

Backend logic analysis before testing begins

General understanding

The game works through the Pragmatic Play backend service, which processes user actions (spins, winnings, balance updates) through a set of REST requests.

- All basic requests are sent to the domain:
<https://demogamesfree.pragmaticplay.net/gs2c/>

This is the server part, which is responsible for the game logic, balance and settings.

- Telemetry data is sent separately to a third-party service:
<https://clctr.ltguevmavv.com/collect>

It is only used to collect statistics and does not affect gameplay.

Main backend endpoints

Request type	Endpoint	Appointment
stats.do	/gs2c/stats.do?mgckey=<SESSION_KEY> (GET)	Sent on the game loads.
doSpin	/ge/v3/gameService (POST)	Spins the reels, places a bet, and receives the game result.
doCollect	/ge/v3/gameService (POST)	Confirms the winnings received and updates the balance.
reloadBalance	/reloadBalance.do (GET)	Returns the user's current balance after spins or page reloads.
saveSettings	/saveSettings.do (POST)	Save user settings (sound, game speed, bet size, etc.).
announcements/unread	/announcements/unread/ (GET)	Checks for new messages or system announcements.
promo/frb/available	/promo/frb/available/ (GET)	Checks for active bonuses (Free Rounds, promotions).
collect	https://clctr.ltguevmavv.com/collect (POST)	Sends technical information about the game's performance (response time, latency, browser data).

Session logic

All requests contain the parameter:

`mgckey=stylename@generic~SESSION@<UUID>`

This is a unique session key that identifies the player.

The key is passed in every request - without it, the server will not process the action.

Since the parameter is visible directly in the request (and not in a cookie or header), it can theoretically be replaced manually.

This is a potential risk area if the server does not check the validity or expiration of the session.

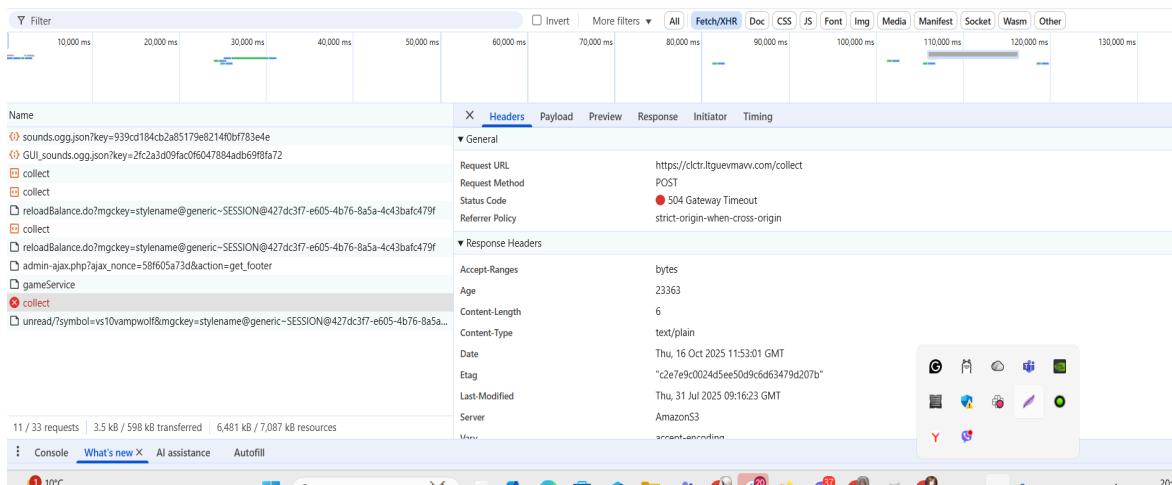
Bug reports

Bug 1- Telemetry POST Occasionally Returns 504 Timeout

Accepted result:

During gameplay, telemetry request occasionally fails with 504 Gateway Timeout instead of expected 202 Accepted.

