Software Test Plan (STP)

Team-Member
Sameer Shaik
Vivek Kommareddy
Prakhya Tummala

Version 1 – November 19, 2024

Software Design Document

AuctionNest

VERSION: 1 REVISION DATE:11/19/2024

Approval of the Software Test Plan (SDTP) indicates an understanding of the purpose and content described in this deliverable. By signing this deliverable, each individual agrees with the content contained in this deliverable.

Approver Name	Title	Signature	Date
Sameer Shaik	Client	*Sameer Shaik*	11/19/2024
Vivek Kommareddy	Developer	*Vivek Kommareddy*	11/19/2024
Prakya Tummala	QA-Tester	*Prakya.tummala*	11/19/2024

Contents

Section 1 Purpose

The AuctionNest is a peer-to-peer platform that redefines the way users buy, sell, and auction items online. By integrating real-time auctions, advanced security measures, and seamless payment systems, AuctionNest ensures trust, security, and user satisfaction. This document outlines the system scope, core functionalities, and comprehensive test cases to ensure a high-quality user experience.

The purpose of this document is to establish a structured test plan and development strategy to validate that AuctionNest meets all functional, security, and performance requirements. The aim is to:

- Ensure secure and efficient transactions.
- Validate seamless operations of auctions, payments, and notifications.
- Provide a reliable and scalable platform for all users.

1.1 Objectives

The objectives of this Software Test Plan are:

The objectives of this Software Test Plan for AuctionNest are:

1.1.1 To validate that all functional requirements, including listing, buying, and auction features, are met as per the Software Design Document.

1.1.2 To ensure that the system performs reliably under various conditions, including high user traffic and simultaneous auction activities.

1.1.3 To verify that the platform maintains robust data security and privacy, especially for user authentication, account management, and payment transactions.

1.2 Scope

AuctionNest encompasses the following:

- **User Account Management**: Secure account registration, profile updates, and two-factor authentication.
- **Auction Management**: Real-time auction creation, bidding, and notifications.
- **Payment Processing**: Built-in secure payment gateways.
- **Customer Reviews and Ratings**: Feedback-driven trust mechanisms.
- **Performance Optimization**: Reliable operations during peak traffic.

Core Functionalities

1. Auction Management

- Create, monitor, and participate in auctions in real time.
- Notify bidders and sellers instantly about auction outcomes.
- Ensure accurate bid tracking and error-free notifications.

2. User Verification

- Authenticate users through two-factor authentication.
- Verify identity via an authenticator app.
- Prevent unauthorized access and protect user data.

3. Payment Gateway Integration

- Enable secure transactions with multiple payment options.
- Generate real-time receipts and ensure seamless payment flow.

4. Profile Management

- Allow users to update personal information and preferences.
- Deliver customized notifications based on user-defined settings.

5. Customer Interaction

- Facilitate in-app messaging between buyers and sellers.
- Build trust with user ratings and reviews**.**

Section 2 Approach and Method

2.1 Overall Approach

The overall approach for testing the **AuctionNest** platform follows a structured methodology to ensure comprehensive verification of the system's functionality, performance, and security. The testing strategy includes the following steps:

1. **Unit Testing**: Test individual components such as user authentication, item listing, and bidding functionalities to ensure they work as intended.
2. **Integration Testing**: Verify that modules, including buying, selling, and auction processes, interact correctly when combined.
3. **Functional Testing**: Ensure that the platform operates according to requirements, covering core features like user account management, item search, and secure transactions.

4. **Performance Testing**: Assess the system’s behavior under load, such as during peak auction activity, to ensure stability, responsiveness, and scalability.
5. **Regression Testing**: Re-run previous tests to check that recent changes have not affected existing functionalities.
6. **User Acceptance Testing (UAT)**: Involve end-users to validate that the final product meets their expectations.

Testing will be conducted iteratively throughout the development cycle to identify defects early and ensure ongoing quality assurance.

2.2 Testing Roles

Test Manager

- **Role Definition**: The Test Manager for AuctionNest will oversee all testing activities, including planning, scheduling, and resource allocation. They will define the test strategy, set up the test environment, and ensure test coverage aligns with the product's goals. The Test Manager will coordinate test execution and report on progress, risks, and outcomes.
- **Responsibilities:**
  Establish test plans and objectives.
  Monitor test progress and adjust strategies as needed.
  Ensure the timely delivery of high-quality test results.

Tester

- **Role Definition**: Testers will execute test cases for AuctionNest, both manually and using automated tools. They will document any bugs or inconsistencies and validate fixes. Their role ensures that the application functions correctly across different scenarios and meets performance benchmarks.
- **Responsibilities:**
  Perform functional, regression, and performance testing.
  Document test results and raise defect tickets.
  Collaborate with developers to reproduce and resolve issues.

Project Manager

- **Role Definition**: The Project Manager coordinates between the testing and development teams, ensuring that project timelines are adhered to and that testing feedback is effectively integrated into the development process. They are accountable for overall project delivery.
- **Reference:** Eisty, Nasir U, and Jeffrey C Carver. “Testing Research Software: A Survey.” *Empirical software engineering: an international journal* 27.6 (2022): n. pag. Web.

