AI TRUST LEDGER

1. Send Verification Code for Registration

Description: This feature generates a random 6-digit verification code and sends it to the user's email during the signup process to verify their email address.

Details:

- **Input:** Email address
- **Process:**
  - Checks if the email is already registered in the users table.
  - If not registered, generates a random 6-digit code (e.g., 123456).
  - Stores the code and email in session variables.
  - Sends an email using PHPMailer with Office 365 SMTP settings.
- **Email Content:** HTML email with the verification code, valid for 10 minutes.
- **Output:** Success message if email is sent, error message if email is already registered or sending fails.

2. Complete Registration

Description: This feature completes the user registration after verifying the code sent to their email, saving their details in the database.

Details:

- **Input:** Full name, email (from session), verification code, password, confirm password, optional referral code
- **Process:**
  - Verifies the input code matches the session-stored verification code.
  - Checks if passwords match.
  - Generates a unique 8-character referral code for the new user.
  - If a referral code is provided, validates it and updates the referrer’s referral count and level.
  - Inserts user data into the users table with initial level "Bronze Star".

    - Referral system updates levels based on referral count (e.g., 2 referrals = Silver Champion).
- **Output:** Redirects to index.php on success, error message on failure (e.g., invalid code, password mismatch).

3. User Login

Description: This feature authenticates a user by checking their email/username and password, allowing access if their email is verified.

Details:

- **Input:** Username (email or full name), password
- **Process:**
    - Queries the users table for a matching email or full name (case-insensitive).
    - Verifies the input password matches the stored password.
    - Checks if the email is verified (email_verified = 'yes').
    - Sets session variables (user_id, email, fullname) on successful login.
- **Output:** Redirects to index.php on success, error message if credentials are incorrect or email isn't verified.

4. Send Password Reset Code

Description: This feature sends a verification code to a user's email when they request a password reset.

Details:

- **Input:** Email address
- **Process:**
    - Checks if the email exists in the users table.
    - Generates a random 6-digit code and stores it in the session with the email.
    - Sends an email using PHPMailer with Office 365 SMTP settings.
- **Email Content:** HTML email with the reset code, valid for 10 minutes.
- **Output:** JSON response with success message if sent, error if email not found or sending fails.

5. Complete Password Reset

Description: This feature updates the user's password after verifying the reset code sent to their email.

Details:

- **Input:** Email (from session), verification code, new password
- **Process:**
  - Verifies the input code matches the session-stored reset code.
  - Updates the password field in the users table with the new password.
  - Clears reset-related session variables.
- **Output:** JSON response with success message on update, error if code is invalid or update fails.

6. Referral System

Description: This feature manages a referral hierarchy, rewarding users with levels based on their referral count during signup.

Details:

- **Input:** Referral code (optional during signup)
- **Process:**
  - Validates the provided referral code against existing users.
  - If valid, increments the referrer's referral_count.
  - Updates the referrer's level based on referral count (e.g., 2 referrals = Silver Champion, 4 = Gold Achiever).
  - Level mapping:
    - 1: Bronze Star
    - 2: Silver Champion
    - 3: Gold Achiever
    - 4: Platinum Leader
    - 5: Diamond Elite
    - Up to 15: Supreme Emperor
  - New level calculated as floor(referral_count / 2) + 1.
- **Output:** Updates referrer's data in the database, no direct user output.

INDEX FILE

1. User Authentication Check

Description: Ensures that only logged-in users can access the dashboard by checking for an active session.

Details:

- **Input:** None (uses session data)
- **Process:**
  - Checks if $_SESSION['email'] is set.
  - If not set, redirects the user to login.php.
- **Output:** Access to dashboard if authenticated, redirection to login page if not.

2. Fetch User Profile Data

Description: Retrieves the logged-in user's details from the database to display on the dashboard.

Details:

- **Input:** User's email from session ($_SESSION['email'])
- **Process:**
  - Queries the users table for user data (id, fullname, level, referral_code, etc.).
  - Sets default values if no data is found (e.g., "User" for fullname, "Bronze Star" for level).
  - Extracts initials from the full name for display.
- **Output:** User-specific variables (e.g., $fullname, $user_level, $referral_code) for use in the dashboard.

3. Team Count Calculation (15 Levels)

Description: Calculates the total active and inactive team members across 15 levels of referrals.

