Anthony Brignano
John Madsen
Paolo Tozzi

Instructions

No additional system requirements are currently necessary in order to run our web application. This can be run through NetBeans, or any equivalent IDE, using the provided source code. The only requirement is that you must have an active Java Database Connection to the JNDI resource named 'jdbc/FinalProjectDB.' If this is done correctly all necessary tables and data structures will be generated automatically for you, the user. With that being said, upon initialization of the application it's very possible your tables will not contain any data therefore you would, theoretically, be required to register an account before proceeding further into the real views and/or business logic.

Core requirements

- JPA entity beans: UserClass, Menu, SubMenu & MenuItem
    - Relationships:
        - UserClass (1 <---> *) Menu
        - Menu (1 <---> *) SubMenu
        - SubMenu (1 <---> *) MenuItem
    - Create
        - "/adduser" controller in DBController.java – creates users when the register controller is called and all conditional statements are met
        - "/savemenu" controller in MenuController.java – persists menus, their submenus, and their submenus items, to the applicable tables once the user submits their menu.
    - Update
        - "/updatemenu" controller in MenuController.java [/editmenu] – provides the functionality to edit some of the variables associated with their menu, which requires the update of said menu based on the user's changes.
    - Delete
        - "/deletemenu" controller in MenuController.java [/deletemenu] – provides the functionality of being able to completely remove saved menus from the database upon user request submission.
    - Find
        - Selection menu in home.jsp [/home] – Implements find to return all menus associated with the sessions current user in a drop-down menu form.
- Complex flow:
    - "/deletemenu" controller in MenuController.java [/deletemenu] – redirects user back to the "/home" view if they still have more menus left in the database, otherwise it will redirect to the "/menucreation" view and alert the user that no menus were found/
    - "/login" controller in DBController.java [/login] – acts in a similar manner to the "/deletemenu" controller and redirects users to the "/home" view, if they

Anthony Brignano
John Madsen
Paolo Tozzi

currently have menus associated with themselves in the database, otherwise they are sent to the "/menucreation" view.

Validation:

- o checkPass() function [/register] – Custom validation, through a JavaScript function, that alerts users if their original password input is the same as the confirmation password input. The background color of the confirmation password textbox is then changed accordingly, via onblur()- green indicates a match and red indicates an error.
- o Custom, post form submission, user credentials validation message [/login] – When a user's login credentials are unsuccessful the POST method controller "/login" will add an attribute "error" to it's response header and return the user to the login page again. Upon the controller's redirection, back to the login view, the user will be alerted that they have just entered an invalid username or password.
- o checkUsername() function [/register] – pre-registration username validation using AJAX, see below for more information.

- AJAX
  - o checkUsername() function [/register] – As mentioned above, makes use of AJAX and the "/checkusername" controller to alert the user if/when the username they have entered already exists using the JavaScript onkeyup() function.
- Good MVC practices:
  - o Login redirection [/login]
    - If the user credentials are invalid the "/login" controller will redirect the user back to login and prompt the user with an error a variable set in the response.
  - o Register redirection [/register]
    - If the user credentials are invalid the "/login" controller will redirect the user back to login and prompt the user with an error a variable set in the response. Additionally, this simple logic prevents multiple database entries being generated with the same username.
  - o Delete menu redirection [/deletemenu]
    - If the user deletes a menu he/she will be redirected to the "/home" view, assuming they still have menus left in the database associated with their user ID. However, should they have just deleted their only, or last, menu from the database they will be instantly redirected back to the "/menucreation" view to start the process over.
  - o Form Submission in [/editmenu.jsp]
    - The form in the editmenu.jsp page has a modelAttribute of "menu". This model attribute is passed in to the editmenu.jsp page by the bindMenu controller within MenuController.java. Text boxes in the form are then populated with data that are acquired from the passed in the model attribute. After a user has edited their menu in this form, when they click submit the values in each text box are tied to the model attribute "menu".

Anthony Brignano
John Madsen
Paolo Tozzi

This attribute can then be received as a parameter from the next controller /updatemenu in order to update any edited values to the menu within the database.

