Test ID	Tile	Vulnerability	Exploit method	Impact	Risk	Remediation
1	Score board	Information disclosure (public scoreboard)	Guess the URL /#/score-board	Shows users' scores and completed challenges. Helps an attacker learn app state or user activity but does not directly give access to accounts or data.	Low	Require authentication or authorization to view the scoreboard, hide or anonymize sensitive details and avoid exposing internal endpoints
2	DOM XSS	DOM-based Cross-Site Scripting — user input is inserted into the page DOM without proper encoding or sanitization.	Enter a script payload in the search box (for example <script>alert(1)</script>)	Can steal session tokens, perform actions as the user, or run malicious code	High	Encode the output before showing to the page, validate and sanitize input server-side
3	XSS Bonus Payload	Cross-Site Scripting (reflected user input is not properly encoded)	Enter a script payload (e.g. <script>alert(1) </script>) into the search box and submit	Attacker can run JavaScript in other users' browsers — steal session cookies, perform actions as the user, or show fake UI.	High	Encode the output before showing to the page, validate and sanitize input server-side
4	Privacy Policy	Non information page intended to be public	Open the /privacy-policy page in a browser	Displays the site's privacy data (e.g., admin emails, etc,..).	Low	No security fix needed if public. Verify the page contains no sensitive data; remove any internal links, credentials, or debug information.
5	Bully Chatbot	Broken anti-automation (insufficient bot protection)	Automate repeated chat messages or spam the chatbot until it returns a coupon or reward.	Attacker can obtain free coupons.	Medium	Apply rate limits and per-user quotas, add CAPTCHA or challenge before issuing coupons, require authentication/verification for rewards, implement server-side abuse detection and logging, and validate coupon issuance on the server.
6	Confidential Document	Sensitive data exposure	Change or guess file URLs to download an unprotected document.	Leaks confidential files	High	Restrict file paths, require login/authorization to download, validate file requests server-side.
7	Exposed Metrics	Security misconfiguration	Visit the /metrics endpoint and read server stats	Reveals internal server info and performance data that can help attackers	Medium	Require authentication for metrics, bind metrics to localhost, or hide/remove the endpoint in production.
8	Missing Encoding	Improper input validation / output encoding	Find the image URL in the page source and modify the broken encoded value	May allow XSS or show unexpected HTML, attackers can run scripts, steal cookies, or alter page content.	Medium	Encode and escape all output (including URLs) and validate inputs server-side.
9	Error Handling	Security misconfiguration	Trigger errors (upload wrong file, send bad input) to show stack traces or secrets.	Reveals stack traces, internal paths, or secrets useful for attackers.	Medium	Catch errors, log details server-side
10	Repetitive Registration	Improper input validation / broken registration rules	Enter different values in the password and repeat password fields and submit; the server accepts the registration even though they don't match.	Creates accounts with unknown or insecure credentials, enables spam/fake accounts, and can confuse or bypass account controls.	Medium	Check on the server that password and confirmation match, require fields, enforce strong-password rules, add rate limits and CAPTCHA, and require email verification before activating accounts.
11	Web3 Sandbox	Misconfigured Web3 sandbox	Visit the /web3-sandbox	Unauthorized blockchain actions, funds loss, or contract abuse.	High	Disable sandbox in production, restrict access, require authentication and strict isolation.
12	Zero Stars	Improper input validation	Submit rating value 0 or negative through API with Burp Suite.	Breaks business logic	Low	Validate rating values server-side; only allow 1–5

13	Login Admin	SQL injection	Input SQL payload like ' OR 1=1-- in login fields.	Bypass authentication, gain admin access.	High	Use prepared statements/ORM parameter binding; never concatenate SQL
14	Admin Section	Broken access control	Visit /#/administration directly without proper rights.	Unauthorized access to admin functions and data.	High	Enforce server-side role checks for admin pages and APIs; deny unauthorized requests.
15	View Basket	Insecure direct object reference (IDOR)	Change basketId in API call to view or edit another user's cart.	Exposes or alters other users' shopping data.	High	Verify ownership of basket IDs server-side before returning data.