Lead Developer

- **Role Definition**: The Project Manager will act as a bridge between the testing and development teams, ensuring project milestones are met and testing feedback is incorporated effectively. They will manage timelines, budgets, and overall project execution for AuctionNest.

- **Responsibilities:** The Lead Developer will work closely with the Test Manager and testers to address bugs and implement feedback. They will ensure AuctionNest's code adheres to technical standards and incorporates necessary fixes identified during testing.

User/Stakeholder

- **Role Definition**: Users and stakeholders will engage in User Acceptance Testing (UAT) to ensure AuctionNest meets their functional and usability requirements. Their feedback will guide final refinements before launch.
- **Responsibilities:**
  Validate software against business requirements.
  Provide feedback on user experience and usability.
  Confirm readiness for deployment based on testing outcomes.

2.3 Testing Definitions

Unit Testing:

Definition: Unit Testing involves testing individual components or pieces of code to ensure they function correctly in isolation.
Example: Testing the auction creation module to ensure it accurately initializes auction details such as starting bid, auction duration, and item details.
Reference: Runeson, P. "A Survey of Unit Testing Practices." IEEE Software 23.4 (2006): 22–29. Web.

Integration Testing:

Definition: Integration Testing focuses on evaluating how different modules or services interact with one another to ensure they work together as expected.
Example: Testing the interaction between the user profile management module and the auction recommendation system to ensure personalized auction suggestions work as intended.
Reference: Pi, Chenchen. "Development and Classification of Computer Software Testing Technology." Journal of Physics: Conference Series 1650.3 (2020): 32111-. Web.

Functional Testing:

Definition: Functional Testing examines the software to ensure that each feature operates in conformance with the specified requirements.
Example: Testing the real-time auction monitoring feature to ensure users can view and bid on items seamlessly as auctions progress.
Reference: Pi, Chenchen. "Development and Classification of Computer Software Testing Technology." Journal of Physics: Conference Series 1650.3 (2020): 32111-. Web.

Performance Testing:

Definition: Performance Testing involves assessing the system's speed, responsiveness, and stability under various conditions.
Example: Simulating high user activity during peak auction hours to evaluate system scalability and identify potential bottlenecks.
Reference: Long, Yan et al. "A Performance Testing Method for HRF in Dual Mode Communication." Journal of Physics: Conference Series 2615.1 (2023): 12001-. Web.

Regression Testing:

Definition: Regression Testing is performed to verify that recent code changes have not negatively impacted existing functionalities.
Example: After implementing a new auction extension feature, rerunning tests on bidding and payment modules to ensure no disruption in core functionalities.
Reference: Parsons, David, Teo Susnjak, and Manfred Lange. "Influences on Regression Testing Strategies in Agile Software Development Environments." Software Quality Journal 22.4 (2014): 717–739. Web.

User Acceptance Testing (UAT):
Definition: Acceptance Testing is conducted to determine whether the system meets the business requirements and is ready for deployment.
Example: Involving real users to evaluate the ease of navigation, usability of the bidding process, and satisfaction with the overall auction experience.
Reference: Scherr, Simon André, Frank Elberzhager, and Konstantin Holl. "Acceptance Testing of Mobile Applications: Automated Emotion Tracking for Large User Groups." 2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft). New York, NY, USA: ACM, 2018. 247–251. Web.

Section 3 Test Script Format

1.Test Case ID

- A unique identifier for each test script to facilitate tracking and referencing.
- *Example*: TC_AUCTION_001 for testing auction creation.

2. Test Scenario/Objective

- A clear description of the test's purpose, such as validating a specific functionality or workflow within the AuctionNest platform.
- *Example*: "Validate that a user can successfully create a new auction listing."

3. Preconditions

- Any setup or prior conditions that must be fulfilled before executing the test, including system configurations or specific user states.
- *Example*: "User must be logged in with a verified seller account."

4. Test Steps

- Detailed instructions outlining the steps to execute the test.
- *Example*:
    1. Navigate to the "Create Auction" page.
    2. Enter auction details (item name, description, starting bid, and duration).
    3. Submit the form.

5. Expected Results

- The anticipated outcomes that would signify a successful test.
- *Example*: "Auction is created successfully, and a confirmation message is displayed with the auction ID."

6. Actual Results

- The actual outcomes observed after test execution.
- *Example*: "Auction was created, but no confirmation message was displayed."

7. Pass/Fail Criteria

- A section for documenting whether the test passed or failed based on a comparison of expected and actual results.
- *Example*:
    - Pass: Confirmation message and auction ID are displayed.
    - Fail: No confirmation message appears, or auction is not listed.

8. Test Data

- Information about any data used during test execution, such as user input or mock data for auctions.
- *Example*: Item name: "Vintage Watch," Starting Bid: $100.

9. Postconditions

- Any clean-up steps or conditions to verify after test execution, such as ensuring data integrity or clearing test records.
- *Example*: "Verify that the created auction appears in the user's dashboard and is visible to other users."

