CORADS System - Comprehensive Test Cases

Version 1.0 | Generated: October 2025

---

1. DRIVER REGISTRATION MODULE

Test Case DR-001: Successful Driver Registration

Priority: High
Requirement: FR-19
Objective: Verify that a driver can successfully register with valid credentials

Preconditions:

- Driver has Android device (API 23+) with internet connectivity
- Driver application is installed and launched
- Driver has valid personal information and vehicle details

Test Steps:

1. Navigate to registration screen
2. Enter valid name: "John Smith"
3. Enter valid email: "john.smith@email.com"
4. Enter valid phone: "+923001234567"
5. Enter password: "SecurePass123!"
6. Confirm password: "SecurePass123!"
7. Upload valid driver's license image
8. Enter license number: "DL12345678"
9. Upload vehicle registration image
10. Enter registration number: "ABC-123"
11. Select car type from dropdown
12. Agree to terms and conditions
13. Tap "Register" button

Expected Results:

- All input fields accept valid data without errors
- Password strength indicator shows "Strong"

- Images upload successfully with confirmation
- Registration request processes within 3 seconds (NFR-4)
- Success message displayed: "Registration successful. Please check your email for verification."
- Verification email sent to provided address
- User redirected to login screen
- New driver record created in database with status "Pending Verification"

Postconditions:

- Driver account created in database
- Verification email sent
- Driver can proceed to login after email verification

Test Case DR-002: Registration with Existing Email

Priority: High
Requirement: FR-19
Objective: Verify system prevents duplicate email registration

Preconditions:

- Email "existing@email.com" already registered in system
- Driver on registration screen

Test Steps:

1. Enter name: "Jane Doe"
2. Enter email: "existing@email.com"
3. Enter phone: "+923009876543"
4. Enter password: "NewPass456!"
5. Confirm password: "NewPass456!"
6. Complete all other required fields
7. Tap "Register" button

Expected Results:

- System validates email against existing records
- Error message displayed: "This email is already registered. Please login or use a different email."
- Registration form remains populated (except password fields)
- No duplicate record created in database
- User remains on registration screen

Postconditions:

- No new account created
- Database integrity maintained

Test Case DR-003: Registration with Invalid Phone Format

Priority: Medium
Requirement: FR-19
Objective: Verify input validation for phone number field

Preconditions:

- Driver on registration screen

Test Steps:

1. Enter valid data in all fields
2. Test phone number with various invalid formats:
    - "123456" (too short)
    - "abcd1234567" (contains letters)
    - "+1234567890123456789" (too long)
    - "03001234567" (missing country code)
3. Attempt to proceed with each invalid format

Expected Results:

- Real-time validation triggers on each invalid entry
- Error message displayed: "Please enter valid phone number in format +92XXXXXXXXXX"
- Register button remains disabled until valid format entered
- No API call made until validation passes

Postconditions:

- User corrects phone format before registration completes

Test Case DR-004: Password Strength Validation

Priority: High
Requirement: FR-19, NFR-11
Objective: Verify password meets security requirements

Preconditions:

- Driver on registration screen

Test Steps:

1. Test various password combinations:
    - "pass" (too short, no uppercase, no numbers, no special chars)
    - "password123" (no uppercase, no special chars)
    - "Password" (no numbers, no special chars)
    - "Pass123" (no special chars, less than 8 chars)
    - "Password123!" (valid - 8+ chars, uppercase, lowercase, number, special char)
2. Observe password strength indicator
3. Attempt registration with each password

Expected Results:

- Weak passwords show red indicator with message: "Password must be at least 8 characters with uppercase, lowercase, number, and special character"
- Strong password shows green indicator
- Registration blocked for weak passwords
- Password stored as secure hash (not plaintext) in database

Postconditions:

- Only strong passwords accepted
- Password security enforced per NFR-11

Test Case DR-005: Image Upload Validation

Priority: Medium
Requirement: FR-24
Objective: Verify vehicle image upload functionality and validation

Preconditions:

- Driver on registration screen
- Device has camera access permission
- Device storage available

Test Steps:

1. Tap "Upload Vehicle Image" button
2. Select "Camera" option
3. Capture vehicle photo
4. Verify image preview displayed
5. Attempt upload with:
    - Valid image (JPEG, <5MB)

    - Oversized image (>10MB)
    - Invalid format (PDF, TXT)
    - Corrupted image file
6. Complete registration with valid image

Expected Results:

- Camera launches successfully (HW-3)
- Image preview shows captured photo
- Valid image uploads with progress indicator
- Oversized image error: "Image must be less than 5MB"
- Invalid format error: "Please upload JPG or PNG format"
- Corrupted file error: "Unable to process image. Please try again"
- Successful upload shows checkmark confirmation
- Image stored with unique identifier in system

Postconditions:

- Valid vehicle image stored in database
- Image retrievable for verification

Test Case DR-006: Registration Without Internet Connectivity

Priority: High
Requirement: NFR-4
Objective: Verify system behavior when network unavailable

Preconditions:

- Driver on registration screen
- All fields completed with valid data
- Device internet connectivity disabled

Test Steps:

1. Disable WiFi and mobile data
2. Complete all registration fields
3. Tap "Register" button
4. Observe system behavior
5. Re-enable internet connectivity
6. Retry registration

Expected Results:

- System detects no connectivity
- Error message: "No internet connection. Please check your network and try again."
- Loading indicator stops
- Form data retained (not cleared)
- Retry button appears
- After connectivity restored, registration completes successfully
- No partial records created in database

Postconditions:

- Registration completes only with valid connection
- User experience gracefully handles network issues

---

2. CAMPAIGN MANAGEMENT MODULE

Test Case CM-001: Create Campaign with Valid Parameters

Priority: High
Requirements: FR-11, FR-12, FR-53, FR-54, FR-55
Objective: Verify advertiser can create campaign with all required parameters

Preconditions:

- Advertiser logged in with verified account
- Wallet balance ≥ PKR 5,000
- Advertiser on campaign creation screen

Test Steps:

1. Enter campaign title: "Summer Sale 2025"
2. Enter description: "50% off on all electronics"
3. Upload 3 advertisement images (JPG, 2MB each)
4. Select target location: "Gulberg, Lahore" from map
5. Set campaign duration: Start Date: 10/10/2025, End Date: 10/20/2025
6. Select number of cars: 5
7. System calculates estimated cost: PKR 4,500
8. Review campaign summary
9. Tap "Create Campaign" button

Expected Results:

- All fields accept valid inputs without errors
- Image upload shows progress (0-100%)

- Map interface allows location selection with draggable pin (SW-3)
- Date picker prevents past dates
- Duration calculated automatically (10 days)
- Cost calculation appears in real-time
- Wallet balance checked against campaign cost
- Confirmation message: "Campaign created successfully. Awaiting admin approval."
- Campaign status set to "Pending" in database (FR-14)
- Campaign ID generated and displayed
- Campaign parameters validated per FR-55
- Response time < 2 seconds (NFR-4)

Postconditions:

- Campaign stored in database with status "Pending"
- Wallet balance remains unchanged until activation
- Campaign appears in advertiser's "Pending Campaigns" list
- Admin notification sent for approval (FR-43)

Test Case CM-002: Create Campaign with Insufficient Wallet Balance

Priority: High
Requirements: FR-11, BR-2
Objective: Verify system prevents campaign creation with insufficient funds

Preconditions:

- Advertiser logged in
- Current wallet balance: PKR 2,000
- Estimated campaign cost: PKR 3,500

Test Steps:

1. Complete all campaign fields
2. Review campaign summary showing cost PKR 3,500
3. Tap "Create Campaign" button

Expected Results:

- System validates wallet balance before submission
- Error message: "Insufficient wallet balance. Please recharge your wallet to proceed."
- "Recharge Wallet" button appears prominently
- Campaign creation blocked
- No database record created
- User redirected to wallet screen if tapping recharge button

Postconditions:

- No campaign created
- User prompted to add funds
- Business rule BR-2 enforced

Test Case CM-003: Campaign with Multiple Image Upload

Priority: Medium
Requirements: FR-12, FR-54
Objective: Verify support for multiple advertisement images

Preconditions:

- Advertiser on campaign creation screen
- 5 images prepared (JPEG format, varying sizes)

Test Steps:

1. Tap "Upload Advertisement Images"
2. Select multiple images (5 images)
3. Observe upload progress for each
4. Verify all images appear in preview grid
5. Remove 2nd image from selection
6. Add 1 additional image
7. Reorder images by dragging
8. Complete campaign creation

Expected Results:

- System supports multiple image selection (up to 10 images)
- Each image shows individual upload progress
- Preview grid displays thumbnails with image names
- Remove button (X) appears on each thumbnail
- Successfully removed image no longer in selection
- Newly added image appends to collection
- Drag-and-drop reordering works smoothly
- Final order reflected in campaign display sequence
- All images stored with campaign association
- Image metadata (size, format, upload time) recorded

Postconditions:

- All selected images linked to campaign

- Images retrievable in specified order

Test Case CM-004: Campaign Assignment to Drivers

Priority: High
Requirements: FR-56, FR-57, FR-58
Objective: Verify system matches and notifies appropriate drivers

Preconditions:

- Admin activated campaign for Lahore area, requiring 5 cars
- 10 drivers available in Lahore area
- 8 drivers meet car type criteria
- 5 drivers currently active

Test Steps:

1. Admin activates campaign (FR-46)
2. System processes driver matching algorithm
3. Verify driver selection criteria:
    - Location proximity to target area
    - Car type compatibility
    - Driver availability status
    - Driver rating > 3.5
4. System sends campaign notifications
5. Monitor driver responses
6. Track acceptance status

Expected Results:

- Algorithm selects 5 most suitable drivers within 10 seconds
- FCM notifications sent to selected drivers (COM-3, FR-85)
- Notifications delivered within 5 seconds (NFR-6)
- Each notification contains: campaign details, payment amount, duration
- Driver app shows campaign request with Accept/Reject options (FR-25, FR-26)
- System tracks responses in real-time (FR-58)
- If driver rejects, system notifies next best match
- Campaign status updates to "Active" when all 5 drivers accept
- Unselected drivers receive no notification
- Database records campaign-driver assignments

Postconditions:

- 5 drivers assigned to campaign

- Campaign status: "Active"
- Driver notifications logged
- Backup drivers identified if rejections occur

Test Case CM-005: Track Active Campaign Location

Priority: High
Requirements: FR-16, FR-73, FR-74, FR-75
Objective: Verify real-time tracking of campaign drivers

Preconditions:

- Campaign active with 3 assigned drivers
- All drivers have GPS enabled
- Drivers moving in target area
- Advertiser logged into app

Test Steps:

1. Navigate to "Active Campaigns"
2. Select campaign "Summer Sale 2025"
3. Tap "Track Drivers" button
4. Observe map interface
5. Verify driver locations update
6. Select individual driver to view details
7. Monitor for 5 minutes

