

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
Specifications:	Page 7 - 9
Modeling:	Pages 10 - 12
Changes in Requirements:	Page 14
Domain Analysis:	Page 15
Risk Analysis Management:	Page 16
Project Planning:	Page 17
Cost Analysis:	Page 18
Schedule for Development and Deployment:	Page 19 - 21
Software Quality and Assurance:	Page 22
Work Breakdown Structure:	Page 23
Software Design:	Page 24
Architectural Design:	Page 25
Interface Design:	Page 26
Detailed Design:	Page 27 - 30
Implementation Plan:	Page 31
Testing:	Page 32
Delivery:	Page 33
Maintenance Plan:	Page 34
Source Code:	Page 35 - 58

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, Javascript, Python, Django, and MySQL. HTML, CSS, and Javascript were used to design and format the web pages. 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 MySQL 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.

Requirements:

Coping Game Requirements

- I. Provide a web game designed to help players learn coping techniques for everyday life situations
- II. Players will be able to make their own account if they don't already have one
 - A) Players can choose their username
 - B) An email will be assigned to their account
- III. 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 coping techniques
 - D) Answering each question will reward the player with tokens they can use in the game store
- IV. 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
 - 2) User pictures will change the picture associated with the players account
- V. Players progress will be saved
 - A) Their token amount
 - B) What theme and user picture they are using
 - C) What themes and user pictures they have unlocked
- VI. 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
- VII. Administrators can play the game as if they are a regular player

Specifications:

The game will be implemented using MySQL, Django, HTML, and CSS. The MySQL 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 JavaScript 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:

***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).

SQLite:

pros: Easy start-up and great testing.

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

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 setup, 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:

First deployment of the database will be on our local computers until a server is 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:

Pluvali Game Database ER Diagram

Members:
Alex Davis
Steve Kosovich
Tim Leikam
Greg Martini

Assumptions: USER is an admin, a PLAYER is registered using Django auth. System, and will get derived attributes UserName, UserID, and password.

Each PLAYER can have as many PURCHASES as he/she would like.

PURCHASES requires a PLAYER

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

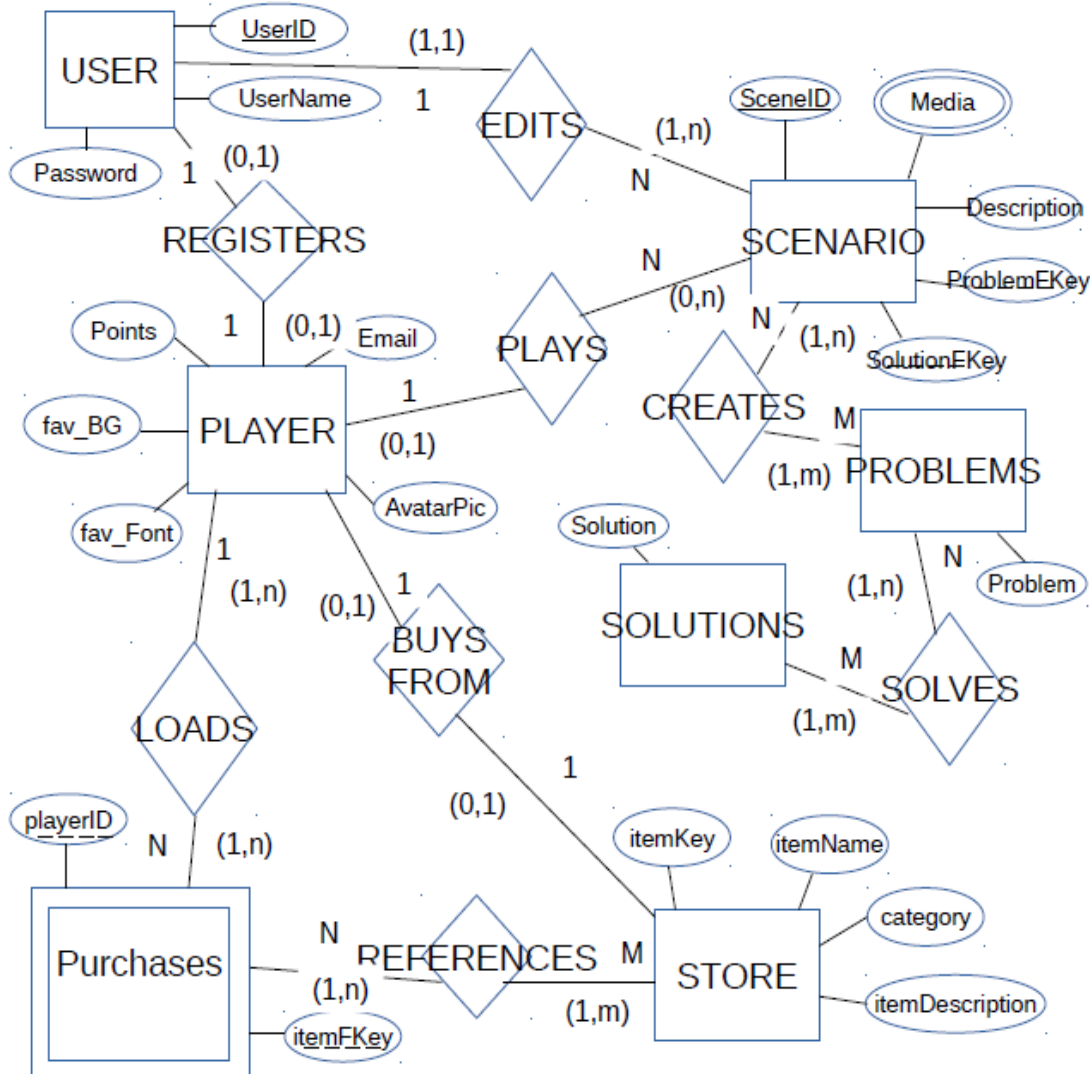
A PLAYER can play available SCENARIOS, but only one at a time (some SCENARIOS are PLAYER specific).

Media in SCENARIOS contains videos or pictures.

Purchases may include UI overlays that don't effect game play.

USER AvatarPics can be uploaded to database.

Every PLAYER gets a default_view PURCHASE from the STORE.



