

University of Colorado Denver
College of Engineering and Applied Sciences
Department of Computer Science and Engineering

Team: Pluvali

Project: Coping Skills Game

Members:

Alexander Davis

Steven Kosovich

Timothy Leikam

Gregory Martini

Instructor:

Dr. Debra Parcheta

Clients:

Katie Taliercio, DDRC

Pamela Martien-Koch, DDRC

Eric Koch, DDRC

Table of Contents:

Overview:	Page 3
Feasibility:	Page 4
Market Analysis:	Page 5
Requirements:	Page 6 - 7
Specifications:	Page 8 - 10
Modeling:	Pages 11 - 14
Changes in Requirements:	Page 15
Domain Analysis:	Page 16
Risk Analysis Management:	Page 17
Project Planning:	Page 18
Cost Analysis:	Page 19
Schedule for Development and Deployment:	Page 20 - 23
Software Quality and Assurance:	Page 24
Work Breakdown Structure:	Page 25 - 26
Software Design:	Page 27
Architectural Design:	Page 28
Interface Design:	Page 29
Detailed Design:	Page 30 - 34
Implementation Plan:	Page 35
Testing:	Page 36 - 37
Delivery/Instructions:	Page 38 - 51
Maintenance Plan:	Page 52
Source Code:	Page 53 - 113

This page intentionally left blank

Overview:

This project is a game designed to help people with disabilities learn to cope with difficult everyday life situations by choosing the best situations in the game.

As the user plays the game, they will be presented with a variety of problems. Each problem will have three different coping techniques that the user can choose from as a way to deal with the situation. All of the possible coping techniques are correct as we are not aiming to punish the user, only show them new techniques that they can learn and use in everyday life.

As the user plays the game they will earn tokens. These tokens can be used to unlock aesthetic customization options for the user (background colors, text colors, etc). This is a strategic approach that we hope will encourage the user to play the game multiple times in order to gather enough tokens to unlock what they desire, all the while being exposed to more and more coping techniques that they will learn and use.

This web game was created using HTML, CSS, Python, Django, YouTube API, and SQLite. HTML and CSS were used to design and format the web pages, and YouTube API was used for the videos. In addition, the web pages contain Python code which is used to interact with the Django framework. This framework gives us a way to get data from our SQLite database, as well as update/store data.

Feasibility:

A FEASIBILITY STUDY ON COPING SKILLS GAMES/APPS FOR SENIOR DESIGN I

EXECUTIVE SUMMARY

A feasibility study on our client's project was conducted only based on the description of the project submitted by the client. Through a careful analysis of the project some of the results have been concluded such as time spending, skills, technology, and cost.

TIME

Time depends on how much content is going into the project. The project seems feasible time wise for a single game/app with text based scenarios as mentioned in the description. Needing to make it available cross-platform (Android, iOS, PC) may change feasibility of having a working product on all platforms.

SKILLS

Programming a game on any system and linking videos are completely feasible. If the client wants us to design the questions for the game, we would not have the skills to do this properly. If the client wants us to design or take the video for the game, we would not have the skills to do this properly either. If the client is prepared and needs software engineers, the skills aspect is feasible.

TECHNOLOGY

The technology required for this project is 100% feasible.

COST

Cost would depend on how prepared the client is. A procedural game would be the cheapest to host and deliver to the end users, and would be entirely within the scope of computer science but probably the least effective at emulating real life scenarios. Generating the video or animation is beyond our skills or time constraint, but hosting the videos in a manner that makes them readily available and responsive for more than just a handful of users is cost prohibitive.

CONCLUSION

This project seems feasible as long as the client is prepared with the content we need to use alongside our application.

Market Analysis:

There is currently nothing on the market quite like this. This game is also made especially for the Developmental Disabilities Resource Center (DDRC), and will have a limited number of users. It will be free to play, and the only way to find out about it is from the DDRC.

This game will have no to no impact on the market, since it will only be available to the DDRC and their users. The game is not intended to be sold.

Requirements:

Coping Game Requirements

- I. Build a web app game that is accessible from desktops, phones, and tablets
- II. The game must focus on introducing new coping techniques to the players for use in everyday life situations
- III. Players must be able to make their own account if they don't already have one
 - A) Players can choose their username and password
 - B) An email will be assigned to their account
- IV. The game will be a "pick your own adventure" style game with scenarios for the player to choose from
 - A) Scenarios will provide the player with a series of problems
 - B) Each problem will have three valid coping techniques to choose and learn from
 - C) Each problem will make use of videos, images, and text to display the problem and the suitable coping techniques
 - D) Answering each question will reward the player with tokens they can use in the game store
- V. The player will be able to customize their web game to a certain extent
 - A) Players will be able to purchase themes and user pictures with tokens they have accumulated
 - 1) Themes will be specific background and text color combinations that will apply to every page of the web game
 - a) The color combinations must contrast and be easily readable
 - 2) User pictures will change the picture associated with the player's account
- VI. A player's progress will be saved
 - A) Their token amount will be regularly updated and saved
 - B) What theme and user picture they are using
 - C) What themes and user pictures they have unlocked
- VII. Administrators will have certain permissions that regular players don't possess
 - A) Add, edit, and remove scenarios and/or problems
 - B) Add, edit, and remove player accounts
- VIII. Administrators can play the game as if they are a regular player

- IX. The game must be intuitive and easy to use, and must use pictures and icons for easier navigation of the game
- A) Buttons to navigate the game will have easily recognized symbols
 - B) Theme colors must have a good contrast ratio for readability

Specifications:

The game will be implemented using SQLite, Django, Pillow, HTML, and CSS. The SQLite database will contain user information, such as their username, password, email, player points, and preferences. We will then implement a user interface through Django, HTML, and CSS to interact with the database providing a choose-your-own-adventure style game. A content creation tool will be made to handle making new levels. This will be done using the languages above. The game will save content progress through the use of tables in the database to allow the player to reuse any previously purchased items from the store.

The choices we made are represented by an asterisk.

Database Considerations:

*SQLite:

pros: Easy start-up and great testing. comes with Django.

cons: No user management nor does it have very many performance optimization features.

My Structured Query (MySQL):

pros: Portable across different DBMS. Designed with a focus on the web. Large amount of support and reliable databases, typically.

cons: Not as mature as other relational database management systems (Didn't start as a RDBMS but later changed to one; regardless, it shouldn't affect a project of our scale). Functionality can be dependent on add-ons (again, shouldn't affect us).

Java Database Connectivity (JDBC):

pros: Portable across different DBMS. Solid performance.

cons: Should have a good understanding of Java

Open Database Connectivity (ODBC):

pros: Portable across different DBMS.

cons: Different drivers are needed for different problems. When so many drivers are being used, the overhead of managing them is also increased (shouldn't affect a small project like this).

Database Interface Considerations:

*Django:

pros: Easy to learn. Able to build custom frameworks for only what we need. More maintained than PHP, and easier to read.

cons: Would have to show the future support team of this program how to familiarize themselves with the custom framework.

PHP:

pros: We have a little previous experience with it already. Easy to learn.

cons: Maintainability can get complex.

Java Servlets:

pros: Functions like standard cgi languages such as php and perl. Effectively, it is a java applet that runs on the server instead of on the client. Because it is based on java, it integrates better with existing classes and it is extendable. A little faster than php and perl.

cons: Java is more complex than php or perl. Harder to modify on a whim than php or perl.

Rails:

pros: Slight edge over Django for web development.

cons: Has a slow learning curve.

Game Framework Considerations:

*HTML:

pros: Easy to learn. Compatible everywhere.

cons: Can't support highly dynamic interfaces.

JavaScript:

pros: Easy to learn. Can handle some features HTML can't.

Cons: Needs some HTML.

Java:

pros: More processing power.

Cons: iPhone doesn't support Java.

Unity:

pros: Strong environment with plenty of support.

cons: Wouldn't work for mobile systems. Too much for a project of this scope.

User Progress Tracking Considerations:

*Database Tables:

pros: Simple to set up, and future proof.

cons: If too many features are added, it can be unnecessarily large.

Data Strings:

pros: Would have easy to set up and quick load times. Not device dependent.

cons: Creating a way to have user made scenarios auto implement this system without the user making the data strings (hard to future proof).

Cookies:

pros: The browser already makes use of cookies, so this wouldn't be hard for the game to use as well.

cons: If a user deletes their cookies without intending to, it would delete their save game. Also, a user account would be linked to a specific device.

Linked with IP Address:

pros: Each IP is unique, so it would be easy to distinguish each player.

cons: Multiple people couldn't play from same network mask.

Resources:

The database will be tested on our local computers, and safe releases will be put up on a server set up at the DDRC.

Editor Considerations:

*Notepad++:

pros: Free, uses Scintilla (a powerful editing component), and adaptive to most languages.

cons: not a full compiler (but does work in conjunction with Django).

Sublime Text Editor:

pros: Uses Intellisense (a powerful editing component), and adaptive to most languages.

cons: not a full compiler (but works in conjunction with Django). Not free.

Eclipse:

pros: Free and powerful IDE.

cons: More powerful than what we need.

Version Control Considerations:

*GitHub:

pros: We are familiar with it, and can select who is allowed to collaborate on it.

cons: No free private repositories.

Concurrent Version System:

pros: Free and compatible across most platforms.

cons: One of the biggest design flaws with the current release of CVS is that it is very difficult to move a file to a different directory or rename it.

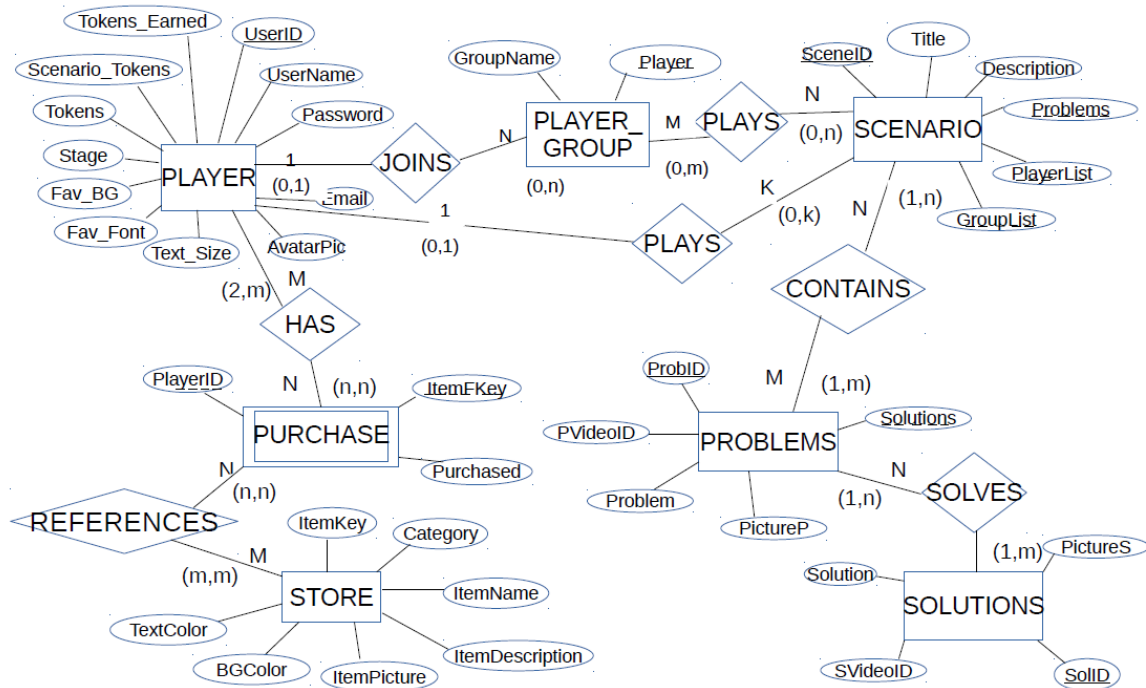
BitBucket:

pros: Free, lightweight, and create private repositories.

cons: Free for only five users (we have a total of six who would need access to it).

Modeling:

ER Diagram



Assumptions: PLAYER is for both admins and users going to play the game

PLAYER AvatarPics can be uploaded to database.

A PLAYER can play available SCENARIOS, but only one at a time (some SCENARIOS are PLAYER specific).

A PLAYER_GROUP is not required but can be used to help add PLAYERS to access SCENARIOS

Every PLAYER gets a default theme and picture PURCHASE from the STORE.

A PURCHASE is added for each STORE item for every PLAYER.

PURCHASES requires a PLAYER

An Admin (PLAYER) can edit any SCENARIO, but only one at a time.

A SCENARIO may have one or more PROBLEMS

A PROBLEM may have one or more SOLUTIONS

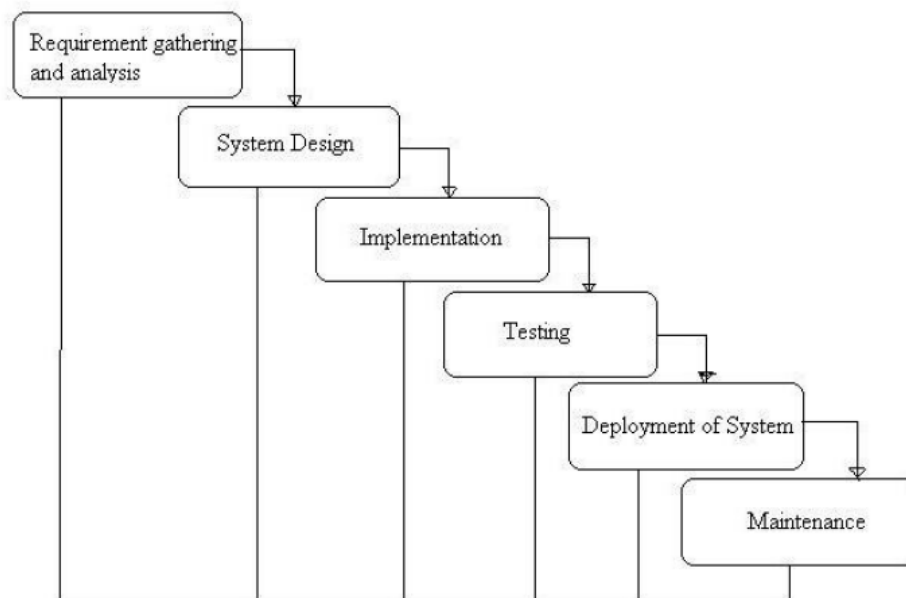
Software Design Model

Pluvali Development Model

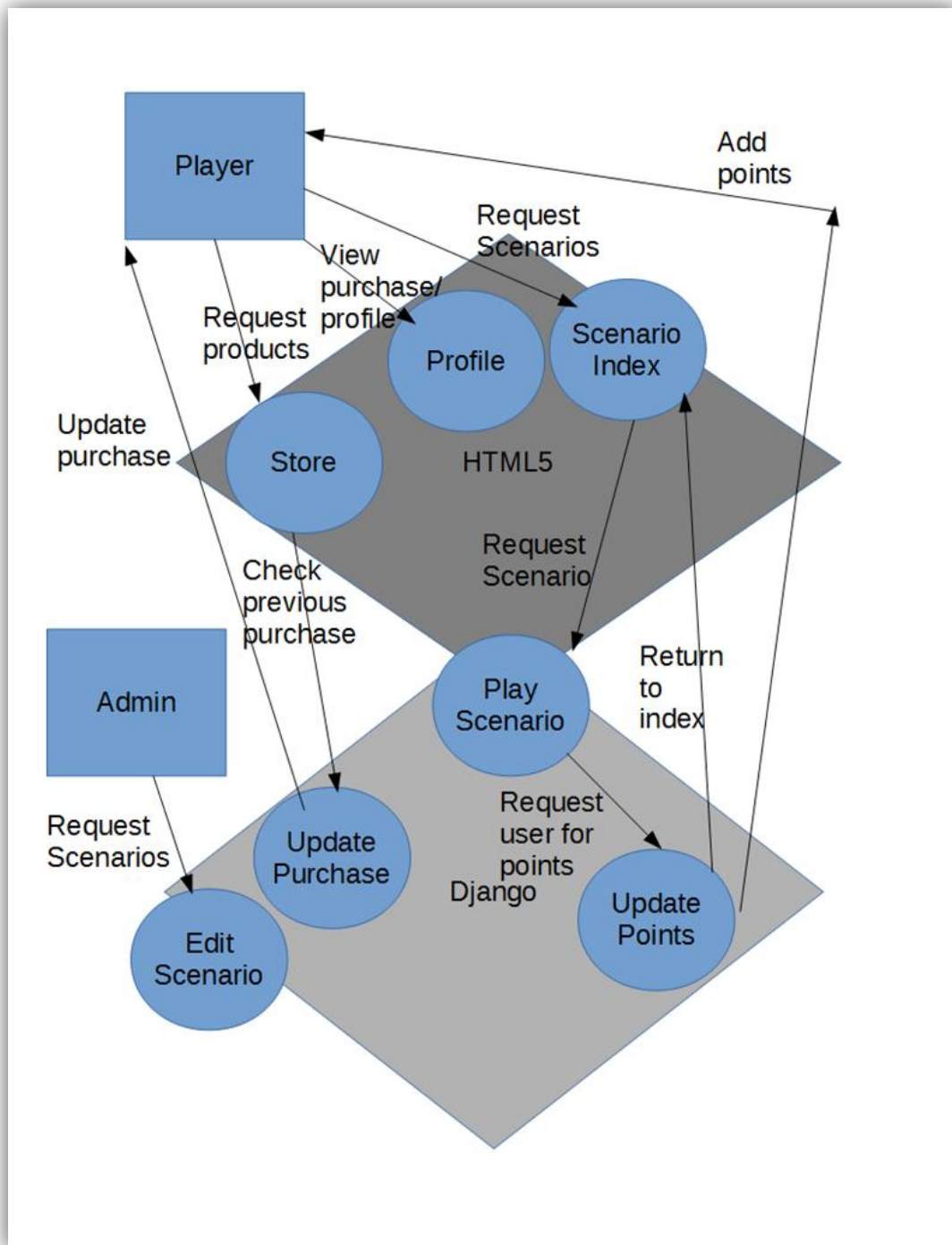
Alex Davis
Steven Kosovich
Greg Martini
Tim Leikam
Debra Parcheta
Katie Taliercio

Our team decided to build our prototype based on the Waterfall Model. We chose this model based on the simplicity of both the model and project. By breaking out the parts into distinct phases, there is no ambiguity in what members should be working on. The key disadvantages of the Waterfall Model are that there is no working software until late in the cycle, once an application is in the testing stage, it is difficult to go back and change implemented methods, there can be high amounts of risk and uncertainty, and it isn't a good model for complex and object-oriented projects. To overcome these, most are met by the fact that our project isn't too complex and the rest are fixed by our organized designs. As a team, we thought out each of the aspects of the game, how they would flow individually and between states, to prepare for development through launch.

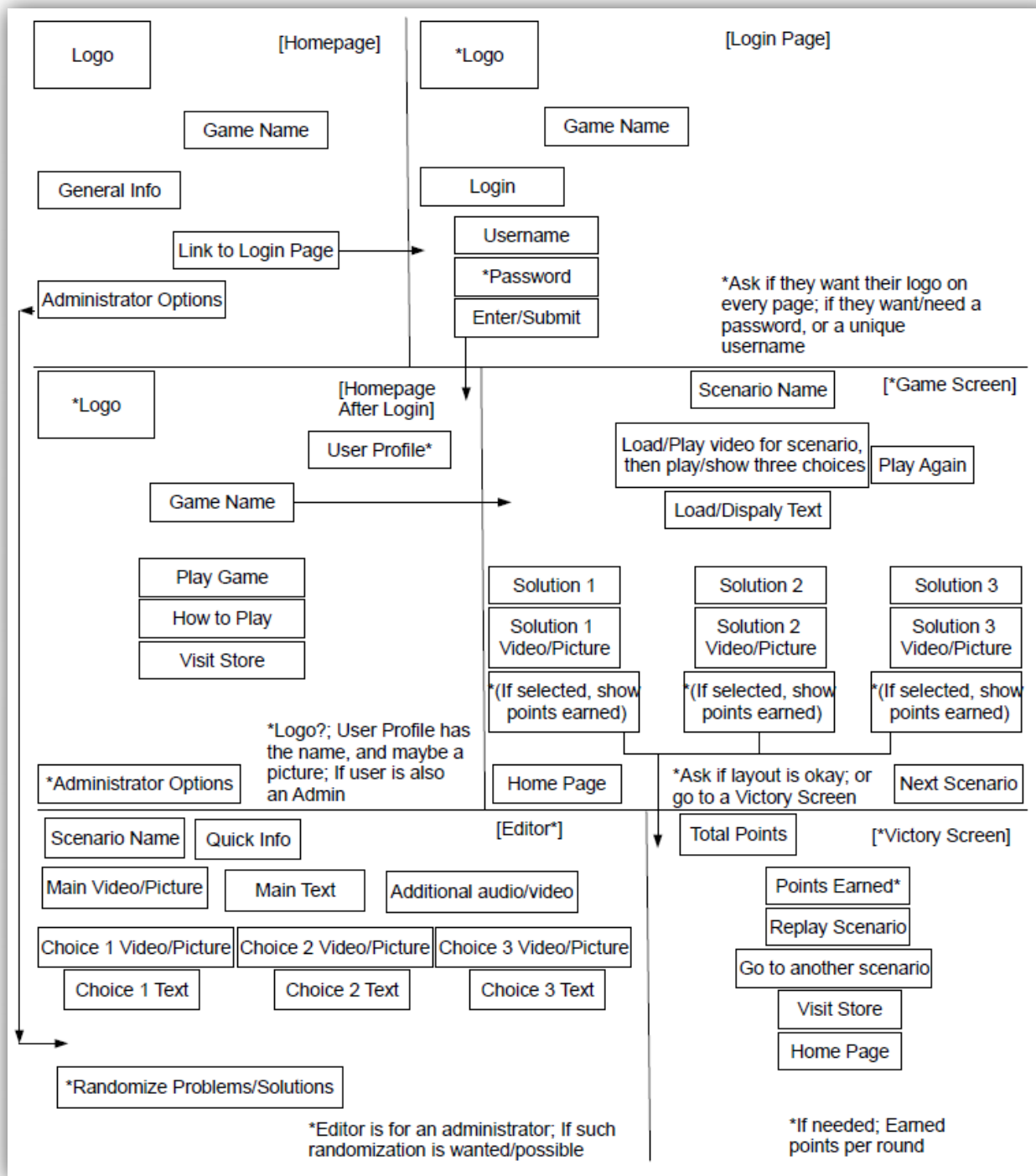
General Overview of "Waterfall Model"



Data Flow Diagram



HTML Pages Diagram



Changes in Requirements During Engineering:

The game went through a few minor changes in Requirements.