Expected Results:

- Map loads showing target area boundaries
- Driver icons appear at current GPS locations (FR-73)
- Each driver icon color-coded by campaign
- Location updates every 30 seconds (NFR-3)
- Driver details shown on icon tap: name, car type, duration active
- Route trail displayed showing driver path
- Real-time distance traveled counter
- Campaign coverage heat map visible (FR-64)
- No lag in map interactions
- Accurate GPS coordinates displayed
- System handles GPS signal loss gracefully (shows last known position)

Postconditions:

- Tracking data logged for analytics

- Route history stored for reporting

Test Case CM-006: Complete Campaign and Generate Report

Priority: High
Requirements: FR-63, FR-64, FR-65
Objective: Verify campaign completion and report generation

Preconditions:

- Campaign end date reached
- All drivers completed assigned duration
- Verification images uploaded by drivers
- Campaign status: "Completed"

Test Steps:

1. Navigate to "Completed Campaigns"
2. Select completed campaign
3. Tap "View Report" button
4. Review report sections:
    - Campaign summary
    - Driver performance
    - Coverage area heat map
    - Total distance traveled
    - Total display time
    - Verification images
5. Export report as PDF
6. Share report via email

Expected Results:

- Report generates within 5 seconds
- Campaign summary shows: duration, number of cars, total cost, target area
- Driver table lists: name, distance traveled, time active, earnings, verification count
- Heat map displays advertisement display concentration (FR-64)
- Color gradient shows high/medium/low exposure areas
- Total metrics accurately calculated from tracking data
- Verification images gallery included with timestamps and locations (FR-65)
- PDF export contains all report elements formatted professionally
- Email share opens client with report attached
- Report data matches database records exactly

Postconditions:

- Report archived in system
- Advertiser has performance analytics
- Data retained per BR-6 (minimum 1 year)

Test Case CM-007: Edit Pending Campaign

Priority: Medium
Requirements: FR-11, FR-14
Objective: Verify advertiser can modify pending campaign before activation

Preconditions:

- Campaign created with status "Pending"
- Campaign not yet activated by admin
- Advertiser logged in

Test Steps:

1. Navigate to "Pending Campaigns"
2. Select campaign to edit
3. Tap "Edit" button
4. Modify campaign title
5. Change end date to extend by 5 days
6. Add 1 additional advertisement image
7. Update target location slightly
8. Save changes

Expected Results:

- Edit screen pre-populated with existing campaign data
- All editable fields unlocked
- Cost recalculated based on new duration
- Wallet balance validation performed again
- Success message: "Campaign updated successfully"
- Updated campaign parameters saved to database
- Original campaign ID retained
- Modification timestamp recorded
- Admin receives update notification

Postconditions:

- Campaign reflects new parameters

- Campaign remains in "Pending" status
- Version history tracked

Test Case CM-008: Cancel Active Campaign

Priority: Medium
Requirements: FR-13
Objective: Verify campaign cancellation process and refund calculation

Preconditions:

- Campaign active for 3 days of planned 10 days
- 5 drivers currently participating
- Total campaign cost: PKR 5,000
- Amount spent so far: PKR 1,500

Test Steps:

1. Navigate to "Active Campaigns"
2. Select campaign to cancel
3. Tap "Cancel Campaign" button
4. Review cancellation summary showing:
    - Days completed: 3
    - Days remaining: 7
    - Amount spent: PKR 1,500
    - Refund amount: PKR 3,500
5. Confirm cancellation reason from dropdown
6. Enter additional comments
7. Confirm cancellation

Expected Results:

- Warning dialog: "Are you sure you want to cancel this campaign? This action cannot be undone."
- Cancellation summary accurately calculates pro-rated costs
- Refund calculation: (Remaining days / Total days) × Total cost
- Upon confirmation, campaign status changes to "Cancelled"
- Notifications sent to all assigned drivers (FR-43)
- Drivers removed from campaign assignment
- Refund amount credited to advertiser wallet within 5 minutes
- Transaction record created showing refund
- Campaign remains visible in history with "Cancelled" status
- Driver earnings for completed portion processed and paid

Postconditions:

- Campaign terminated
- Partial refund issued
- Drivers compensated for work completed
- All parties notified

---

3. PAYMENT SYSTEM MODULE

Test Case PS-001: Wallet Recharge via JAZZ CASH

Priority: High
Requirements: FR-9, FR-66, FR-67
Objective: Verify successful wallet recharge through payment gateway

Preconditions:

- Advertiser logged in
- Current wallet balance: PKR 500
- JAZZ CASH account with sufficient balance
- Internet connectivity active

Test Steps:

1. Navigate to "Wallet" screen (FR-7)
2. Tap "Recharge Wallet" button
3. Enter recharge amount: PKR 2,000
4. Select payment method: "JAZZ CASH"
5. Review transaction summary:
    - Amount: PKR 2,000
    - Processing fee: PKR 40 (2%)
    - Total: PKR 2,040
6. Tap "Proceed to Payment"
7. Redirected to JAZZ CASH gateway
8. Enter JAZZ CASH MPIN
9. Confirm transaction
10. Return to CORADS app

Expected Results:

- Amount input field accepts only numeric values
- Minimum amount: PKR 100

- Maximum amount: PKR 50,000 per transaction
- Processing fee calculated and displayed clearly
- Secure redirect to JAZZ CASH (HTTPS, COM-1)
- Payment gateway loads within 3 seconds
- Transaction processed securely per PCI DSS (NFR-13)
- Success callback received from JAZZ CASH
- Wallet balance updates immediately: PKR 500 → PKR 2,500
- Success message: "Wallet recharged successfully. New balance: PKR 2,500"
- Transaction appears in transaction history (FR-8) with:
  - Transaction ID
  - Date/Time
  - Amount: +PKR 2,000
  - Status: "Completed"
  - Payment method: "JAZZ CASH"
- SMS/Email confirmation sent to user
- Transaction record stored in database with all details

Postconditions:

- Wallet balance increased by PKR 2,000
- Transaction logged for audit (NFR-16)
- Payment gateway notified of successful transaction

Test Case PS-002: Wallet Recharge with Voucher

Priority: Medium
Requirements: FR-10, FR-71, FR-72
Objective: Verify voucher-based wallet recharge

Preconditions:

- Advertiser logged in
- Valid voucher code exists: "SUMMER2025" worth PKR 1,000
- Current wallet balance: PKR 500
- Voucher not yet redeemed by this user

Test Steps:

1. Navigate to "Wallet" screen
2. Tap "Recharge Wallet" button
3. Select payment method: "Voucher"
4. Enter voucher code: "SUMMER2025"
5. Tap "Apply Voucher"

6. Review voucher details displayed
7. Confirm redemption

Expected Results:

- Voucher input field accepts alphanumeric codes
- Real-time validation on "Apply Voucher" click (FR-72)
- System checks: voucher validity, expiration date, usage status, user eligibility
- Voucher details displayed: value (PKR 1,000), expiry date, terms
- No processing fee for voucher recharge
- Confirmation message: "Voucher applied successfully"
- Wallet balance updates: PKR 500 → PKR 1,500
- Voucher marked as "Redeemed" in database
- User ID associated with voucher redemption record
- Transaction recorded in history showing "Voucher: SUMMER2025"
- Email confirmation sent

Postconditions:

- Voucher status: "Redeemed"
- Wallet increased by voucher value
- Voucher cannot be reused

Test Case PS-003: Invalid Voucher Code

Priority: Medium
Requirements: FR-10, FR-72
Objective: Verify system handles invalid voucher codes correctly

Preconditions:

- Advertiser on wallet recharge screen
- Testing various invalid scenarios

Test Steps:

1. Test expired voucher: "EXPIRED2024" (valid until 12/31/2024)
2. Test non-existent voucher: "INVALID123"
3. Test already used voucher: "USED2025" (redeemed by same user)
4. Test deactivated voucher: "DEACTIVATED" (manually disabled by admin)

Expected Results:
For expired voucher:

- Error: "This voucher has expired on 12/31/2024"

For non-existent voucher:

- Error: "Invalid voucher code. Please check and try again."

For already used voucher:

- Error: "You have already redeemed this voucher."

For deactivated voucher:

- Error: "This voucher is no longer active."

General behavior:

- No wallet balance change occurs
- No transaction record created
- Error messages clear and specific
- User can retry with different code
- Validation occurs before any processing

Postconditions:

- Wallet balance unchanged
- Invalid vouchers not processed
- System integrity maintained

Test Case PS-004: Driver Earnings Calculation

Priority: High
Requirements: FR-68, BR-4
Objective: Verify accurate calculation of driver earnings based on campaign participation

Preconditions:

- Driver completed campaign assignment
- Campaign parameters:
  - Duration: 10 days
  - Daily rate: PKR 200/day
  - Total potential earnings: PKR 2,000
- Driver verification records:
  - Day 1-7: Verified participation (7 days)
  - Day 8: No verification image uploaded
  - Day 9-10: Verified participation (2 days)

- Driver current wallet: PKR 500

Test Steps:

1. Campaign reaches end date
2. System processes completion
3. Calculate driver earnings:
    - Verified days: 9 out of 10
    - Base earnings: 9 × PKR 200 = PKR 1,800
    - Platform commission: 10% = PKR 180
    - Net earnings: PKR 1,620
4. System credits driver wallet
5. Driver views wallet and transaction history

Expected Results:

- System validates verification records (FR-79, FR-80)
- Unverified days excluded from payment calculation (BR-4)
- Earnings calculation transparent and itemized:
    - Verified days: 9
    - Rate: PKR 200/day
    - Gross: PKR 1,800
    - Commission (10%): -PKR 180
    - Net: PKR 1,620
- Driver wallet updates: PKR 500 → PKR 2,120
- Transaction record created showing:
    - Campaign ID and name
    - Calculation breakdown
    - Payment date
    - Status: "Completed"
- Driver receives notification: "Congratulations! You earned PKR 1,620 for completing Campaign: Summer Sale 2025"
- Earnings processed within 24 hours of campaign completion
- If driver queries missing day payment, system can show verification gap

Postconditions:

- Driver paid accurately for verified work
- Transaction recorded for both driver and advertiser
- Business rule BR-4 enforced

Test Case PS-005: Discount Coupon Application

Priority: Medium
Requirements: FR-70, FR-72
Objective: Verify discount coupon reduces campaign cost correctly

Preconditions:

- Advertiser creating new campaign
- Campaign cost calculated: PKR 5,000
- Valid coupon "FIRST20" exists: 20% off, max discount PKR 1,000
- Advertiser has not used this coupon before
- Wallet balance: PKR 4,500

Test Steps:

1. Complete campaign creation form
2. Review cost summary showing PKR 5,000
3. Tap "Apply Coupon" field
4. Enter coupon code: "FIRST20"
5. Tap "Apply"
6. Review updated cost
7. Complete campaign creation