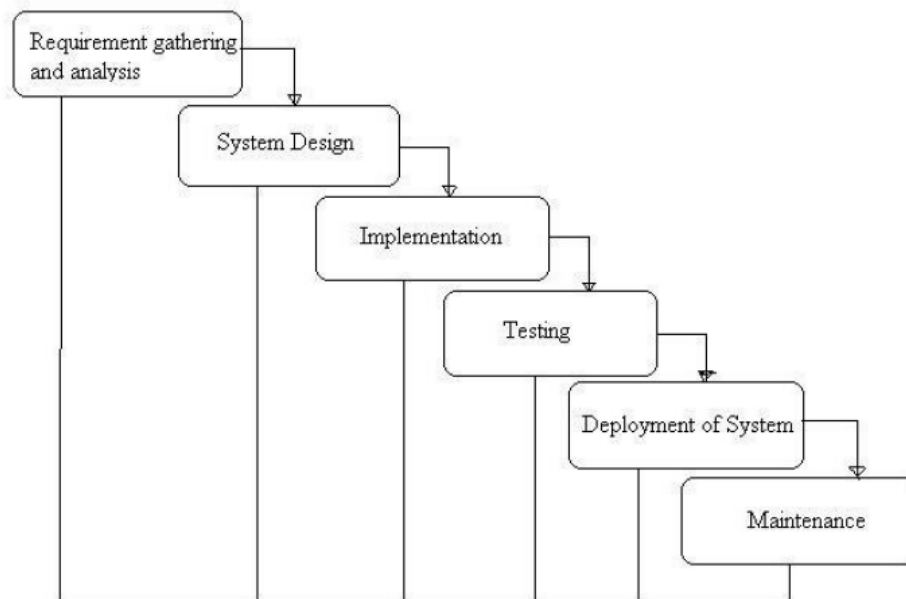
Software Design Model

Pluvali Development Model

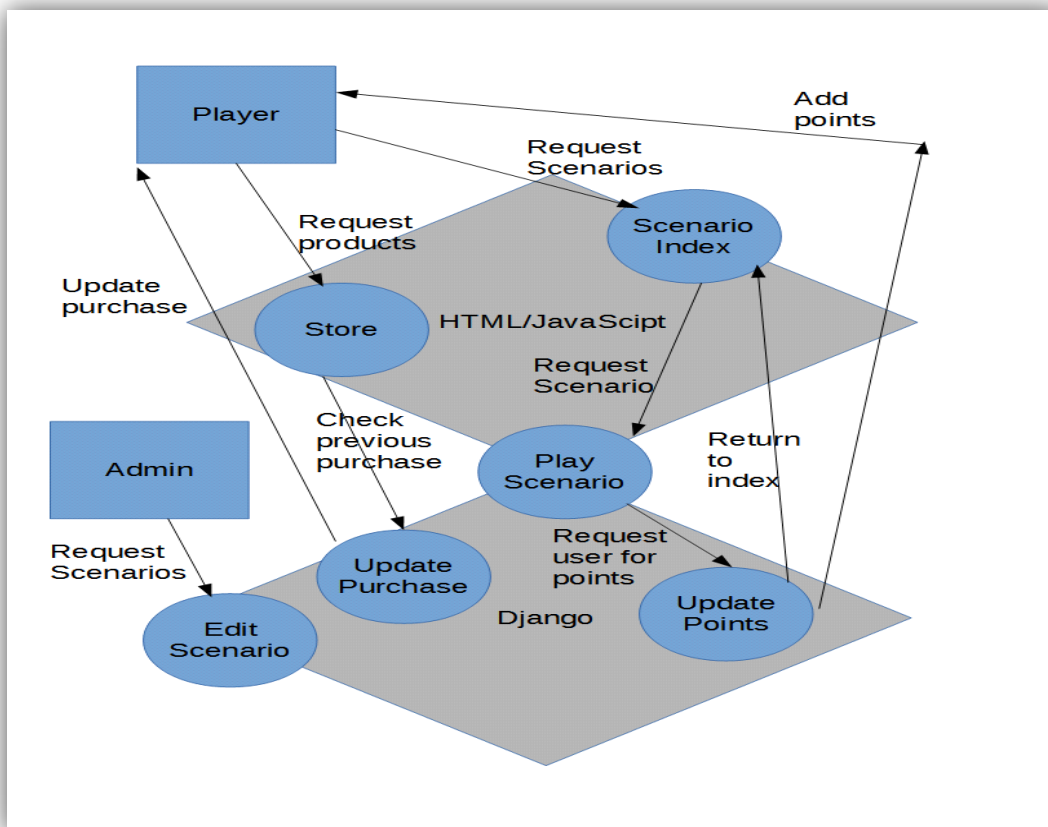
Alex Davis
Steven Kosovich
Greg Martini
Tim Leikam
Debra Parcheta
Katie Taliercio

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.

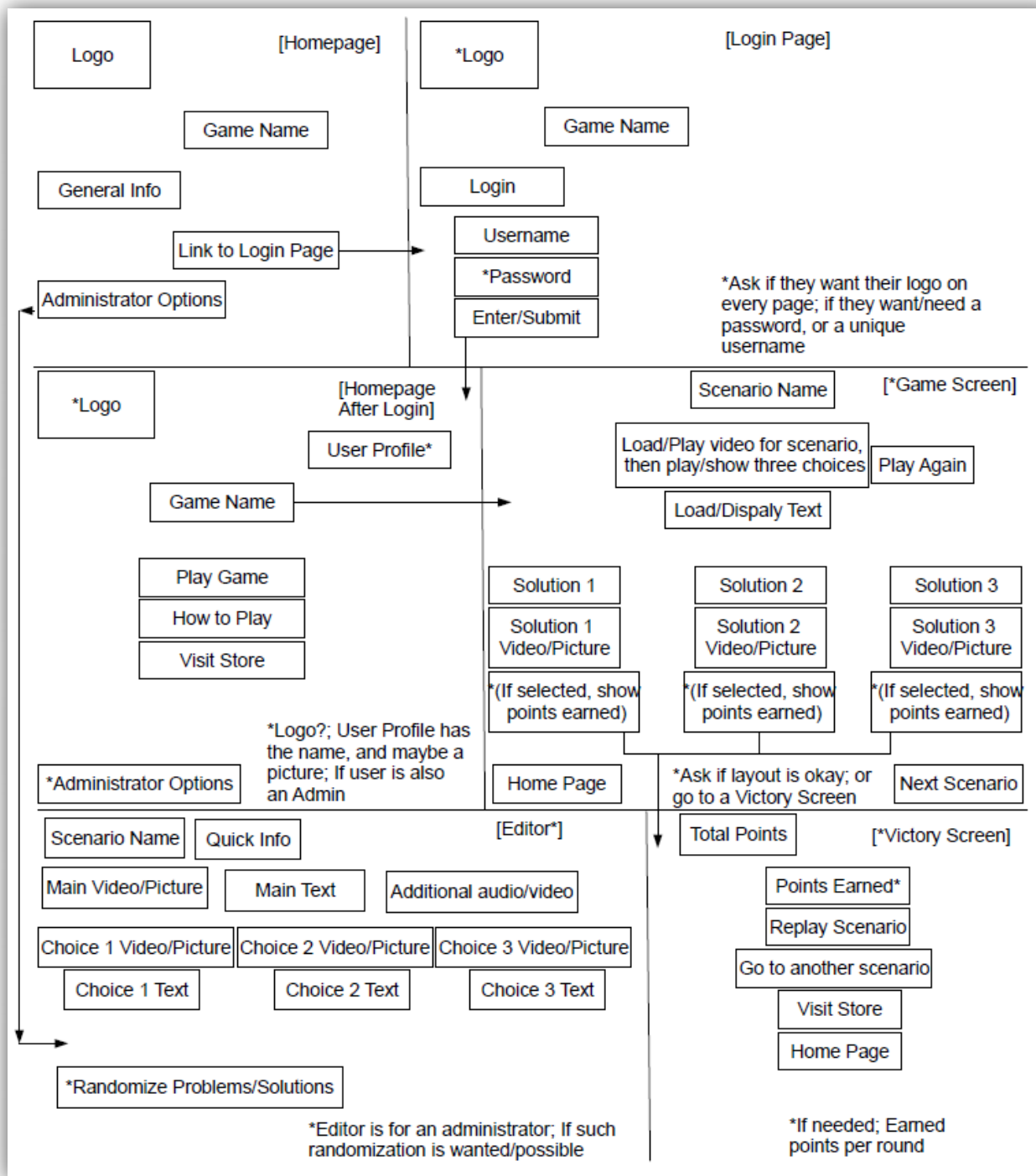
General Overview of "Waterfall Model"