One of the desired features that wasn't initially realized in the beginning of this project's life was the ability to add new scenarios. This led to the Admin Page being a new requirement, so that it would be easy for someone with administrator privileges but limited technical expertise to make new scenarios as they pleased.

Another change was the use of JavaScript; originally, we decided not to use it, but as we progressed through the project, it was decided that it would be easier to implement certain features by using JavaScript than to try and use just HTML and CSS. However, in the second semester we decided that we would not need to use JavaScript, so in the end we did not use it at all in the final implementation.

Domain Analysis:

There are no existing products that exactly match what this one is, and there is also no reuse from a similar product. This project was built entirely from the ground up from scratch, without taking any code from anywhere else.

See page 13 in this document for a data flow diagram that shows the relationships of entities for this project.

Risk Analysis and Management:

There were no risks associated with this project. There is no time frame that the product must be delivered to, and it is a free-to-play product, so there is no financial risk whatsoever in making this product.

Project Planning:

The project started the planning phase in the beginning of September 2014, and continued until the end of the month.

See pages 6 -14 for the Requirements, Specifications, and models.

See pages 20 -23 for detailed timelines of the project for both semesters.

Cost Analysis:

Estimated:

The estimated cost for this project was very little. There is only the time that the members of the team had to spend on it, plus the cost of a small server.

Actual:

The actual cost was the same as the estimated cost, with the server as the only financial expense.

However, if the cost for labor was taken into account and the laborers were paid, the cost would be as follows (amounts in USD):

Four people at \$20/hr, at about 8hrs/wk each = \$640/wk

32 weeks of the project at \$640 = \$20,480 total for labor (assuming competitive CS intern rate of \$20/hr)

Schedule for Development and Deployment:

Initial:

The initial schedule for Development and Deployment is shown below in the original Timeline:

Team PLUVALI's Timeline

Project leaders: Greg Martini 9/9 – 9/29

Steve Kosovich 9/30 – 10/20

Tim Leikam 10/21 – 11/10

Alex Davis 11/11 – 12/9

Requirements: 9/24 - All members

Done: Presented by Greg.

Specifications: 9/26 - All members

Done: Presented by Steve.

Prototyping: 9/27 – 9/30 - All members

Setup/install MySQL database, Django, Notepad++.

Timeline: 10/2 - All members

To be presented by Tim.

Coding: 9/30 – 11/1 (coding and database framework should be finished by 11/1)

Web browser design - Steve and Tim

Main page; Login screen intro with links to level editor, Profiles, Game Screen, Help Page menu, User Preferences, and the Content Store.

Database design - Alex and Greg

Creating tables for User Profiles, Character Information for the game, plus Scenario Data.

Web page and database connectivity - All members

Using the databases' data to display questions to the user, accurately track and save a user's progress (preferences, points, and unlocked or "purchased" content from points earned), and verify if the user is an admin with access to the Level editor and Profiles or just a regular user who will play the game.

Prototype coding may continue after 11/1, but it should be mostly functional and complete by this point in time.

Client Review: 11/1 - All members

Presentation (if there is one) to be presented by: TBD

Project Management Review: 11/4 - All members

Presentation (if there is one) to be presented by: TBD

Project Schedule Review: 11/6 - All members

Presentation (if there is one) to be presented by: TBD

Next Semester planning: 11/4 – 11/6 - All members

Presentation (if there is one) to be presented by: TBD

Prototype Demo: 11/18 - All members

Presentation (if there is one) to be presented by: TBD

Client Review: 12/2 - All members

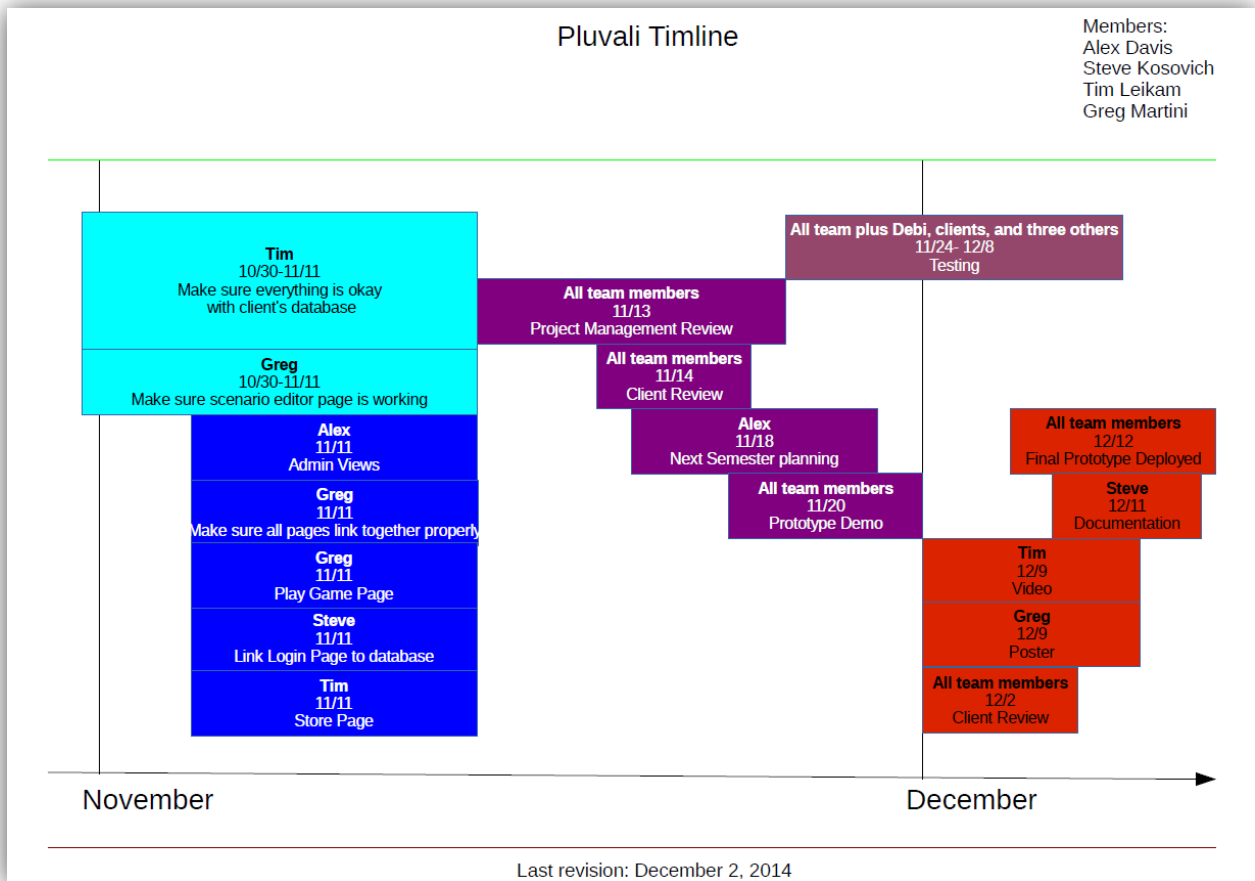
Presentation (if there is one) to be presented by: TBD

Final Prototype Deployed: 12/11 - All members

Presentation to be presented by all members

Actual:

The actual schedule for Development and Deployment from the end of October and on is reflected in this graphical updated Timeline (the dates for September above are the actual schedule dates), as shown in the next page:

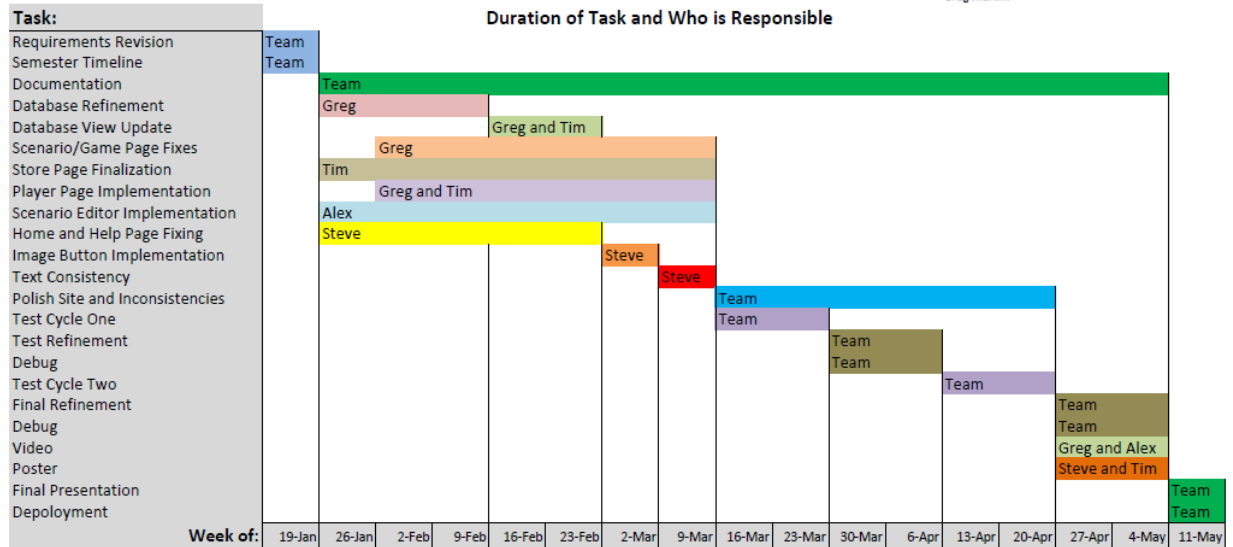


The Timeline for the second half of the semester is summed up in another graphical timeline, shown below:

Team Pluvali - Spring 2015 Timeline

Alex Davis
Steve Kosovich
Tim Leikam
Greg Martini

Debra Parcheta
Katie Taliercio



Software Quality Assurance Activities and Results:

Our team did our utmost to ensure that this project was of the highest quality that we could produce with the amount of time and recourses that were at our disposal.

The team did multiple tests on every single piece of the product, and anything that was found that needed to be fixed was worked on as soon as possible.

Whenever a defect in the software is found, we try our best to fix it through testing and debugging to the best of our abilities. Many problems that have occurred were fixed in this way, and we will continue to work on it to ensure that the final product is of the utmost quality.

Work Breakdown Structure:

The way that our team decided to split up the group was to basically have two sides: The HTML side, and the Django side. Steve and Tim did the HTML, and Greg and Alex did the Django side. This made it easier since both sides had two team members, and so that multiple things could get done at the same time.

Of course, many things in this project required other tasks to be completed first, so often people from one side would switch over and work on the other in order to accomplish the task that needed to be done.

All of the team members presented our progress in class, and helped make all the documentation, presentation slides, and diagrams. Below is a breakdown for each member:

Steve - Worked on coding some of the HTML and CSS, and made some of the slides such as the Requirements and the Testing Plan. He also made some of the diagrams such as the Pages Diagram and the final Timeline, and did a lot of the final documentation. He attended all meetings with the team and the client, and was the team lead from 3/17 - 4/20. He helped design the layout and structure of some parts of the project, and helped make the design of the poster. He spent an average of 8 hours every week working on the project for the first semester.

He also made various changes to the HTML, and added visual functionality like picture buttons to the game. He edited and looked over spelling, formatting, and general fixes to the game. Also, he did a lot of the documentation such as updated Requirements, Specifications, and the final documentation. He spent roughly around 5 hours every week in the second semester.

Greg - Worked mostly on initializing the Django database, setting up the models (tables), player views, urls, player registration and log in. As for documentation, he did first draft of the Specifications and Timeline and also made the E.R. diagram and software design model. Greg also helped with initial design of the project layout and of the posters. He was team lead from 4/21 - 5/16, and attended all of the team meetings. He spent on average 8 hours per week working on the project for the first semester.

He also added Python code to handle new instances of Store Items, and he edited the game, profile, admin, forms, and init files for errors and ease of use. He also changed HTML pages for errors and added video players using YouTube APIs. Finally, he edited parts of the documentation to reflect the changes done in the code. He spent roughly about 5 hours per week on average for the second semester.

Tim - Worked mostly on coding HTML and CSS for the project. He also made some of the presentation slides, and he also made the final video. For documentation, he made the testing

questions document, and also the Timeline in its presentation format. He attended all of the meetings with the team and the client, and was the team leader from 2/17 - 3/16. He spent an average of 8 hours a week working on the project in the first semester.

He also finalized the store page and handled all the errors associated with it. He also made any necessary changes to the database for the store page. And in general, he helped with bug fixes and finalized the profile page and store pages, and also helped format and add features to every page like the hover links. He spent about 5 hours a week working on the project on average for the second semester.

Alex - Worked on mostly brainstorming ideas as well as getting the admin page working. He also did occasional bug fixes from time to time as needed. He attended all meetings with the team and client (although one was through Skype), and was the team lead from 1/20 - 2/16. For documentation, he double checked the work to make sure there were not any errors and contributed verbally during team meetings. He spent an average of 6-8 hours each week on various aspects of the project in the first semester.

He worked primarily on the admin page, trying to make it simpler. He also participated in meetings and gave feedback towards functionality of other features. He worked about 2 hours a week on average for the project.

Software Design:

Design Concepts and Principles:

The concept and principles of our team's design was to use the Waterfall Model of Software Engineering, to ensure the amount of flexibility needed for a project of this scope. The Waterfall Model is shown and explained in more detail on page 12 of this document.

We also used many common tools and languages to implement our design, such as HTML, CSS, and Django. The reasons we chose these tools are explained in more detail on pages 8 - 10 of this document in the Specifications.

Reengineering:

This project was never completely reengineered, but parts of it were. For instance, the design of the web pages changed a few times and had to be redesigned. Also, the way the game was going to be played had to be reengineered, since in the design phase we didn't have a victory page after playing a scenario, so that had to be implemented.

Architectural Design:

The design of this project was based on a Django framework, supplemented by HTML pages.

Please refer to pages 11 - 14 of this document for all the diagrams related to the architectural design of this project.

Interface Design:

The design of this project was implemented using HTML and CSS. The HTML was used for the majority of the code and is the main part of the project. CSS was used in conjunction with HTML to add visual enhancements and improvements.

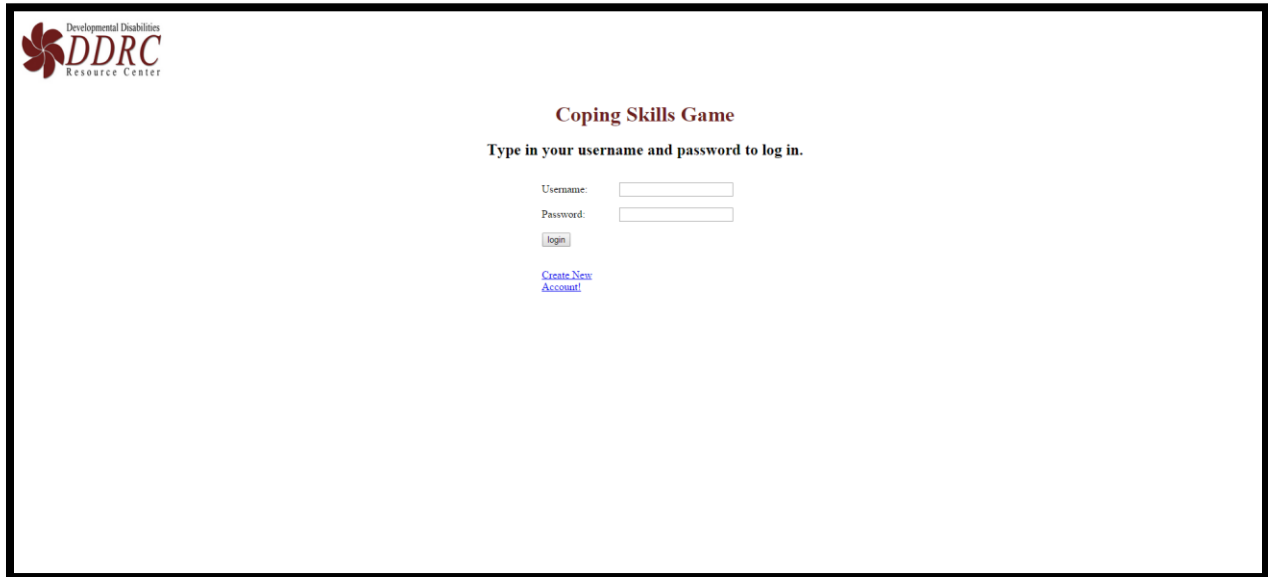
The interface was designed to be intuitive and easy to use. For users, there is a sign in page, then a homepage, a help page, a scenario page, a store page, and the game pages. When a game is completed, there is a victory page as well. So the interface is very simple with very few pages, so that almost anyone could use the product without much difficulty.

See page 14 of this document for an example of how the pages are laid out and connect to each other.

Detailed Design:

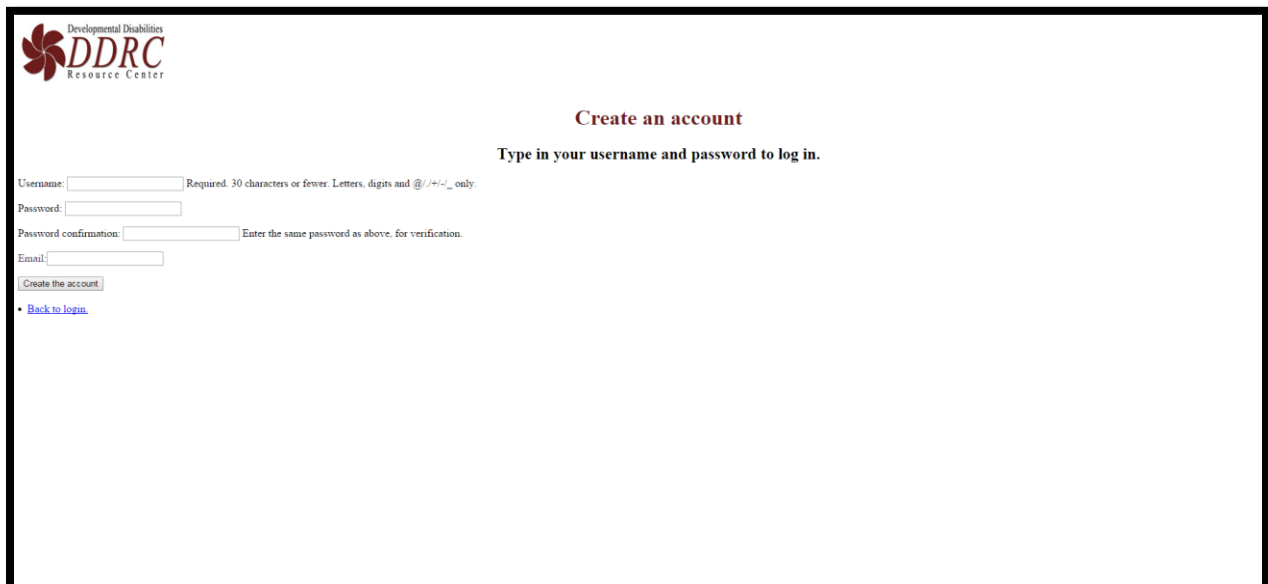
See pages 6 - 14 for the Requirements, Specifications, and Design Diagrams. The screenshots of the game's design are shown below:

Login page



The screenshot shows the login page for the Coping Skills Game. In the top left corner is the logo for the Developmental Disabilities Resource Center (DDRC), which features a stylized red flower icon and the text "Developmental Disabilities DDRC Resource Center". The main heading is "Coping Skills Game" in a bold, dark red font. Below this is the instruction "Type in your username and password to log in." in a standard black font. The login form consists of two input fields: "Username:" and "Password:", each followed by a text box. Below the password field is a "login" button. At the bottom of the form is a blue link that says "Create New Account!".

Create a new account page

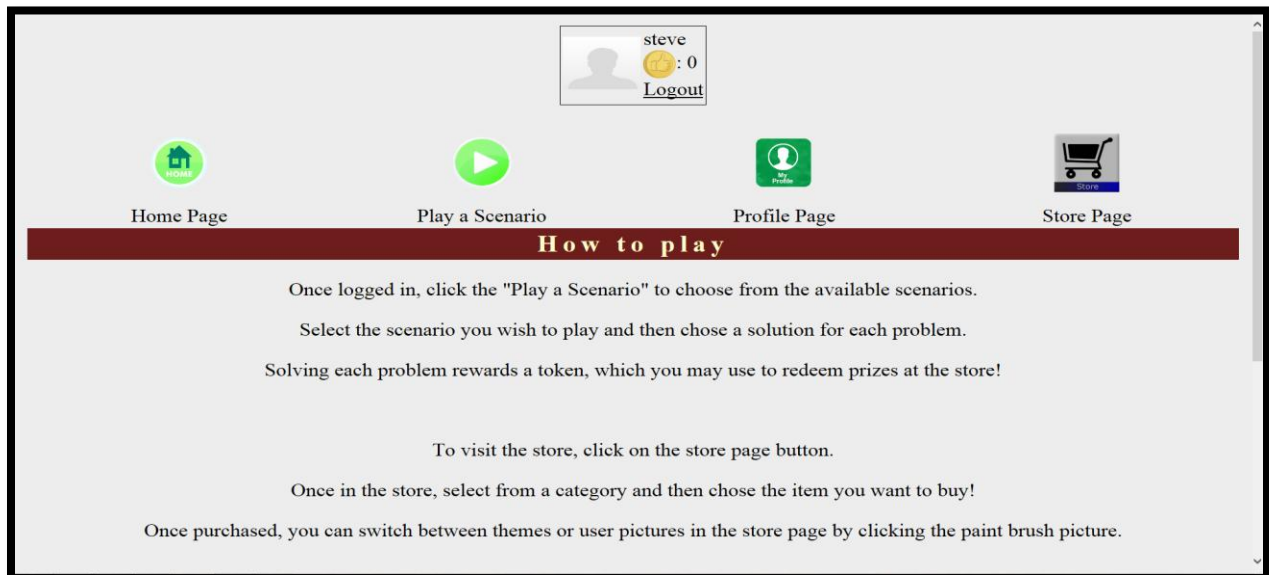


The screenshot shows the "Create an account" page. The DDRC logo is in the top left. The heading "Create an account" is in a bold, dark red font, followed by the instruction "Type in your username and password to log in." in black. The form includes four input fields: "Username:" with a note "Required. 30 characters or fewer. Letters, digits and @/+/+/- only.", "Password:", "Password confirmation:" with a note "Enter the same password as above, for verification.", and "Email:". Below these fields is a "Create the account" button. At the bottom left, there is a blue link that says "• Back to login."

