Indiana University Southeast – CSCI Department

CS Capstone I – Software Detailed Design

Fall 2016

Software Detailed Design

Joshua Donahoe

Ryan Silver

Stephen Rumpel

Table Of Contents:

[Data Design](#_362z39i0quau)

[Interface Design](#_wnxgttfaciws)

[Figure 1.2 “Home page”](#_wppwwvbbyluh)

[Figure 1.3 “About Page”](#_a6qpf8k0eoti)

[Figure 1.4 “Tutorials Page”](#_57cqjeh6v3a0)

[Figure 1.5 “Account Creation”](#_ed39hj5gh1e9)

[Figure 1.6 “Login”](#_d35hoq3xy58n)

[Figure 1.7 “Accounts Page” -Student](#_ezw64jtn9kx4)

[Figure 1.8 “Accounts Page” -Teacher](#_4scxrj93z8ef)

[Figure 1.9 “Stats Page”](#_a5r5buw6k6le)

[Figure 1.10 “Games Page”](#_nycp55tp36qz)

[Figure 1.11 “Game Page”](#_mojrgx6ww3nc)

[Procedural Design:](#_ivoz424b01in)

[Pages:](#_9z1xxpbxemyz)

[Home](#_lz13tnjb21jc)

[About](#_1im75ncuz18t)

[Accounts](#_gkzuvx67uu3)

[Tutorial](#_n2x8ycz9a5n9)

[Stats](#_j7blzpa6a1dm)

[Procedures](#_f2t93zghw3d6)

[Account creation:](#_mpqwdurzjylv)

[Logging in:](#_eml7gyvz1tdj)

[Viewing old games:](#_wv6xcqjfyomn)

[Creating a new game:](#_dsfpis688tqu)

[Breakdown of individual contributions](#_iq3pxrozpf4e)

[Key Personnel information](#_7tyoyp3l63mx)

Reference: https://en.wikipedia.org/wiki/Software\_design\_description

The SDD should contain the following information:

1. The [*data design*](https://en.wikipedia.org/wiki/Data-driven_design) describes structures that reside within the software. Attributes and relationships between [data objects](https://en.wikipedia.org/wiki/Data_object) dictate the choice of [data structures](https://en.wikipedia.org/wiki/Data_structures).

# Data Design

We will be using sql to store the bulk of our information storage needs. the database is necessary to facilitate accounts and the storage of games. without it many useful features will not be available or will be based on cookies(that get deleted by our users and disappear forever).

Input to our database tables will become an insert statement following account creation. the website will have to tailor, prune and encrypt information for this insert statement.

output will also have to be pruned. and will come in the form of select statements. these statements will have to follow the syntax of SQL whatever specific flavor we choose in the end. (MYsql?)

**account database schema:**

CREATE TABLE users(userNumber INTEGER Primary key AUTOINCREMENT, username text unique NOT NULL , password text NOT NULL, eMail text, isTeacher boolean, teachercode int unique, isStudent boolean, schoolCode int, joined text, verified boolean)

(Meets syntax of Sqlite3, not tested or finalized)

**UserNumber**- Int field that is used to reference a particular user

**Username**- Text field where duplicates are not allowed and the username “” is also not allowed.

**Password**- Text field that we plan on encrypting the password “” is also not allowed we may add additional restraints on this field to prevent users from using passwords that are short or weak.

**Email**- Text non required field.

**IsTeacher**- a boolean field that is not required and can identify a user as a teacher if used.

**TeacherCode**- an integer field that can be used to bind teachers to students in an encapsulated way most likely through tailored select statements like “select userNumber from users where teachercode=123 inner join (select UserNumber from users where isStudent=true)”this combines the selections in such a way that only students with the teacherCode 123 will show up. this in a unique field so duplicates will throw an error. (these select statements are not final. Nor are they tested.)