References

1. "Best Practices for Writing Effective Test Scripts." *International Journal of Software Engineering and Technology*, 2018.
2. "A Comprehensive Approach to Test Script Automation." *Software Testing, Verification & Reliability Journal*, 2020.

Section 4 Unit Testing Test Script

1. 1. User Registration & Login

Objective: Ensure that the user registration and login process work as expected.

Unit Test Scenarios:

User Registration:

- **Test Case**: When a new user tries to register with valid details (e.g., username, email, password), the system should successfully create the user account and store the details in the database.
    - **Expected Outcome**: The user should be successfully registered, and their information (like username and email) should be saved correctly in the system.

1. **Email Validation**:
    - **Test Case**: If a user tries to register with an invalid email format (e.g., "invalidemail.com" without an "@" symbol), the system should reject the registration attempt and return an error message.
        - **Expected Outcome**: The system should not allow registration with an invalid email and prompt the user to correct the email address.
2. **Login Validation**:
    - **Test Case**: When a user enters valid login credentials (email and password), the system should authenticate them and grant access to the application.
        - **Expected Outcome**: The user should be logged in successfully and directed to the homepage or dashboard.
3. **Incorrect Password Handling**:
    - **Test Case**: If a user tries to log in with an incorrect password, the system should display an error message indicating the credentials are invalid.
        - **Expected Outcome**: The login attempt should fail, and an error message should be displayed to the user.
4. **Logout Process**:
    - **Test Case**: When a logged-in user clicks the logout button, the system should end the session and log the user out, ensuring they are redirected to the login page.
        - **Expected Outcome**: The user should be logged out, and their session data should be cleared.

2. **Auction Bid Management:**

Objective: Test the functionality that allows users to manage auctions effectively, including selling, buying, and bidding.

Unit Test Scenarios:

1. **Creating an Auction Listing**:
    - **Test Case**: When a seller creates a new auction listing, the system should save the listing details, including item name, description, starting price, and auction duration.
        - **Expected Outcome**: The auction listing should be successfully created and visible to other users.
2. **Placing a Bid**:
    - **Test Case**: When a buyer places a bid on an auction item, the system should save the bid details and update the highest bid if the new bid is valid.
        - **Expected Outcome**: The bid should be accepted, and the highest bid should be updated if applicable.
3. **Winning a Bid**:
    - **Test Case**: When the auction ends, the system should determine the highest bidder as the winner and notify them.
        - **Expected Outcome**: The winner should be notified, and the auction status should be updated to "Closed."
4. **Bid Rejection for Lower Bids**:
    - **Test Case**: If a user tries to place a bid lower than the current highest bid, the system should reject the bid and display an error message.
        - **Expected Outcome**: The bid should be rejected, and an error message should inform the user to increase their bid.
5. **Selling an Item Without Auction**:
    - **Test Case**: When a user opts to sell an item directly, the system should allow buyers to purchase at the listed price without bidding.
        - **Expected Outcome**: The item should be marked as "Sold" once purchased.

3. **Order & Delivery Scheduling:**

Objective: Test that the order and delivery scheduling features work as intended.

Unit Test Scenarios:

1. **Order Placement**:
    - **Test Case**: When a buyer purchases an item (via auction win or direct sale), the system should correctly capture the order details and store them in the database.
        - **Expected Outcome**: The order should be created with the correct products, quantities, and customer information.
2. **Delivery Scheduling**:
    - **Test Case**: When a buyer schedules delivery, the system should allow them to select a date and time for delivery and store the schedule.
        - **Expected Outcome**: The system should save the delivery details, and the buyer should view their scheduled delivery information.
3. **Order Status Updates**:

- **Test Case**: After order placement, the system should update the status as it progresses (e.g., "Pending," "Shipped," "Delivered").
  - **Expected Outcome**: The order status should be updated correctly, and the user should be notified of status changes.

4. **Handling Delivery Failures**:
   - **Test Case**: If a delivery fails (e.g., due to an incorrect address), the system should notify the buyer and allow them to reschedule or update their delivery details.
     - **Expected Outcome**: The buyer should be notified and able to resolve the issue by updating the delivery information.

4. **Authenticator Integration:**

Objective: Validate that the authentication system functions correctly and securely across all platform features.

Unit Test Scenarios:

1. **Two-Factor Authentication (2FA)**:
   - **Test Case**: When a user enables 2FA, the system should prompt for a second authentication step (e.g., OTP or authenticator app code) during login.
     - **Expected Outcome**: The user must provide a valid OTP/code to gain access.
2. **Session Timeout**:
   - **Test Case**: If a user remains inactive for a certain period, the system should automatically log them out and require re-authentication.
     - **Expected Outcome**: The user should be redirected to the login page upon session timeout.
3. **Unauthorized Access Prevention**:
   - **Test Case**: If an unauthenticated user attempts to access a restricted page, the system should deny access and redirect them to the login page.
     - **Expected Outcome**: The unauthorized user should not access restricted features.

Section 5 Function Testing Test Script

Test Script 1: User Registration and Login

Test Scenario: Verify that a new user can successfully register and log in to the system.
Requirements Addressed:

- User registration and login functionality.
- Validation of email format during registration.
- Authentication mechanism during login.