Home page



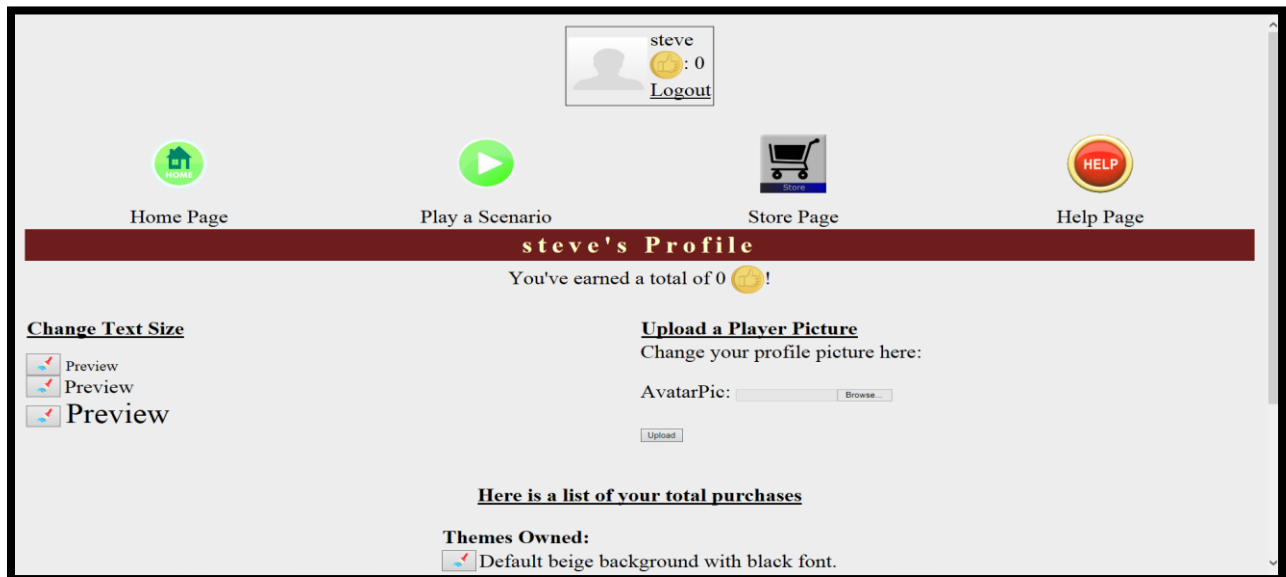
Help page



Store Page



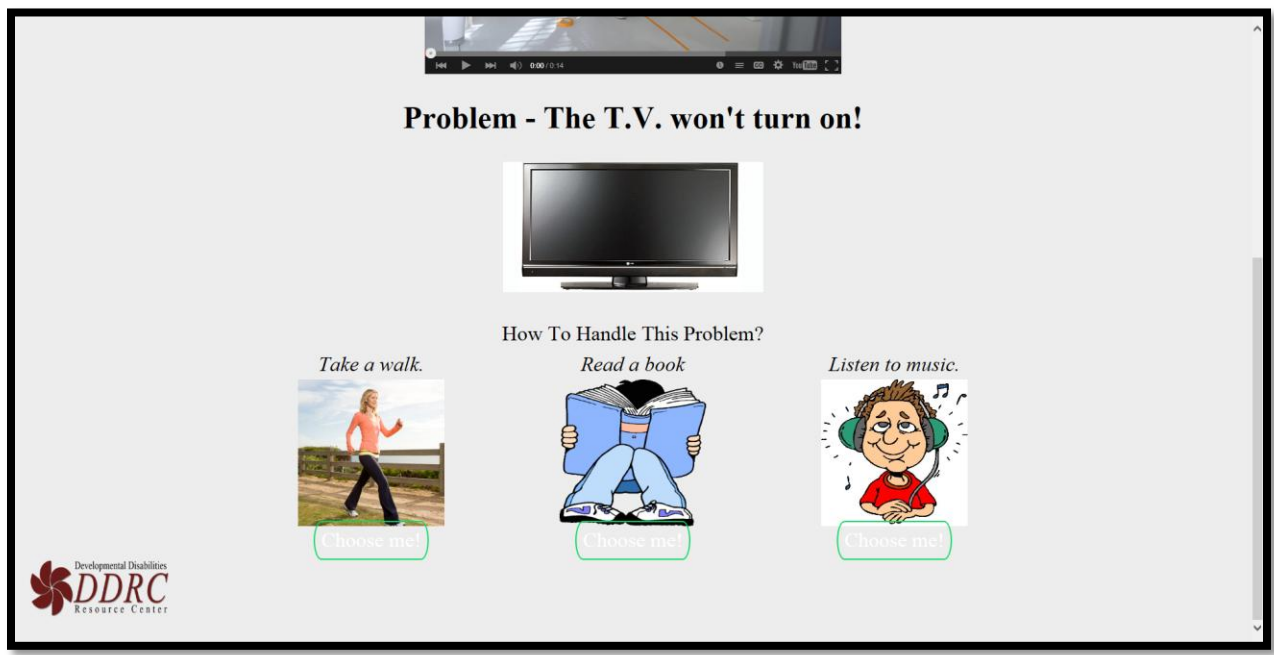
Profile page



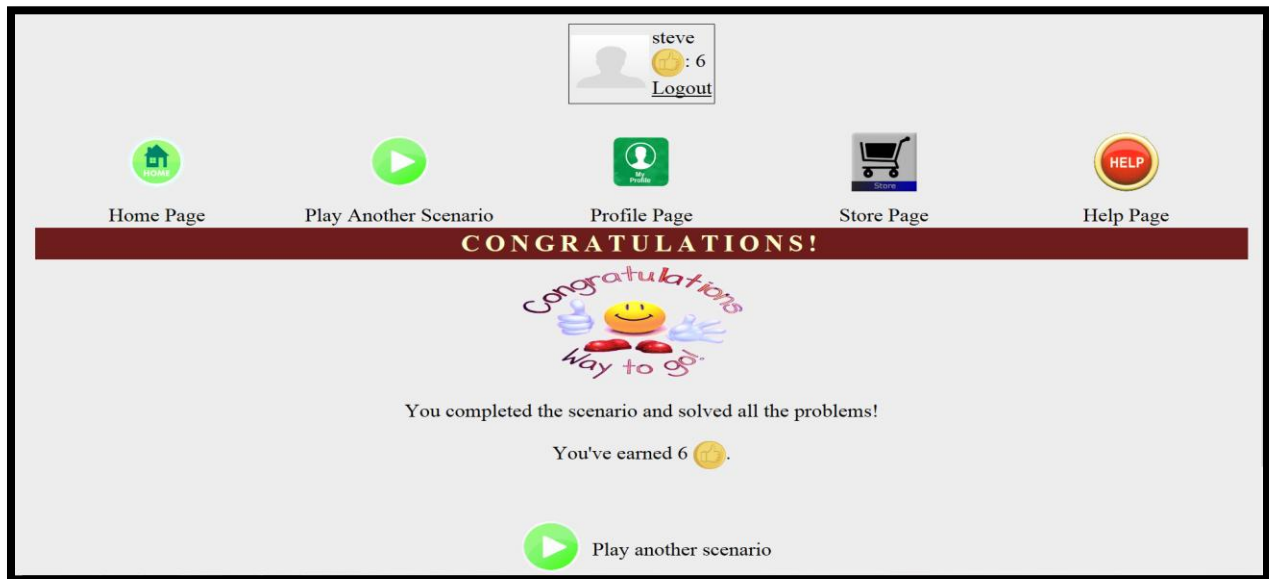
Game page



Example of a game scenario:



Victory page



Implementation Plan:

Initial:

Initially, we planned on having our product prototype ready to be implemented by November 18. For more information on what our schedule looked like initially, see pages 20 - 21.

We planned on having our final version ready by April 4

Actual:

However, due to unforeseen events, the product was finally ready for testing on November 25th, which was also the date that the prototype was implemented on. The prototype was implemented on November 26. See page 22 for the plan for the first semester of the project, and page 23 for the most recent version of our timeline, to see when we had things planned.

The actual final product was ready on April 26th.

Testing:

The team made a list of questions for the game, and sent them to our testers to ensure that everything worked as it was supposed to. Below is a list of the testing questions for the actual game:

Are you able to:

- Create a user account
- Log in using your accounts' credentials
- Log out successfully (located in the top of the screen)
- Navigate to and from the following pages:
 - Home Page (this is the page you're directed to when you log in)
 - Scenario Page
 - Store Page
 - Help Page
 - Profile Page
- Select a scenario to play on the scenario page
- Answer the problems by selecting a coping technique
- Verify that your token total increases when you answer questions (located in the top of your screen above the logout button)
- Verify that you are directed to the victory page when you complete a scenario
- On all pages, make sure that the links to the other pages work correctly
- In the store page, verify that different tables appear depending on what category you select on the left hand side
- With the "Themes" category selected on the store page, make sure that you can purchase new themes by clicking on the purchase button (will only appear if you have enough tokens to purchase a new theme)
- Verify that your theme stays the same after you log out and log back in using your accounts credentials
- Verify that you can switch your picture in the store page by clicking on the paintbrush apply button next to the pictures you own
- Verify that you can switch your theme in the store page by clicking on the paintbrush apply button next to the themes you own
- Verify that your theme works on every page
- Verify that you can upload a picture from your computer in the profile page
- Verify that after uploading a picture from your computer, you can switch back to the default picture in the profile page
- Verify that you can switch your theme on the profile page by clicking on the paintbrush apply button next to a theme you own
- Verify that you can change the font size of the site by clicking on a paintbrush apply button next to a desired font size

All of the questions above had a comment section, where feedback could be given to improve to product. Most of the features worked, and the ones that did not work either completely or partially are now fixed to our knowledge.

The team also made a list of questions for the administrator part of the game, and sent them to our testers to ensure that everything worked as it was supposed to. Below is a list of the testing questions for the administrative part of the game:

- Log in as an admin
- Create a Scenario
- Create a Solution
- Create a Problem
- Familiarize yourself with how you would create a Player (you do not need create players, just know how to do it in case you ever want to)
- Familiarize yourself with how you would delete a Player (you do not need delete players, just know how to do it in case you ever want to)
- Familiarize yourself with Player Groups (you do not need to do anything, just know how to do it in case you ever want to)
- In Purchases, test out setting the Purchased Boolean (see if the “Owned” box is checked) to give or remove player purchases by clicking on someone's purchases (for example, player1-YellowPurple)
- Delete a Scenario (NOTE: If you delete a scenario you will not be able to get it back easily. Don't delete the television problems, as it is the default scenario)
- Add/delete Store items, except the default store items (add a store item yourself before deleting one)

Again, all of the questions above had a comment section, where feedback could be given to improve to product. Most of the features worked, and the ones that did not work either completely or partially are now fixed as far as we know.

Delivery:

Installation:

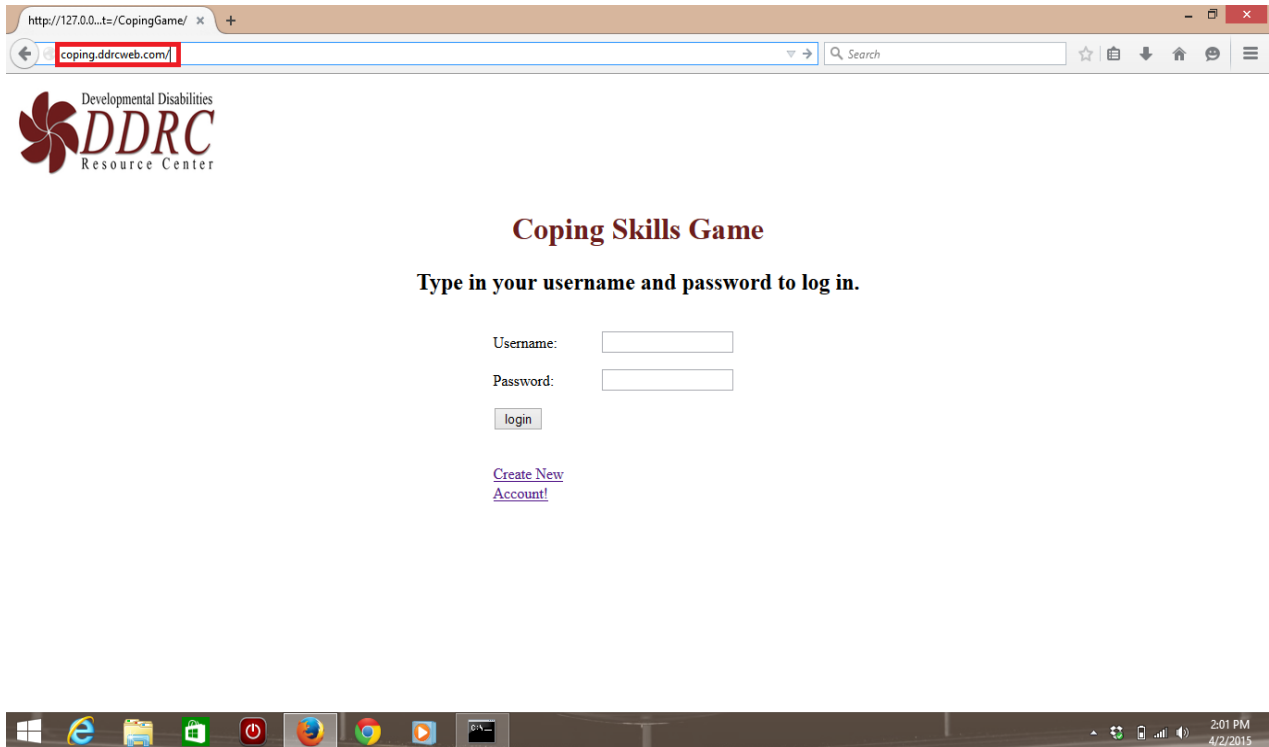
Team Pluvali has installed all of the most current components of the project on to DDRC's server. Although this took longer than anticipated and had a few issues, it is now fully functional and ready to use. The user does not have to install the product on their own device, since the product can be used using only a device and an Internet connection.

User Manual:

- Go to <http://coping.ddrcweb.com>
- Create a new user profile, or sign in with an existing one
- Navigate to the help page, store page, profile page, or scenario page as you like (if you are an administrator, you can also get to the administrator page through the home page)
- Play scenarios and earn points to spend on aesthetic upgrades for your profile
- Enjoy!

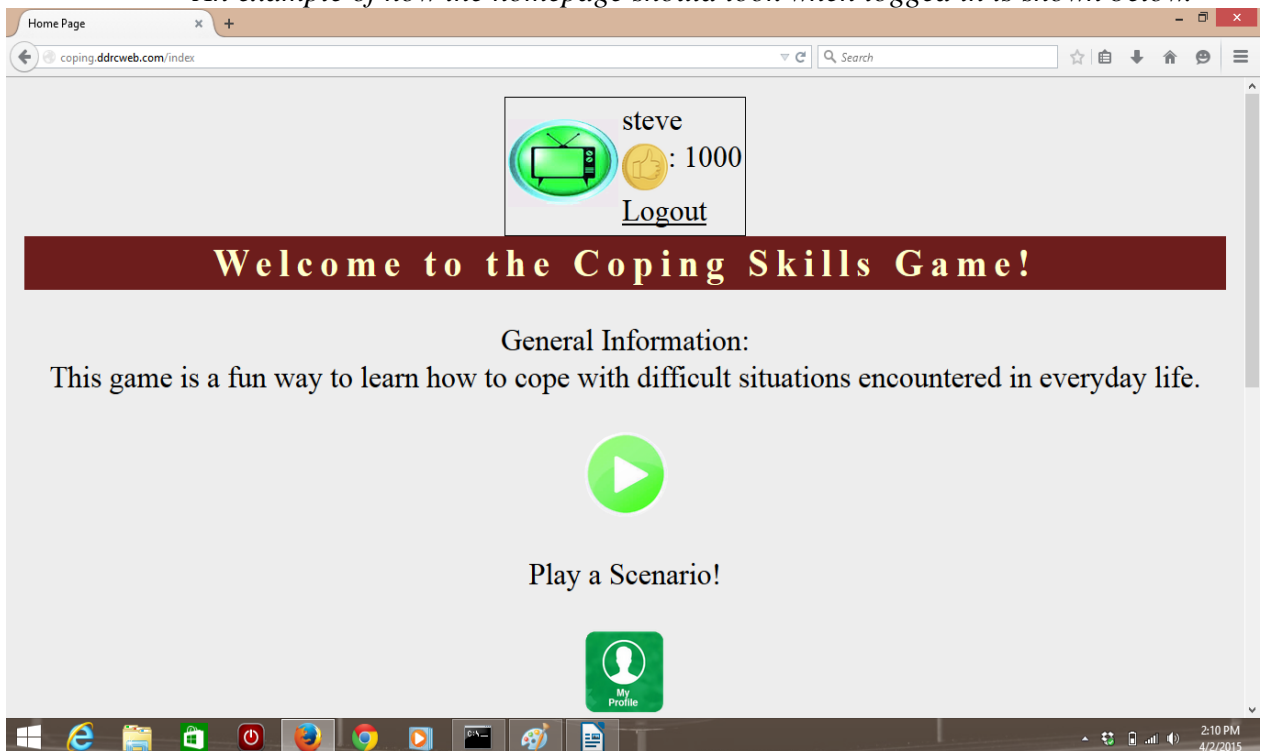
Instructions for playing the game:

- 1) Turn on and log in to a computer or tablet.
- 2) While connected to the Internet, open a browser (Internet Explorer, Firefox, Chrome, Safari, etc.) and go to: <http://coping.ddrcweb.com>
You can copy the link above and paste it in the URL, as shown below.



- 3) You are now ready to start testing the game! Click on "Create New Account!" to make your account, log in using your newly made credentials, then navigate and test the game! When finished, log out and close your web browser.

An example of how the homepage should look when logged in is shown below.

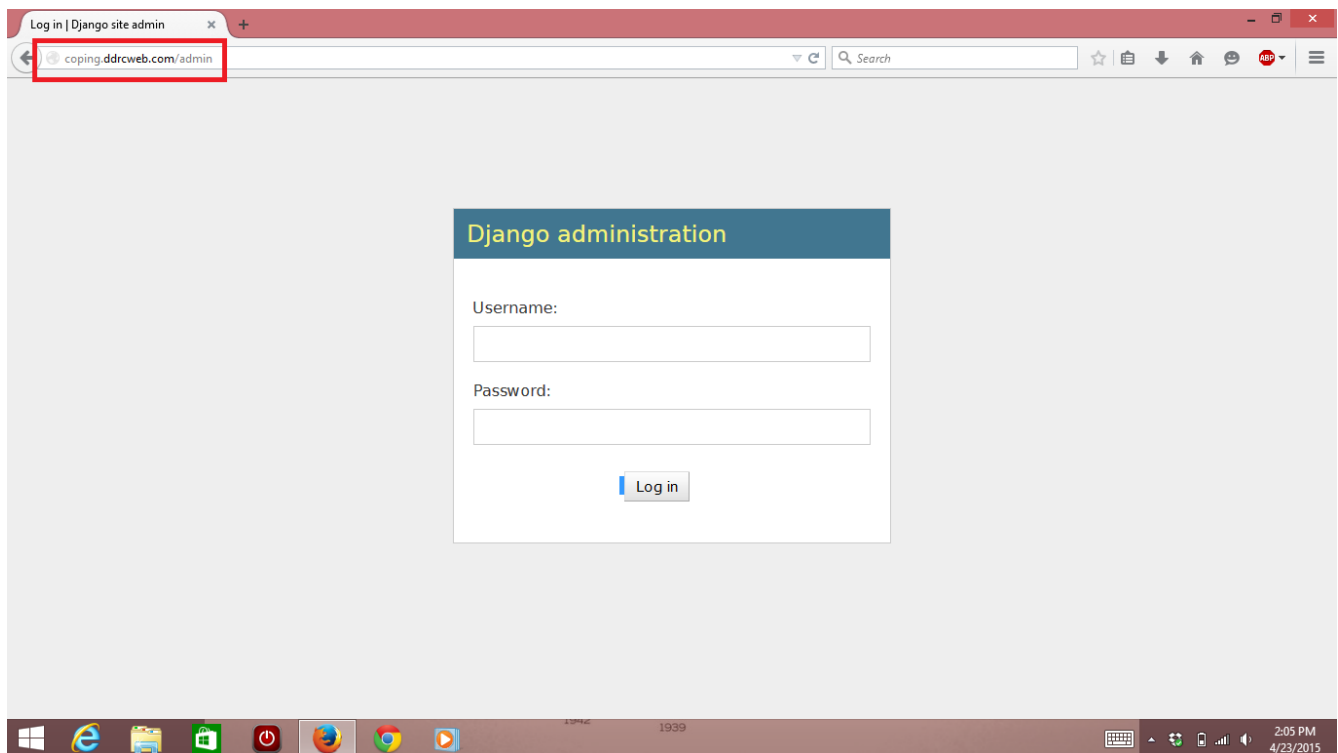


- Note that the only scenarios that will show up when you hit the play button are the default scenarios (if there are any) and any scenarios given to each specific player through the administrator site. The administrator site has its own set of instructions.

Instructions for getting to and using the administrator (admin) page:

1) Turn on and log in to a computer, tablet, or phone.

