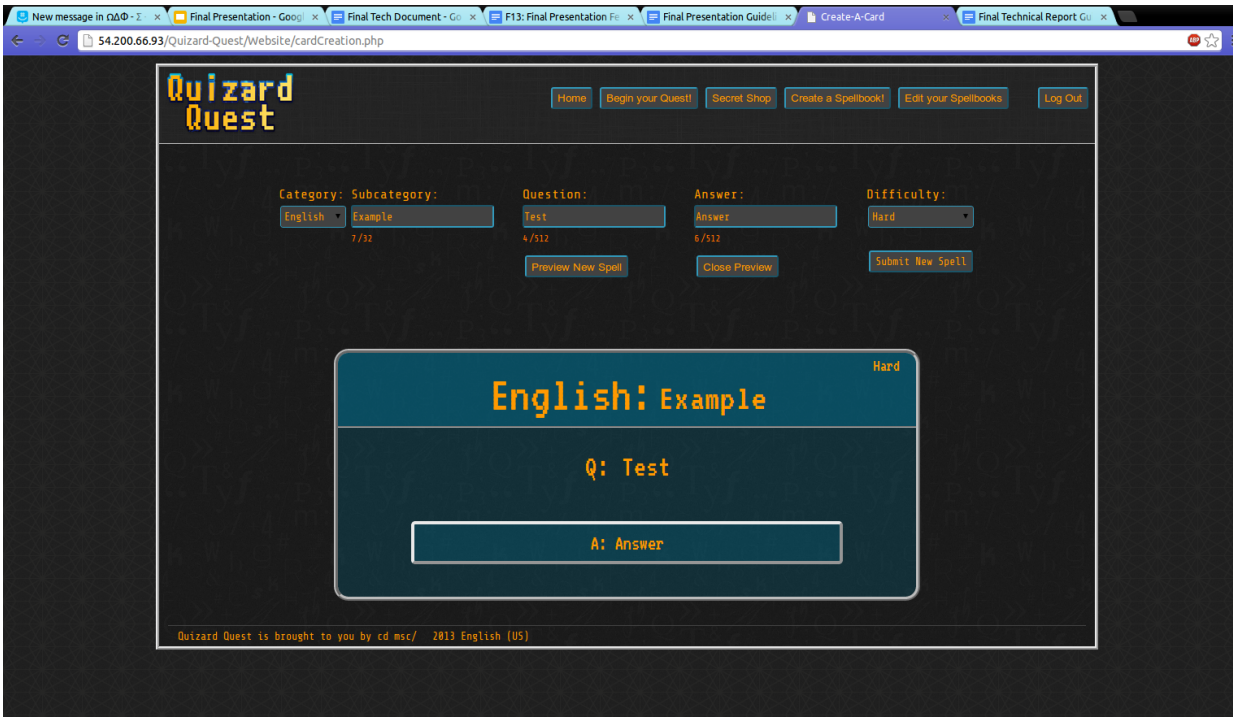
➢ **This is the main page. Shows the users stats and decks**