Details:

- **Input:** User's referral code
- **Process:**
  - Uses a queue-based approach to traverse the referral tree up to 15 levels.
  - Checks each referred user's deposit_amount to determine if they are active (> 0) or inactive (0).
  - Avoids infinite loops by tracking processed referral codes.
- **Output:** Array with active and inactive team counts (e.g., $total_active_team, $total_inactive_team).

4. Total Withdrawal Calculation

Description: Computes the total amount withdrawn by the user from completed transactions.

Details:

- **Input:** User's email
- **Process:**
  - Queries the transactions table for withdrawal transactions with status "completed".
  - Sums the amount column for matching records.
- **Output:** Total withdrawal amount (e.g., $total_withdrawal).

5. Daily Reward Status

Description: Determines the availability and status of the user's daily reward.

Details:

- **Input:** last_reward_date and daily_reward from user data

- **Process:**
  - Compares last_reward_date with the current date.
  - Sets status as "claimed" if claimed today, "unclaimed" if not.
  - Disables the claim button if already claimed or reward is zero.
- **Output:** Reward status ($reward_status) and button state ($reward_disabled).

6. Levels Information Display

Description: Retrieves and displays the user's progress through the level system.

Details:

- **Input:** None (queries levels table)
- **Process:**
  - Fetches all levels from the levels table (name, required active members, earning percentage).
  - Determines if each level is unlocked based on user's active referral count or default settings.
  - Marks the current user level.
- **Output:** Array of levels with status (e.g., $levels) for display in the dashboard.

7. Total Business Calculation

Description: Calculates the total deposit amount from all referrals (direct and indirect).

Details:

- **Input:** User's referral code
- **Process:**
  - Uses a queue-based traversal to sum deposit_amount from all referred users.
  - Processes all levels without a strict limit.
- **Output:** Total business amount (e.g., $total_business).

8. Notification System

Description: Manages and displays notifications with an unread count and modal interface.

Details:

- **Input:** None (queries notifications table)
- **Process:**
  - Fetches unread notification count based on last_notification_viewed session variable.
  - Retrieves all notifications in descending order of creation time.
  - Updates count dynamically via JavaScript and marks notifications as viewed when the modal is opened.
  - Allows clearing all notifications via a button.
- **Output:** Notification count ($unread_count) and list for display in a modal.

9. Email Support Query System

Description: Provides a form for users to send support queries with optional image attachments.

Details:

- **Input:** Name, email, message, optional image file
- **Process:**
  - Displays a floating email icon that opens a modal form.
  - Submits form data (including image) via AJAX to send_query.php.
  - Shows image preview before submission.
- **Output:** Success or error message in the modal, closes on success.

10. Dashboard Display

Description: Renders the main dashboard with user statistics and interactive elements.

Details:

- **Input:** User data variables (e.g., $balance, $total_team)

- **Process:**
  - Displays cards for balance, deposits, withdrawals, team stats, etc.
  - Includes a welcome message with user name styling.
  - Features an image slider with navigation buttons.
  - Shows daily reward section with claim button and timer.
- **Output:** Interactive HTML dashboard interface.

11. Edit Profile Section

Description: Allows users to update their profile information and password.

Details:

- **Input:** Full name, email, current password, new password, confirm password
- **Process:**
  - Form submission to edit-profile.php for updating profile data.
  - Separate section for password change with validation fields.
- **Output:** Form interface for profile editing.

12. Delete Account Option

Description: Provides an option for users to permanently delete their account.

Details:

- **Input:** None (triggered by button)
- **Process:**
  - Form submission to edit-profile.php with delete_account action.
  - Displays a confirmation message before proceeding.
- **Output:** Delete button in the edit profile section.

13. Referral Link Sharing

Description: Allows users to copy their referral link for inviting others.

Details:

- **Input:** User's referral code
- **Process:**
  - Generates a referral link (https://aitrustledger.com/signup.php?ref=CODE).
  - Copies the link to the clipboard via JavaScript when the button is clicked.
- **Output:** Referral code and copy button on the dashboard.

14. Dynamic Level Income Update

Description: Fetches and updates the user's level income dynamically via AJAX.

Details:

- **Input:** None (fetches from user_data.php)
- **Process:**
  - Calculates total level income by summing income from 15 levels.
  - Updates the dashboard display with the latest value.
- **Output:** Updated level income value (e.g., $level_income).

15. Daily Reward Claim with Timer

Description: Manages the claiming of daily rewards with a 24-hour cooldown timer.