Test Case 1: User Registration with Valid Details

1. **Test Steps:**
   a. Open *TheAuctionNest* web/mobile application.
   b. Navigate to the "Register" section.
   c. Enter the following valid details:
   i. Name: "Jane Doe"

ii. Email: "janedoe@example.com"
iii. Password: "securePass123"
d. Click the "Register" button.
e. Verify that the system displays a "Registration successful" message.
f. Verify that the user's information (name and email) is stored in the system’s database.

2. **Expected Outcome:**
a. The user should be successfully registered.
b. The system should store user details (name, email, password) securely.
c. A success message should confirm registration.

Test Case 2: Login with Valid Credentials

1. **Test Steps:**
a. Open *TheAuctionNest* application.
b. Navigate to the "Login" section.
c. Enter the following valid credentials:
i. Email: "janedoe@example.com"
ii. Password: "securePass123"
d. Click the "Login" button.
e. Verify that the system logs the user in and redirects them to the dashboard.
2. **Expected Outcome:**
a. The user should be successfully authenticated and logged in.
b. The system should redirect the user to their dashboard.

Test Case 3: Login with Invalid Credentials

1. **Test Steps:**
a. Open *TheAuctionNest* application.
b. Navigate to the "Login" section.
c. Enter invalid login details:
i. Email: "janedoe@example.com"
ii. Password: "wrongPass123"
d. Click the "Login" button.
2. **Expected Outcome:**
a. The system should reject the login attempt and display "Invalid email or password."
b. The user should not gain access.

Test Script 2: Auction Bid Management

Test Scenario: Verify that users can list items, place bids, and manage bids effectively.
Requirements Addressed:

- Functionality for listing items for auction.
- Ability to bid on items.
- Management of bidding activities.

Test Case 1: Listing an Item for Auction

1. **Test Steps:**
   a. Log in as a seller (e.g., "janedoe@example.com").
   b. Navigate to the "Sell Item" section.
   c. Enter item details:
   i. Item Name: "Antique Vase"
   ii. Starting Bid: "$50"
   iii. Auction Duration: "7 days"
   d. Click "List Item."
   e. Verify that the item is displayed in the auction listing.
2. **Expected Outcome:**
   a. The item should be successfully listed for auction.
   b. The auction should display the correct starting bid and duration.

Test Case 2: Placing a Bid on an Item

1. **Test Steps:**
   a. Log in as a buyer (e.g., "johnsmith@example.com").
   b. Navigate to the auction listing for "Antique Vase."
   c. Enter a bid amount higher than the current bid (e.g., "$60").
   d. Click "Place Bid."
   e. Verify that the system updates the current highest bid.
2. **Expected Outcome:**
   a. The bid should be successfully placed and recorded.
   b. The system should update the auction with the new highest bid.

Test Case 3: Updating a Bid

1. **Test Steps:**
   a. As the same buyer, place a new higher bid on the "Antique Vase" (e.g., "$70").
   b. Click "Update Bid."
   c. Verify that the system reflects the updated bid amount.
2. **Expected Outcome:**
   a. The auction should display the updated highest bid.

Test Script 3: Product Search and Filtering

Test Scenario: Verify that users can search for items and filter auction results based on categories.
Requirements Addressed:

- Search functionality.
- Filtering auctions by categories (e.g., “Electronics,” “Antiques”).

Test Case 1: Searching for an Item by Name

1. **Test Steps:**
   a. Open *TheAuctionNest* application.
   b. Navigate to the "Search" bar.
   c. Enter "Antique Vase" in the search field.
   d. Click "Search."
   e. Verify that the system displays relevant auction listings for "Antique Vase."
2. **Expected Outcome:**
   a. The system should return auctions related to the search term.

Test Case 2: Filtering Auctions by Category

1. **Test Steps:**
   a. Open *TheAuctionNest* application.
   b. Navigate to the "Filter" options.
   c. Select the category "Antiques."
   d. Click "Apply Filter."
2. **Expected Outcome:**
   a. The displayed auction list should only include items in the "Antiques" category.

Test Case 3: Combining Multiple Filters

1. **Test Steps:**
   a. Open *TheAuctionNest* application.
   b. Select filters for "Antiques" and "Bids under $100."
   c. Click "Apply Filters."
2. **Expected Outcome:**
   a. The auction list should include only items that meet both filter criteria.

Section 6 Integration Testing Test Script

Integration Test Script 1: User Registration and Database Integration

Test Scenario: Verify that the user registration process in *The AuctionNest* properly integrates with the database, ensuring user data is stored correctly.

Requirements Addressed:

- User registration process.
- Database integration to store user details.
- Data persistence in the system.

Test Case 1: Register User and Verify Data Persistence in Database

1. Test Steps:
a. Open *The AuctionNest* web application.
b. Navigate to the "Register" section.
c. Enter the following valid details:

- Name: "Jane Doe"
- Email: "janedoe@auctionnest.com"
- Password: "securePassword123"
  d. Click the "Register" button.
  e. Verify that the system shows a "Registration successful" message.
  f. Access the system's database (via an admin panel or SQL query tool).
  g. Run a query to check the Users table for the new entry with the following details:
- Name: "Jane Doe"
- Email: "janedoe@auctionnest.com"
- Password: The password should be hashed (not stored as plaintext).
  h. Verify that the data is correctly stored and consistent.

