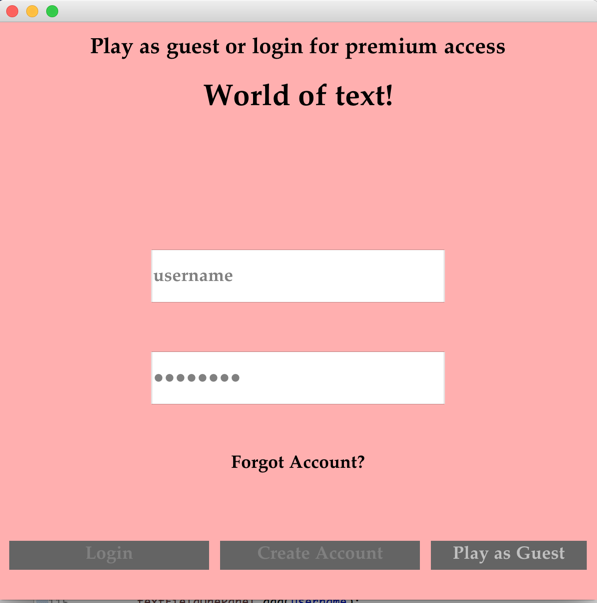
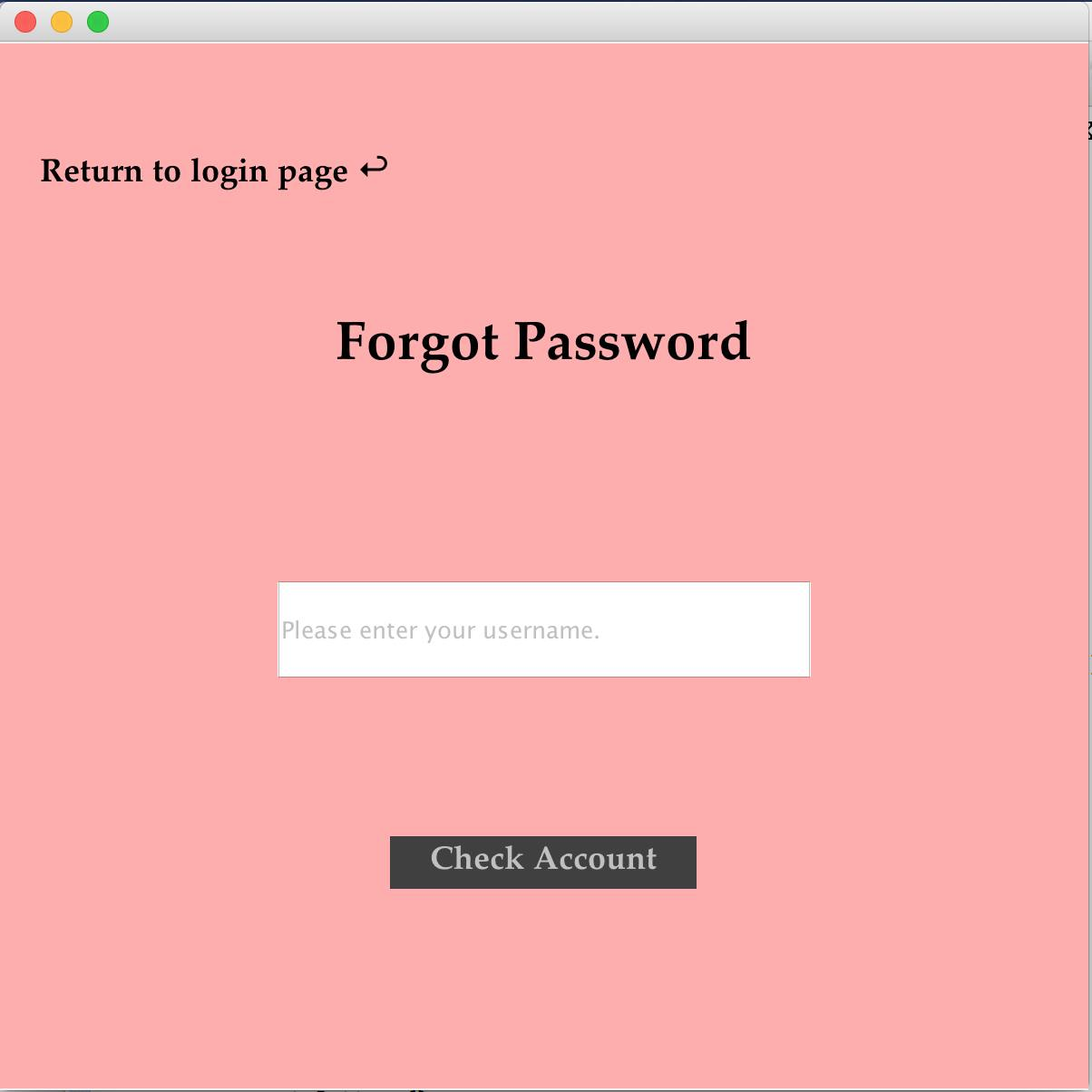
No significant changes in the requirements and specifications documents were made.

