Lucky Rollers

- A WEB APPLICATION -

Created By:

De Leon, Thomas
Robles, Joseph Isaac

2CSB

Submitted To:

Mr. Lawrence G. Decamora III, LPT, OCPJP, MAEd

Table of Contents

Lucky Rollers is a simple web lottery application. Users can try their luck by generating tickets consisting of 3 two-digit numbers and see if they win when the winning numbers are drawn once they validate their ticket.

Lucky Rollers came about from the developers' desire to create something that was relatively simple and familiar to them given their familiarity with the systems incorporated into the games they frequently play, particularly that being the "lootbox" and "gachapon" (gacha) system which rewards the player with random rewards from a certain prize pool at varying chances.

The developers wanted to create an application that simulates the "randomness" aspect of that particular system and create a close representation of how it operates in relation to its real-world equivalent: lottery and sweepstakes.

* The developers are **NOT** gambling addicts. They simply play a lot of games which incorporates these systems.
  Please believe us we don't have issues WE SWEAR
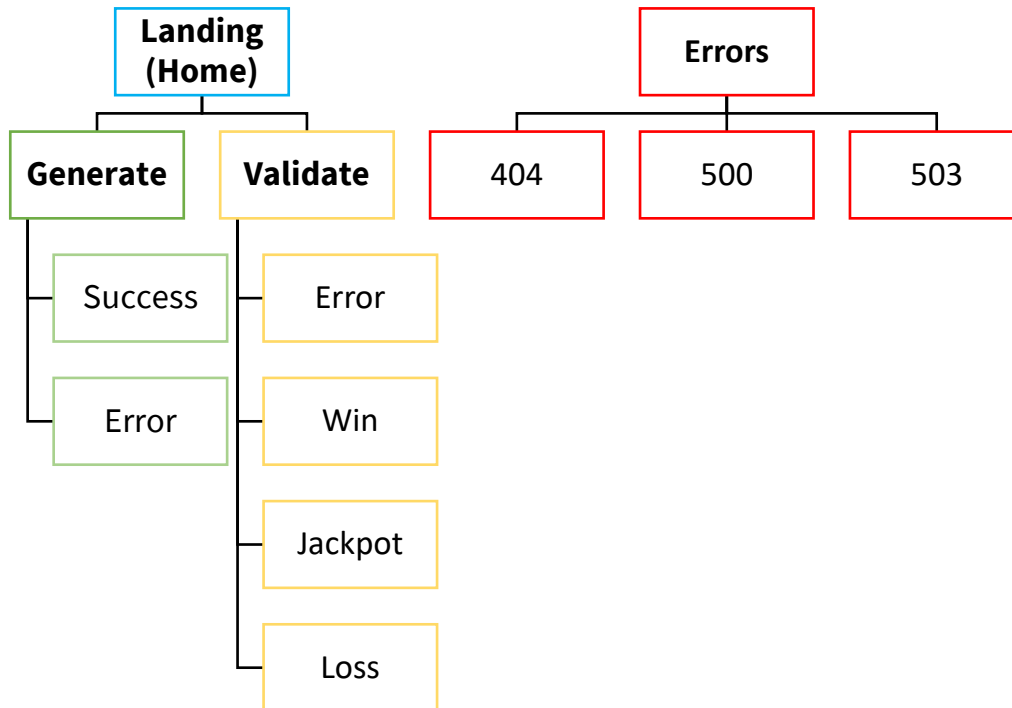
THE WEB APPLICATION

Diagram 1: Site Navigation Hierarchy

The web application consists of 12 different pages as JSPs. Users start from the **Landing** page and can freely navigate to either the **Generate** or **Validate** pages. Each of these two pages come with their own subpages that are accessed the further their parent pages are explored and used.
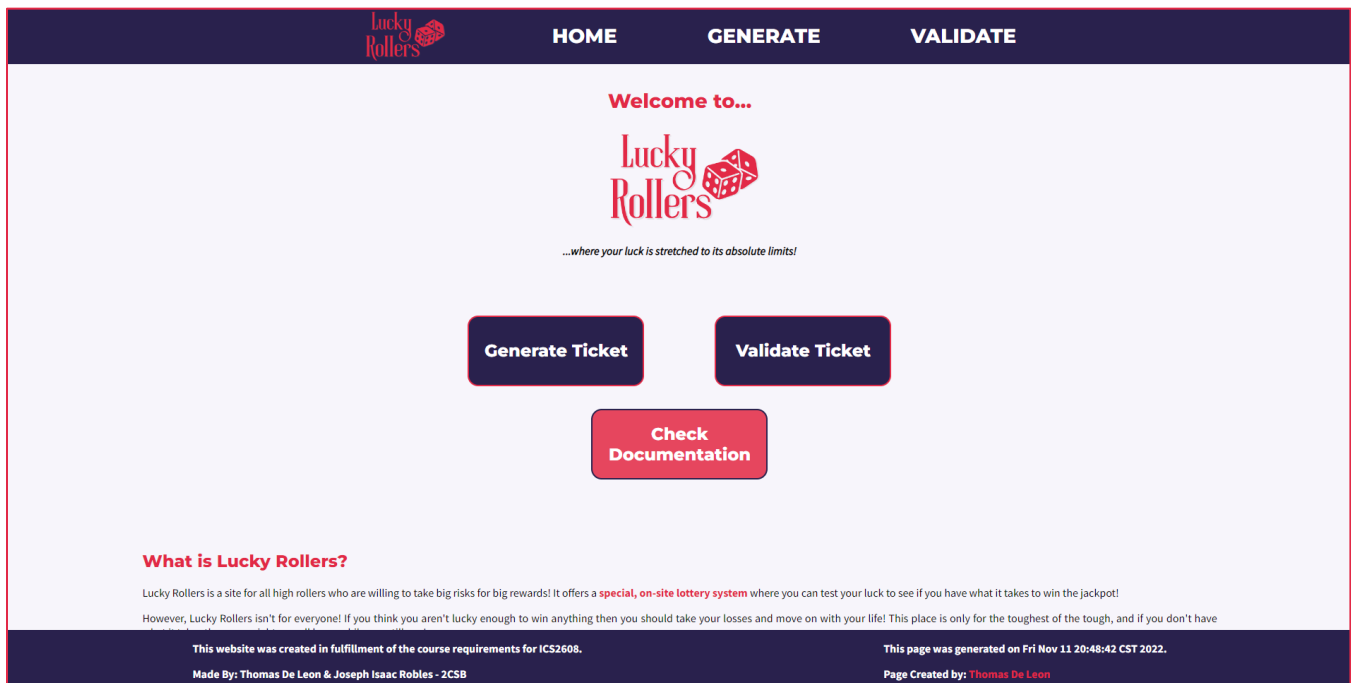

Figure 1: The Landing (Home) Page

The **Generate** page should typically be used first to create tickets that house lottery numbers to be checked in the **Validate** page. These pages, along with how they function internally, will be discussed in subsequent sections below.

The application also features customized **Error** pages are automatically accessed when the application does not function properly, although the chances of this are very rare unless deliberate actions were taken to trigger them to appear.
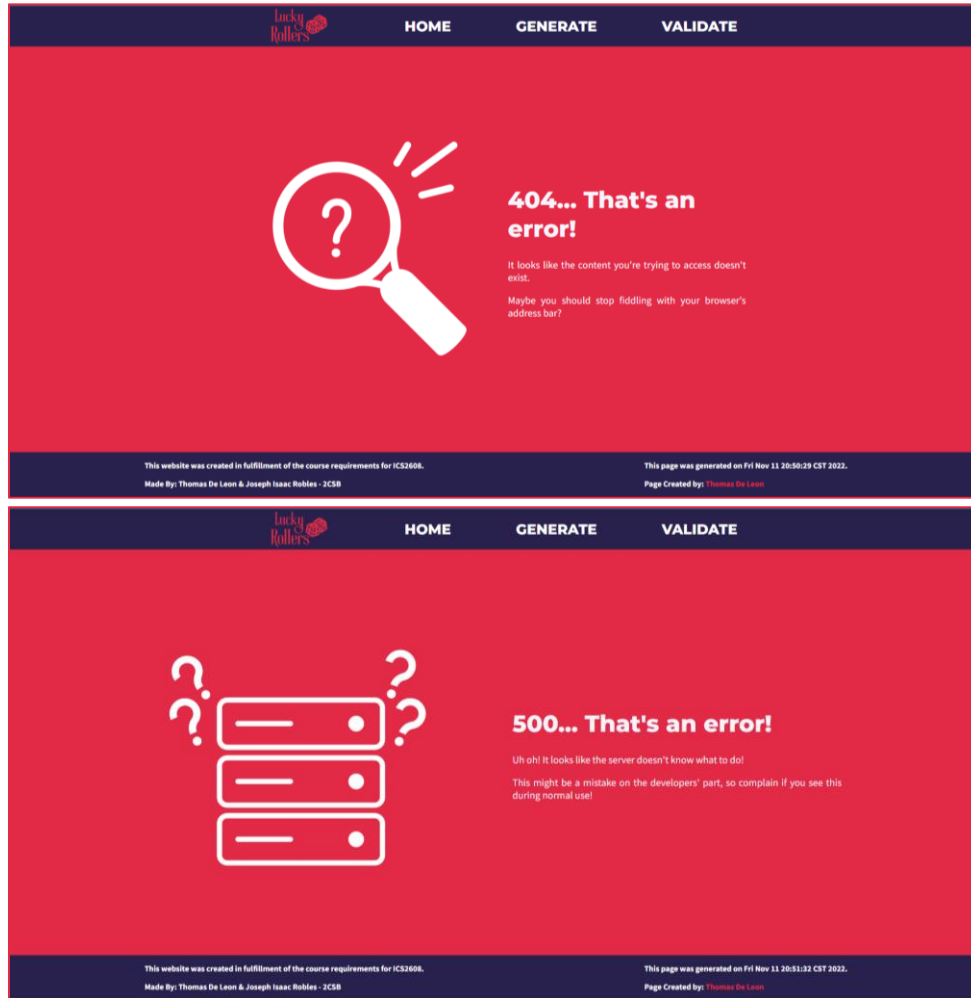

Figure 2: Some Customized Error Pages

Aside from that, the site also features a **customized, responsive design** using CSS that adapts to any **computer** screen size, automatically scaling its content as needed. This ensures that elements do not clip nor are positioned weirdly when the screen shrinks/grows to any size.
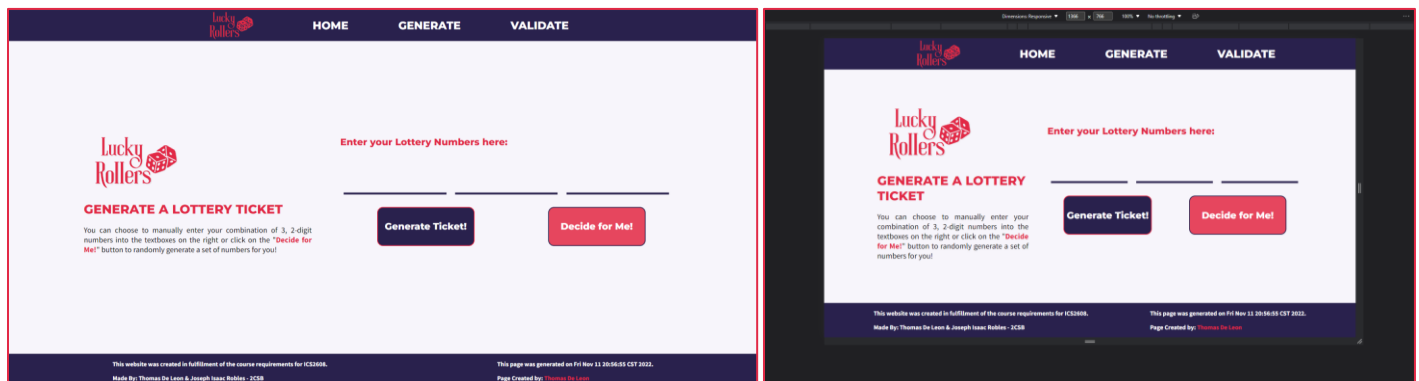

Figure 3: Adaptation based on Screen Size
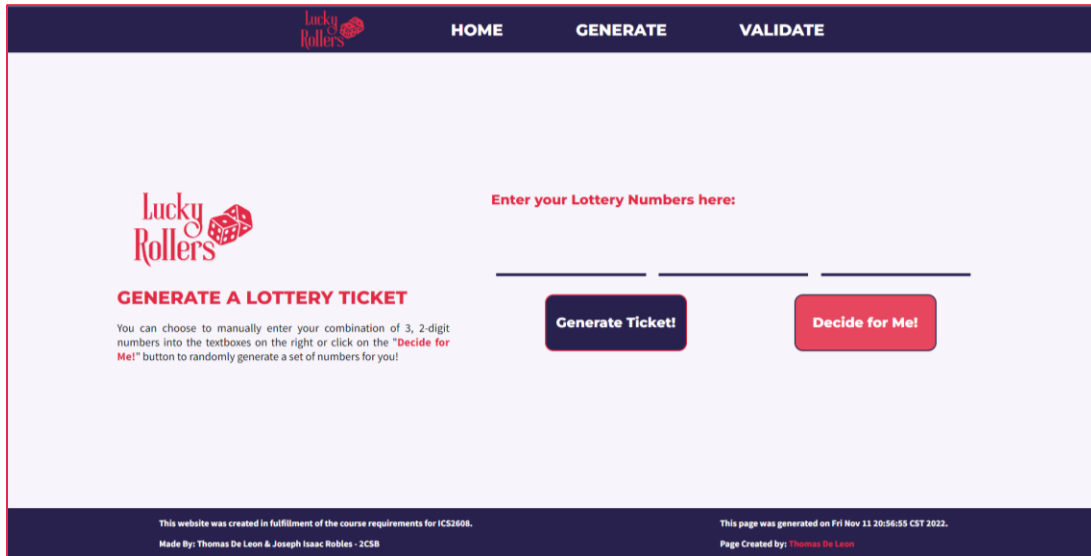
GENERATING TICKETS

Figure 4: The Generate Page

The **Generate** page is where users go to create their lottery tickets. They can choose to enter their own tickets on the spaces provided on the right or click on the **Decide for Me!** button to have its handling servlet (`TicketServlet`) generate and return ticket numbers automatically using its `DiscountGachaSimulator` class.

The "Decide for Me!" function is guaranteed to generate a `Set(java.util.Set)` of 3 unique numbers before it `forward()`s the result back to the user. In case the user enters duplicate lottery numbers, the page's handling servlet will also check to see if any of the entered numbers are duplicates using its `DupeFinder` class.

When duplicate entries are found, it returns the `generate-error` page to the user to inform them that there is a problem with their ticket entry and have them try again. Otherwise, it saves the entered numbers as attributes and send it to another servlet (`SavingTicketServlet`) so that it could save the ticket information locally in the web application. The application also generates a unique ID for the ticket using its `IDGenerator` class as sets it as an attribute to be included when sending the `generate-success` page to the user, confirming that their ticket has been successfully created and saved on the server.
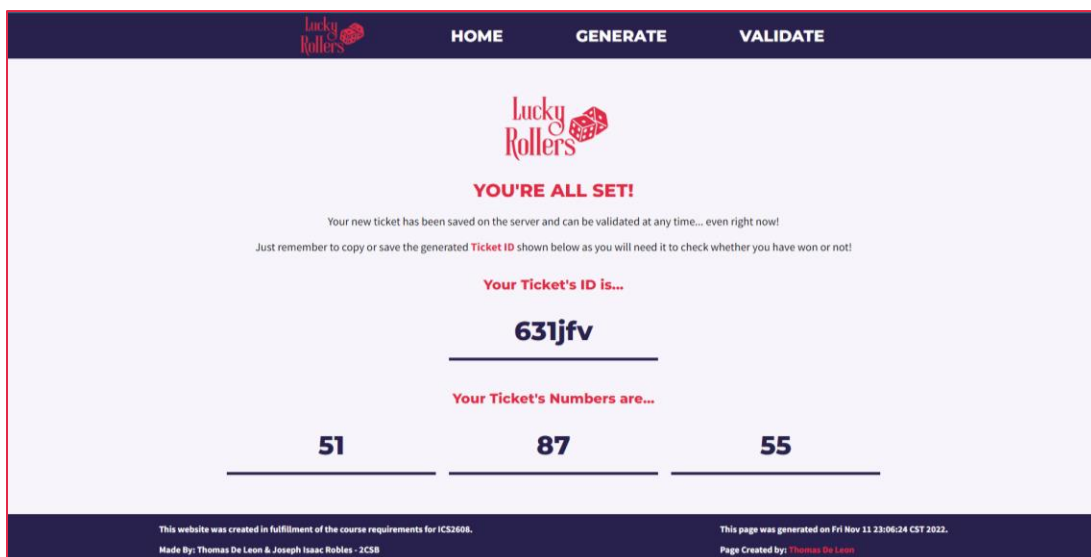

Figure 5: The Generate Success Page

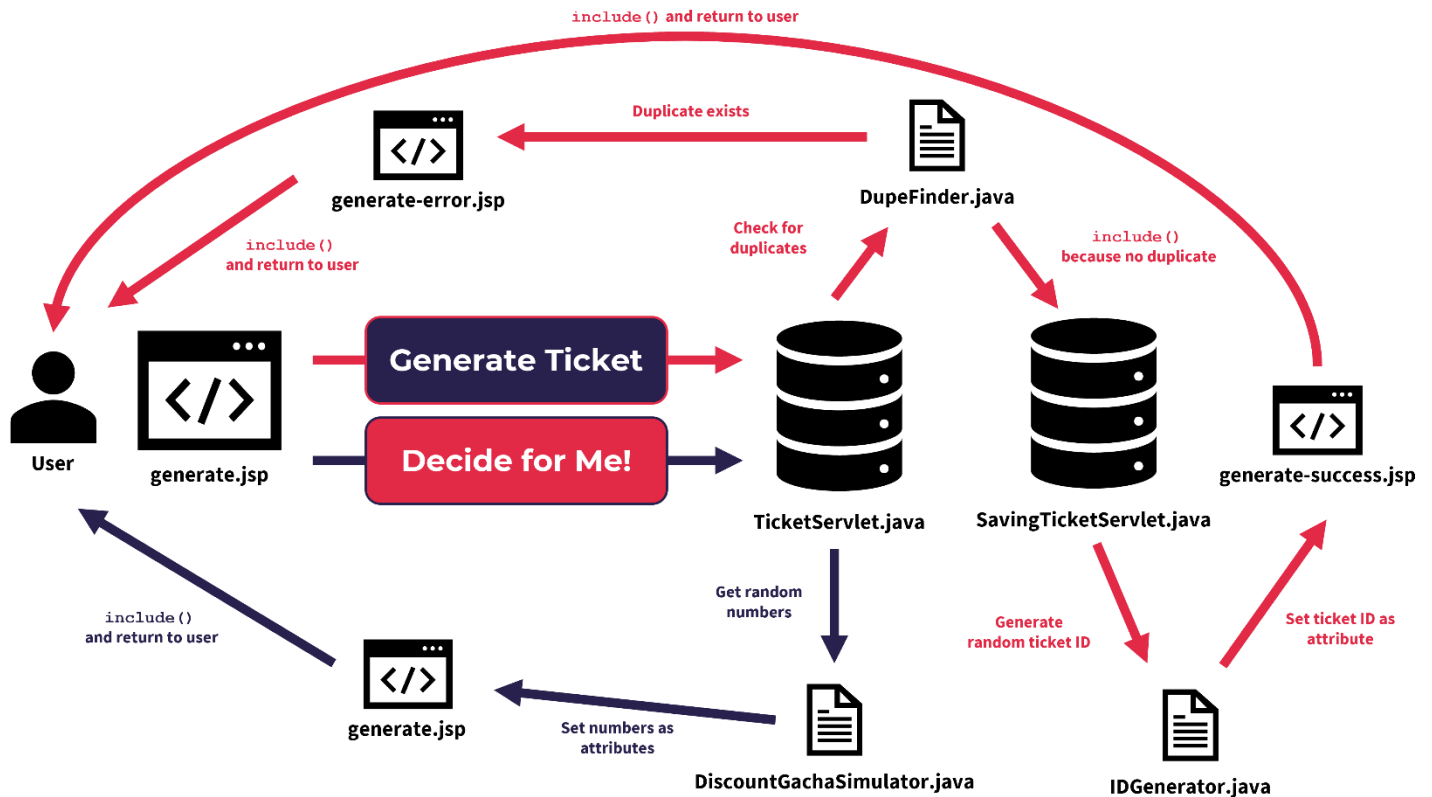Below is a diagram illustrating the entire ticket generation process along with other related subprocesses.

Diagram 2: Ticket Generation Process

Figure 6: The Validate Page with a specified Ticket ID

The **Validate** page is where users go to check whether their lottery tickets have won or not. To do so, they must enter the ID that was generated when their ticket was created for the application to know which numbers it needs to retrieve among the all the ticket numbers it has stored. After entering their ticket ID and clicking on **Check Ticket**, the application will invoke the `CheckTicketServlet` and do a series of tasks.

The servlet loads the ticket data stored on the application and checks if there is an entry for the ID specified by the user with its `TicketChecker` class. If it does not exist, it returns the `validate-error` page to inform the user that the ticket with the ID they have specified does not exist. When it does exist, it calls other methods from the same class to retrieve the specified ticket's numbers and draw the corresponding winning numbers for that ticket.

The application then matches the ticket numbers to the drawn numbers. A "match" occurs when a ticket number is the same as the winning number at a certain position. Finally, the application returns the corresponding results page depending on the number of matches: `validate-loss` for no matches, `validate-win` for 1/2 matches, and `validate-jackpot` for all 3 matches. The ticket with that specified ID is then deleted from the list of tickets.

Figure 7: One of the 3 Validate Results Pages – Loss

Below is a diagram illustrating the entire ticket validation process along with other related subprocesses.

forward() and return to user

generate-(result).jsp

determines corresponding...

Check Ticket

User

validate.jsp

include() and return to user

ticket with ID exists

CheckTicketServlet.java

calls methods from

validate-error.jsp

processes specified ticket ID

calls methods from

ticket with ID does not exist

loadTicket()

TicketChecker.java

loads numbers of specified ticket

gets random winning numbers
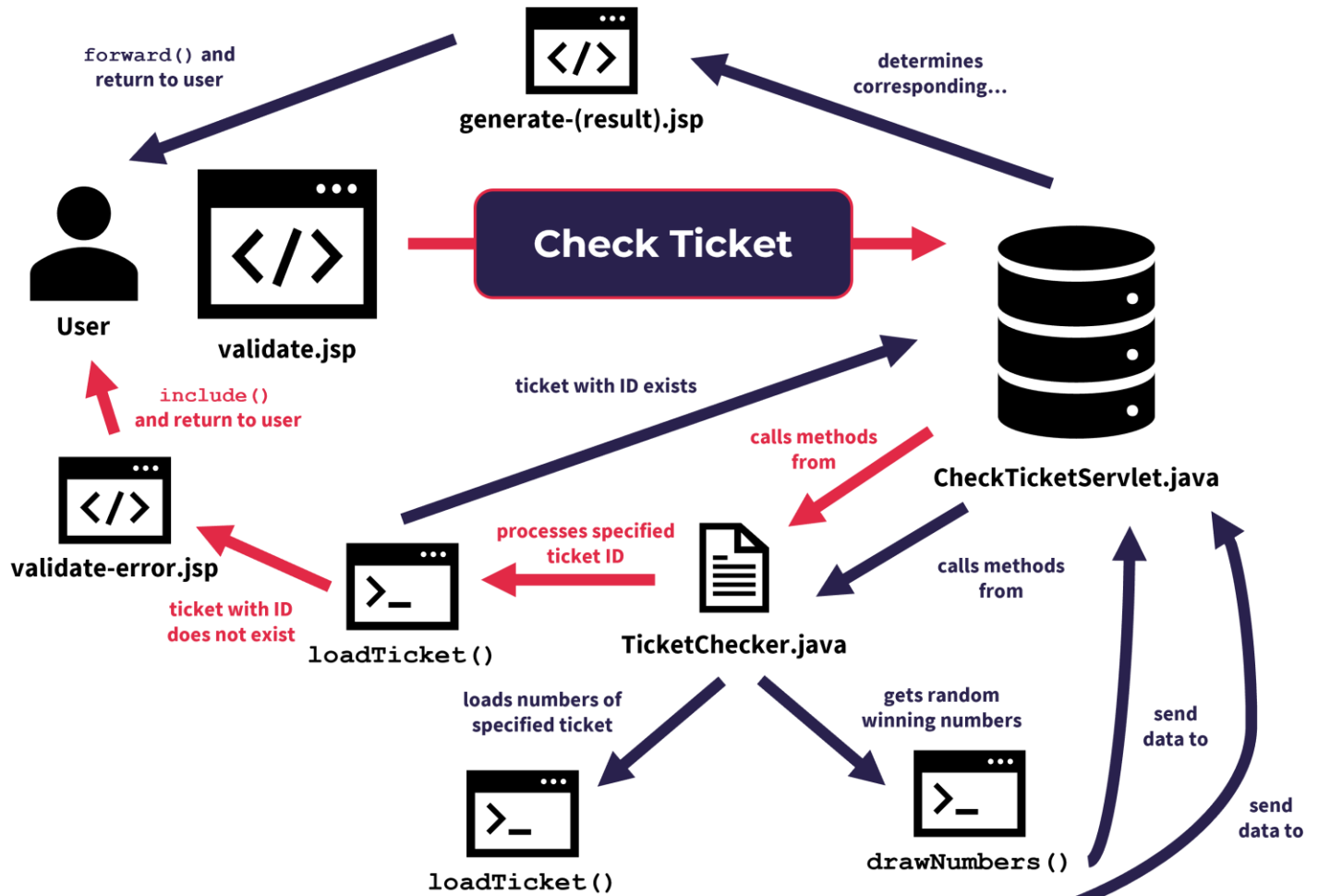
send data to

loadTicket()

drawNumbers()

send data to

Diagram 3: Ticket Validation Process