Telemetry request (POST https://clctr.ltguevmavv.com/collect) returns 504
Gateway Timeout



Expected result:

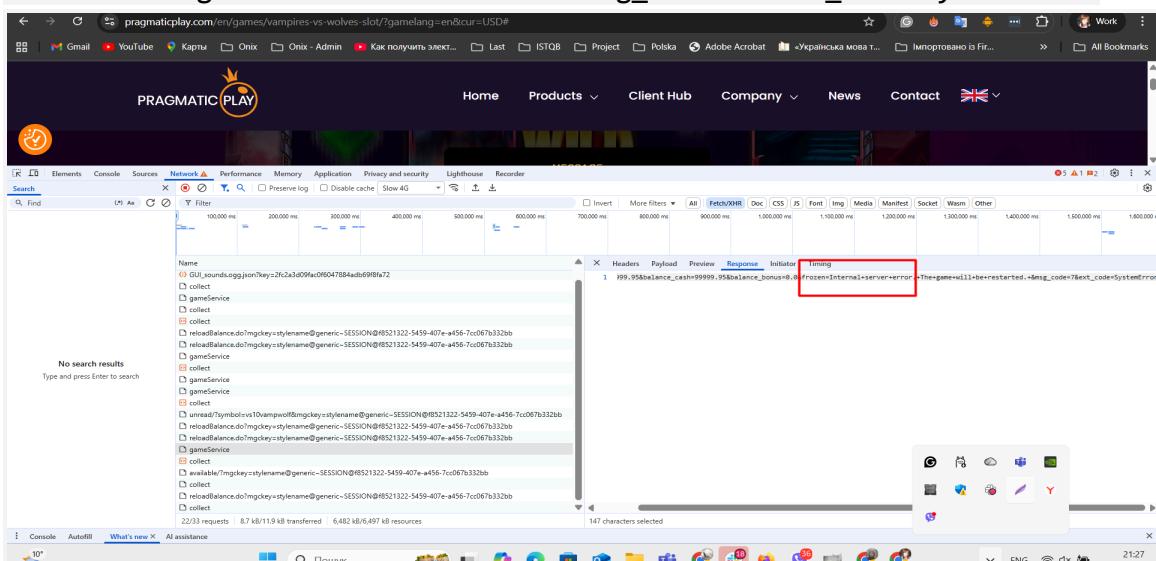
Telemetry service (POST <https://clctr.ltguevmavv.com/collect>) should always respond with **HTTP 202**

Bug 2 - Session freeze (Internal Server Error)

Accepted result:

When the old `doSpin` request is sent via Postman or another client using the same `mgckey` but with the previous index/counter, a 200 OK response is returned by the server:

balance=99999.95&balance_cash=99999.95&balance_bonus=0.0&frozen=Internal+server+error.+The+game+will+be+restarted.&msg_code=11&ext_code=SystemError



Pay attention The same issue bug if the request is sent via Postman with the previous index/counter and then the user returns to the website.

Expected result:

Server rejects duplicate or out-of-order requests without affecting the session.

Bug 3 - Duplicate spin request with the same index and counter is processed as the new spin instead of being ignored or replayed.

- POST <https://demogamesfree.pragmaticplay.net/gs2c/ge/v3/gameService>

Steps for reproducing

1. Start the game “Vamp Wolf”.

2. Send a spin request:

action=doSpin&symbol=vs10vampwolf&c=0.01&l=10&sInfo=n&index=2&counter=3

A screenshot of the Postman application interface. The 'Payload' tab is selected. Under 'Form data', there is a single entry: 'action=doSpin&symbol=vs10vampwolf&c=0.01&l=10&sInfo=n&index=2&counter=3'. The entire URL and payload are highlighted with red boxes.

→ Server responds correctly (index=2, balance=99999.90, ntp=-0.10).