Details:

- **Input:** None (triggered by button click)
- **Process:**
  - Submits claim request to claim-reward.php via AJAX.
  - Starts a countdown timer using localStorage to track the 24-hour period.
  - Updates balance and status on successful claim.
- **Output:** Updated balance, reward status, and timer display.

1. Deposit Page Authentication Check (deposit.php)

Description: Ensures only logged-in users can access the deposit page.

Details:

- **Input:** None (uses session data)
- **Process:**
  - Checks if $_SESSION['email'] is set.
  - Redirects to login.php if not authenticated.
- **Output:** Access to deposit page if logged in, redirection to login if not.

2. Display Wallet Address and QR Code (deposit.php)

Description: Shows the static wallet address for deposits and generates a QR code for easy scanning.

Details:

- **Input:** Hardcoded wallet address (0xa9f1F39183a6d96ae103499Db7717f596585a63F)
- **Process:**
  - Displays the wallet address in a readonly input field.
  - Generates a QR code by calling qr-generator.php with the wallet address as a parameter.
  - Provides a "Copy Address" button to copy the address to the clipboard.
- **Output:** Wallet address and QR code image on the deposit page.

3. Deposit Form Input Collection (deposit.php)

Description: Collects deposit amount, transaction hash, and proof image from the user.

Details:

- **Input:** Amount (min $25), Transaction Hash (TxID), Image file (proof)
- **Process:**
  - Validates amount (minimum $25) and ensures TxID and proof are provided.

    - Supports file upload with drag-and-drop functionality and displays the selected file name.
- **Output:** Form data ready for submission to process-deposit.php.

4. Submit Deposit Request (deposit.php)

Description: Submits the deposit request to the server for processing.

Details:

- **Input:** Amount, TxID, wallet address, proof image
- **Process:**
    - Validates inputs client-side (amount ≥ $25, TxID and proof required).
    - Sends data via AJAX to process-deposit.php using FormData.
    - Disables the submit button during processing to prevent multiple submissions.
- **Output:** Success or error message displayed on the page; button text changes to "Submitted" on success.

5. Process Deposit Request (process-deposit.php)

Description: Handles the backend processing of a deposit request and stores it in the database.

Details:

- **Input:** Amount, TxID, wallet address, proof image (from POST and FILES)
- **Process:**
    - Verifies user is logged in via session.
    - Validates amount (≥ $25) and TxID presence.
    - Uploads the proof image to /admin/uploads/ with a unique filename, restricting to image formats.
    - Inserts a pending transaction record into the transactions table.
    - Uses a database transaction to ensure data integrity.
- **Output:** JSON response with success message and transaction details, or error message if validation/upload fails.

6. Withdraw Page Authentication Check (withdraw.php)

Description: Ensures only logged-in users can access the withdrawal page.

Details:

- **Input:** None (uses session data)
- **Process:**
  - Checks if $_SESSION['email'] is set.
  - Redirects to login.php if not authenticated.
- **Output:** Access to withdraw page if logged in, redirection to login if not.

7. Fetch User Balance Data (withdraw.php)

Description: Retrieves the user's balance and related financial data for withdrawal.

Details:

- **Input:** User's email from session
- **Process:**
  - Queries the users table for balance, deposit_amount, deposit_bonus, daily_reward, and level_income.
  - Ensures balance is non-negative using max(0, balance).
- **Output:** Variables (e.g., $balance, $deposit_amount) for display on the withdraw page.

8. Withdraw Type Selection (withdraw.php)

Description: Allows users to choose between principal and profit withdrawal types.

Details:

- **Input:** None (user interaction)
- **Process:**
  - Displays two tabs: "Principal Withdraw" and "Profit Withdraw".
  - Toggles visibility of respective sections using JavaScript (showSection function).

- **Output:** Active section displayed based on user selection (principal or profit).

9. Withdraw Form Input Collection (withdraw.php)

Description: Collects wallet address and amount for withdrawal.

Details:

- **Input:** Wallet address, amount (min $10)
- **Process:**
  - Displays current balance (principal or profit) for reference.
  - Allows copying of wallet address to clipboard.
  - Validates amount (≥ $10) and ensures wallet address is provided.
- **Output:** Form data ready for submission to withdraw_request.php.

10. Submit Withdraw Request (withdraw.php)

Description: Submits the withdrawal request to the server for processing.

