**Online Bio-Informatics Toolkit**

**Architectural Spike Report**

**By**

**Mason Monday**

**Sadaira Packer**

**Paul Ryu**

**Ben Williams**

**Ansleigh Yancey**

**Submitted in partial fulfillment of the requirements for COMP 4710 Senior Design to the Department of Computer Science and Software Engineering, Samuel Ginn College of Engineering, Auburn University**

**Auburn, Alabama**

**February 4, 2019**

**System Metaphor**

There is a lack of methods that organize genetic variants for diseases, tissues, and cell types. Advanced computational methods and software are needed to create a database and online web server that can address these issues. Our objective is to develop an interactive, online database that is searchable by disease type and/or tissue and cell line. Users will also be able to request the entry or removal of information in the database.

**Cycle Intent**

Our main goal for this architectural spike cycle was to create a database for labeling and storing information of genetic variants in the human genome. This involved inputting the necessary data from our sponsor into the database and searching for the most ideal web server for maintaining our project. In addition, we intended to create a simple user interface for users to search for desired data and request additions and deletions.

**User Stories**

**User Story:** Add Entry Request (Text Box)

**Brief Description:** The actor sends a request to add an entry to the database using the text box.

**Actor:** User

**Precondition:** The user is connected to the internet and has navigated to the website.

**Basic Flow of Events:**

1. The actor navigates to the webpage.
2. The actor chooses the “send add request” option.
3. The actor enters information on a genetic mutation or genome signal into the provided text box.
4. The actor’s request to add the entry to the database is sent to the administrator.

**Extensions:**

1a. Failure to send entry request:

* The text box entry was empty, and no request was sent to the administrator.

**Post-Conditions:**

1a. The Entry is added to the database:

* The administrator reviewed the actor’s request and permitted its addition to the existing database.
* The requesting actor is notified.

1b. The Entry is not added to the database:

* The administrator reviewed the actor’s request and did not permit its addition to the existing database.
* The requesting actor is notified.

**User Story:** Add Entry Request (Text File)

**Brief Description:** The actor sends a request to add an entry to the database by uploading a .txt file.

**Actor:** User

**Precondition:** The user is connected to the internet and has navigated to the website.

**Basic Flow of Events:**

1. The actor navigates to the webpage.
2. The actor chooses the “send add request” option.
3. The actor uploads a file with information on a genetic mutation or genome signal.
4. The actor’s request to add the entry to the database is sent to the administrator.

**Extensions:**

1a. Failure to send entry request:

* No file was uploaded and no request was sent to the administrator.
* The file uploaded was not in .txt format.

**Post-Conditions:**

1a. The Entry is added to the database:

* The administrator reviewed the actor’s request and permitted its addition to the existing database.
* The requesting actor is notified.

1b. The Entry is not added to the database:

* The administrator reviewed the actor’s request and did not permit its addition to the existing database.
* The requesting actor is notified.

**User Story:** Add Entry Via Text Box (Administrator)

**Brief Description:** The actor adds an entry to the database using the text box.

**Actor:** Administrator

**Precondition:** The actor is connected to the internet and has navigated to the website and is an administrator of the site.

**Basic Flow of Events:**

1. The actor navigates to the admin webpage.
2. The actor chooses the “add entry” option.
3. The actor enters information on a genetic mutation or genome signal into the provided text box.

**Extensions:**

1a. Failure to add entry:

* The text box entry was empty, and no entry was added.

**Post-Conditions:**

1. The Entry is added to the database

**User Story:** Add Entry Via File (Administrator)

**Brief Description:** The actor adds an entry to the database by uploading a file.

**Actor:** Administrator

**Precondition:**

1a. The actor is connected to the internet and has navigated to the website.

1b. The actor is an administrator of the site.

**Basic Flow of Events:**

1. The actor navigates to the admin webpage.
2. The actor chooses the “add entry” option.
3. The actor uploads a file with the information on a genetic mutation or genome signal.

**Extensions:**

1a. Failure to add entry:

* The file uploaded was not in the .txt format, and no entry was added.

**Post-Conditions:**

1. The Entry is added to the database.

**User Story:** Delete Entry Request

**Brief Description:** The actor sends a request to delete an existing entry from the database.

**Actor:** User

**Precondition:** The user is connected to the internet and has navigated to the website.

**Basic Flow of Events:**

1. The actor navigates to the webpage.
2. The actor finds the entry that he/she would like to be removed.
3. The actor chooses the “send delete request” option.
4. The actor enters his/her reasoning for deletion (optional).
5. The actor’s request to delete the existing entry is sent to the administrator.

**Post-Conditions:**

1a. The Entry is deleted from the database:

* The administrator reviewed the actor’s request and permitted its removal from the database.
* The requesting actor is notified.

1b. The Entry is not deleted from the database:

* The administrator reviewed the actor’s request and did not permit its removal from the database.
* The requesting actor is notified.

**User Story:** Delete Entry (Administrator)

**Brief Description:** The actor deletes an existing entry from the database.

**Actor:** Administrator

**Precondition:**

1a. The actor is connected to the internet and has navigated to the website.

1b. The actor is an administrator of the site.

**Basic Flow of Events:**

1. The actor navigates to the webpage.
2. The actor finds the entry that he/she would like to be removed.
3. The actor chooses the “delete entry” option.

**Post-Conditions:**

1a. The Entry is deleted from the database:

* The administrator removed the desired entry from the database.

**User Story:** Edit Entry Request

**Brief Description:** The actor sends a request to edit an existing entry from the database.

**Actor:** User

**Precondition:** The user is connected to the internet and has navigated to the website.

**Basic Flow of Events:**

1. The actor navigates to the webpage.
2. The actor finds the entry that he/she would like to edit.
3. The actor chooses the “send edit request” option.
4. The actor enters his/her requested changes and his/her reasoning for said changes in the provided text box.
5. The actor’s request for the existing entry to be edited is sent to the administrator.

**Extensions:**

1a. Failure to add entry:

* The text box entry was empty, and no edit request was sent.

**Post-Conditions:**

1a. The Existing Entry is Edited:

* The administrator reviewed the actor’s request and permitted the suggested changes. The entry is edited on the database.
* The requesting actor is notified.

1b. The Existing Entry is Not Edited:

* The administrator reviewed the actor’s request and denied the suggested changes. The entry remains the same on the database.
* The requesting actor is notified.

**User Story:** Edit Entry (Administrator)

**Brief Description:** The actor edits an existing database entry.

**Actor:** Administrator

**Precondition:**

1a. The actor is connected to the internet and has navigated to the website.

1b. The actor is an administrator of the site.

**Basic Flow of Events:**

1. The actor navigates to the admin webpage.
2. The actor finds the entry he/she desires to edit.
3. The actor chooses the “edit entry” option.
4. The actor alters the existing entry’s information using the text box.

**Extensions:**

1a. Failure to add entry:

* The text box was empty, and no edit was submitted.

**Post-Conditions:**

1. The Entry is edited:

* The Administrator’s changes to the existing database entry are saved.

**User Story:** Search for Database Entry (Cell Line)

**Brief Description:** The actor searches the database for the desired genetic mutation and/or human genome signal.

**Actor:** User

**Precondition:** The user is connected to the internet and has navigated to the website.

**Basic Flow of Events:**

1. The actor navigates to the webpage.
2. The actor completes the search entry based on predefined search options in the “cell line” drop down box.
3. The actor chooses the “search database” option.

**Extensions:**

1a. Failure to search for entry:

* The search options were incorrectly completed.
* The search options were incomplete.

**Post-Conditions:**

1a. The Existing Entry is Found:

* The search options for the cell line yields one or more results.
* The information is displayed for the actor.

1b. The Existing Entry is Not Found:

* The search options chosen by the actor do not yield a result.
* There is no existing entry in the database.

**User Story:** Search for Database Entry (disease type)

**Brief Description:** The actor searches the database for the desired genetic mutation and/or human genome signal.

**Actor:** User

**Precondition:** The user is connected to the internet and has navigated to the website.

**Basic Flow of Events:**

1. The actor navigates to the webpage.
2. The actor completes the search entry based on predefined search options in the “disease type” drop down box.
3. The actor chooses the “search database” option.

**Extensions:**

1a. Failure to search for entry:

* The search options were incorrectly completed.
* The search options were incomplete.

**Post-Conditions:**

1a. The Existing Entry is Found:

* The search options for the disease type yields one or more results.
* The information is displayed for the actor.

1b. The Existing Entry is Not Found:

* The search options for the disease type chosen by the actor do not yield a result.
* There is no existing entry in the database.

**User Story:** Search for Database Entry (cell line & disease type)

**Brief Description:** The actor searches the database for the desired genetic mutation and/or human genome signal.

**Actor:** User

**Precondition:** The user is connected to the internet and has navigated to the website.

**Basic Flow of Events:**

1. The actor navigates to the webpage.
2. The actor completes the search entry based on predefined search options in the “cell line” and “disease type” drop down boxes.
3. The actor chooses the “search database” option.

**Extensions:**

1a. Failure to search for entry:

* The search options were incorrectly completed.
* The search options were incomplete.

**Post-Conditions:**

1a. The Existing Entry is Found:

* The search options for the cell line and disease type yields one or more results.
* The information is displayed for the actor.

1b. The Existing Entry is Not Found:

* The search options for the cell line and disease type do not yield a result.
* There is no existing entry in the database.

**User Story:** Download Search Results

**Brief Description:** The actor downloads information about a genetic mutation and/or human genome signal.

**Actor:** User

**Precondition:** The user is connected to the internet and has navigated to the website.

**Basic Flow of Events:**

1. The actor navigates to the webpage.
2. The actor completes the search entry based on predefined search options in the “cell line” and/or “disease type” drop down boxes.
3. The actor chooses the “search database” option.
4. The actor chooses the download type (excel, text, word document).

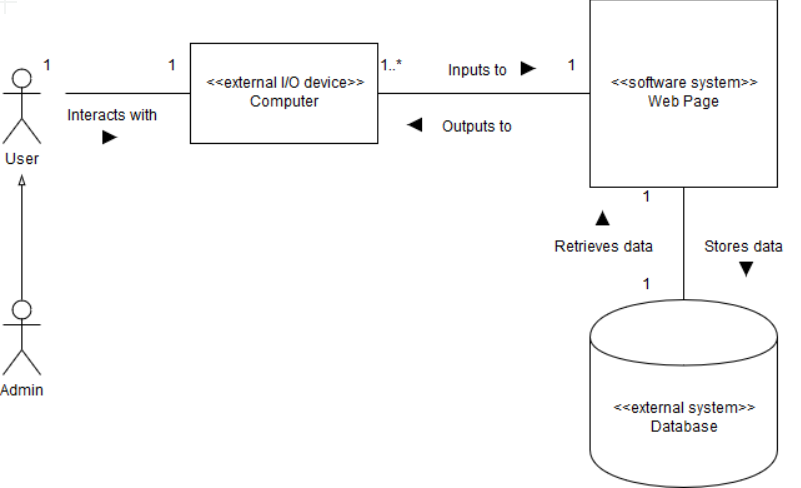
**Post-Conditions:**

1. The Existing Entry is Found:

* The actor saves the file to his/her personal device.

**Design Documentation**

Architecture: The human genome data is stored on a database developed in R. This database is linked to a webpage developed with R and hosted on the web server Shiny that our users will interact with. Users will input the values for the specific piece of data they wish to access into the webpage, which will then search our database for the data. Once found, the webpage retrieves the data and displays it to the user. Also, users may request changes to the database, such as adding a new entry. Upon submitting a change, the webpage holds the change until it is approved, after which the webpage will edit the desired entry.



Structure: The database application will be created using the R Shiny framework, and will be hosted using shinyapps.io. The data will be contained within a table containing seven columns and a row for each entry, constructed with the following attributes:

* ID: the unique value assigned to each entry in the database
* chrom: the name of the chromosome or scaffold
* chromStart: the starting position of the chromosome or scaffold
* chromEnd: the ending position of the chromosome or scaffold
* name: the name of the genetic variant
* score: the score given to the genetic variant
* blocks: the specific block the genetic variant resides on

Users may submit requests to add and delete entries. Upon approval by an administrative party, the table will be updated. The majority of the UI and server, as well as methods to CRUD (create, read, update, or destroy entries) will be hosted in a file called “main.R” . The table will, by default, be populated with just over 4.3 million entries from a provided .bed file. As development will be done in the RStudio IDE using Git as the primary file sharing service, the system will also contain default RStudio project files and a Git repository. Documentation on the table entries can be found at <https://genome.ucsc.edu/FAQ/FAQformat.html#format1>.

**Lessons Learned**

**Ansleigh Yancey:** From this portion of the project, I learned a great deal about web design and databases. Prior to this, I have had limited experience with R and RStudio, so the learning curve for a database with over 4 million entries has been steep but rewarding. I have learned to listen to the client and do my best to fulfill requests. This is also the first time I have made a user interface for a web page, so that has been challenging yet interesting. I have enjoyed the experience thus far and look forward to the next stage.

**Mason Monday:** Aside from having to learn R and the Shiny framework, most of what I learned included properly communicating with our sponsor and teammates. Getting specific information from our sponsor involved a bit of pressuring to get what we needed, showing me how to properly discuss our needs to him. Keeping up with discussion minutes and topics during meetings has helped us with our book-keeping, and looking back on what we’ve talked about previously has been a big help.

**Benjamin Williams:** During the Architectural Spike phase, I learned how to develop a basic web application with R and R Shiny. This involved learning the basics of databases in general, as I had never had experience with that prior to this project. I also experienced working with a real client, which included the difficulties of communication; since the client defines the criteria and the subject matter is foreign to us, understanding the vision for the project was difficult in the beginning.

**Paul Ryu:** I learned many things about the R language and the package that came along with it - Shiny. I always thought that R was just another boring language for statistics and computing but it has turned out to be so much more and can even make graphics that look more advanced than the outputs of its modern language counterparts.