3. Send the next spin request manually via Postman:

index=3&counter=5

A screenshot of the Postman application interface. The 'Body' tab is selected. Under 'x-www-form-urlencoded', there are seven fields: 'c' (0.01), 'l' (10), 'sInfo' (n), 'index' (3), 'counter' (5), 'repeat' (0), and 'mgckey' (stylename@generic~SESSION@e200eb86-c6c7-400e-b53d-61d99ec57e4c). The 'index' and 'counter' fields are highlighted with red boxes. The URL is https://demogamesfree.pragmaticplay.net/gs2c/ge/v3/gameService.

→ Server responds correctly (balance=99999.80, ntp=-0.20)

A screenshot of the Postman application interface. The 'Test Results' tab is selected. The response body shows a JSON object with various keys like tw, balance, index, etc. The 'tw' field has a value of 0.00 and the 'balance' field has a value of 99,999.80. The 'index' field is set to 3 and the 'counter' field is set to 5. The URL is https://demogamesfree.pragmaticplay.net/gs2c/ge/v3/gameService.

4. Resend the same spin request again with the same index=3 and counter=5 (from the game client).

A screenshot of the Postman application interface. The 'Payload' tab is selected. Under 'Form data', there is a single entry: 'action=doSpin&symbol=vs10vampwolf&c=0.01&l=10&sInfo=n&index=3&counter=5&repeat=0&mgckey=stylename@generic~SESSION@e200eb86-c6c7-400e-b53d-61d99ec57e4c'. The 'index' and 'counter' fields are highlighted with red boxes. The URL is https://demogamesfree.pragmaticplay.net/gs2c/ge/v3/gameService.

5. Observe that the response contains an updated counter=7 and a new stime value, instead of being treated as a duplicate request.

A screenshot of the Postman application interface. The 'Test Results' tab is selected. The response body shows a JSON object with updated values. The 'tw' field has a value of 0.00 and the 'balance' field has a value of 99,999.80. The 'index' field is now 3 and the 'counter' field is 7. A new 'stime' field is present with a value of 1760730145117. The URL is https://demogamesfree.pragmaticplay.net/gs2c/ge/v3/gameService.

6. The game then continues to process the next spin normally (index=4, counter=8).

A screenshot of the Postman application interface. The 'Payload' tab is selected. Under 'Form data', there is a single entry: 'action=doSpin&symbol=vs10vampwolf&c=0.01&l=10&sInfo=n&index=4&counter=8&repeat=0&mgckey=stylename@generic~SESSION@e200eb86-c6c7-400e-b53d-61d99ec57e4c'. Two red arrows point to the 'index' and 'counter' fields, which are highlighted with red boxes. The URL is https://demogamesfree.pragmaticplay.net/gs2c/ge/v3/gameService.

Actual Result:

The backend processes a repeated spin request with the same `index` and `counter` as a **new spin** and returns a response with a different `counter` and `stime`.

This can cause:

- double bet deduction,
- inconsistent player balance,
- desynchronization between client and server.

Expected Result:

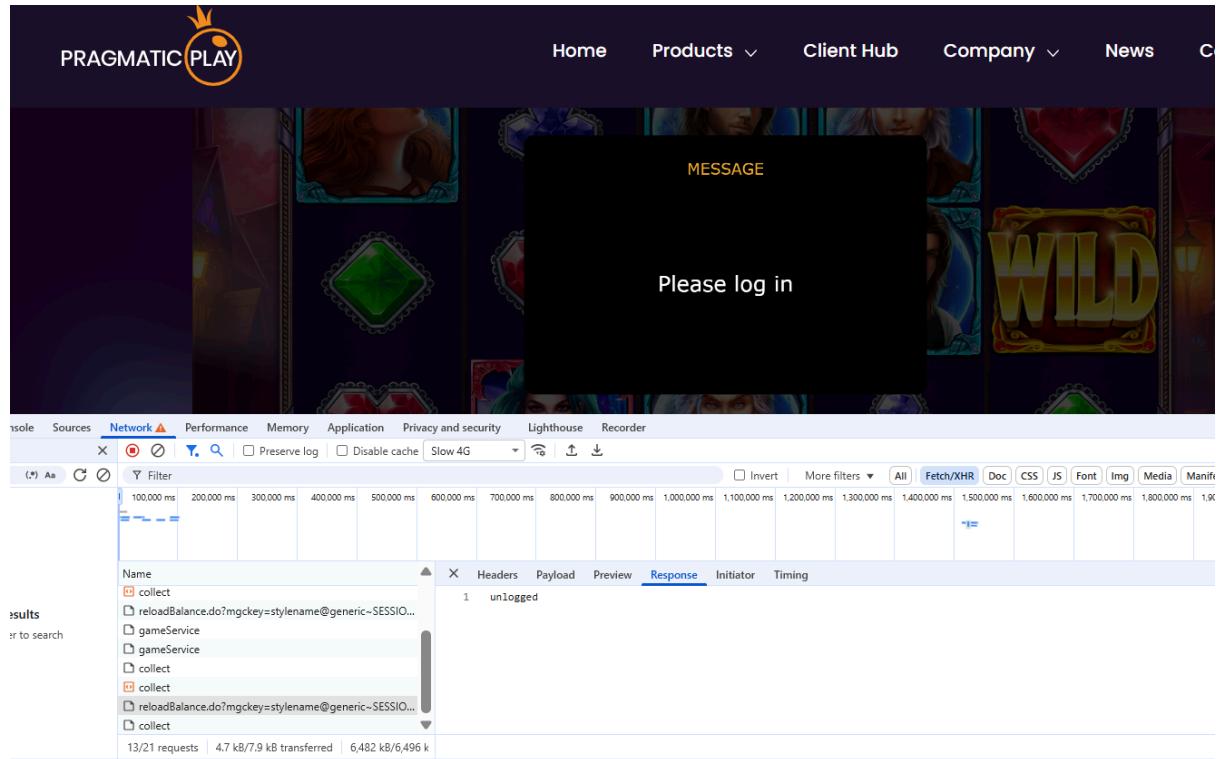
The backend should recognize repeated requests (same `index` and `counter`) as **duplicates** and either:

- return the **same response** as before (replay protection) OR
- return an error / flag such as “`duplicate request detected`”, without changing the game state or balance.

Bug 4 - Misleading “Please log in” on demo session expiry

Actual Result:

When the game session expires in demo mode, the backend returns a generic “*Please log in*” response, causing misleading behavior on the client side.



Expected Result:

When the demo session expires, the server should return a clear message such as "Session expired, please reload the page" or "Demo session expired, please reload", rather than "Please log in" which implies real account authentication.

Bug 5 - Endpoint does not validate session key (mgckey)

- GET /announcements/unread
- GET /promo/frb/available
- POST /saveSettings.do

Actual Result:

Endpoint does not validate session key (mgckey) and returns 200 OK even without authentication. It is possible to send requests with an invalid, expired, or missing mgckey.

The screenshot shows a Postman request for `https://demogamesfree.pragmaticplay.net/gs2c/announcements/unread/?symbol=vs10vampwolf`. The 'Params' tab is selected, showing 'symbol' with value 'vs10vampwolf'. The 'Body' tab is selected, showing XML response content:

```
1 <EmptyGetUnreadAnnouncementsResponseDTO>
2   <error>0</error>
3   <description>OK</description>
4   <announcements/>
5 </EmptyGetUnreadAnnouncementsResponseDTO>
```

The status bar indicates **200 OK**, 235 ms, 665 B.

The screenshot shows a Postman request for `https://demogamesfree.pragmaticplay.net/gs2c/promo/frb/available/`. The 'Params' tab is selected, showing 'mgckey' with value 'stylename@generic~SESSION@d6212206-d553-4793-...'. The 'Body' tab is selected, showing XML response content:

```
1 <PromoDisabledResponse>
2   <error>4</error>
3   <description>Promo system is disabled</description>
4 </PromoDisabledResponse>
```

The status bar indicates **200 OK**, 247 ms, 637 B.