2. Expected Outcome:

- User details are successfully stored in the database.
- Database entry contains the correct data.
- Password is securely hashed.

Integration Test Script 2: Registration and Email Notification Integration

Test Scenario: Verify that after successful registration on *The AuctionNest*, an email notification is sent to the user.

Requirements Addressed:

- User registration process.
- Email service integration for confirmation notifications.

Test Case 1: Verify Email Notification After Successful Registration

1. Test Steps:
a. Open *The AuctionNest* web application.
b. Navigate to the "Register" section.
c. Enter the following valid details:

- Name: "Mark Smith"
- Email: "marksmith@auctionnest.com"
- Password: "password456"
  d. Click the "Register" button.
  e. Verify that the system shows a "Registration successful" message.

f. Check the inbox of the registered email account (marksmith@auctionnest.com).
g. Verify that a confirmation email is received with the following:

- A welcome message.
- Confirmation of successful registration.
- A link to set up account preferences or start bidding.

2. Expected Outcome:

- A confirmation email is sent to the user's registered email.
- The email contains accurate and relevant information.

Integration Test Script 3: User Registration and Auction Setup Integration

Test Scenario: Verify that after user registration, the user can list an item for auction on *The AuctionNest*.

Requirements Addressed:

- User registration process.
- Integration of auction listing functionality.
- Data synchronization between the user profile and auction system.

Test Case 1: Verify Auction Setup After Registration

1. Test Steps:
a. Open *The AuctionNest* web application.
b. Register as a new user with valid details:

- Name: "Emily Clark"
- Email: "emilyclark@auctionnest.com"
- Password: "emilyPassword123"

c. Navigate to the "Auction Dashboard" after registration.
d. Click "Create New Auction."
e. Enter the following auction details:

- Item Name: "Antique Vase"
- Starting Bid: $100
- Auction Duration: 7 days

f. Click "Submit Auction."
g. Verify that the auction is listed under the user's active auctions.
h. Access the database and verify that the auction details are correctly stored in the Auctions table.
i. Verify that the system sends a confirmation email with the auction details.

2. Expected Outcome:

- The user can list an item for auction immediately after registration.
- The auction is correctly stored in the database.

- A confirmation email is sent with auction details.

Section 7 Performance Testing Test Script

Performance Testing for *The AuctionNest*

Performance testing is crucial to ensure that *The AuctionNest* platform can handle the expected load and perform optimally under various user interactions. Below are test scripts for load, stress, and concurrency testing to verify the platform's scalability, reliability, and efficiency in critical areas, including product search, order placement, and inventory management.

Test Script 1: Load Testing for Auction Search

Test Scenario:
Verify that *The AuctionNest* platform can handle a large number of concurrent product search requests without significant degradation in performance.

Requirements Addressed:

- Scalability of the auction search functionality.
- Response time for search queries under load.
- System's ability to handle concurrent searches.

Test Steps:

1. Open the *The AuctionNest* web/mobile application.
2. Simulate 500 concurrent users searching for auction items (e.g., art, antiques, collectibles) using different filters (e.g., highest bid, new listings, categories).
3. Monitor the average response time for each search query.
4. Track system performance, including CPU and memory usage.
5. Gradually increase the number of concurrent users to 1000 and repeat the search requests.
6. Measure the time taken for each search request to be processed and responded to by the system.
7. Verify that the system does not crash or display errors during the load.

Expected Outcome:

- The system should maintain an average response time of less than 3 seconds for each search request under load.
- The platform should support up to 1000 concurrent users without significant performance degradation (e.g., response times exceeding 5 seconds).
- No errors or crashes should occur during the test.

Test Script 2: Stress Testing for Bid Placement and Checkout

Test Scenario:

Verify that *The AuctionNest* platform can handle high traffic during bid placement and checkout processes without delays or system failures.

Requirements Addressed:

- System's ability to process a high number of simultaneous bid placements.
- Throughput of the bid placement and checkout process.
- Performance of payment gateway integration under peak load.

Test Steps:

1. Open the *The AuctionNest* web/mobile application.
2. Simulate 300 concurrent users placing bids on various items and proceeding to the checkout page.
3. Monitor and log the time taken from when a user initiates the checkout to when the system confirms the bid and payment is processed.
4. Integrate the payment gateway (e.g., Stripe/PayPal) and simulate the payment process.
5. Gradually increase the number of concurrent users to 500.
6. Continue to monitor CPU, memory, and network utilization during the bid placement and payment process.
7. Record the transaction success/failure rate, and check for system errors or performance bottlenecks.

Expected Outcome:

- Each user's bid placement and payment confirmation should complete within 5 seconds (from initiating checkout to receiving payment confirmation).
- The system should handle up to 500 simultaneous bid placements without degradation in response time or payment failures.
- No errors should occur during payment processing under peak load.

Test Script 3: Stress Testing for Inventory Management and Auction Updates

Test Scenario:

Verify that *The AuctionNest* platform can handle simultaneous updates to auction listings, inventory (e.g., item details), and bid statuses without performance degradation.

Requirements Addressed:

- Performance of the inventory and auction management system under load.
- System's ability to process multiple updates concurrently.
- Database handling of concurrent write operations, such as listing updates, bid placements, and item status changes.

Test Steps:

1. Open the *The AuctionNest* web/mobile application.
2. Simulate 100 concurrent users accessing the auction management system (for both sellers and administrators).
3. Each user should perform one of the following actions: a. Add a new auction listing to the platform. b. Update bid details or auction item descriptions. c. Remove an existing auction listing.
4. Monitor database transactions and measure the time it takes to complete updates to auction listings and bids.
5. Gradually increase the number of users to 200, continuing to simulate auction updates.
6. Monitor database query performance, ensuring there are no deadlocks, delays, or transaction issues.
7. Measure system resource usage (CPU, memory, disk I/O) while auction actions are being performed.

Expected Outcome:

- Auction updates (add, update, remove) should be processed in under 3 seconds per user under load.
- The system should be able to handle up to 200 concurrent inventory management and auction-related actions without locking or delays.
- No database performance issues, such as deadlocks or excessive query times, should occur during the test.

Conclusion

By performing these load, stress, and concurrency tests, *The AuctionNest* platform will be validated for scalability, system reliability, and efficiency during peak user activity, such as product search, bid placement, and auction management. Ensuring that the system can handle high traffic volumes without performance degradation will provide a smooth and reliable user experience, crucial for the success of any online auction platform.

Section 8 User Acceptance Testing Test Script

Test Script 1: User Registration and Profile Management

Test Scenario:

Verify that users can successfully register, log in, and manage their profile settings (e.g., contact information, notification preferences, bid settings).

Requirements Addressed:

- User registration process
- Profile management (e.g., contact details, notifications, bid settings)
- Secure login and authentication

Test Steps:

1. Open the AuctionNest web/mobile application.
2. Navigate to the "Register" section.
3. Enter the following valid details to create an account:
   a. Name: "John Doe"
   b. Email: "johndoe@example.com"
   c. Password: "johnPassword123"
4. Click the "Register" button.
5. Verify that the system displays a "Registration successful" message.
6. Log in to the system using the registered email address and password.
7. Navigate to the "Profile" section.
8. Update the following profile settings:
   a. Contact information: "123 Main Street"
   b. Enable notifications for new auctions and bid status updates.
   c. Set default bid preferences for auctions (e.g., max bid limit).
9. Save the changes and verify that they are correctly applied to the profile.
10. Log out and log in again to ensure the profile settings persist.

Expected Outcome:

- The user should successfully register, log in, and be able to update their profile settings.
- The system should save the updated contact information and notification settings correctly.
- Upon logging out and logging in again, the profile settings should persist.
- Registration, login, and profile update processes should be seamless and error-free.

Test Script 2: Product Search and Add to Cart for Auction Items

Test Scenario:

Verify that users can search for auction items, filter based on categories (e.g., art, electronics), and add items to their bid list.

Requirements Addressed:

- Auction item search functionality
- Ability to filter items based on categories and bidding preferences
- Add to bid list functionality
- User-friendly auction browsing experience

Test Steps:

1. Open the AuctionNest web/mobile application.
2. Log in with an existing user account (e.g., "johndoe@example.com").
3. Navigate to the "Search" section.

4. Search for "Vintage Watch" using the search bar.
5. Apply filters for categories such as "Art" and "Vintage".
6. Verify that the search results show only relevant auction items based on the applied filters.
7. Click on the "Vintage Watch" item to view its details.
8. Add the item to the "Bid List" or "Watch List".
9. Navigate to the "Bid List" section and verify that the item is listed with correct details (item name, auction time, current bid).
10. Verify that users can track upcoming auction times for their added items.

Expected Outcome:

- The product search should return relevant results based on the applied filters (category, auction type).
- The user should be able to add auction items to their bid list or watch list successfully.
- The bid list should accurately display the selected auction items with correct details (auction time, current bid).
- Any updates to the bid list should reflect the correct item information.

Test Script 3: Bid and Auction Participation
Test Scenario:
Verify that users can place bids on auction items and receive notifications when they are outbid or when an auction ends.

Requirements Addressed:

- Bid placement process
- Auction participation and notification system
- Secure and accurate bid processing

Test Steps:

1. Open the AuctionNest web/mobile application.
2. Log in with an existing user account (e.g., "johndoe@example.com").
3. Navigate to the "Bid List" section and select an auction item to participate in.
4. Place a bid by entering a bid amount that is higher than the current bid.
5. Verify that the system processes the bid and updates the auction with the new bid amount.
6. Enable notifications for outbid status and auction close alerts.
7. Monitor notifications and verify that the user receives an alert when they are outbid or when the auction closes.
8. Confirm that the system correctly updates the bid history and winning status once the auction ends.

Expected Outcome:

- The user should be able to place bids on auction items without errors.
- The system should update the auction with the new bid and display the updated bid amount.
- Notifications should be sent when the user is outbid or when the auction ends.
- After the auction ends, the system should correctly indicate the winning bid and the user's status.

Test Script 4: Checkout and Payment for Auction Wins

Test Scenario:

Verify that users can successfully complete a payment for an auction item they won, receive a confirmation, and view their order history.

Requirements Addressed:

- Seamless checkout process for auction wins
- Secure payment processing
- Order confirmation and history

Test Steps:

1. Open the AuctionNest web/mobile application.
2. Log in with an existing user account (e.g., "johndoe@example.com").
3. Navigate to the "Won Auctions" section and select a won item.
4. Click on the "Proceed to Payment" button.
5. Enter valid shipping details (e.g., address, phone number).
6. Select a payment method (e.g., Credit Card, PayPal) and enter payment information.
7. Click on the "Pay Now" button.
8. Verify that the system processes the payment and displays an order confirmation screen with:
   a. Order number
   b. Products purchased
   c. Delivery date and time
9. Check the registered email address for the order confirmation email.
   a. Verify that the email contains order details including the order number, products, and expected delivery date.
10. Navigate to the "Order History" section and verify that the new order appears with correct details.

Expected Outcome:

- The user should be able to successfully complete the payment for auction items they won, including entering shipping details.
- After the payment is processed, the system should display an order confirmation page with accurate order details.
- An order confirmation email should be sent to the user's registered email address with correct order information.
- The user should be able to see the new order in their order history.

Section 9 References

1.Best Practices for Writing Effective Test Scripts

Authors: Ashish P., Tejal P.

Published: February 2019

This paper presents a comprehensive guide on best practices for writing test scripts that are efficient and effective. It explores key principles such as clear test case design, reuse of test scripts, and proper test data management. Emphasis is placed on creating scripts that are not only functional but also scalable and maintainable, ensuring long-term value for software testing teams.

2.Testing Research Software: A Survey

Authors: Eisty, Nasir U.; Carver, Jeffrey C.

Published: June 2021

In this survey, the authors examine the unique challenges of testing research software, which often involves complex, interdisciplinary systems. The paper outlines common issues faced when testing research-based applications, such as managing reproducibility, dealing with incomplete specifications, and navigating the integration of diverse software components. The study also highlights the importance of open-source tools and collaborative testing practices in this field.

3.Influences on Regression Testing Strategies in Agile Software Development Environments

Authors: David Parsons, Teo Susnjak, Manfred Lange

Published: March 2022

This paper delves into the strategies for regression testing within agile software development environments. The authors investigate how agile principles such as iterative development, fast feedback loops, and frequent changes to codebases influence regression testing practices. They emphasize the importance of test automation, collaboration between developers and testers, and continuous integration in maintaining effective regression testing.

4.A Performance Testing Method for HRF in Dual Mode Communication

Authors: Yan Long, Yongli Chen, Zhihua Bai, Xiaoqing Zhao

Published: October 2023

Focusing on performance testing for Hybrid Relay Forwarding (HRF) in dual-mode communication, this paper presents a specialized testing methodology for assessing the performance of communication systems that combine different modes. The authors provide a framework for evaluating key performance metrics, such as throughput and latency, in these systems. This approach is designed to ensure that communication protocols meet the demands of modern, multifaceted communication environments.

5.Acceptance Testing of Mobile Applications: Automated Emotion Tracking for Large User Groups