Screen shots

Fig 1.0 – the main login page, where users are redirected when they first visited the website
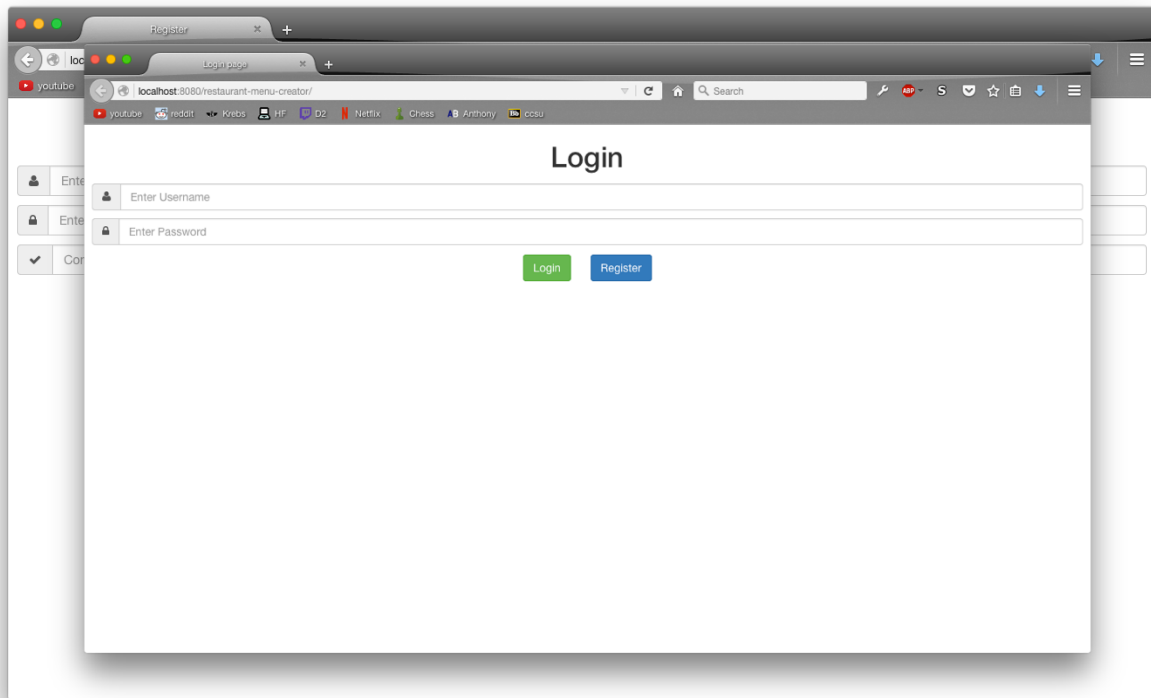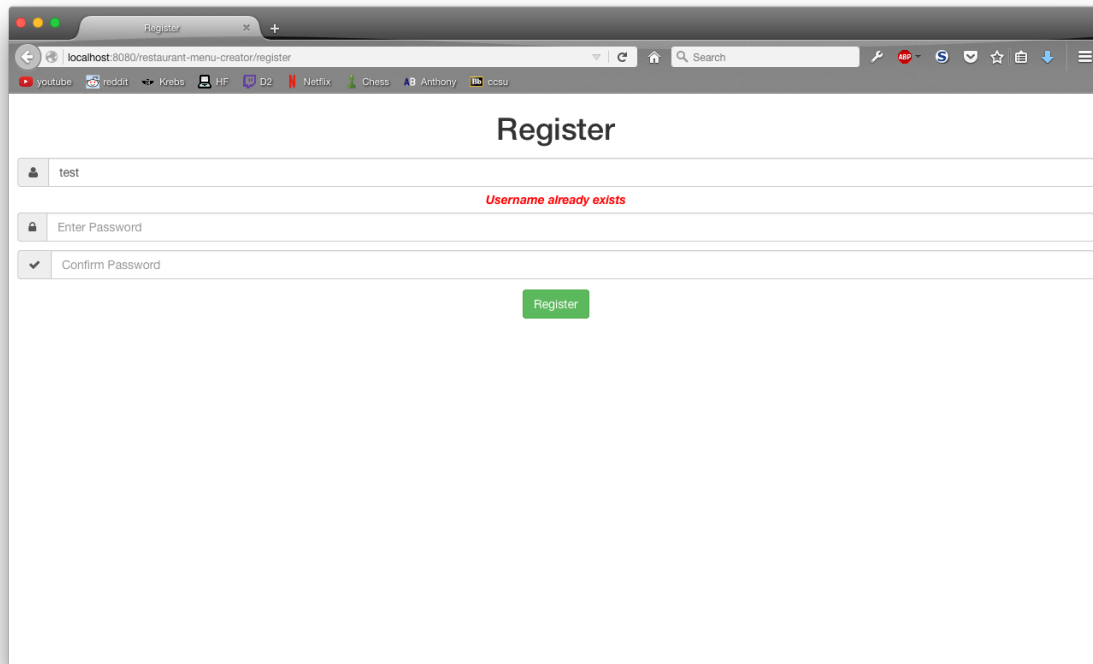

Fig 1.1 – the default register page view, accessed via the 'Register' button

Anthony Brignano
John Madsen
Paolo Tozzi

Fig. 1.2 – AJAX validation of username, alerts the user if the username input is already in use

Anthony Brignano
John Madsen
Paolo Tozzi
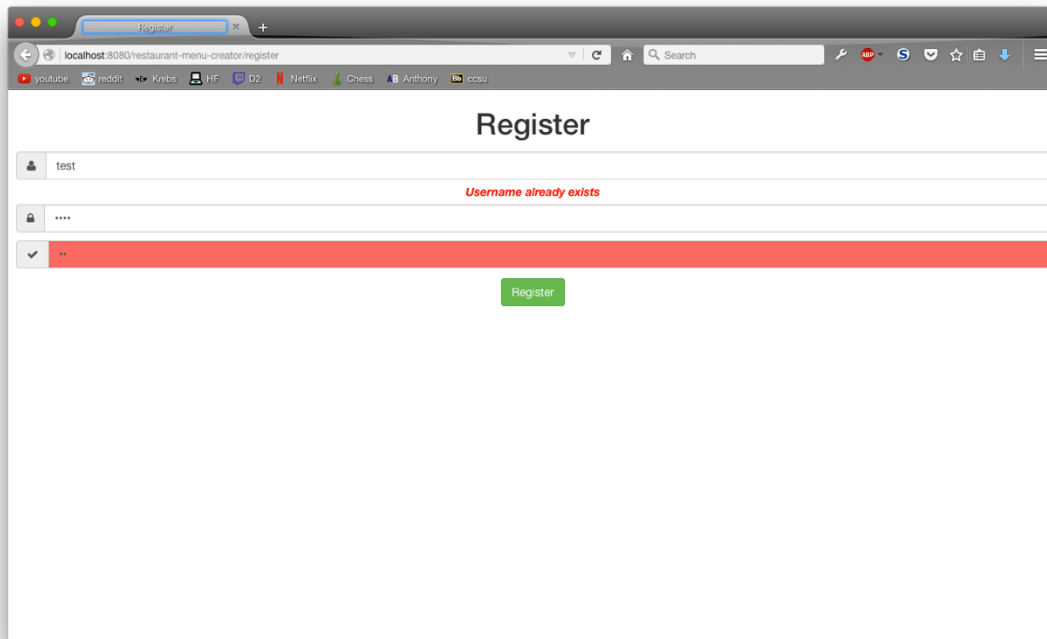
Fig. 1.3 – our custom password validation, to determine if two input passwords match or not
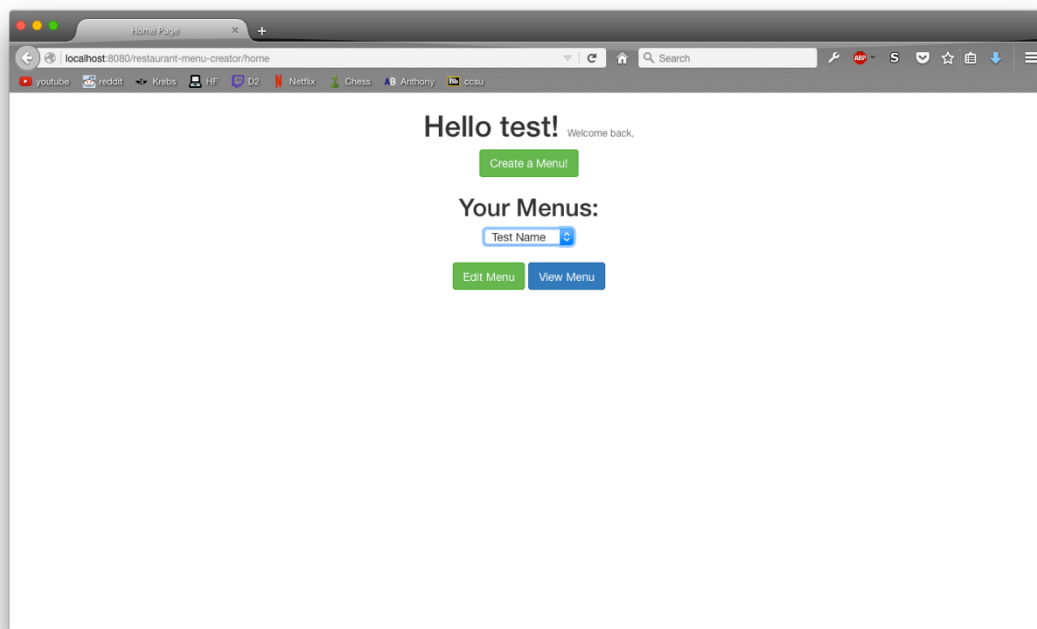

Fig. 1.4 – where a user would be redirected to after logging into the application, if they have menus saved in the database already

Anthony Brignano
John Madsen
Paolo Tozzi

Fig. 1.5 – where a user would be redirected to after logging into the application, if they have no menus saved in the database already
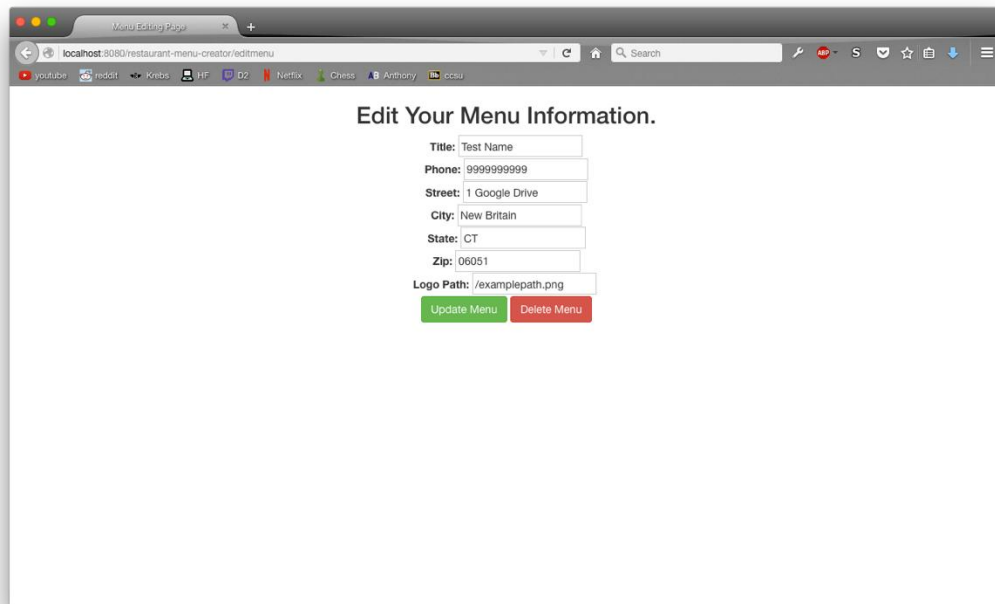
Anthony Brignano
John Madsen
Paolo Tozzi
Fig. 1.6 – standard view menu page

Fig. 1.7 – standard edit menu page