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 much easier to implement certain features by using JavaScript than to try and use just HTML and CSS.

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 12 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 7 -13 for the Requirements, Specifications, and models.

See pages 19 - 21 for the detailed timeline of the project.

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.

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

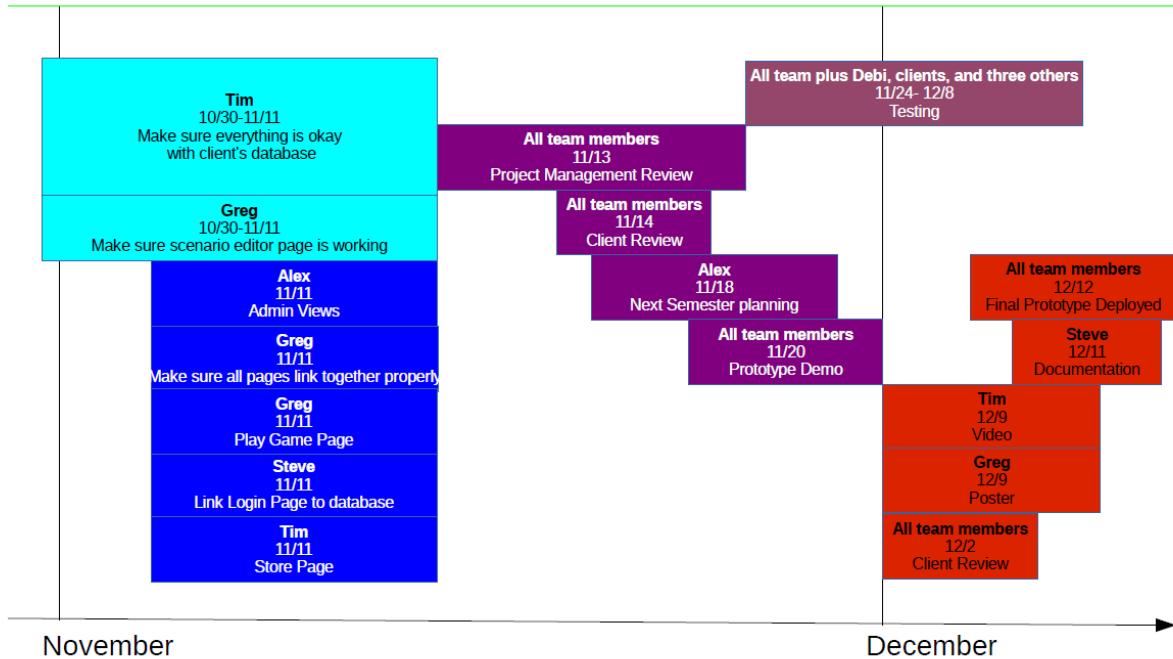
Presentation to be presented by all members

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:

Pluvali Timeline

Members:
Alex Davis
Steve Kosovich
Tim Leikam
Greg Martini



November

December

Last revision: December 2, 2014

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 resources 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 9/30 - 10/20. 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.

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 9/09 - 9/29, and attended all of the team meetings. He spent on average 8 hours per week working on the project.

Tim - Worked mostly on coding HTML, CSS, and JavaScript 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 10/21 - 11/10. He spent an average of 8 hours a week working on the project.

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 11/11 to 12/12. 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.

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 11 of this document.

We also used many common tools and languages to implement our design, such as HTML, CSS, JavaScript, and Django. The reasons we chose these tools are explained in more detail on pages 7 - 9 of this document.

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 10 - 12 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, CSS, and JavaScript. The HTML was used for the majority of the code and is the main part of the project. CSS and JavaScript are used in conjunction with HTML, to add visual enhancements and improvements.

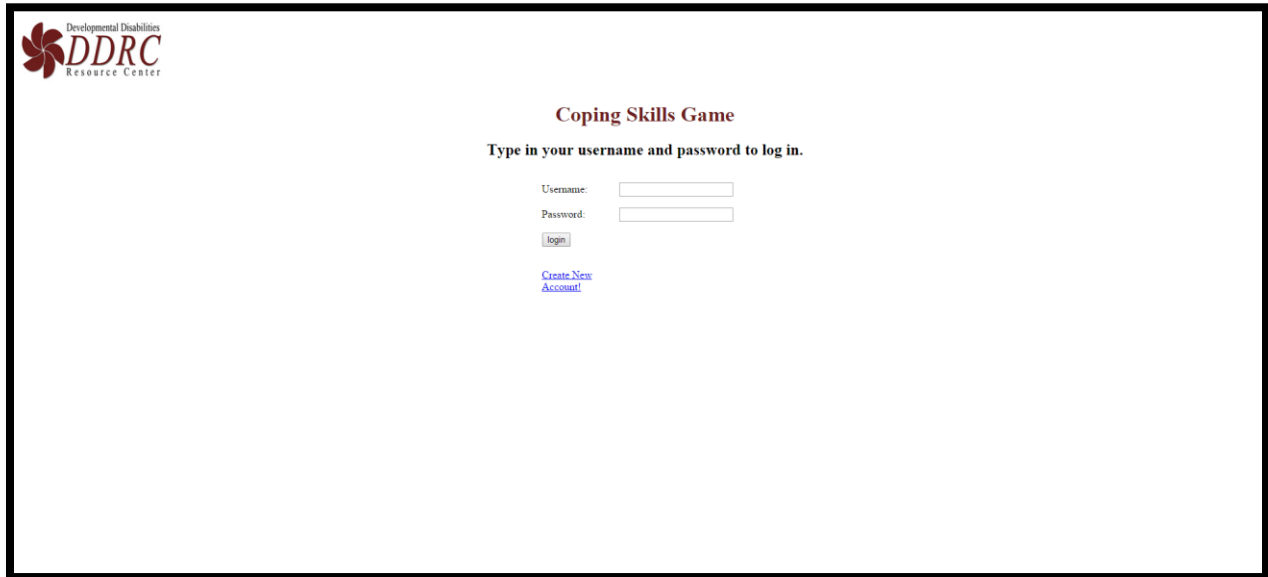
The interface was designed to be intuitive and easy to use. For users, there is a sign in page, then a homepage, a help page, a scenario page, a store page, and the game pages. When a game is completed, there is a victory page as well. So the interface is very simple with very few pages, so that almost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

See page 13 of this document for an example of how the pages are laid out and connect to each other.

Detailed Design:

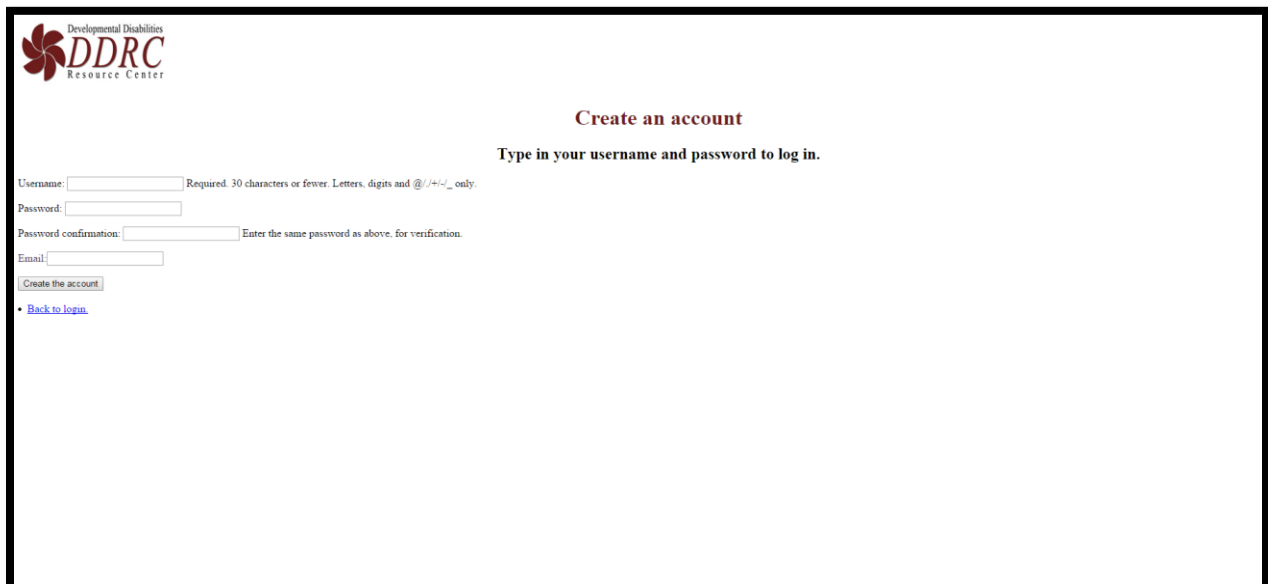
See pages 4 - 13 for the Requirements, Specifications, and Design Diagrams. The screenshots of the game's design are shown below:

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 consists of a red flower-like 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. There are two input fields: "Username:" and "Password:". Below the password field is a small "login" button. At the bottom, there 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. Below it is the instruction "Type in your username and password to log in." in a standard black font. The form includes several input fields with associated labels and instructions: "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:". At the bottom left is a "Create the account" button. Below the button is a blue link that says "• Back to login."

Home page





- [steve](#)
- [Tokens: 13](#)
- [Logout](#)

Welcome to the Coping Skills Game!


General Information:


This game is a fun way to learn how to cope with difficult situations encountered in everyday life.

- [Play A Scenario.](#)
- [Visit Store.](#)
- [Help Page.](#)

You are now able to play the game and go to the store!

Help page






- [steve](#)
- [Tokens: 13](#)
- [Logout](#)

How to play


- Once logged in, click the "Play a Scenario" to choose from the available scenarios.
- Select the scenario you wish to play and then choose a solution to each problem.
- Solving each problem rewards a token, which you may use to redeem prizes!
- From the homepage, you may also visit the store.
- Once inside, select from a category and then chose the item you want to buy it!

- [Back to the homepage.](#)

Store Page



Developmental Disabilities
DDRC
Resource Center



• steve
• Tokens: 13
• Logout

Back to Index Page

Welcome to the Store

CATEGORIES

Themes

User Pictures

Click on the "Purchase" button to buy the item to its left.

Themes - 50 Tokens Each

PREVIEW

Purchase

PREVIEW

Purchase

PREVIEW

Purchase


PREVIEW

Purchase


PREVIEW

Purchase

Scenario list page



Developmental Disabilities
DDRC
Resource Center




• steve
• Tokens: 13
• Logout


Please choose the scenario you want to play!

- Television Problems

- Back to the homepage.

Game page






greg

Tokens: 0

Logout


Scenario - Television Problems

Problem - TV won't turn on.




How To Handle This Problem?

Walk




Select

Music




Select


Read



Select

Victory page





steve

Tokens: 15

Logout

CONGRATULATIONS!

You completed the scenario and solved all the problems!

You've earned a total of 15 points.

- [Back to scenario index.](#)
- [Back to the homepage.](#)

Implementation Plan:

Initial:

Initially, we planned on having our product ready to be implemented by November 18. For more information on what our schedule looked like initially, see pages 19 - 20.

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 21 for the most recent version of our timeline, to see when we have things planned.

We hope to implement the final finished version of the product by the end of April 2015.

Testing:

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

Are you able to:

- Create a user account
- Log out successfully (located in the top right corner of the screen)
- Log in again using your accounts' credentials
- 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
- 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 right corner of your screen above the logout button)
- Verify that you are directed to the victory page when you complete a scenario
- From the scenario page, make sure you can navigate back to the scenario page and to the home page
- 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 you can switch what theme you're using by clicking on the purchase button of the theme you wish to use
- Verify that your theme stays the same after you log out and log back in using your accounts credentials
- Verify that your theme works on every page (the victory page has had issues with themes)

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 either fixed or will be fixed in the future.

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 many issues, it is 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 an Internet connection.

User Manual:

- Go to <http://coping.ddrcweb.com>
- Create a user profile, or sign in as an existing one
- Navigate to the help page, store page, or scenario page as you like
- Play scenarios and earn points to spend on aesthetic upgrades for your profile
- Enjoy!