The screenshot shows a Postman request for `https://demogamesfree.pragmaticplay.net/gs2c/saveSettings.do`. The 'Body' tab is selected, showing x-www-form-urlencoded data:

id	vs10vampwolf
settings	SoundState=true_true_true_false_false;FastPlay=false;...
mgckey	stylename@generic~SESSION@16e1825e-2230-48ab...

The status bar indicates **200 OK**, 295 ms, 780 B.

Expected Result:

Each endpoint (`/announcements/unread`, `/promo/frb/available`, `/saveSettings.do`) must validate mgckey.

Requests with missing, invalid, or expired keys should return unlogged.

Bug 6 - The 500 Internal Server Error returns after changing the HTTP method for all endpoints

Actual result:

Server returns 500 Internal Server Error when unsupported HTTP methods (GET, PUT, PATCH, DELETE) are sent to endpoints such as `/ge/v3/gameService`, `/reloadBalance.do`, `/saveSettings.do`, `/announcements/unread/`, `/promo/frb/available/`.

Instead of rejecting invalid methods, the backend fails internally without a clear validation message.

- POST `/ge/v3/gameService` from POST to GET/DELETE/PUT/PATCH

DELETE https://demogamesfree.pragmaticplay.net/gs2c/ge/v3/gameService

Params Authorization Headers (9) Body Scripts Settings Cookies

Body (none) form-data x-www-form-urlencoded raw binary GraphQL

<input checked="" type="checkbox"/> counter	5
<input checked="" type="checkbox"/> repeat	0

Body Cookies Headers (9) Test Results | 500 Internal Server Error 252 ms 380 B Save Response ...

Raw Preview Debug with AI

1 500

- GET `/reloadBalance.do` - change to PATCH, PUT, DELETE

PATCH https://demogamesfree.pragmaticplay.net/gs2c/reloadBalance.do?mgckey=stylename@generic~SESSION@883c55fe-67e7-4e2b-8de3-75120ac4f02

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params

<input checked="" type="checkbox"/> Key	Value	Description	... Bulk Edit
<input checked="" type="checkbox"/> mgckey	stylename@generic~SESSION@883c55fe-67e7-4e2b-8de3-75120ac4f02		
Key	Value	Description	

Body Cookies Headers (12) Test Results | 500 Internal Server Error 235 ms 463 B Save Response ...

HTML Preview Debug with AI

1

- GET `/announcements/unread/` - change to PATCH, PUT, DELETE

POST https://demogamesfree.pragmaticplay.net/gs2c/announcements/unread/?symbol=vs10vampwolf&mgckey=stylename@generic~SESSION@883c55fe-67e7-4e2b-8de3-75120ac4f02

Params Authorization Headers (8) Body Scripts Settings

Query Params

<input checked="" type="checkbox"/> Key	Value	Description	...
<input checked="" type="checkbox"/> symbol	vs10vampwolf		
<input checked="" type="checkbox"/> mgckey	stylename@generic~SESSION@883c55fe-67e7-4e2b-8de3-75120ac4f02		
Key	Value	Description	

Body Cookies Headers (11) Test Results | 500 Internal Server Error 323 ms 424 B Save Response ...

Raw Preview Debug with AI

1

- The same situation to GET `/promo/frb/available/`, POST `/saveSettings.do`

Expected Result:

Server must reject unsupported methods with a `405 Method Not Allowed` or a clear error response. No 500 Internal Server Error should occur.

Bug 7 - Non-secure HTTP request accepted – no HTTPS enforcement

Actual result:

Server accepts non-secure (HTTP) requests and returns 200 OK instead of rejecting or redirecting to HTTPS. The bug is repeated for all endpoints.

The screenshot shows the Postman interface with a POST request to `http://demogamesfree.pragmaticplay.net/gs2c/ge/v3/gameService`. The Headers tab is selected, showing 'Content-Type: application/x-www-form-urlencoded'. The Body tab contains a table with data: index (4), counter (7), repeat (0), and mgckey (stylename@generic~SESSION@331dd012-d4ae-43fc-...). The Response tab shows a 200 OK status with a response time of 328 ms and a body size of 781 B. The response body is a JSON object with fields tw, balance, balance_cash, reel_set, balance_bonus, na, c, l, stime, and s. A red arrow points from the URL bar to the Headers tab, and another red arrow points from the '200 OK' status to the response body.