**Project Details**

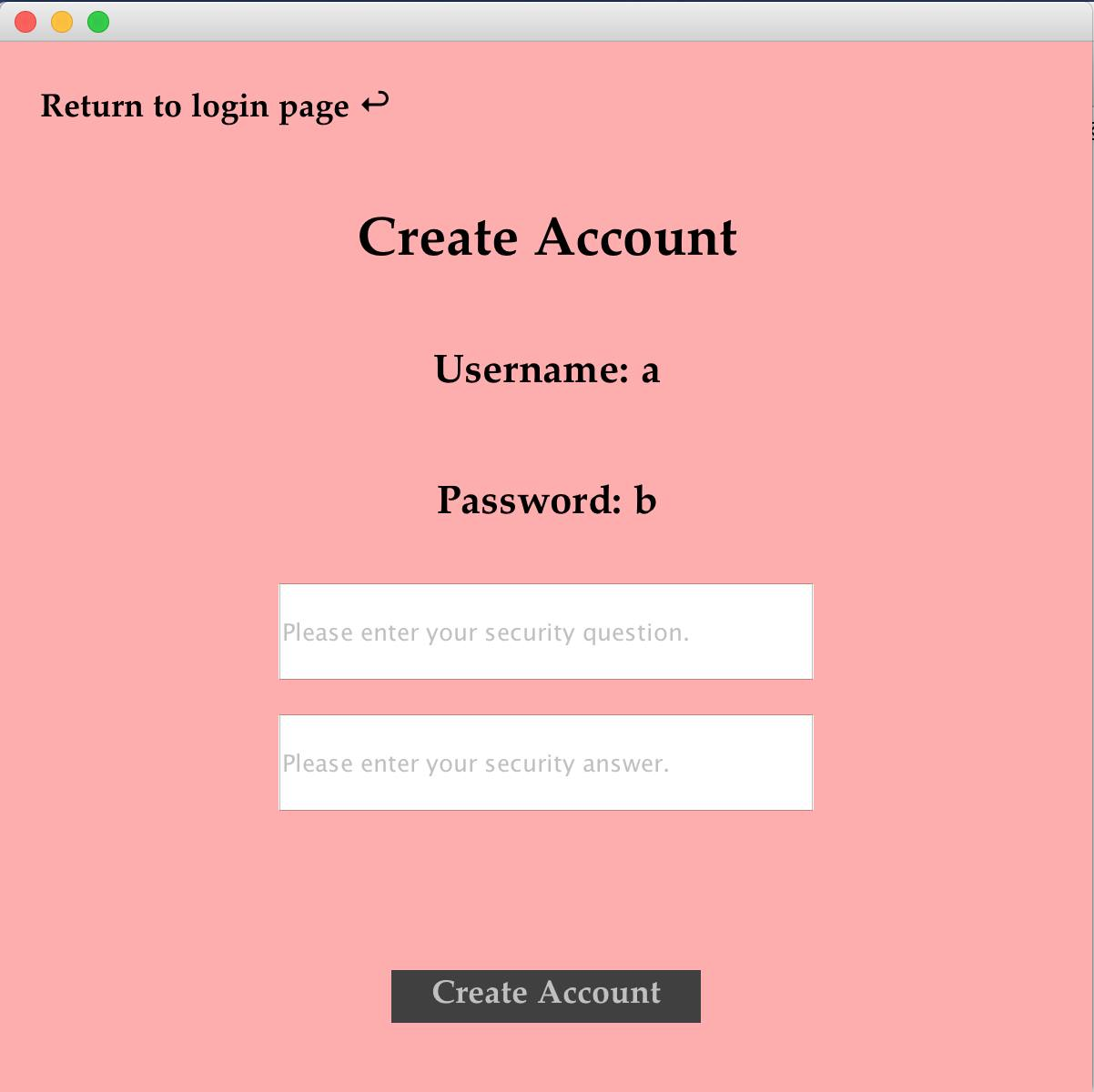
* User Facing
  + Webapp (Primary Work Area) (~18 hrs)
    - Will require:
      * Construction of a web server (in Backbone/Angular/React?) (~5 hrs)
      * UI Implementation in JS (~ 7 hrs)
      * AJAX calls from front-end to server (~6 hrs)
      * Migration of pre-existing Java code to JS (~4 hrs)
    - Our webdev platform will come after we have delivered a working version within Java for all users; once we are able to build our program as intended in Java, we will move to display it via a webpage.
  + Login Screen (3 hours)



