Project Part 2

Team

Steven Jace Conflenti, Jennifer Michael, and Cameron Taylor

Title

Studysaurus

Project Summary

A study tool which acts as a fast-paced quiz. User-centered key terms will fall down the screen as asteroids, threatening to destroy the dinosaur population at the bottom of the screen. Students must provide the corresponding value to the key term in order to save the dinosaurs.

Project Requirements

We do not have any business requirements.

User Requirements			
ID	**Requirement**	**Topic Area**	**Priority**
UR-01	As a user, I need to be able to select a default set of terms and then play the game.	Default Sets	High
UR-02	As a user, I need to be able to create by own set of terms and values to be be tested against in the game.	Customized Sets	High
UR-03	As a user, I need to be able to import a set of terms and values from a text file to be tested against in the game.	Import / Export	Low
UR-04	As a user, I need to be able to edit or delete a set of key terms and values after initially creating it.	Customized Sets	Medium
UR-05	As a user, I need to be able to view my scores for different sets of test terms.	Scoring	Medium
UR-06	As a user, I need to be able to export a set of key terms and values to a text file.	Import / Export	Low
UR-07	As a user, I need to be able to type the corresponding value of a key term in the game in	Scoring	High

	order to earn points.		
UR-08	As a user, I can specify the difficulty level of the game.	Gameplay	Low
UR-09	As a user, I want to be able to see how I've improved over time while playing with my sets by viewing a scoreboard.	Scoring	Low
UR-10	As a user, I want to be able to quit the game from the Home page.		Medium

Functional Requirements

ID	Requirement	Topic Area	Priority
FR-01	When a user types the value of a key term displayed in the game, the score needs to be updated and the asteroid needs to be destroyed.	Scoring	Medium
FR-02	The difficulty level must be proportional to the amount of time the user has to type the value for a given term.	Gameplay	Low
FR-03	When a term falls off the screen, a dinosaur must disappear from the screen as well.	Gameplay	Medium
FR-04	When all dinosaurs are no longer on the bottom of the screen, the game must end and display the score to the user.	Gameplay	High
FR-05	When all terms have been asked and there is at least one dinosaur remaining, the game must end and display the score to the user.	Gameplay	High
FR-05	When a user enters the value of a key term it needs to be validated against the correct value, disregarding differences in punctuation such as apostrophes, dashes, commas, and periods as well as disregarding differences in capitalization.	Scoring	Low

Non-functional Requirements

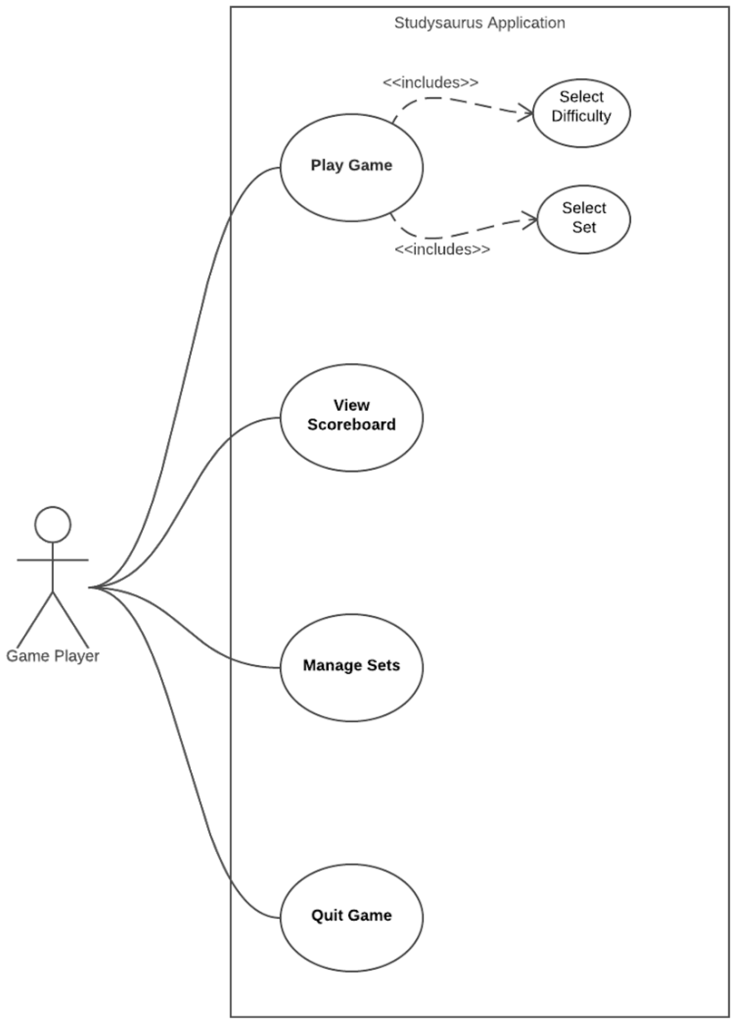
ID	Requirement	Topic Area	Priority
NFR-01	When a user selects a set to play with, all of the keys and values from the set must be loaded in from the database.	Performance	Low
NFR-02	Game play will be easy to learn because	Usability	Low

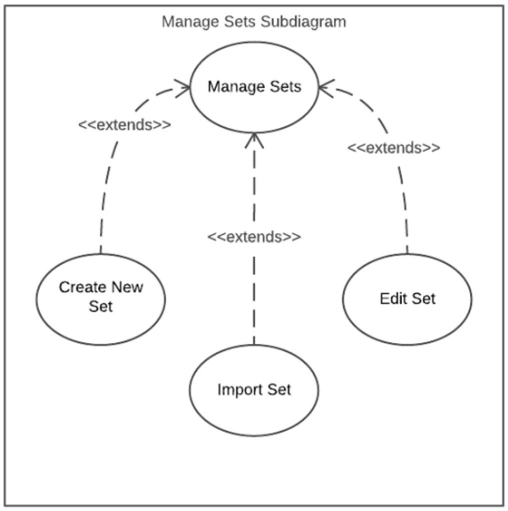
	instructions will be available on the Home screen.		
NFR-03	The asteroid containing a key term must disappear within one second after it’s value is correctly entered by the user.	Performance	Medium
NFR-04	The game must run in a Linux environment.	Platform	High
NFR-05	Terms and values entered by the user must be checked to ensure user input doesn’t compromise database integrity.	Security	High

Use Case Diagram

Basic Use Case Diagram

Jennifer | October 12, 2016





Use Case Documents

Use Case ID:	UC-01
Use Case Name:	Play with default set
Description:	Player can select a default set and play the game with the keys and terms in that set.

Actors:	User		
Pre-conditions:	None		
Post-conditions:	The key terms in the asteroids on the screen during game play are keys from the default set selected and the asteroids are destroyed when the user enters the corresponding value term of a key term on the screen.		
Frequency of Use:	Frequent- 1 out of every 10 plays		
Flow of Events:		**Actor Action**	**System Response**
	1	User clicks 'Play Game' button	Open Game Options page with displayed sets and levels.
	2	Select desired set from 'Default Sets' choices and select difficulty level.	Stores name of the set and difficulty level
	3	User clicks 'Play Game' button.	Retrieve set pairs from database, store pairs, instantiate asteroids with term/value pairs from the set. Load Game Play page and display term and falling asteroid.
	4.0	Types in correct term given the displayed value.	Validates answer is correct, removes an asteroid from the screen, and increments the score. Then displays a new value and asteroid.
Variations:	**4.1**	Types in incorrect term given the displayed value or fails to enter term before time's up.	Validates answer is incorrect or empty and removes a dinosaur from the screen. Then displays a new value and asteroid.
	4.2	Types correct term with one term left.	Validates answer is correct, removes asteroid from the screen, and increments the score. Then ends the game and displays the score.

	4.3	Types incorrect term or fails to enter term before time’s up with one dinosaur left.	Validates answer is incorrect or empty and removes dinosaur from the screen. Then ends game and displays score.
Exceptions:	If retrieving the chosen set from the database fails, an error will be displayed to the user.		
Developer Notes:			

Use Case ID:	UC-02
Use Case Name:	Play with customized set
Description:	User can select a set to play the game with from a list of set that they entered into the system.

Actors:	User		
Pre-conditions:	User has already entered in a customized set of key-value terms using options in Manage Sets (either Create Set or Import Set).		
Post-conditions:	The key terms in the asteroids on the screen during game play are keys from the customized set selected and the asteroids are destroyed when the user enters the corresponding value term of a key term on the screen.		
Frequency of Use:	Frequent- 1 out of every 3 plays		
Flow of Events:		**Actor Action**	**System Response**
	1	User clicks ‘Play Game’ button	‘Select a Set’ options are displayed to the user: choice of default and custom.
	2	Select desired set from ‘Custom Sets’ choices and select difficulty level.	Stores name of the set and difficulty level
	3	User clicks ‘Play Game’ button.	Retrieve set pairs from database, store pairs, instantiate asteroids

			with term/value pairs from the set. Load Game Play page and display term and falling asteroid.
	4	Types in correct term given the displayed value.	Validates answer is correct, removes an asteroid from the screen, and increments the score. Then displays a new value and asteroid.
Variations:	Same variations as UC-02-- game play did not change, only the fact that a custom set was chosen.		
Exceptions:	If retrieving the chosen set from the database fails, an error will be displayed to the user.		
Developer Notes:	The names of the Custom Sets must be loaded in from the database prior to reaching this use case.		

Use Case ID:	UC-03
Use Case Name:	Create New Set
Description:	User can create a new set which can be used to play the game.

Actors:	User		
Pre-conditions:	N/A		
Post-conditions:	New set is created and can be selected from the “Play Game” screen.		
Frequency of Use:	Frequently by users.		
Flow of Events:		**Actor Action**	**System Response**

	1	Click ‘Manage Sets’	Display ‘Manage Sets’ page
	2	Click ‘Create New Set’	Display ‘Create Set’ page
	3	Enter set name	
	4	Enter term	
	5	Enter value	
	6	Click ‘Save Pair’	Check term and value to ensure database integrity. Save pair to database under the designated set. Clear term and value text fields
	7	Click ‘Done’	Display ‘Home’ page
Variations:			
Exceptions:	If a term or value compromises the database’s integrity, the user will be shown an error message and that pair won’t be inserted. Also, if saving a pair to the database fails the user will be shown an error message.		
Developer Notes:			

Use Case ID:	UC-04
Use Case Name:	Import Set
Description:	User can create import a set which can be used to play the game.

Actors:	User
Pre-conditions:	N/A
Post-conditions:	Set is imported and can be selected from the “Play Game” screen.

Frequency of Use:	Frequently by Game Players that study in groups; otherwise, infrequently.		
Flow of Events:		**Actor Action**	**System Response**
	1	Click on “Manage Sets” from the 'Home' page	Display ‘Manage Sets’ page
	2	Click on “Import Set”	Display ‘Import Set’ page
	3	Select file containing set	
	4	Enter set name	
	5	Click “Import”	Open file, parse by line, and create a term/value from each line. Save all term/value pairs as a set in the database. Display the 'Success' page
	6	Click “Done”	Display ‘Home’ page
Variations:			
Exceptions:	If any of the following are true a ‘Failure’ screen will be displayed instead of the ‘Success’ page: ● the file selected by the user is invalid (can’t be opened, is of the wrong type, or is corrupt) ● a term or value would compromise the database’s integrity ● saving the set to the database fails		
Developer Notes:			

Use Case ID:	UC-05
Use Case Name:	Edit Current Set
Description:	User can edit the non-default sets.

Actors:	User		
Pre-conditions:	Game must have a non-default set.		
Post-conditions:	Set will have updated term and value selections.		
Frequency of Use:	Frequently by users.		
Flow of Events:		**Actor Action**	**System Response**
	1	Click on “Manage Sets” from the homepage.	“Manage Sets” page is displayed.
	2	Click on “Edit Set” from the “Manage Sets” page.	“Select Set” page is displayed. Custom set names are pulled from database into drop-down box.
	3	Click on the set to be edited from drop-down box.	
	4	Press “Next” button.	“Edit Set” page is displayed. Selected set’s pairs are loaded into drop-down box.
	5	Select old pair using drop-down.	Pair is loaded into text boxes.
	6	Edit old pair using text boxes. Press “Save” button.	Set deletes old pair and adds new pair to its array.
	7	Click “Done”.	Updates the database with new Set. Homepage is shown.
Variations:	Alternatively, the user could opt to do the below instead of steps 5 and 6 if deciding to either add or delete a pair instead of modify an existing pair. The below steps as well as steps 5 and 6 combined can happen as often as the user desire before clicking “Done” in steps 7 and 8.		
		Add new pair using text boxes. Press “Save” button.	Set adds new pair to its array.
		Select old pair using drop-down. Press “Delete” button	Set deletes old pair from array.

Exceptions:	If any of the following are true a ‘Failure’ screen will be displayed immediately instead of the ‘Home’ page: ● a term or value would compromise the database’s integrity ● saving the set to the database fails
Developer Notes:	Terms and definitions must be checked to ensure database integrity.

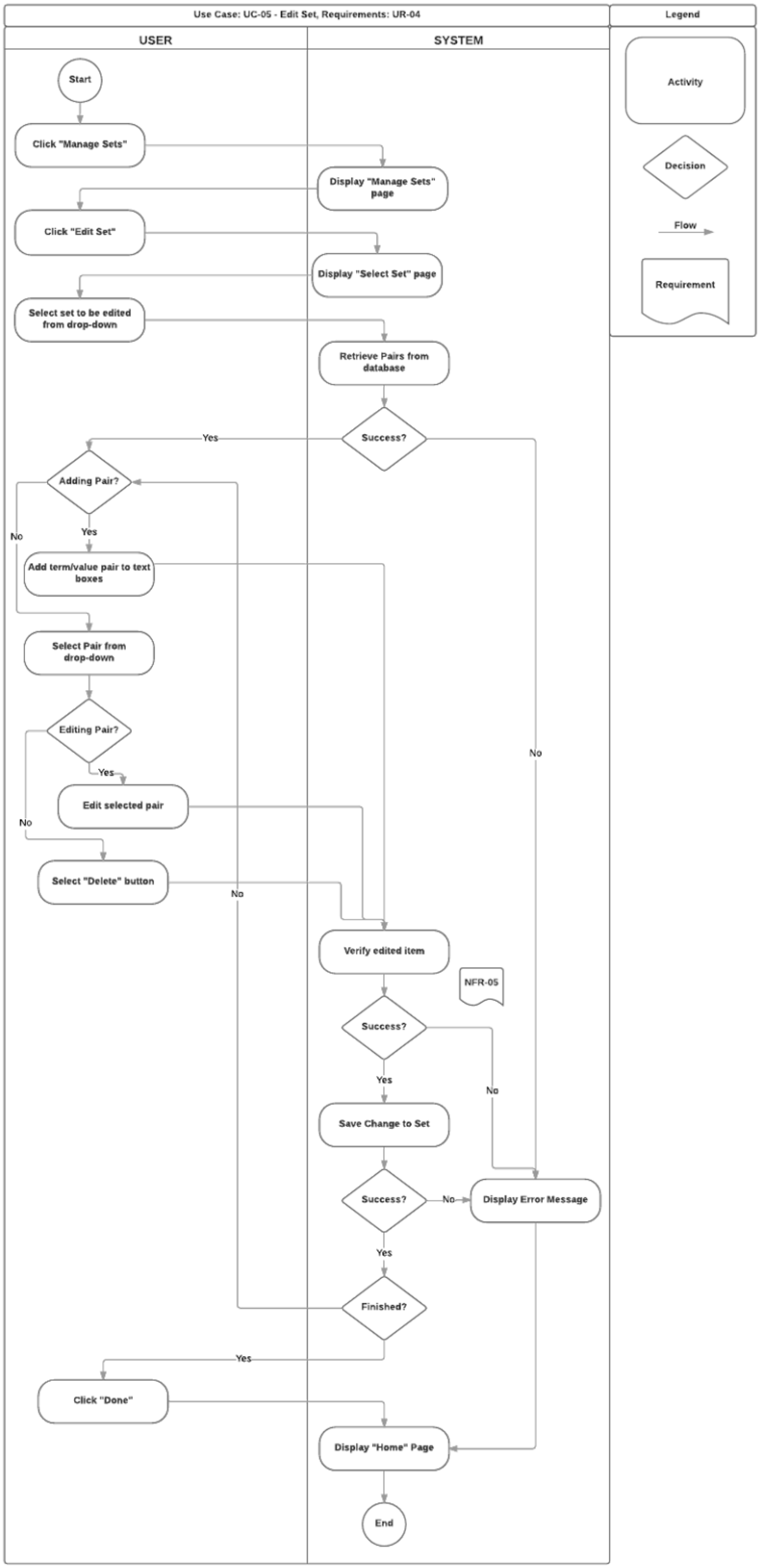
Use Case ID:	UC-06		
Use Case Name:	View Scores		
Description:	User can view the scores recorded by previous games.		
Actors:	User		
Pre-conditions:	Games must have been played for scores to have been recorded.		
Post-conditions:	Previous scores can be viewed by User on screen.		
Frequency of Use:	Rarely by users.		
Flow of Events:		**Actor Action**	**System Response**
	1	Click on “Scoreboard” from the homepage.	“Scoreboard” page is displayed. Scores are loaded from database using DatabaseConnector.
	2	Click “Back”.	The homepage is displayed.
Variations:	N/A		
Exceptions:			
Developer Notes:	Scoreboard should display top 10 scores and the sets they were earned on.		

Use Case ID:	UC-08
Use Case Name:	Quit Game
Description:	User can exit out of the game when desired.
Actors:	User

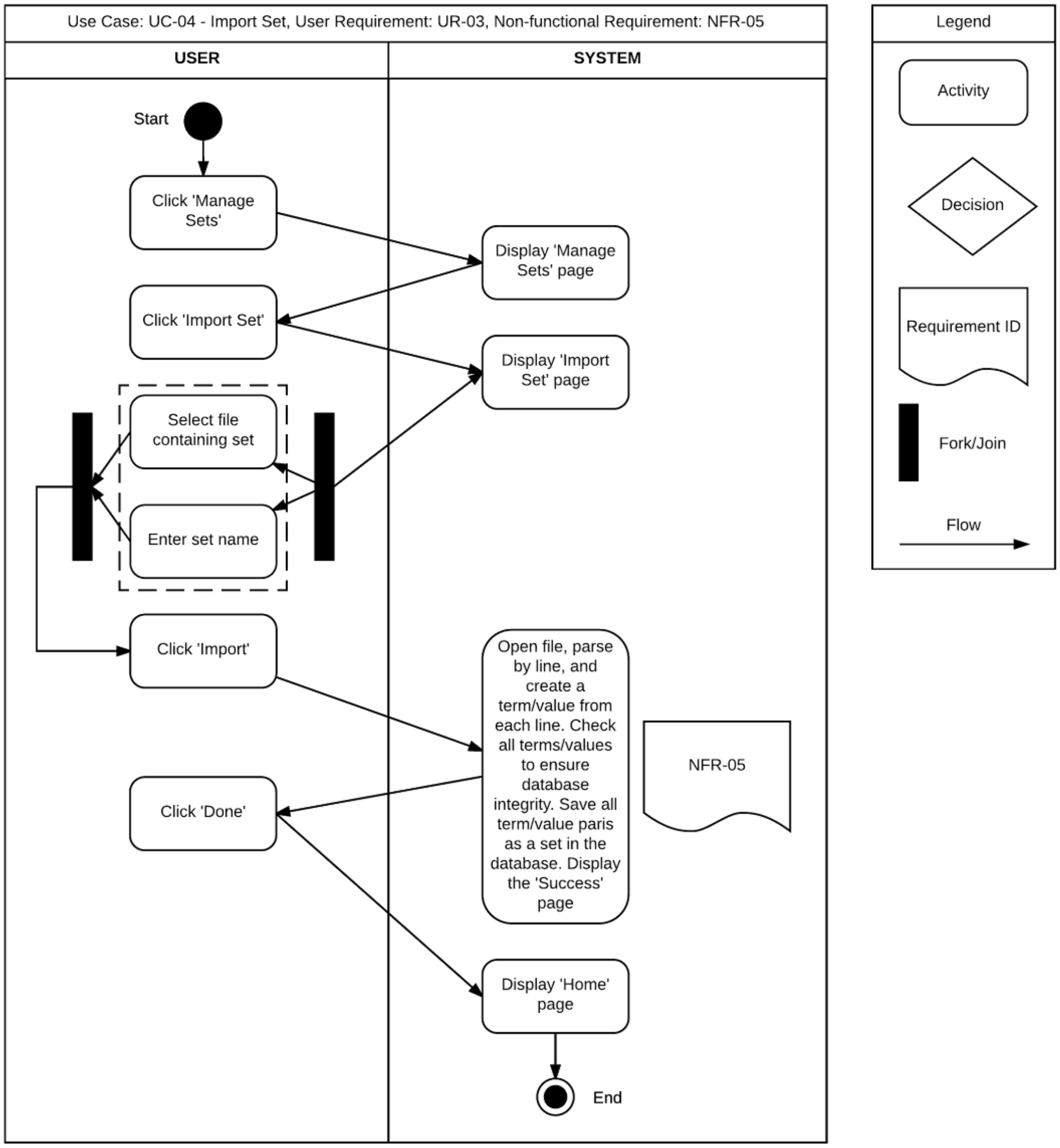
Pre-conditions:	The game is open and user is at the Home page.		
Post-conditions:	The game window is no longer open and the program is no longer running.		
Frequency of Use:	Once per use..		
Flow of Events:		**Actor Action**	**System Response**
	1	Click ‘Exit’ button.	Program terminates
Variations:	N/A		
Exceptions:			
Developer Notes:	Deallocate on close.		

Activity Diagrams

Cameron Taylor



Steven Jace Conflenti

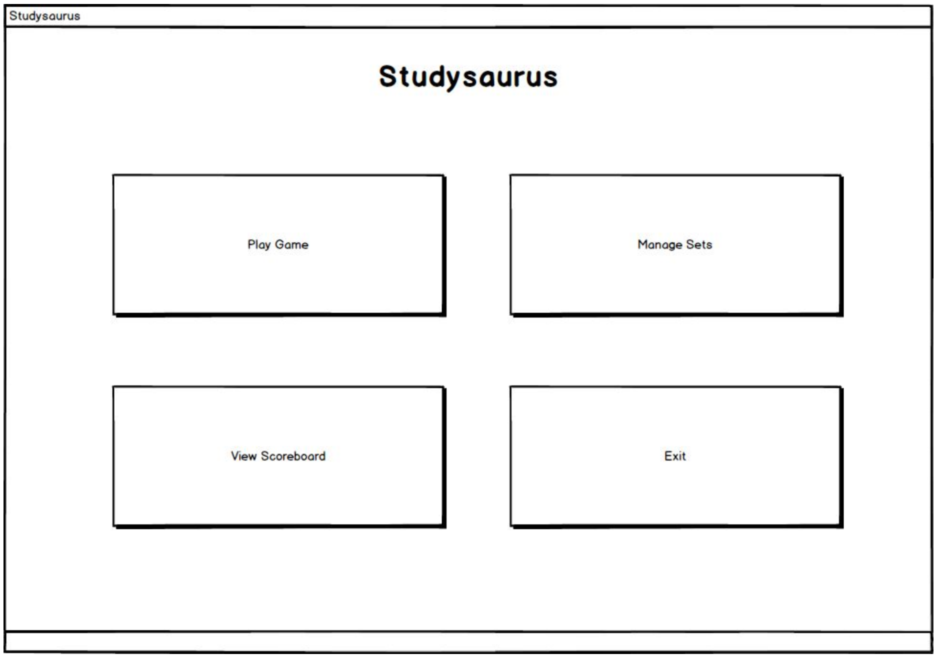

Use Case: UC-04 - Import Set, User Requirement: UR-03, Non-functional Requirement: NFR-05
USER
SYSTEM
Start
Click 'Manage Sets'
Display 'Manage Sets' page
Click 'Import Set'
Display 'Import Set' page
Select file containing set
Enter set name
Click 'Import'
Open file, parse by line, and create a term/value from each line. Check all terms/values to ensure database integrity. Save all term/value paris as a set in the database. Display the 'Success' page
NFR-05
Click 'Done'
Display 'Home' page
End
Legend
Activity
Decision
Requirement ID
Fork/Join
Flow

Data Storage

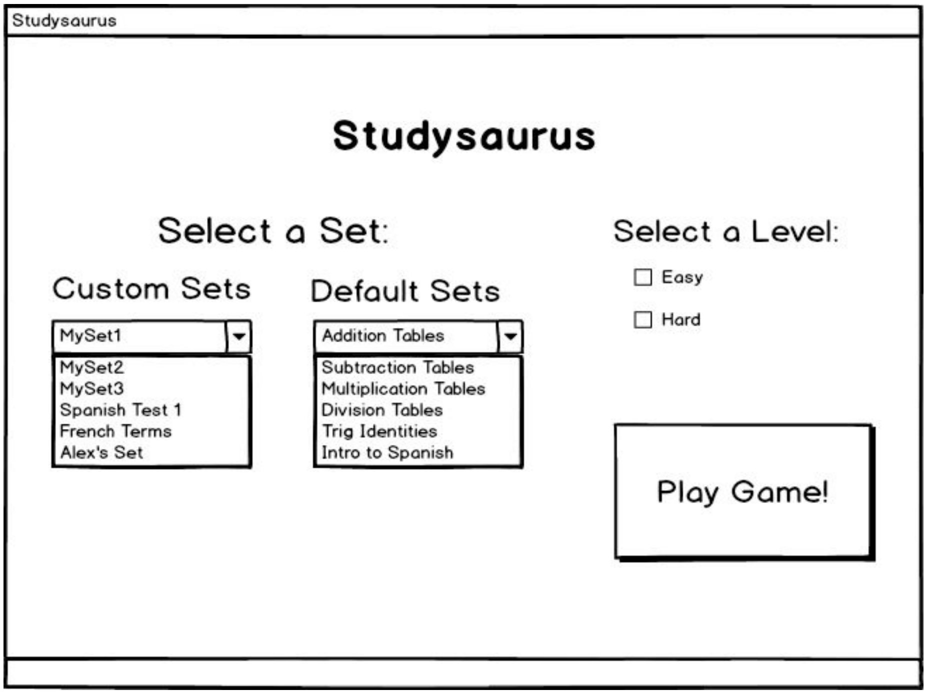
We will be using Hibernate to store our sets and scores in a MySQL database.

UI Mockups

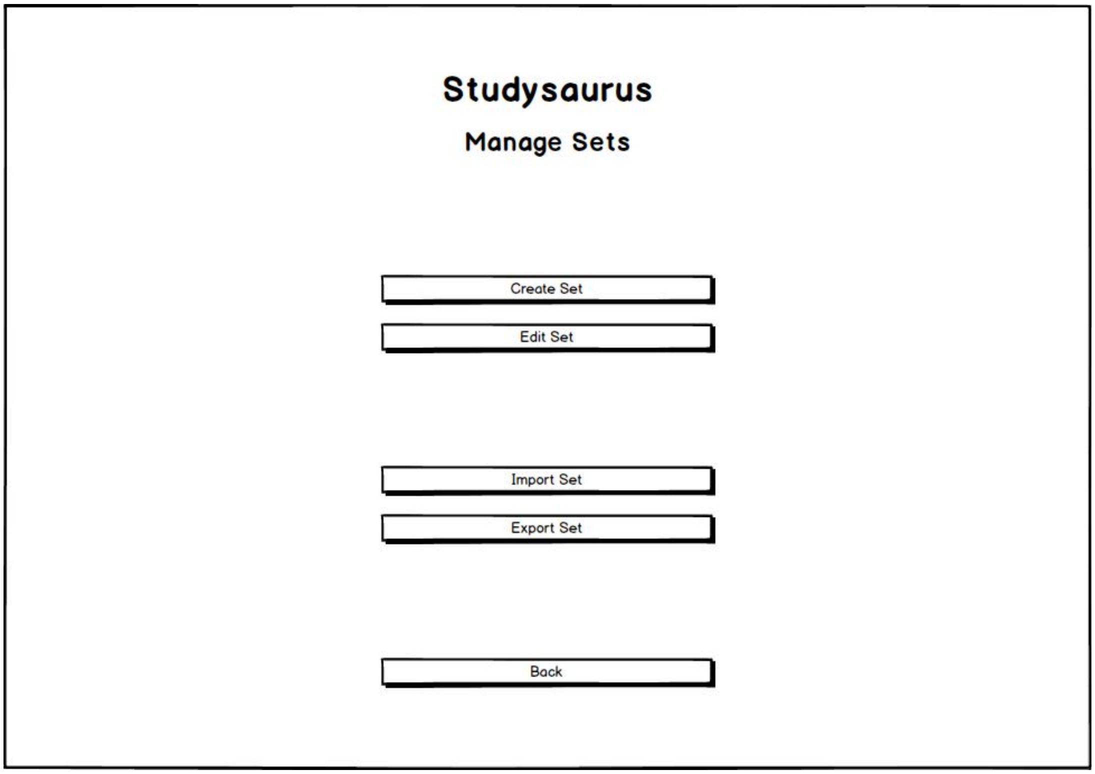
HomePage



GameOptionsPage



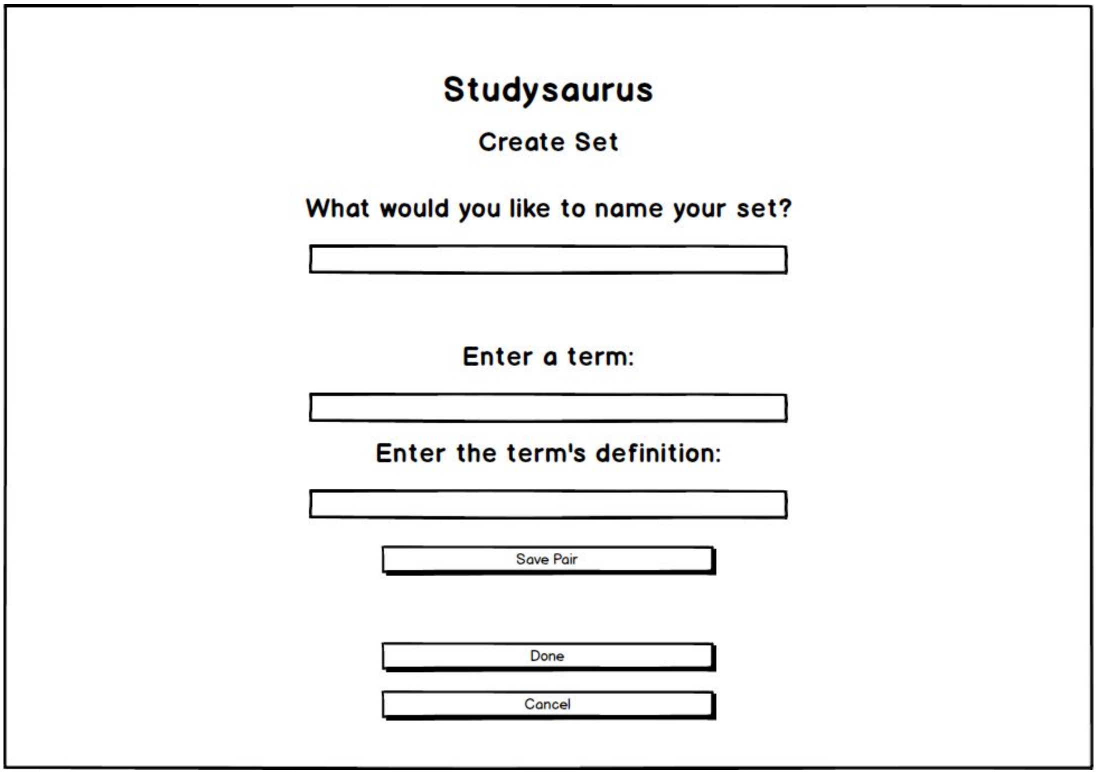
ManageSetsPage



Studysaurus

Manage Sets

CreateSetPage



Studysaurus

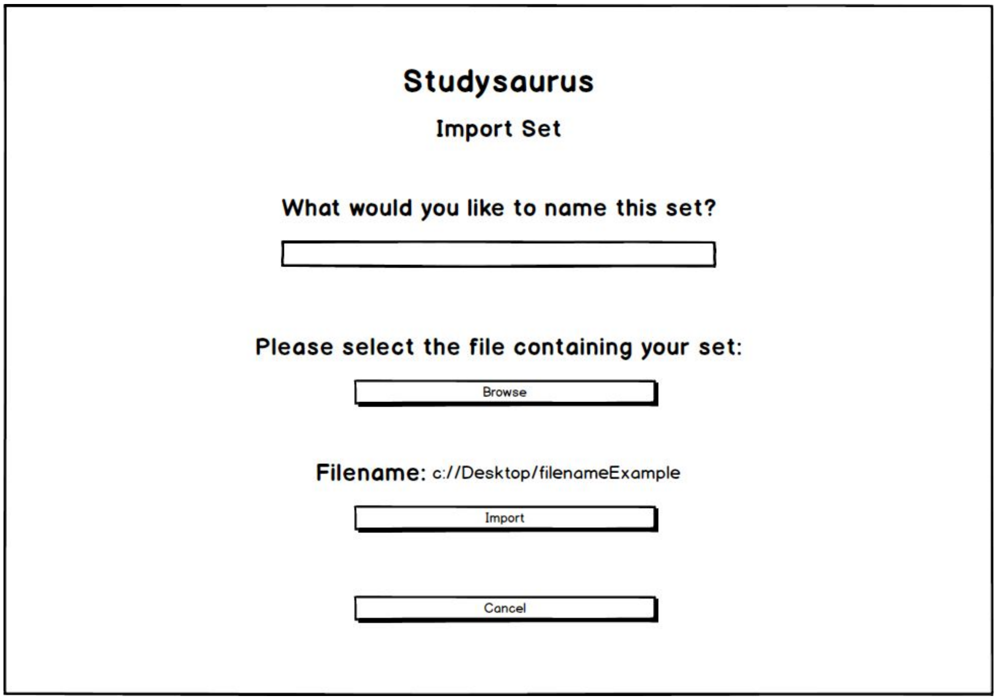
Create Set

What would you like to name your set?

Enter a term:

Enter the term's definition:

ImportSetsPage



Studysaurus

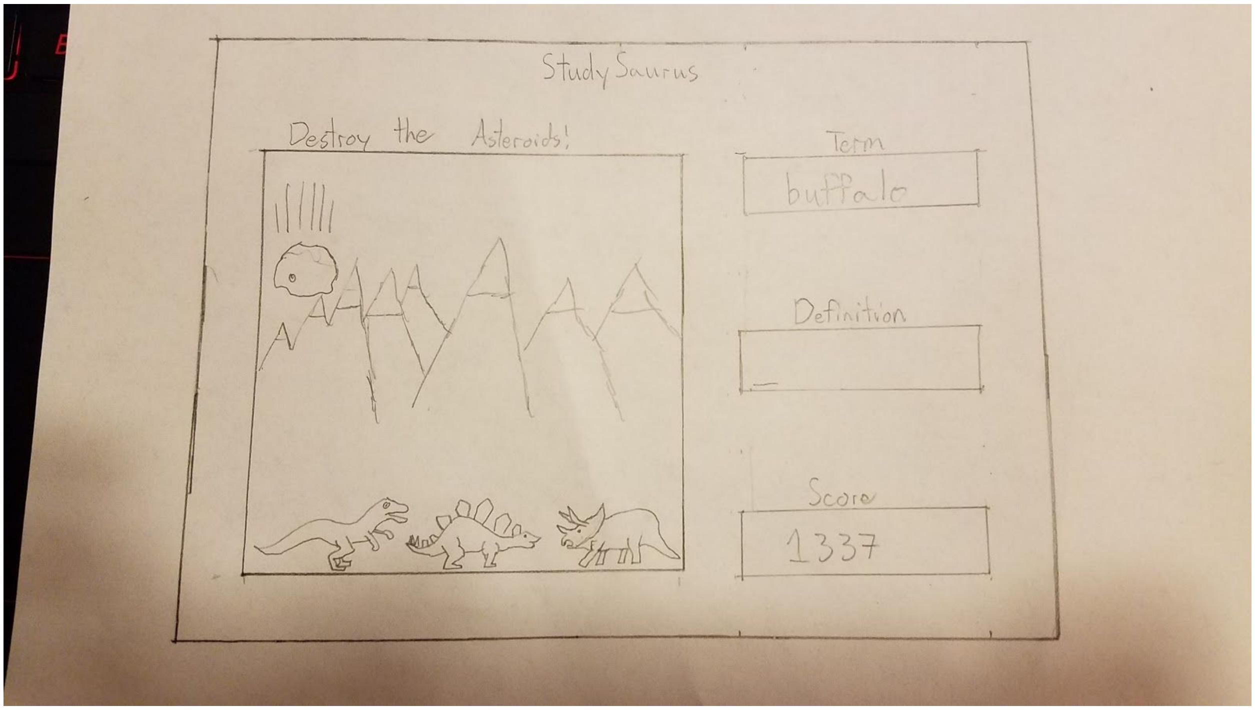
Import Set

What would you like to name this set?

Please select the file containing your set:

Filename: c://Desktop/filenameExample

PlayGamePage:



StudySaurus

Destroy the Asteroids!

Term

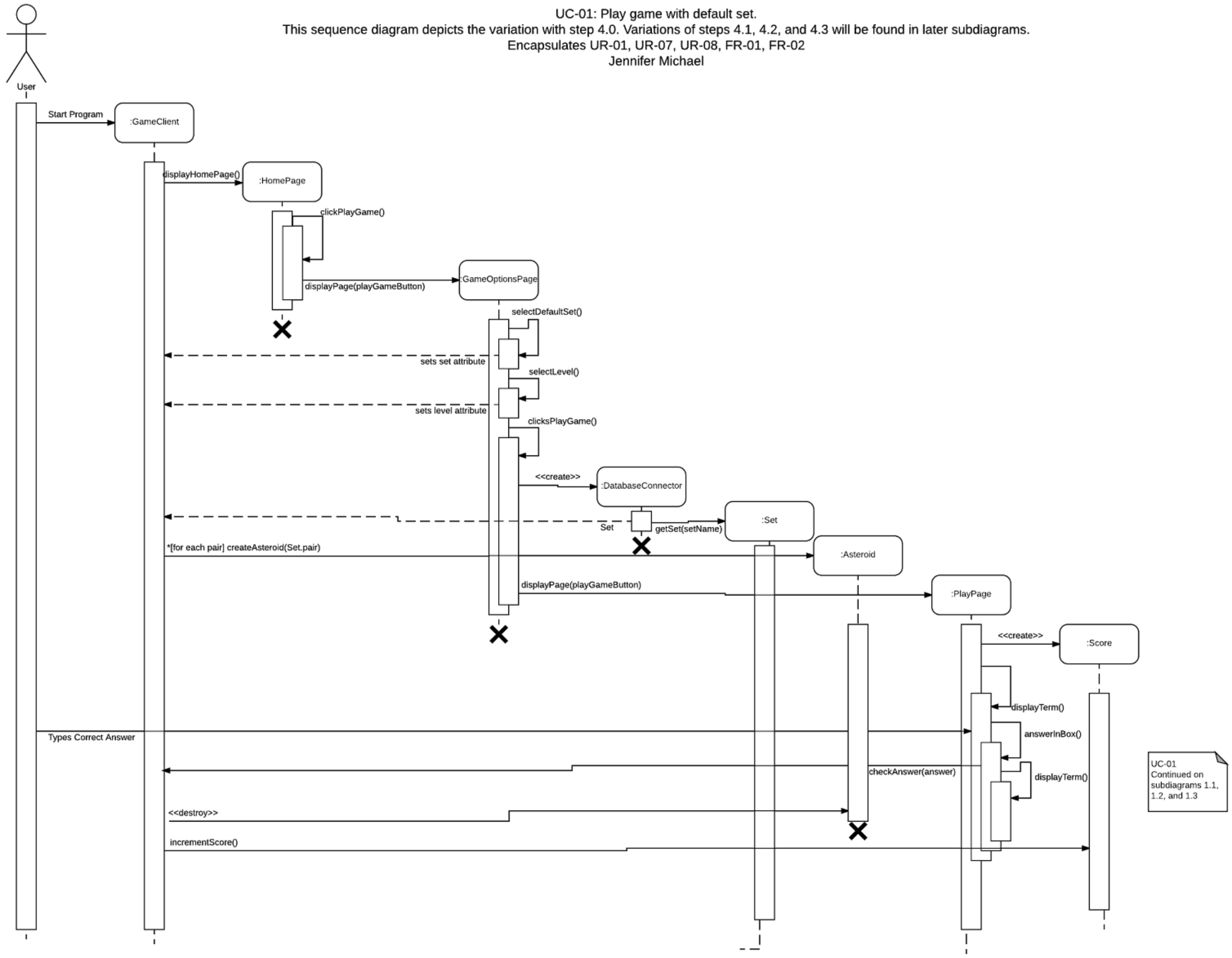
buffalo

Definition

Score

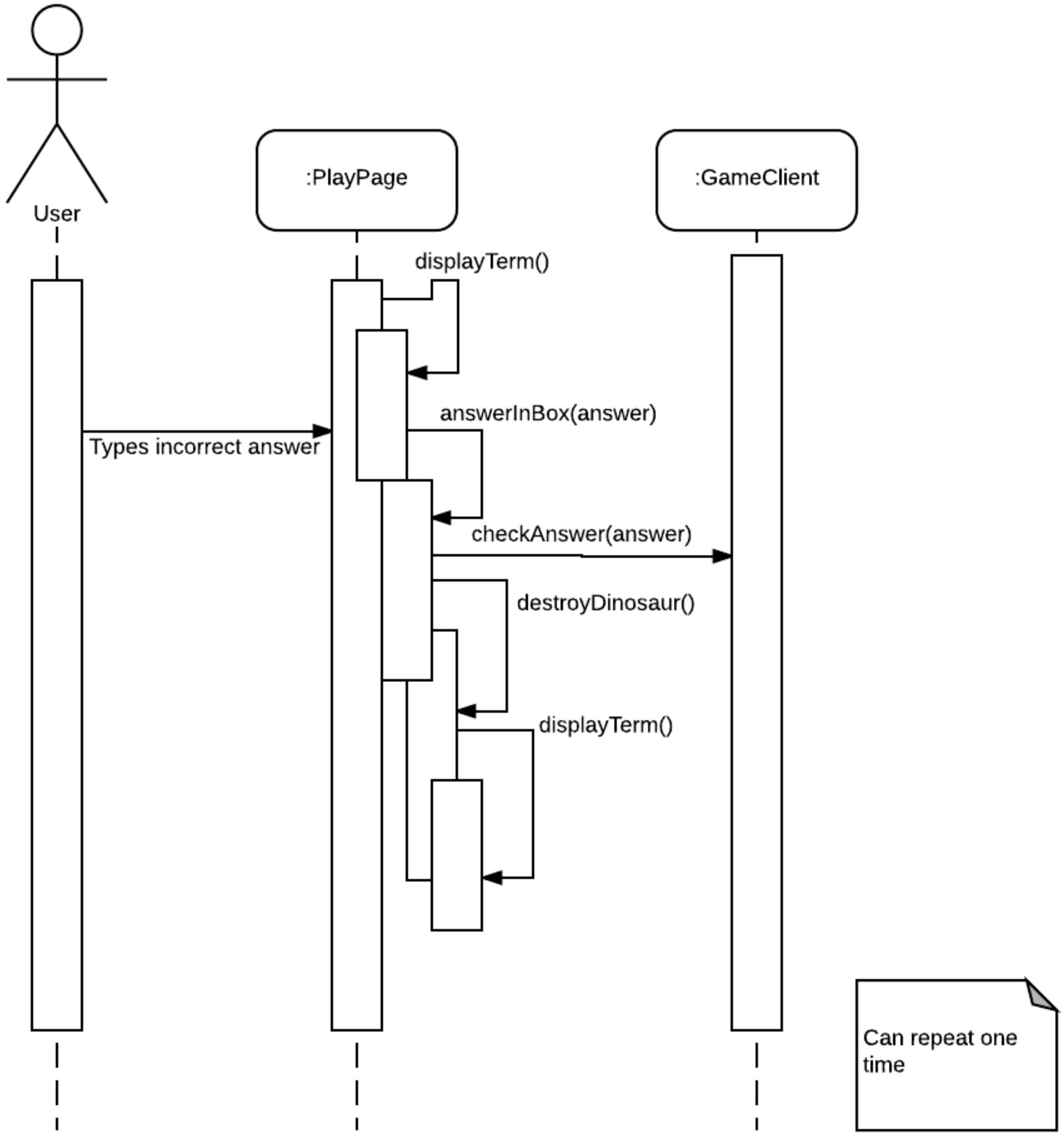
1337

Sequence Diagrams

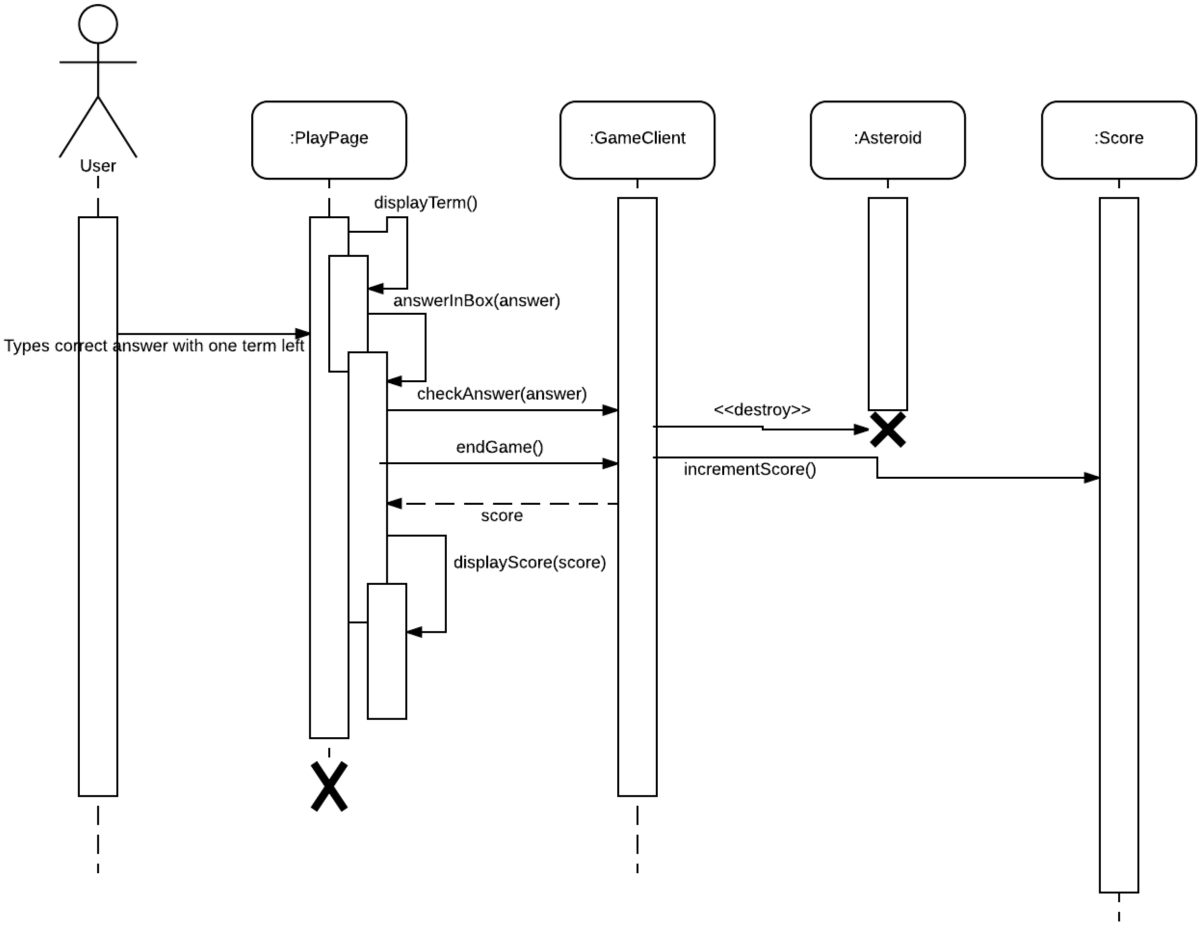


UC-01: Play game with default set.
This sequence diagram depicts the variation with step 4.0. Variations of steps 4.1, 4.2, and 4.3 will be found in later subdiagrams.
Encapsulates UR-01, UR-07, UR-08, FR-01, FR-02
Jennifer Michael

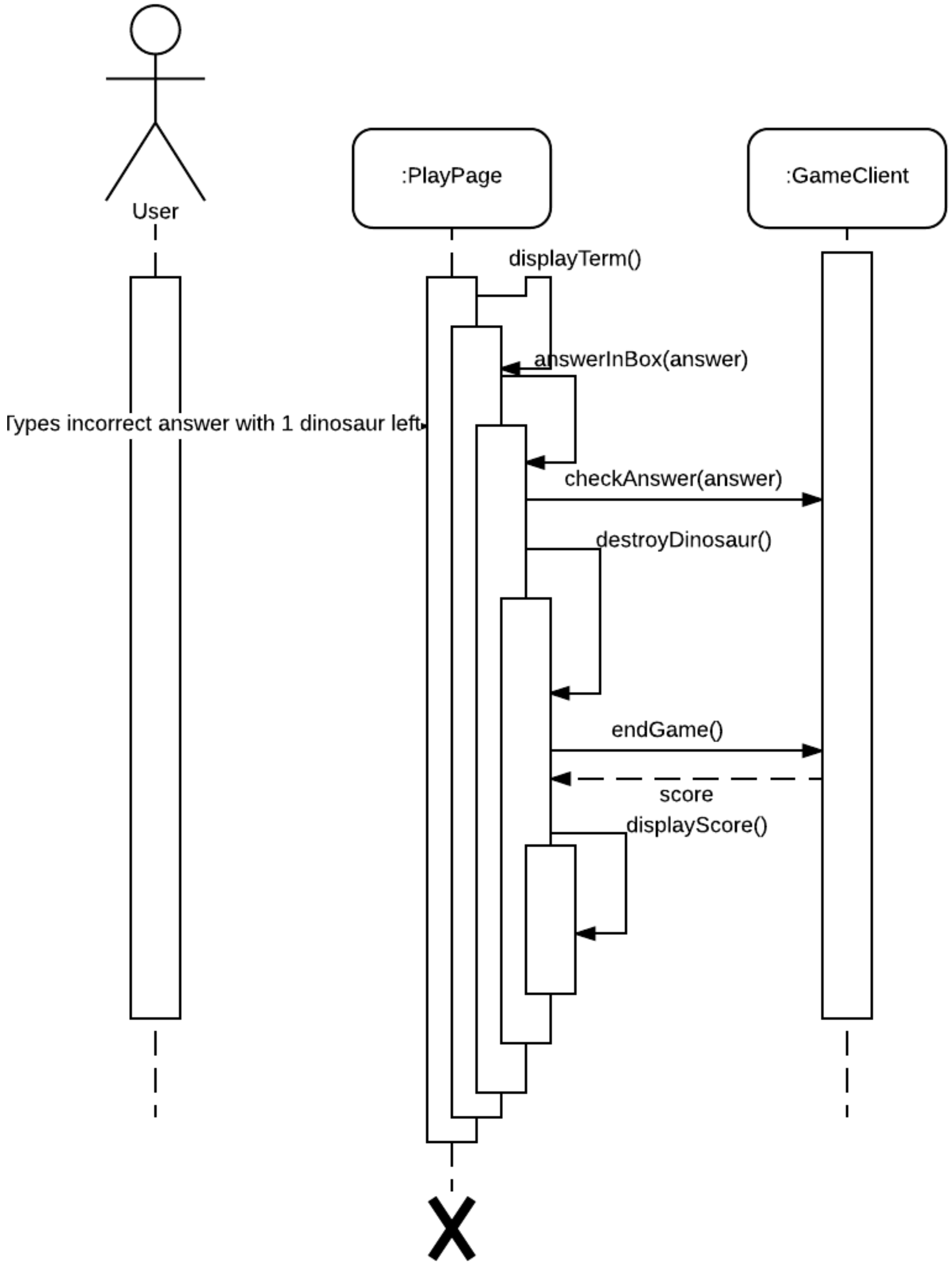
UC-01 Subdiagram 1.1 : Play game with default set
Variation with step 4.1 : User enters incorrect term



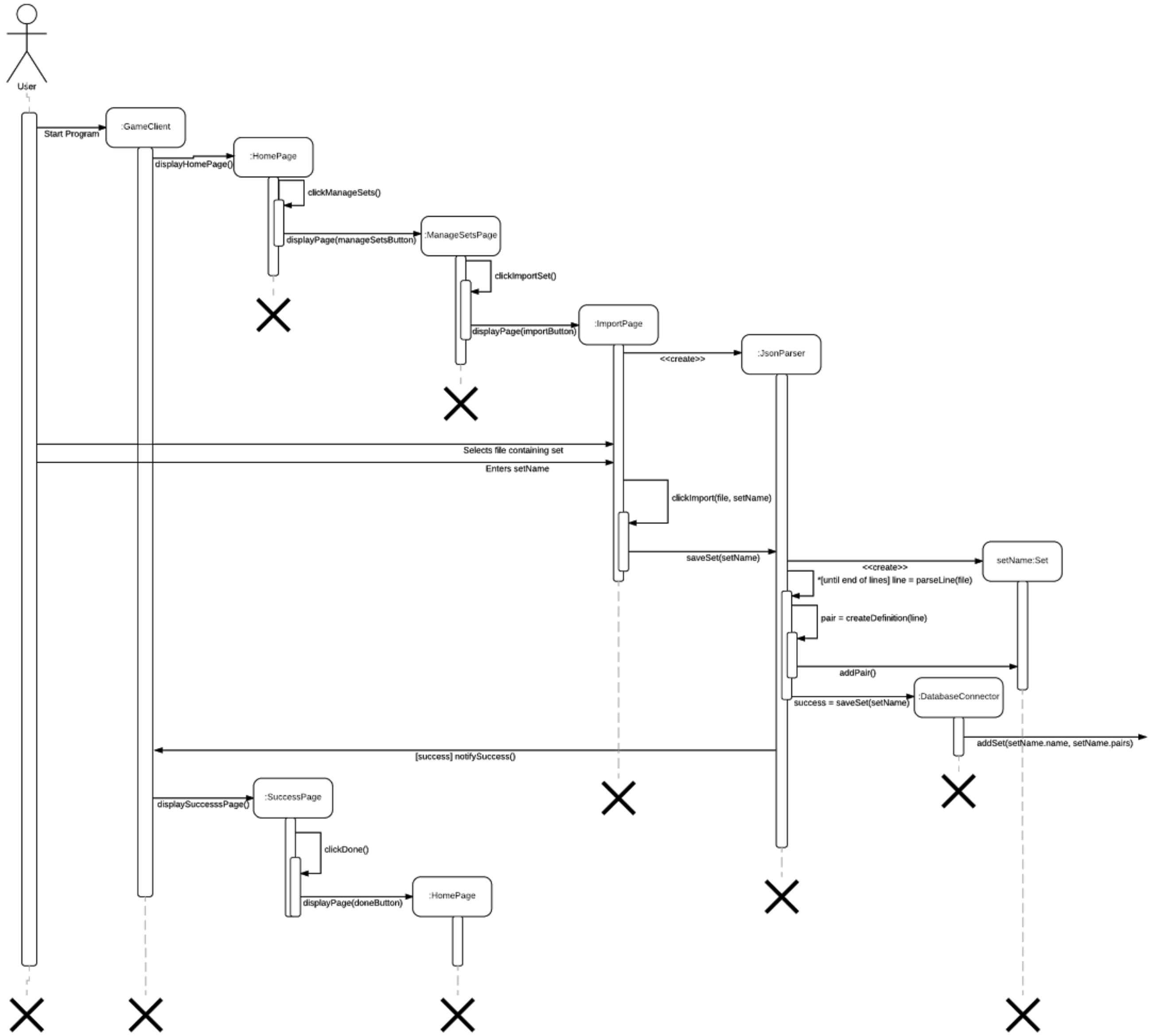
UC-01 Subdiagram 1.2 : Play game with default set
Variation wth step 4.2 : User enters correct term with one term left



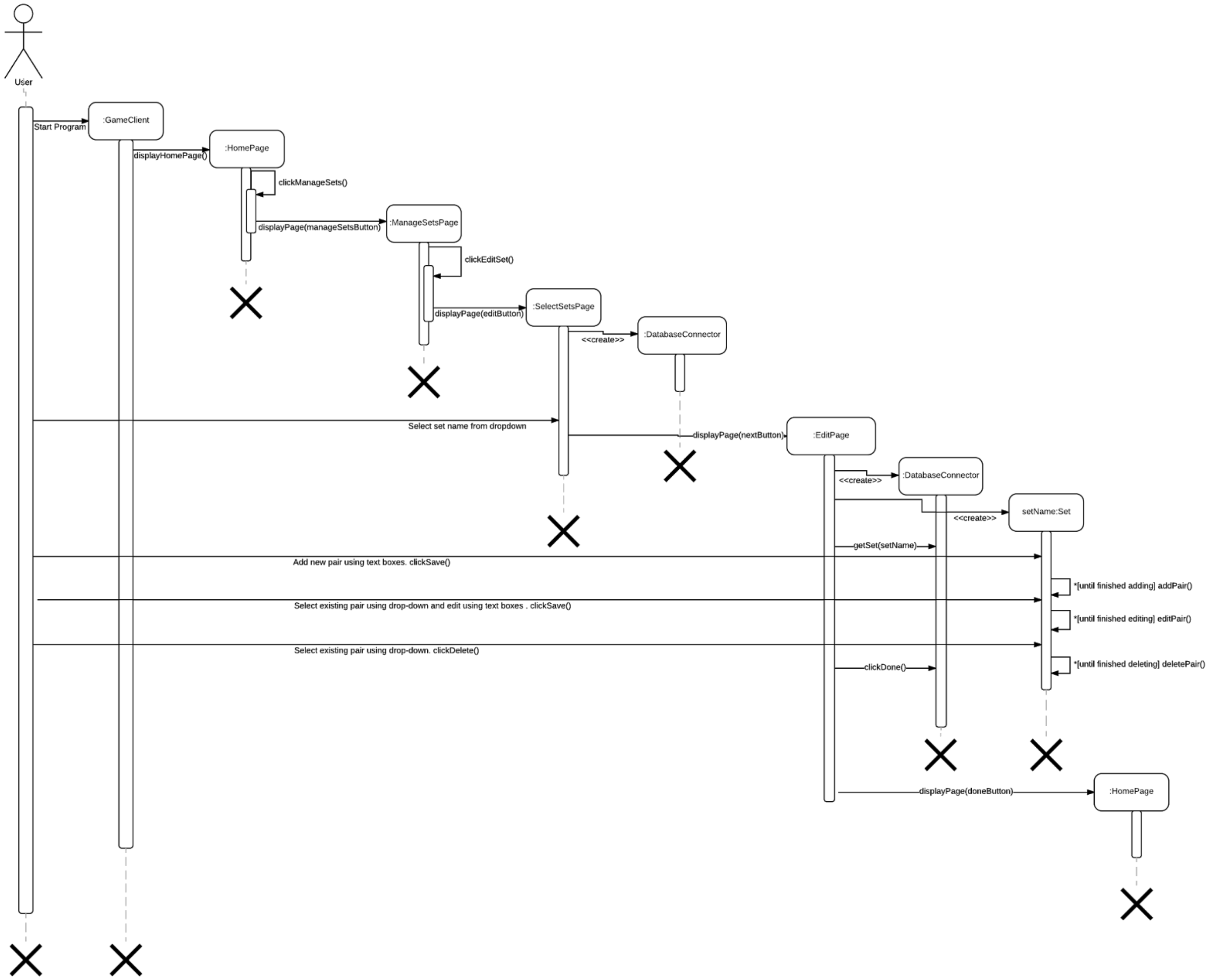
UC-01 Subdiagram 1.3 : Play game with default set
Variation wth step 4.3 : User enters incorrect term with one dinosaur left



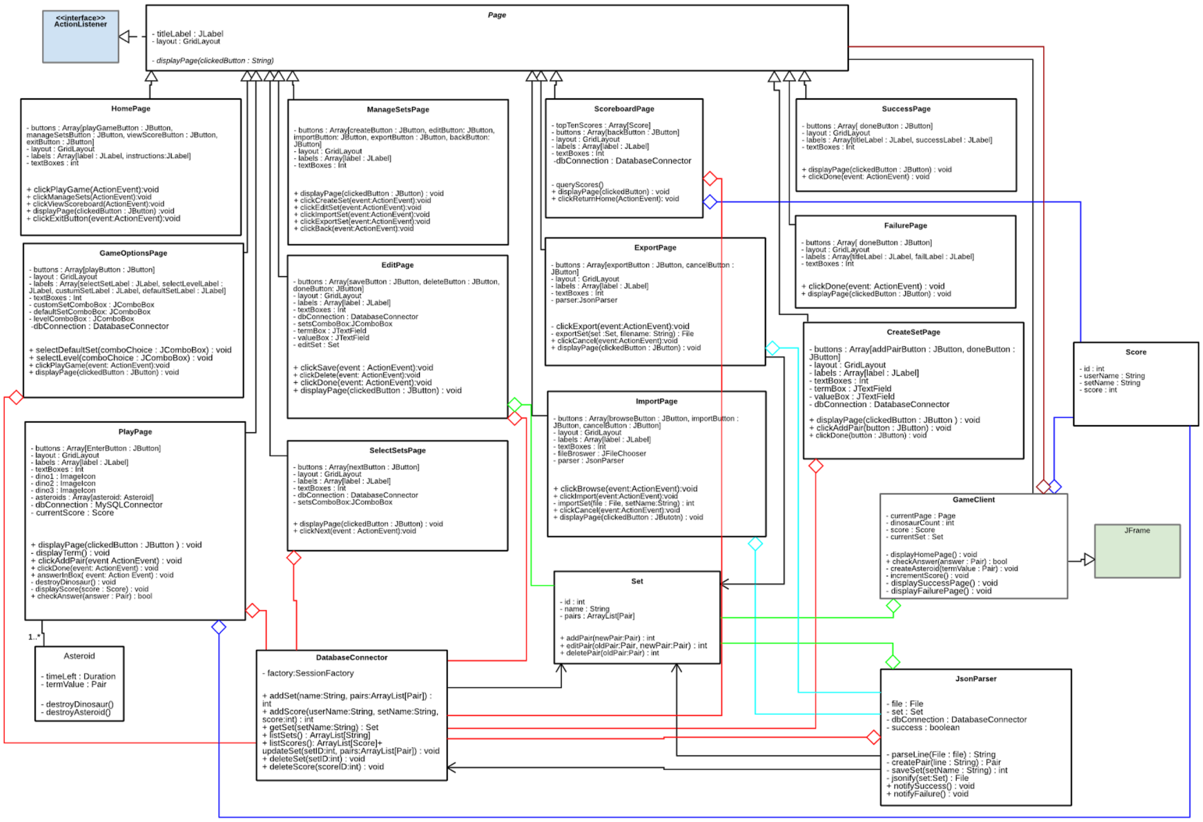
BASIC SEQUENCE DIAGRAM | UC-04: Import Set | User Requirement: UR-03 | Non-functional Requirement: NFR-05 Steven Conflenti | October 12, 2016



BASIC SEQUENCE DIAGRAM | UC-05: Edit Set | User Requirement: UR-04 | Non-functional Requirement: NFR-05 Cameron Taylor | October 12, 2016


User
:GameClient
Start Program
:HomePage
displayHomePage()
clickManageSets()
:ManageSetsPage
displayPage(manageSetsButton)
clickEditSet()
:SelectSetsPage
displayPage(editButton)
:DatabaseConnector
<<create>>
Select set name from dropdown
displayPage(nextButton)
:EditPage
:DatabaseConnector
<<create>>
setName:Set
<<create>>
getSet(setName)
Add new pair using text boxes. clickSave()
*[until finished adding] addPair()
Select existing pair using drop-down and edit using text boxes . clickSave()
*[until finished editing] editPair()
Select existing pair using drop-down. clickDelete()
clickDone()
*[until finished deleting] deletePair()
displayPage(doneButton)
:HomePage

Class Diagram


<<interface>>
ActionListener
Page
- titleLabel : JLabel
- layout : GridLayout
- displayPage(clickedButton : String)
HomePage
- buttons : Array[playGameButton : JButton, manageSetsButton : JButton, viewScoreButton : JButton, exitButton : JButton]
- layout : GridLayout
- labels : Array[label : JLabel, instructions:JLabel]
- textBoxes : int
+ clickPlayGame(ActionEvent):void
+ clickManageSets(ActionEvent):void
+ clickViewScoreboard(ActionEvent):void
+ displayPage(clickedButton : JButton) :void
+ clickExitButton(event:ActionEvent):void
ManageSetsPage
- buttons : Array[createButton : JButton, editButton: JButton, importButton: JButton, exportButton : JButton, backButton: JButton]
- layout : GridLayout
- labels : Array[label : JLabel]
- textBoxes : Int
+ displayPage(clickedButton : JButton) : void
+ clickCreateSet(event:ActionEvent):void
+ clickEditSet(event:ActionEvent):void
+ clickImportSet(event:ActionEvent):void
+ clickExportSet(event:ActionEvent):void
+ clickBack(event:ActionEvent):void
ScoreboardPage
- topTenScores : Array[Score]
- buttons : Array[backButton : JButton]
- layout : GridLayout
- labels : Array[label : JLabel]
- textBoxes : Int
-dbConnection : DatabaseConnector
- queryScores()
+ displayPage(clickedButton) : void
+ clickReturnHome(ActionEvent): void
SuccessPage
- buttons : Array[doneButton : JButton]
- layout : GridLayout
- labels : Array[titleLabel : JLabel, successLabel : JLabel]
- textBoxes : Int
+ displayPage(clickedButton : JButton) : void
+ clickDone(event: ActionEvent) : void
FailurePage
- buttons : Array[doneButton : JButton]
- layout : GridLayout
- labels : Array[titleLabel : JLabel, failLabel : JLabel]
- textBoxes : Int
+ clickDone(event: ActionEvent) : void
+ displayPage(clickedButton : JButton) : void
GameOptionsPage
- buttons : Array[playButton : JButton]
- layout : GridLayout
- labels : Array[selectSetLabel : JLabel, selectLevelLabel : JLabel, custumSetLabel : JLabel, defaultSetLabel : JLabel]
- textBoxes : Int
- customSetComboBox : JComboBox
- defaultSetComboBox: JComboBox
- levelComboBox : JComboBox
-dbConnection : DatabaseConnector
+ selectDefaultSet(comboChoice : JComboBox) : void
+ selectLevel(comboChoice : JComboBox) : void
+ clickPlayGame(event: ActionEvent):void
+ displayPage(clickedButton : JButton) : void
EditPage
- buttons : Array[saveButton : JButton, deleteButton : JButton, doneButton: JButton]
- layout : GridLayout
- labels : Array[label : JLabel]
- textBoxes : Int
- dbConnection : DatabaseConnector
- setsComboBox:JComboBox
- termBox : JTextField
- valueBox : JTextField
- editSet : Set
+ clickSave(event : ActionEvent):void
+ clickDelete(event: ActionEvent):void
+ clickDone(event: ActionEvent):void
+ displayPage(clickedButton : JButton) : void
ExportPage
- buttons : Array[exportButton : JButton, cancelButton : JButton]
- layout : GridLayout
- labels : Array[label : JLabel]
- textBoxes : Int
- parser:JsonParser
- clickExport(event:ActionEvent):void
- exportSet(set :Set, filename: String) : File
+ clickCancel(event:ActionEvent):void
+ displayPage(clickedButton : JButton) : void
CreateSetPage
- buttons : Array[addPairButton : JButton, doneButton : JButton]
- layout : GridLayout
- labels : Array[label : JLabel]
- textBoxes : Int
- termBox : JTextField
- valueBox : JTextField
- dbConnection : DatabaseConnector
+ displayPage(clickedButton : JButton) : void
+ clickAddPair(button : JButton) : void
+ clickDone(button : JButton) : void
Score
- id : int
- userName : String
- setName : String
- score : int
PlayPage
- buttons : Array[EnterButton : JButton]
- layout : GridLayout
- labels : Array[label : JLabel]
- textBoxes : Int
- dino1 : ImageIcon
- dino2 : ImageIcon
- dino3 : ImageIcon
- asteroids : Array[asteroid: Asteroid]
- dbConnection : MySQLConnector
- currentScore : Score
+ displayPage(clickedButton : JButton) : void
- displayTerm() : void
+ clickAddPair(event ActionEvent) : void
+ clickDone(event: ActionEvent) : void
+ answerInBox(event: Action Event) : void
- destroyDinosaur() : void
- displayScore(score : Score) : void
+ checkAnswer(answer : Pair) : bool
SelectSetsPage
- buttons : Array[nextButton : JButton]
- layout : GridLayout
- labels : Array[label : JLabel]
- textBoxes : Int
- dbConnection : DatabaseConnector
- setsComboBox:JComboBox
+ displayPage(clickedButton : JButton) : void
+ clickNext(event : ActionEvent):void
ImportPage
- buttons : Array[browseButton : JButton, importButton : JButton, cancelButton : JButton]
- layout : GridLayout
- labels : Array[label : JLabel]
- textBoxes : Int
- fileBroswer : JFileChooser
- parser : JsonParser
+ clickBrowse(event:ActionEvent):void
+ clickImport(event:ActionEvent):void
- importSet(file : File, setName:String) : int
+ clickCancel(event:ActionEvent):void
+ displayPage(clickedButton : JButotn) : void
GameClient
- currentPage : Page
- dinosaurCount : int
- score : Score
- currentSet : Set
- displayHomePage() : void
+ checkAnswer(answer : Pair) : bool
- createAsteroid(termValue : Pair) : void
- incrementScore() : void
- displaySuccessPage() : void
- displayFailurePage() : void
JFrame
Set
- id : int
- name : String
- pairs : ArrayList[Pair]
+ addPair(newPair:Pair) : int
+ editPair(oldPair:Pair, newPair:Pair) : int
+ deletePair(oldPair:Pair) : int
1..*
Asteroid
- timeLeft : Duration
- termValue : Pair
- destroyDinosaur()
- destroyAsteroid()
DatabaseConnector
- factory:SessionFactory
+ addSet(name:String, pairs:ArrayList[Pair]) : int
+ addScore(userName:String, setName:String, score:int) : int
+ getSet(setName:String) : Set
+ listSets() : ArrayList[String]
+ listScores(): ArrayList[Score]+ updateSet(setID:int, pairs:ArrayList[Pair]) : void
+ deleteSet(setID:int) : void
+ deleteScore(scoreID:int) : void
JsonParser
- file : File
- set : Set
- dbConnection : DatabaseConnector
- success : boolean
- parseLine(File : file) : String
- createPair(line : String) : Pair
- saveSet(setName : String) : int
- jsonify(set:Set) : File
+ notifySuccess() : void
+ notifyFailure() : void

Easier to read and better class boxes: https://www.gliffy.com/go/share/stmoti08vi4r5oknb53l

Note: We made the aggregation arrows different colors so that they are easier to discern from the others!

Note: Getters and setters are implied for private class variables to reduce diagram complexity.