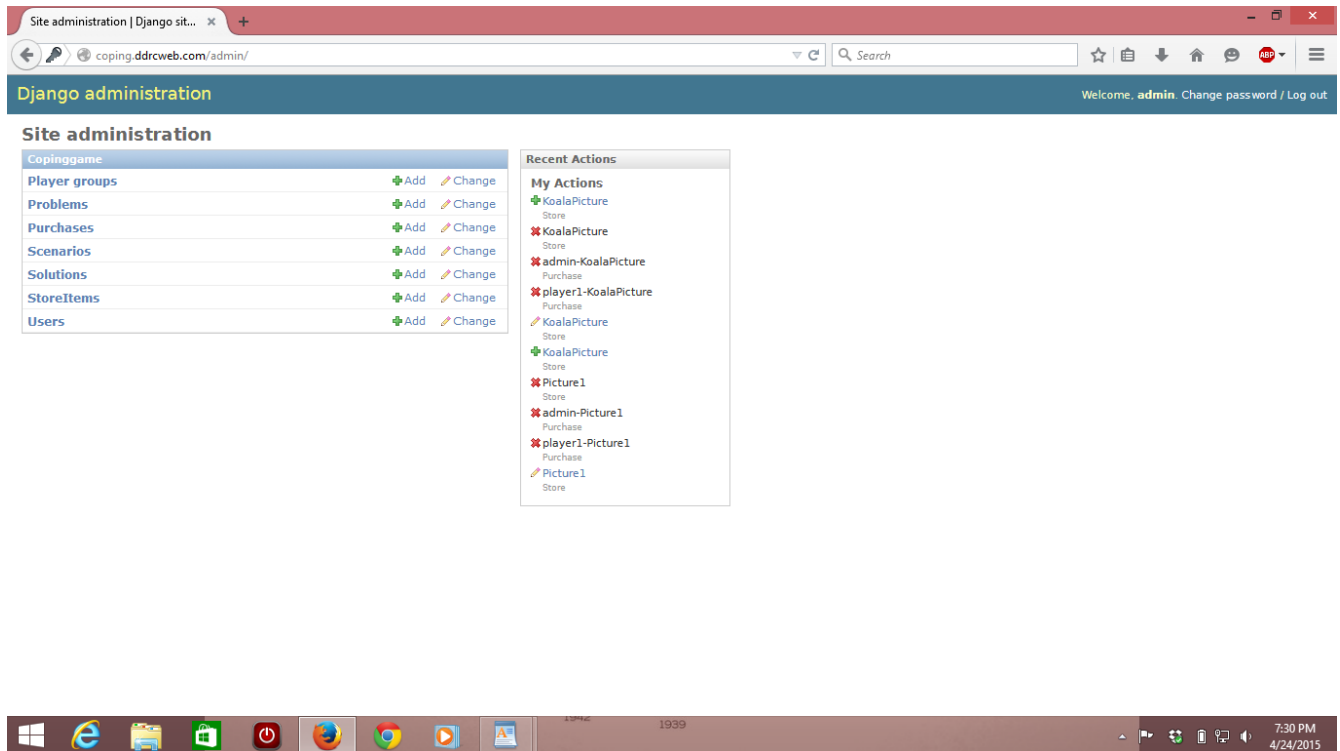
2) While connected to the Internet, open a browser (Internet Explorer, Firefox, Chrome, Safari, etc.) and go to: <http://coping.ddrcweb.com/admin> . You can also just go to the game's main site, log in as an administrator, and there will be a link to the admin page in the top left corner of the Home page.



You can copy the link above and paste it in the URL, as shown below.

3) Now log in with an administrator account, and you will be brought to the interface.

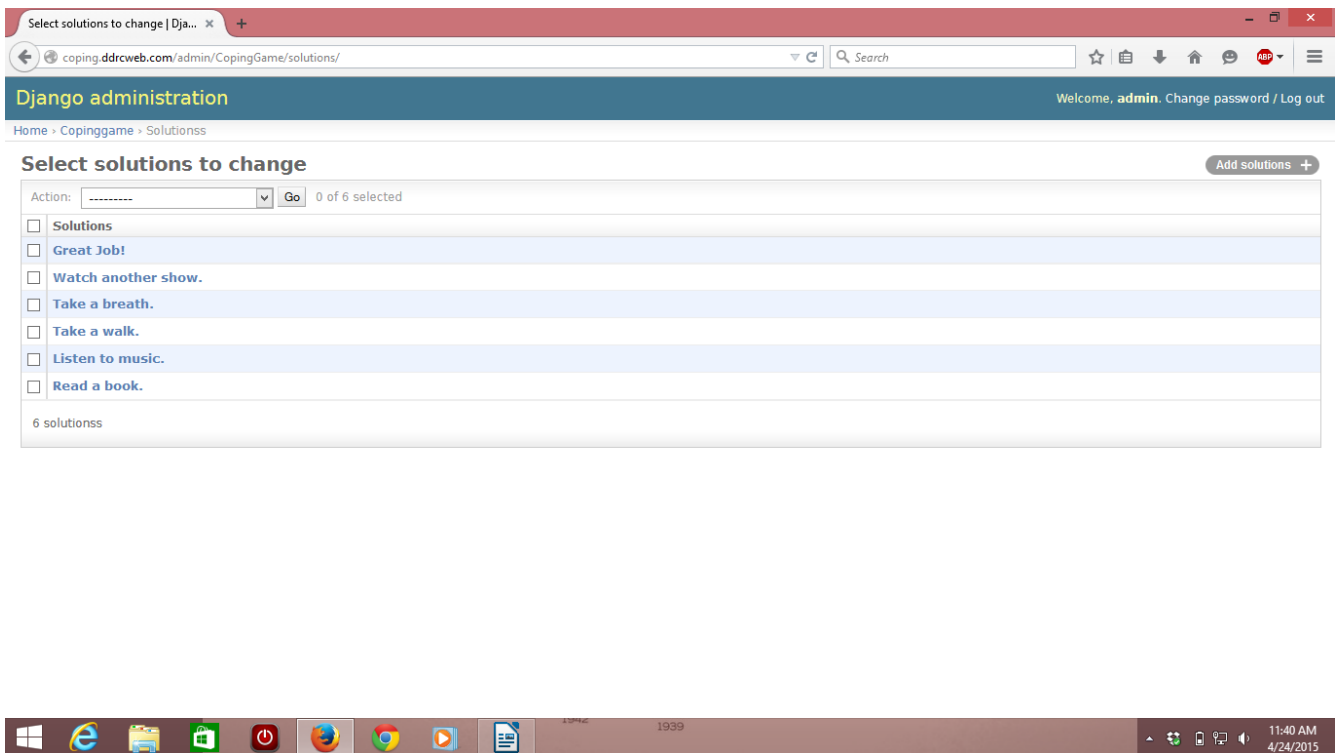
The username: admin and the password: admin will log you in as an admin for now. A picture of the interface after you log in is shown below:



4) To create a new scenario from scratch, you'll need to create your own set of problems and solutions, but you may also use problems and scenarios already in the database. In order to create your own problem, you'll first need to create your own solutions if you wish to use your own.

If you don't wish to create your own solutions, then please go to step 5. Clicking on “Solutions” will take you to the following page. Each problem will need 1 solution and allows a maximum of 3 solutions.

The Solutions page is shown below:



To add a new solution, click on the “Add solutions +” button located near the top right corner of the screen.

You will need to add a picture from your computer/tablet/phone for the solution. You may use a picture you already have or save one from the Internet and then upload it. In addition, you'll also need to name your solution. This name will be shown to players as the name of the solution, for example “Go on a walk.” Lastly, you'll need to add a SvideoId to the solution. This can be found by going to YouTube, finding a video that fits your solution, and copying the portion of the URL of the YouTube video (shown below) and paste it into the SvideoId text area (copy everything after the = sign). This video will be used to show the player a visual example of the solution you're creating.

The Add solutions page is shown below, along with a picture of what to get from the URL in a YouTube video:

Add solutions | Django site ad... x The Ventures "Walk Don't ... x +

← copingddrcweb.com/admin/CopingGame/solutions/add/ Search

Django administration Welcome, admin. Change password / Log out

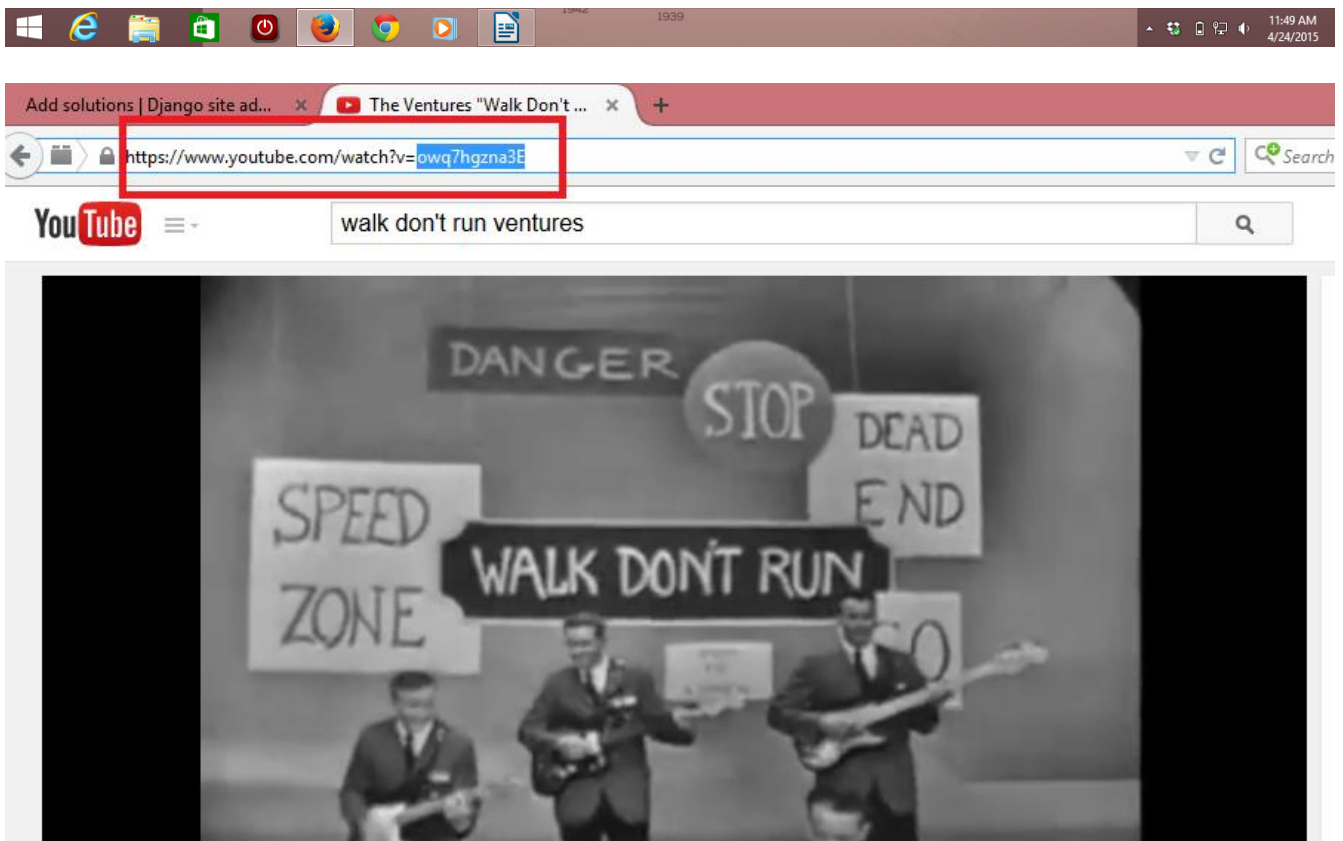
Home > Copinggame > Solutions > Add solutions

Add solutions

PictureS: adminolutions.png

Solution:

SVideoId:

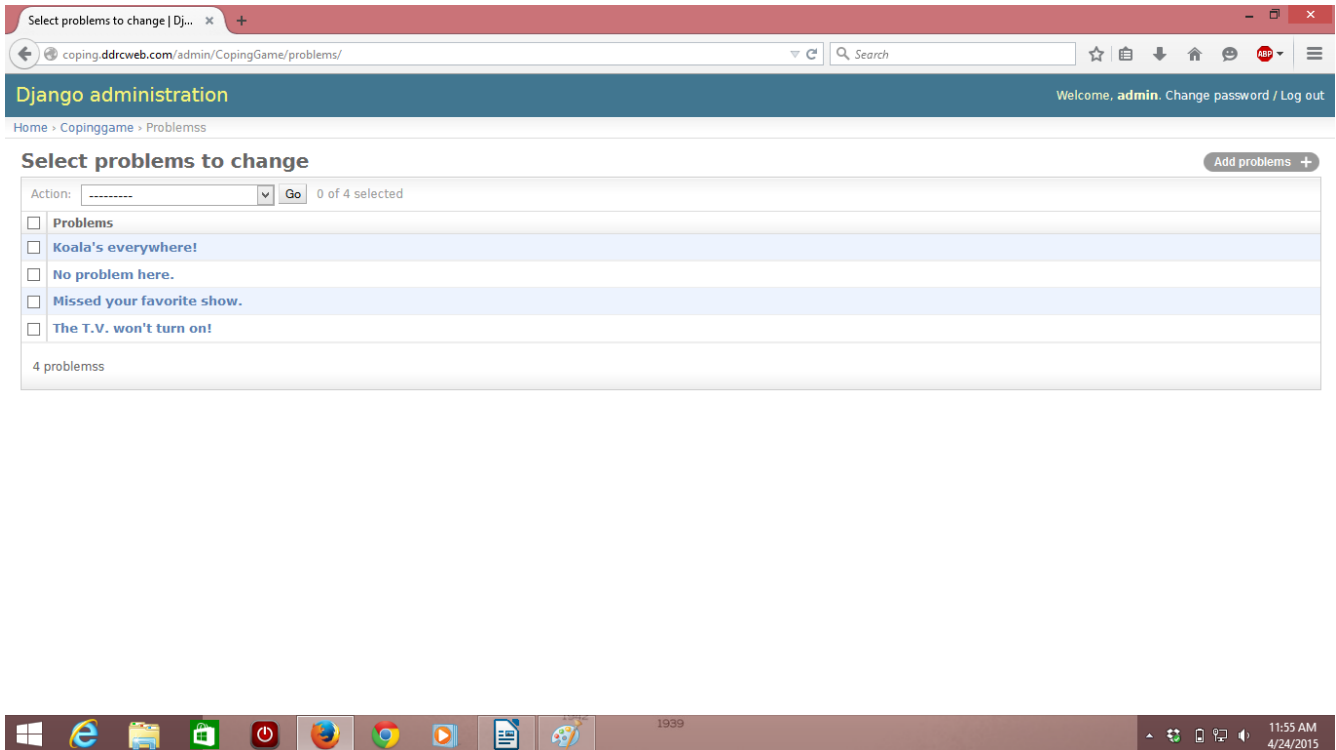


After you have the picture uploaded, the solution named, and a video selected, click on the “Save” button in the Add solutions page to save your solution.

5) If you do not wish to create your own problems, then please go to step 6. Click on “Problems” from

the main admin page (you can get back to the main admin page by clicking on “Copinggame” in the path at the top left of the page, or by going to the URL coping.ddrcweb.com/admin). Each scenario needs a minimum of 1 problem, and allows a maximum of 5 problems (3 is recommended for optimal formatting).

The Problems page is shown below:



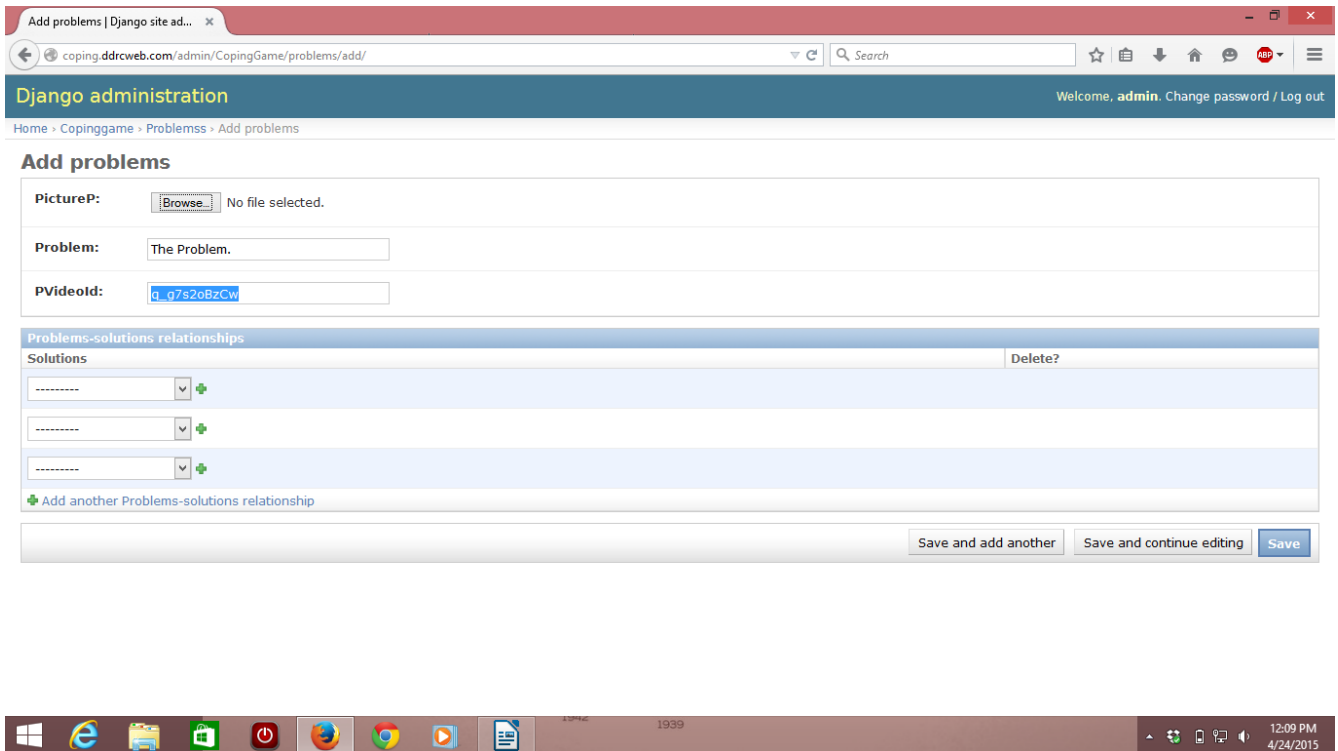
To create a new problem, click on “Add problems +” in the top right corner of the page.

As with solutions, the PictureP must be uploaded from your computer. This picture will display as the overall picture to represent the problem you're making.

The Problem field will need to be filled with whatever you wish to describe your problem as. For example, “My neighbor is being loud.”

PvideoID works the same way as it did in solutions, you'll need to find a video to represent your problem on YouTube and copy the following portion of the URL into the text area (A YouTube video will always go something like: <https://www.youtube.com/watch?v=>. You do not need to worry about that. You need to get what is after the = in the URL to put in PvideoID). This video will play automatically and will give a video example of what your problem is.

The Add problems page is shown below, with the PvideoID highlighted in blue:



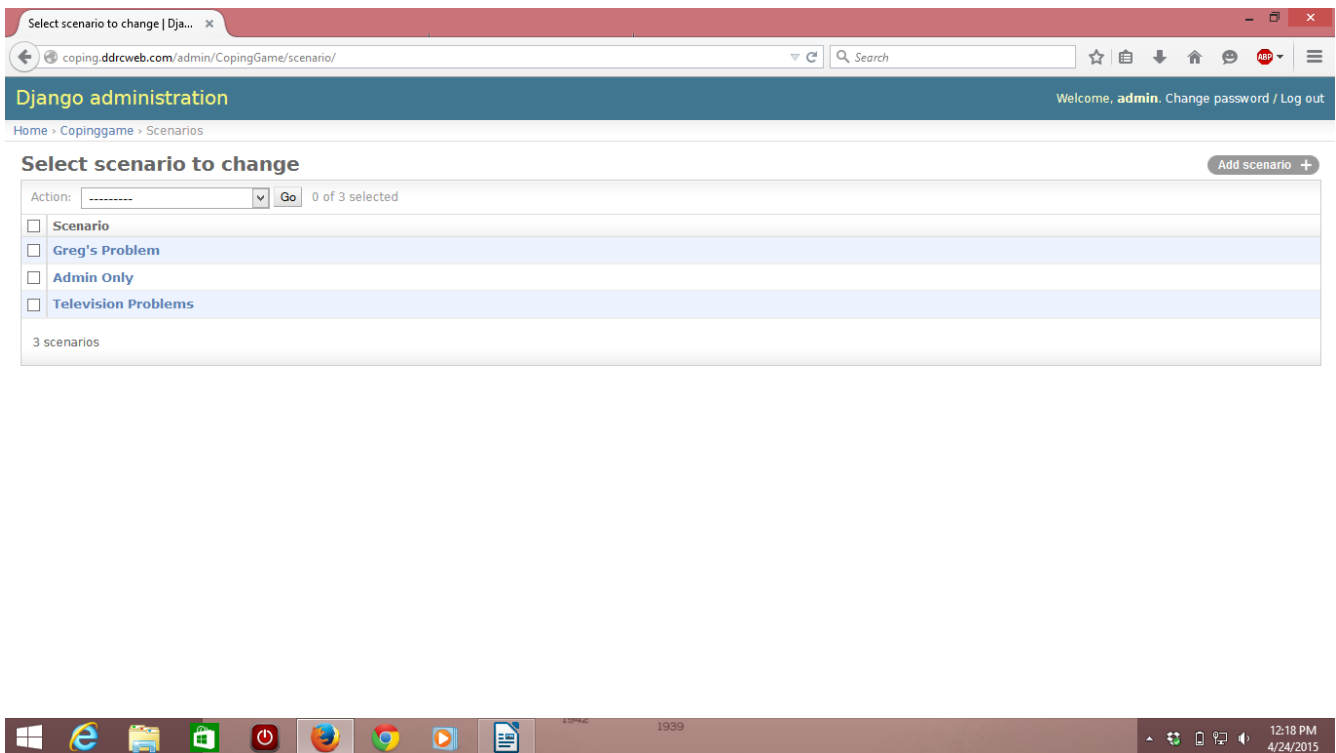
After this, select which solutions you want the players to choose between as a solution to your problem. Click on the drop down arrow and select the solutions you wish to use.

If a solution you need doesn't exist that you want to use, click the green plus button in the Solutions to make one.

Once you have a picture, description, video, and solutions selected, click on the “Save” button in order to save your problem.

6) Navigate back to the main admin page, if you're not already there, by clicking on the blue “Copinggame” link in the path located near the top left of the page. Then click on the “Scenarios” from the main admin page to get to the Scenarios page.

The Scenarios page is shown below:



To add your own scenario, click on the “Add scenario +” located in the top right corner of the page. Fill in the “Title” field with what you want your scenario to be named, like “Roommate Problems.” You can add a description of the scenario if you wish to have one, but it's not required. Select which problems you want your scenario to have by using the drop down arrows in the “problems” section of the page shown below.

To be able to choose which players can see and play your scenario, you can do this in one of two ways:

- 1) Select every player individually using drop down arrows in the “players” section. You will have to add more people with the green plus sign if you have more than three.
- 2) Select the player groups you wish to have access to your scenario using the drop down arrows in the “player groups” section. If you have more than three player groups, you will need to add more with the green plus sign.

After naming your scenario, choosing which problems are in it, and selecting which players can play your scenario, click on the “Save” button.