Expected Results:

- Coupon validated against database (FR-72)
- System checks: code validity, expiration, usage limits, user eligibility
- Discount calculation: 20% of PKR 5,000 = PKR 1,000 (within max limit)
- Cost breakdown updates:
    - Base cost: PKR 5,000
    - Discount (FIRST20): -PKR 1,000
    - Final cost: PKR 4,000
- Green checkmark with message: "Coupon applied! You saved PKR 1,000"
- Final cost now within wallet balance
- Campaign creation proceeds successfully
- Wallet debited PKR 4,000 (not original PKR 5,000)
- Coupon marked as used for this user
- Transaction record shows coupon code and discount amount

Postconditions:

- Campaign created at discounted price
- Coupon usage recorded
- Advertiser cannot reuse same coupon (if single-use)

Test Case PS-006: Transaction History Display

Priority: Medium
Requirements: FR-8, FR-32
Objective: Verify transaction history is complete and accurate

Preconditions:

- User account with 15 transactions over 3 months:
  - 5 wallet recharges
  - 3 campaign payments
  - 4 driver earnings
  - 2 refunds
  - 1 voucher redemption

Test Steps:

1. Navigate to "Wallet" screen
2. Tap "Transaction History"
3. View default display (most recent first)
4. Apply filters:
   - Date range: Last 30 days
   - Transaction type: "Recharges only"
5. Search for specific transaction by ID
6. Export transaction history as PDF
7. Tap individual transaction for details

Expected Results:

- All 15 transactions displayed in chronological order (newest first)
- Each transaction shows:
  - Date and time
  - Transaction ID (unique, 8-12 characters)
  - Type (Recharge/Payment/Earning/Refund/Voucher)
  - Amount with +/- indicator
  - Status (Completed/Pending/Failed)
  - Balance after transaction
- Date range filter reduces display to matching transactions
- Type filter shows only recharge transactions (5 results)
- Search function locates transaction by ID instantly
- Transaction details page shows:
  - All basic info plus payment method, reference numbers, notes
- PDF export generates within 3 seconds with all transaction data formatted

- Infinite scroll or pagination for large transaction lists
- Real-time balance displayed at top
- Data matches database records exactly

Postconditions:

- User has complete financial transparency
- Export available for personal records

Test Case PS-007: Payment Failure Handling

Priority: High
Requirements: FR-9, FR-66, FR-67
Objective: Verify system handles payment gateway failures gracefully

Preconditions:

- Advertiser attempting wallet recharge
- Simulated payment gateway failure scenarios

Test Steps:

1. Initiate PKR 1,000 recharge via JAZZ CASH
2. Simulate various failure scenarios:
    - Scenario A: Gateway timeout (no response after 60 seconds)
    - Scenario B: Insufficient balance in JAZZ CASH account
    - Scenario C: Transaction declined by bank
    - Scenario D: User cancels payment at gateway
    - Scenario E: Network interruption during processing

Expected Results:
Scenario A (Timeout):

- Loading indicator shows for 60 seconds
- Timeout error: "Payment gateway is not responding. Please try again later."
- No wallet balance change
- Transaction status: "Failed - Timeout"

Scenario B (Insufficient balance):

- Gateway returns error
- Message: "Insufficient balance in your JAZZ CASH account. Please add funds and retry."
- User returned to CORADS app
- No wallet balance change

Scenario C (Declined):

- Message: "Transaction declined. Please contact your bank or try a different payment method."
- Transaction status: "Failed - Declined"

Scenario D (User cancellation):

- User returned to recharge screen
- Message: "Payment cancelled by user"
- No transaction record created

Scenario E (Network interruption):

- System detects connection loss
- Message: "Network error occurred. Checking transaction status..."
- System queries gateway for transaction status (prevents double charging)
- Displays actual status once network restored

General behavior:

- No partial/duplicate charges
- All failures logged with error codes
- User can retry immediately
- Clear error messages guide next steps
- No wallet balance changes unless payment confirmed successful
- Transaction records show failure reason

Postconditions:

- System state remains consistent
- No money lost in failed transactions
- Clear audit trail maintained

---

4. REAL-TIME TRACKING MODULE

Test Case RT-001: GPS Location Tracking Accuracy

Priority: High
Requirements: FR-73, NFR-3, HW-2
Objective: Verify accurate and timely GPS location tracking

Preconditions:

- Driver accepted campaign and started assignment

- Driver app running with GPS enabled
- Location permissions granted
- Driver moving in vehicle in target area
- Advertiser monitoring campaign

Test Steps:

1. Driver starts campaign at installation center
2. Driver travels predefined 10 km route over 30 minutes
3. System tracks location continuously
4. Advertiser views real-time tracking map
5. Record actual GPS coordinates vs displayed locations
6. Monitor tracking frequency and accuracy

Expected Results:

- Initial location captured at campaign start with accuracy within 10 meters
- Location updates transmitted every 30 seconds (NFR-3)
- GPS coordinates accurate within 10-20 meters of actual position
- Map marker position updates smoothly without jumping
- Route trail appears on map showing path traveled
- Distance counter increments accurately (±5% of actual distance)
- Location updates continue even when app in background
- Battery usage optimized (location accuracy balanced with power consumption)
- No location updates when driver stationary for >2 minutes (to save data)
- Updates resume when movement detected
- If GPS signal lost (tunnel, building), last known position displayed with indicator
- Signal restoration automatically resumes tracking
- All location data transmitted via HTTPS (COM-1)
- Location data stored with timestamps in database

Postconditions:

- Complete route history recorded
- Location data available for campaign reporting
- No gaps in tracking timeline (except during signal loss)

Test Case RT-002: Simultaneous Multiple Driver Tracking

Priority: High
Requirements: FR-74, NFR-2
Objective: Verify system handles tracking multiple drivers concurrently

Preconditions:

- Campaign active with 10 drivers
- All drivers moving in different areas of city
- System under normal load conditions
- Advertiser viewing campaign tracking map

Test Steps:

1. All 10 drivers have campaign active simultaneously
2. Each driver traveling different routes
3. Advertiser opens tracking map
4. System loads and displays all driver locations
5. Monitor real-time updates for all drivers
6. Verify system performance metrics
7. Check database query efficiency

Expected Results:

- Map loads within 3 seconds showing all 10 driver markers (NFR-1)
- Each driver represented by unique colored marker with ID/name
- All markers update independently every 30 seconds
- No performance degradation with multiple concurrent trackers
- System supports minimum 1000 concurrent users (NFR-2) - test subset of 10 drivers
- Map remains responsive to user interactions (zoom, pan)
- No UI lag or freezing
- Websocket connections maintained for each driver (COM-2)
- Database queries optimized (indexed location tables)
- Server CPU usage remains below 70%
- Memory usage stable (no leaks)
- Network bandwidth utilized efficiently (batched updates)
- Location data segregated by campaign for multi-campaign scenarios
- Heat map overlay updates to reflect coverage concentration

Postconditions:

- All drivers tracked accurately without performance issues
- System demonstrates scalability for production load

Test Case RT-003: Route Deviation Detection

Priority: Medium
Requirements: FR-76, FR-77, FR-78

Objective: Verify system detects when driver deviates from target area

Preconditions:

- Campaign defined for "Gulberg, Lahore" with 5 km radius
- Driver accepted campaign and active
- Driver traveling within target area initially
- Expected route behavior established

Test Steps:

1. Driver starts campaign within target area
2. Driver travels normally for 15 minutes (within boundary)
3. Driver exits target area boundary by 2 km
4. System monitors route patterns
5. Admin views route analysis dashboard
6. Driver returns to target area
7. Review deviation alerts and logs

Expected Results:

- Geofence created around target area (5 km radius from center point)
- System continuously monitors driver position against geofence
- When driver exits boundary:
    - Real-time alert triggered within 60 seconds
    - Admin notification: "Driver [Name] has left campaign area for Campaign [ID]"
    - Alert appears on admin dashboard with timestamp
    - Driver location highlighted in red on map
    - Warning notification sent to driver: "You are outside the campaign area. Please return to continue earning."
- Deviation recorded in database:
    - Time exited: timestamp
    - Distance from boundary: 2 km
    - Duration outside: tracked continuously
- Route analysis shows:
    - Percentage of time in target area: calculated
    - Total distance outside boundaries: logged
- When driver returns to boundary:
    - Marker returns to normal color
    - "Returned to area" notification sent to admin
    - Resume time logged
- Payment calculation adjusted for time outside area (if per business rules)
- Route pattern analysis identifies:
    - Intentional vs accidental deviations

    - Frequency of boundary crossings
    - Anomalous movement patterns

Postconditions:

- Complete route deviation history maintained
- Data available for verification and payment calculation
- Patterns flagged for quality control

Test Case RT-004: Verification Image with Location Data

Priority: High
Requirements: FR-79, FR-80, FR-81, HW-3
Objective: Verify drivers can submit verification images with embedded location

Preconditions:

- Driver has active campaign
- Campaign requires verification image every 2 hours
- Driver at valid location within target area
- Camera and location permissions granted
- 4 hours have elapsed since campaign start (2 verifications due)

Test Steps:

1. Driver app displays notification: "Verification image required"
2. Driver taps notification to open camera
3. Driver captures photo of vehicle with advertisement visible
4. System embeds GPS metadata in image
5. Driver submits verification image
6. System validates submission
7. Admin reviews verification in dashboard
8. Repeat for second verification image

Expected Results:

- Verification prompt appears at 2-hour intervals
- Camera opens with overlay showing verification instructions
- Image capture includes:
    - GPS coordinates (latitude, longitude)
    - Timestamp (date, time)
    - Accuracy reading (<20 meters)
- Image preview shows before submission
- Retake option available if image unclear

- On submission:
  - GPS data extracted and validated (FR-80)
  - Location checked against current tracking position (within 100m tolerance)
  - Location verified within campaign target area
  - Image uploaded to cloud storage
  - Unique identifier generated
  - Database record created linking: image, driver, campaign, GPS, timestamp
- Success message: "Verification submitted successfully"
- Verification counter updates in driver dashboard (1/5 verifications completed)
- Admin dashboard shows verification:
  - Thumbnail image
  - Location on map with marker
  - Timestamp
  - Status: "Verified" (green indicator)
- If GPS mismatch detected (image location differs from tracking location by >500m):
  - Warning flag raised
  - Manual review required
  - Alert sent to admin for investigation
- Images compressed for efficient storage without quality loss
- Metadata preserved in EXIF data

Postconditions:

- Verification image stored with complete metadata
- Driver participation verified for payment eligibility (BR-4)
- Audit trail maintained per FR-81
- Admin can review verification history anytime

Test Case RT-005: Offline Mode Location Caching

Priority: Medium
Requirements: FR-73, NFR-3
Objective: Verify location tracking continues during temporary connectivity loss

Preconditions:

- Driver has active campaign running
- Initial connectivity established
- GPS signal strong
- Driver traveling through area with intermittent connectivity

Test Steps:

1. Driver campaign active with normal tracking
2. Simulate network disconnection (WiFi/data off)
3. Driver continues traveling for 10 minutes
4. Verify app behavior during offline period
5. Restore network connectivity
6. Monitor data synchronization

Expected Results:

- App detects connectivity loss immediately
- Offline mode indicator appears in driver app
- GPS tracking continues using device hardware (HW-2)
- Location data cached locally in device storage (HW-4):
    - Every 30 seconds per normal schedule
    - Stored in local SQLite database
    - Queue created for pending uploads
- User interface shows:
    - "Operating in offline mode"
    - "Location data will sync when connected"
    - Last sync timestamp displayed
- App remains functional for core features
- No crashes or errors during offline operation
- When connectivity restored:
    - Auto-detect within 10 seconds
    - Sync process initiates automatically
    - Cached location data uploaded to server in chronological order
    - Progress indicator shows "Syncing X of Y locations"
    - Server reconciles and validates uploaded data
    - Gaps in real-time tracking timeline filled retroactively
    - Success message: "All data synchronized successfully"
- Advertiser's tracking view updates with complete route (no gaps)
- Database contains complete continuous tracking record
- Duplicate location entries prevented during sync
- Battery usage optimized during offline mode

Postconditions:

- No location data lost during connectivity issues
- Complete tracking history maintained
- System gracefully handles network instability

Test Case RT-006: Heat Map Generation for Campaign Coverage

Priority: Medium
Requirements: FR-64
Objective: Verify accurate heat map showing advertisement exposure concentration

Preconditions:

- Campaign completed with 5 drivers over 10 days
- Extensive location data collected (approximately 14,400 data points per driver)
- Campaign target area: Gulberg, Lahore
- Advertiser viewing completed campaign report

Test Steps:

1. Navigate to completed campaign
2. Tap "View Performance Report"
3. Access "Coverage Heat Map" section
4. Review heat map visualization
5. Interact with heat map controls:
    - Zoom in/out
    - Toggle data layers
    - Filter by date range
    - Filter by specific drivers
6. Export heat map as image

Expected Results:

- Heat map loads within 5 seconds with all location data processed
- Map overlays campaign target area
- Color gradient visualization:
    - Red: High exposure (multiple driver passes, long dwell times)
    - Orange: Medium exposure
    - Yellow: Low exposure
    - No color: No exposure
- Intensity calculated based on:
    - Number of driver passes through area
    - Total time spent in area
    - Number of different drivers covering area
- Major roads and high-traffic areas show intense colors
- Coverage extends across target area with identifiable patterns
- Zoom functionality works smoothly (maintains data granularity)
- Layer toggles allow viewing:
    - Individual driver routes
    - Time-based coverage (different days)
    - Peak hours vs off-peak

- Date range filter recalculates heat map for selected period
- Driver filter shows coverage from specific drivers only
- Interactive tooltips on hover show:
    - Number of passes
    - Total exposure time
    - Drivers who covered area
- Legend clearly explains color intensity scale
- Export generates high-resolution PNG/JPEG image
- Heat map provides actionable insights:
    - Coverage gaps identifiable
    - High-traffic areas visible
    - ROI estimation possible based on exposure

Postconditions:

- Advertiser has visual representation of campaign reach
- Data-driven insights available for future campaigns
- Export available for presentations/reports

Test Case RT-007: Battery Optimization During Extended Tracking

Priority: Medium
Requirements: NFR-7, HW-2
Objective: Verify tracking system optimizes battery usage during long campaigns

Preconditions:

- Driver starting 8-hour campaign
- Smartphone with 100% battery
- GPS and data connectivity enabled
- Baseline battery consumption measured for normal usage

Test Steps:

1. Driver starts campaign with full battery
2. System initiates location tracking
3. Monitor battery consumption over 8 hours
4. Measure battery usage breakdown:
    - GPS module
    - Data transmission
    - App background processes
    - Screen time
5. Compare against baseline consumption

6. Test adaptive tracking features

Expected Results:

- Location tracking optimized for battery efficiency:
  - GPS accuracy set to "Balanced" mode (not "High accuracy" constantly)
  - Location updates every 30 seconds (not continuous)
  - Batch uploads of location data (every 5 minutes, not real-time individual)
  - Background GPS throttling when stationary
- Battery consumption for 8-hour campaign: approximately 25-35% total battery
- App battery usage breakdown:
  - GPS: ~15-20%
  - Network: ~5-8%
  - Processing: ~3-5%
- Adaptive features activated:
  - When battery <20%: Location update interval increases to 60 seconds
  - When battery <10%: User prompted with low battery warning and option to pause campaign
  - Power-saving suggestions offered
- No significant heat generation from device
- App does not drain battery faster than specified in NFR
- Driver can complete full 8-hour campaign without battery death
- If device enters power-saving mode, tracking continues but at reduced frequency
- User notified of any tracking adjustments due to battery optimization
- Location accuracy maintained despite optimization (within 20m)

Postconditions:

- Campaign completed with sufficient battery for driver safety
- Tracking data completeness maintained despite optimizations
- Driver experience not negatively impacted

---

5. SECURITY MODULE

Test Case SEC-001: Password Hashing and Storage

Priority: Critical
Requirements: NFR-11, NFR-13
Objective: Verify passwords are securely hashed and never stored in plaintext

Preconditions:

- Access to backend database (test environment)

- Multiple user accounts created
- Database inspection tools available

Test Steps:

1. Create new user account with password: "TestPass123!"
2. Query database for user record
3. Examine password field in database
4. Attempt to reverse-engineer password from hash
5. Verify hashing algorithm used
6. Test password comparison during login
7. Change user password and verify new hash generated
8. Review password storage across all user types (advertiser, driver, admin)

Expected Results:

- Password never transmitted or stored in plaintext
- Database password field contains hashed value only
- Hash appears as random string (e.g., "$2y$10$abcd1234...") approximately 60 characters
- Strong hashing algorithm used (bcrypt, Argon2, or PBKDF2)
- Hash includes salt (unique per password, prevents rainbow table attacks)
- Same password for different users produces different hashes (due to unique salts)
- Hash is irreversible (one-way function)
- Password verification uses secure comparison function (time-constant to prevent timing attacks)
- Failed reverse-engineering attempts confirm hash security
- Login process:
    - User submits password
    - System hashes submitted password
    - Compares hash against stored hash (not plaintext comparison)
    - Grants access only on match
- Password change generates entirely new hash with new salt
- Old password hash permanently replaced (not stored in history for this basic test)
- All user types employ same secure hashing mechanism
- No password visible in server logs
- No password in API request/response bodies
- HTTPS encryption additional layer (COM-1) during transmission

Postconditions:

- Password security verified to industry standards
- NFR-11 compliance confirmed
- Database secure against password exposure attacks

Test Case SEC-002: Session Management and Timeout

Priority: High
Requirements: NFR-14
Objective: Verify user sessions timeout after 30 minutes of inactivity

Preconditions:

- User logged into advertiser mobile app
- Active session established
- User on dashboard screen

Test Steps:

1. User logs in successfully at 10:00 AM
2. User navigates to various screens (activity within session)
3. User leaves app idle at 10:15 AM (no interaction)
4. Monitor session status over time
5. Attempt to access protected features at:
    - 10:30 AM (15 minutes idle)
    - 10:45 AM (30 minutes idle)
    - 10:46 AM (31 minutes idle)
6. Test session refresh on activity
7. Verify session data cleanup

Expected Results:

- Session token (JWT) generated on successful login
- Token includes expiration timestamp (30 minutes from last activity)
- User activity (screen navigation, button taps, API calls) refreshes session timer
- At 10:30 AM (15 minutes idle):
    - Session still valid
    - User can access protected features
    - No interruption to user experience
- At 10:45 AM (30 minutes idle):
    - Session still valid (at boundary)
    - Warning notification may appear: "Your session will expire soon due to inactivity"
    - User can continue using app
- At 10:46 AM (31 minutes idle):
    - Session expired
    - Any API request returns 401 Unauthorized
    - User automatically redirected to login screen
    - Message: "Your session has expired due to inactivity. Please log in again."

    - User data cleared from app memory
    - Sensitive information removed from cache
- Session refresh behavior:
    - Each user action extends timeout by 30 minutes from that action
    - Multiple rapid actions don't cause multiple session refreshes (debounced)
- Security token invalidated on server after expiration
- User must re-authenticate to create new session
- Different session for each device (multi-device login tracked separately)
- Admin can view active sessions per user in dashboard
- Forced logout doesn't count as inactivity timeout

Postconditions:

- Inactive sessions automatically terminated (NFR-14 compliant)
- Session security maintained
- User data protected from unauthorized access on abandoned devices

---

Test Case SEC-003: SQL Injection Prevention

Priority: Critical
Requirements: NFR-12, NFR-15
Objective: Verify system is protected against SQL injection attacks

Preconditions:

- Test environment with database monitoring enabled
- User at login screen
- Various input fields available for testing across application

Test Steps:

1. Test login form with SQL injection attempts:
    - Username: `admin' OR '1'='1`
    - Password: `password' OR '1'='1`
2. Test search functionality with malicious queries:
    - Campaign search: `test'; DROP TABLE campaigns; --`
    - User search: `' UNION SELECT * FROM users WHERE '1'='1`
3. Test profile update with injection:
    - Name field: `John'; UPDATE users SET role='admin' WHERE name='John`
4. Test campaign creation with payload:
    - Description: `Sale<script>alert('XSS')</script>`
5. Monitor database logs for actual executed queries

6. Review input sanitization mechanisms

Expected Results:

- All SQL injection attempts blocked before reaching database
- Input validation and sanitization applied to all user inputs:
  - Special SQL characters escaped or rejected
  - Parameterized queries/prepared statements used exclusively
  - User input never concatenated directly into SQL queries
- Login attempt with injection string:
  - Treated as literal username/password strings
  - Authentication fails (no user with that exact username)
  - No database structure exposed
  - No unauthorized access granted
  - Error message: "Invalid username or password" (generic, no details)
- Search injection attempts:
  - Malicious code not executed as SQL
  - Table drop command not executed
  - Database schema remains intact
  - UNION SELECT attack prevented
  - Search returns no results or error message
- Profile update injection:
  - User role not elevated
  - Database entry not maliciously modified
  - Name saved as literal string (including SQL syntax as text)
- XSS in campaign description:
  - Script tags escaped or stripped during input
  - Output encoding applied during display
  - Script not executed when rendered
  - Displayed as plain text: `Sale<script>alert('XSS')</script>` or `Salealert('XSS')`
- Database logs show parameterized queries only:
  - Example: `SELECT * FROM users WHERE username = ? AND password = ?`
  - User input bound as parameters, not in query string
- Rate limiting triggers on repeated injection attempts (NFR-15):
  - After 5 failed attempts from same IP in 5 minutes
  - IP temporarily blocked for 15 minutes
  - CAPTCHA challenge presented
- WAF (Web Application Firewall) rules detect and block malicious patterns
- Error messages generic (don't reveal database structure or query details)
- All API endpoints protected with input validation middleware