Training:

No official training is necessarily needed to play the game, but training may be necessary for maintaining and updating the database and user accounts. This will be done on a as-needed basis by a member of the team.

Maintenance Plan:

This product will be maintained by the staff of DDRC, and Team Pluvali while we are working on 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 with the Django, CSS, and JavaScript imbedded in it, just as it is in our product:

Code for login.html, the code that lets you log in to play the game:

```
<!DOCTYPE html>

<!-- A prototype 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" cellspacing="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>

```

Code for register.html, the page that lets you make your profile:

```

<!DOCTYPE html>

<!-- A prototype 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;">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>

```

Code for index.html, the home page that lets you go to the store, the help page, or play scenarios:

```

<!DOCTYPE html>

<!-- Another prototype 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 style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text }}">

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

    <!-- Table that has the user's profile when logged in -->

    <table id="usertable" align="right" style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text }}">
        <tr>
            <!-- user picture -->
            <td rowspan="2" colspan="2"></td>
            <td>
                {% if user.is_authenticated %}
                <!-- user information -->
                <li>{{ user.get_username }}</li>

```

```

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

<br>

<!-- The Header -->
<h1>Welcome to the Coping Skills Game!</h1>
<h2>

</h2>

<!-- Here is some general information about the game -->
<br><br><br>

<!--<p style ="text-align:center;color:#306565">General Information: <br>This game is a fun
way to learn how to cope with difficult situations encountered in everyday life.</p> -->
    <p 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>

    <br>
    <li><a href="/CopingGame/scenarios/" style="color:{ { player.fav_text } }">Play A
Scenario.</a></li>
    <li><a href="/CopingGame/store/" style="color:{ { player.fav_text } }">Visit Store.</a></li>
    <li><a href="/CopingGame/help/" style="color:{ { player.fav_text } }">Help Page.</a></li>

    <!--<p style ="text-align:center;color:#4c3365;font-size:200%">You are now able to play the
game and go to the store!</p> -->
    <p style ="text-align:center;color:{ { player.fav_text } };font-size:200%">You are now able to
play the game and go to the store!</p>

</body>

</html>

```

Code for help_page.html, the page that tells you how to play the game:

```

<!DOCTYPE html>

<!-- Quick page by Greg -->
{ % load staticfiles % }

```

```

<html>
<link rel="stylesheet" type="text/css" href="{% static 'CopingGame/style_home_page.css' %}" />
<body style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text }}">

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

    <!-- Table that will have the user profile once logged in -->
    <table id="usertable" align="right" style="background-color:{{ player.fav_bg }}; color:{{
player.fav_text }}">
        <tr>
            <td rowspan="2" colspan="2"></td>
            <td>
                {% if user.is_authenticated %}
                <!-- user information -->
                <li>{{ user.get_username }}</li>
                <li>Tokens: {{ player.points }}</li>
                <li><a href="/logout/" style="color:{{ player.fav_text }}">Logout</li>
                {% else %}
                Not currently logged in, please log in.
                {% endif %}</td>
        </tr>
        <tr>
            <td></td>
        </tr>
    </table>
<br>

    <!-- The Header -->
    <h1>How to play</h1>
    <br>
    <li>Once logged in, click the "Play a Scenario" to choose from the available scenarios.</li>
    <li>Select the scenario you wish to play and then chose a solution to each problem.</li>
    <li>Solving each problem rewards a token, which you may use to redeem prizes!</li>
    <li>From the homepage, you may also visit the store.</li>
    <li>Once inside, select from a category and then chose the item you want to buy it!</li>
    <br>
    <li><a href="/CopingGame/" style="color:{{ player.fav_text }}">Back to the
homepage.</a></li>
</body>
</html>

```

Code for store_page.html, the page that lets you buy improvements for your profile based on the amount of points you have from playing the game:


```

<!-- A prototype store page - Tim -->

{% load staticfiles %}

<html>
<head>

    <!-- Title the page and link the style sheet and javascript -->
    <title>Store Page</title>
    <link rel="stylesheet" type="text/css" href="{% static 'CopingGame/style_store_page.css' %}">
    <script type="text/javascript" src="{% static 'CopingGame/store_page.js' %}"></script>

</head>

<body style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text }}">
    <!-- A link to ddrcc's website in a new tab by using their logo -->
    </a>

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

    <br><br>
    <FORM align="center">
        <INPUT Type="BUTTON" VALUE="Back to Index Page" style="height:5%;
width:10%" ONCLICK="window.location.href='/CopingGame/'">
    </FORM>
    <br>

    <h1>Welcome to the Store</h1>

    <!-- Category button's on the left -->

```

```

<div id="categories">
    <h3>CATEGORIES</h3>
    <ul id="catul">
        <li id="catli"><button class="catbutton" id="theme" onclick="chkbtn(this,
themes)" onmouseover="overbtn(this)" onmouseout="outbtn(this)">Themes</button></li>
        <li id="catli"><button class="catbutton" id="up" onclick="chkbtn(this,
userpics)" onmouseover="overbtn(this)" onmouseout="outbtn(this)">User Pictures</button></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 "Purchase" button to buy
the item to its left.</h2>

        <!-- Themes table -->
        <table class="cattable" id="themes" align="center">

            <tr>
                <td colspan="2" style="text-decoration:underline">Themes - 50
Tokens Each</td>
            </tr>

            <tr>
                <td bgcolor="blue" style="color:white">PREVIEW</td>
                <!--<td><button type="button">Purchase</button></td> -->
                <form action="" method="post" >
                    { % csrf_token % }
                    <td><button type="submit" value="bluewhite"
name="bluewhite">Purchase</button></td>
                </form>
            </tr>

            <tr>
                <td bgcolor="firebrick" style="color:cornsilk">PREVIEW</td>
                <!--<td><button type="button">Purchase</button></td> -->
                <form action="" method="post">
                    { % csrf_token % }
                    <td><button type="submit" value="firebrickcornsilk"
name="firebrickcornsilk">Purchase</button></td>
                </form>
            </tr>

            <tr>
                <td bgcolor="yellow" style="color:purple">PREVIEW</td>
                <!--<td><button type="button">Purchase</button></td> -->
                <form action="" method="post">
                    { % csrf_token % }
                    <td><button type="submit" value="yellowpurple"
name="yellowpurple">Purchase</button></td>

```

```

        </form>
    </tr>

    <tr>
        <td bgcolor="black" style="color:orange">PREVIEW</td>
        <!--<td><button type="button">Purchase</button></td> -->
        <form action="" method="post">
            { % csrf_token % }
            <td><button type="submit" value="blackorange"
name="blackorange">Purchase</button></td>
        </form>
    </tr>

    <!-- PROTOTYPE VALUE ----- Will be removed after testing -->
    <tr>
        <td bgcolor="#ededed" style="color:black">PREVIEW</td>
        <!--<td><button type="button">Purchase</button></td> -->
        <form action="" method="post" >
            { % csrf_token % }
            <td><button type="submit" value="defaultblack"
name="defaultblack">Purchase</button></td>
        </form>
    </tr>
</table>

<!-- User Pictures table -->
<table class="cattable" id="userpics" align="center">

    <tr>
        <td colspan="2" style="text-decoration:underline">User Pictures
- 50 Tokens Each</td>
    </tr>

    <tr>
        <td></td>
        <!--<td><button type="button">Purchase</button></td> -->
        <form action="" method="post">
            { % csrf_token % }
            <td><button type="submit">Purchase</button></td>
        </form>
    </tr>

    <tr>
        <td></td>
        <!--<td><button type="button">Purchase</button></td> -->
        <form action="" method="post">
            { % csrf_token % }

```

```
 <button type="submit">Purchase</button></td> </form> </tr>  <tr>  </td> <!--<td><button type="button">Purchase</button></td> --> <form action="" method="post">     { % csrf_token % }     <td><button type="submit">Purchase</button></td> </form> </tr> </table>  </div> </div>  </body> </html> | |
```

Code for scenario_index.html, the page that lets you choose what kinds of scenarios to play:

```

<!DOCTYPE html>

<!-- Another prototype home page - Steve -->
{ % load staticfiles % }
<html>
<link rel="stylesheet" type="text/css" href="{ % static 'CopingGame/style_home_page.css' % }" />
<body style="background-color:{ { player.fav_bg } }; color:{ { player.fav_text } }">

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

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

    <table id="usertable" align="right" style="background-color:{ { player.fav_bg } }; color:{ {
player.fav_text } }">
        <tr>
            <td rowspan="2" colspan="2"></td>
            <td>
                { % if user.is_authenticated % }
                <!-- user information -->
                <li>{ { user.get_username } }</li>
                <li>Tokens: { { player.points } }</li>
            </td>
        </tr>
    </table>

```

```

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

<br>

<!-- The Header -->
<h1>Please choose the scenario you want to play!</h1>

{% if scenario_list % }
    <ul>
        {% for scenario in scenario_list % }
            <li><a href="/CopingGame/{ { scenario.sceneID } }/" style="color:{ {
player.fav_text } }">{ { scenario.title } }</a></li>
        {% endfor % }
    </ul>
    {% else % }
        <p>No scenarios are available.</p>
    {% endif % }
    <br>
    <br>
    <li><a href="/CopingGame/" style="color:{ { player.fav_text } }">Back to the
homepage.</a></li>

</body>

</html>

```

Code for game_page.html, the page that lets you play the game:

```

<!DOCTYPE html>

<!-- A prototype game page - Tim -->

{% 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' % }"

/>

```

```

</head>

<body style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text }}">

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

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

    <br>

    <!-- What is the current scenario and problem -->
    <h1>Scenario - {{ scene.title }}</h1>
    <h2>Problem - {% if player.stage == 0 %}{{ stage1 }}
        {% elif player.stage == 1 %}{{ stage2 }}
        {% elif player.stage == 2 %}{{ stage3 }}
        {% elif player.stage == 3 %}{{ stage4 }}
        {% elif player.stage == 4 %}{{ stage5 }}
        {% endif %}</h2>

    <!-- Image representing the problem -->
    <!-- WILL NEED FOR FINAL PRODUCT, needs pictures in the DB first
    {% if player.stage == 0 %}
        
    {% elif player.stage == 1 %}
        
    {% elif player.stage == 2 %}
        
    {% elif player.stage == 3 %}
        
    {% elif player.stage == 4 %}
        

```

```

    {% endif % }
-->
<!-- temp fix for prototype -->
    {% if player.stage == 0 % }
        <center></center>
    {% elif player.stage == 1 % }
        <center></center>
    {% endif % }
<br>

<!-- 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">
    {% if player.stage == 0 % }
        {% for sol in stage1.solutions.all % }
            <td>{{ sol.solution }}</td>
        {% endfor % }
    {% elif player.stage == 1 % } {{ stage2 }}
        {% for sol in stage2.solutions.all % }
            <td>{{ sol.solution }}</td>
        {% endfor % }
    {% elif player.stage == 2 % } {{ stage3 }}
        {% for sol in stage3.solutions.all % }
            <td>{{ sol.solution }}</td>
        {% endfor % }
    {% elif player.stage == 3 % } {{ stage4 }}
        {% for sol in stage4.solutions.all % }
            <td>{{ sol.solution }}</td>
        {% endfor % }
    {% elif player.stage == 4 % } {{ stage5 }}
        {% for sol in stage5.solutions.all % }
            <td>{{ sol.solution }}</td>
        {% endfor % }
    {% endif % }

    </tr>

    <tr>
<!-- WILL NEED FOR FINAL PRODUCT, needs pictures in the DB first
    {% if player.stage == 0 % }

```

```

                {% for sol in stage1.solutions.all %}
                    <td></td>
                {% endfor %}
            {% elif player.stage == 1 %}
                {% for sol in stage2.solutions.all %}
                    <td></td>
                {% endfor %}
            {% elif player.stage == 2 %}
                {% for sol in stage3.solutions.all %}
                    <td></td>
                {% endfor %}
            {% elif player.stage == 3 %}
                {% for sol in stage4.solutions.all %}
                    <td></td>
                {% endfor %}
            {% elif player.stage == 4 %}
                {% for sol in stage5.solutions.all %}
                    <td></td>
                {% endfor %}
            {% endif %}
-->    {% if player.stage == 0 %}
        <td></td>
        <td></td>
        <td></td>
        {% elif player.stage == 1 %}
            <td></td>
            <td></td>
            <td></td>
        {% endif %}
    </tr>

    <tr>
        <form action="" method="post">
            {% csrf_token %}
            {% if player.stage == 0 %}
                {% for s in stage1.solutions.all %}
                    <td><button type="submit">Select</button></td>
                {% endfor %}
            {% elif player.stage == 1 %}

```



```

        {% for s in stage2.solutions.all %}
            <td><button type="submit">Select</button></td>
        {% endfor %}
    {% elif player.stage == 2 %}
        {% for s in stage3.solutions.all %}
            <td><button type="submit">Select</button></td>
        {% endfor %}
    {% elif player.stage == 3 %}
        {% for s in stage4.solutions.all %}
            <td><button type="submit">Select</button></td>
        {% endfor %}
    {% elif player.stage == 4 %}
        {% for s in stage5.solutions.all %}
            <td><button type="submit">Select</button></td>
        {% endfor %}
    {% endif %}

</form>
</tr>
</table>

</body>
</html>

```

Code for victory_page.html, the page that shows you how many points you've earned by playing the scenario:

```

<!DOCTYPE html>

<!-- Quick page by Greg -->
{% load staticfiles %}
<html>
<link rel="stylesheet" type="text/css" href="{% static 'CopingGame/style_home_page.css' %}" />
<body style="background-color:{{ player.fav_bg }}; color:{{ player.fav_text }}">

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

    <!-- Table that will have the user profile once logged in -->
    <table id="usertable" align="right" style="background-color:{{ player.fav_bg }}; color:{{
player.fav_text }}">
        <tr>
            <td rowspan="2" colspan="2"></td>
            <td>

```

```

        {% if user.is_authenticated % }
            <!-- user information -->
            <li>{{ user.get_username }}</li>

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

<!-- The Header -->
<h1>CONGRATULATIONS!</h1>
<br>
<p>You completed the scenario and solved all the problems!</p>
<p>You've earned a total of {{ player.points }} points.</p>
<br>
<li><a href="/CopingGame/scenarios/" style="color:{{ player.fav_text }}">Back to scenario
index.</a></li>
    <li><a href="/CopingGame/" style="color:{{ player.fav_text }}">Back to the
homepage.</a></li>
</body>
</html>

```

[Document last updated on December 9, 2014]

Django Source Code:

#global_defs.py

```
TEXT_BLACK = 1
TEXT_BLUE = 2
TEXT_GREEN = 3
TEXT_YELLOW = 4
TEXT_RED = 5
TEXT_PURPLE = 6
TEXT_PINK = 7
TEXT_ORANGE = 8
TEXT_WHITE = 9
BG_BLACK = 10
BG_BLUE = 20
BG_GREEN = 30
BG_YELLOW = 40
BG_RED = 50
BG_PURPLE = 60
BG_PINK = 70
BG_ORANGE = 80
BG_WHITE = 90
```

#pluvali/settings.py

```
""" Django settings for pluvali project.    For more information on this file,
see https://docs.djangoproject.com/en/1.7/topics/settings/    For the full list of settings and their values,
see https://docs.djangoproject.com/en/1.7/ref/settings/ """    # Build paths inside the project like this:
os.path.join(BASE_DIR, ...)    import os    BASE_DIR =
os.path.dirname(os.path.dirname(__file__))    # Quick-start development settings - unsuitable for
production    # See https://docs.djangoproject.com/en/1.7/howto/deployment/checklist/    # SECURITY
WARNING: keep the secret key used in production secret!    SECRET_KEY = 'b--
==6)u3m3(rj_br^b&k3m(2tsfd+j!+mzrvb1l4fx%uodu&!'    # SECURITY WARNING: don't run with
debug turned on in production!    DEBUG = True    TEMPLATE_DEBUG =
True    ALLOWED_HOSTS = []    # Application definition    INSTALLED_APPS = (
'django.contrib.admin',    'django.contrib.auth',    'django.contrib.contenttypes',
'django.contrib.sessions',    'django.contrib.messages',    'django.contrib.staticfiles',
    'CopingGame', )    MIDDLEWARE_CLASSES = (
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',    'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware', )    ROOT_URLCONF =
'pluvali.urls'    WSGI_APPLICATION = 'pluvali.wsgi.application'    # Database    #
```

```

https://docs.djangoproject.com/en/1.7/ref/settings/#databases    DATABASES = {    'default': {
'ENGINE': 'django.db.backends.sqlite3',    'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
} }    TEMPLATE_DIRS = (    "pluval/html/templates/default", )    # Internationalization #
https://docs.djangoproject.com/en/1.7/topics/i18n/    LANGUAGE_CODE = 'en-us'    TIME_ZONE =
'America/Denver'    USE_I18N = True    USE_L10N = True    USE_TZ = True    # Static files
(CSS, JavaScript, Images) # https://docs.djangoproject.com/en/1.7/howto/static-files/    STATIC_URL
= '/static/'    MEDIA_URL = '/media/'    LOGIN_URL = 'index'    LOGOUT_URL =
'logout'    LOGIN_REDIRECT_URL = 'index'

```

```
#pluval/html/urls/.py
```

```

from django.conf.urls import include, url
from django.contrib import admin
import CopingGame

```

```
from django.views.generic.base import RedirectView
```

```

urlpatterns = [
    url(r'^CopingGame/', include('CopingGame.urls')),
    url(r'^admin/', include(admin.site.urls)),
        url(r'^login/$', 'django.contrib.auth.views.login'),
        url(r'^logout/$', 'django.contrib.auth.views.logout', {'next_page': '/login/'}),
        url(r'^register/$', CopingGame.views.register, name='register'),
        url(r'^$', RedirectView.as_view(url='CopingGame/', permanent=False), name='index'),
]

```

```
#CopingGame/admin.py
```

```

from django.contrib import admin
from CopingGame.models import *

```

```

# Register your models here.
admin.site.register(Player)
admin.site.register(Solutions)
admin.site.register(Problems)
admin.site.register(Scenario)
admin.site.register(Store)
admin.site.register(Purchases)

```

```
#CopingGame/forms.py
```

```
from django import forms
```

```
from django.core import validators
from CopingGame.models import Problems
```

```
class UploadProblemPicForm(forms.ModelForm):
    class Meta:
        model = Problems
        fields = ('pictureP',)
```

```
#CopingGame/models.py
import os
from django.db import models
from django.contrib.auth.models import User
import global_defs as defs

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

```
def get_image_path(instance, filename):
    return os.path.join('users', str(instance.id),filename)
```

```
class Player(models.Model):
    user = models.OneToOneField(User)
    email = models.CharField(max_length=30)
    points = models.IntegerField(default=0)
    #avatarPic = models.ImageField(upload_to=get_image_path, blank=True, null=True)
    fav_bg = models.CharField(max_length=30, default='#ededed')
    fav_text = models.CharField(max_length=30, default='black')
    stage = models.IntegerField(default=0) #used for iterating through scenarios
    def passw(self):
        return self.user.password
    def __str__(self):
        return self.user.username
    pass
```

```
class Solutions(models.Model):
    pictureS = models.ImageField(upload_to='/media/solutions/', blank=True, null=True)
    solution = models.CharField(max_length=300, default="Solution")
    def __str__(self):
        return self.solution
    pass
```

```
def upload_path_handler(instance, filename):
```

```

        return filename

class Problems(models.Model):
    pictureP = models.ImageField(upload_to=upload_path_handler, blank=True, null=True)
    problem = models.CharField(max_length=225, default="The Problem.")
    solutions = models.ManyToManyField(Solutions)
    def __str__(self):
        return self.problem