16	Deprecated Interface	Improper file upload validation / attachment handling error	Upload a .xml file even though attachments should block .xml; the server crashes when it processes the file	Application crash	High	Enforce server-side file type and content validation
17	Empty User Registration	Improper input validation / missing required-field checks	Intercept the registration request with Burp Suite and send it with empty username or password; the server accepts and creates the account.	Creates invalid or ghost accounts, enables spam or bypasses, and may break authentication logic.	Low	Enforce server-side validation
18	Five-Star Feedback	Broken business logic	Use API to delete or change all 5-star reviews without permission	Data manipulation and reputation damage.	Low	Only allow the original reviewer or admin to modify feedback, enforce permissions server-side.
19	Login MC SafeSearch	SQL injection	Use crafted email like mc-safesearch@…' -- to tamper SQL query	Bypass login or leak data.	High	Parameterize SQL queries and sanitize inputs.
20	Meta Geo Stalking	Sensitive data exposure	Extract photo to find hints for security answers.	Reveals location data or answers to recovery questions.	Medium	Strip metadata from uploads and avoid using personal data as security questions.
21	NFT Takeover	Sensitive data exposure / cryptographic misuse	Find wallet keys in comments	Theft of NFTs or crypto assets.	High	Never store seed phrases in app data, secure keys properly, educate users, and remove exposed secrets.
22	Outdated Allowlist	Unvalidated redirect	Change to parameter to redirect users to malicious sites.	Phishing and user redirection attacks.	Medium	Use a whitelist of allowed redirect targets and validate to against it.
23	Visual Geo Stalking	Sensitive data exposure (image metadata)	Read photo metadata to guess answers to security questions.	Account takeover via guessed reset answers.	Medium	Strip metadata from uploads and avoid using personal data as security questions.
24	Login Jim	SQL injection	Use jim@juice-sh.op' -- to manipulate SQL and bypass checks	Unauthorized access.	High	Parameterize queries and validate inputs.
25	Login Bender	SQL injection	Use bender@juice-sh.op' -- to bypass login via SQL tampering.	Unauthorized access.	High	Parameterize queries and validate inputs.

26	Weird Crypto	Weak cryptography / poor key management	Crack weak tokens, insert md5 in Feedback input box.	Forged tokens, coupon abuse, or data tampering.	High	Use strong, standard cryptographic algorithms and protect keys securely.
27	Christmas Special	Blind SQL injection	Send crafted requests using product IDs to extract product data or manipulate the purchase flow and buy items.	Data leakage, unauthorized orders, price manipulation, and potential financial loss.	High	Use parameterized queries/prepared statements, validate and sanitize all inputs server-side
28	Forged Feedback	Broken authentication / insufficient server-side checks	Change user ID when submitting feedback to impersonate another user.	Fake reviews and reputation damage	Medium	Associate feedback with the logged-in user server-side and ignore client-sent user IDs.
29	Reset Jim's Password	Weak security question	Guess answer (e.g., “Samuel”) or find it in public data to reset password	Account takeover through insecure password reset.	High	Use secure reset tokens sent to verified email, MFA, and avoid personal questions.
30	Payback Time	Improper input validation	Send negative price or quantity in API to trigger refunds or incorrect balance.	Financial manipulation or unintended refunds	High	Validate numeric inputs server-side and disallow negative values.
31	CAPTCHA Bypass	Broken anti-automation	Bypass (Submit 10 or more customer feedbacks within 20 seconds.)	Automated abuse, spam, or fraud.	Medium	Validate CAPTCHA server-side, expire tokens after use, and rate-limit attempts
32	Security Policy	None — information meant to be public (security. txt)	Open /.well-known/security.txt (or /security.txt) in a browser to read the published contact and disclosure policy	Informational — shows security contact, PGP key, and disclosure preferences; only risky if it accidentally contains internal URLs, credentials, or sensitive details	Low	No fix needed if content is intentionally public
33	Expired Coupon	Client-side validation only	Override client Date or tamper JavaScript to accept expired coupons.	Use of expired coupons by valid it's date	High	Validate coupon dates and eligibility on the server.
34	Bjoern's Favorite Pet	security question answer is guessable/public	Find Bjoern’s pet name online or from metadata, use it to answer the “Forgot Password” question and reset his account password.	Account takeover — attacker gains full access to Bjoern’s account and any associated data or privileges.	High	Do not rely on easily guessable personal question
35	Database Schema	SQL Injection (UNION-based dump of SQLite schema)	Replace the parameter value with the payload apple')) UNION SELECT sql,2,3,4,5,6,7,8,9 FROM sqlite_master --	Reveals DB schema, making data exfiltration easier	High	Use prepared statements/parameterized queries, validate inputs, remove DB output from responses, run DB with least privilege.
36	Deluxe Fraud	Payment tampering / Broken transaction integrity	Intercept the payment request with a Burp and modify the payment method, amount, or recipient before it reaches the server	Attacker can get unauthorized upgrades or free items, cause incorrect charges, financial loss, and chargebacks	High	Validate all payment data on the server, use payment-gateway tokens (never trust client-side data)
37	GDPR Data Exposure	SQL injection	end a crafted query to /rest/products/search-> /rest/products/search?q=qwert')) +UNION+SELECT+id,email,password,4,5,6,7,8,9 +FROM+users+-- to UNION-inject and return records from the users table	Attacker can read user emails and passwords (or hashes), causing account takeover, GDPR violation, reputational damage, and legal fines	High	Use parameterized queries / prepared statements, validate and sanitize all inputs, restrict DB user privileges, never return sensitive fields in API responses, store passwords with strong hashing
38	Easter Egg	Null-byte / double-encoded path injection (file path truncation / arbitrary file read)	Try direct path to /ftp/eastere.gg%2500.md	Disclosure of hidden files or sensitive content; may allow reading arbitrary files if input is not normalized.	High	Normalize and canonicalize file paths server-side, reject null bytes and double-encoded sequences, enforce a strict allowlist for served files, require authentication for file access, disable directory listing, and validate/escape user input

39	Nested Easter Egg	Security by obscurity / exposed hidden resource	Open http://localhost:3000/the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg in a browser	Reveals internal messages or files developers left behind; may disclose hints, credentials, or sensitive comments	Medium	Remove sensitive content from public folders, require authentication for internal pages, avoid relying on secret URLs for security, and scan the repo for stray files before deployment.
40	Forgotten Developer Backup	Sensitive data exposure / backup file disclosure	Request http://localhost:3000/ftp/package.json.bak%2500.md by guessing backup filenames like package.json.bak to download developer backup files.	Leaks configuration, credentials, API keys, or dependency info; enables further attacks like credential misuse or privilege escalation.	High	Remove backup files from web root and source control; prevent serving .bak, .old, .swp files
41	Forgotten Sales Backup	Sensitive data exposure / backup file disclosure	Request http://localhost:3000/ftp/coupons_2013.md.bak%00.md	Leaks historical coupons, promo codes, pricing or sales data that attackers can reuse or analyze, attcker may reveal business tactics or sensitive info.	High	Remove backup files from web root and source control; prevent serving .bak, .old, .swp files
42	Login Bjoern	Hardcoded / exposed credentials	Decode the password or use the supplied credentials to log in (username bjoern.kimminich@googlemail.com / password : bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvaml=)	Account takeover of Bjoern's account; possible privilege abuse or access to sensitive data.	High	Remove hardcoded credentials, rotate the password immediately, require users to set strong unique passwords, store secrets securely
43	GDPR Data Theft	Weak email obfuscation causing identifier collisions (insecure data handling) — effectively a broken authorization / IDOR-like issue	Create an order, get Order ID, track it to see obfuscated email, register an email that obfuscates the same, then export data to receive the other user's order.	Allows an attacker to receive other users' order data	High	Stop using obfuscated email as a lookup key, use a unique non-sensitive identifier
44	Misplaced Signature File	Sensitive file exposure via path traversal / null-byte injection	Direct to secure path http://localhost:3000/ftp/suspicious_errors.yml%2500.md	An attacker can download private files signatures, keys, or configs, exposing secrets that allow forgery or further compromise.	High	Equire authentication/authorization for downloads. Remove sensitive files from public folders. Disable directory listing. Log and monitor access.
45	Reset Bender's Password	Insecure password-reset using a guessable	Find the answer with OSINT (Wikipedia/Futurama or Futurama wiki), go to http://localhost:3000/#/forgot-password, enter bender@juice-sh.op, answer Stop'n'Drop as the company, and set a new password.	Account takeover — attacker can reset Bender's password and access or abuse the account	High	Replace security-question resets with time-limited reset tokens sent to verified email
46	Vulnerable Library	Use of a vulnerable third-party library (sanitize-html v1.4.2) that can be bypassed and lead to XSS	Submit feedback containing the strings sanitize-html and 1.4.2	Stored XSS — attacker can run JavaScript in other users' browsers, steal cookies, or perform actions as those users.	High	Upgrade to a patched sanitize-html version, remove/escape user-supplied HTML server-side
47	Legacy Typosquatting	Trusting or recording package names from untrusted input which could lead to using a malicious package	Submit epilogue-js in the feedback input box	A malicious package could execute code, steal data, or compromise the server or developer environment.	High	Never install or execute packages from user input
48	Manipulate Basket	Insecure Direct Object Reference (IDOR) / Broken access control	Change the basketId in the Burp to another user's basket ID and add items	Attacker can add, remove or buy items from other users' carts, cause financial loss, order theft, or privacy leaks	High	Verify basket ownership server-side for every request, use non-guessable IDs, enforce authorization checks, rate-limit actions and log suspicious activity.
49	Login Amy	Weak authentication / lack of rate-limiting	Automated password using Burp intruder (password K1f.....)	Attacker can access Amy's data, perform actions as her, and escalate privileges.	High	Implement account lockout or progressive delays, enforce strong password policies, enable multi-factor authentication
50	Privacy Policy Inspection	Client-side exposure of internal routes via hidden markup	Inspect HTML for text, join the words into the localhost URL and visit it	Reveals internal endpoints or functionality that could be abused or leak sensitive data.	High	Remove or encode hidden route fragments from public content, require authentication for internal endpoints, and avoid embedding discoverable secrets in client-side HTML.
51	Upload Size	Client-side checks can be bypassed	Upload a small allowed file (<100KB), intercept the request with Burp, replace the file payload with a larger file (>100KB), then forward the request — server accepts the oversized file.	Attacker can store very large files to consume disk/quota (DoS), bypass client limits, or upload large malicious files that increase attack surface.	Medium	Enforce server-side max upload size and validate actual byte length, configure webserver/app upload limits, stream and reject oversize uploads early, rate-limit uploads, authenticate users, and scan uploaded files.

52	Upload Type	Improper file upload validation / unrestricted file upload	Upload an allowed file, intercept the request with Burp, replace the file contents or extension with a disallowed type (e.g., PHP web-shell) and forward the request so the server stores/accepts it.	Attacker can upload and execute malicious files (RCE), host malware, steal or delete data, deface the site, or pivot to internal systems.	Medium	Validate file type server-side (check MIME and file signature), use a strict whitelist, scan uploads for malware, store files outside the web root with no execute permissions, sanitize filenames, enforce size/type limits server-side, and require authentication/authorization for uploads.
53	NoSQL Manipulation	NoSQL Injection	Update multiple reviews with { "id": { "$ne": -1 }, "message": "NoSQL Injection!" } using Burp	Attacker can change queries to read, modify, or delete data, bypass access checks, inject fake reviews, or leak sensitive information	Medium	Validate and sanitize all input; forbid or escape NoSQL operators (e.g., keys starting with $); use parameterized queries or safe query builders/ODMs; enforce strict schema/type validation server-side (e.g., MongoDB $jsonSchema); use least-privilege DB accounts; add server-side authorization checks, rate limits, and logging
54	Access Log	Sensitive data exposure / missing access control (logs accessible publicly)	Open http://localhost:3000/support/logs in a browser or curl the URL	Attackers can read server logs that may contain PII, session tokens, API keys, internal IPs, stack traces, or debugging secrets — enabling account takeover, targeted attacks, or info leakage.	High	Restrict access to logs with strong authentication and admin-only authorization; move logs off the web root
55						