Postconditions:

- System demonstrates strong defense against SQL injection
- Database integrity maintained
- Attack attempts logged for security monitoring (NFR-16)

Test Case SEC-004: API Rate Limiting and Brute Force Protection

Priority: High
Requirements: NFR-15
Objective: Verify rate limiting prevents brute force attacks

Preconditions:

- Automated testing tool configured
- User account with known credentials
- Clean rate limit state (no previous violations)

Test Steps:

1. Attempt login with incorrect password repeatedly:
    - Send 20 login requests in 60 seconds with wrong password
    - Monitor system response after each attempt
2. Test API endpoint rate limiting:
    - Send 100 campaign creation requests in 1 minute (exceeds NFR-5 limit)
3. Test successful login after rate limiting triggered
4. Wait for rate limit window to reset
5. Test rate limiting across different IP addresses

Expected Results:

- Login attempt rate limiting (NFR-15):
    - Attempts 1-3: Normal processing, returns "Invalid credentials"
    - Attempts 4-5: Warning added to response: "Multiple failed attempts detected"
    - Attempt 6+: Account temporarily locked
    - Response: "Too many failed login attempts. Please try again in 15 minutes."
    - HTTP Status: 429 Too Many Requests
    - Countdown timer shown to user
    - Email sent to account owner about suspicious activity
- Campaign creation rate limiting:
    - First 100 requests within 1 minute processed normally (per NFR-5 limit)
    - Requests beyond limit rejected immediately
    - Response: "Rate limit exceeded. Maximum 100 campaigns per minute allowed."

    - HTTP Status: 429 Too Many Requests
    - Retry-After header included in response (seconds until reset)
- Rate limit implementation:
    - Per-IP address tracking
    - Per-user account tracking (authenticated requests)
    - Sliding window algorithm (not fixed time buckets)
    - Redis or in-memory cache stores rate limit counters
- Successful login after lockout period:
    - After 15-minute wait, login succeeds with correct credentials
    - Rate limit counter resets
    - Normal operation resumes
- Different rate limits for different endpoints:
    - Login: 5 attempts per 5 minutes
    - API reads: 1000 requests per hour
    - API writes: 100 requests per hour
    - Password reset: 3 attempts per hour
- Rate limits differentiated by user type:
    - Admin: Higher limits
    - Advertiser/Driver: Standard limits
    - Unauthenticated: Lowest limits
- IP-based blocking prevents distributed attacks
- CAPTCHA challenge presented after repeated violations
- Legitimate users not impacted by rate limiting during normal usage
- Rate limit bypass attempts logged (NFR-16)

Postconditions:

- Brute force attacks effectively mitigated
- System remains available for legitimate users
- Security incidents logged for investigation

Test Case SEC-005: Data Encryption in Transit

Priority: Critical
Requirements: NFR-12, COM-1
Objective: Verify all API communications encrypted using TLS/HTTPS

Preconditions:

- Network traffic monitoring tools installed (Wireshark, Fiddler)
- Mobile app and admin portal active
- Test environment with valid SSL certificate

Test Steps:

1. Capture network traffic during various operations:
    - User login (username, password transmission)
    - Campaign creation (including images)
    - Payment processing (wallet recharge)
    - Location data transmission
    - Admin portal access
2. Analyze captured packets
3. Verify encryption protocols used
4. Attempt to intercept and read unencrypted data
5. Test HTTPS enforcement (HTTP redirect)
6. Verify certificate validity

Expected Results:

- All HTTP connections automatically redirect to HTTPS
- TLS 1.2 or higher used for encryption (TLS 1.3 preferred)
- Valid SSL/TLS certificate installed on server:
    - Not expired
    - Matches domain name
    - Issued by trusted certificate authority
    - Certificate chain complete
- All API endpoints enforce HTTPS (no HTTP option)
- Network traffic analysis shows:
    - Encrypted packet payload (unreadable binary data)
    - No plaintext credentials, personal data, or sensitive information visible
    - SSL/TLS handshake successful
    - Strong cipher suites used (AES-256, AES-128)
- Login credentials transmission:
    - Username and password encrypted in transit
    - Visible only as encrypted data in packets
    - Decryption requires private key (unavailable to attacker)
- Payment data transmission:
    - Credit card/JAZZ CASH details encrypted
    - Complies with PCI DSS requirements (NFR-13)
    - Tokenization used where applicable
- Location data encrypted during transmission:
    - GPS coordinates not readable in captured traffic
    - Real-time updates protected
- Image uploads encrypted:
    - Binary data encrypted in transit
    - No image content visible in intercepted packets

- Mobile app certificate pinning (optional but recommended):
  - App validates server certificate against expected certificate
  - Prevents man-in-the-middle attacks
- Admin portal enforces HTTPS:
  - HTTP requests (http://admin.corads.com) redirect to HTTPS (https://admin.corads.com)
  - HSTS header present (HTTP Strict Transport Security)
  - Prevents protocol downgrade attacks
- WebSocket connections also encrypted (WSS://)
- Error messages don't expose encryption details

Postconditions:

- All data in transit protected per NFR-12
- Interception attacks ineffective
- User data privacy maintained

Test Case SEC-006: Admin Role-Based Access Control

Priority: High
Requirements: FR-37, NFR-16
Objective: Verify admin roles have appropriate access levels and permissions

Preconditions:

- Three admin accounts created with different roles:
  - Super Admin (full access)
  - Campaign Manager (campaign and user management only)
  - Support Admin (view-only, support ticket management)
- Admin portal accessible

Test Steps:

1. Login as Super Admin:
   - Access all admin features
   - Perform sensitive operations (delete user, modify payment settings)
   - View audit logs
2. Login as Campaign Manager:
   - Attempt to access all features
   - Try to delete user accounts
   - Try to modify system configuration
   - Manage campaigns and users successfully
3. Login as Support Admin:
   - Attempt write operations

    - View support tickets and user data
    - Try to modify campaigns or users
4. Test unauthorized access attempts:
    - Manipulate URL to access restricted pages
    - Send API requests for unauthorized operations
5. Review audit logs for all actions

Expected Results:

- Super Admin access:
    - All menu items and features visible and accessible
    - Can perform all operations: create, read, update, delete
    - Dashboard shows comprehensive system statistics
    - Can add/remove other admin accounts
    - Can modify system configurations (coupons, vouchers, car types - FR-49, FR-50, FR-51)
    - Can view complete audit logs (NFR-16)
    - All actions logged with admin ID and timestamp
- Campaign Manager access:
    - Campaign management features fully accessible (FR-45, FR-46, FR-47, FR-48)
    - User management features available (FR-39, FR-40, FR-41, FR-42)
    - Can send notifications (FR-43, FR-44)
    - System configuration menu hidden/disabled
    - Attempt to delete user account succeeds (within role permissions)
    - Attempt to access system settings returns: "403 Forbidden - Insufficient permissions"
    - Audit log access limited to own actions
    - Dashboard shows relevant statistics only (campaigns, users)
- Support Admin access:
    - View-only access to users, campaigns, drivers
    - Support ticket management fully accessible (FR-82, FR-83, FR-84)
    - Can view user profiles and campaign details
    - Can respond to support tickets
    - Cannot modify user accounts: "Edit" buttons disabled or return error
    - Cannot activate/deactivate campaigns: "Action not permitted for your role"
    - Cannot modify system configurations
    - Dashboard shows support-related metrics only
    - All write operation attempts logged as unauthorized
- URL manipulation protection:
    - Direct navigation to /admin/system-config as Campaign Manager redirects to dashboard with error message
    - API endpoint /api/admin/users/delete requires Super Admin role
    - Unauthorized API calls return 403 Forbidden with error: "Insufficient permissions"
- Role verification:

    - Every protected operation checks user role against required permissions
    - Role stored in JWT token, validated on every request
    - Role cannot be modified by user (stored server-side)
    - Role changes require Super Admin authentication
- Audit logging (NFR-16):
    - All admin actions logged: user ID, role, action, timestamp, IP address, result
    - Unauthorized access attempts logged with alert flag
    - Logs immutable (append-only)
    - Logs retained indefinitely for security compliance
- Session management per role:
    - Different timeout periods possible per role
    - Simultaneous sessions tracked separately

Postconditions:

- Role-based access control effectively enforced
- Sensitive operations restricted to authorized admins only
- Complete audit trail of admin activities
- System security posture maintained

---

6. EDGE CASE TESTING

Test Case EC-001: Campaign Creation at Exact Midnight

Priority: Medium
Requirements: FR-11, FR-12
Objective: Verify system handles campaign scheduling edge cases

Preconditions:

- Advertiser logged in at 11:58 PM
- System clock accurate and synchronized

Test Steps:

1. Start creating campaign at 11:58 PM
2. Set start date: "Today" (current date)
3. Set start time: 12:00 AM (midnight)
4. Set end date: 7 days later
5. Complete other campaign fields
6. Submit campaign at exactly 11:59:59 PM
7. Verify campaign status after midnight (12:00:01 AM)

Expected Results:

- Date/time picker handles midnight boundary correctly
- Start date "Today" resolves to correct date even when created before midnight
- System timezone handling consistent (server time vs local time)
- Campaign submitted at 11:59:59 PM recorded with correct timestamp
- Campaign start time validation:
  - If start time = 12:00 AM and current time = 11:59 PM, validation accepts (starts in 1 minute)
  - Campaign state transitions correctly at midnight
- After midnight:
  - Campaign automatically becomes eligible for activation
  - Campaign moves from "Pending" to "Ready" if admin-approved
  - Date calculations for duration remain accurate (exactly 7 days)
- No off-by-one errors in duration calculation
- Campaign end date calculated correctly: 7 days from start, at 11:59:59 PM of final day
- Daylight saving time changes considered if applicable
- Multiple campaigns created at midnight don't conflict
- Database timestamp stored in UTC, displayed in user's local timezone

Postconditions:

- Campaign scheduled accurately despite timing edge case
- No temporal anomalies in campaign lifecycle

Test Case EC-002: Location Tracking at Country Border

Priority: Low
Requirements: FR-73, FR-76
Objective: Verify tracking behavior when driver crosses international border

Preconditions:

- Driver campaign active in Lahore (near Pakistan-India border region)
- Campaign target area near border
- Driver has necessary travel documents (hypothetical scenario)

Test Steps:

1. Driver traveling within Pakistan territory in target area
2. Driver inadvertently crosses into neighboring territory
3. Monitor location tracking behavior
4. Verify system responses
5. Driver returns to Pakistan

Expected Results:

- GPS continues tracking regardless of political boundaries
- Location coordinates recorded accurately on both sides of border
- If campaign geofence is country-specific:
  - Alert triggered when crossing border (similar to target area exit)
  - Admin notification: "Driver has left country boundary"
  - Driver notification: "You have exited the campaign country. Campaign paused."
  - Campaign time tracking pauses automatically
  - Earnings calculation excludes time outside country
- Location data stored with country identifier
- Map display may show different map provider data for different countries
- No data loss during border crossing
- Google Maps API handles international coordinates correctly (SW-3)
- Currency considerations if payment involves exchange rates (out of scope but noted)
- Legal/regulatory compliance checked for cross-border data collection
- System respects data sovereignty laws
- Driver safety prioritized (no tracking in restricted military zones if applicable)

Postconditions:

- Complete tracking history maintained
- Cross-border movement documented
- Campaign integrity preserved

Test Case EC-003: Simultaneous Campaign Completion

Priority: Medium
Requirements: FR-68, FR-63
Objective: Verify system handles multiple campaigns ending at exact same time

Preconditions:

- 50 different campaigns all scheduled to end at 11:59:59 PM on same date
- All campaigns have different advertisers and driver assignments
- System approaching end time

Test Steps:

1. Monitor system as campaigns approach end time
2. All 50 campaigns reach end time simultaneously
3. System processes campaign completions
4. Verify report generation for all campaigns

5. Verify payment calculations for all drivers
6. Check system performance during bulk processing

Expected Results:

- System handles concurrent campaign completions without failure
- Background job queue processes completions:
  - Jobs queued for each campaign
  - Processing begins within 60 seconds of end time
  - Jobs processed in parallel (up to system capacity)
- Each campaign completion:
  - Status updated from "Active" to "Completed"
  - Final location data collected
  - Driver verification records validated
  - Performance report generated (FR-63)
  - Driver earnings calculated (FR-68)
  - Payment processing initiated
  - Advertisers notified
  - Drivers notified
- No database deadlocks or race conditions
- Transaction isolation prevents data corruption
- Server resource usage monitored:
  - CPU spikes acceptable (up to 90% temporarily)
  - Memory usage remains within limits
  - Database connections managed efficiently
- All 50 campaigns complete processing within 10 minutes
- Reports generated correctly for each campaign (independent, no data mixing)
- Payments calculated accurately (no rounding errors accumulated)
- Email/notification queues handle bulk messaging
- No campaign marked completed incorrectly
- System remains responsive to other user requests during processing
- Logging captures all completion events with timestamps

Postconditions:

- All campaigns successfully transitioned to completed status
- All stakeholders notified
- System stability maintained under peak load

Test Case EC-004: Maximum Character Input in Text Fields

Priority: Low

Requirements: FR-11, FR-12

Objective: Verify system handles maximum length text inputs correctly

Preconditions:

- User on campaign creation screen
- Text prepared for maximum length testing

Test Steps:

1. Test campaign title field:
    - Enter 255 characters (assumed maximum)
    - Attempt to enter 256 characters
2. Test campaign description field:
    - Enter 5,000 characters (assumed maximum)
    - Attempt to enter 5,001 characters
3. Test profile name field:
    - Enter 100 characters
4. Submit forms with maximum length inputs
5. Verify data storage and retrieval

Expected Results:

- Each text field has defined maximum length:
    - Campaign title: 255 characters
    - Campaign description: 5,000 characters
    - User name: 100 characters
    - Email: 255 characters
- Field validation enforces maximum:
    - Character counter displays remaining characters (e.g., "50/255 characters")
    - Input prevented when maximum reached (typing doesn't add more characters)
    - Paste operations truncated to maximum length
    - Warning message: "Maximum character limit reached"
- Maximum length inputs accepted and saved successfully
- No truncation during database storage (VARCHAR fields sized appropriately)
- Retrieved data matches submitted data exactly (no data loss)
- Display formatting handles long text gracefully:
    - Word wrapping applied
    - Scrollable containers for long content
    - Ellipsis with "Show more" option where appropriate
- Special characters counted correctly (Unicode, emojis count as expected)
- Database schema supports maximum defined lengths:
    - Title: VARCHAR(255)

    - Description: TEXT or VARCHAR(5000)
- No buffer overflow vulnerabilities
- Form submission succeeds with maximum length data
- API payload size acceptable (within limits)
- No performance degradation with maximum length inputs

Postconditions:

- Maximum length constraints enforced
- Data integrity maintained
- User experience clear regarding limits

Test Case EC-005: Wallet Balance Exactly Zero During Campaign Creation

Priority: Medium
Requirements: FR-7, FR-11, BR-2
Objective: Verify system behavior when wallet balance reaches exactly zero

Preconditions:

- Advertiser wallet balance: PKR 3,000
- Creating campaign with cost: PKR 3,000
- No other pending charges

Test Steps:

1. Initiate campaign creation
2. Complete all campaign parameters
3. System calculates cost: PKR 3,000
4. Review and confirm campaign
5. System processes payment
6. Verify wallet balance after deduction
7. Immediately attempt to create another campaign (cost PKR 100)

Expected Results:

- Initial wallet display shows: PKR 3,000
- Campaign cost calculation: PKR 3,000
- Validation check passes (balance >= cost)
- Campaign creation proceeds
- Payment processing:
    - Wallet debited: PKR 3,000

    - New balance: PKR 0.00
    - Transaction recorded
- Wallet screen displays: PKR 0.00 (not negative, not error)
- Attempt to create second campaign (PKR 100):
    - Validation fails immediately
    - Error: "Insufficient wallet balance. Please recharge your wallet."
    - Campaign creation blocked per BR-2
    - Prominent "Recharge Wallet" button displayed
- Edge case handling:
    - Balance = 0 treated as insufficient for any campaign
    - No campaigns allowed with zero balance
    - Minimum campaign cost > 0 enforced
- No race condition if concurrent campaign creation attempted
- Database transaction ensures atomicity:
    - Balance deduction and campaign creation succeed together or fail together
    - No partial state (campaign created but balance not deducted)
- Wallet transaction history shows accurate balance after each transaction
- System doesn't allow negative balance under any circumstance

Postconditions:

- Wallet at exactly zero handled correctly
- Further spending blocked until recharge
- Data consistency maintained

Test Case EC-006: Driver Accepts Multiple Campaign Requests Simultaneously

Priority: High
Requirements: FR-25, FR-26, FR-58
Objective: Verify system prevents driver from accepting conflicting campaigns

Preconditions:

- Driver available and online
- 3 different campaigns send requests to same driver simultaneously:
    - Campaign A: 10-12 PM, Lahore
    - Campaign B: 11 AM-1 PM, Lahore (overlaps with A)
    - Campaign C: 2-4 PM, Lahore (no overlap)
- All requests arrive within 2-second window

Test Steps:

1. Driver receives 3 notifications simultaneously
2. Driver views all three campaign requests on screen
3. Driver attempts to accept Campaign A
4. Before Campaign A acceptance processes, driver taps accept on Campaign B
5. Driver then attempts to accept Campaign C
6. Monitor system response to each acceptance

Expected Results:

- All 3 campaign requests displayed in driver app with details
- Driver can review each request independently
- First acceptance (Campaign A):
    - Processing begins immediately
    - Loading indicator shown
    - Database record created: driver-campaign assignment
    - Driver status changed to "Assigned"
- Second acceptance attempt (Campaign B) while A processing:
    - System checks driver availability in real-time
    - Detects time overlap: Campaign A (10-12 PM) vs Campaign B (11 AM-1 PM)
    - Prevents acceptance
    - Error: "You have already accepted a campaign during this time. Please complete or cancel your current assignment."
    - Campaign B request remains in "Pending" state
    - Another driver can be notified for Campaign B
- Third acceptance attempt (Campaign C):
    - System checks for conflicts
    - No time overlap detected (2-4 PM vs 10-12 PM)
    - Acceptance allowed if driver can handle multiple campaigns
    - Or blocked if business rule allows only one campaign at a time
    - If allowed: Success message, second assignment created
    - If blocked: "You can only accept one campaign at a time"
- Concurrent acceptance protection:
    - Database-level locking prevents race conditions
    - Optimistic locking or pessimistic locking employed
    - Only first acceptance commits if simultaneous
    - Second attempt receives error due to changed state
- Campaign status updates:
    - Campaign A: Driver assigned confirmed
    - Campaign B: Continues searching for alternative driver
    - Campaign C: Assigned to driver (if no conflicts) or available for others
- Notifications updated:
    - Accepted campaigns show "Accepted" status

    - Rejected/failed campaigns show "Available" or removed
- Driver dashboard reflects accurate current assignments

Postconditions:

- No conflicting campaign assignments created
- One driver doesn't get double-booked
- Campaign distribution system maintains integrity

Test Case EC-007: Image Upload During Poor Network Conditions

Priority: Medium
Requirements: FR-12, FR-54, FR-80
Objective: Verify image upload resilience during unstable connectivity

Preconditions:

- Driver submitting verification image
- Image size: 3 MB
- Network condition: Slow 3G with intermittent drops (simulated)

Test Steps:

1. Driver captures verification image
2. Driver taps "Submit"
3. Upload begins over slow/unstable network
4. Simulate network conditions:
    - Initial upload: 30% complete
    - Network drop: 5-second disconnection
    - Resume: Network restored
    - Slow speed: Upload continues at 50 kbps
    - Another drop: 3-second disconnection
    - Final resume: Upload completes
5. Monitor upload process and system behavior

Expected Results:

- Upload progress indicator displayed (0-100%)
- Initial upload begins, progress bar shows 30%
- First network drop:
    - Upload pauses automatically
    - Status message: "Connection lost. Retrying..."
    - No error thrown immediately
    - Progress bar remains at 30% (not reset to 0)

- Automatic retry mechanism:
  - System detects network restoration within 10 seconds
  - Upload resumes from last successful chunk (30%)
  - Message: "Connection restored. Resuming upload..."
  - Chunked upload protocol ensures resume capability (not restart from beginning)
- Slow network handling:
  - Upload continues despite low speed
  - Progress bar updates gradually
  - Estimated time remaining displayed and updates
  - No timeout error if steady progress continues
  - User can cancel if desired (Cancel button available)
- Second network drop:
  - Same resilient behavior as first drop
  - Resume from 60% (wherever it paused)
  - Retry counter tracked (max 5 retries)
- Upload completion:
  - Final chunk uploaded successfully
  - Progress reaches 100%
  - Server validates complete file integrity (checksum/hash verification)
  - Success message: "Image uploaded successfully"
  - Image metadata (location, timestamp) saved with image
- Failure scenario handling (if 5 retries exhausted):
  - Error message: "Upload failed due to poor connection. Please try again when you have better signal."
  - Image cached locally for retry later
  - Manual retry button provided
  - Driver's campaign participation marked as "Verification pending"
- Battery usage optimized during extended upload
- No partial/corrupted images stored on server
- Only complete, verified images accepted