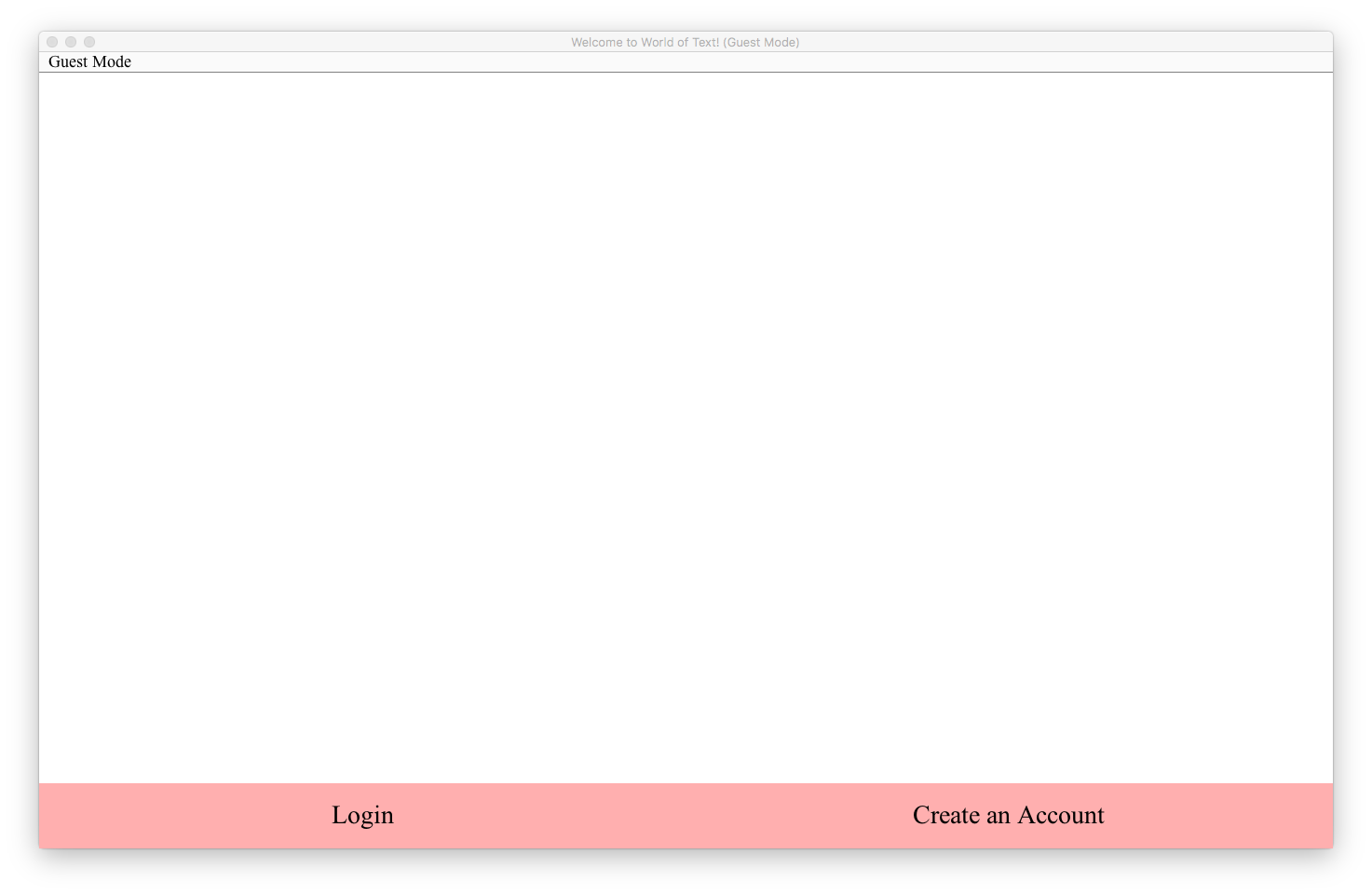
* + - Login option
      * JTextField for Username, JPasswordField for Password
    - Create Account button
      * User types in username and password
      * When button is pressed, check if username is already created
    - Forgot Password Button
      * Takes user to forgot password screen
      * Ask user for username and when submit button is clicked, look for a matching username and show password
      * If username is not found, a JLabel should say username is not found
  + Forgot Password Window (3 hours)