The Add scenarios page is shown below:

Add scenario | Django site admin

copingddrcweb.com/admin/CopingGame/scenario/add/

Django administration

Welcome, admin. Change password / Log out

Home > Copinggame > Scenarios > Add scenario

Add scenario

Title: Roommate Problems

Description: Scenario Description

Scenario-problems relationships

Problems	Delete?
.....	
.....	
.....	

+ Add another Scenario-problems relationship

Scenario-player relationships

Player	Delete?
.....	
.....	
.....	

+ Add another Scenario-player relationship

Scenario-playergroup relationships

Playergroup	Delete?
.....	
.....	
.....	

+ Add another Scenario-playergroup relationship

Save and add another Save and continue editing Save

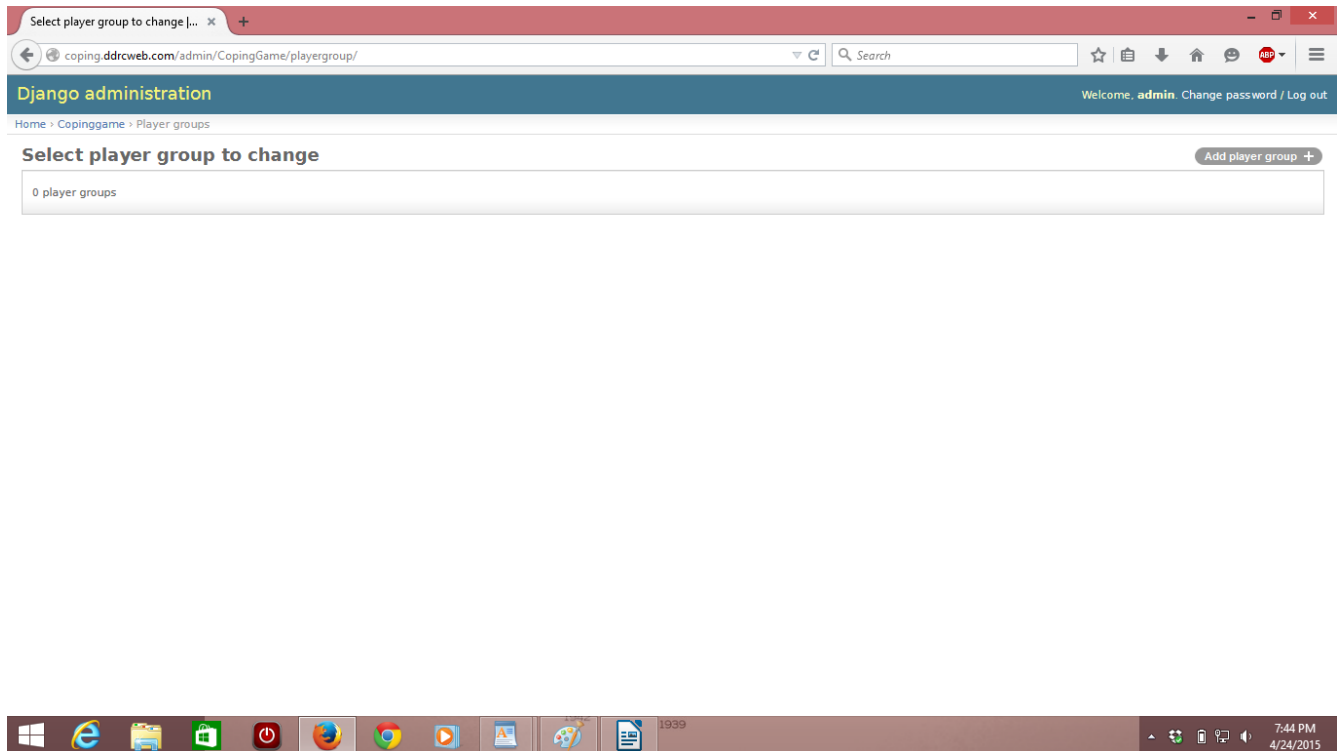
If nobody is selected in either of the two fields (Player or Playergroup), it will still work, but nobody will be able to play the scenario.

If a player is in one field but not the other (say you add someone in Player but not in Playergroup), that player will be able to see and play the scenario in the game.

If a player is added to both Player and Playergroup, they should be able to see and play the scenario in the game.

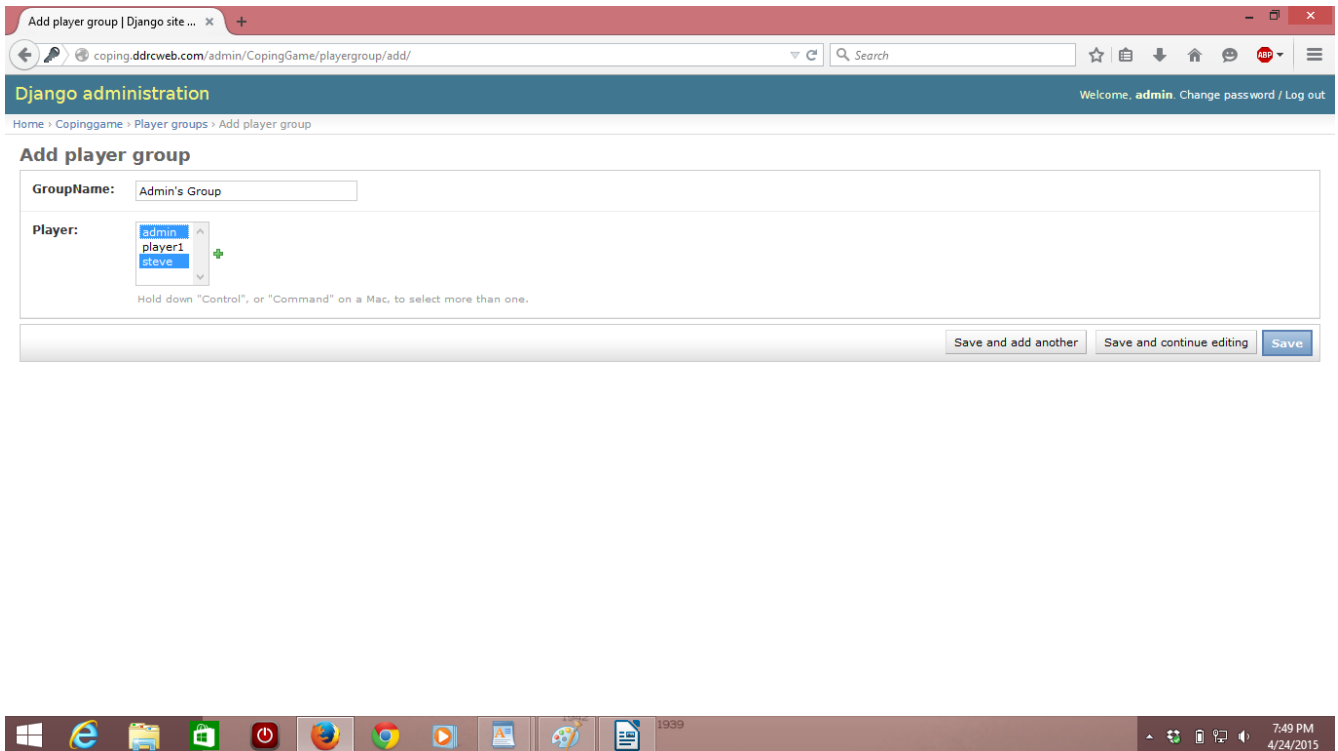
7) Navigate back to the main admin page, if you're not already there, by clicking on the blue “Copinggame” link in the path located near the top left of the page. Then click on the “Player groups” link from the main admin page to get to the Player groups page.

The Player Groups page is shown below:



Click on the Add player group button to make a new player group. It's very easy. First, you give the player group a name, then select the players you want in the group by either Ctrl+left click to select one player at a time, or Shift+left click to select all the players from click to click, inclusively.

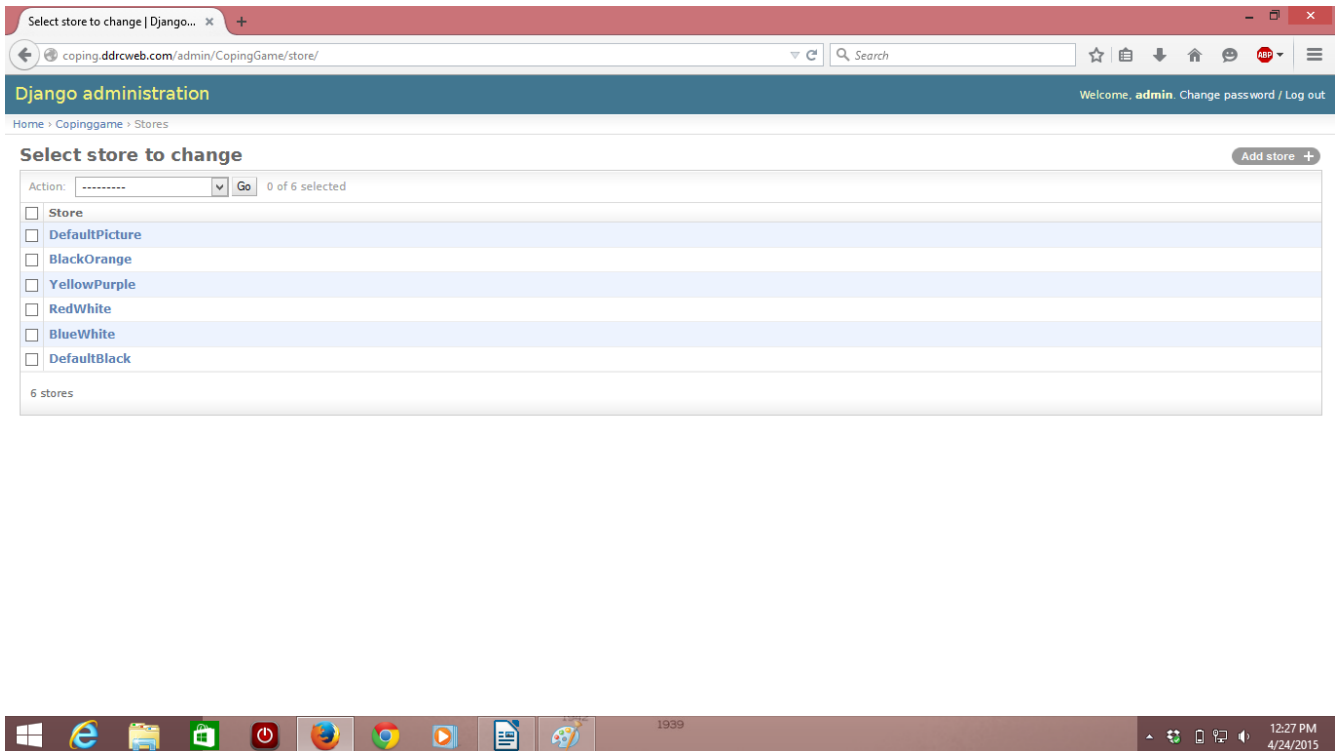
A screenshot showing “admin” and “steve” being added to a new group called “Admin's Group” is shown below:



ADDING STORE ITEMS:

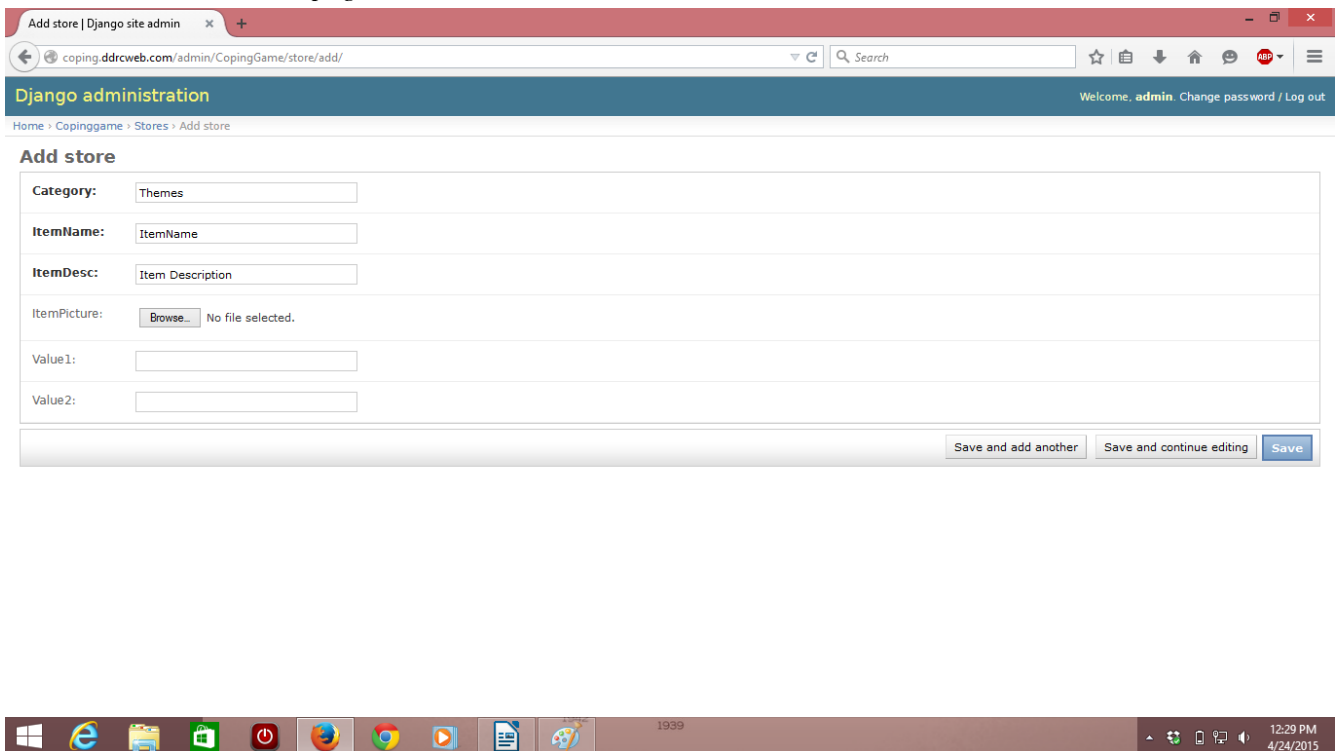
Go to the “Stores” link from the main admin page (click on the Coppinggame link near the top left of the page if you are not already there).

The Store page is shown below:



Once there, click on the “Add store +” button in the top right of the page. The store only has two categories, Themes and Pictures.

The Add store page is shown below:



THEMES:

To add a Theme, leave the Category field as “Themes”. Name your theme by filling in the ItemName field with whatever you wish, but it's best to name it something logical (for example, the color combination of the theme, like “GreenWhite”). Also give your theme a description to describe it, like “Green background with white text.”

You'll use the “Bg” and “Text” fields. Bg correlates to the background color of the web page, and Text is what the text color will be. You can use most color names (like “red”, “green”, “orange”, etc), or you may use the Hex value of a color, which the following website can help you look up.

http://www.w3schools.com/cssref/css_colorsfull.asp

Ignore the ItemPicture field when creating themes. Once you have the appropriate category, name, description, and color values, click the “Save” button to save your theme.

PICTURES:

To add a picture, change the category field from “Themes” to “Picture” by typing it in.

In order to add a picture, you'll upload a picture from your computer. In order to add a picture from online, you'll have to save a picture from online. Click on the browse button next to “ItemPicture” and choose whatever picture you wish to add to the store.

Ignore the Bg and Text fields when adding a picture. Once you have the appropriate category, name, description, and picture selected, click the “Save” button to save your picture.

Note that for testing purposes, it is advisable not to delete scenarios and purchases (unless you created them), as it is difficult or impossible to get them back. To delete something, from a page (say the Players page), check the box next to the item you want to delete, then click on the dropdown called “Action” and choose “Delete selected players” and press “GO”. You will ask if you really want to do it, and say “Yes I'm sure” if you mean it, or back out if you don't.

If something does get deleted or removed, please send us an email and we will try to fix it as soon as possible. It is especially advisable not to delete the default Scenarios, Solutions, Player groups, Store, Players, Problems and Purchases (like the television problems, player1 and admin players, the BlueWhite themes, etc.).

Training:

No official training is necessarily needed to play the game, but training for using the admin page may be necessary for maintaining and updating the database and user accounts. The instructions on how to use the admin page are shown above.

Maintenance Plan:

This product will be maintained by the staff of DDRC, and Team Pluvali while we are working on it. Once we are finished with the product, someone from one of the groups below will maintain it:

The System Administrator will ensure that the server, the software on the server, and the server's connection to the Internet will be properly maintained and updated as needed.

The other staff of DDRC who are involved in this project will make sure that all of the user's profiles are in good working order.

It is also a possibility that members of The University of Colorado Denver's (UCD) Association for Computing Machinery will maintain and update the code as needed.

Source Code:

Below is the HTML code along with the Django, Python, and CSS files, just as it is in our actual product:

```
<!DOCTYPE html>

<!-- The login page -->

{% load staticfiles %}

<html>
<head>

<!-- Possibly change to put in its own css file -->
<style type="text/css">
    h1 {color:#6d1d1c;}
</style>

    <!-- A picture of DDRC's logo -->
    </a>

    <!-- The title of the page and a link to the logged-in home page
-->
    <h1 style="text-align:center;">Coping Skills Game</h1>
    <link rel="stylesheet" href="style_login_page.css">
</head>
<body>

    <!-- The instructions and the fields for logging in -->
    <h2 style="text-align:center"; color:"#4a6530">Type in your
username and password to log in.</h2>

    <!-- If no user -->
    {% if form.errors %}
        <p> Your username and/or password didn't match. Please try
again.</p>
    {% endif %}

    <form action="{% url 'django.contrib.auth.views.login' %}"
method="post">
    {% csrf_token %}
    <table border="0" cellpadding="15" width="345" align="center">
        <tr>
            <td width="100">{{ form.username.label_tag }}</td>
            <td>{{ form.username }}</td>
        </tr>
```

```

        <tr>
            <td class="align-left">{{ form.password.label_tag
}}</td>
            <td>{{ form.password }}</td>
        </tr>
        <tr><td><input type="submit" value ="login" align="center"
/></td></tr>
        <tr><td><input type="hidden" name ="next" value="{{next}}"
/></td></tr>
        <tr><td><a href="/register/">Create New
Account!</a></td></tr>
    </table>
</form>

</body>

</html>

<!DOCTYPE html>

<!-- The registration page -->

{% load staticfiles %}

<html>
<head>

<!-- Possibly change to put in its own css file -->
<style type="text/css">
    h1 {color:#6d1d1c;}
</style>

    <!-- A picture of DDRC's logo -->
    </a>

    <!-- The title of the page and a link to the logged-in home page
-->
    <h1 style="text-align:center;">Create an account</h1>
    <link rel="stylesheet" href="style_login_page.css">
</head>
<body>

    <!-- The instructions and the fields for logging in -->
    <h2 style="text-align:center"; color:"#4a6530">Type in your
username and password to log in.</h2>

    <!-- If no user -->
    {% if form.errors %}

```

```

        <p> Your username and/or password didn't match. Please try
again.</p>
        {% endif %}

        <form action="" method="post">
        {{ form.as_p }}
        <p style="color:#4c3365;">Email:<input type="text"
name="email"></p>

        {% csrf_token %}
        <input type="submit" value ="Create the account" align="center"
/>

        </form>

        <br>
        <li><a href="/login/">Back to login.</a></li>

</body>
</html>

```

```

<!DOCTYPE html>

```

```

<!-- The admin page -->
{% load staticfiles %}
<html>
<link rel="stylesheet" type="text/css" href="{% static
'CopingGame/style_home_page.css' %}" />
<body>
{% if player_list %}
<table>
<tr>
{% for player in player_list %}
<td>
{{ player }} <br/>
{{ player.passw }} <br/>
{{ player.email|default:"No email" }} <br/>
{{ player.points }} <br/>
{{ player.fav_bg }} <br/>
{{ player.fav_text }} <br/>
{{ player.stage }} <br/>
<b>TODO: Add form to edit fields</b>
<form>
        Set Email: <input type="text" name="email">
        Set Password: <input type="password" name="email">
</form>
</td>
{% endfor %}
</tr>
</table>
{% else %}
No players are available.

