**Windows Software for Human Studies**

**CSCI 6838 – Capstone**

**November 10, 2018**

Dr. Ioannis Pavlidis

UH Computational Physiology Lab (CPL)

Shaila Zaman

UH Computational Physiology Lab (CPL)

Dr. Pradeep Buddharaju

University of Houston – Clear Lake

Academic Mentor1

University of Houston – Clear Lake

Ceaser, Kaleigh

CeaserK2139@UHCL.edu

Garcia, John

GarciaJ3794@UHCL.edu

Kaur, Ramanjit

RamanjitKaur8160@UHCL.edu

**Acknowledgements**

**Table of Contents**

**Topic Page #**

**Introduction ……………………………………………………….……… 01**

**Market Analysis .…….…………………………………………………… 02**

**Requirements ……………………………………………………………… 03**

**System Architecture ……..………………………………………………… 04**

**System Design ……………………………………………………………… 05**

**Implementation ……………..……………………………………………… 06**

**Milestones …………………………………………………………………... 07**

**Conclusion …………………………………………………………………. 08**

**Future Work …………….………………………………………………… 09**

**References ……………….………………………………………………… 10**

**Glossary ……………………………………………………………….……. 11**

**Appendix A - Code ……………….………………….……………………. 12**

**Introduction**

1. **Introduction:**

“Window Software for Human Studies” project is used to build an interface in C# that will dynamically deliver the contents to students and marking the beginning and end times. This is the universal website that can run on the computer, Microsoft surface or writing tablet. The student has to response with the stylus. The software will be used to capture all the relevant stylus variable (trajectory, speed etc.). The responses from subjects are saved in the AWS S3 bucket.

This application has 2 modules namely, Subject Interface and Experimenter Interfaces integrated through Amazon Web Services (AWS). Experimenter module is responsible for creating exams and store it in AWS database. Subject module would be responsible for taking exams created by Experimenter. Responses of the exams taken by Subjects would be stored in AWS database and then in central repository called Subject Book. Exams created by Experimenter would also be stored in Subject Book.

* 1. **Background:**

“Window Software for Human Studies” is an application for getting information about users, also called Subjects, through exams. Subjects will be asked to take exams and the responses from users will be saved in AWS database and then in Subject Book, which will eventually be used in research for managing stress of humans. This application is a part of ongoing research project being conducted in Computational Physiology Lab (CPL). It was founded in 2002 by Dr. Ioannis Pavlidis and since it has been appreciated worldwide for its active involvement in research activities. At this moment the lab has three research lines and an educational research effort on science ethics. CPL is an interdisciplinary lab and draws additional expertise from partner labs in The Mayo Clinic, The University of Texas Medical School, and The Houston Methodist Hospital.

There are three software modules that are being used by this lab, namely: Subject Book, U- Interface, and S-Interface for management of stress. These interfaces communicate via Google drive. The S-Interface has been built as a modular system in C#. It is a communal software development project, where anybody can contribute modules with specific requirements. This interface has real time multiple sensor channels during the experimental session, uploading the collected the data to a designated Google Drive. The U-Interface captures the subject’s response to questionnaires’, uploading the collected data to a designated Google Drive. Once in the Google Drive, much like the sensors data are curated and managed by Subject Book.

The “Window Software for Human Studies” is an application similar to U-Interface that will also contribute to the data about users stored in Subject Book through getting responses from users by asking them to take exams. Users, also referred as Subjects, will be asked to take exams consisting questions from various fields. These questions would be fed to Subjects by Experimenters. Experimenters are the one who are responsible for creating exams for Subjects.

* 1. **Purpose:**

The purpose of the tool integrated with the project is to collect data about Subjects. This data will be helpful to analyze users. The project will develop two User Interfaces (UI) that are integrated with the AWS. These interfaces are:

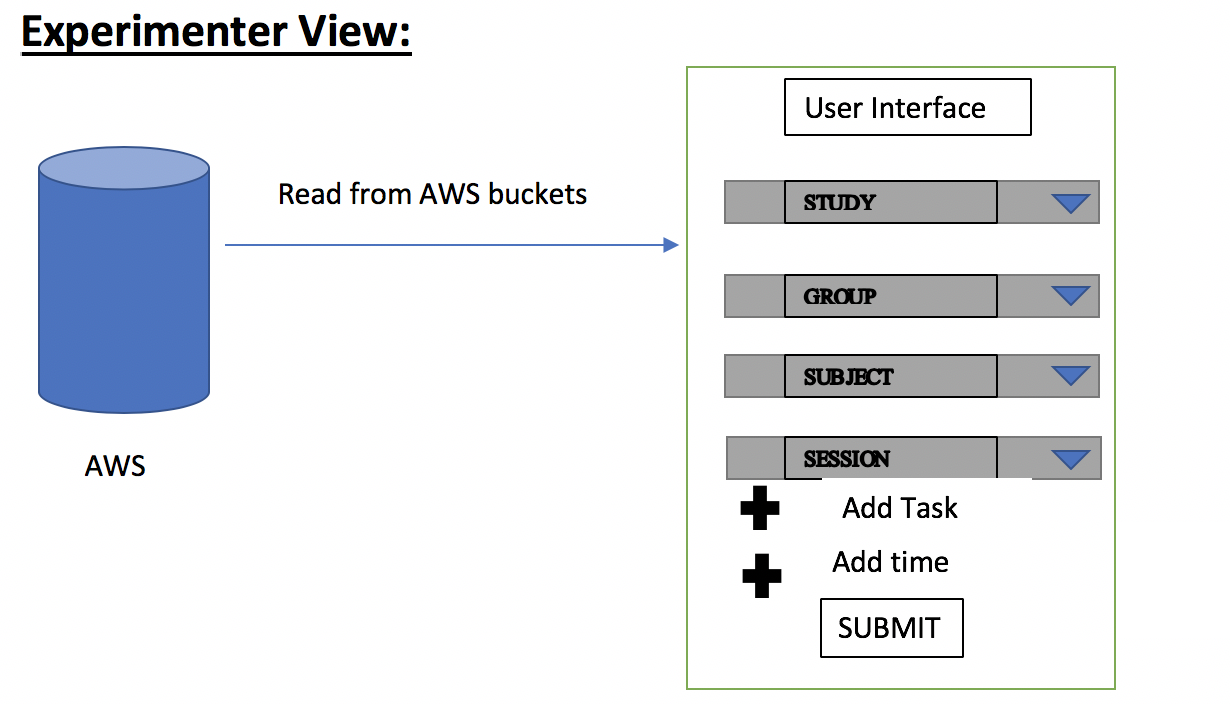
1. Experimenter Interface
2. Subject Interface

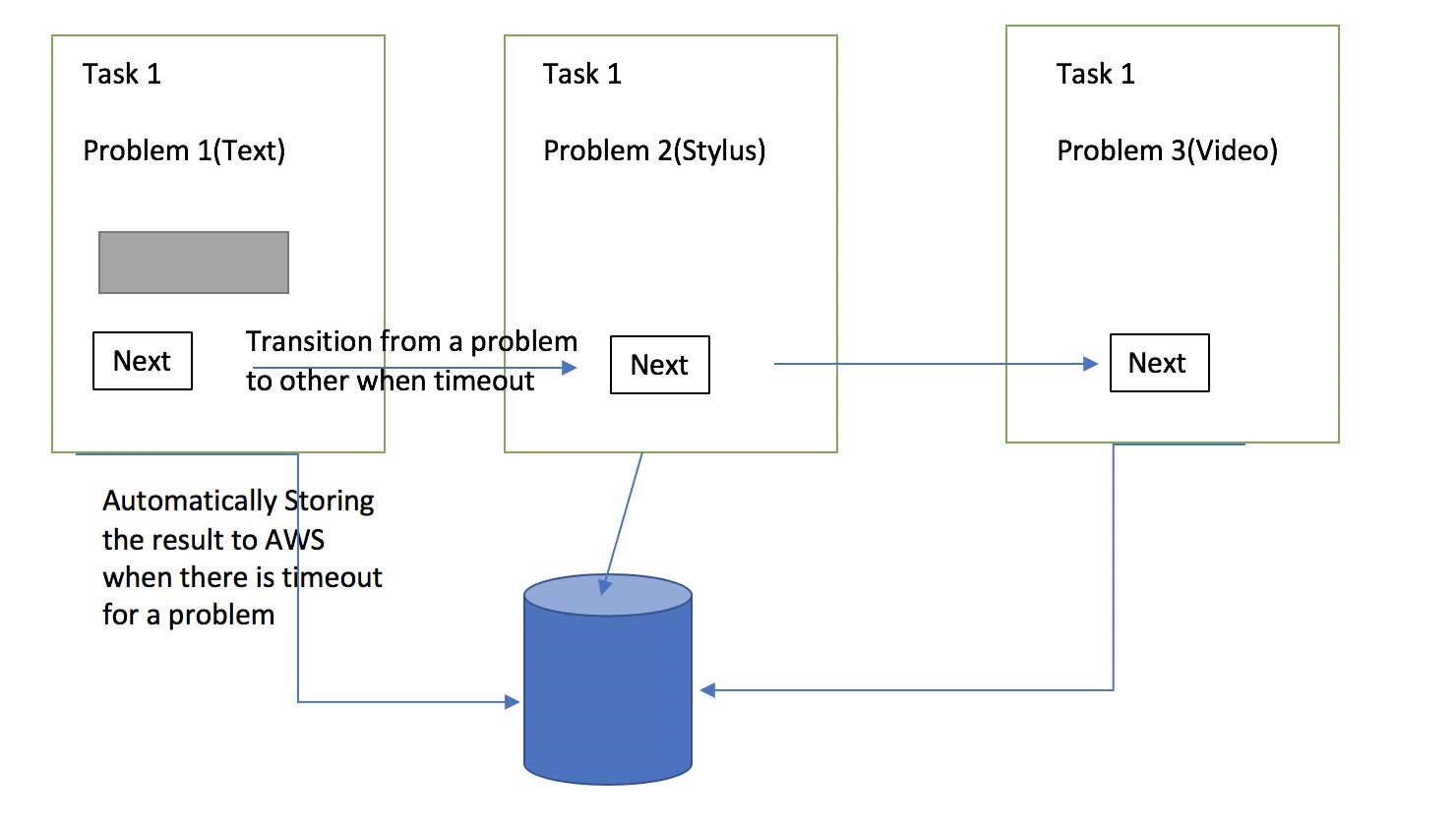
**Experimenter Interface**: This interface is used to create tasks for subjects. Each task contains questionnaire of various formats like audio, video, text and image. Questions will be timed and untimed as well. Experimenter can set the duration of the task. Experimenter can load the questions that are predefined in the AWS or can make the new task. For loading a new task, experimenter has to set the study, group, subject and the session. These Variable are saved in the config file in the AWS. For creating task, experimenter needs to define the task type as well as the duration of the task.

**Subject Interface**: Subject interface is designed for the subjects to do the task. These tasks are timed or untimed. For the timed questions, subjects are required to answer with the allotted time showing on screen. If the user fails to provide the response, user will be navigated to other view that contain the other problem and blank response will be saved. Subject won’t get the option to go back to answer that question. These responses are capture and stored in the AWS S3 bucket. Each task response as well as task duration is stored in the CSV file.

**AWS integration:** Experimenter is delivering the questions to subjects via AWS. Experimenter can send the predefined task to the users or can upload the new task for the user. For uploading the new task, session variable is used that dynamically deliver the questions to the subject interface. For sending, the predefined task, experimenter will interact with the AWS. For retrieving the task and uploading the responses, GetObject and PutObject functions of AWS are used.

The basic diagram of how this interaction works is given below:





* 1. **Project Scope:**

1. Stress is a big issue as it hampers productivity.
2. Measuring stress level is important in order to take actions to prevent it.
3. To prevent or to lessen the stress, gathering data is very important.
4. This application is used to gather required data for analysis purpose.

**Market Analysis**

If the theme of the capstone project is research based, then this section would be entitled *Related Research*. The idea is to identify research similar research and how your research differs (and adds value) from other research.

The capstone project embraces this theme and puts a pragmatic spin on the idea.

Market Analysis seeks to identify the competitors for the product in development. Specify the features of the other products. Identify common features. Also, what features may be present in the other products not necessarily present in the capstone version.

**Requirements**

Create a web-based application that will allow a user, known as the Experimenter, to create and assign tasks to another user referred to as the Subject.

**Business Requirements:**

**BR01**. Experimenter needs to be able to login into their account

**BR02**. Experimenter needs to be able to create a configuration

**BR03**. Experimenter needs to be able to choose a saved configuration in Amazon AWS

**BR04**. Experimenter needs to be able to save a configuration

**BR05**. Experimenter needs to be able to assign a configuration to a specific session

**BR06**. Subject needs to be able to view tasks within a configuration (one task at a time)

**BR07**. Subject needs to be able to respond to tasks if that option is selected in a task

**BR08**. Web application needs to limit access to Subject.

**BR09**. Web application needs to capture the start and end times of each task

**BR10**. Web application needs to terminate a task when Subject has exceeded the amount of time

allotted.

**BR11**. Subject responses need to be stored on Amazon AWS in csv file

**Technical Requirements:**

**TR01.01**. Provide a login view using username and password

**TR02.01**. Provide Experimenter with a view that displays created tasks and an option to

create new tasks.

**TR03.01**. Provide Experimenter with a dropdown menu of saved configurations files stored

on Amazon AWS

**TR04.01** Provide Experimenter with a button to save a configuration to Amazon AWS

**TR05.01**. Provide Experimenter a button that switches to a web-view for the Subject to

view the specified configuration

**TR06.01**. Provide a view that dynamically view different tasks.

**TR07.01**. Provide Subject with a text box to type responses.

**TR08.01**. Create two membership roles, one for the Experimenter and one for Subject. When

the role is set to Subject disable or hide information that is not necessary.

**TR09.01**. Get a timestamp when a task is started and when the task is terminated (either by

the Subject or the application due to a timeout)

**TR10.01**. Keep track of elapsed time and terminate task when task duration is met.

**TR11.01**. Stream responses to Amazon AWS

**System Requirements:**

* Hardware
  + Microsoft Surface Pro Tablet
  + Stylus
* Software
  + C#
  + .NET framework
  + AWS

**System Architecture**

**Initial High-Level Design**

Our initial high-level design used in developing the application is divided into three components:

1. Experimenter User Interface Web Forms
2. Subject User Interface Web Forms
3. Business Layer and Storage Libraries

The steps for the flow of data through the application are as follows:

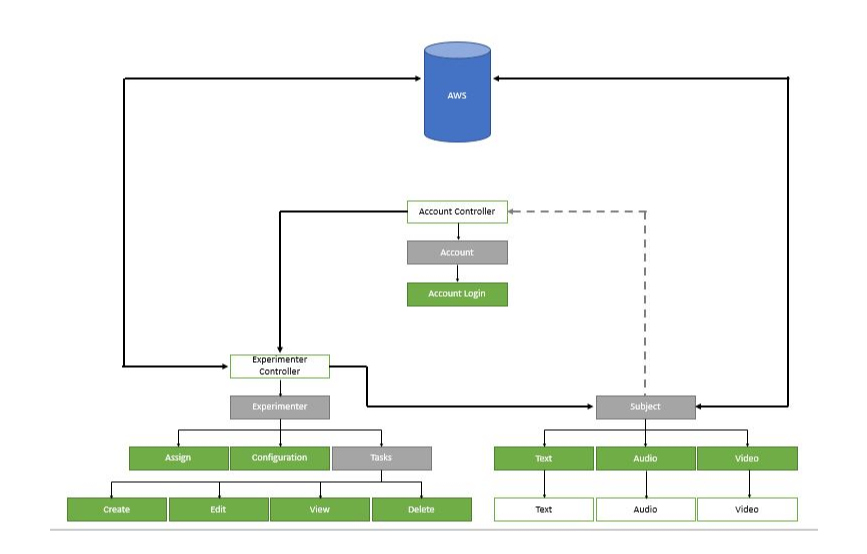
1. Read Amazon Web Services S3 file structure and populate Experimenter Form drop down lists.
2. Input to the Experimenter UI to pick task type, content, duration and feedback type for each Subject task.
3. Submit task list from Experimenter UI to Subject UI.
4. Take more input from the Test Subject in the Subject UI as form answers.
5. Output these answers to responses.csv.
6. Store responses.csv to Amazon S3 Storage Services to appropriate folder.

The round-trip engineering components consisted of the following technologies:

1. ASP.NET version 4.6.1 with C# 7.0 as a web development framework.
2. Azure DevOps and BitBucket Cloud used as source version control platforms.
3. Third Party Libraries as helper methods for audio\video playback and timer functions.
4. Amazon S3 Software Development Kit as a collection of methods for cloud storage functions.
5. Stylus input tracking library as a list of functions to gather data about the user input.
6. Visual Studio 2017 plus AWS toolkit is as an Integrated Development Environment.

**Final High-Level Design**

The final high-level design is the same as the initial design except that an additional function to pass data from the experimenter module to the subject module has been added and is used to bridge the invocation and data passing between two ASP.NET project types. This was done to ease the integration of the two modules within the application. The final source is a hybrid of web forms and a model-view-controller design as shown in Figure 1. The data flows from the Storage folder structure inputs to Experimenter UI to the Subject UI to responses in AWS.



**Figure 1 –** High Level Final Design Diagram

**Implementation Details**

Implementation consisted of producing and reviewing the wire frames for the user interfaces and developing the business layer and storage library methods to be called from the user interface code behind files. ASP.NET produced the correct presentation layer and java script to be rendered by the browser seamlessly making this solution compatible with any hardware capable of running a so called “uplevel” web browser, that is a browser that supports at least the following:

* ECMAScript (JScript, JavaScript) version 1.2.
* HTML version 4.0.
* The Microsoft Document Object Model (MSDOM).
* Cascading style sheets (CSS).[1]

[1] <https://docs.microsoft.com/en-us/previous-versions/aspnet/x3k2ssx2(v=vs.100)>

An interesting aspect of the implementation was how to make the timer common to the subject forms and expire each task at the right time.

**Testing Details**

The testing objective for the application is to ensure that either the experimenter or the test subject has an intuitive and error free user experience using a variety of hardware and web browsers.

The following web browsers were tested with the application:

1. Google Chrome version 70.0.3538.77
2. Internet Explorer 11 to Microsoft Edge 42.17134.1.0
3. Safari 11.1.2 to 12.0.1

The following tests were performed on each browser to verify the application:

1. Use the same three versions of type task mix and test subject responses on each browser. Does the application produce equivalent outputs?
2. Is the application responsive for each mix of tasks and responses?
3. Is the application intuitive or is the content distracting to either the user or experimenter?
4. Does the application guide or prompt the user regarding proper input formats?

**System Design**

This section could include various UML diagrams including, but not limited to *Use Case diagrams* or *Sequence diagrams*. Include only if it helps better understand the project.

**Implementation**

Discuss the most important/interesting aspects of implementation. It probably won't be possible to discuss everything - give a rationale for what you do discuss. Explain how the program/system was verified.

**Milestones**

This section describes the project milestones, their due dates, and their current status. The milestones in this report should match those on the team web site. If any milestone was late or not completed, you may provide an explanation.

**Conclusion**

The conclusion bring the document to a closure. Review the context and motivation for doing this project. Describe how this project helps the client in their business. Explain various features of the project. Wrap up the conclusion by discussing the impact the project solution will have for your client, e.g., time-saved, accuracy improved, throughput increased...whatever. Add some specific numbers if possible.

Finally, talk about broader impacts (if any) that your product might have, i.e., other contexts or clients that might be able to use it, potential for broader markets, and so on.

Close with some reflective comments on your team, the project process, and the Capstone class.

**Future Work**

Every good project will generate new ideas beyond the scope of the initial requirements that were not implemented. Discuss any additional features that could be implemented in future versions of this project.

**References**

Please see the *ACM Citation Guidelines* document for details on how to set up references.

**Glossary**

If this document includes any terms that have special meaning (domain-specific terminology, for example), provide those definitions here. Place the terms in alphabetical order.

**Appendix A - Code**

The first appendix will include the code developed during the project. The code should be documents with header comments which are comments at the beginning of each file. There will also be comments for each function/procedure. Use the courier font for the code.

Please see the *Documented Coding Example* file for an example on how to document your code.

Comment: Include any documents that you feel make your design specification easier to understand but are not central to the project's description.

There could be more than one appendix. If so, label them Appendix A, Appendix B, etc.