**IsStudent**- boolean field, makes select statements easy if you want to find students.

**School code**- int school code use this list for school codes <https://ifap.ed.gov/ifap/fedSchoolCodeList.jsp>

**Joined**- a date in the form of text that can be used for administration

**verified**- boolean value a verification email will set this to true.

**Games database schema:**

CREATE TABLE games(gameNum INTEGER PRIMARY KEY AUTOINCREMENT, turns int, winner boolean, turnHistory text, userID INTEGER)

(Meets syntax of Sqlite3, not tested or finalized)

**GameNum**- int field used to reference a particular game

**turns**- how many turns until the conclusion of the game.

**winner**- boolean true if userID was the winner of the game

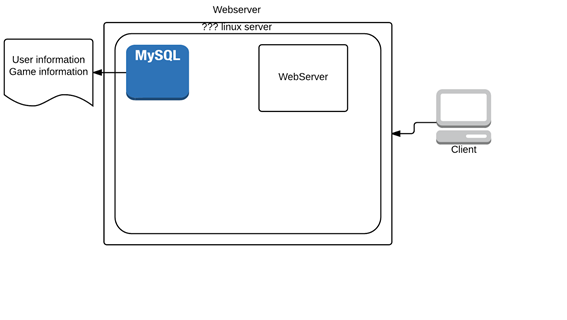
**turnHistory-** a raw link that can be used to get a full history of turns in the game. can also be a blob.

**userId**- reference to the user’s ID for cross reference.

In turn history we will be using a link to a CSV file that we can parse to replay a game or just to view the game as a list of actions performed by the player and the computer.

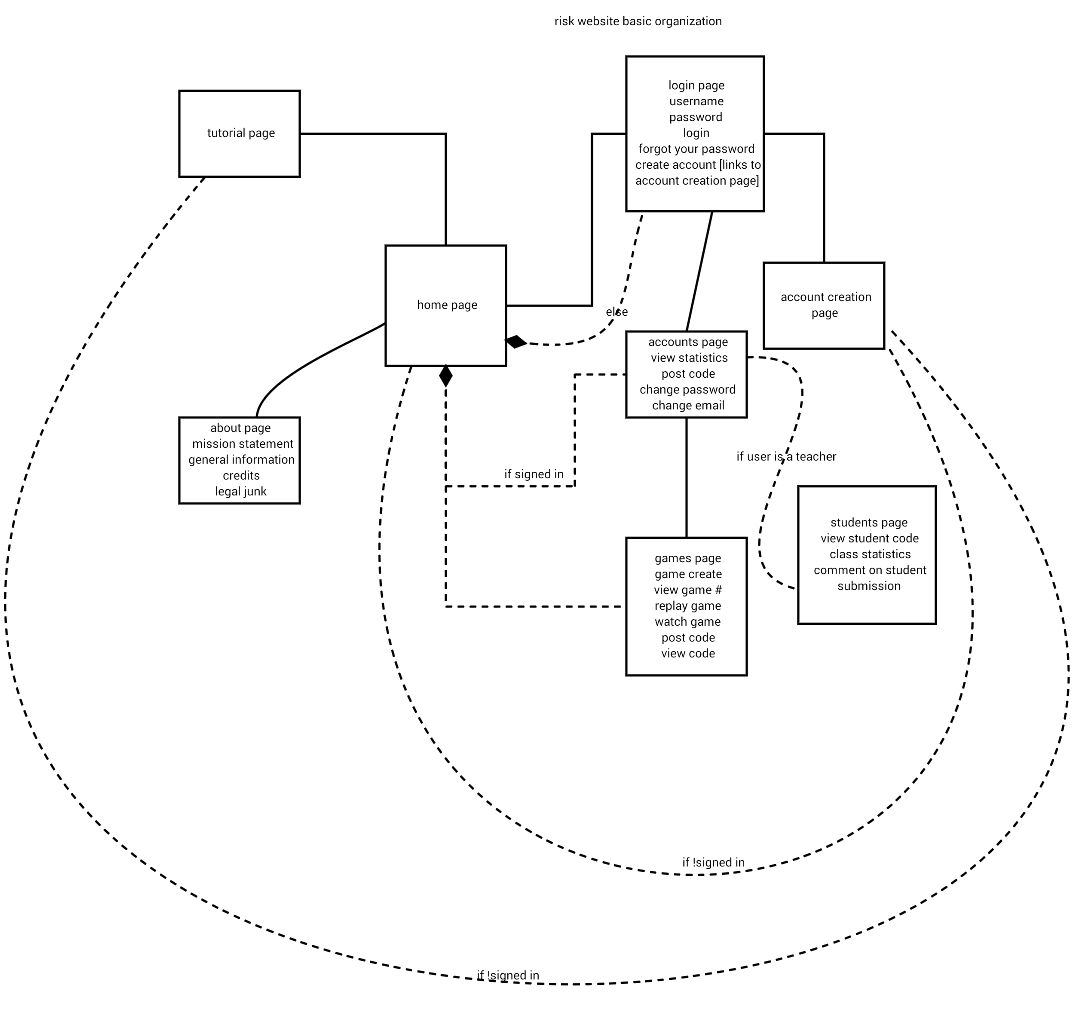
Certain aspects of the actual Risk program use separate structures, however, these will not affect the browser portion of this project.

