User Acceptance Testing (UAT) Plan - Betwise

1. Document Overview (at least 3 features to be tested)

- Project: Betwise
- Team: Nathan Megersa, Adam Wagner, Linley Denslow, Jackie Auerbach, Connor Edelheit
- Features Covered:
  1. Authentication: Register and Login with clear error feedback
  2. Wallet: Add credits and Update wallet balance after games and update database
  3. Mines Game Flow: Place bet, play game, display outcome, update credit balance, update displays

2. Test Environment (localhost / cloud)

- Environment Type: Localhost
- App URL: http://localhost:3000
- API URL (if any): N/A
- Database: betwise (engine: Postgres)

3. Test Data (description)

Provide the exact data you'll use so runs are repeatable.

- Accounts/Credentials:
  - Valid user: John / Doe / jode123@gmail.com / test_user / Password$123
  - Invalid user (wrong password): test_user / johndoe
  - Invalid user (unknown user): johndoe / johndoe123
  - Existing user (for duplicates): test_user
- Per-Feature Inputs:
  - Registration fields: First Name, Last Name, email, Username, Password
  - Login fields: Username, Password
  - Wallet fields: Credit balance
  - Game bet: Bet Amount
- Relevant Tables:
  - betwise.users (expected columns: [user_id, fname, lname, email, username, password_hash, balance, wins, created_at])
  - betwise.transactions (expected columns : [id, user_id, type, amount, description, created_at])

4. User Acceptance Testers (who is testing)

- UAT Lead: Nathan Megersa — coordinates runs & collects results
- Observer/Recorder: Adam Wagner — screenshots, logs, DB checks
- Tester 1 (end user): Jackie Auerbach
- Tester 2 (end user): Linley Denslow
- Tester 3 (end user): Connor Edelheit

5. Test Results Format (how results will be captured)

For each test case, record:

- Run ID / Tester / Date
- Environment (localhost/cloud)
- Inputs Used (actual values)
- Observed Result (UI + DB)
- Status: Pass / Fail
- Artifacts: [Screenshot link(s)], [Query output], [Log snippet]
- Notes/Defects: [Short description + severity]

6. Feature Test Plans (repeat for each feature)

Feature 1: Authentication - Register and Login

User Story:
"As a new user, I want to register and login so I can play/use the application"

Acceptance Criteria:

- Cannot submit registration without username and password → shows a clear validation error (e.g., 'All fields required.')
- Registration with an already-taken username → shows "Username already taken."
- Successful registration inserts a new row in users and returns the Login page with a success message (e.g., "Account created. Please log in.").
- Invalid login shows "Invalid username or password." (generic)

- Successful login creates a session and redirects to /transition (page renders), after which the user can access /home.

Test Case [TC-1.A] — Registration: Happy Path (new user)

- **Preconditions:**
  - User is logged out
  - Users table has no row with username='test_user'
  - App running on localhost, DB = betwise
- **Test Data:**
  - First Name = John
  - Last Name = Doe
  - Email = jode123@gmail.com
  - Username = test_user
  - Password = Password$123
- **User Steps:**
  - Go to /register
  - Enter First name: John, Last name: Doe, email: jode123@gmail.com, username: test_user, password: Password$123
  - Click enter (keyboard)
- **Expected Results:**
  - UI: Successful registration inserts a new row and returns the Login page.
  - DB: A new row is inserted with that username
    - SELECT user_id, username FROM users WHERE username = 'test_user';
- **Actual Results** (Week 4): [to be filled]
- **Status: [**Pass]
- **Artifacts**: [screenshot/log/query link]
- **Notes/Defects:** [if any]

Test Case [TC-1.B] — Registration: Duplicate Username

- **Preconditions:**
  - Row already exists with username = 'test_user'
  - Logged out
- **Test Data:**
  - First Name = John
  - Last Name = Doe
  - Email = jode123@gmail.com
  - Username = test_user
  - Password = Password$123

- **User Steps:**
  - Go to /register
  - Enter First name: John, Last name: Doe, email: jode123@gmail.com, username: test_user, password: Password$123
  - Click enter (keyboard)
- **Expected Results:**
  - UI: Register page re-renders with error banner: “Username already taken” (HTTP 200 Render)
  - DB: No additional row with that username, count stays at 1
    - SELECT COUNT(*) FROM users WHERE username = ‘test_user’;
- **Actual Results** (Week 4): [to be filled]
- **Status:** [Pass/Fail]
- **Artifacts:** [screenshot/log/query link]
- **Notes/Defects:** [if any]

Test Case [TC-1.C] — Login: Happy Path (session + transition)

- **Preconditions**:
  - Users contains row with username = ‘test_user’ with a known password hash for password = ‘Password$123’
- **Test Data:**
  - Username = test_user
  - Password = Password$123
- **User Steps:**
  - Go to /login
  - Enter username: test_user, password: Password$123
  - Click enter (keyboard)
- **Expected Results:**
  - UI: Server sets session and redirects to /transition
    - /transition now renders
    - Navigating to /home now works
  - DB:
    - SELECT user_id, username FROM users WHERE username = ‘test_user’;
- **Actual Results** (Week 4): [to be filled]
- **Status:** [Pass/Fail]
- **Artifacts:** [screenshot/log/query link]
- **Notes/Defects:** [if any]

Feature 2: Wallet - Add credits and Balance Consistency

User Story:
“As a player, I want to be able to add practice credits to my wallet so I can place bets in games”

Acceptance Criteria:

- Top-up with a positive amount increases wallet balance.
- A transaction record is created for each top-up.
- Reject zero/negative or non-numeric amounts with a clear error (no DB write).
- Balance is persistent across page reloads and visible in the UI (e.g., header).