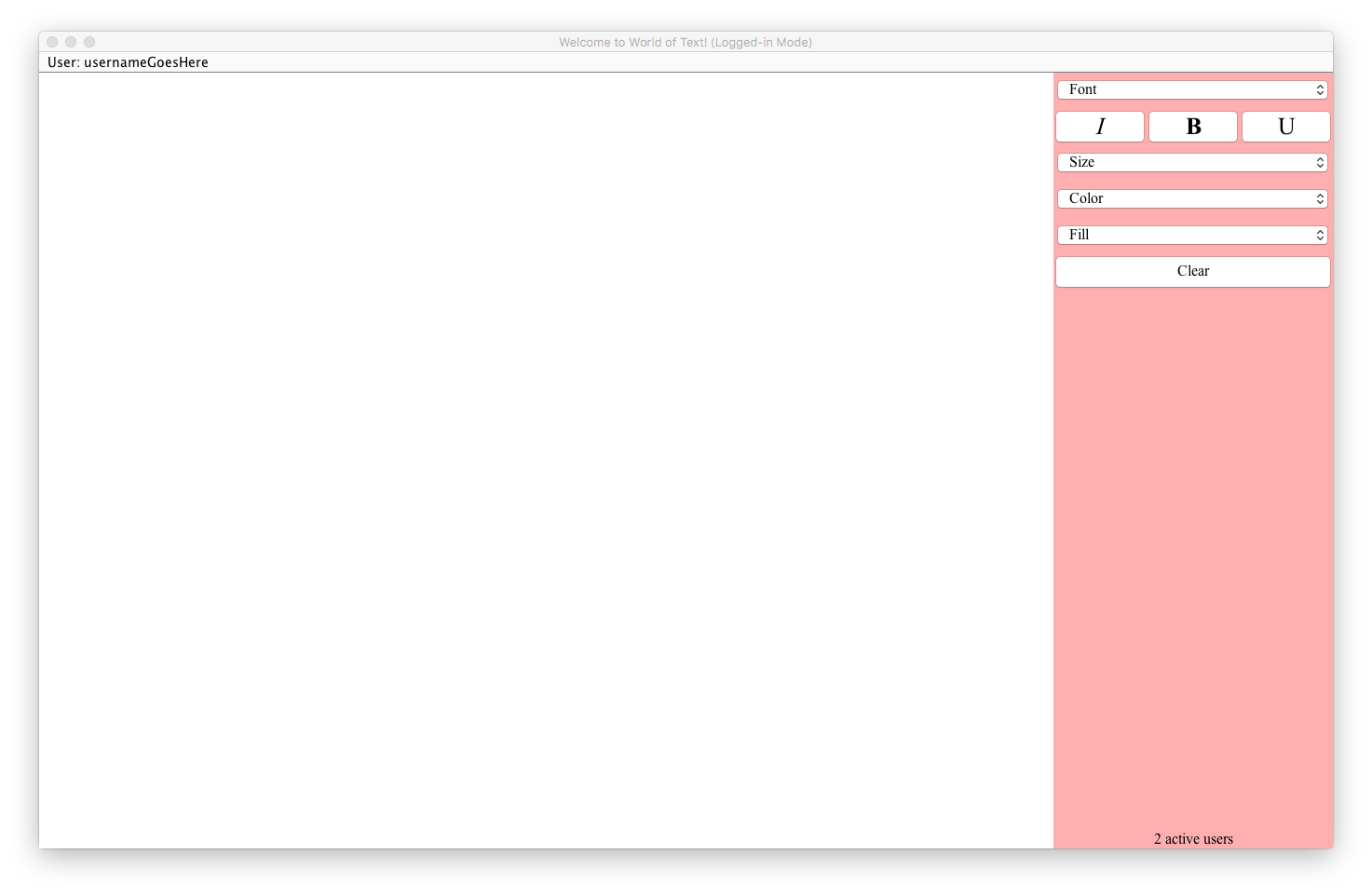
* + - User must enter a username in the database
    - User must correctly answer their security question
  + Create Account Window (3 hour)