1. The [*architecture design*](https://en.wikipedia.org/wiki/Software_architecture) uses information flowing characteristics, and maps them into the program structure. The transformation mapping method is applied to exhibit distinct boundaries between incoming and outgoing data. The data flow diagrams allocate control input, processing and output along three separate modules.

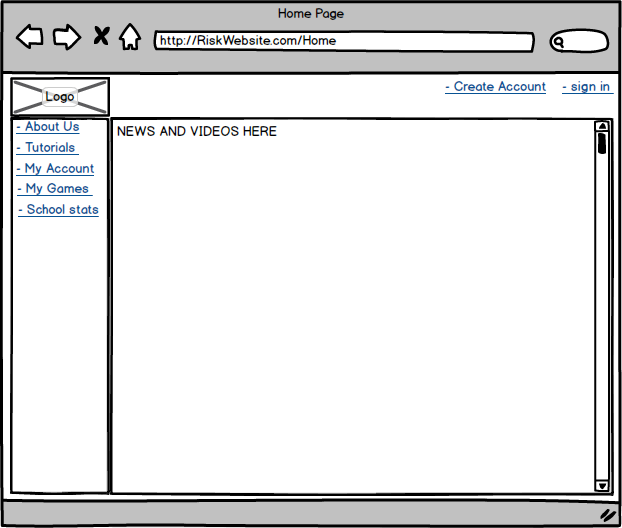


# Interface Design

In figure 1.1 the basic “flow” of the website is described visually. Pages within figure 1.1 will be further elaborated upon below in figures 1.2-1.11.

