**BCIT**

**Comp 4952 HCI for Application Development**

**Technical Programming Option**

# **Option Head Mirela Gutica**

**Fall 2022**

Mark: \_\_\_\_\_\_\_\_ /100

Lab 2

This is a **small team project**. Work with your project partner. **No late assignments will be accepted**.

**Requirements**:

1. Read the Web Form example.

2. Answer the questions:

A. What are the options for managing the state in ASP.NET Web Forms? Give each a short description.

**Application state:** Stored within the memory of a server programmatically. A resourceful and performative technique.

**Control state:** a data cache necessary for controls to function correctly on a page.

**Session state:** a collection of key-value pairs stored as objects on the server, per-client.

**View state:** A dictionary object within a hidden field of the page object, containing info of the page.

**Hidden fields:** A non-encrypted/protected, tucked away, invisible control to save values on the client-side.

**Cookies:** Expirable, small tokens of data stored on the client’s machine from HTTP-based interactions between server and client.

**Query strings:** data stored in the url which is used to pass data between pages/web forms.

**Profile properties:** data stored from an individual user in a persistent format without need of a database.

B. Explain how session variables are implemented. Refer to correct implementation.

Session variables are kept in Session objects in key-value pair formats. They are created client-side and then stored on the server for every unique user. Storing a new session variable includes indexing the key name and either writing the new value or creating a new object if the key is not found.

Session[“index”] = value; //write to session variable

string Value = Session[“index”].ToString; //read from session variable

In the example web form, session variables can be found in the Default.aspx.cs file, on lines 64-69:

Setting/adding session variables (the first parameter is where the session variable will be stored and named, second parameter is the value from the client)

Master.AddSession(Global.\_FirstName, current\_name);

Master.AddSession(Global.\_Age, current\_age.ToString());

Master.AddSession(Global.\_Address, current\_address);

Master.AddSession(Global.\_Phone, curremt\_phone);

To read session variables from the server, examples can be found in Option.aspx.cs on lines 57-61:

CurrentUser.CurrentStudent.FirstName = Session[Global.\_FirstName].ToString();

CurrentUser.CurrentStudent.Age = Convert.ToInt32(Session[Global.\_Age].ToString());

CurrentUser.CurrentStudent.Address = Session[Global.\_Address].ToString();

CurrentUser.CurrentStudent.PhoneNumber = Session[Global.\_Phone].ToString();

C. Explain how cookies are implemented. Refer to correct implementation.

Cookie data is stored in a small file on the user’s machine AppData directory. In every HTTP interaction with the client and server, a header of info containing the request and response is sent. The server receives a GET request and parses the header for info regarding the request and any cookies of the client’s machine and stores it. An implementation would look something like this:

Master.AddCookie(string name, string value); //Adds key-value pair as a cookie on client’s machine

CurrentUser.CurrentStudent.FirstName = Request.Cookies[Global.\_FirstName].Value; //Gets cookie from client and sets them in the server (from example code)

In the example web form, adding cookies to the client can be found in the Default.aspx.cs file, on lines 56-59:

Master.AddCookie(Global.\_FirstName, current\_name);

Master.AddCookie(Global.\_Age, current\_age.ToString());

Master.AddCookie(Global.\_Address, current\_address);

Master.AddCookie(Global.\_Phone, curremt\_phone);

To read session variables from the server, examples can be found in Recommend.aspx.cs on lines 42-45:

CurrentUser.CurrentStudent.FirstName = Request.Cookies[Global.\_FirstName].Value;

CurrentUser.CurrentStudent.Age = Convert.ToInt32(Request.Cookies[Global.\_Age].Value);

CurrentUser.CurrentStudent.Address = Request.Cookies[Global.\_Address].Value;

CurrentUser.CurrentStudent.PhoneNumber = Request.Cookies[Global.\_Phone].Value;

D. Explain how application variables are implemented. Refer to correct implementation.

Application variables are handled in a Global.asax file. These files declare application level events and objects. High level application events would include implementations for an application starting/ending session starting/ending. Those application/session-level events are triggered by ASP.NET or HTTP modules. The defined objects in Global.asax have application/session-wide scope, and are applied to all resources in the web app. Examples of implementation (from example code):

private static int \_totalHits; //declare an object to be updated for all users on web app

private void Application\_Start(object sender, EventArgs e) { Code that runs on application startup }

2. Decide on a project. Follow the instructions contained at **Stage 1: Low-Fidelity Prototype.** Start working towards your goals. The outcome of this assignment will be:

2.1. A draft document named Low-Fidelity Prototype Draft containing:

a. A description of your project:

i. Description of the interface (what the users see and interact with)

**Home Page**

Users will land on a home page, where they will see a brief about section about the app and buttons to redirect to the other main pages (play, login, about/contact)

**Play Setup Page**

Play page: user is shown a start game page which has instructions on how to play, as well as 3 buttons choosing difficulties (easy: 1-3 keywords per prompt, medium: 4-6 4 keywords per prompt, expert: 7 keywords per prompt+). Once a difficulty is chosen, they are shown a screen with 4 images generated with the prompt with the number of keywords they chose in the difficulty. Then the page loads the play page elements.

**Play Page**

The page would have 1-4 images AI generated given the prompt in the middle of the screen, the number of keywords remaining in the prompt (text field) on the left side of the screen, an input field where users would enter the keywords below the images (with an enter button to the right for users to submit a guessed keyword), and the lifeline text saying how many tries they have left to the right of the pictures. Once they either guess all the keywords in the prompt or run out of lifelines, they are shown the results page for that image (the actual keywords in the prompt and how they compared to other users who guessed that image), as well as “play again (same difficulty)” and “new difficulty” buttons to play again

**Contact Us Page**

Simple page where users are given a brief description about ourselves and a contact email and forum with a text field for bugs/suggestions.

**User Page**

Page after user logs in. Here there would be a profile icon image of the user, the username, and overall stats of the games played by the user. At the top right of the screen is a big button to start a game. There are also sign out and delete account buttons on the page, with the delete account button having a dialog to confirm action.

**Login/Signup Page**

Entered by clicking on the respective buttons on the home page. Identical appearance wise with buttons at the top right redirecting between login/sign up, and on the create account page there is two password input fields to confirm password. Successfully creating an account will take the user back to the sign in page and signing in will take the user to the user page.

ii. Description your system/application as a list of requirements

Front-end: homepage, login page, signup page, users page, contact page, play game page(selecting difficulty and results)

Back-end: database containing prompts associated with 4 images and user information, 4 or more APIs to generate images based on a prompt, random prompt generator (could be API, could just use the database)

Technologies: Visual Studio, ASP.NET, ADO.NET, Mongo DB/Firebase, possibly Heroku/Azure to deploy the app

Feature Requirements:

* Play AI Art keyword guessing game
* Create & log into account
* Delete account
* View user overall stats

b. A draft justification of a user-centred design (based on the HCI heuristics and on Fitts’s law - discussed in class)

We plan to bring user-centred design choices to our application. Below the 8 Golden Rules are listed and the user-centred design choices that will adhere to those principles are described.

1. Strive for consistency:

* Pages will use consistent colour themes, button sizes and shapes, font will be similar and differentiated from user typed text
* Red text should be for alerts (incorrect password fields, guessing incorrect words, guessing the same words)
* Game page
  + 4 images displayed will be fitted to be the same size and resolution

1. Seek universal usability
   1. Font size slider
   2. Short descriptions, big font
   3. Visual effects to indicate actions (guessing incorrectly and losing a lifeline would show an “X”, and shake the text a little, guessing correctly would show a check mark)
2. Offer informative feedback
   1. Button offer feedback for: pressed, hover, inactive
   2. Game Page
      1. Prevent use of special characters, numbers, 2 char. minimum, whitespace
      2. Prevent use of prior guessed words
      3. (optional) validation of English word
   3. Login Page
      1. Invalid email/username or password alert
3. Design dialogs to yield closure
   1. Login Page
      1. Prompt to show username after login ex. Logged is as: Jon Doe
   2. Signup Page
      1. “Account is registered + verification email is sent” alert
   3. Game Page
      1. State game is finished after winning/losing
      2. Exiting game
4. Prevent errors:
   1. Game Page
      1. Exiting game has an alert & confirmation
      2. Initial GET of game setup shows difficulty option buttons as unpressed and the play button as greyed out. After clicking a difficulty option button, that button will appear as pressed and the play button becomes active. Prevents mistakenly playing a game before correct difficulty is selected
5. Permit easy reversal of actions
   1. Game Page
      1. User can go back and choose difficulties if they choose the wrong one
      2. If user want to not playing the game, they can click the X to exit, but it will pump an alert box to ask user are u sure u want to leave the game
   2. User Page
      1. When user accidentally clicked the deleted account button, it will pump out a confirmation alert box.
6. Keep users in control
   1. Tab key functionality shortcut to navigate between buttons faster
   2. Login & Signup Page
      1. Field is not cleared after wrong input (i.e. invalid email/password, incorrect password structure). No tedious field re-entry
7. Reduce short-term memory
   1. Login Page
      1. User can see their input history when they clicked the input box(login username input etc.)
   2. Game Page
      1. Display the user’s prior incorrect guesses

c. Draft task analysis and user profile

**User Roles:**

* Signed-in User
* Anonymous User (Not signed-in)

**User Profile:**

User (with account)

* Demographics
  + Age Majority: young adults (18-28)
  + Specialists, hobbyists, or interested in software/digital processing technologies; works in the field
  + Slight male gender majority
* Personality differences
  + can be sensitive to imagery regarding triggering topics
* Cultural/international differences
  + can be sensitive to imagery regarding religion
* Beliefs
  + Have no stance on AI digital art controversy
  + In favour of using AI generated art as a legitimate tool for digital art
* Needs
  + Want to diagnose the accuracy of AI generated art
  + Find what AI art generators struggle with
  + Want a fun casual guessing game
* Preferences
  + Wants an easy interface
  + Quick game setup

Guest User (no account)

* Demographics
  + Age Majority: young adults (18-28)
  + Specialists, hobbyists, or interested in software/digital processing technologies; works in the field
  + Slight male gender majority
* Personality differences
  + can be sensitive to imagery regarding triggering topics
  + Wants to try game
* Cultural/international differences
  + can be sensitive to imagery regarding religion
* Beliefs
  + Have no stance on AI digital art controversy
  + In favour of using AI generated art as a legitimate tool for digital art
* Needs
  + Wants to diagnose the accuracy of AI generated art
  + Find what AI art generators struggle with
  + Wants a fun casual guessing game
* Preferences
  + Wants an easy interface
  + Quick game setup

**Task Analysis:**

| **Task** | **Description** | **Scenario** | **Considerations/ influences** | **Critical points** | **Functionality** |
| --- | --- | --- | --- | --- | --- |
| **Setup a game** | The initial steps before playing the game. | A user initiates the game setup process, chooses a difficulty, and starts the game. | May initially question how long this process may take | User error in choosing difficulty | -Click play button  -Click easy/medium/expert button  -Click start game button |
| **Play a game** | The process of guessing the keywords used in creating AI generated images. A complete game will have the user either win or lose. | A user guesses the keywords of the images until they win. | -How do I win?  -How do I lose? | -How much more tries they have before the loss scenario (must be apparent and eye-catching)  -actual keywords in prompt are not well reflected in images (unfair or very hard to guess)  -sensitive imagery could be produced by AI images in game | -Input guess into field  -Press enter or guess button  -Loop until win/lose (allowed 5 wrong guesses) |
| **Quit a game** | Exiting a game in progress. | Exit out of a game before a win/loss condition has been met. | n/a | n/a | -Click exit button  -Click yes on exit confirmation |
| **View overall stats (User-only)** | Look at stats spanning from all the games while the user was logged in. | Go to user page | -What kind of stats are viewable  -Can I reset stats  -Are their game specific stats for my previous games  -are their public stats | n/a | -Press user icon |
| **Login (User-only)** | Log into account | User enters their account credentials and logs into account | n/a | n/a | -Select email field  -Input email  -Select password field  -Input password  -Click login button |
| **Signup (Guest User)** | Sign up for an account to log into | User enters credentials to be associated with that account and creates an account. | n/a | n/a | -Select email field  -Input email  -Select username field  -Input username  -Select password field  -Input password  -Select confirm password field  -Input password -Click signup button |
| **Signout (User-only)** | Sign out of account | Access user page to sign out of account | n/a | n/a | -Click delete account (if not there already)  -Click signout button |
| **Delete Account (User-only)** | Delete current account logged onto | Access user page to delete account and verify deletion | n/a | n/a | -Click delete account (if not there already)  -Click confirmation to delete account |

d. A draft list of risks and issues

* Learning ASP.net while working with the app at the same time
* A firebase/MongoDB might not have enough storage to contain many AI generated images
* We cannot control the content of images and prompts that may be inappropriate/controversial
* We are not experienced with hosting a website yet
* AI generated art is still new and not the best (sometimes images generated will be inaccurate and not represent the prompts)

E. Pictures or screen shots of your prototype

