Team Pluvali
Alex Davis
Steve Kosovich
Tim Leikam
Greg Martini

Debra Parcheta
Katie Taliercio

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.

<u>Linked with IP Address:</u>

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

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

<u>Resources:</u>

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:

<u>*Notepad++:</u>

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).

<u>Sublime Text Editor:</u>

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.

<u>Eclipse:</u>

pros: Free and powerful IDE.

cons: More powerful than what we need.

Version Control Considerations:

<u>*GitHub:</u>

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

cons: No free private repositories.

<u>Concurrent Version System:</u>

pros: Free and compatible across most platforms.

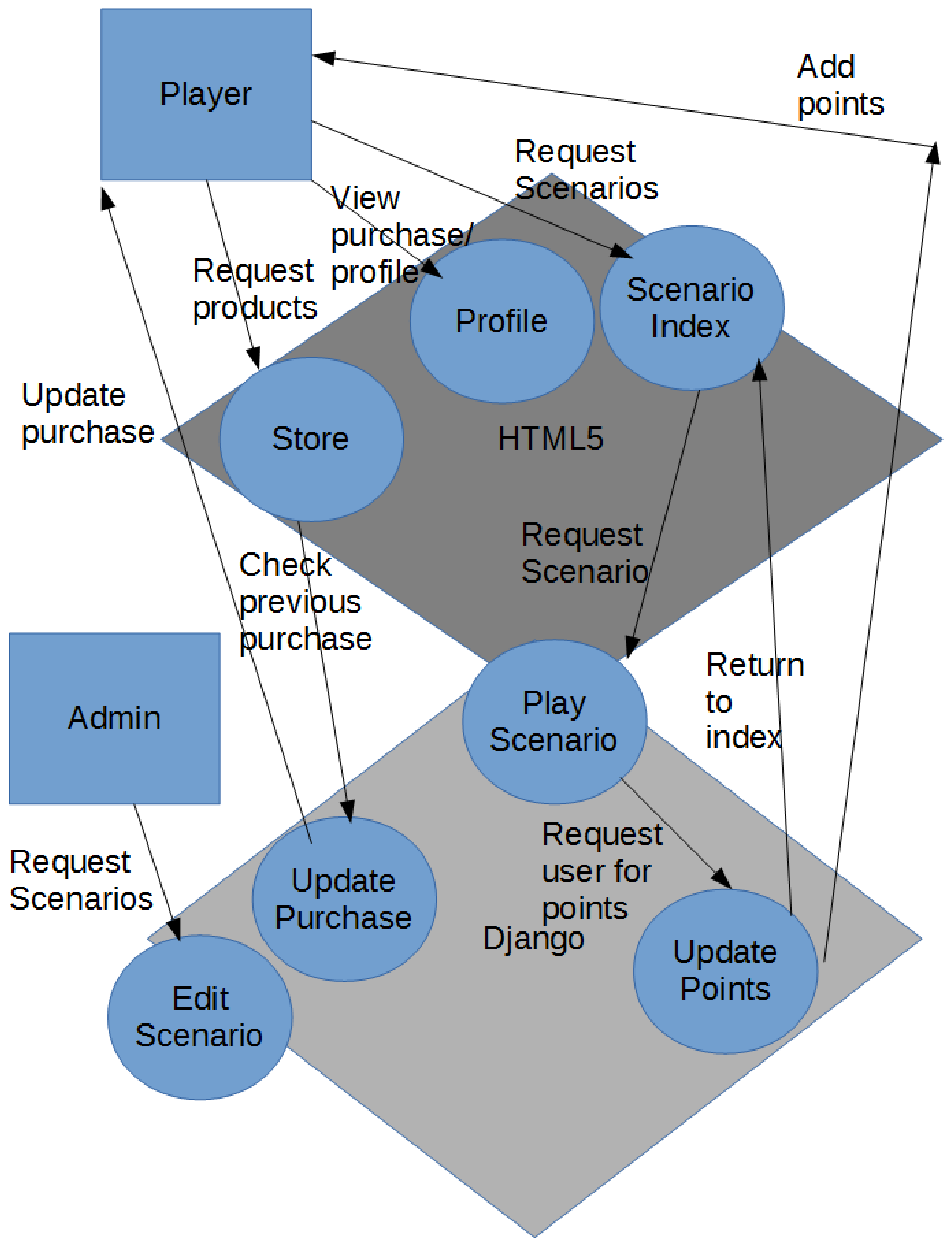
cons: One of the biggest design flaws with the current release of CVS is that it is very difficult to move a file to a different directory or rename it.

<u>BitBucket:</u>

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).

Data Flow Diagram:


Player
Add points
Request Scenarios
View purchase/ profile
Request products
Profile
Scenario Index
Update purchase
Store
HTML5
Request Scenario
Check previous purchase
Admin
Play Scenario
Return to index
Request Scenarios
Update Purchase
Request user for points
Django
Update Points
Edit Scenario