YoStretch

For a healthier lifestyle

Ly Harriet Bui

I. Introduction and Motivation

Nowadays, an averaged office worker sits for about 10 hours. All of these hours are spent in front of a computer checking emails, doing research, or making phone calls. These sitting hours are increased by the time watching videos and surfing the Internet during their free time. Medical researches have found that prolonged sitting is positively correlated with heart disease, diabetes, obesity, cancer, and depression. It also contributes to muscle and joint problems. Some of them claim that sitting is even worse for health than smoking and more fatal than HIV. Rigorous exercises after an extended period of sitting cannot compensate for the health loss. (Schulte, 2015), YoStretch application will be attempting to solve this health problem.

YoStretch is a yoga app that encourages people to stand up and stretch every one or two hours after sitting for an extended period of time. The YoStretch is an offline application that allows user to track how long they have been sitting in front of a computer using facial recognition tools such as IHart and the computer's webcam. The user can use default setting or customize the timer. After, for example, one hour, the app will ring and alert the user to get up and stretch. At this point, user can choose beginning, intermediate, or advanced level of yoga poses they would like to do. Then, images will appear to guide them how to do the poses. The user can click a start button to start a countdown timer that recommends a certain stretch time. After user finish stretching, they will be given points based how long they have stretched within the day. If they stretch long enough, they will keep Olaf alive, otherwise, he will be melting and die. They user can also choose to record their stretching frequency by hitting the button *Add To Calendar*. The application will keep a stretching calendar for the user.

II. Updated Project Proposal

I managed to finish and implement a timer, and functions for user to select different yoga pose level, chose a random group of four yoga poses, and view instruction text and image of a specific yoga pose.

III. Milestone achieved

- Obtain final project approval
- Finish pitch document
- Develop project presentation
- Give project presentation
- Create repository for project
- Define yoga pictures and yoga terms data structure
- Enter yoga pictures and yoga terms into the data structure
- Code to search for yoga pictures based on yoga terms and evaluate how many pictures are missing, fix bugs
- Stub prototype of GUI
- Matchup backend with GUI stubs
- Final Report complete

IV. Actual Timeline Achieved

Week 1: March 28 - April 1

- Finish Pitch Document
- Obtain final project approval
- Create repository for the project

Week 2: April 4 - April 8

- Develop project presentation
- Define yoga pictures and yoga terms data structure
- Enter yoga pictures and yoga terms into the data structure

Week 3: April 11 - April 15

Give project presentation

- Code to search for yoga pictures based on yoga terms and evaluate how many pictures are missing, fix bugs
- Code to search for yoga yoga log based on dates and evaluate how many data are missing, fix bugs
- Built a database in CSV file
- Read in from CSV file

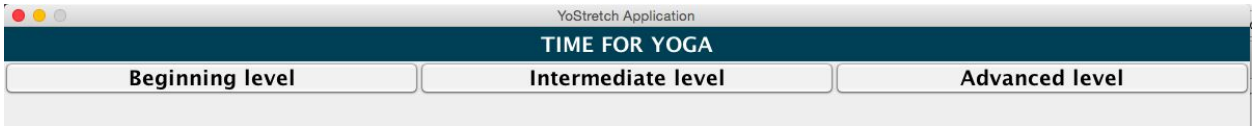
Week 4: April 18 - April 26

- Stub prototype of GUI
- Matchup backend with GUI stubs
- Final Report complete
- 4/26 Final project due

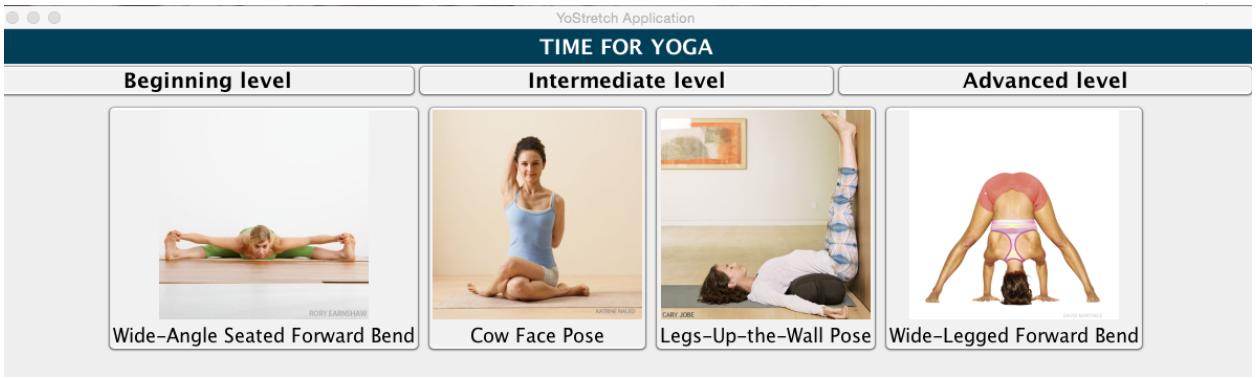
IV. The final product

The application aims to remind people to get up and stretch after every one hour of sitting in front of the screen. Therefore, a timer has been implemented to start when the program is started.

After one hour, the text “TIME FOR YOGA” will appear. In the future, I will implement audio ringtone to remind user.



The user than can choose different levels of yoga stretching and strengthening exercise they are comfortable with. For example, in the following picture, the beginner level is chosen



Each time the user clicks a level buttons, 4 new random pose will be generated

Then the user may click on one of the pose to look at detailed instruction text and image:

V. Data structure

At the moment, I use ArrayList in Java as the temporary data structure and LinkedList as the main data structure. After reading the CSV file, I store the information into 3 arrays: yoga name, yoga description, and yoga image path. The ArrayList in Java conveniently allows to fast access to information stored in the array via indices and insert the correct yoga name, yoga description and yoga image path into a yogaPose object. However, Arraylist takes up more space compared to other data structure such as Linked List as it has to double its size every time a new element is added into the array.

Array List

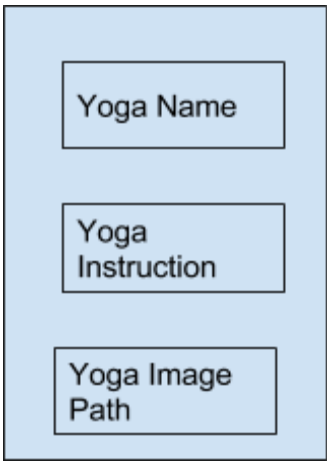
Yoga name	Yoga name	Yoga name			

Yoga description	Yoga description	Yoga name description			

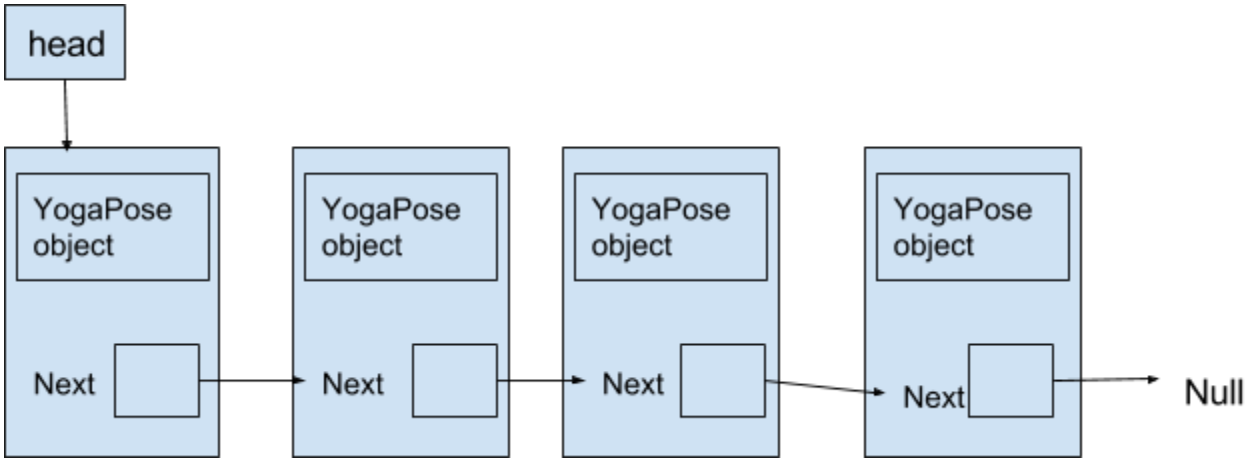
Yoga image path	Yoga image path	Yoga image path			

My main data structure implemented in the project is three linked list for different yoga levels: Each linked list stores a list of yogaPose objects. Linked List has an advantage of fast insert speed at a constant. It can be changed dynamically and save space. Unlike an arraylist, a linkedlist occupies as much space as the amount of data present.