#     def save(self, *args, **kwargs):
#         self.pk = self.pk
#         problem_id = self.problem_id
#         problem_id_str = str(problem_id)
#
#         new_file = generate_image_name_hash()
#         new_file_main = new_file + '.jpg'
#         new_file_root_path = settings.PROBLEMS_UPLOAD_PATH + '/' + problem_id_str + '/'
# new_file_main
#
#         if self.pictureP:
#             self.pictureP.name = settings.PROBLEMS_UPLOAD_PATH + '/' +
# problem_id_str + '/' + new_file_main
#
#         super(Problems, self).save(*args, **kwargs)
##
#
#     if self.pictureP and self.pk:
#         old_img = Picture.objects.get(pk=self.pk)
#         old_img_instance = old_img.pictureP
#
#
#     if self.pictureP:
#         if self.pk:
#             pictureP = old_img_instance
#         else:
#             pictureP = self.pictureP
#
#
#     super(Problems, self).save(*args, **kwargs)
#
#     #if in form
#     if self.pictureP:
#         tmp_file = Image.open(self.pictureP.path)
#         if tmp_file.mode != 'RGB':
#             tmp_file = tmp_file.convert('RGB')
#
#         image = Image.open(self.pictureP.path)

```

```

#             if pictureP.mode != 'RGB':
#                 pictureP = pictureP.convert('RGB')
#                 pictureP.save(new_file_root_path, 'JPEG', quality=100)
pass

class Scenario(models.Model):
    sceneID = models.AutoField(primary_key=True)
    title = models.CharField(max_length=15, default="Scenario Title")
    description = models.CharField(max_length=250, default="Scenatio Decription")
    problems = models.ManyToManyField(Problems)
    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")
    def __str__(self):
        return self.itemName

class Purchases(models.Model):
    player = models.ForeignKey(Player)
    itemFKey = models.ForeignKey(Store)
    def __str__(self):
        return self.itemFKey.itemName

#CopingGame/urls.py
from django.conf.urls import url, patterns
from django.core.urlresolvers import reverse
from CopingGame 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/', views.store, name='store'),
    url(r'^admin_page/', views.admin_page, name='admin_page/'),
    url(r'^(?P<problem_id>\d+)/upload/$', views.upload_problem_pic, name='upload_problem_pic'),
]

```

```

#CopingGame/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 import forms
from django.contrib.auth.forms import UserCreationForm
from CopingGame.forms import UploadProblemPicForm
import random

from CopingGame.models import Player, Scenario, Problems, Solutions, Store
from django.contrib.auth.models import User

#login page uses default Django sessions

#link for homepage, requires login
@login_required(login_url='/login')
def index(request):
    player = Player.objects.get(user=User.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(user=User.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(user=User.objects.get(username=request.user))
    scenario_list = Scenario.objects.order_by('title').distinct()
    player.stage = 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('user').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(user=User.objects.get(username=request.user))
    scene = get_object_or_404(Scenario, pk = sceneID)
    max_stage = scene.problems.count()
    stage1 = scene.problems.all()[0]
    if(max_stage >= 2):
        stage2 = scene.problems.all()[1]
    if(max_stage >= 3):
        stage3 = scene.problems.all()[2]
    if(max_stage >= 4):
        stage4 = scene.problems.all()[3]
    if(max_stage >= 5):
        stage5 = scene.problems.all()[4]
    if(player.stage > max_stage):
        player.stage = 0
    if(max_stage >= 1):
        context = {'scene':scene, 'player':player, 'stage1':stage1 }
    if(max_stage >= 2):
        context = {'scene':scene, 'player':player, 'stage1':stage1,'stage2':stage2}
    if(max_stage >= 3):
        context = {'scene':scene, 'player':player, 'stage1':stage1,'stage2':stage2,'stage3':stage3}
    if(max_stage >= 4):
        context = {'scene':scene, 'player':player,
'stage1':stage1,'stage2':stage2,'stage3':stage3,'stage4':stage4}
    if(max_stage >= 5):
        context = {'scene':scene, 'player':player,
'stage1':stage1,'stage2':stage2,'stage3':stage3,'stage4':stage4,'stage5':stage5}
    if request.method == 'POST':
        player.points += random.randint(2,5)
        player.stage += 1
        player.save()
        if(player.stage == max_stage):
            return render(request, 'CopingGame/victory_page.html', {'player':player})

    return render(request, 'CopingGame/game_page.html', context)

```

```

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

@login_required(login_url='/login')
def store(request):
    player = Player.objects.get(user=User.objects.get(username=request.user))
    items_list = Store.objects.order_by('itemName')
    context = {'items_list':items_list, 'player':player}
    if request.method == 'POST':
        #if player.points >= 50: uncomment next two lines for final version
        #player.points -= 50
        if 'bluewhite' in request.POST: #for testing that themes can at least be switched
            and saved
            player.fav_bg = 'blue' #will be implemented in user defaults
            page next semester
            player.fav_text = 'white'
            if 'firebrickcornsilk' in request.POST:
                player.fav_bg = 'firebrick'
                player.fav_text = 'cornsilk'
            if 'yellowpurple' in request.POST:
                player.fav_bg = 'yellow'
                player.fav_text = 'purple'
            if 'blackorange' in request.POST:
                player.fav_bg = 'black'
                player.fav_text = 'orange'
            if 'defaultblack' in request.POST:
                player.fav_bg = '#ededed'
                player.fav_text = 'black'
            player.save()
    return render(request, 'CopingGame/store_page.html', context)

#User registration
def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        player_email = request.POST['email']
        if form.is_valid():
            new_user = form.save()
            new_player =

```

```

Player.objects.create(user=User.objects.get(username=request.POST['username']), email=player_email)
                    new_user = authenticate(username=request.POST['username'],
password=request.POST['password1'])
                    return HttpResponseRedirect("/CopingGame/")
                else:
                    return render(request, 'registration/register.html', {'form': form,})
            else:
                form = UserCreationForm()
                return render(request, "registration/register.html", {'form': form,})

def handle_uploaded_file(file):
#     filename = file.name
#     path = settings.MEDIA_ROOT
    with open('media/name.jpg', 'wb+') as destination:
        for chunk in file.chunks():
            destination.write(chunk)
    destination.close()

def upload_problem_pic(request, problem_id):
    problem = get_object_or_404(Problems, pk=problem_id)
    if request.method == 'POST':
        form = UploadProblemPicForm(request.POST, request.FILES)
        if form.is_valid():
            handle_uploaded_file(request.FILES['file'])
            problem.pictureP.save(request.FILES['file'].name, content, save=True)
            #instance = Problems.pictureP(file_field=request.FILES['file'])
            #instance = Problems.problem(file_field=request.POST['problem'])

            form.save()
            return HttpResponseRedirect('/index/')
        else:
            return render(request, 'CopingGame/upload_pic.html', {'form': form,})
    else:
        form = UploadProblemPicForm()
        return render(request, 'CopingGame/upload_pic.html', {'problem':problem, 'form':form})

```