Expected Result:

Server should reject all non-secure (HTTP) requests with 400 Bad Request or 301 Redirect to HTTPS.

Only HTTPS traffic must be processed successfully.

Bug 8 - the 500 Internal Server Error when sending with the empty body

Actual result:

The 500 Internal server error returns after sending the request with empty body.

The same situation after sending body with the wrong format using application/json instead of form-data.

- POST /ge/v3/gameService

The screenshot shows the Postman interface with a POST request to `http://demogamesfree.pragmaticplay.net/gs2c/ge/v3/gameService`. The Headers tab is selected, showing 'Content-Type: application/x-www-form-urlencoded'. The Body tab shows a message: 'This request does not have a body'. The Response tab shows a 500 Internal Server Error with a response time of 332 ms and a body size of 380 B. The response body is a simple '500'. A red arrow points from the 'This request does not have a body' message to the 500 Internal Server Error status.

The screenshot shows the Postman interface with a POST request to `http://demogamesfree.pragmaticplay.net/gs2c/ge/v3/gameService`. The Headers tab is selected, showing 'Content-Type: application/json'. The Body tab contains a JSON object with fields action (doSpin), symbol (vs10vampwolf), c (0.01), l (10), sInfo (n), index (4), counter (7), repeat (0), and mgckey (stylename@generic~SESSION@eaf39208-5c26-4fde-9293fea844e15132). The Response tab shows a 500 Internal Server Error with a response time of 332 ms and a body size of 380 B. The response body is a simple '500'. A red arrow points from the JSON body to the 500 Internal Server Error status.

Expected Result:

Server should handle missing or malformed request bodies gracefully — returning a 400 Bad Request or 422 Unprocessable Entity error.

The system must not return 500 Internal Server Error or crash.

Bug 9 - The Internal Server error when required parameter symbol is missing.

Actual Result:

Server returns 500 Internal Server Error when the required parameter **symbol** is missing.
No descriptive validation message is provided in the response body.

- GET /announcements/unread/

The screenshot shows a Postman request for `https://demogamesfree.pragmaticplay.net/gs2c/announcements/unread/`. The 'symbol' parameter is present with value `vs10vampwolf`, but the 'mgckey' parameter is missing. The response status is **500 Internal Server Error**.

Expected Result:

When required parameters (e.g. **symbol**) are missing, the server should return a structured 4xx validation error (e.g. "error": "missing parameter") without crashing or returning 500.

Bug 10 - the 200 OK status after sending the request without params

Actual Result:

Server returns 200 OK even when the request contains no parameters or payload.
No validation error or warning is triggered.

- GET /promo/frb/available

The screenshot shows a Postman request for `https://demogamesfree.pragmaticplay.net/gs2c/promo/frb/available/`. The 'mgckey' parameter is missing. The response status is **200 OK**, and the response body is XML indicating the system is disabled.

Expected Result:

If no parameters are provided, the backend should respond with a 4xx error (400 Bad Request) indicating invalid or missing input. It should never return 200 OK.

Bug 11 - No validation in saveSettings endpoint

Actual Result:

The server accepts and returns any settings values (`saveSettings`) without validation - it responds with 200 OK and echoes the data sent. This is a lack of server validation/sanitization, which leads to stability and security risks (should return 4xx for invalid or malicious input).

- POST `/saveSettings.do`

The screenshot shows a Postman interface with the following details:

- Method: POST
- URL: <https://demogamesfree.pragmaticplay.net/gs2c/saveSettings.do>
- Headers: (9)
- Body: x-www-form-urlencoded
- Table Data:

Key	Value	Description
id	vs10vampwolf	
settings	SoundState=1_1_1_0_0;FastPlay=maybe;Intro=TRUE;St....	
mgckey	stylename@generic~SESSION@16e1825e-2230-48ab-...	
- Response: 200 OK (291 bytes)
- Body Content: SoundState=1_1_1_0_0;FastPlay=maybe;Intro=TRUE;StopMsg=zero;TurboSpinMsg=none;...

Expected Result:

The `/saveSettings.do` endpoint should validate incoming user preferences. Invalid, out-of-range, or malicious values must trigger a **4xx validation error**; the server must not echo or store arbitrary data.

Bug 12 - The session is frozen after sending endpoint with invalid params

Actual Result:

Negative input validation: server must return 4xx on invalid parameters; currently it accepts/echoes values (or returns 200) - validation missing.

Session stability: sending malformed/out-of-order requests causes sessions to become **frozen** (Internal Server Error); expected behaviour is to reject input without corrupting session.

- POST `/ge/v3/gameService`

The screenshot shows a Postman interface with the following details:

- Method: POST
- URL: <http://demogamesfree.pragmaticplay.net/gs2c/ge/v3/gameService>
- Headers: (9)
- Body: x-www-form-urlencoded
- Table Data:

c	5
I	10
sinfo	n
index	-1
counter	2
repeat	0
mgckey	stylename@generic~SESSION@f9a2...
- Response: 200 OK (319 ms, 669 B)
- Body Content: balance=99999.9&balance_cash=99999.9&balance_bonus=0.0&frozen=Internal+server+error.+The+game+will+be+restarted.++&msg_code=11&ext_code=SystemError

Expected Result:

Requests containing invalid parameter types or negative values should be rejected with 4xx errors, without affecting active sessions.

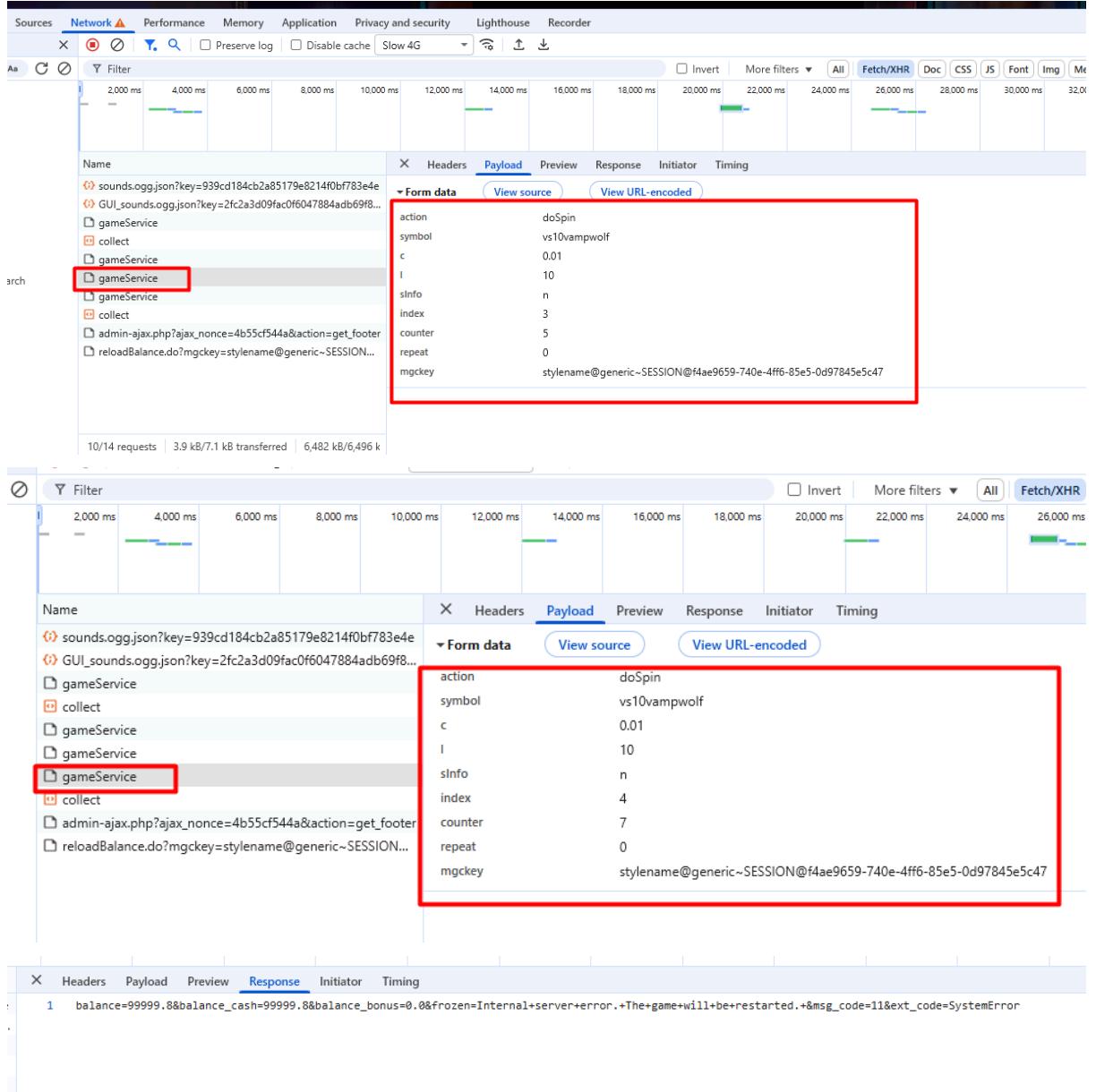
The backend must maintain session integrity and avoid Internal Server Errors.

Bug 13 - The session is frozen after sending the duplicate request from the Network.

Actual result:

When duplicate doSpin or doCollect requests are sent from the network, the backend processes them repeatedly instead of ignoring duplicates. As a result, the session becomes frozen, and further requests stop being processed correctly.

- POST doSpin /ge/v3/gameService
- POST doCollect /ge/v3/gameService



Expected Result:

Backend should detect and ignore **duplicate network requests** (for doSpin or doCollect) to prevent double balance updates or crashes.

Only one request should be processed; repeated ones should be ignored or return the same response safely.

Test Design Techniques Used

During backend and functional testing of *the Vampires vs Wolves game*, the following ISTQB test design techniques were applied:

- **Equivalence Partitioning (EP)**

Used to validate API parameters and input data ranges (e.g. bet size **c**, lines **1**, valid/invalid **mgckey**).

Applied in tests verifying correct and incorrect parameter values, missing parameters, and invalid types (Bugs 9-12).

- **Boundary Value Analysis (BVA)**

Applied to test edge values for numeric inputs (e.g. minimum/maximum bet amount, line count, session lifetime).

Used in spin and balance validation scenarios where bet or balance limits are reached.

- **Decision Table Testing**

Used to verify combinations of valid/invalid conditions, such as request method + authentication + mgckey state.

For example, verifying responses for valid/expired/missing session keys (Bugs 2, 5) or method change behavior (Bug 6).

- **State Transition Testing**

Applied to verify correct state changes between “Idle → Spinning → Win → Collect → Idle” and session transitions.

Used in replay protection, duplicate requests (Bugs 2, 3, 13), and session expiration (Bug 4).

- **Error Guessing**

Based on tester intuition and experience to find unhandled errors like internal server errors (500), timeouts, or incorrect error messages.

Applied in tests for invalid HTTP methods, empty bodies, and non-secure HTTP requests (Bugs 6-8).

- **Use Case Testing**

Used to simulate realistic player behavior: spin, win, collect, reload balance, and adjust settings.

Ensured backend logic worked correctly across normal game flow and promo availability scenarios.

- **Exploratory Testing**

Performed to investigate unexpected backend behavior during concurrent sessions and replayed requests.

Helped uncover race conditions, duplicate handling issues, and inconsistent balance updates (Bugs 2, 3, 13).