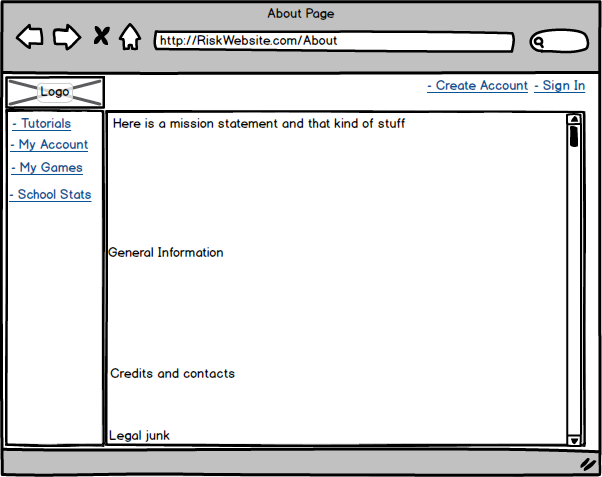


**Figure 1.1 “Flow”**



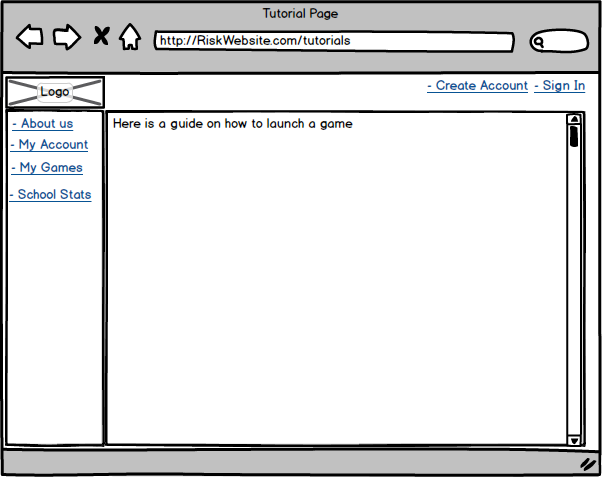
## Figure 1.2 “Home page”

The home page allows for ample space to present our product and show featured links and other information that the admin will be able to edit and add to, if we have time we can even do a RSS feed so this process is automated. In all Pages the image “Logo” will serve as a link back to the home page.



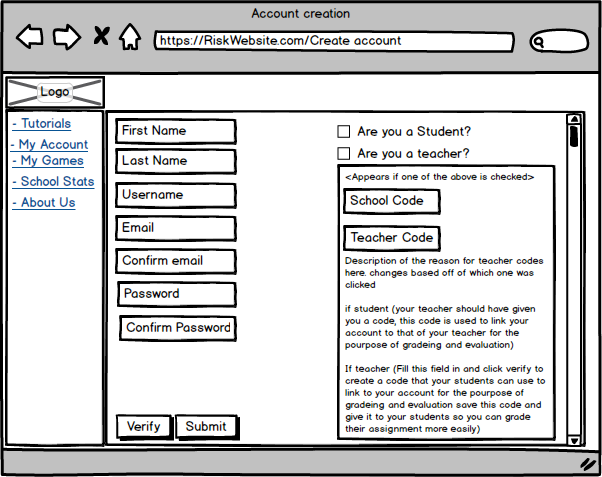
## Figure 1.3 “About Page”

The about page is for the authors to take credit for their work and to display general information and legal information about the website. Information like licenses and agreements will all be displayed at the footer (as a link to this page) and here in detail.



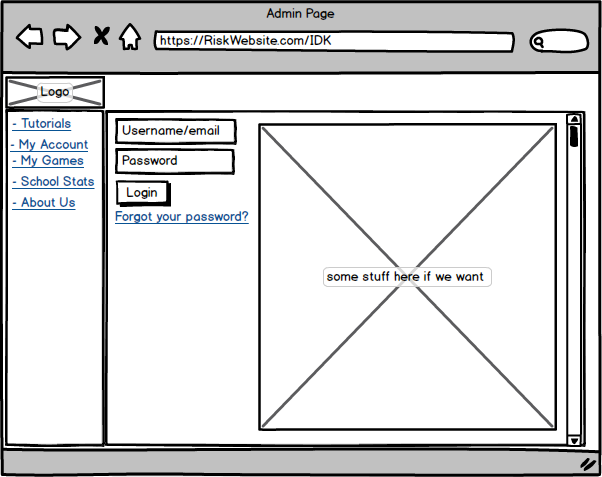
## Figure 1.4 “Tutorials Page”

This page will be something of a manual for users to make use of in their programming efforts. it will also include links to research papers on AI development that may be of use and a guide on how to launch a game. might even include the code for random here.



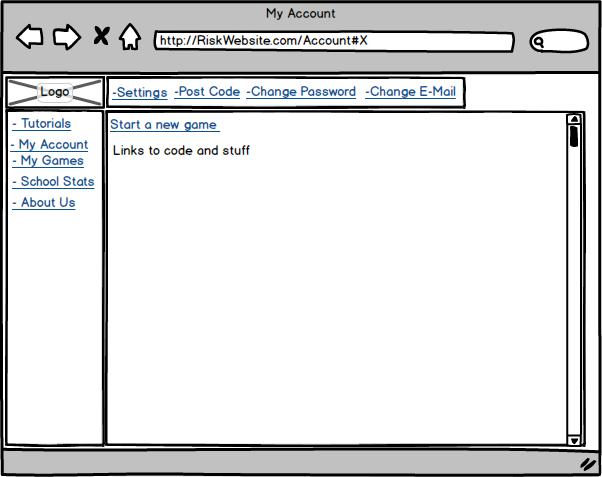
## Figure 1.5 “Account Creation”

On this page one can fill in the form and hit Submit to create an account. verify checks the email, password,school code, teacher code, and the confirm fields to make sure they are 1, available and 2, match their respective confirm field. if teacher is clicked after student. the check in student will disappear, and change teacher to checked and vice versa.