YogaPose object



Linked List



I generate random numbers within a linked list size every time user click beginning, intermediate, or advanced button to choose four random yoga poses next to each other. This method is not the best to generate a random sequence of yoga poses as the organization of yoga poses within a linked list remains unchanged.

In order to output the right yoga instruction after the user clicking on a yoga pose button including a yoga name and an image of the pose I have to iterate through the whole linked list until I reach the node containing a yogaPose object that has the same yoga name as the one in the targeted button. The worse run time for search in a linked list is O(n). This is not a maximum solution for this problem. In order to reduce the search runtime to O(log n). However a red black tree will increase insearch runtime to O(log n) instead of a constant as a link list does. However, the insert method is only implemented one while search method will be called every time the user pick a pose to practice. Thus, the advantage of short search runtime outweigh the disadvantage of longer insert runtime.

VI. Code sample

I use Java for the backend and Java swing to build my GUI. I find that Java swing has a lot of disadvantages regarding displaying several pages. It was challenging for me to wrap texts for buttons and resize different elements. I have been trying to clean the instruction texts and image when the user click the level buttons. However, I have not been successful.

I have attempted to replace the linked lists with red black trees. I discovered that my implementation of the red black tree is currently not supporting inserting more than 20 data. Therefore, I still need to work on fixing this bug.

GUI Implementation

```
/**
    * if the buttons get clicked
    */
   public void actionPerformed(ActionEvent e) {

      // i  f beginningButton button got clicked
      if (e.getSource() == beginningButton) {
         //refresh JPanel
         yogaPosePanel.removeAll();
         //get random yogaPose
         getRandomPose(yoStretchController.linkedListBeginner);
         //display poses
         displayYogaPose(yoStretchController.linkedListBeginner.getFirstNode());
         System.out.println("temp node " + temp.getData().getYogaName());

      }

      // if intermediateButton button got clicked
      if (e.getSource() == intermediateButton) {
         //refresh JPanel
         yogaPosePanel.removeAll();
```

```
            //get random yogaPose
            getRandomPose(yoStretchController.linkedListIntermediate);
            //display poses
            displayYogaPose(yoStretchController.linkedListIntermediate.getFirstNode());


        }

        // if advanceButton button got clicked
        if (e.getSource() == advanceButton) {
            //refresh JPanel
            yogaPosePanel.removeAll();
            instruction().repaint();
            //get random yogaPose
            getRandomPose(yoStretchController.linkedListAdvanced);
            //display poses
            displayYogaPose(yoStretchController.linkedListAdvanced.getFirstNode());


        }
    }

/**
* Get 5 different yoga poses, display yoga name and corresponding image to a JButton
* If the button displaying a certain yoga pose get clicked, the instruction image and texts
for that pose apprears
* @param node
*/

public void displayYogaPose(LinkedListNode<YogaPose> node) {
        //numb variable to count the number of yoga poses displayed
        int numb = 0;
        //if the poses is smaller than 5
        while (numb != YOGAPOSENUM) {
            //create new button
            yogaPoseButton = new JButton("<html>" + yogaInstruction.replaceAll("\\n",
"<br>") + "</html>");
            //add ActionListener to that button
            yogaPoseButton.addActionListener(new ActionListener() {
```

```
            //add actionListener to yoga pose buttons
            public void actionPerformed(ActionEvent e) {
               //get the name displayed on the button
               String namePose = ((JButton) e.getSource()).getText();
               //temporary node moving along the linkedlist
               LinkedListNode<YogaPose> temp2 = node;

               //get the instruction of the yoga pose and yoga image selected
               while (temp2 != null) {
                  System.out.println("button");

                  //find the node containing the right instruction
                  if (namePose == temp2.getData().getYogaName()) {
                     //get yoga instruction
                     yogaInstruction = temp2.getData().getYogaInstruction();
                     //get image of the instruction and scale the image
                     icon = new
ImageIcon(this.getClass().getResource(temp2.getData().getYogaImagePath()), "star");
                     Image image = icon.getImage(); // transform it
                     Image newimg = image.getScaledInstance(450, 450,
                           java.awt.Image.SCALE_SMOOTH); // scale it the smooth way
                     icon = new ImageIcon(newimg); // transform it back
                     break;
                  }
                  //get the next node in the linkedlist
                  temp2 = temp2.getNext();


               }
               //display yoga instruction
               yogaInstructionText.setText(yogaInstruction);
               //display image
               yogaInstructionLabel.setIcon(icon);

            }
         });
```

```
        //set text
        yogaPoseButton.setText(temp.getData().getYogaName());
        //set font
        yogaPoseButton.setFont(fontYogaName);

        // create an ImageIcon variable and store the picture in the variable
        ImageIcon icon = new
ImageIcon(this.getClass().getResource(temp.getData().getYogaImagePath()), "star");
        Image image = icon.getImage(); // transform it
        Image newimg = image.getScaledInstance(200, 200,
              java.awt.Image.SCALE_SMOOTH); // scale it the smooth way
        icon = new ImageIcon(newimg); // transform it back
        yogaPoseButton.setIcon(icon);
        // to remote the spacing between the image and button's borders
        yogaPoseButton.setMargin(new Insets(0, 0, 0, 0));
        //set text position on button
        yogaPoseButton.setVerticalTextPosition(SwingConstants.BOTTOM);
        yogaPoseButton.setHorizontalTextPosition(SwingConstants.CENTER);

        //add buttons to yogaPosePanel
        yogaPosePanel.add(yogaPoseButton, c);
        yogaPosePanel.validate();
        yogaPosePanel.repaint();

        //add the next yoga pose
        numb++;
        temp = temp.getNext()
 }
```

Linked List and Array List

```
public class YoStretchController {

  //linked list to holds yogaPose object
  protected LinkedList<YogaPose> linkedListBeginner;
  protected LinkedList<YogaPose> linkedListIntermediate;
  protected LinkedList<YogaPose> linkedListAdvanced;
```

```
  /**
   * Constructor initializing linked lists and open files
   */
  public YoStretchController(){

      //initialize three linked lists for different yoga level
      linkedListBeginner = new LinkedList();
      linkedListIntermediate = new LinkedList();
      linkedListAdvanced = new LinkedList();

      //read in csv file to get yoga name, descripion and image path
      openFile("YogaDataBeginning.csv", linkedListBeginner);
      openFile("YogaDataIntermediate.csv", linkedListIntermediate);
      openFile("YogaDataAdvanced.csv", linkedListAdvanced);


  }




  /**
   * read from dictionary.txt and input the word into a List
   * generate a random word from the list as hidden word by creating random number
   */
  public void openFile(String fileName, LinkedList<YogaPose> linkedList) {

      /**
       * try read in fileName, insert yoga name, description and image path into yogaPose
object, than insert them
       * into linkedlist
       */
      try {
          //csv file containing data
          //create new file
          File fileCSV = new File(fileName);
          // get the path to the file
          String fileNamePathCSV = fileCSV.getAbsolutePath();
          //reader reads csv file
```

```
        CSVReader reader = new CSVReader(new FileReader(fileNamePathCSV), ',', '"',
'|');

        //lists to hold yogaName
        List<String> yogaNameList = new ArrayList<>();
        //lists to hold yogaDescription
        List<String> yogaDescriptionList = new ArrayList<>();
        //lists to hold image path
        List<String> yogaImagePath = new ArrayList<>();

        //ignore header in csv file
        String[] yogaDetail = reader.readNext();

            //read in column 0,1 and 2 in csv file
            while ((yogaDetail = reader.readNext()) != null) {
                //add information into appropriate list
                yogaNameList.add(yogaDetail[0]);
                yogaDescriptionList.add(yogaDetail[1]);
                yogaImagePath.add(yogaDetail[2]);


        }

        for (int i=0; i<yogaNameList.size(); i++) {
            //add yogaName, yogaDescription and yogaImagePath into yogaPose object
            YogaPose yogaPose = new YogaPose(yogaNameList.get(i),
yogaDescriptionList.get(i), yogaImagePath.get(i));
            //insert them yogaPose object into linnkedList
            linkedList.insertFirst(yogaPose);

        }




    }
```

VIII. Conclusion

I have learnt that one of the most important step to build a successful application is the initial design process. This design will help connect GUI and the backbend in an efficient way and speed up the coding process. I would like to further complete and improve this project in the future. Specifically, here are features I will implement.

- Implement ringtone to remind users' to stretch
- Change linkedlist into redblacktree to reduce search time from O(n) to O(logn)
- Create a database
- Implement pages for GUI
- Implement IHart to recognize users' face
- Implement calendar to track user's progress
- Develop a website for the application

Reference

Schulte, Brigid. Health experts have figured out how much time you should sit each day. (n.d.). Retrieved April 26, 2016, from https://www.washingtonpost.com/news/wonk/wp/2015/06/02/medical-researchers-have-figured-out-how-much-time-is-okay-to-spend-sitting-each-day/