Postconditions:

- Image uploaded successfully despite network issues
- System demonstrates resilience to connectivity problems
- User experience managed gracefully with clear feedback

Test Case EC-008: Campaign with Start Date in Past

Priority: Medium
Requirements: FR-11, FR-55

Objective: Verify validation prevents campaigns with invalid dates

Preconditions:

- Advertiser on campaign creation screen
- Current date: October 5, 2025

Test Steps:

1. Attempt to set campaign start date: October 1, 2025 (4 days ago)
2. Set end date: October 10, 2025 (valid future date)
3. Complete other fields
4. Attempt to submit campaign
5. Test various date combinations:
    - Start date = Today, End date = Yesterday
    - Start date = End date (same day)
    - Start date 1 year in past, End date 1 year in future

Expected Results:

- Date picker validation (client-side):
    - Start date picker disables all past dates
    - Only current and future dates selectable
    - If user manually enters past date (typing), validation triggers
- Attempt to set start date October 1, 2025:
    - Real-time error: "Start date cannot be in the past"
    - Field highlighted in red
    - Submit button disabled until corrected
- Server-side validation (backup):
    - If client-side bypassed, server validates on submission
    - Request rejected with 400 Bad Request
    - Error: "Invalid campaign dates. Start date must be today or in the future."
- End date before start date:
    - Error: "End date must be after start date"
    - Date range validation enforces logical order
- Same-day campaign (start = end):
    - Allowed if business rules permit (minimum duration = 1 day)
    - Or rejected if minimum duration > 1 day
    - Clear error message if rejected: "Minimum campaign duration is 2 days"
- Far future dates (1 year ahead):
    - Allowed if within business rules (max advance booking = 90 days)
    - Or rejected with: "Cannot schedule campaigns more than 90 days in advance"
- Edge case: Campaign starting at current time:
    - Start date = Today, start time = Current time +5 minutes

    - Allowed (immediate campaign)
    - Admin activation required before campaign begins
- Validation applied per FR-55 before campaign creation
- Helpful error messages guide user to correct input
- Date format handling: System accepts various formats and converts consistently

Postconditions:

- Only valid date ranges accepted
- Temporal integrity of campaigns maintained
- User guided to correct inputs

---

7. INTEGRATION TESTING

Test Case INT-001: End-to-End Campaign Workflow

Priority: Critical
Requirements: Multiple (FR-11 through FR-81)
Objective: Verify complete campaign lifecycle from creation to completion

Preconditions:

- All system components operational (mobile apps, backend, database, external services)
- Advertiser account with PKR 10,000 wallet balance
- 5 available drivers in target area
- Admin account ready for approvals

Test Steps:

1. **Advertiser: Create Campaign**
    - Login to advertiser app
    - Navigate to "Create Campaign"
    - Enter title: "Integration Test Campaign"
    - Upload 3 advertisement images
    - Set location: Gulberg, Lahore
    - Set duration: Start tomorrow 9 AM, End in 3 days at 5 PM
    - Select 5 cars
    - Review cost: PKR 7,500
    - Apply coupon: "TEST10" (10% discount)
    - Final cost: PKR 6,750
    - Submit campaign

2. **System: Process Campaign Creation**

    - Validate all inputs
    - Deduct amount from wallet
    - Create campaign record in database
    - Send notification to admin

3. **Admin: Approve Campaign**

    - Login to admin portal
    - Review pending campaign
    - Verify advertiser details and campaign content
    - Activate campaign
    - System matches 5 suitable drivers

4. **System: Notify Drivers**

    - Send FCM notifications to selected 5 drivers
    - Display campaign details in driver app

5. **Drivers: Accept Campaign**

    - All 5 drivers review and accept campaign
    - System records acceptances
    - Campaign status updates to "Drivers Assigned"

6. **Drivers: Navigate to Installation Center**

    - Drivers receive navigation instructions
    - Google Maps integration provides directions
    - Drivers arrive at installation center

7. **Admin: Verify Installation**

    - Admin at installation center verifies installations
    - Starts driver campaigns in system
    - Campaign status updates to "Active"

8. **Drivers: Execute Campaign**

    - Drivers travel throughout target area over 3 days
    - GPS tracking records positions every 30 seconds
    - Drivers submit verification images (4 images per driver over 3 days)

9. **Advertiser: Monitor Campaign**

    - Track drivers in real-time on map

- View campaign statistics
- Check verification images

10. **System: Campaign Completion**

    - Campaign end time reached (3 days, 5 PM)
    - System processes completion automatically
    - Validates all verification records
    - Calculates driver earnings
    - Generates performance report

11. **Drivers: Receive Payment**

    - Earnings credited to driver wallets
    - Notifications sent to drivers

12. **Advertiser: Review Report**

    - Access completed campaign
    - View comprehensive report with heat map
    - Export report as PDF

Expected Results:

- **Campaign Creation:**
    - Form submission successful
    - Wallet debited PKR 6,750 (after discount)
    - Coupon marked as used
    - Campaign ID generated
    - Transaction recorded
    - Response time < 2 seconds (NFR-4)
- **Admin Approval:**
    - Campaign appears in admin pending queue
    - Activation triggers driver matching algorithm
    - 5 drivers selected based on proximity, rating, availability
    - Campaign status: "Drivers Assigned"
- **Driver Notifications:**
    - FCM notifications delivered within 5 seconds (NFR-6)
    - All 5 drivers receive notifications simultaneously
    - Notification content includes: campaign details, payment, duration
- **Driver Acceptance:**

- Each acceptance recorded with timestamp
- Campaign-driver associations created in database
- When all 5 accept, campaign fully assigned

- **Navigation Integration:**
  - Google Maps API provides accurate directions (SW-3)
  - Drivers reach installation center successfully

- **Installation Verification:**
  - Admin can start campaigns from portal
  - GPS tracking initiates for all drivers
  - Real-time location data begins flowing

- **Active Campaign:**
  - 3 days of continuous tracking (259,200 seconds)
  - Approximately 8,640 location updates per driver (every 30 seconds per NFR-3)
  - Total: 43,200 location records across 5 drivers
  - All updates transmitted via HTTPS (COM-1)
  - Data stored in database efficiently

- **Verification Images:**
  - 20 total images uploaded (4 per driver × 5 drivers)
  - Each image includes GPS metadata (FR-79, FR-80)
  - Location validation confirms drivers in target area
  - Images stored in cloud with database references
  - All verifications logged (FR-81)

- **Real-time Monitoring:**
  - Advertiser map updates every 30 seconds
  - All 5 drivers visible with distinct markers
  - Route trails displayed
  - Statistics accurate (distance, time)

- **Campaign Completion:**
  - Automatic completion at end time (3 days, 5 PM)
  - Status changed to "Completed"
  - Driver verification validated: 20/20 images verified
  - Earnings calculated: Each driver earned PKR 1,350 (PKR 450/day × 3 days)
  - Platform commission deducted (10%): Each driver nets PKR 1,215
  - Total payout: PKR 6,075 (5 drivers × PKR 1,215)

- Platform revenue: PKR 675 commission

- **Payment Processing:**
  - Driver wallets credited within 1 hour of completion
  - Transaction records created for each driver
  - Email/SMS notifications sent

- **Performance Report:**
  - Report generated with all metrics:
    - Total distance: ~500 km (combined all drivers)
    - Total display time: 15 days (3 days × 5 drivers)
    - Coverage area: Heat map shows concentration
    - Verification: 20 images with timestamps/locations
    - ROI estimation: Impressions based on traffic data
  - PDF export successful, professionally formatted
  - Report data accurate against database

- **Data Integrity:**
  - All database transactions committed correctly
  - No orphaned records
  - Referential integrity maintained
  - Audit logs complete (NFR-16)

- **System Performance:**
  - No system crashes or errors
  - Response times within specifications (NFR-1, NFR-4)
  - 99.5% uptime maintained (NFR-18)
  - Concurrent user support demonstrated (NFR-2)

- **External Integrations:**
  - AWS hosting stable
  - Firebase notifications reliable
  - Google Maps accurate
  - JAZZ CASH payment processing successful (campaign creation)
  - All API calls authenticated with JWT
  - All communications encrypted (NFR-12)

Postconditions:

- Complete campaign lifecycle validated
- All system components verified working together

- Data consistency across all modules
- All stakeholders satisfied with outcomes
- System ready for production deployment

Test Case INT-002: Firebase Cloud Messaging Integration

Priority: High
Requirements: FR-85, COM-3, NFR-6
Objective: Verify FCM notifications delivered correctly across all scenarios

Preconditions:

- Mobile apps with FCM SDK integrated
- Firebase project configured correctly
- Server has Firebase Admin SDK
- Multiple test devices available (Android 6.0 to latest)

Test Steps:

1. Register device FCM tokens for test users
2. Send various notification types:
    - Campaign request to driver
    - Campaign activated (to advertiser)
    - Payment received (to driver)
    - Verification reminder (to driver)
    - Support response (to both)
    - System announcement (broadcast to all)
3. Test notification delivery scenarios:
    - App in foreground
    - App in background
    - App closed/killed
    - Device screen off
    - Device in low-power mode
4. Test notification interactions:
    - Tap notification to open app
    - Dismiss notification
    - Action buttons (Accept/Reject campaign)
5. Test at scale: Send 100 notifications simultaneously

Expected Results:

- Device token registration:

    - FCM token generated on app first launch
    - Token sent to backend and stored in database
    - Token associated with user account
    - Token refreshed automatically if expired

- Notification delivery:
    - All notifications delivered within 5 seconds (NFR-6)
    - Delivery confirmation via Firebase Cloud Messaging API
    - Retry mechanism for failed deliveries (up to 3 attempts)

- App foreground:
    - Notification data received by app immediately
    - In-app alert or banner displayed
    - Notification sound plays
    - App can handle notification data programmatically

- App background:
    - System notification appears in notification tray
    - Notification title, body, icon displayed correctly
    - Badge count increments on app icon
    - Notification persists until dismissed or tapped

- App closed/killed:
    - Notification still delivered by system
    - Tapping notification launches app to relevant screen
    - Deep linking works (e.g., campaign request opens campaign details)

- Device screen off / Low-power mode:
    - Notification wakes device briefly (screen lights up)
    - Notification sound/vibration works
    - Notification visible when screen turned on
    - No significant battery drain

- Notification content:
    - Campaign request notification includes:
        - Title: "New Campaign Request"
        - Body: "[Advertiser Name] - [Campaign Title]"
        - Action buttons: "Accept" and "Reject"
        - Campaign ID in data payload for deep linking
    - Payment notification includes:

        - Title: "Payment Received"
        - Body: "You earned PKR [Amount] for [Campaign Name]"
        - Deep link to wallet screen
    - Custom icons and colors per notification type