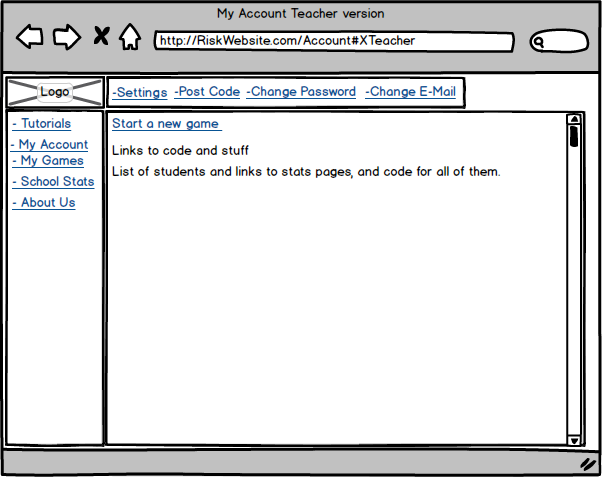
## Figure 1.6 “Login”

On this page a user can type their credentials and hit the login button to log into their account. clicking “forgot your password?” takes them to a similar page. except the password field is not there and the login button says “submit” instead. (Note admin page at the top is a mistake: it should read “Login”)



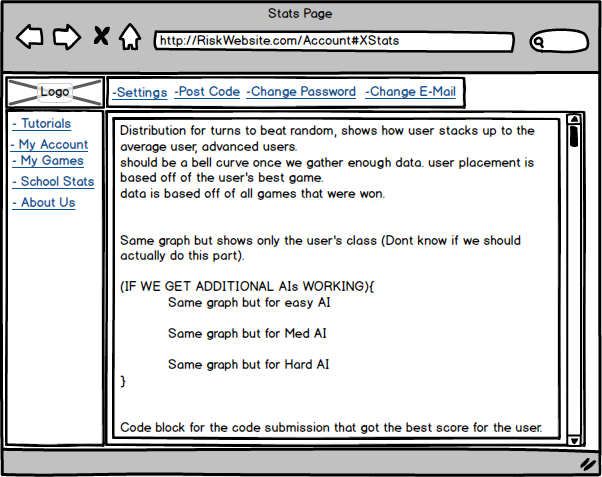
## Figure 1.7 “Accounts Page” -Student

There are two main account types, Teacher, and Student. a user that signs up without checking either box is treated as a student for all intents and purposes. This page is the account page for a student.



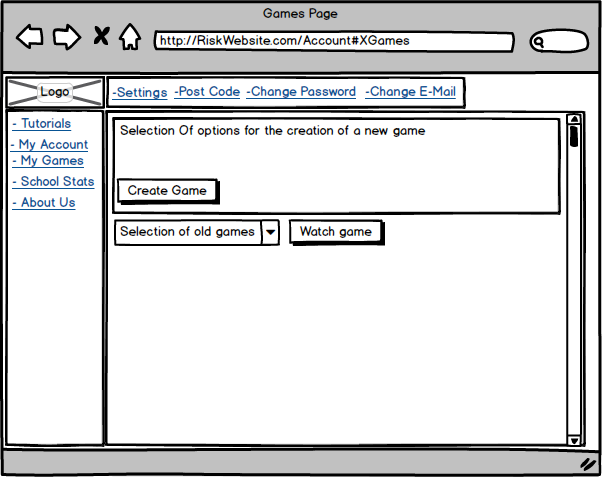
## Figure 1.8 “Accounts Page” -Teacher

In this version of the accounts page the user is a teacher. Links to stat pages are displayed for every student with the same teacher code as the teacher.