➢ **This is the page where you begin your Challenge.**

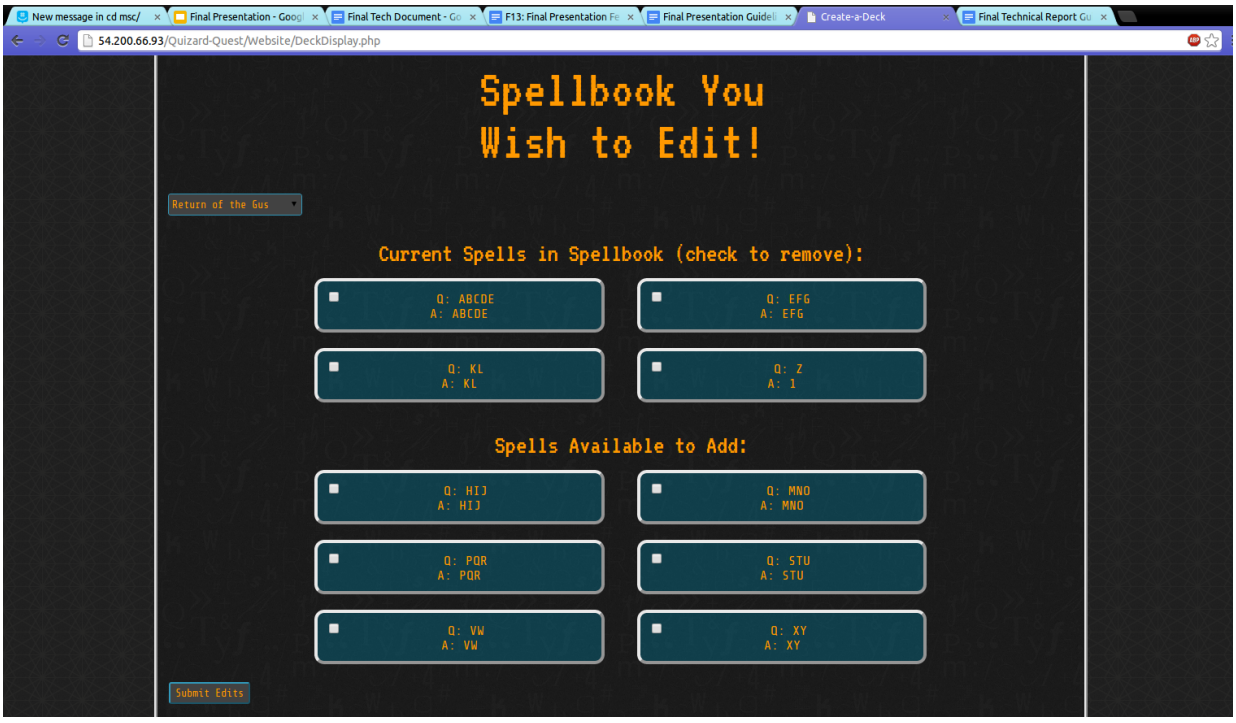


➢ **This is the shop where you can buy new avatars.**

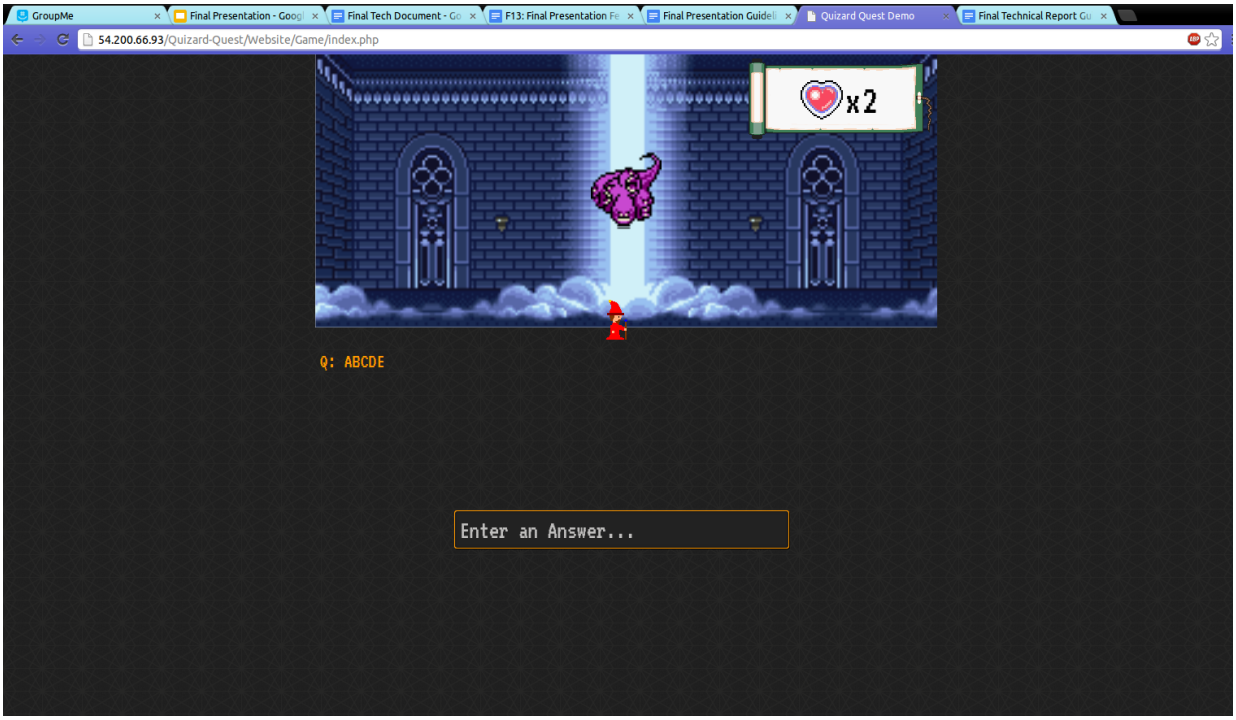


➢ **This is the page where you create a new Spell.**