* + - Must input username, password
    - Inputting security question/answer is in a separate window
  + Editing Suites (~9 hours)
    - “Editing Suite” Class that contains authenticated and guest mode functionality (GUI: ~3 hours)
      * Very similar in design to font-changer created in lecture (Border Layout with lists and radio buttons)
    - Depending on a “guest mode” boolean value, we will make all of the features visible or invisible
      * Boolean variable will be private (not shown on GUI)
      * Value will depend on user’s choice of logging in (Guest vs. Authenticated)
    - Guest Mode Functionality



* + - * Adding / Deleting text using cursor (Disallowing editing of text using keyboard shortcuts) (.5 hours)
      * Collaborative text editor access using multithreading and server/client connections in Java (~2 hours)
    - Authenticated Mode Functionality



* + - * Gains feature to edit text font / size / color (~1 hour)
      * Gains ability to mass delete text (adding keyboard functionality) (~.5 hours)
      * Access to chatroom using multithreading and server/client connections in Java (~2 hours)
* Backend(~15 hrs)
  + Interface with Java server on the backend
    - Using JDBC to connect from application to MySQL database **(~3 hrs)**
  + Store specific features in MySQL database
    - MySQL Database to store user info(~5 hours construction of script/tables/keys and access/write w/ Java)
    - Will store User ID #, Login, Password, Authentication Level, and post up-to-date texts
  + Master document with all changes to be written/accessed by all forms of program
    - Storage in master textfile/Markdown(?) document on server (~7 hours to get it set up/refreshing/mutating with every program)
* Main Window for User Interaction
  + BorderLayout.CENTER shows the main GridLayout of a lot of textfields. So many that a user can click anywhere to be able to start typing text. Similar to Microsoft Excel (without borders).
  + Size of editable space will be fixed, but set to a big enough number (e.g. ~2000x1400 pixels) to prevent space from filling up too quickly. Time permitting, we can change this to be extendable space to completely prevent running out of space.
  + For BorderLayout.WEST/EAST there is an editing suite to change font, size, color.
    - Use StyledDocument to change text while typing OR
    - Let user highlight text then click on a “change font/size/color” button to change it accordingly