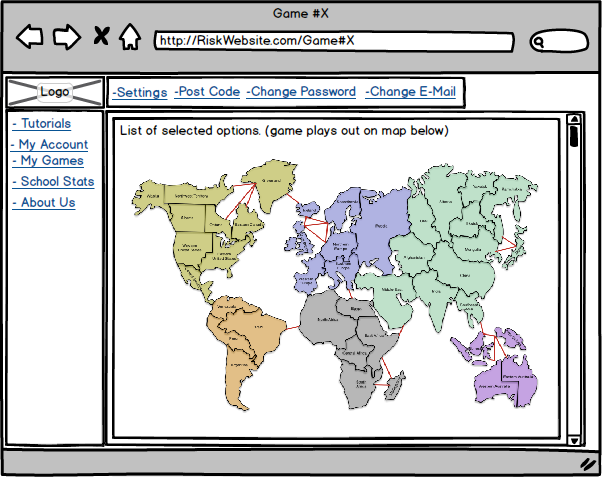
## **Figure 1.9 “Stats Page”**

On this page the user will be able to view statistics described above in the picture.



## Figure 1.10 “Games Page”

in the drop down menu users will be able to select a game and watch the game play out as it did when the game first took place. replays are based off of a CSV file that is fetched from a database (may not have time to implement)



## Figure 1.11 “Game Page”

This is where the magic takes place, a game of risk will play out in front of you (possibly on this fancy map I found).

1. The [*procedural design*](https://en.wikipedia.org/wiki/Procedural_design) describes structured programming concepts using graphical, tabular and textual notations. These design mediums enable the designer to represent procedural detail, that facilitates translation to code. This blueprint for implementation forms the basis for all subsequent software engineering work.

# Procedural Design:

## Pages:

### **Home**

* Introduction/Purpose of this Component/Entity

This is the starting point for the website. It will be the first thing new users and visitors see.

* Input for this Component/Entity

From this page, users can reach the About, Login, Tutorial, and Games pages.

* Output for this Component/Entity

This page merely displays the Home screen and listens for user input to advance to another page.

* Component/Entity Process to Convert Input to Output

Each button is a link to another page. Once a button is clicked, the browser moves to the selected link, changing the page.

* Design constraints and performance requirements of this Component/Entity

The Page should be responsive and able to load quickly. It should be designed to be functional and readable on a wide variety of display resolutions and sizes.

* Process (pseudo-code algorithm)

<html>

<header>Home<header/>

<body>Link(About), Link(Login), Link(Tutorial), link(Games)<body/>

<html/>

### **About**

* Introduction/Purpose of this Component/Entity

This page informs people not familiar with the website what its intentions and purpose are.

* Input for this Component/Entity

From this page, users can reach the Home, Login, Tutorial, and Games pages.

* Output for this Component/Entity

This page merely displays the About text and listens for user input to advance to another page.

* Component/Entity Process to Convert Input to Output

Each button is a link to another page. Once a button is clicked, the browser moves to the selected link, changing the page.

* Design constraints and performance requirements of this Component/Entity

The Page should be responsive and able to load quickly. It should be designed to be functional and readable on a wide variety of display resolutions and sizes.

* Process (pseudo-code algorithm)

<html>

<header>About<header/>

<body>Link(Home), Link(Login), Link(Tutorial), link(Games)<body/>

<html/>

### **Accounts**

* Introduction/Purpose of this Component/Entity

A page that allows you to manage your account - with slightly different functionality depending upon if you’re a teacher or student.

* Input for this Component/Entity

From this page, users can navigate to the tutorials, games, school stats, and about us pages. The user can also upload code, and change account credentials with button clicks.

* Output for this Component/Entity

Uploaded code will be stored in the database if that button is clicked. It also displays text and listens for user input to advanced to a new page. User data will also be changed depending upon what buttons/text fields are clicked.

