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

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

**1) Team Members**: Alexis Tinoco David Wang

**2) Candidate Objects**

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

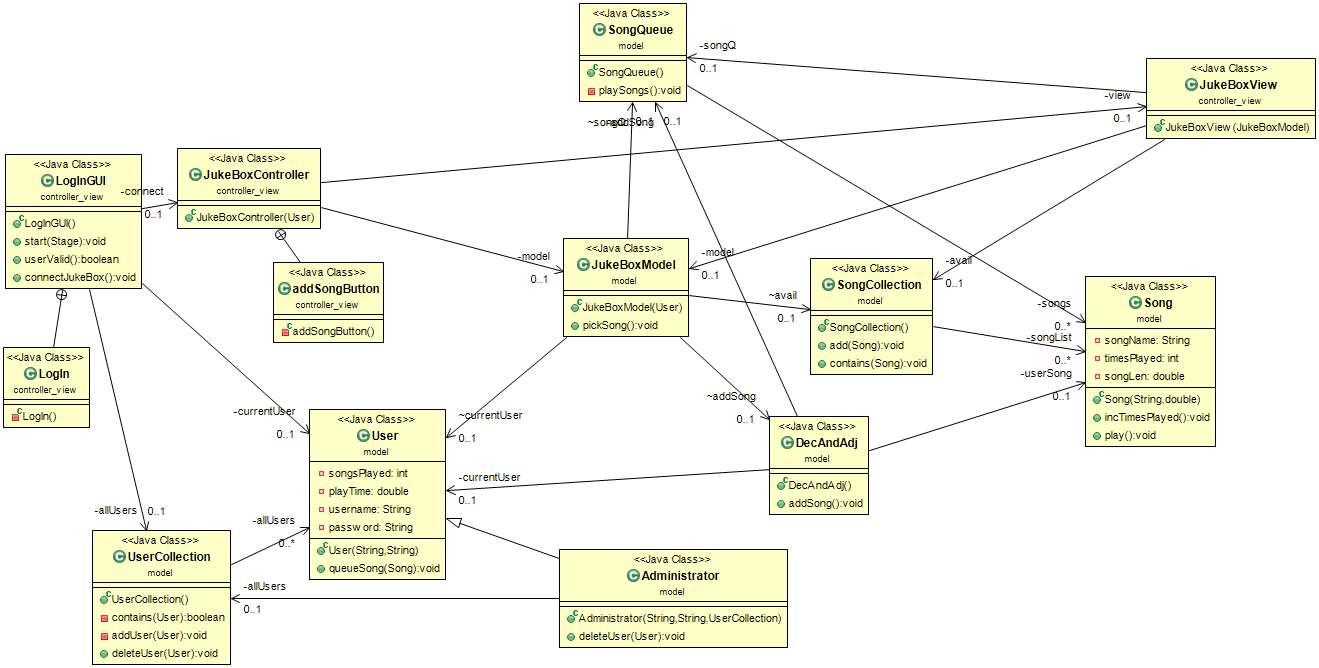
|  |  |
| --- | --- |
| **Candidate Object** | **Responsibility in 1 or 2 sentences** |
| User | Each user can log into the Jukebox (if validated by LogInGUI). They can pick up  to 3 songs and play for 1500 minutes. This keeps track of both fields. |
| Administrator | Extends user. This user has the additional option of adding or removing accounts. |
| UserCollection | This collection keeps track of all valid users. |
| Card Reader | Reads card and instantiates a user from card information.  May not be a real thing, at least for iteration 1. |
| Song | This represents a song and can be played. It also keeps track of how many times the  song has been played. |
| SongCollection | This collection keeps track of all available songs. |
| SongQueue | This queues up songs to be played in FIFO order. |
| DeciderAndAdjuster | This object, as the name suggests, decides whether a song can be added to the  song queue. It reads and adjust various fields that can affect this decision. |
| JukeboxView | The view for the Jukebox. It extends BorderPane and “shows” the model.  Events interact with the model. |
| JukeboxModel | The model for the jukebox. It models how the user picks songs in the Jukebox. |
| JukeboxController | The controller for the jukebox. It coordinates interactions with the model and the view. |
| LogInGUI | Debatably, we could use a second application to handle a user trying to login instead  of including it into the jukebox proper. |

**3) 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.

**Optional:**  *Instead of using pencil, paper and scanning to insert an image, use a new Eclipse Project to add classes, methods, and instance variables. No implementation needed. Then use Object Aid UML tool to reverse engineer your code to get the UML class diagram.*

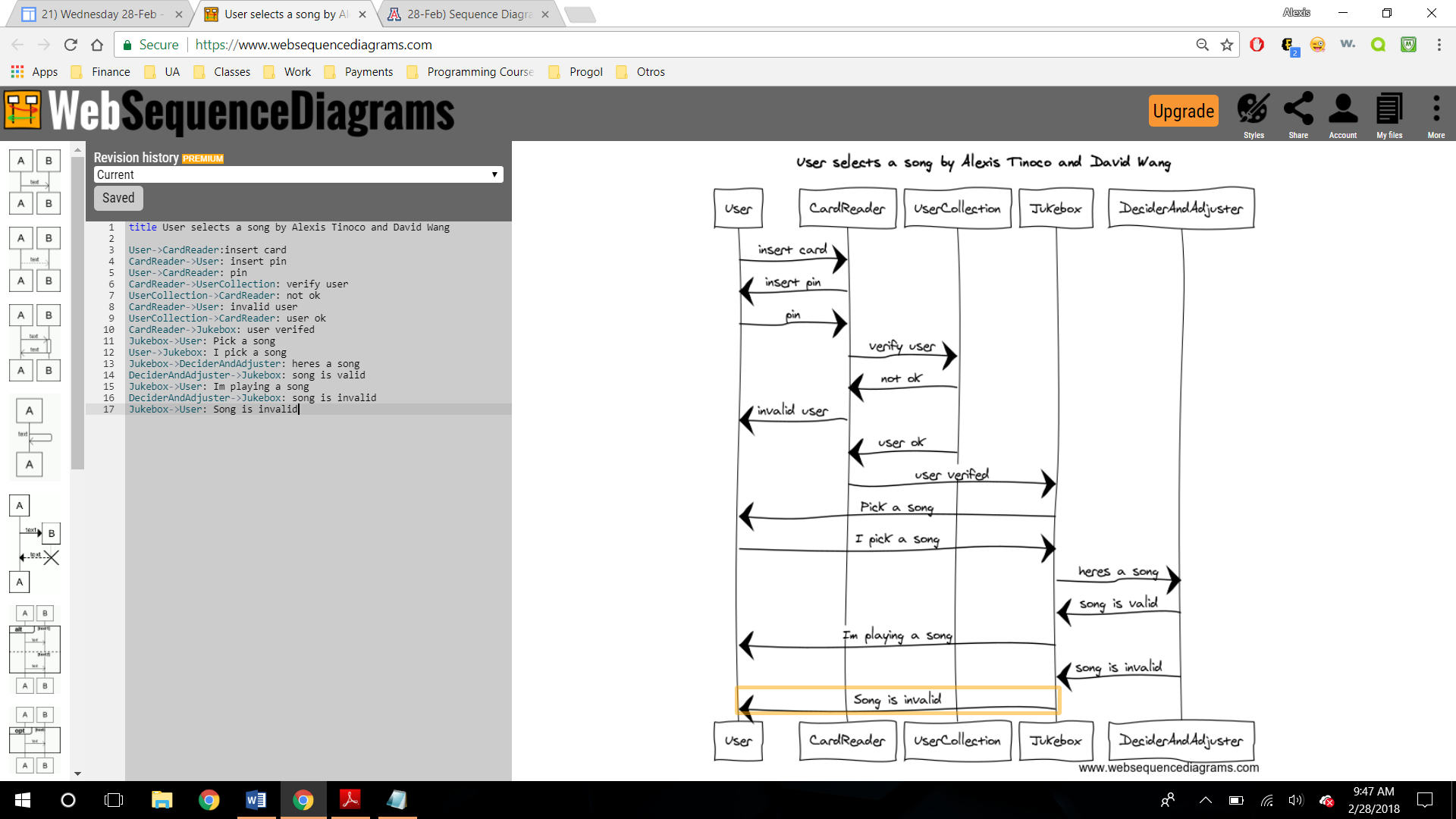
*Installation of Eclipse Plugin in case you don't have it:*

<http://www.objectaid.com/installation>

*Export and Image and copy and paste it here*

**4) Sequence Diagram:** 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.

**Recommended:** *Use the Sequence Diagram editor found at*

 <https://www.websequencediagrams.com/>