Test Case [TC-2.A] — Top-Up Happy Path (+credits)

- **Preconditions:**
    - Logged in as ‘test_user’
    - Users row exists with balance = 1000 for ‘test_user’
- **Test Data:**
    - Amount = 100
- **User Steps:**
    - Navigate to /wallet, note current balance (1000).
    - Enter amount = 100 in “Add credits” form and click “Deposit”
    - Follow redirect back to /wallet.
- **Expected Results:**
    - UI: Wallet page re-renders, balance now 1100, recent transaction row visible.
    - DB: Balance updated for user in users table and new transaction row added for deposit
- **Actual Results** (Week 4): [Balance: 1100]
- **Status:** [Pass]
- **Artifacts:** [screenshot/log/query link]
- **Notes/Defects:** [if any]

Test Case [TC-2.B] — Reject invalid amount (negative/zero)

- **Preconditions:**
    - Logged in
    - Balance known
- **Test Data:**
    - Amount = 0, -50
- **User Steps:**
    - Submit top-up with 0
    - Look at UI response

    - Check balance
- **Expected Results:**
    - UI: Should display error messages like, “Invalid input.”
    - DB: To be implemented in Week 4
- **Actual Results** (Week 4): [to be filled]
- **Status**: [Pass/Fail]
- **Artifacts:** [screenshot/log/query link]
- **Notes/Defects:** [if any]

Test Case [TC-2.C] — Balance Persists Across Reload Cycle/Login Cycle

- **Preconditions:**
    - Logged in; balance is 1100 from TC-2.A
- **Test Data:**
    - none
- **User Steps:**
    - Refresh /home a few times
    - Logout
    - Log in as ‘test_user’ and open /home
- **Expected Results**:
    - UI: Balance is still 1100 after reload and re-login
    - DB: To be implemented in Week 4
- **Actual Results** (Week 4): [to be filled]
- **Status:** [Pass/Fail]
- **Artifacts**: [screenshot/log/query link]
- **Notes/Defects**: [if any]

<u>Feature 3</u>: Mines — Place Bet, Round Result, Balance Update

User Story:
“As a player, I want to place a bet in Mines, play a round, and see my balance change correctly based on the outcome.”

Acceptance Criteria:

- Can start a round only with a valid bet (positive, ≤ balance).
- On cash-out/win, wallet balance increases by bet×multiplier; on mine/lose, bet is deducted.
- Each round creates at least one transactions row representing the bet and/or payout
- Errors for invalid bet (empty/negative/over balance) with no DB writes.

Test Case [TC-3.A] — Start Round with Valid Bet, Cash-Out Win

- **Preconditions:**
  - Logged in as ‘test_user’
  - Wallet balance ≥ 100
- **Test Data:**
  - bet = 50
- **User Steps:**
  - On /mines, enter bet 50 → click Start
  - Click a safe tile or use “Cash Out” when allowed.
- **Expected Results:**
  - UI: Round starts; on cash-out, show payout and new balance.
  - DB: To be implemented in Week 4
- **Actual Results:** [ ]
- **Status:** [Pass/Fail]
- **Artifacts:** [screens of start & win states], [queries]
- **Notes:** [ ]

Test Case [TC-3.B] — Start Round with Valid Bet, Hit Mine (Lose)

- **Preconditions**: Balance ≥ 50.
- **Test Data:**
  - bet = 50
- **User Steps:**
  - Start with bet 50.
  - Select a mined tile → round ends as loss.
- **Expected Results:**
  - UI: Loss message; balance decreased by 50.
  - DB: To be implemented in Week 4
- **Actual Results:** [ ]
- **Status:** [Pass/Fail]
- **Artifacts:** [screens], [queries]
- **Notes:** [ ]

Test Case [TC-3.C] — Reject Invalid Bet (Over Balance / Negative / Empty)

- **Preconditions:** Balance = 100 for deterministic check.
- **Test Data:**
  - bet = 1000 *(> balance)*, then bet = -10, then blank
- **User Steps:**
  - Attempt to start with 1000.

    - Attempt with -10.
    - Attempt with a blank bet.
- **Expected Results:**
    - UI: Validation/error message (e.g., “Bet must be positive and ≤ balance.”).
    - DB: To be implemented in Week 4
- **Actual Results:** [ ]
- **Status:** [Pass/Fail]
- **Artifacts:** [screens], [queries]
- **Notes:** [ ]

7. Actual Test Results (after execution)

- Run Log Link or Table: [paste or link]
- Pass Rate: [X/Y cases passed]
- Defects Found: [count by severity]
- Key Observations: [usability, error clarity, performance notes]
- Follow-ups/Fixes: [brief list + owner]