```



```

{% endif %}

</body>

</html>

<!DOCTYPE html>

<!-- The game page -->

{% load staticfiles %}
<html>
<head>
    <!-- Title the page and link the style sheet -->
    <title>Game Page</title>
    <link rel="stylesheet" type="text/css" href="{% static
'CopingGame/style_game_page.css' %}" />
    <script type="text/javascript">
        //Install iframe API
        var tag = document.createElement('script');
        tag.src = "https://www.youtube.com/iframe_api";
        var firstScriptTag =
document.getElementsByTagName('script')[0];
        firstScriptTag.parentNode.insertBefore(tag,
firstScriptTag);

        var problemPlayer;

        //Load and start iframe video
        function onYouTubeIframeAPIReady() {
            problemPlayer = new YT.Player('problemPlayer', {
                height: '390',
                width: '640',
                //Initial problem video
                videoId: '{{stage0.pVideoId}}',
                playerVars :{
                    //Pull playlist from database
                    {% if player.stage == 0 %}
                        'playlist': ['{% for sol in
stage0.solutions.all %}{{ sol.sVideoId }}, {% endfor %}']
                    {% elif player.stage == 1 %}
                        'playlist': ['{% for sol in
stage1.solutions.all %}{{ sol.sVideoId }}, {% endfor %}']
                    {% elif player.stage == 2 %}
                        'playlist': ['{% for sol in
stage2.solutions.all %}{{ sol.sVideoId }}, {% endfor %}']
                    {% elif player.stage == 3 %}
                        'playlist': ['{% for sol in
stage3.solutions.all %}{{ sol.sVideoId }}, {% endfor %}']
                    {% elif player.stage == 4 %}

```

```

        'playlist': ['{% for sol in
stage4.solutions.all %}{{ sol.sVideoId }}, {% endfor %}']
        {% endif %}
    },
    events: {
        'onReady': onPlayerReady,
        'onStateChange': onPlayerStateChange
    }
});
}

//Play video when loaded
function onPlayerReady(event) {
    event.target.playVideo();
}

//When video ends on last of playlist, show solutions div
function onPlayerStateChange(event) {
    if (event.data == YT.PlayerState.ENDED /*&& end of
playlist*/) {
        var divsToShow =
document.getElementsByClassName("button");
        for(var i = 0; i < divsToHide.length; i++)
        {
            divsToShow[i].style.visibility="visible";
        }
    }

    //Function to show divs after all videos have played
    function showSolutions() {
        var divsToShow =
document.getElementsByClassName("button");
        for(var i = 0; i < divsToHide.length; i++)
        {
            divsToShow[i].style.visibility="visible";
        }
    }

    //Hide answer divs to force Player to watch videos
    //function hideSolutions() {
        //var divsToHide =
document.getElementsByClassName("button");
        //for(var i = 0; i < divsToHide.length; i++)
        //{
            //divsToHide[i].style.visibility="hidden";
        //}
    //}
    //setTimeout("hideSolutions()", 0); //UNCOMMENT

```

```

        //Focus on the video player
        window.location.hash = '#Video1Div';
    </script>
</head>
<body style="background-color:{{ player.fav_bg }}; color:{{
player.fav_text }}">
<font size="{{ player.text_size }}">
    <table id="page" align="center" style="background-color:{{
player.fav_bg }}; color:{{ player.fav_text }}">
        <tr>

            <!-- Table that will have the user profile once logged
in -->

            <table id="usertable" align="center"
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
}}">

                <tr>

                    <td rowspan="2" colspan="2"></td>

                    <td>
                        {% if user.is_authenticated %}
                            <!-- user information -->
                            {{ user.get_username }}
                            <br>
                            :
                            {{ player.tokens }}

                            <br>
                            <a href="/logout/" style="color:{{
player.fav_text }}">Logout

                        {% else %}
                            Not currently logged in, please log
in.

                        {% endif %}</td>
                    </td>
                </tr>
                <tr>
                    <td></td>
                </tr>
            </table>
        </tr>
        </br>
        <tr>
            <!-- What is the current scenario and problem -->
            <h1>Scenario - {{ scene.title }}</h1>
        </tr>
        <tr>
            <!-- Video Player Div -->
            <div id="Video1Div" align="center">
                <div id="problemPlayer" align="center">
                    </div>
                </div>
            </tr>

```

```

<tr>
  <!-- Header with current problem -->
  <h2>Problem - {% if player.stage == 0 %}{{ stage0 }}
    {% elif player.stage == 1 %}{{ stage1 }}
    {% elif player.stage == 2 %}{{ stage2 }}
    {% elif player.stage == 3 %}{{ stage3 }}
    {% elif player.stage == 4 %}{{ stage4 }}
    {% endif %}

  </h2>
</tr>
<tr>
  <!-- Image representing the problem -->
  {% if player.stage == 0 %}
    
    {% elif player.stage == 1 %}
    
    {% elif player.stage == 2 %}
    
    {% elif player.stage == 3 %}
    
    {% elif player.stage == 4 %}
    
    {% endif %}
  </tr>
  <br>
  <tr>
    <!-- Table containing the coping choices and their
associated images -->
    <table id="coping" align="center">
      <!-- Width of the three columns -->
      <col width="400"/>
      <col width="400"/>
      <col width="400"/>
      <tr id="handle">
        <td colspan="3">How To Handle This
Problem?</td>
      </tr>
      <tr id="techniques">
        <!-- Text associated with the solution -->
        {% if player.stage == 0 %}
          {% for sol in stage0.solutions.all %}
            <td>{{sol.solution}}</td>
          {% endfor %}
        {% elif player.stage == 1 %}
          {% for sol in stage1.solutions.all %}
            <td>{{sol.solution}}</td>
          {% endfor %}

```

```

        {% elif player.stage == 2 %}
            {% for sol in stage2.solutions.all %}
                <td>{{sol.solution}}</td>
            {% endfor %}
        {% elif player.stage == 3 %}
            {% for sol in stage3.solutions.all %}
                <td>{{sol.solution}}</td>
            {% endfor %}
        {% elif player.stage == 4 %}
            {% for sol in stage4.solutions.all %}
                <td>{{sol.solution}}</td>
            {% endfor %}
        {% endif %}
    </tr>
    <tr>
        <!-- Solution's picture -->
        {% if player.stage == 0 %}
            {% for sol in stage0.solutions.all %}
                <td></td>
            {% endfor %}
        {% elif player.stage == 1 %}
            {% for sol in stage1.solutions.all %}
                <td></td>
            {% endfor %}
        {% elif player.stage == 2 %}
            {% for sol in stage2.solutions.all %}
                <td></td>
            {% endfor %}
        {% elif player.stage == 3 %}
            {% for sol in stage3.solutions.all %}
                <td></td>
            {% endfor %}
        {% elif player.stage == 4 %}
            {% for sol in stage4.solutions.all %}
                <td></td>
            {% endfor %}
        {% endif %}
    </tr>

    <!--Show how many tokens the player has earned-->
    <tr>
    <div>
    <form action="" method="post">
    {% csrf_token %}
        {% if player.stage == 0 %}
            {% for s in stage0.solutions.all %}

```

```

                                <td>
                                    <div id="answerDiv">
                                        <!-- Button to open
pop-up -->
                                            <a class="button"
href="#openModal" style="text-decoration:none">Choose me!</a>
                                                <!-- Tokens pop-up --
>
                                            <div id="openModal"
class="modalDialog">
                                                <div
style="background:{{player.fav_bg}}; background: -moz-linear-
gradient({{player.fav_bg}} 80%, {{ player.fav_text }}); background: -
webkit-linear-gradient({{player.fav_bg}} 80%, {{ player.fav_text }});
background: -o-linear-gradient({{player.fav_bg}} 80%, {{
player.fav_text }});">
                                                                <h2
style="color:{{ player.fav_text }}">Great Choice!</h2>
                                                                <p>You
earned {{ tokens0 }} !</p>
                                                                <input
type="submit" href="#close" title="Close" class="close"
value="X"></input>
                                                                </div>
                                                                </div>
                                                                </div>
                                </td>
                                {% endfor %}
                                {% elif player.stage == 1 %}
                                {% for s in stage1.solutions.all %}
                                <td>
                                    <div>
                                        <!-- Button to open
pop-up -->
                                            <a class="button"
href="#openModal1" style="text-decoration:none">Choose me!</a>
                                                <!-- Tokens pop-up --
>
                                            <div id="openModal1"
class="modalDialog">
                                                <div
style="background:{{player.fav_bg}}; background: -moz-linear-
gradient({{player.fav_bg}} 80%, {{ player.fav_text }}); background: -
webkit-linear-gradient({{player.fav_bg}} 80%, {{ player.fav_text }});
background: -o-linear-gradient({{player.fav_bg}} 80%, {{
player.fav_text }});">

```

```

<h2
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
}}">Perfect!</h2>

<p>You
earned {{ tokens1 }} !</p>

<input
type="submit" href="#close" title="Close" class="close"
value="X"></input>

</div>
</div>
</div>

</td>
{% endfor %}
{% elif player.stage == 2 %}
{% for s in stage2.solutions.all %}
<td>
<div>
<!-- Button to open
pop-up -->
<a class="button"
href="#openModal2" style="text-decoration:none">Choose me!</a>

<!-- Tokens pop-up --
>
<div id="openModal2"
class="modalDialog">
<div
style="background:{{player.fav_bg}}; background: -moz-linear-
gradient({{player.fav_bg}} 80%, {{ player.fav_text }}); background: -
webkit-linear-gradient({{player.fav_bg}} 80%, {{ player.fav_text }});
background: -o-linear-gradient({{player.fav_bg}} 80%, {{
player.fav_text }});">
<h2
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
}}">Awesome!</h2>
<p>You
earned {{ tokens2 }} !</p>
<input
type="submit" href="#close" title="Close" class="close"
value="X"></input>

</div>
</div>
</div>

</td>
{% endfor %}
{% elif player.stage == 3 %}

```

```

                                {% for s in stage3.solutions.all %}
                                <td>
                                <div>
                                    <!-- Button to open pop-up
-->
                                    <a class="button"
href="#openModal3" style="text-decoration:none">Choose me!</a>

                                    <!-- Tokens pop-up -->
                                    <div id="openModal3"

class="modalDialog">
                                        <div
style="background:{{player.fav_bg}}; background: -moz-linear-
gradient({{player.fav_bg}} 80%, {{ player.fav_text }}); background: -
webkit-linear-gradient({{player.fav_bg}} 80%, {{ player.fav_text }});
background: -o-linear-gradient({{player.fav_bg}} 80%, {{
player.fav_text }});">
                                            <h2
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
}}">Good Answer!</h2>
                                            <p>You earned {{
tokens3 }} !</p>
                                            <input
type="submit" href="#close" title="Close" class="close"
value="X"></input>
                                        </div>
                                    </div>
                                </div>
                                </td>
                                {% endfor %}
                                {% elif player.stage == 4 %}
                                {% for s in stage4.solutions.all %}
                                <td>
                                <div>
                                    <!-- Button to open pop-up
-->
                                    <a class="button"
href="#openModal4" style="text-decoration:none">Choose me!</a>

                                    <!-- Tokens pop-up -->
                                    <div id="openModal4"

class="modalDialog">
                                        <div
style="background:{{player.fav_bg}}; background: -moz-linear-
gradient({{player.fav_bg}} 80%, {{ player.fav_text }}); background: -
webkit-linear-gradient({{player.fav_bg}} 80%, {{ player.fav_text }});
background: -o-linear-gradient({{player.fav_bg}} 80%, {{
player.fav_text }});">

```



```

                                <h2
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
}}">Well Done!</h2>

                                <p>You earned {{
tokens4 }} !</p>

                                <input
type="submit" href="#close" title="Close" class="close"
value="X"></input>

                                </div>
                                </div>
                                </div>

                                </td>
                                {% endfor %}
                                {% endif %}
                                </form>
                                </tr>
                                </table>
                                </tr>
                                <tr>
                                <!-- A picture of DDRC's logo -->
                                </a>
                                </tr>
                                </table>
                                </body>
                                </html>

<!DOCTYPE html>

<!-- The help page -->
{% load staticfiles %}
<html>
<head>
    <title>Help Page</title>
    <link rel="stylesheet" type="text/css" href="{% static
'CopingGame/style_home_page.css' %}" />
</head>
<body style="background-color:{{ player.fav_bg }}; color:{{
player.fav_text }}">
<font size="{{ player.text_size }}">
    <table id="page" align="center" style="background-color:{{
player.fav_bg }}; color:{{ player.fav_text }}">
        <tr>
            <!-- Table that will have the user profile once logged
in -->
            <table id="usertable" align="center"
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
}}">

```

```

        <tr>
            <td rowspan="2" colspan="2"></td>
            <td>
                {% if user.is_authenticated %}
                <!-- user information -->
                {{ user.get_username }}
                <br>
                :
                {{ player.tokens }}
                <br>
                <a href="/logout/" style="color:{{
player.fav_text }}">Logout
                {% else %}
                Not currently logged in, please log
in.
                {% endif %}</td>
        </tr>
        <tr>
            <td></td>
        </tr>
    </table>
</tr>
<br>
<tr>
    <div id="links" align="center">
        <table style="width:100%">
            <col style="width:25%">
            <col style="width:25%">
            <col style="width:25%">
            <col style="width:25%">
            <!-- The button to return to the home page -->
            <tr align="center">
                <td align="center"><a href="/CopingGame/"
style="text-decoration: none">
                    
                    </a></td>
                <td align="center"><a
href="/CopingGame/scenarios/" style="text-decoration: none">
                    
                    </a></td>
                <td align="center"><a
href="/CopingGame/profile/" style="text-decoration: none">

```

```

                                
                                </a></td>
                                <td align="center"><a
href="/CopingGame/store_themes/" style="text-decoration: none">
                                
                                </a></td>
                                </tr>
                                <tr>
                                <td align="center">Home Page</td>
                                <td align="center">Play a Scenario</td>
                                <td align="center">Profile Page</td>
                                <td align="center">Store Page</td>
                                </tr>
                                </table>
                                </div>
                                </tr>
                                <tr>
                                <!-- The Header -->
                                <h1>How to play</h1>
                                </tr>
                                <tr>
                                <p align="center">Once logged in, click the "Play a
Scenario" to choose from the available scenarios.</p>
                                </tr>
                                <tr>
                                <p align="center">Select the scenario you wish to play
and then chose a solution for each problem.</p>
                                </tr>
                                <tr>
                                <p align="center">Solving each problem rewards a
token, which you may use to redeem prizes at the store!</p>
                                </tr>
                                <br>
                                <tr>
                                <p align="center">To visit the store, click on the
store page button.</p>
                                </tr>
                                <tr>
                                <p align="center">Once in the store, select from a
category and then chose the item you want to buy!</p>
                                </tr>
                                <tr>
                                <p align="center">Once purchased, you can switch
between themes or user pictures in the store page by clicking the
paint brush picture.</p>
                                </tr>

```

```

        <br>
        <tr>
            <p align="center">You can also switch between
purchased themes and user pictures in the profile page.</p>
        </tr>
        <tr>
            <p align="center">From the profile page you can also
change the font size, and also upload a picture from your device!</p>
        </tr>
        <br>
        <tr>
            <p align="center">To navigate to a page you want to go
to, click on the button of your choice near the top of the screen.</p>
        </tr>
        <tr>
            <p align="center">In order to log out of the game when
finished, press the logout button in the top of your screen.</p>
        </tr>
        <br>
        <tr>
            <!-- A picture of DDRC's logo -->
            </a>
        </tr>
    </table>
</body>
</html>

```

```
<!DOCTYPE html>
```

```

<!-- The home page -->
{% load staticfiles %}
<html>
<head>
    <!-- The title of the page and a link to the style sheet -->
    <title>Home Page</title>
    <link rel="stylesheet" type="text/css" href="{% static
'CopingGame/style_home_page.css' %}" />
</head>
<body align="center" style="background-color:{{ player.fav_bg }};
color:{{ player.fav_text }}">
<font size="{{ player.text_size }}">
    <table id="page" align="center" style="background-color:{{
player.fav_bg }}; color:{{ player.fav_text }}">
        <tr>
            <!-- Table that has the user's profile when logged in
-->
            <table id="usertable" align="center"
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
}}">
                <tr>

```

```
 </td>  <td></td> </tr> </table> </tr>  <br>  <tr>     <!-- The Header -->     <h1>Welcome to the Coping Skills Game!</h1> </tr> <tr>     <p align ="center" style ="text-align:center;color:{{ player.fav_text }}">General Information: <br>This game is a fun way to learn how to cope with difficult situations encountered in everyday life.</p> </tr>  <br>  <tr>     <div id="links" align="center">         <table style="width:100%">             <col style="width:25%">             <col style="width:25%">             <col style="width:25%"> | |
```

```

        <col style="width:25%">
        <!-- The buttons go to the home page, profile page, or
help page -->
        <tr align="center">

                <td align="center"><a
href="/CopingGame/scenarios/" style="text-decoration: none">
                
                </a></td>
                <td align="center"><a href="/CopingGame/profile/"
style="text-decoration: none">
                
                </a></td>
                <td align="center"><a
href="/CopingGame/store_themes/" style="text-decoration: none">
                
                </a></td>
                <td align="center"><a href="/CopingGame/help/"
style="text-decoration: none">
                
                </a></td>
        </tr>
        <tr>
                <td align="center">Play a Scenario</td>
                <td align="center">Profile Page</td>
                <td align="center">Store Page</td>
                <td align="center">Help Page</td>
        </tr>
</table>
</div>
</tr>

<br>

<tr>
        <!-- A picture of DDRC's logo -->
        </a>
        </tr>
</table>

```

```

</body>

</html>

<!DOCTYPE html>

<!-- The profile page -->
{% load staticfiles %}
<html>
<head>
    <title>{{ user.get_username }}'s Profile Page</title>
    <script>
        {% if messages %}
            {% for message in messages %}
                alert(message);
            {% endfor %}
        {% endif %}
        {% if profileForm.errors %}
            {% for field in form %}
                {% for error in field.errors %}
                    <div class="alert alert-error">
                        <strong>{{ error|escape }}</strong>
                    </div>
                {% endfor %}
            {% endfor %}
            {% for error in form.non_field_errors %}
                <div class="alert alert-error">
                    <strong>{{ error|escape }}</strong>
                </div>
            {% endfor %}
        {% endif %}
    </script>
</head>
<link rel="stylesheet" type="text/css" href="{% static
'CopingGame/style_profile_page.css' %}" />
<body style="background-color:{{ player.fav_bg }}; color:{{
player.fav_text }}">
<font size="{{ player.text_size }}">
    <table id="page" align="center" style="background-color:{{
player.fav_bg }}; color:{{ player.fav_text }}">
        <tr>
            <!-- Table that has the user avatar, user name, and
their tokens -->
            <table id="usertable" align="center"
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
}}">
                <tr>
                    <td rowspan="2" colspan="2"></td>
                    <td>

```

```

        {% if user.is_authenticated %}
            <!-- user information -->
            {{ user.get_username }}
            <br>
            :
            {{ player.tokens }}

            <br>
            <a href="/logout/" style="color:{{
player.fav_text }}">Logout
        {% else %}
            Not currently logged in, please log
in.

        {% endif %}</td>
    </tr>
    <tr>
    <td></td>
    </tr>
</table>
</tr>

<br>

<tr>
    <div id="links" align="center">
    <table style="width:100%">
        <col style="width:25%">
        <col style="width:25%">
        <col style="width:25%">
        <col style="width:25%">
        <!-- The button to return to the home page -->
        <tr align="center">
            <td align="center"><a href="/CopingGame/"
style="text-decoration: none">
                
                </a></td>
            <td align="center"><a
href="/CopingGame/scenarios/" style="text-decoration: none">
                
                </a></td>
            <td align="center"><a
href="/CopingGame/store_themes/" style="text-decoration: none">
                

```



```

        </a></td>
        <td align="center"><a href="/CopingGame/help/"
style="text-decoration: none">
        
        </a></td>
    </tr>
    <tr>
        <td align="center">Home Page</td>
        <td align="center">Play a Scenario</td>
        <td align="center">Store Page</td>
        <td align="center">Help Page</td>
    </tr>
</table>
</div>
</tr>

<tr>
    <!-- The Header -->
    <h1>{{ user.get_username }}'s Profile</h1>
</tr>

<tr>
    <!-- PLAYER TOKENS -->
    <center>
        <td align="center">You've earned a total of {{
player.tokens }} </td>
    </center>
</tr>
<tr>
<br>
</tr>
<tr>
    <table id="contentTable" class="t1" style="float:center-
right" style="background-color:{{ player.fav_bg }}; color:{{
player.fav_text }}">
        <thead>
            <tr><th align="left" style="text-
decoration:underline">Change Text Size</th><th align="left"
style="text-decoration:underline">Upload a Player Picture</th>
        </thead>
        <tbody>
            <col style="width:50%">
            <col style="width:50%">
            <tr>
                <td><!-- Change Font Size -->
                <!--<p>Change text size here:</p>-->

```

```

                                <form id="textSizeOption"
action="/CopingGame/profile/" method="post">
                                {% csrf_token %}
                                <div>
                                    <button name="smallFont"
type="submit" value="{{ purchase.itemFKey.itemName }}"></button>
                                    <font size="5">Preview
                                </div>
                                <div>
                                    <button name="mediumFont"
type="submit" value="{{ purchase.itemFKey.itemName }}"></button>
                                    <font size="6">Preview
                                </div>
                                <div>
                                    <button name="largeFont"
type="submit" value="{{ purchase.itemFKey.itemName }}"></button>
                                    <font size="7">Preview
                                </div>
                                </form>
                                <font size="{{ player.text_size }}">
                            </td>
                            <td><!-- UPLOAD PICTURE -->
                                Change your profile picture here:
                                <form id="profilePicture"
action="/CopingGame/profile/" method="post" enctype="multipart/form-
data">
                                    {% csrf_token %}
                                    {{ profileForm.as_p }}
                                    <div><input type="submit" value="Upload"
name="upload"/></div>
                                </form>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </tr>
            <br>
            <tr>
                <!-- PLAYER PURCHASES -->
                <p align="center" style="text-
decoration:underline"><b>Here is a list of your total purchases</b><p>
            </tr>
            <tr>
                <form action="" method="post">
                    {% csrf_token %}

```

```

    {% if purchase_list %}
        <table align="center">

            <!-- Themes the player owns -->
            <tr>
                <td colspan="2" align="left"><b>Themes
Owned:</b></td>

                </tr>
                {% for purchase in purchase_list %}
                    {% if purchase.itemFKey.category ==
"Themes" %}

                        <!-- Button to switch theme, and
display a preview of the theme next to the button -->
                        <tr>
                            <td align="center"><button
name="theme" type="sumbit" value="{{ purchase.itemFKey.itemName
}}"/></button></td>
                            <td style="background-
color:{{ purchase.itemFKey.bg }}; color:{{ purchase.itemFKey.text
}}">{{ purchase.itemFKey.itemDesc }}</a></td>
                        </tr>
                    {% endif %}
                {% endfor %}

                <!-- Blank row between themes and pictures
-->

                <tr height="30px">
                </tr>

                <!-- Pictures the player owns -->
                <tr>
                    <td colspan="2"
align="left"><b>Pictures Owned:</b></td>
                </tr>
                {% for purchase in purchase_list %}
                    {% if purchase.itemFKey.category ==
"Pictures" %}

                        <!-- Button to switch picture,
and display the picture they own next to the button -->
                        <tr>
                            <td align="center"><button
name="picture" type="submit" value="{{
purchase.itemFKey.itemPicture.url }}"></button></td>
                            <td></td>
                        </tr>
                    {% endif %}

```

```

        {% endfor %}
    </table>
</tr>
<br>
<!-- Should never display since the player will have
default background and default user picture from the start -->
    {% else %}
        <p>You haven't purchased any items yet.</p>
    {% endif %}

</br>

</form>
</tr>
<tr>
<!-- A picture of DDRC's logo -->
    </a>
</tr>
</table>
</body>
</html>

<!DOCTYPE html>

<!-- The scenario index page -->
{% load staticfiles %}
<html>
<head>
    <title>Scenario Selection Page</title>
    <link rel="stylesheet" type="text/css" href="{% static
'CopingGame/style_home_page.css' %}" />
</head>
<body style="background-color:{{ player.fav_bg }}; color:{{
player.fav_text }}">
<font size="{{ player.text_size }}">
    <table id="page" align="center" style="background-color:{{
player.fav_bg }}; color:{{ player.fav_text }}">
        <tr>
            <!-- Table that will have the user profile once logged
in -->
            <table id="usertable" align="center"
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
 }}">
                <tr>
                    <td rowspan="2" colspan="2"></td>
                    <td>
                        {% if user.is_authenticated %}
                            <!-- user information -->

```

```

{{ user.get_username }}
<br>
:
{{ player.tokens }}

<br>
<a href="/logout/" style="color:{{
player.fav_text }}">Logout
{% else %}
    Not currently logged in, please log
in.
{% endif %}</td>
</tr>
<tr>
<td></td>
</tr>
</table>
</tr>
<br>
<tr>
<div id="links" align="center">
<table style="width:100%">
<col style="width:25%">
<col style="width:25%">
<col style="width:25%">
<col style="width:25%">
<!-- The button to return to the home page -->
<tr align="center">
<td align="center"><a href="/CopingGame/"
style="text-decoration: none">

</a></td>
<td align="center"><a
href="/CopingGame/profile/" style="text-decoration: none">

</a></td>
<td align="center"><a
href="/CopingGame/store_themes/" style="text-decoration: none">

</a></td>
<td align="center"><a
href="/CopingGame/help/" style="text-decoration: none">

```

```

                                
                                </a></td>
                                </tr>
                                <tr>
                                    <td align="center">Home Page</td>
                                    <td align="center">Profile Page</td>
                                    <td align="center">Store Page</td>
                                    <td align="center">Help Page</td>
                                </tr>
                                </table>
                                </div>
                                </tr>
                                <tr>
                                    <!-- The Header -->
                                    <h1>Please choose the scenario you want to play!</h1>
                                </tr>
                                <center>
                                    <tr><td style="vertical-align:middle;"><b>Press  next to the scenario you want to
play</b></td></tr>
                                    </br>
                                    </br>

                                    <table align="center">
                                        {% if scenario_list %}
                                        {% for scenario in scenario_list %}
                                            <tr>
                                                <td>
                                                    <a href="/CopingGame/{{
scenario.sceneID }}/">
                                                        
                                                        </a>
                                                </td>
                                                <td style="vertical-align:middle">
                                                    {{ scenario.title }}
                                                </td>
                                            </tr>
                                        {% endfor %}
                                        {% else %}
                                            <tr>No scenarios are available.</tr>
                                        {% endif %}
                                    </table>
                                    <br>
                                </center>

```

```

        <tr>
            <div align="left">
                <!-- A picture of DDRC's logo -->
                </a>
            </div>
        </tr>
    </table>
</body>
</html>

```

```

<!DOCTYPE html>

<!-- Store page for themes-->
{% load staticfiles %}

<html>
<head>
    <!-- Title the page and link the style sheet -->
    <title>Store Themes Page</title>
    <link rel="stylesheet" type="text/css" href="{% static
'CopingGame/style_store_page.css' %}">
</head>

<body style="background-color:{{ player.fav_bg }}; color:{{
player.fav_text }}"> <!-- Load players theme for page -->
<font size="{{ player.text_size }}">

    <!-- Page in a table so it can scale to any screen without much
issue -->
    <table id="page" align="center" style="background-color:{{
player.fav_bg }}; color:{{ player.fav_text }}">
        <tr>
            <!-- Table that has the user avatar, user name, and
their tokens -->
            <table id="usertable" align="center"
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
}}">
                <tr>
                    <td rowspan="2" colspan="2"></td>
                    <td>
                        {% if user.is_authenticated %}
                        <!-- user information -->
                        {{ user.get_username }}
                        <br>
                        :
                        {{ player.tokens }}
                    </td>
                </tr>
            </table>
        </tr>
    </table>

```

```

                                <a href="/logout/" style="color:{{
player.fav_text }}">Logout
                                {% else %}
                                Not currently logged in, please log
in.
                                {% endif %}</td>
                                </tr>
                                <tr>
                                <td></td>
                                </tr>
                                </table>
                                </tr>

                                </br>

                                <!-- Links to Home, Help, and Profile pages -->
                                <tr>
                                <div id="links" align="center">
                                <table style="width:100%">
                                <col style="width:25%">
                                <col style="width:25%">
                                <col style="width:25%">
                                <col style="width:25%">
                                <!-- The buttons go to the home page, profile page, or
help page -->
                                <tr align="center">
                                <td align="center"><a href="/CopingGame/"
style="text-decoration: none">
                                
                                </a></td>
                                <td align="center"><a
href="/CopingGame/scenarios/" style="text-decoration: none">
                                
                                </a></td>
                                <td align="center"><a href="/CopingGame/profile/"
style="text-decoration: none">
                                
                                </a></td>
                                <td align="center"><a href="/CopingGame/help/"
style="text-decoration: none">
                                <img class="link" src="{% static
'CopingGame/images/help5.png' %}"

```



```

style="width:100px;height:100px;border:0" style="color:{{
player.fav_text }}">
        </a></td>
    </tr>
    <tr>
        <td align="center">Home Page</td>
        <td align="center">Play a Scenario</td>
        <td align="center">Profile Page</td>
        <td align="center">Help Page</td>
    </tr>
</table>
</div>
</tr>

<!-- Page banner -->
<tr>
    <td align="center"><h1>Welcome to the Store</h1>
</tr>

<!-- Categories and Purchase areas -->
<tr>
    <td align="center">
        <!-- Category button's on the left -->
        <div id="categories">
            <ul id="catul">
                <li id="catli"><button class="current"
id="theme">Themes</button></li>
                <li id="catli"><a
href="/CopingGame/store_user_pictures/"><button class="other"
id="up">User Pictures</button></a></li>
            </ul>
        </div>

        <!-- Main portion of the page that displays purchase
options -->
        <div id="main">
            <div id="inner">
                <h2 style="color:{{ player.fav_text
}}">Click on the 
button to buy the theme to its left.</h2>
                <h2 style="color:{{ player.fav_text
}}">Click on the  button to
use the theme you own to its left.</h2>

                <!-- Themes table -->
                <table class="cattable" id="themes"
align="center">
                    <tr>
                        <td colspan="2" style="text-
decoration:underline;color:{{ player.fav_text }}">Themes - 50  Each</td>
</tr>

<!-- Make list of all themes -->
{% for theme in theme_list %}
    <!-- Make a row for each theme -
->
        <tr>
            <!-- Have a preview of the
theme in the left column -->
                <td bgcolor="{ {
theme.itemFKey.bg }}" style="color:{ { theme.itemFKey.text
}}">PREVIEW</td>

                    <!-- See if the player owns
the theme -->
                    {% if theme.owned == True
%}
                        <!-- Display apply
button since they own the theme-->
                        <form action=""
method="post">
                            {% csrf_token %}
                            <td><button
class="tabbut" type="submit" name="change" value="{ {
theme.itemFKey.itemName }}"></button></td>
                                </form>

                                    <!-- If this is reached,
then they don't own the item -->
                                    {% else %}
                                        <!-- See if they have
enough tokens for the item -->
                                        {% if player.tokens
>= 50 %}
                                            <!-- Display
purchase button -->
                                            <form action=""
method="post">
                                                {%
csrf_token %}
                                                    <td><button class="tabbut" type="submit" name="buy" value="{ {
theme.itemFKey.itemName }}"></button></td>
                                                        </form>

```

```

have enough tokens, say they don't -->
                                <!-- If they don't
                                {% else %}
                                <td
style="color:{{ player.fav_text }}">Not enough .</td>
                                {% endif %}
                                {% endif %}
                                </tr>
                                {% endfor %}
                                </table>
                                </div>
                                </div>
                                </tr>
                                <tr>
                                <!-- A picture of DDRC's logo -->
                                <div align="left">
                                
                                </div>
                                </tr>
                                </table>

</body>
</html>

<!DOCTYPE html>

<!-- Store page for user pictures -->
{% load staticfiles %}

<html>
<head>
    <!-- Title the page and link the style sheet -->
    <title>Store User Pictures Page</title>
    <link rel="stylesheet" type="text/css" href="{% static
'CopingGame/style_store_page.css' %}">
</head>

<body style="background-color:{{ player.fav_bg }}; color:{{
player.fav_text }}"> <!-- Load players theme for page -->
<font size="{{ player.text_size }}">

    <!-- Page in a table so it can scale to any screen without much
issue -->
    <table id="page" align="center" style="background-color:{{
player.fav_bg }}; color:{{ player.fav_text }}">
        <tr>

```

```

        <!-- Table that has the user avatar, user name, and
their tokens -->
        <table id="usertable" align="center"
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
}}">
            <tr>
                <td rowspan="2" colspan="2"></td>
                <td>
                    {% if user.is_authenticated %}
                    <!-- user information -->
                    {{ user.get_username }}
                    <br>
                    :
                    {{ player.tokens }}
                    <br>
                    <a href="/logout/" style="color:{{
player.fav_text }}">Logout
                    {% else %}
                    Not currently logged in, please log
in.
                    {% endif %}</td>
                </tr>
                <tr>
                <td></td>
                </tr>
            </table>
        </tr>

        </br>

        <!-- Links to Home, Help, and Profile pages -->
        <tr>
            <div id="links" align="center">
            <table style="width:100%">
                <col style="width:25%">
                <col style="width:25%">
                <col style="width:25%">
                <col style="width:25%">
            <!-- The buttons go to the home page, profile page, or
help page -->
            <tr align="center">
                <td align="center"><a href="/CopingGame/"
style="text-decoration: none">
                    
                    </a></td>

```

```
 <a href="/CopingGame/scenarios/" style="text-decoration: none">          </a></td>  <a href="/CopingGame/profile/" style="text-decoration: none">          </a></td>  <a href="/CopingGame/help/" style="text-decoration: none">          </a></td> </tr> <tr>   | | | |
```

```

        <div id="main">
            <div id="inner">
                <h2 style="color:{{ player.fav_text
}}">Click on the 
button to buy the picture to its left.</h2>
                <h2 style="color:{{ player.fav_text
}}">Click on the  button to
use the picture you own to its left.</h2>

                <!-- Themes table -->
                <table class="cattable" id="themes"
align="center">
                    <tr>
                        <td colspan="2" style="text-
decoration:underline;color:{{ player.fav_text }}">User Pictures - 50
 Each</td>
                    </tr>

                    <!-- Make list of all pictures -->
                    {% for picture in picture_list %}
                        <!-- Make a row for each picture
-->
                            <tr>
                                <!-- Have a preview of the
picture in the left column -->
                                    <td></td>

                                    <!-- See if the player owns
the picture -->
                                        {% if picture.owned == True
%}

                                            <!-- Display apply
button since they own the picture-->
                                                <form action=""
method="post">
                                                    {% csrf_token %}
                                                    <td><button
class="tabbut" type="submit" name="change" value="{{
picture.itemFKey.itemName }}"></button></td>

                                                        </form>

                                <!-- If this is reached,
then they don't own the item -->
                                    {% else %}

```

```

enough tokens for the item -->
>= 50 %}

purchase button -->
method="post">
csrf_token %}

        <td><button class="tabbut" type="submit" name="buy" value="{{
picture.itemFKey.itemName }}"></button></td>

                                </form>

                                <!-- If they don't
                                have enough tokens, say they don't -->
                                {% else %}
                                <td
                                style="color:{{ player.fav_text }}">Not enough .</td>
                                {% endif %}
                                {% endif %}
                                </tr>
                                {% endfor %}
                                </table>
                                </div>
                                </div>
                                </tr>

                                <tr>
                                <!-- A picture of DDRC's logo -->
                                <div align="left">
                                
                                </div>
                                </tr>
                                </table>

</body>
</html>

<!DOCTYPE html>

<!-- The victory page -->
{% load staticfiles %}
<html>
<head>

```

```

        <title>Victory!</title>
        <link rel="stylesheet" type="text/css" href="{% static
'CopingGame/style_home_page.css' %}" />
    </head>
    <body style="background-color:{{ player.fav_bg }}; color:{{
player.fav_text }}">
    <font size="{{ player.text_size }}">
        <table id="page" align="center" style="background-color:{{
player.fav_bg }}; color:{{ player.fav_text }}">
            <tr>
                <!-- Table that will have the user profile once logged
in -->
                <table id="usertable" align="center"
style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text
}}">
                    <tr>
                        <td rowspan="2" colspan="2"></td>
                        <td>
                            {% if user.is_authenticated %}
                            <!-- user information -->
                            {{ user.get_username }}
                            <br>
                            :
                            {{ player.tokens }}
                            <br>
                            <a href="/logout/" style="color:{{
player.fav_text }}">Logout
                            {% else %}
                                Not currently logged in, please log
                                in.
                            {% endif %}</td>
                        </tr>
                    </tr>
                </table>
            </tr>
        </table>
    </tr>
    <br>
    <tr>
        <div id="links" align="center">
            <table style="width:100%">
                <col style "width:20%">
                <col style="width:20%">
                <col style="width:20%">
                <col style="width:20%">
                <col style="width:20%">
                <!-- The button to return to the home page -->
            <tr align="center">

```



```
 <a href="/CopingGame/" style="text-decoration: none">          </a></td>  <a href="/CopingGame/scenarios/" style="text-decoration: none">          </a></td>  <a href="/CopingGame/profile/" style="text-decoration: none">          </a></td>  <a href="/CopingGame/store_themes/" style="text-decoration: none">          </a></td>  <a href="/CopingGame/help/" style="text-decoration: none">          </a></td> </tr> <tr>   | | | | | |
```

```

        <center>
        <td align="center">
            <!-- CONGRATULATIONS Picture -->
            
        </td>
    </tr>
    <br>
    <tr>
        <td align="center">
            <p align="center">You completed the scenario and
solved all the problems!</p>
            <p align="center">You've earned {{
player.scenario_tokens }} .</p>
        </td>
    </tr>
    <br>
    <table>
    <tr>
        <td>
            <a href="/CopingGame/scenarios/" style="text-
decoration: none">
                
            </a>
        </td>
        <td style="vertical-align:middle">
            Play another scenario
        </td>
    </tr>
    </table>
    <br>
    <tr>
        <div align="left">
            <!-- A picture of DDRC's logo -->
            </a>
        </div>
    </tr>
    </table>
</body>
</html>

```

//Javascript file for the store page

//Determines which button is highlighted
var selectedbtn;

```

//Change button formatting and determines which table is visible
function chkbtn(btn, tab) {
    //Get all the category buttons in an array
    var catbuttons = document.getElementsByClassName('catbutton');

    //Set which button was selected
    selectedbtn = btn;

    //Format the buttons based on if it was selected or not
    for(var i = 0; i < catbuttons.length; i++) {
        if(catbuttons[i].id === selectedbtn.id) {
            catbuttons[i].style.background = '#ffc';
            catbuttons[i].style.color = 'black';
        }
        else {
            catbuttons[i].style.background = '#6d1d1c';
            catbuttons[i].style.color = 'white';
        }
    }

    //Get all possible category tables
    var cattables = document.getElementsByClassName('cattable');

    //Set which one should be displayed
    var selectedtab = tab;

    //Make the selected one visible, make all others not visible
    for(var i = 0; i < cattables.length; i++) {
        if(cattables[i].id === selectedtab.id) {
            cattables[i].style.display = 'table';
        }
        else {
            cattables[i].style.display = 'none';
        }
    }
}

//Determines what happens if the mouse is over the button
function overbtn(btn) {
    btn.style.background = '#ffc';
    btn.style.color = 'black';
}

//Determines what happens when the mouse leaves the button
function outbtn(btn) {
    if(btn !== selectedbtn) {
        btn.style.background = '#6d1d1c';
        btn.style.color = 'white';
    }
}

```

```

/* CSS style sheet for the game page */

/* Style of the background and margins */
body {
    background-color: #ffc;
    margin: 20px;
    padding: 0;
}

/* Style of the "scenario" bar */
h1 {
    color: #ffc;
    background-color: #6d1d1c;
    /* font-size: 40px; */
    margin: 0;
    margin-bottom: 7px;
    padding: 4px;
    font-style: bold;
    text-align: center;
    letter-spacing: 8px;
}

/* Style of the "problem" bar */
h2 {
    text-align: center;
    /*font-size: 32px;*/
    font-style: bold;
}

/* Style of the user table at the top of the page */
#usertable {
    background-color: ffc;
    border: 1px solid black;
}

/* Style of the pic associated with the problem */
#problempic {
    display: block;
    margin-left: auto;
    margin-right: auto;
}

/* Style of the table containing coping technique options */
#coping {
    text-align: center;
    /* font-size: 24px; */
}

/* Style of "How to handle this problem" row of the coping table */
#handle {
    height: 50px;
}

```

```

        /* font-size: 32px; */
    }

    /* Style of the coping technique options (text only) of the coping
    table */
    #techniques {
        height: 40px;
        /* font-size: 20px; */
        font-style: italic;
    }

    /* Add blue background when hover over */
    button:hover {
        background: DeepSkyBlue;
    }

    a:hover {
        border-color: #999;
        box-shadow: 0 1px 3px rgba(0,0,0,0.25);
    }

    .box {
        width: 20%;
        margin: 0 auto;
        background: rgba(255,255,255,0.2);
        padding: 35px;
        border: 2px solid #fff;
        border-radius: 20px/50px;
        background-clip: padding-box;
        text-align: center;
    }

    .button {
        font-size: 1em;
        padding: 10px;
        color: #fff;
        border: 2px solid #06D85F;
        border-radius: 20px/50px;
        text-decoration: none;
        cursor: pointer;
        transition: all 0.3s ease-out;
    }

    .button:hover {
        background: #06D85F;
    }

    .overlay {
        position: absolute;
        top: 0;
        bottom: 0;
        left: 0;
        right: 0;
    }

```

```

    background: rgba(0, 0, 0, 0.7);
    transition: opacity 500ms;
    visibility: hidden;
    opacity: 0;
}
.overlay:target {
    visibility: visible;
    opacity: 1;
}

.popup {
    margin: 70px auto;
    padding: 20px;
    background: #fff;
    border-radius: 5px;
    width: 30%;
    position: relative;
    transition: all 5s ease-in-out;
}

.popup h3{
    margin-top: 0;
    color: #333;
    font-family: Tahoma, Arial, sans-serif;
}

.popup .close {
    position: absolute;
    top: 20px;
    right: 30px;
    transition: all 200ms;
    font-size: 30px;
    font-weight: bold;
    text-decoration: none;
    color: #333;
}

.popup .close:hover {
    color: #06D85F;
}

.popup .content {
    max-height: 30%;
    overflow: auto;
}

@media screen and (max-width: 700px){
    .box{
        width: 70%;
    }
    .popup{
        width: 70%;
    }
}

```

```

.modalDialog {
    position: fixed;
    font-family: Arial, Helvetica, sans-serif;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    background: rgba(0,0,0,0.8);
    z-index: 99999;
    opacity:0;
    -webkit-transition: opacity 400ms ease-in;
    -moz-transition: opacity 400ms ease-in;
    transition: opacity 400ms ease-in;
    pointer-events: none;
}

.modalDialog:target {
    opacity:1;
    pointer-events: auto;
}

.modalDialog > div {
    width: 400px;
    position: relative;
    margin: 10% auto;
    padding: 5px 20px 13px 20px;
    border-radius: 10px;
}

.close {
    background: #606061;
    color: #FFFFFF;
    line-height: 25px;
    position: absolute;
    right: -12px;
    text-align: center;
    top: -10px;
    width: 24px;
    text-decoration: none;
    font-weight: bold;
    -webkit-border-radius: 12px;
    -moz-border-radius: 12px;
    border-radius: 12px;
    -moz-box-shadow: 1px 1px 3px #000;
    -webkit-box-shadow: 1px 1px 3px #000;
    box-shadow: 1px 1px 3px #000;
}

.close:hover { background: #00d9ff; }

```

```

/* CSS style sheet for the home page*/

/* Style of the background and margins */
body {
    background-color: #eddedd;
    margin: 20px;
    padding: 0;
}

/* Style of the "coping skills game" bar */
h1 {
    color: #ffc;
    background-color: #6d1d1c;
    font-size: 40px;
    margin: 0;
    margin-bottom: 7px;
    padding: 4px;
    font-style: bold;
    text-align: center;
    letter-spacing: 8px;
}

/* Style of the "you must log in" bar */
h2 {
    color: #4a6530;
    text-align: center;
    font-size: 32px;
    font-style: bold;
}

/* Style of the user table at the top of the page */
#usertable {
    background-color: ffc;
    border: 1px solid black;
}

/* Style of the table for the button that goes to the login page */
#loginbutton {
    text-align: center;
    font-size: 20px;
}

/* Add blue background when hover over */
button:hover {
    background: DeepSkyBlue;
}

.link {
    width: 150px;
    border: 1px solid #9325BC;
    padding: 10px;
}

```



```

.link:hover {
    -moz-box-shadow: 0 0 10px DeepSkyBlue;
    -webkit-box-shadow: 0 0 10px DeepSkyBlue;
    box-shadow: 0 0 10px DeepSkyBlue;
    background: DeepSkyBlue;
}

.scenario{
    width: 75px;
    border: 1px solid;
    padding: 10px;
}

.scenario:hover {
    -moz-box-shadow: 0 0 10px #00CC00;
    -webkit-box-shadow: 0 0 10px #00CC00;
    box-shadow: 0 0 10px #00CC00;
    background: #00CC00;
}

/*a:hover {
    background:DeepSkyBlue;
}*/

/* CSS style sheet for the prototype home page once logged in */

/* Style of the background and margins */
body {
    background-color: #ededed;
    margin: 20px;
    padding: 0;
}

/* Style of the "coping skills game" bar */
h1 {
    color: #ffc;
    background-color: #6d1d1c;
    font-size: 40px;
    margin: 0;
    margin-bottom: 7px;
    padding: 4px;
    font-style: bold;
    text-align: center;
    letter-spacing: 8px;
}

/* Style of the "welcome!" bar */
h2 {
    color: #4a6530;

```

```

        text-align: center;
        font-size: 32px;
        font-style: bold;
    }

    /* Style of the user table at the top of the page */
    #usertable {
        background-color: ffc;
        border: 1px solid black;
    }

    /* Style of the table for the button that goes to the game page */
    #gamebutton {
        text-align: center;
        font-size: 20px;
    }

    /* Style of the table for the button that goes to the store page */
    #storebutton {
        text-align: center;
        font-size: 20px;
    }

    /* Add blue background when hover over */
    button:hover {
        background:DeepSkyBlue;
    }

    a:hover {
        background:DeepSkyBlue;
    }

    /* CSS style sheet for the login page*/

    /* Style of the background and margins */
    body {
        background-color: #ededed;
        margin: 20px;
        padding: 0;
    }

    /* Style of the "coping skills game" bar */
    h1 {
        color: #ffc;
        background-color: #6d1d1c;
        font-size: 40px;
        margin: 0;
        margin-bottom: 7px;
        padding: 4px;
        font-style: bold;
        text-align: center;
    }

```

```

        letter-spacing: 8px;
    }

    /* Add blue background when hover over */
    button:hover {
        background:DeepSkyBlue;
    }

    a:hover {
        background:DeepSkyBlue;
    }

    /* CSS style sheet for the profile page*/

    /* Style of the background and margins */
    body {
        background-color: #ffc;
        margin: 20px;
        padding: 0;
    }

    /* Style of the Player Profile bar */
    h1 {
        color: #ffc;
        background-color: #6d1d1c;
        font-size: 40px;
        margin: 0;
        margin-bottom: 7px;
        padding: 4px;
        font-style: bold;
        text-align: center;
        letter-spacing: 8px;
    }

    /* Style of the user table at the top of the page */
    #usertable {
        background-color: ffc;
        border: 1px solid black;
    }

    /* Table with upload picture and change font size */
    table.t1 {
        width:100%
    }

    .t1 td {
        padding: 0;
    }

    /* Add blue background when hover over */

```

```

button:hover {
    background:DeepSkyBlue;
}

.link {
    width: 150px;
    border: 1px solid #9325BC;
    padding: 10px;
}

.link:hover {
    -moz-box-shadow: 0 0 10px DeepSkyBlue;
    -webkit-box-shadow: 0 0 10px DeepSkyBlue;
    box-shadow: 0 0 10px DeepSkyBlue;
    background: DeepSkyBlue;
}

/* CSS style sheet for the store pages */

/* Style of the background and margins */
body {
    background-color: #ededed;
    margin: 20px;
    padding: 0;
}

/* Style of the "welcome" bar */
h1 {
    color: #ffc;
    background-color: #6d1d1c;
    font-size: 40px;
    margin: 0;
    margin-bottom: 7px;
    padding: 4px;
    font-style: bold;
    text-align: center;
    letter-spacing: 8px;
}

/* Style of the "click to purchase" bar */
h2 {
    color: black;
    font-size: 30px;
    text-align: center;
}

/* Style of the "categories" title */
h3 {
    text-align: center;
    text-decoration: underline;
    margin: 0;
}

```

```

        padding: 0;
        font-size: 25px;
    }

/* Style of the user table at the top of the page */
#usertable {
    background-color: #ededed;
    border: 1px solid black;
}

/* style of the "themes" table */
#themes {
    font-size: 6; /* 25px */
    border-collapse: separate;
    border-spacing: 1em;
    /* display: none; */
    text-align: center;
    font-weight: bold;
}

/* style of the "userpics" table */
#userpics {
    font-size: 25px;
    border-collapse: separate;
    border-spacing: 1em;
    /* display: none; */
    text-align: center;
    font-weight: bold;
}

/* Style of the categories bar */
#categories {
    width: 15%;
    float: left;
    display: inline;
    margin: 0;
    padding: 0;
    margin-right: 10px;
    background-color: #6d1d1c;
    color: #ffc;
    vertical-align: top;
}

#catul {
    list-style-type: none;
    font-size: 6; /* 20px */
    margin: 0;
    padding: 10px;
}

#catli {
    text-align: center;

```

```

        padding: 5px;
    }

/* Everything right of the categories menu */
#main {
    width: 85%;
    padding: 0;
    display: inline;
    margin: auto auto;
    vertical-align: top;
}

/* Center the purchase table */
#inner {
    margin-right: 15%;
}

.current {
    border: 1px solid #ffc;
    background: #ffc;
    padding: 7px 14px;
    color: #000000;
    font-size: 100%; /*20px;*/
    font-family: Georgia, serif;
    text-decoration: none;
    vertical-align: middle;
    width: 100%;
}

.other {
    border: 1px solid #ffc;
    background: #6d1d1c;
    padding: 7px 14px;
    color: white;
    font-size: 100%; /*20px;*/
    font-family: Georgia, serif;
    text-decoration: none;
    vertical-align: middle;
    width: 100%;
}

.other:hover {
    background: #ffc;
    color: #000000;
}

.other:focus {
    background: #ffc;
    color: #000000;
}

```

```
/* Add blue background when hover over */
.tabbut:hover {
    background:DeepSkyBlue;
}

.link {
    width: 150px;
    border: 1px solid #9325BC;
    padding: 10px;
}

.link:hover {
    -moz-box-shadow: 0 0 10px DeepSkyBlue;
    -webkit-box-shadow: 0 0 10px DeepSkyBlue;
    box-shadow: 0 0 10px DeepSkyBlue;
    background: DeepSkyBlue;
}
```

```

#manage.py
#!/usr/bin/env python
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE",
        "pluvali.settings")

    from django.core.management import execute_from_command_line

    execute_from_command_line(sys.argv)

#admin.py
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from django.contrib.auth.forms import UserChangeForm, UserCreationForm
from django.contrib.auth.models import Group
from CopingGame.models import *
from django import forms

class SolutionsInline(admin.TabularInline):
    model = Problems.solutions.through
    extra = 3

class ProblemAdmin(admin.ModelAdmin):
    inlines = [SolutionsInline]
    exclude = ('solutions',)

class ProblemsInline(admin.TabularInline):
    model = Scenario.problems.through
    extra = 3

class Player_listInline(admin.TabularInline):
    model = Scenario.player_list.through
    extra = 3

class Group_listInline(admin.TabularInline):
    model = Scenario.group_list.through
    extra = 3

class ScenarioAdmin(admin.ModelAdmin):
    inlines = [ProblemsInline, Player_listInline, Group_listInline]
    exclude = ('problems', 'player_list', 'group_list')

class PlayerChangeForm(UserChangeForm):
    class Meta(UserChangeForm.Meta):
        model = Player

class PlayerCreationForm(UserCreationForm):
    class Meta(UserCreationForm.Meta):

```



```

        model = Player

    def clean_username(self):
        username = self.cleaned_data['username']
        try:
            Player.objects.get(username=username)
        except Player.DoesNotExist:
            return username
        raise
forms.ValidationError(self.error_messages['duplicate_username'])

class PlayerAdmin(UserAdmin):
    form = PlayerChangeForm
    add_form = PlayerCreationForm
    fieldsets = UserAdmin.fieldsets + (
        (None, {'fields': ('tokens', 'avatarPic',
'fav_bg', 'fav_text', 'text_size',)})),
    )

admin.site.unregister(Group)
admin.site.register(Player, PlayerAdmin)
admin.site.register(PlayerGroup)
admin.site.register(Scenario, ScenarioAdmin)
admin.site.register(Problems, ProblemAdmin)
admin.site.register(Solutions)
admin.site.register(Store)
admin.site.register(Purchase)

#apps.py
from django.apps import AppConfig

class MyAppConfig(AppConfig):
    name = 'CopingGame'

    def ready(self):
        import CopingGame.signals

#forms.py
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from CopingGame.models import Player

class PlayerProfileForm(forms.ModelForm):
    class Meta:
        model = Player
        fields = ('avatarPic',)

class MyRegistrationForm(UserCreationForm):
    email = forms.EmailField(required=False)

    class Meta:

```

```

        model = Player
        fields = ('username', 'email', 'password1', 'password2')

    def save(self, commit=True):
        player=super(MyRegistrationForm, self).save(commit=False)
        player.email=self.cleaned_data['email']
        if commit:
            player.save()
        return player

#models.py
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.core.files.storage import FileSystemStorage

#NOTE ImageField requires Pillow
# run from pluvali dir to install>> pip install pillow

class Player(AbstractUser):#Extended from User class
    tokens = models.IntegerField(default=0)
    avatarPic = models.ImageField(upload_to='userPic/',
default='userPic/default-user.png')
    fav_bg = models.CharField(max_length=30, default='#ededed')
    fav_text = models.CharField(max_length=30, default='black')
    text_size = models.IntegerField(default=6)
    stage = models.IntegerField(default=0) #used for iterating
through scenarios
    scenario_tokens = models.IntegerField(default=0) #used for
tracking player's tokens in a scenario
    tokens_earned = models.CharField(max_length=9,
default="0,0,0,0,0") #Used to randomly generate tokens earned in a
scenario
    class Meta:
        verbose_name = 'Player'
        verbose_name_plural = 'Players'
    def passw (self):
        return self.password
    def __str__(self):
        return self.username
    pass

class PlayerGroup(models.Model):
    groupName = models.CharField(max_length=35, default="Admin's
Group")
    player = models.ManyToManyField(Player)
    def __str__(self):
        return self.groupName
    pass

class Solutions(models.Model):

```

```

        pictures = models.ImageField(upload_to='solutions/', blank=True,
null=True)
        solution = models.CharField(max_length=300, default="Solution")
        sVideoId = models.CharField(max_length=50, default="q_g7s2oBzCw")
#Clorox commercial, change this
        def __str__(self):
            return self.solution
        class Meta:
            verbose_name_plural = 'Solutions'
        pass

class Problems(models.Model):
    pictureP = models.ImageField(upload_to='problems/', null=True)
    problem = models.CharField(max_length=225, default="The
Problem.")
    solutions = models.ManyToManyField(Solutions)
    pVideoId = models.CharField(max_length=50, default="q_g7s2oBzCw")
#Clorox commercial, change this
    def __str__(self):
        return self.problem
    class Meta:
        verbose_name_plural = 'Problems'
    pass

class Scenario(models.Model):
    sceneID = models.AutoField(primary_key=True)
    title = models.CharField(max_length=35, default="Scenario Title")
    problems = models.ManyToManyField(Problems)
    player_list = models.ManyToManyField(Player, blank=True)
    group_list = models.ManyToManyField(PlayerGroup, blank=True)
    def __str__(self):
        return self.title

class Store(models.Model):
    itemKey = models.AutoField(primary_key=True)
    category = models.CharField(max_length=20, default="Themes")
    itemName = models.CharField(max_length=15, default="ItemName")
    itemDesc = models.CharField(max_length=50, default="Item
Description")
    itemPicture = models.ImageField(upload_to='store/', blank=True,
null=True)
    bg = models.CharField(max_length=10, blank=True)
    text = models.CharField(max_length=10, blank=True)
    def __str__(self):
        return self.itemName
    class Meta:
        verbose_name_plural = 'StoreItems'

class Purchase(models.Model):
    player = models.ForeignKey(Player)
    itemFKey = models.ForeignKey(Store)
    owned = models.BooleanField(default=False)

```

```

    def __str__(self):
        name = self.player.username+'-'+self.itemFKey.itemName
        return name

def upload_path_handler(self):
    return

from CoppingGame.signals import *

#signals.py
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.conf import settings
from CoppingGame.models import Player, Store, Purchase

@receiver(post_save, sender=Store)
def model_post_save(sender, instance, created, **kwargs):
    if created:
        players_list = list(Player.objects.all())
        for p in players_list:
            purchase = Purchase.objects.create(player=p,
            itemFKey=instance, owned=False)
            purchase.save()

#@receiver(pre_save, sender=Player)
#def upload_picture_pre_save(sender, instance, created, **kwargs):
#    if created:

#urls.py
from django.conf.urls import url, patterns
from django.core.urlresolvers import reverse
from CoppingGame import views

urlpatterns = [
    url(r'^$', views.index, name='index'), #homepage
    url(r'^help/', views.help, name='help'), #help page
    url(r'^scenarios/', views.scenario_index, name='scenarios'),
#scenario index
    url(r'^(?P<sceneID>\d+)/$', views.game, name='game'), #play game
page
    url(r'^victory/', views.victory, name='victory'),
    url(r'^store_themes/', views.store_themes, name='store_themes'),
#store theme page
    url(r'^store_user_pictures/', views.store_user_pictures,
name='store_user_pictures'), #store user pictures page
    url(r'^admin_page/', views.admin_page, name='admin_page/'),
    url(r'^profile/', views.profile, name='profile'),
]

#views.py
from django.http import HttpResponseRedirect
from django.contrib.auth.decorators import login_required

```

```

from django.contrib.auth import authenticate, login, logout
from django.shortcuts import redirect, render, get_object_or_404
from django.core.urlresolvers import reverse_lazy
from django.core.context_processors import csrf
from django.db.models import Q
from django import forms
from django.contrib.auth.forms import UserCreationForm
from CopingGame.forms import PlayerProfileForm
import random

from CopingGame.models import Player, Scenario, Problems, Solutions,
Store, Purchase

#login page uses default Django sessions

#link for homepage, requires login
@login_required(login_url='/login')
def index(request):
    player = Player.objects.get(username=request.user)
    context = {'player':player}
    return render(request, 'CopingGame/index.html', context)

#link for help page
def help(request):
    player = Player.objects.get(username=request.user)
    context = {'player':player}
    return render(request, 'CopingGame/help_page.html', context)

#link for scenario index
@login_required(login_url='/login')
def scenario_index(request):
    player = Player.objects.get(username=request.user)
    #Pull scenario list by associated to the PLAYER by Player_List or
    Player_Group_List
    scenario_list = Scenario.objects.filter(Q(player_list=player) |
    Q(group_list__player=player)).distinct()
    #set stage to 0 in case viewing different scenario than
    previously.
    player.stage = 0
    player.scenario_tokens = 0
    player.tokens_earned = str("0,0,0,0,0")
    player.save()
    context = {'scenario_list': scenario_list, 'player':player}
    return render(request, 'CopingGame/scenario_index.html', context)

#link for admin page
@login_required(login_url='/login')
def admin_page(request):
    player_list = Player.objects.order_by('username').distinct()
    context = {'player_list':player_list}
    return render(request, 'CopingGame/admin_page.html', context)

```

```

#link for game page
@login_required(login_url='/login')
def game(request, sceneID):
    player = Player.objects.get(username=request.user)
    scene = get_object_or_404(Scenario, pk = sceneID)
    #In case Player navigates by typing scenario in url
    max_stage = scene.problems.count()
    if(player.stage > max_stage):
        player.stage = 0
        player.save()
    #Check if player is on first stage and tokens_earned hasn't been
    populated
    if(player.stage == 0 and player.tokens_earned ==
    str("0,0,0,0,0")):
        #populate tokens earned for this scenario
        player.tokens_earned =
        random.randint(2,5),random.randint(2,5),random.randint(2,5),random.ran
        dint(2,5),random.randint(2,5)
        player.save()
        #Parse into list
        tokens_list =
        str(player.tokens_earned).lstrip('(').rstrip(')').split(',')
        #Get max number of stages in scenario
        stage0 = scene.problems.all()[0]
        if(max_stage >= 2):
            stage1 = scene.problems.all()[1]
        if(max_stage >= 3):
            stage2 = scene.problems.all()[2]
        if(max_stage >= 4):
            stage3 = scene.problems.all()[3]
        if(max_stage >= 5):
            stage4 = scene.problems.all()[4]
        #Fill context
        if(max_stage == 1):
            context = {'scene':scene, 'player':player, 'stage0':stage0,
            'tokens0':tokens_list[0] }
        if(max_stage == 2):
            context = {'scene':scene, 'player':player,
            'stage0':stage0,'stage1':stage1, 'tokens0':tokens_list[0],
            'tokens1':tokens_list[1]}
        if(max_stage == 3):
            context = {'scene':scene, 'player':player,
            'stage0':stage0,'stage1':stage1,'stage2':stage2,
            'tokens0':tokens_list[0], 'tokens1':tokens_list[1],
            'tokens2':tokens_list[2]}
        if(max_stage == 4):
            context = {'scene':scene, 'player':player,
            'stage0':stage0,'stage1':stage1,'stage2':stage2,'stage3':stage3,
            'tokens0':tokens_list[0], 'tokens1':tokens_list[1],
            'tokens2':tokens_list[2], 'tokens3':tokens_list[3]}
        if(max_stage == 5):

```

```

        context = {'scene':scene, 'player':player,
'stage0':stage0,'stage1':stage1,'stage2':stage2,'stage3':stage3,'stage
4':stage4, 'tokens0':tokens_list[0], 'tokens1':tokens_list[1],
'tokens2':tokens_list[2], 'tokens3':tokens_list[3],
'tokens4':tokens_list[4]}
        #When player selects an answer, award tokens and move to next
stage
        if request.method == 'POST':
            if(player.stage == 0):
                player.scenario_tokens = int(tokens_list[0])
                player.tokens += int(tokens_list[0])
            if(player.stage == 1):
                player.scenario_tokens += int(tokens_list[1])
                player.tokens += int(tokens_list[1])
            if(player.stage == 2):
                player.scenario_tokens += int(tokens_list[2])
                player.tokens += int(tokens_list[2])
            if(player.stage == 3):
                player.scenario_tokens += int(tokens_list[3])
                player.tokens += int(tokens_list[3])
            if(player.stage == 4):
                player.scenario_tokens += int(tokens_list[4])
                player.tokens += int(tokens_list[4])
            #Move player to next stage in scenario
            player.stage += 1
            player.save()
            if(player.stage == max_stage):
                return HttpResponseRedirect("/CopingGame/victory/")
        return render(request, 'CopingGame/game_page.html', context)

@login_required(login_url='/login')
def victory(request):
    player = Player.objects.get(username=request.user)
    player.stage = 0
    player.save()
    context = {'player':player}
    return render(request, 'CopingGame/victory_page.html', context)

#Store themes section
@login_required(login_url='/login')
def store_themes(request):
    player = Player.objects.get(username=request.user)
    purchase_list = Purchase.objects.filter(player=player)
    theme_list = purchase_list.filter(itemFKey__category='Themes')
    theme_names = theme_list.values('itemFKey__itemName').distinct()
    context = {'player':player, 'purchase_list':purchase_list,
'theme_list':theme_list, 'theme_names':theme_names}
    if request.method == 'POST':
        if 'change' in request.POST:
            theme =
Store.objects.get(itemName=request.POST.get("change","")) #Get the
specific theme from store items

```

```

        player.fav_bg = theme.bg #Set background color to the
themes background color
        player.fav_text = theme.text #Set the text color to
the themes text color
        player.save() #Save the player settings
    if 'buy' in request.POST:
        player.tokens -= 50 #Deduct the players token total
        theme =
purchase_list.get(itemFKey__itemName=request.POST.get("buy","")) #Get
the specific theme the user wishes to buy
        theme.owned = True #Say that the player now owns the
theme

        theme.save() #Save the theme purchase
        player.fav_bg = theme.itemFKey.bg #Set background
color to the themes background color
        player.fav_text = theme.itemFKey.text #Set the text
color to the themes text color
        player.save() #Save the player settings
    return render(request, 'CopingGame/store_themes.html', context)

#Store user pictures section
@login_required(login_url='/login')
def store_user_pictures(request):
    player = Player.objects.get(username=request.user)
    purchase_list = Purchase.objects.filter(player=player)
    picture_list =
purchase_list.filter(itemFKey__category='Pictures')
    picture_names =
picture_list.values('itemFKey__itemName').distinct()
    context = {'player':player, 'purchase_list':purchase_list,
'picture_list':picture_list, 'picture_names':picture_names}
    if request.method == 'POST':
        if 'change' in request.POST:
            picture =
Store.objects.get(itemName=request.POST.get("change","")) #Get the
specific picture from store items
            player.avatarPic = picture.itemPicture.url #Change the
users picture

            player.save() #Save the player settings
        if 'buy' in request.POST:
            player.tokens -= 50 #Deduct the players token total
            picture =
purchase_list.get(itemFKey__itemName=request.POST.get("buy","")) #Get
the specific picture the user wishes to buy
            picture.owned = True #Say that the player now owns the
picture

            picture.save() #Save the theme purchase
            player.avatarPic = picture.itemFKey.itemPicture.url
#Change the users picture to the purchased one
            player.save() #Save the player settings
    return render(request, 'CopingGame/store_user_pictures.html',
context)

```



```

from CopingGame.forms import MyRegistrationForm
#Player registration
def register(request):
    if request.method == 'POST':
        form = MyRegistrationForm(request.POST)
        player_email = request.POST['email']
        if form.is_valid():
            #create new PLAYER from form
            new_player = form.save()
            new_player =
authenticate(username=request.POST['username'],
password=request.POST['password1'])
            #add all store items to account
            store_list = list(Store.objects.all())
            for item in store_list:
                #set default purchases to owned
                if item.itemName == 'DefaultBlack':
                    purchase =
Purchase.objects.create(player=new_player, itemFKey=item, owned=True)
                    purchase.save()
                    new_player.fav_bg = item.bg
                    new_player.fav_text = item.text
                    new_player.save()
                elif item.itemName == 'DefaultPicture':
                    purchase =
Purchase.objects.create(player=new_player, itemFKey=item, owned=True)
                    purchase.save()
                    new_player.avatarPic = item.itemPicture
                    new_player.save()
                #set other purchases to not owned
                else:
                    purchase =
Purchase.objects.create(player=new_player, itemFKey=item, owned=False)
                    purchase.save()
                    #give access to default scenario
                    defaultScenario = Scenario.objects.get(pk=1)
                    defaultScenario.player_list.add(new_player)
                    defaultScenario.save()
                    return HttpResponseRedirect("/CopingGame/")
            else:
                return render(request, 'registration/register.html',
{'form': form,})
        else:
            form = UserCreationForm()
            return render(request, "registration/register.html",
{'form': form,})

#Player profile
@login_required(login_url='/login')
def profile(request):
    player = Player.objects.get(username=request.user)

```

```

        purchase_list = Purchase.objects.filter(player=player,
owned=True)
        context = {'player':player, 'purchase_list':purchase_list}
        if request.method == 'POST':
            if 'smallFont' in request.POST:
                player.text_size = 5
                player.save()
            elif 'mediumFont' in request.POST:
                player.text_size = 6
                player.save()
            elif 'largeFont' in request.POST:
                player.text_size = 7
                player.save()
            elif 'upload' in request.POST:
                profileForm = PlayerProfileForm(request.POST,
request.FILES)
                if profileForm.is_valid():
                    try:
                        picture=request.FILES['avatarPic']
                        player.avatarPic=picture
                        player.save()
                        return redirect('CopingGame/profile.html')
                    except Exception as e:
                        return redirect('CopingGame/profile.html')
                elif 'theme' in request.POST:
                    theme =
Store.objects.get(itemName=request.POST.get("theme","")) #Get the
specific theme
                    player.fav_bg = theme.bg #Set background color to the
themes background color
                    player.fav_text = theme.text #Set the text color to
the themes text color
                    player.save()
                elif 'picture' in request.POST:
                    player.avatarPic = request.POST.get("picture", "")
#Change user picture to the selected one
                    player.save()
                return render(request, "CopingGame/profile.html", context)
            else:
                profileForm = PlayerProfileForm()
                context = {'player':player, 'purchase_list':purchase_list,
'profileForm':profileForm}
                return render(request, "CopingGame/profile.html", context)

```

[Document last updated on 05/06/2015]