Authors: Simon André Scherr, Frank Elberzhager, Konstantin Holl

Published: 2018

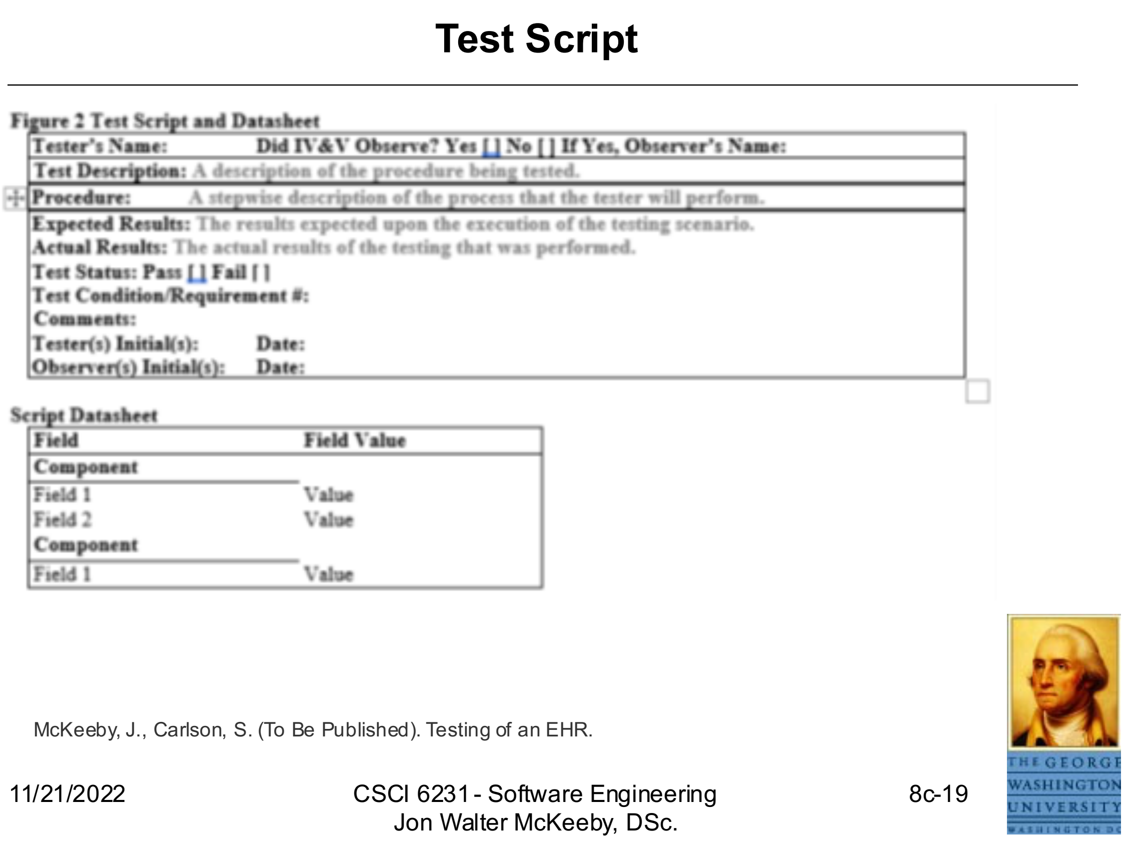
This research introduces a novel approach to acceptance testing for mobile applications, using automated emotion tracking to measure user reactions during testing. The paper outlines a method for tracking emotions across large user groups in real-time, providing insights into user satisfaction, usability, and overall app experience. This methodology is particularly useful in understanding the emotional impact of mobile apps on users, helping developers optimize user interfaces and functionality.

Section 10 Glossary:

1. **Acceptance Testing**: Testing conducted to confirm that the system meets its requirements and is ready for deployment to users.
2. **API (Application Programming Interface)**: A set of protocols and tools that allow different software applications to communicate and interact with each other.
3. **Cloud Storage**: Remote data storage services accessible over the internet, allowing users to store and retrieve files online.
4. **Data Flow Diagram (DFD)**: A visual representation of how data moves through a system, highlighting inputs, outputs, processes, and data storage.
5. **Entity-Relationship Diagram (ERD)**: A diagram that shows the structure of a database, illustrating the entities (e.g., tables) and their relationships to one another.
6. **Functional Testing**: Testing to ensure that each feature of the software operates according to the defined requirements.
7. **Integration Testing**: Testing that combines individual software modules to ensure they work together as expected.
8. **Load Testing**: Testing to evaluate the system's performance under both normal and peak load conditions, assessing its behavior under expected usage.
9. **Regression Testing**: Testing to ensure that recent changes or updates to the system haven't negatively impacted the existing functionality.
10. **Scalability**: The ability of a system to handle increased loads, such as more users or data, without performance degradation.
11. **SQL (Structured Query Language)**: A programming language used to interact with and manage data in relational databases.
12. **Stress Testing**: Testing the system's performance by subjecting it to extreme conditions, such as high traffic or unusual operations, to determine its breaking point.
13. **Test Case**: A specific scenario, with defined inputs, actions, and expected outcomes, used to validate a particular feature or functionality of the system.
14. **Unit Testing**: Testing of individual components or units of a system in isolation to ensure they function as intended.
15. **UML (Unified Modeling Language)**: A standardized modeling language used to visualize, specify, construct, and document the components of software systems.
16. **User Interface (UI)**: The visual components of a system that users interact with, such as buttons, menus, and icons.
17. **User Experience (UX)**: The overall experience and satisfaction of users when interacting with a system, focusing on usability, functionality, and emotional appeal.

18. **Verification**: The process of checking if a system meets the specified design and development standards.
19. **Validation**: The process of ensuring that the system satisfies the needs and requirements of the end users.
20. **Web Application**: A software application that runs in a web browser, requiring no installation on the user's device, accessible through the internet.

Appendix A: Example Test Script by prof Jon McKeeby



Test Script

Figure 2 Test Script and Datasheet

Tester's Name:	**Did IV&V Observe? Yes [ ] No [ ] If Yes, Observer's Name:**
Test Description: A description of the procedure being tested.	
Procedure: A stepwise description of the process that the tester will perform.	
Expected Results: The results expected upon the execution of the testing scenario. **Actual Results:** The actual results of the testing that was performed. **Test Status: Pass [ ] Fail [ ]** **Test Condition/Requirement #:** **Comments:** **Tester(s) Initial(s):** **Date:** **Observer(s) Initial(s):** **Date:**	

Script Datasheet

Field	**Field Value**
Component	
Field 1	Value
Field 2	Value
Component	
Field 1	Value

McKeeby, J., Carlson, S. (To Be Published). Testing of an EHR.

11/21/2022 CSCI 6231 - Software Engineering 8c-19
Jon Walter McKeeby, DSc.

THE GEORGE WASHINGTON UNIVERSITY WASHINGTON DC