* Component/Entity Process to Convert Input to Output

* Design constraints and performance requirements of this Component/Entity

The Page should be responsive and able to load quickly. It should be designed to be functional and readable on a wide variety of display resolutions and sizes.

* Process (pseudo-code algorithm)

### **Tutorial**

* Introduction/Purpose of this Component/Entity
* Input for this Component/Entity
* Output for this Component/Entity
* Component/Entity Process to Convert Input to Output
* Design constraints and performance requirements of this Component/Entity
* Process (pseudo-code algorithm)

### **Stats**

* Introduction/Purpose of this Component/Entity
* Input for this Component/Entity
* Output for this Component/Entity
* Component/Entity Process to Convert Input to Output
* Design constraints and performance requirements of this Component/Entity
* Process (pseudo-code algorithm)

## Procedures

### Account creation:

Reason:

users will want to be able to securely create accounts to use the service with.

Proc:

User enters information into account creation form

each string is parsed for non string information like comments "tics" (',") and other characters that can mess with SQL queries if a character is found highlight field(s) that were rejected in red let the user re enter information.

If -user is a teacher check teacher code to make sure it is not already being used. If it is tell the user and highlight teachercode field in red

if user is a student make sure there is a teacher with that teacher code the student entered.

Send info over to server to be inserted

before inserting password apply a random salt and then use a cryptographic hashing algorithm to protect passwords

insert info into users table

### Logging in:

Reason:

users will want to be able to actually use the accounts they create.

Proc:

user enter information into login form

each string is parsed for non string information like comments "tics" (',") and other characters that can mess with SQL queries if a character is found highlight field(s) that were rejected in red let the user re enter information.

Send info over to server to be checked

verify user name and password. And get a user account #

send a login ack and give user a Cookie so he/she can stay connected until he/she logs out.

### Viewing old games:

Reason:

users will want to be able to replay games where they did particularly well and teachers may want to look at a game for grading reasons.

Proc:

get game’s CSV from database(CSV SCHEMA: Player, ACTION,VICTOR(victor will be a boolean it is the winner of the action as it happened in the actual game))

parse through Game CSV updating map to show actions of the players.

Add a short pause so the game does not end in a quarter second.

### Creating a new game:

reason:

Games will need to have a record to ensure other features of the website work properly.

Proc:

select highest game # in games DB add one to this number

create a new page with a reference to this game # send player to this page.

After game is complete fill out a SQL insert statement for the new game

# Breakdown of individual contributions

Joshua Donahoe - Team Lead, Web Contact. front-end web development, Manages Team website and progress.

Ryan Silver - Back-end Web service and security, builds product site and functionality.

Stephen Rumpel - Programmer, translates old code to modern languages to provide game system for product site.

# Key Personnel information

Joshua Donahoe - [jdonahoe@ius.edu](mailto:jdonahoe@ius.edu) Languages: Java, C++, C#, Scala, Assembly,

Ryan Silver - Email: [silverr@ius.edu](mailto:silverr@ius.edu) . Languages: java C++, C#, Assembly, F#, python, and R.

Stephen Rumpel - Email: [srumpel@ius.edu](mailto:srumpel@ius.edu). Languages: Java, C++, C#, Javascript, Assembly, Scala,

Each procedural component should include one section devoted to the detailed description of the component

* Introduction/Purpose of this Component/Entity
* Input for this Component/Entity
* Output for this Component/Entity
* Component/Entity Process to Convert Input to Output
* Design constraints and performance requirements of this Component/Entity
* Process (pseudo-code algorithm)

The product to be produced is a I expect a system specification.  
  
You should be able to pass the document off to another team of very junior programmers and they could build that system.  
  
All programs should be described in detail, list all major methods/routines and the English one sentence description of each method/routine. And all should be very computable (no magic occurs here).  
  
The database (or all data store) should be described  at the ER Diagram level.  
  
Products for hardware platform, database and language for all programs should be specified.

Also, this report should be posted to the team website.

Portions of previous assignments can be used within this document in appropriate sections.

Note: Portions of this document are from <https://en.wikipedia.org/wiki/Software_design_description> .

To do: