**C Sc 335 Analysis and Design Artifacts for Jukebox**

*Each team complete this form, put it in your project in a folder named documents and push to Github. This will be part of your Iteration 1 grade*

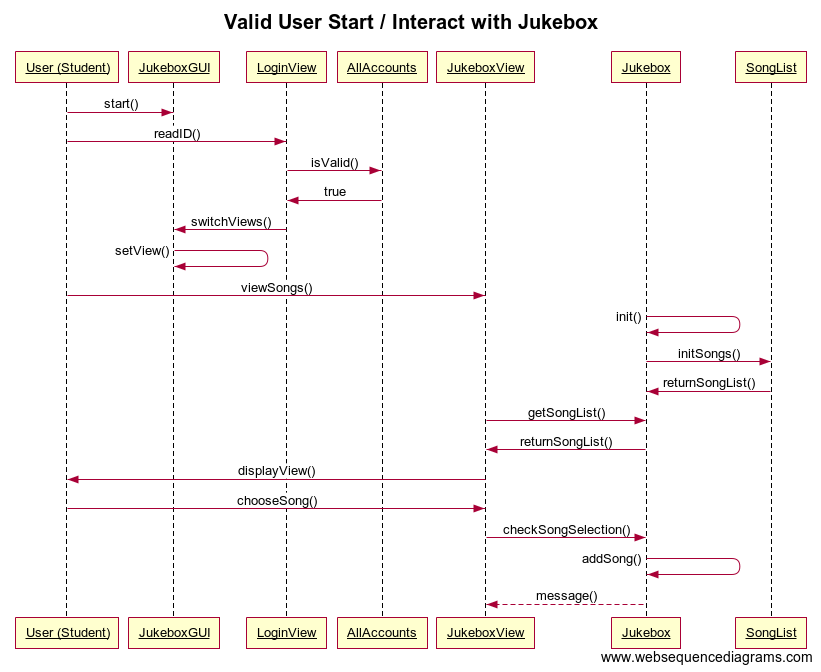
**1) Team Members**: Ashley Mains Victor Gomes

**2) Candidate Objects**

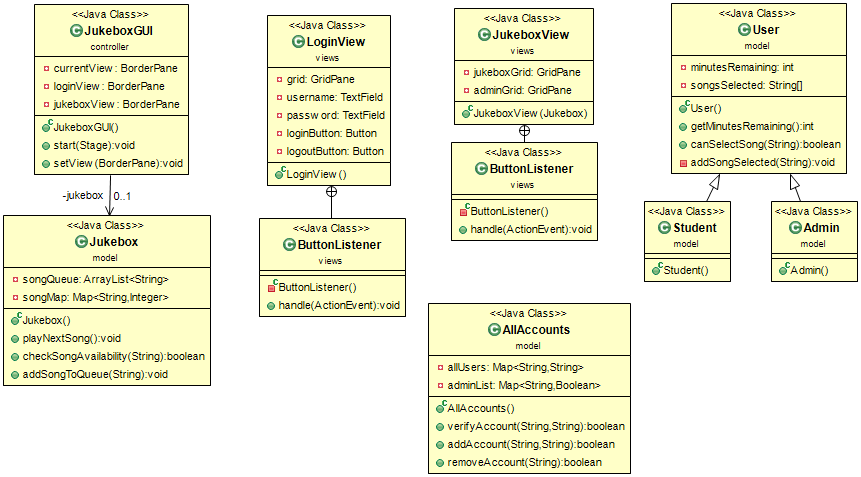
List the most important objects, or an inheritance hierarchy name, and the responsibility of each.

|  |  |
| --- | --- |
| **Candidate Object** | **Responsibility in 1 or 2 sentences** |
| Jukebox | Keeps track of the song queue (queue of songs to be played) and song map (keeps track of the number of times each song has been selected that day). |
| SongList (enum) | Keeps track of basic song information (song name, time). |
| User | Abstract object that provides the basic functionality of an account to access the Jukebox. Keep track of how many songs the user has selected that day and the time the user has remaining. |
| Admin (User) | Adds extra functionality to allow the Admin to add and remove accounts on top of basic song selection. |
| Student (User) | Provide basic song selection and tracking for User. |
| JukeboxView | Provides the user an interactive view of the Jukebox. |
| AllAccounts | Keeps track of all valid Users and passwords and provides functionality to an Admin to add and remove accounts. |
| LoginView | Interactive view for the user to login. |
|  |  |
|  |  |
|  |  |

**3) Sequence Diagram 1:** Write a UML Sequence Diagram should show the most important scenario you can think of. Your sequence diagram should show most of your candidate objects you listed above and how they communicate with each other.



**4) Class Diagram:** Write a UML Class Diagram that shows all of your candidate objects from above. Show any relationships between them the classes such as inheritance or interface implementation. Draw general associations such as dependency or aggregation. Label some to help explain things. Add any multiplicity adornments that seem appropriate. Use notes to explain things if you feel it will help. Each UML class must show the class name. For full credit, each class must have an average of at least one attribute per class. There must be an average of about 1.5 methods per class.

**

**5) Estimate and Assign Tasks** For each Iteration 1 task, estimate its difficulty using the numbers 1, 2, 3, 5, or 8. These are points that represent the relative complexity of the task. Mark 8 for the most difficult and/or time consuming and 1 for what appears to be the easiest. Indicate which person (s) will complete the task before iteration 1 due date.

|  |  |  |
| --- | --- | --- |
| **Points** | **Who will**  **complete this?** | **Task** |
| 1-2 | Both | Your GitHub repo has a completed copy of this analysis and design document in a folder named **doc** that represents a serious effort to analyze and design Jukebox |
| 3 | A – Victor | Users can log in and log out |
| 5 | B – Ashley | Songs can be ***selected*** up to a maximum of 3 times per calendar date. Use the time the song is added to the song queue, not when the song ends |
| 5 | B – Ashley | Any valid user can select a maximum of 3 songs per calendar date. Use the time the song is added to the queue, not when the song ends. Reset time occurs at midnight, so users could have three new plays tomorrow.  A song that has played 3 times today, could be played 3 times tomorrow. |
| 2-3 | A – Victor | The administrator can add and remove accounts |
| 3-5 | A – Victor | Songs can be played in FIFO order, like a Jukebox, the first one plays on order. |
| 8 | Both | Complete a functional spike to determine the interactions are actually working. JukeboxStartGUI.java in package controller is an event-driven program with a graphical user interface to affirm the functionality all Iteration 1 tasks have been completed and are working correctly.  We will use this to test your code for the first 100 points of Jukebox. |

**6) Estimate and Assign Tasks** For each Iteration 2 task, estimate its difficulty using the numbers 1, 2, 3, 5, or 8.  These are points that represent the relative complexity of the task.  Mark 8 for the most difficult and/or time consuming and 1 for what appears to be the easiest. Indicate which person (s) will complete the task before iteration 2 due date.

|  |  |  |
| --- | --- | --- |
| **Points** | **Who will  complete this?** | **Task** |
| 3-5 | Victor | Build a TableView that allows users to select a Song as it displays the number of times the song played today with its title, artist, and play time. Should be sortable by column |
| 3 | Victor | Use a ListView to show the current queue of songs playing (or to be played). |
| 3-5 | Ashley | Implement system persistence (reading and writing files) |
| ? | Both | Write Javadocs for all classes and methods. |
|  |  |  |