➢ This is page where you create a new Spellbook from you Spells.



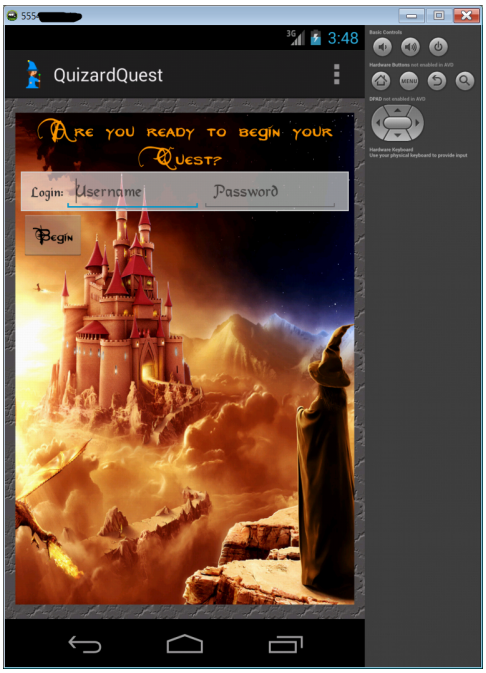
➢ This is the page where you can edit your Spellbooks.



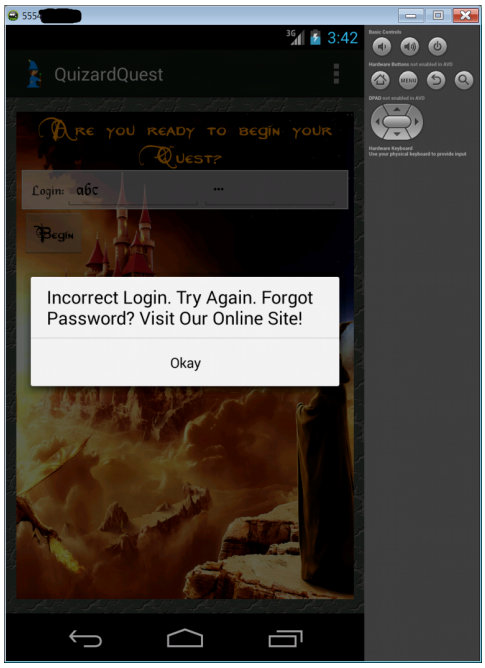
➢ This is the how the Quest begins.

➢ This is the end screen at the end of a successful Quest.

Android UI:

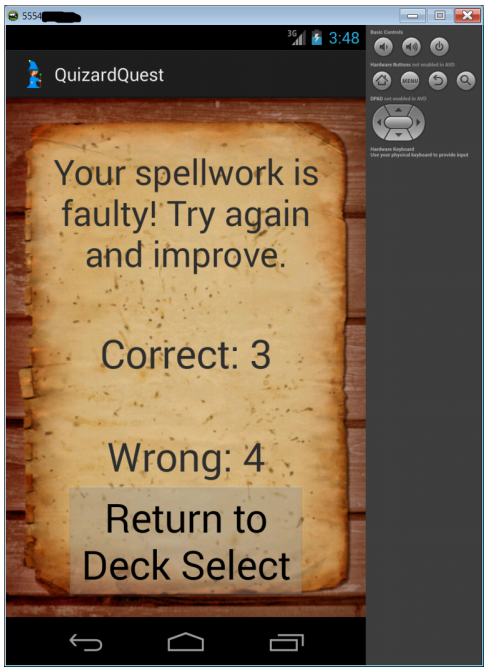


➢ **This is the login screen for the Android app.**



➢ **This is the app's response to an incorrect login.**

➢ **This is the screen in the app where the player selects which Spellbook to use.**



➢ **This is the app's response to a player failing to get 70% of the Spells correct.**

Testing:

How did we test: We gave our testers an explanation of the way the game worked out what they needed to do. Then we asked them to create 5 cards, a deck and to play through a challenge and asked to explain their thought process as they went through it.

Results:

Tester 1:

Creating a card: She felt that it was straight forward, but was a bit confused of where the card went after it was done.

Creating a deck: Had to look at it for a bit before she realized what she had to do. She then made a deck. " All the cards popping up at once and the small text box for the name. It confused me a little bit because I was not sure what was going on."

Going through a challenge: The tester said that the dungeon entrance page was straight forward but she did not know what each dungeon mode did. so she just chose training. Going through the dungeon she said she did not know how to submit her answer. " The dungeon was really cool but I was confused on the first question because I did not know how to submit my answer to continue."

Tester 2:

Creating a card: He felt that creating a card was intuitive. The inputs really guided him in knowing exactly what to do. The pop-up reassured him that the card was created.

Creating a deck: Made a lot of cards so his create-a-deck was really full. Said that a filter or card search would really be useful. Felt overcrowded with so many cards displayed. Scrolling was annoying and so was searching for cards. " I am looking for one of the cards I just added. It would be useful I could just filter to only math questions."

Going through a challenge: Tester said the front page was easy to understand but the lack of description of each challenge mode really threw him off. He suggested put descriptions of each game mode somewhere because that was frustrating. Going through the challenge was really easy and he said that he enjoyed the monsters. “I chose quest mode because I could assume that it was the medium difficulty. The dungeon was awesome, answering questions and seeing the different monsters really made it feel like I was playing a game and not studying.”

In-Class Test Group Findings:

Since the other group was also doing this project they looked for ways to break our website the way regular users could not. They found many errors we overlooked. Over password resetting, errors editing the decks, overflow of characters, they helped us find those errors that had to be brought to surface, as well as make some edits to the website to make it easier to use. We added some hints and changed some text to make it easier for a user to understand.

As a Test Team:

We began with aggressive attempts to break the other teams project. We quickly found many errors the other team had not considered or believed were already dealt with. Most of the errors were with critical functionality, but the other team was slow in resolving issues. We never really got beyond basic functionality testing because there was nothing more available to test.

Team Reflection:

Challenges:

Web Challenges:

- One of the major challenges was learning how to dynamically load data from the databases. Our Website required a lot of pulling data based on what users entered, so we had to know how to make AJAX calls and XMLHTTP request to get the data for the user without having to refresh the page. For example: in the deck edit page, we have to be able to load the deck the user wants to edit based on which deck the user selects.
- Another of the major challenges was the API middle layer. We did not want to use a framework so we constructed our own. We made a php page with all of the function that we needed to interact with the database and then we made individual php functions that called each function so that we could get the data to our website and app.
- Another huge challenge pivotal to our success was figuring out to create a video game and to make sure it was playable and fun. We had to learn some of the principles of game design, such as the game loop and animation, and build the game in a way so that it would run at a good speed. Even after putting all of the resources together, we needed to integrate the game and the main page, which required AJAX and PHP function calls to read in the cards and ensure that the gold and experience earned wouldn't be lost.

Android Challenges:

- One of the biggest challenges in developing the Android Application was creating an API level to interact between the Database and the Application itself. While we had learned about connecting Android to the internet in the Graphical User Interface class, and we learned the PHP in Databases--connecting the two required a high level of cooperation between people focusing in both fields. Errors, when encountered, had to be traced to either the Server or Client and this presented some difficulty in narrowing down the exact locations of errors.

Project Take Aways:

Completing this project lead us to understand better the difficulty in designing and building a fully-fledged web application from scratch and the work ethic required to push through to the end without giving up. Not all of us expected this project to be so thoroughly challenging and time-consuming even in what seems like a basic feature in a website: Login. We learned that there is much more power in JavaScript and JQuery that we have yet to learn how to unlock. We have learned that Android is vast environment with massive capabilities. We have learned that there is still much to be learned to become masters of web development.

We found that debugging (as always) takes up most of the time spent on the project, and, through this, we learned perseverance and ingenuity in solving problems that we could not at first understand. Most importantly, we realized that if we wanted to learn what we needed there was no time to wait around for answers to come floating by. There is no hand holding in the professional world: You must strive for answers and understanding.

None of us think there is anything we could really change from what we already did this semester-the amount of work we put in is so substantial. Perhaps the one thing that would have helped us would have been to focus on the soft skills, such as working with a team and keeping up with the workflow. Many of the lessons we learned through the nightmare of Custom Cupcakes were essential to our development as a group. However, if we knew at the start of this project all that we learned while completing this project, we would have moved with a pace to finish most, if not all, of our 2.0 features listed below.

2.0 Features:

1. Allow users to friend each other and share decks.
2. Allow more items and features to be bought in the store, for example:
    a. Different themes for the website
    b. Additional avatars.
    c. Different heart icons.
3. Implement a rating system for cards and decks.
4. Allow Facebook Login through the android app.
5. Add achievements to be unlocked through gameplay.
6. Allow users to create cards to immediately go into a new deck.
7. Link the rest of the game functionality and modes to the Android Application.

Data Dictionary:

- The player table contains most of the data associated with a user, excluding stats, settings, achievements, and their security question.
    - **userID** is an INT that is AUTO_INCREMENTED and is the PRIMARY KEY. This is used to identify a user between tables quicker than using varchars.
    - **username** is a VARCHAR(50) that is a UNIQUE KEY and CANNOT BE NULL. We do not want users to have the same username, so it is unique.
    - **password** contains the unsalted hash of the user's password. It is a VARCHAR(256) and CANNOT BE NULL
    - **salt** is the salt to the user's password. It is a VARCHAR(256) and CANNOT BE NULL.
    - **email** is a VARCHAR(50), CANNOT BE NULL, and is a UNIQUE KEY.
    - **fName** is the user's first name. It is a VARCHAR(50) and CANNOT BE NULL.
    - **lName** is the user's last name. It is a VARHCAR(50) and CANNOT BE NULL.
    - **grade** is the user's academic level. It is a VARCHAR(50).
    - **gender** is the user's gender. It is a VARCHAR(1).
    - **permissions** tell whether the user is an admin or not. It is an INT and DEFAULTS TO 0.
- The decks table is a deck that contains the name of the deck, along with its own ID and its user's ID.
    - **deckID** is an INT that is AUTO_INCREMENTED and is the PRIMARY KEY. This is used to identify a deck within the deckCards table
    - **userID** is an INT and is a FOREIGN KEY to userID in the player table. We have this key cascade on update and delete so that if a user's account is deleted, their decks will be removed as well.
    - **name** is the name of the deck given by the user. It is a VARCHAR(50) and CANNOT BE NULL.

- The cards table contains all of the information assigned to a card, except for which deck(s) it belongs to.
    - **cardID** is an INT that is AUTO_INCREMENTED and is the PRIMARY KEY. This is used to identify a card within the deckCards table.
    - **userID** is an INT and is a FOREIGN KEY to userID in the player table. We have this key cascade on update and delete so that if a user's account is deleted, their decks will be removed as well.
    - **question** is the actual question that will appear on the card. It is a VARCHAR(512) and CANNOT BE NULL.
    - **answer** is the answer the user must supply to be correct. It is a VARCHAR(512) and CANNOT BE NULL.
    - **category** is an INT and CANNOT BE NULL. An int is used to store category since we predefine the categories users can use.
    - **subCategory** is a VARCHAR(32). This is a category user's can use to specify what kind of question the card is inside of the broader categories.
    - **difficult** is an INT and CANNOT BE NULL. Similar to category, an int is used to represent the 4 different difficulties the users can choose from.
    - **rating** is an INT and DEFAULTED TO 0. On later iterations, this would represent the net number of "upvotes" to "downvotes" given by other users.
- The **deckCards** table simply connects cards and decks together since multiple decks can have the same card, and multiple cards are in the same deck.
    - **deckCardID** is an INT that is AUTO_INCREMENTED and is the PRIMARY KEY. This is used to to help identify the different tuples between deckID and cardID
    - **deckID** is an INT that is a FOREIGN KEY to deckID in the decks table. This is cascaded on update and delete in case the user decides to delete the deck.
    - **cardID** is an INT that is a FOREIGN KEY to cardID in the cards table. This is cascaded on update and delete in case the user decides to delete the card.

- The **options** table holds the various options that user can choose from.
    - **avatar** is a VARCHAR(100) and DEFAULTS TO the file location of the default avatar. This will hold the file location to whatever avatar the user chooses to use.
    - **cardBorder** is a VARCHAR(100). In later iterations, this is going to hold the file path to different background to cards that users could buy from the shop.
    - **bgColor** is a VARCHAR(10). In later iterations, this is going to hold the RGB value of backgrounds of the entire website that the user could buy in the store.
    - **userID** is an INT and is a FOREIGN KEY to userID in the player table.
- The **stats** table hold the different amounts of experience a user has in each category, along with total experience, and the amount of gold
    - **userID** is an INT and is a FOREIGN KEY to userID in the player table.
    - **mathExp** is an INT that DEFAULTS TO 0. This represents experience in the math category.
    - **sciExp** is an INT that DEFAULTS TO 0. This represents experience in the science category.
    - **ssExp** is an INT that DEFAULTS TO 0. This represents experience in the social studies category.
    - **engExp** is an INT that DEFAULTS TO 0. This represents the experience in the english category.
    - **langExp** is an INT that DEFAULTS TO 0. This represents the experience in the foreign language category.
    - **totExp** is an INT that DEFAULTS TO 0. This represents the total experience the user has gained.
    - **gold** is an INT that DEFAULTS TO 100. This represents the total amount of gold the user has.

- The **achievements** table in future iterations will keep track of which achievements a player has earned.
    - **userID** is an INT and is a FOREIGN KEY to userID in the player table.
    - **perfectGame** is an INT that DEFAULTS TO 0. This represents if the user has earned completed a without any incorrect answers.
    - **closeCall** is an INT that DEFAULTS TO 0. This represents if the user finished a save the world mode with less than 10 seconds left on the timer.
    - **mathQuizard** is an INT that DEFAULTS TO 0. This represents whether a user has gained enough experience to qualify as a math quizard.
    - **sciQuizard** is an INT that DEFAULTS TO 0. This represents whether a user has gained enough experience to qualify as a science quizard.
    - **ssQuizard** is an INT that DEFAULTS TO 0. This represents whether a user has gained enough experience to qualify as a social studies quizard.
    - **engQuizard** is an INT that DEFAULTS TO 0. This represents whether a user has gained enough experience to qualify as an english quizard.
    - **langQuizard** is an INT that DEFAULTS TO 0. This represents whether a user has gained enough experience to qualify as a foreign language quizard.
- The **securityQuestions** table holds the question and answer that the user used for password recovery.
    - **userID** is an INT and is a FOREIGHN KEY to userID in the player table.
    - **questionID** is an INT and CANNOT BE NULL. The ID corresponds to different security questions that we provide.
    - **answer** is a VARCHAR(32) and CANNOT BE NULL. The answer is the answer to the security question.