**Project Details Explanation**

We have divided up our project into two primary sections: that which faces the user and our backend. In short, we will communicate between the two sources to effectively render our app in real time and modularize our work. There will be one master environment in which our users will operate, and both guest and authenticated users will be able to interact with each other simultaneously.

Our primary deployment environment will be in Java. We eventuallyplan to construct a webapp, where a user will not need to download any files or run a program -- it will all be done on the webpage. Upon accessing the website using any web browser, the user will be taken to a login screen, where they can enter as either a guest user or an authenticated user. Our webdev platform will come after we have delivered a working version within Java for all users; once we are able to build our program as intended in Java, we will move to display it via a web page.

The Login page will have a Text field titled “Login”, and a separate text field titled “Password”. Underneath these, there must be four buttons: login, forgot password, guest, and create account. These will take you to the separate pages: Authenticated page, Forgot password page, Guest Page, and Create Account page.

In the Create Account it will use the username and password entered before hitting the create account button on the login page. If the username already exists in the database, there must be an appropriate error message displayed. Lastly, there will be a labeled text field for a security question, which will be custom made by the user, and then a security answer. Before being able to create an account the user must be alerted that there is no way to retrieve an account without knowing either the security answer or the password. Also, in the upper left there will be a return to main page button.

In the Forgot Password page the user will be prompted to enter their username. Upon verification of the username, the user will be prompted to answer their own security question by Dialogue box. If they get the answer wrong 3 times, the account will be deleted.

The Guest Page will mostly be bare, and deprived of the features of advanced users. You will only be able to use the cursor to edit and delete text - no keyboard functions (i.e. Ctrl + A) will be allowed or changing of font color/size. You can select where the cursor lies on the page with your mouse or with the keyboard arrow keys and change the text to your desired input in real time.

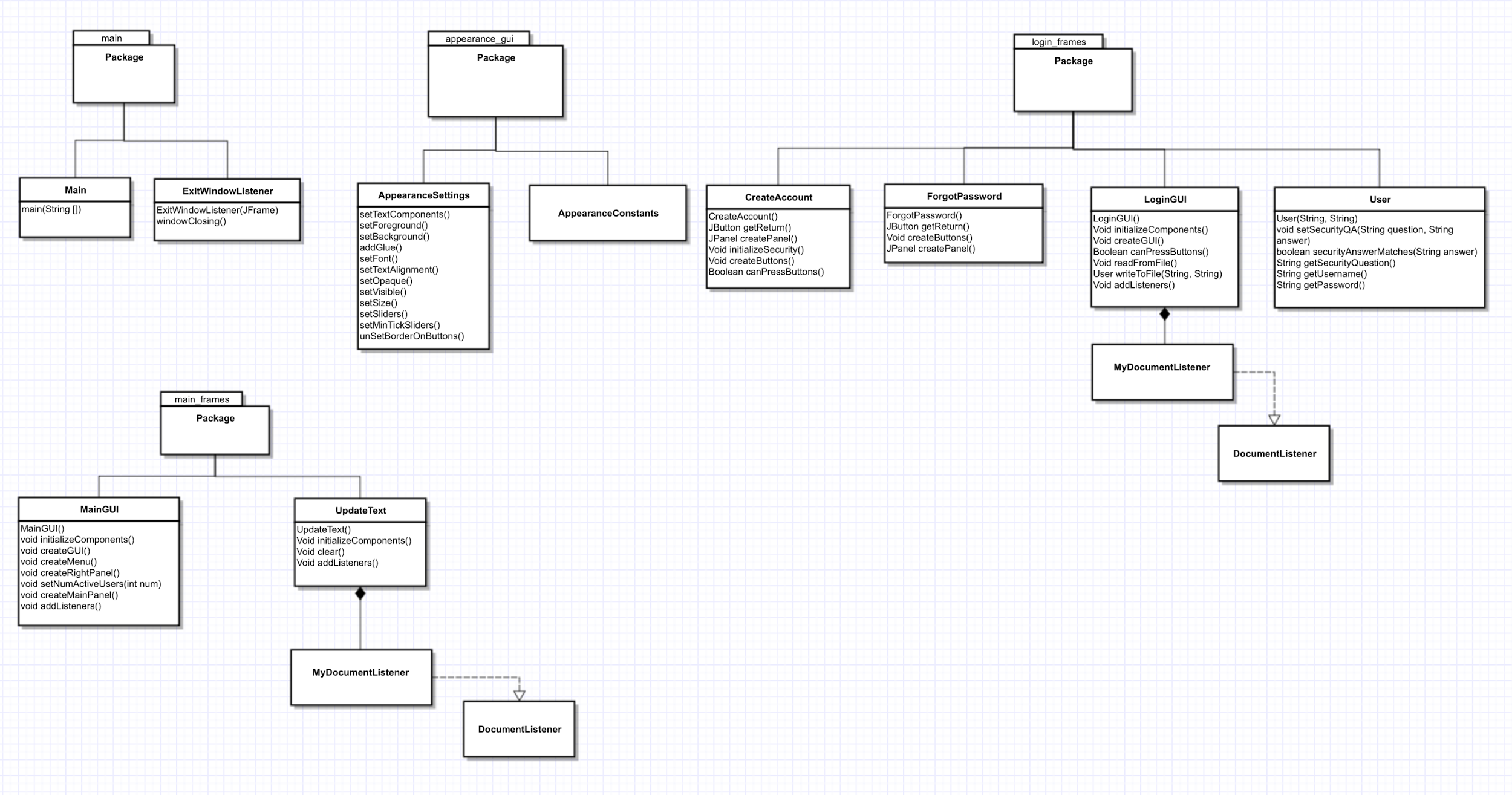
The Authenticated Page will contain all of the Guest Page’s features, with the addition of several premium features through access to an “advanced” editing suite. This editing suite will include mass addition and deletion of text which will be enabled through keyboard shortcuts (i.e. Ctrl + A) and the ability to select large amounts of text with the mouse. It will also include a paint function which will allow an authenticated user to switch from text to brush, including the ability to specify the size and color of the brush. Finally, it will enable the user to insert and resize images in addition to the basic features provided in the guest suite.Theauthenticated user will also have access to a chat room (will be implemented depending on time since there is the feature of multiple users for networking) that includes all other authenticated users who are online.

For main window for gameplay, in addition to using grid layout initialized with a bunch of text fields and a border layout either in the east or west for editing, every time a user edits text, the text will be pushed back to the database in order to have a master key of current texts. So that when another user logs in or plays the game as a guest, the user will also see the current texts.

We will integrate our website with a Java backend that stores all the user information, including usernames and their corresponding passwords. We will be using MySQL Database to store all the user information and features. For guest users, the database will allow them to access a temporary account that has very limited access to the text editing features (most likely only text editing and no font changes) and only anonymous chat (if we implement chat). When the guest user leaves our webpage, the guest will not be able to retrieve or edit the changes he or she has made during that session as all information stored in the temporary guest account will be deleted. For authenticated users, the database will store their security question and answer to ensure account safety as well as their individual chat histories and post histories (both texts and images), allowing them to access and make changes when they log in next time.

Our database will holds each user account information, for example, their name, password and security questions. Besides user accounts, our MySQL database will also store copies of the changes that users have made to front page, including a master version of all the texts and drawings that are being posted to the website, allowing them to be combined and updated accordingly. We plan to interface with our MySQL database with JDBC, allowing us to make our requests natively in Java. Within our webdev framework, we should still be able to use JDBC, but we might require the use of alternative query mechanisms (i.e. AJAX, JQuery, and others). We have also found several services that can transfer Java Swing to HTML5/CSS (webswing), which we are looking at using to help us accomplish this task.

**Class Diagram**



**Package main**

**Class Main**

* Void main(String [])

**Class ExitWindowListener extends WindowAdapter**

* ExitWindowListener(JFrame)
* Void windowClosing()

**Package appearance\_gui**

**Class AppearanceConstants**

* // Colors and icons are set here

**Class AppearanceSettings**

* Void setTextComponents()
* setForeground()
* setBackground()
* addGlue()
* setFont()
* setTextAlignment()
* setOpaque()
* setVisible()
* setSize()
* setSliders()
* setMinTickSliders()
* unSetBorderOnButtons()

**Package login\_frames**

**Class CreateAccount extends JFrame** // window user sees when they click “create account” on login window

* CreateAccount()
* JButton getReturn()
* JPanel createPanel()
* Void initializeSecurity()
* Void createButtons()
* Boolean canPressButtons()

**Class ForgotPassword extends JFrame** // window user sees when they click “forgot password” on login window

* ForgotPassword()
* JButton getReturn()
* Void createButtons()
* JPanel createPanel()

**Class LoginGUI extends JFrame**

// creates login window, the first thing user see when running the program

// opens main window (MainGUI) when user successfully logs in or enters in guest mode

* LoginGUI()
* Void initializeComponents()
* Void createGUI()
* Boolean canPressButtons()
* Void readFromFile()
* User writeToFile(String, String)
* Void addListeners()
  + loginButton listener
  + forgotPassword listener
  + createAccount listener
* Private class MyDocumentListener implements DocumentListener
  + Listener for createAccount and loginButton

**Class User** // stores user information

* User(String, String)
* void setSecurityQA(String question, String answer) // sets security question and answer for this user
* boolean securityAnswerMatches(String answer) // returns true if user’s answer matches one in database
* String getSecurityQuestion() // returns security question
* String getUsername() // returns username
* String getPassword() // returns password

**Package main\_frames**

**Class MainGUI extends JFrame**

// creates the editing window that user sees

// has the customizing functions on the right side of window

// text-fields are on left

// interacts with UpdateText class

* MainGUI()
* void initializeComponents() // Initialize components
* void createGUI() // Set color and appearance
* void createMenu() // menu bar shows username and allows user to log out
* void createRightPanel() // where the customizing buttons go (editing suite)
* void setNumActiveUsers(int num) // sets number of active users to be shown
* void createMainPanel() // Create components and add to MainPanel
* void addListeners()

**Class UpdateText** **extends JPanel**

// whenever user edits text, updates the database and frontend

* UpdateText()
* Void initializeComponents()
* Void clear() // clears / mass deletes text on screen / resets components
* Void addListeners()
* Private inner class MyDocumentListener implements DocumentListener
  + Whenever user insert/removes/changes text, update the text in database

**Hardware Requirements**

* Computer

**Software Requirements**

* MySQL
* Eclipse
* JDBC

**Database Schema (MySQL):**

* Logininfo Table
  + Columns:
    - User ID (INT (11))
    - Login (VARCHAR (50))
    - Password (VARCHAR (50))
    - Security Question (VARCHAR (100))
    - Security Answer (VARCHAR (100))
    - User Permissions (BOOL)
    - Logged In (BOOL)
* ER Diagram:

