Secure Rummy Web Game

Project Report
ISA/S
WE 681 - Fall 2016
12/10/2016

Jimmie Hardaway III
Rasika Mohod

INTRODUCTION

Secure Rummy Web Game is a variation of one of the most popular card game - Rummy. The game can be played by two players, each receiving ten cards and player's object of the game is to dispose of all the cards in his hand. In this implementation of the Rummy game, the winner is determined by the individual that reach max points of 100.

ARCHITECTURE

Our project RummyGame is implemented using the web framework - built using Spring MVC framework which includes three different tiers for segregation of actions:

- Model - To implement the code logic and database operations
- View - To generate the view and display of the application
- Controller - To create the connection channel between the model and view of the application and control the passage of data between the above two tiers.

The Maven Framework has been used to resolve any dependencies required to implement the web application. Along with our code you will find attached, the projects Maven pom.xml detailing the dependencies required to stand up the application.

The secured connections are developed on Transport layer using TLS/SSL protocol and as well an in depth security is implemented by forcing users to go through an authentication layer and validate to enter into the system. The application is then developed at the back end to render our game logic mappings to the resources needed to play a the Rummy Game.

The Secure Rummy game is built on Tomcat v7.0 to server. The Java servlets are used as the Model and Controller and JSPs provide the View i.e. the game display.

MySQL database is utilized as the backend in Database tier of our web application, containing multiple tables used for user information, passwords, audit data and game statistics.

Given below is the description of important code files of the application:

View

- Welcome.jsp
  Landing page of the application
- Register.jsp
  Web page to allow users to register into the application
- Login.jsp
  Web page to allow users to login into the application
- Home.jsp
  Web page for each individual user to start the new game and view the game dashboard about number of wins and losses
- PlayerView.jsp
  Web page for each player to view his audit data of the previous and current games

Model

- **Database Utilities**
  - DatabaseConnect.java
    A singleton class to establish database connection.
  - DataService.java
    A class to perform common database operations
- **Authentication**
  - AuthenticationOperations.java

    A class to perform actions for authentication such as register user, validate user etc.

- PasswordEncryption.java
    A class to perform password encryption

- **Game Play**
    - Card.java
        A class to define card objects which creates a card providing the suit and rank of a card
    - CardRank.java
        A class to define CardRank ENUM which provides the rankings of a deck of cards and their total values
    - CardSuit.java
        A CardSuit class provides the suits of a deck of cards

    - Rummy.java
        This object provides representational object of a dealer in a Rummy game. Shuffles cards, deals to player and provides the stock and discard pile.
    - RummyAction.java
        This object provides individual player actions that can be utilized throughout the game.

    - RummyDAO.java
        Data access object that provides methods to query the database
    - Player.java
        Object-relational mapping object to its table in the database
    - Stats.java
        Object-relational mapping object to its table in the database
    - Games.java
        Object-relational mapping object to its table in the database
    - Audit.java
        Object-relational mapping object to its table in the database

    - DealerMessage.java
        This object provides log messages for the dealer
    - PlayerMessage.java
        This object provides messages for the players

ACCESS GAME

Dependencies:

Tomcat 7 server
Maven build environment
Java compatible IDE
MySQL database configured (RummyGame_SQL_Commands.txt file is attached)

Once the project is run on Tomcat server, you can **access** it at:

Welcome Page:	https://localhost:8443/RummyLocal/
Login Page:	https://localhost:8443/RummyLocal/loginUser
Registration Page:	https://localhost:8443/RummyLocal/registerUser

GAME RULES

The Deck

One standard deck of 52 cards is used. Cards in each suit rank, from low to high:

Ace 2 3 4 5 6 7 8 9 10 Jack Queen King.

The cards have values as follows:

Face cards (K,Q,J)	10 points
Ace	1 point

Number Cards are worth their face value - e.g. a six is worth 6 points, a four is 4 points.

The Deal

The first dealer is chosen randomly, and the turn to deal alternates between the two players. In this game, each player is dealt a hand of ten cards. The cards are dealt one at a time, and after the deal, the next card is placed face up on the table to start the discard pile, and the remainder of the deck is placed face down beside it to form the stock. The players look at and sort their cards.

Object of the Game

The object of the game is to dispose of all the cards in your hand. There are three ways to get rid of cards: melding, laying off, and discarding.

- **Melding**
  Melding is taking a combination of cards from your hand, and placing it face up in front of you on the table, where it stays. There are two kinds of combination which can be melded: sequences (also known as runs) and groups (also known as sets or books).
  - A **sequence** or **run** consists of three or more cards of the same suit in consecutive order, such as ♣**4,** ♣**5,** ♣**6** or ♥**8,** ♥**9,** ♥**10,** ♥**J**.
  - A **group**, **set** or **book** is three or four cards of the same rank, such as ♦**7,** ♥**7,** ♠**7.**

- **Laying off**
  Laying off is adding a card or cards from your hand to a meld already on the table. The cards added to a meld must make another valid meld.
  - For example to the ♣**4,** ♣**5,** ♣**6** you could add the ♣**3** or the ♣**7**. You are not permitted to rearrange the melds in the process.
  - For example, ♣**2,** ♥**2,** ♦**2,** ♠**2** and ♠**3,** ♠**4,** ♠**5** have been melded, you are not permitted to move the ♠**2** from the group to the sequence, so as to lay off the ♠**A**.

- **Discarding**
  Discarding is playing a card from your hand on top of the discard pile. You get rid of one card this way at the end of each turn.

GAME PLAY

Following are the details of each step and action performed during the Secure Web Rummy Game.

Play

The two players take alternate turns starting with the non dealer.

Each turn consists of the following parts:

- **The Draw**

  You must begin by taking one card from either the top of the Stock pile or the top card on the discard pile, and adding it to your hand. The discard pile is face up, so you can see in advance what you are getting. The stock is face down, so if you choose to draw from the stock you do not see the card until after you have committed yourself to take it. If you draw from the stock, you add the card to your hand without showing it to the other players.

- **Melding**

  If you have a valid group or sequence in your hand, you may lay one such combination face up on the table in front of you. You cannot meld more than one combination in a turn (but see House Rules). Melding is optional; you are not obliged to meld just because you can.

- **Laying off**

  This is also optional. If you wish, you may add cards to groups or sequences previously melded by yourself or others. There is no limit to the number of cards a player may lay off in one turn.

- **The Discard**

  At the end of your turn, one card must be discarded from your hand and placed on top of the discard pile face up. If you began your turn by picking up the top card of the discard pile you are not allowed to end that turn by discarding the same card, leaving the pile unchanged - you must discard a different card. You may however pick up the discard on one turn and discard that same card at a later turn. If you draw a card from the stock, it can be discarded on the same turn if you wish.

If the stock pile has run out and the next player does not want to take the discard, the discard pile is turned over, without shuffling, to form a new stock, and play continues.

A player wins an individual hand by either melding, laying off, or discarding all of his or her cards. Getting rid of your last card in one of these ways is called going out. As soon as someone goes out, play ceases. There can be no further melding or laying off, even if the other players have valid combinations in their hands.

Scoring

When a player goes out, the other players add up the value of all the cards still remaining in their hands, as follows:

- Face cards (K,Q,J) are worth 10 points each
- Aces are worth 1 point each
- Number Cards are worth their face value - for example a six is worth 6 points, a four is 4 points, and so on.

The total value of all the cards in the hands of the other players is added to the winner's cumulative score.

The game continues with further deals until a player reaches the points target that was decided before the game began, or until the agreed number of deals has been played.

Betting

Each player will be given a pot of $500 dollars per game, and will be able to wager between a specified amount before the start of each round. Each of the player's wager will remain in the bank until the end of a round, and provided to the winner.

Game Completion

In this implementation of the game Rummy, the winner is determined by the individual that reach max points of 100. Each time an individual successfully empties their hand, this will be consider a round. Rounds will occur until the max points are achieved by one of the two players.

SECURITY FEATURES

The Secure Rummy Web Game is developed with the basic essential security objectives:

- Confidentiality - No unauthorized read
  E.g.: Each user/player is given an authentication system to login to his individual dashboard. No other user/player is authenticated to view details of another user. A player can see only his cards for the particular running game.

- Integrity - No unauthorized modification
  E.g.: A player can start a game and play for himself A player cannot view cards of opponent or another player and also cannot make moves for others.
- Availability - Keeps working in presence of attack
  E.g.: A user's account is locked on attempt of Brute Force attack and also a game session is closed and deactivated if any of the player shows inactivity for a certain period of time (i.e. player is not playing game but busy in recording some other activities on the application).

Also other security objectives of auditing and logging are achieved with different respective functionalities.

Following are the detailed arguments placed to describe security features which have been implemented at each stage of the Secure Rummy Web Game application:

TLS

TLS is implemented in our web application to provide protection of our data during transmission. All requests to our application is be required to process data over the secure connection provided by Java's Secure Socket Extension (JSSE).
Tomcat is configured to utilize a keystore that is specific to the server running the application.

Authentication

The Rummy web application is utilizing username/password combination to provide access to the the system to play the game. This mechanism allows the system to track users by registering their data and create a secured environment for legitimate players.

The following mechanisms are used to enhance the protection of our application:

- **Username/Password Validation**
  - **Validation**
    To protect the application from injection, at the time of new user registration we test the user input to ensure username to be only alphanumeric (3 - 20 characters) and password has constraint of at least one number, one uppercase, one lowercase letter, and at least 8 or more characters.

Setting up a strong password helps the system to be prevented from Dictionary attacks.

- **Error Response**
  The error messages displayed on failed login attempts do not note if the account is valid/invalid and give minimum information to users.
  This helps to prevent identity of legitimate users and makes system safe from hackers/ unethical users.

- **Restrict Failed Attempts**
  Users are only allowed up to 3 consecutive failed attempts to log into their account. Failing to log in after 3 unsuccessful attempts, the user account is locked forever and can only be unlocked after contacting the administrator of the game system.
  This features helps the system to be prevented from Brute force attack.

- **Password Encryption**
  Along with input validation and constraints being put on passwords, the system performs encryption of the user registered passwords. The passwords are not stored as original text but are converted to salted hashed passwords using unique algorithm and then stored in database in the encrypted form.

- **Discrete Tables to Store User Details and Password**
  While registration user details (username and user email) and user password are not stored in the same table. Two different tables are used in database to store user details and user password which are mapped using unique field called IdUsers.
  This method prevents access to user password directly through user table and thus improves security of the user data in the application.
- **TOCTOU Prevention**
  The password encryption is performed in the system just before inserting the password in database.
  This enables the system to reduce the vulnerable TOCTOU gap in the system to be negligible.

- **Session Management**
    - **Session Creation**
      After a user successfully logs in the system, an unique user session is maintained to verify the user's state throughout the application.

    - **Session Close**
      A discrete 'Logout' button is enabled on each user's home page which enables user to close the session manually.
      This helps in preventing session hijacking.

    - **Inactivity Session Timeout**
      Also, along with manual session close by Logout option, system maintains the count of period (in seconds) for which user is inactive. Inactivity includes, no mouse movement, no document click and no key press. The session is closed automatically by system and user is logged out due to 'session timeout' after a period of inactivity.

  This feature helps the system to prevent its user's security being exposed and taken advantage of by unethical user.

Database

The database operations in the system are all performed in one single tier. A singleton class is used which ensures the secured connection to the database and one another single class is used to perform all common database operations (such as executing a query, fetching result set from database etc.) with all security and exception handling being taken care of.

This enables the system to adhere to the benefits of the well-defined data-integrity system:

1. Stability: One centralized system performing all data integrity operations
2. Performance: All data integrity operations are performed in the same tier as the consistency model
3. Re-usability: All applications benefit from a single centralized data integrity system
4. Maintainability: One centralized system for all data integrity administration

Other security features used in accessing database are as follows:

- **Prepared statement**
  Prepared statements are used to execute database queries. This prevents the systems against SQL Injection attacks.

- **Property File**
  All the database connection properties (driver, url, username, password, dbname) are retrieved from a private properties file while establishing database connection. This enables system to prevent attackers from establishing non legitimate database connections.
  A config.properties file is used to store all the properties of the application.

- **ATOMIC Transactions**
  The dependent database transactions are performed at a time which ensure that database is not brought to an illegal state.

Logging

Logging is implemented in the application which logs the information *(LOGGER.info())* throughout the flow of the application and game and also logs the errors *(LOGGER.error())* encountered while running the application.

This provides the crucial forensics needed to investigate after a breach in the application, and perhaps more importantly, a change to detect security issues as they happen. The logging also helps in debugging and diagnostic purposes, along with the security logging.

Simple Logging Facade for Java (SLF4J) is used as a tool for implementing Logging feature in the application. SLF4J provides a Java logging API by means of a simple facade pattern. Irt provides the separation of the client API from the logging backend which thus reduces the coupling between an application and any particular logging framework.

The logging gives an easy way in logging security events, tracking extra forensic information like the who (username), what (event type), and where (IP address, server name) needed for forensics. It also provides a means for classifying the information in log messages and applying masking if necessary.

As evidences to the above arguments, the test runs have been performed on the application to validate each of the security functionality.