- Notification interactions:
    - Tap notification: App opens to relevant screen (campaign details, wallet, support ticket)
    - "Accept" button: Campaign acceptance processed without opening app
    - "Reject" button: Campaign rejected, another driver notified
    - Dismiss (swipe away): Notification cleared, no action taken
    - Notification history maintained (can view old notifications)

- Scalability test:
    - 100 notifications sent simultaneously
    - All delivered within 10 seconds
    - Firebase Cloud Messaging handles batching
    - No duplicate notifications
    - No missed notifications
    - Server doesn't crash under batch load

- Error handling:
    - Invalid token: Server detects, removes from database
    - User uninstalls app: Token invalidated, server handles gracefully
    - Firebase service outage: Notifications queued for retry
    - Network unavailable on device: Notification delivered when network restored

- Security:
    - FCM tokens treated as sensitive, not exposed
    - Notification content doesn't include sensitive data (passwords, full credit card numbers)
    - Authentication required to send notifications (server-side)

- Analytics:
    - Notification delivery tracked (sent, delivered, opened)
    - Open rates calculated
    - Failed deliveries logged for investigation

Postconditions:

- FCM integration fully functional and reliable
- Users receive timely notifications for all important events
- Notification system scales to production requirements

Test Case INT-003: Google Maps API Integration

Priority: High
Requirements: SW-3, FR-27, FR-73, FR-74
Objective: Verify Google Maps integration for navigation and tracking

Preconditions:

- Google Maps API key configured
- API key has necessary permissions (Maps SDK, Directions API, Places API)
- Adequate API quota available
- Test devices with internet connectivity

Test Steps:

1. **Map Display:**
    - Open advertiser app campaign tracking screen
    - Load map with campaign area
2. **Driver Location Markers:**
    - Display 5 driver locations on map
    - Test marker clustering at zoomed-out view
    - Test marker details on tap
3. **Navigation:**
    - Driver accepts campaign
    - Driver taps "Navigate to Installation Center"
    - Google Maps directions launched
4. **Real-time Updates:**
    - Monitor map as drivers move
    - Verify marker positions update
5. **Geofencing:**
    - Define campaign target area boundary
    - Monitor driver entries/exits
6. **Heat Map Overlay:**
    - Generate heat map on campaign completion
    - Display coverage intensity

7. **API Error Handling:**
    - Simulate API quota exceeded
    - Simulate invalid API key
    - Simulate network timeout

Expected Results:

- **Map Loading:**
    - Map loads within 2 seconds
    - Correct geographic area displayed (Lahore, Pakistan)
    - Map controls functional (zoom, pan, tilt, rotate)
    - Map style appropriate (roadmap view default)
    - Responsive to touch gestures
- **Driver Markers:**
    - Each driver represented by custom marker icon (car icon)
    - Markers positioned accurately at GPS coordinates
    - Marker clustering activates when zoomed out (groups nearby markers)
    - Cluster expands when tapped
    - Individual marker tap shows info window:
        - Driver name
        - Car type
        - Campaign duration active
        - "View Details" button
- **Marker Updates:**
    - Position updates every 30 seconds (synced with location tracking NFR-3)
    - Smooth animation between positions (not teleporting)
    - Marker rotation indicates direction of travel
    - Trail/polyline shows path traveled
    - Old trail fades after time period (last 1 hour visible)
- **Navigation Integration:**
    - "Navigate" button launches Google Maps app (if installed)
    - Or opens Google Maps in browser
    - Destination set correctly (installation center address)
    - Starting point set to driver's current location
    - Directions displayed with:
        - Estimated time of arrival
        - Distance

        - Step-by-step instructions
        - Real-time traffic conditions
        - Alternative routes
    - Driver can use voice navigation
    - Navigation works seamlessly, driver can return to CORADS app

- **Geofencing:**

    - Campaign area boundary drawn as circle or polygon overlay
    - Semi-transparent fill color
    - Boundary color contrasts with map
    - Geofence calculations accurate (point-in-polygon algorithm)
    - Entry/exit events detected within 30 seconds
    - No false positives from GPS jitter (debouncing applied)

- **Heat Map:**

    - Heat map overlay rendered using Maps API or custom overlay
    - Color gradient: Red (high) → Orange → Yellow → Green (low)
    - Intensity based on location data density
    - Adjustable opacity slider for overlay
    - Zoom-independent (scales appropriately)
    - Performant rendering (no lag with thousands of data points)

- **API Communication:**

    - All API calls use HTTPS
    - API key included in request headers
    - Rate limiting respected (burst limits, daily limits)
    - Caching applied for static data (map tiles cached locally)
    - Efficient API usage (minimize redundant calls)

- **Error Handling:**

    - Quota exceeded:
        - Map displays cached version
        - Warning: "Map service temporarily unavailable. Showing cached data."
        - Graceful degradation (basic functionality maintained)
    - Invalid API key:
        - Map fails to load
        - Error: "Map service configuration error. Please contact support."
        - Fallback: Text-based location display (coordinates, address)
    - Network timeout:
        - Loading indicator shows for 10 seconds

        - Retry button appears
        - Error: "Unable to load map. Please check your connection."
    - GPS signal unavailable:
        - Last known position displayed with indicator
        - Message: "Waiting for GPS signal..."

- **Performance:**
    - Map rendering doesn't block UI thread
    - Smooth 60 FPS scrolling and zooming
    - Memory usage reasonable (map deallocated when not visible)
    - Battery usage optimized (GPS accuracy balanced)

- **Accessibility:**
    - Map zoom accessible via + / - buttons (not just pinch)
    - Current location button centers map on user
    - Address search functionality works (Places API)
    - Map supports dark mode (if app theme supports it)

Postconditions:

- Google Maps integration provides excellent user experience
- Navigation and tracking features fully functional
- API usage optimized and within quotas

---

8. PERFORMANCE AND LOAD TESTING

Test Case PERF-001: System Performance Under Peak Load

Priority: Critical
Requirements: NFR-1, NFR-2, NFR-4, NFR-5
Objective: Verify system meets performance requirements under maximum expected load

Preconditions:

- Load testing tools configured (JMeter, Gatling, or similar)
- Test environment mirrors production infrastructure
- Database populated with realistic data (10,000 users, 1,000 active campaigns)
- Monitoring tools active (CPU, memory, network, response times)

Test Steps:

1. **Baseline Performance:**
    - Measure response times with 10 concurrent users
    - Establish baseline metrics
2. **Gradual Load Increase:**
    - Increase concurrent users: 50, 100, 250, 500, 750, 1000
    - Maintain each level for 10 minutes
    - Monitor system behavior at each level
3. **Peak Load Test:**
    - Simulate 1000 concurrent users (per NFR-2)
    - Users perform various operations:
        - 30% browsing campaigns
        - 25% creating campaigns (up to 100/minute per NFR-5)
        - 20% tracking active campaigns
        - 15% managing wallets
        - 10% viewing reports
    - Sustain for 30 minutes
4. **Stress Test:**
    - Increase beyond capacity: 1500, 2000 users
    - Identify breaking point
    - Verify graceful degradation
5. **Mobile App Load Time:**
    - Test on minimum spec device (Android 6.0)
    - Measure app load time (NFR-1)
6. **API Response Time:**
    - Measure response times under normal load
    - Verify < 2 seconds per NFR-4

Expected Results:

- **Baseline (10 users):**
    - Average API response: <500ms
    - Page load: <1 second
    - Database queries: <100ms
    - Zero errors

- **Scaled Load (100 users):**
  - Average API response: <1 second
  - Page load: <2 seconds
  - CPU usage: 30-40%
  - Memory usage: 40-50%
  - No errors or timeouts
- **High Load (500 users):**
  - Average API response: <1.5 seconds
  - 95th percentile: <3 seconds
  - CPU usage: 60-70%
  - Memory usage: 60-70%
  - Error rate: <0.1%
  - Database connections: 80% of pool
- **Peak Load (1000 concurrent users per NFR-2):**
  - Average API response: <2 seconds (NFR-4 compliance)
  - 95th percentile: <4 seconds
  - 99th percentile: <6 seconds
  - Throughput: 1000+ requests/second
  - Campaign creation: 100/minute sustained (NFR-5 compliance)
  - Error rate: <1%
  - CPU usage: 75-85%
  - Memory usage: 75-85%
  - Database query time: <200ms average
  - Network bandwidth: 70% utilization
  - Load balancer distributes traffic evenly
  - No server crashes
  - All core functionality remains available
- **Location Tracking at Scale:**
  - 1000 drivers with active tracking
  - 1000 updates every 30 seconds = 33 updates/second
  - All updates processed within 2 seconds
  - Database write performance: <50ms per insert
  - Real-time map updates delivered to advertisers without lag
- **Stress Test (Beyond Capacity):**
  - 1500 users: Response times degrade to 3-5 seconds
  - 2000 users: Response times 5-8 seconds

    - Error rate increases to 5-10%
    - System remains stable (no crashes)
    - Graceful degradation:
        - Static assets served from CDN (unaffected)
        - Critical APIs prioritized
        - Non-essential features may timeout
        - Error messages clear: "System experiencing high load. Please try again."
    - System recovers when load reduced

- **Mobile App Load Time (NFR-1):**
    - Cold start on Android 6.0 device: <3 seconds
    - Warm start: <1 second
    - Screen transitions: <500ms
    - Splash screen displays during load
    - Progressive loading (UI before all data loaded)

- **Database Performance:**
    - Connection pooling effective (100 connections max)
    - Query optimization evident (indexed fields)
    - No long-running queries (>2 seconds)
    - Read replicas utilized for reporting queries
    - Caching reduces database hits (Redis/Memcached)

- **CDN Performance:**
    - Static assets (images, JS, CSS) served from CDN
    - 90% cache hit rate
    - Average CDN response: <100ms
    - Reduces server load significantly

- **Auto-scaling (if implemented):**
    - Additional server instances spawn at 70% CPU
    - Load distributed to new instances within 2 minutes
    - System capacity increases elastically
    - Instances terminate when load decreases

Postconditions:

- System demonstrates ability to handle peak production load
- Performance requirements (NFR-1, NFR-2, NFR-4, NFR-5) validated
- Bottlenecks identified and documented for optimization
- System resilience confirmed

9. USABILITY AND USER EXPERIENCE TESTING

Test Case UX-001: First-Time User Onboarding Experience

Priority: High
Requirements: NFR-17
Objective: Verify new users can complete basic tasks without assistance

Preconditions:

- Test participants: 10 users unfamiliar with CORADS
- 5 potential advertisers (small business owners)
- 5 potential drivers (car owners)
- Recording equipment for observation
- Task completion checklist prepared

Test Steps:

1. **Advertiser Onboarding:**
    - Install app (no prior instructions given)
    - Register account
    - Verify email
    - Add funds to wallet
    - Create first campaign
    - Track campaign (simulated)
2. **Driver Onboarding