Software Test Plan (STP) – All Scenarios

Section	Details
Test Plan ID	STP_SLOTGAME_ALL_001
Project	Casino Slot Machine Microservices – QA Automation Exam
Objective	To validate positive, negative, edge-case, and security scenarios of the slot machine game system across all interconnected microservices.
Scope	Covers functional and security test flows including winning and losing rounds, invalid inputs, min/max bets, SQL injection, unauthorized access, exact balance conditions, full game logic with mock validation, response delays, and concurrent sessions.
Out of Scope	Authentication logic not implemented in backend, UI testing, and performance benchmarking.
Test Types	Functional, Integration, Concurrency, Edge Case Validation, Negative Testing, Security Testing
Tools & Frameworks	Python, Pytest, Mock Helper, Requests, Socket
Test Environment	Local with mocked or real API responses. CI-compatible.
Test Data	User ID, balances, transaction IDs, bet amounts, invalid payloads, malformed tokens, large numbers.
Assumptions	Mocks simulate APIs where needed; real endpoints used for security validation.
Risks	Security risks might not fully simulate production environment.
Entry Criteria	Mockable or accessible API, test framework setup, test data ready.
Exit Criteria	All test cases (positive, negative, edge, and security) pass with expected results.
Deliverables	STP, STD, test scripts, README.

Software Test Design (STD) – Positive Test Cases

Test Case ID	Title	Type	Description	Expected Result
TC_POS_01	Full Game Win Flow	Positive	Validates the full game flow when the user wins, including balance deduction, spin result, payout, and notification.	All steps succeed and final balance is updated correctly.

Test Case ID	Title	Type	Description	Expected Result
TC_POS_02	Full Game Loss Flow	Positive	Simulates a full game flow where the user loses. Verifies deduction, LOSE spin result, and notification.	Balance is deducted, spin returns LOSE, and user is notified.
TC_POS_03	Extreme Bet Values	Positive / Edge	Tests behavior for min, max, and beyond max bets. Validates deduction, spin result, and payout.	Each bet returns expected result; balances are updated accurately.
TC_POS_04	Spin Response Time Delay Handling	Positive / Resilience	Simulates delayed response from the spin endpoint and verifies system resilience under latency.	Spin still succeeds with WIN outcome; payout and notification work.
TC_POS_05	Multiple Parallel Successful Rounds	Positive / Concurrency	Runs 3 game rounds in parallel to test system's ability to handle concurrent user interactions.	All parallel rounds succeed without conflict or interference.

Software Test Design (STD) – Negative Test Cases

Test Case ID	Title	Type	Description	Expected Result
TC_NEG_01	Insufficient Balance	Negative	User attempts to place a bet when their balance is lower than the bet amount.	400 Bad Request with error message 'Insufficient balance'.
TC_NEG_02	Invalid Transaction ID	Negative	Spin is triggered using a non-existent transaction ID.	404 Not Found with error message 'Transaction not found'.
TC_NEG_03	Missing User ID	Negative	Place bet request is sent without a user ID.	400 Bad Request with error message 'Missing userId'.
TC_NEG_04	Payout Without Win	Negative	Attempt to payout a win amount of zero.	403 Forbidden with error message 'No win to payout'.
TC_NEG_05	Duplicate Transaction ID	Negative	Same transaction ID is reused for multiple place_bet calls.	409 Conflict with error message 'Transaction already exists'.

Test Case ID	Title	Type	Description	Expected Result
TC_NEG_06	Empty Notification Message	Negative	Notification is triggered with an empty message string.	400 Bad Request with error message 'Message cannot be empty'.
TC_NEG_07	Spin Before Place Bet	Negative	Spin is triggered before any place_bet call has been made.	404 Not Found with error message 'Bet not found'.

Software Test Design (STD) – Edge Case

Why These Edge Cases Were Selected:

These edge cases represent realistic and sensitive boundary scenarios often mishandled in systems involving financial logic. If not tested, such situations may result in:

- Bet acceptance failures when balance equals bet
- Incorrect reel structure or format
- Incorrect final balance after bet, spin, and payout
- Partial logic validation that doesn’t reflect true integration

Edge Cases:

Test Case ID	Title	Type	Description	Expected Result
TC_EDGE_01	Bet With Exact Balance & 3-Reel Validation	Edge Case	Validates full game flow when user balance equals bet amount. Checks balance becomes zero, 3 reels, payout, and structure.	WIN is returned with exactly 3 reels, payout succeeds, balance becomes 0, notification is sent.

Software Test Design (STD) – Full Game Flow

Test Case ID	Title	Type	Description	Expected Result
TC_FLOW_01	Full Game Flow (Win/Lose) With Mock Logic	Full Integration	Validates end-to-end game logic using mocked API: verifies WIN/LOSE paths, structure, balance correctness, and message.	Game completes with either WIN or LOSE. Final balance, payout, and notification are correct.

Software Test Design (STD) – Security Test Cases

Test Case ID	Title	Type	Description	Expected Result
TC_SEC_01	Endpoint Availability	Security / Smoke	Ensure that all public endpoints are reachable and return a valid response	Status codes in (200, 400, 403, 422) depending on input
TC_SEC_02	Unauthorized Access	Security	Send requests without authorization headers	Server returns 401 or 403 for each restricted endpoint
TC_SEC_03	SQL Injection in userId	Security	Attempt SQL injection through userId field	Server returns 400/422 with error message or input validation failure
TC_SEC_04	Large Bet Amount	Security	Send extremely large numeric values in request	Server responds with validation error: 400 or 422