Details:

- **Input:** Wallet address, amount, withdraw type (principal/profit)
- **Process:**
  - Validates inputs client-side (amount ≥ $10, sufficient balance, wallet address required).
  - Sends data via AJAX to withdraw_request.php using FormData.
  - Updates displayed balance on success and disables the submit button.
- **Output:** Success or error message; balance updated on success.

11. Process Withdraw Request (withdraw_request.php)

Description: Handles the backend processing of a withdrawal request and updates the database.

Details:

- **Input:** Wallet address, amount, withdraw type (from POST)

- **Process:**
  - Verifies user is logged in via session.
  - Validates amount (≥ $10) and sufficient balance (principal or profit).
  - Applies fee: 10% for principal, 3% for profit.
  - Generates a unique TxID (e.g., WD-randomhex).
  - Updates users table: reduces deposit_amount (principal) or balance (profit), increases withdrawal_amount.
  - Inserts a pending withdrawal record into the transactions table.
  - Uses a database transaction for data integrity.
- **Output:** JSON response with success message, fee, final amount, and TxID, or error message if validation fails.

1. Principal Withdrawal History Authentication Check (withdraw-record.php)

Description: Ensures only logged-in users can view their principal withdrawal history.

Details:

- **Input:** None (uses session data)
- **Process:**
  - Checks if $_SESSION['email'] is set.
  - Redirects to login.php if not authenticated.
- **Output:** Access to withdrawal history page if logged in, redirection to login if not.

2. Fetch Principal Withdrawal Records (withdraw-record.php)

Description: Retrieves and displays the user's principal withdrawal transactions.

Details:

- **Input:** User's email from session
- **Process:**

    - Queries the transactions table for withdrawal records where type = 'withdrawal' and withdraw_type = 'principal'.
    - Orders results by created_at in descending order.
- **Output:** Array of withdrawal records ($withdrawals) for display.

3. Display Principal Withdrawal History (withdraw-record.php)

Description: Renders a card-based interface for principal withdrawal records.

Details:

- **Input:** Withdrawal records array
- **Process:**
    - Displays each withdrawal in a card with details: ID, amount, fee, final amount, wallet address, date, and TxID.
    - Shows status (pending, completed, rejected) with color coding.
    - Displays a “No records” message if the array is empty.
- **Output:** Responsive grid of withdrawal cards or a no-records message.

4. Deposit History Authentication Check (deposit-record.php)

Description: Prevents unauthorized access to the deposit history page.

Details:

- **Input:** None (uses session data)
- **Process:**
    - Checks if $_SESSION['email'] is set.
    - Terminates script with an error message if not authenticated.
- **Output:** Access to deposit history page if logged in, error message if not.

5. Fetch Deposit Records (deposit-record.php)

Description: Retrieves the user’s deposit transactions from the database.

Details:

- **Input:** User's email from session
- **Process:**
  - Queries the transactions table for records where type = 'deposit'.
  - Orders results by created_at in descending order.
- **Output:** Result set with amount, status, and creation date.

6. Display Deposit History (deposit-record.php)

Description: Shows a list of deposit transactions in a simple layout.

Details:

- **Input:** Deposit records result set
- **Process:**
  - Displays each deposit with amount, status, and date in a flexbox layout.
  - Uses color-coded status indicators (completed: green, pending: yellow, rejected: red).
  - Shows a "No history" message if no records exist.
- **Output:** List of deposit entries or a no-records message.

7. Indirect Team Authentication Check (team.php)

Description: Ensures only logged-in users can view their indirect team members.

Details:

- **Input:** None (uses session data)
- **Process:**
  - Checks if $_SESSION['email'] is set.
  - Redirects to login.php if not authenticated.
- **Output:** Access to indirect team page if logged in, redirection to login if not.

8. Fetch Indirect Team Members (team.php)

Description: Retrieves indirect team members (level 2 and beyond) up to 15 levels.

Details:

- **Input:** User's referral code
- **Process:**
  - Excludes direct referrals by collecting their codes first.
  - Uses a queue-based traversal to fetch team members from level 2 to 15.
  - Collects ID, name, deposit, joining date, level, and referral code.
- **Output:** Array of indirect team members ($indirect_team) with team level info.

9. Display Indirect Team (team.php)

Description: Renders a table of indirect team members.

Details:

- **Input:** Indirect team array
- **Process:**
  - Displays a table with columns: ID, Name, Deposit, Joining Date, Level, Team Level, Referral Code.
  - Uses a sticky header and scrollable body for large datasets.
  - Shows a "No members" message if the array is empty.
- **Output:** Responsive table of indirect team members or a no-data message.

10. Direct Team Authentication Check (direct-team.php)

Description: Restricts access to the direct team page to logged-in users.

Details:

- **Input:** None (uses session data)
- **Process:**
  - Checks if $_SESSION['email'] is set.
  - Redirects to login.php if not authenticated.
- **Output:** Access to direct team page if logged in, redirection to login if not.

11. Fetch Direct Team Members (direct-team.php)

Description: Retrieves the user's direct referrals (level 1 team members).

Details:

- **Input:** User's referral code
- **Process:**
  - Queries the users table for records where referred_by matches the user's referral code.
  - Fetches ID, name, deposit, joining date, level, and referral code.
- **Output:** Array of direct team members ($direct_team).

12. Display Direct Team (direct-team.php)

Description: Shows a table of direct team members.

Details:

- **Input:** Direct team array
- **Process:**
  - Displays a table with columns: ID, Name, Deposit, Joining Date, Level, Referral Code.
  - Uses a sticky header and scrollable body for large datasets.
  - Shows a "No members" message if the array is empty.
- **Output:** Responsive table of direct team members or a no-data message.

13. User Data Authentication Check (user_data.php)

Description: Ensures only logged-in users can access their data via this API.

Details:

- **Input:** None (uses session data)
- **Process:**
  - Checks if $_SESSION['email'] is set.
  - Returns a JSON error if not authenticated.

- **Output:** Access to user data if logged in, error response if not.

14. Fetch User Data (user_data.php)

Description: Retrieves comprehensive user data for dashboard updates.

Details:

- **Input:** User's email from session
- **Process:**
  - Queries the users table for various fields (balance, referral count, level income, etc.).
  - Updates last_login timestamp.
  - Calculates daily reward status based on last_reward_date.
- **Output:** User data variables (e.g., $balance, $daily_reward) for further processing.

15. Calculate Level-Wise Income (user_data.php)

Description: Computes income from each referral level based on commission rates.

Details:

- **Input:** User's referral code, commission rates array
- **Process:**
  - Uses recursive traversal up to 15 levels to calculate income.
  - Applies commission rates (e.g., 20% for level 1, 2% for level 15) to active downline daily rewards.
  - Considers active referral count thresholds for eligibility.
- **Output:** Array of level-wise incomes ($level_incomes).

16. Calculate Total Indirect Team Size (user_data.php)

Description: Determines the total number of indirect team members.

Details:

- **Input:** User's referral code

- **Process:**
  - Uses queue-based traversal to count team members from level 2 to 15.
  - Excludes direct referrals by tracking their codes.
- **Output:** Total indirect team size ($total_indirect_team_size).

17. Return User Data Response (user_data.php)

Description: Sends a JSON response with all calculated user data.

Details:

- **Input:** Processed user data and calculations
- **Process:**
  - Compiles data into a response array including balance, team sizes, level incomes, etc.
  - Formats numbers with two decimal places.
- **Output:** JSON response with user statistics and financial data.

ADMIN

1. Admin Dashboard Authentication Check (admin-dashboard.php)

Description: Ensures only logged-in admins can access the dashboard.

Details:

- **Input:** None (uses session data)
- **Process:**
  - Checks if $_SESSION['admin_logged_in'] is set.
  - Redirects to admin-login.php if not authenticated.

- **Output:** Access to dashboard if authenticated, redirection to login if not.

2. Fetch Dashboard Statistics (admin-dashboard.php)

Description: Retrieves key statistics for the admin dashboard.

Details:

- **Input:** None (database queries)
- **Process:**
  - Total Users: Counts all rows in users table.
  - New Users (7 Days): Counts users created in the last 7 days.
  - Total Deposits: Sums deposit_amount from users.
  - Total Withdrawals: Sums withdrawal_amount from users.
  - Total Earnings: Sums daily_reward, deposit_bonus, and level_income.
- **Output:** Variables ($totalUsers, $newUsers, $totalDeposits, etc.) for display.

3. Display Dashboard Cards (admin-dashboard.php)

Description: Shows key statistics in a card-based layout.

Details:

- **Input:** Statistics variables
- **Process:**
  - Displays five cards: Total Users, New Users, Total Deposits, Total Withdrawals, Total Earnings.
  - Uses FontAwesome icons and formatted numbers.
- **Output:** Responsive grid of dashboard cards.

4. Admin Users Authentication Check (admin-users.php)

Description: Restricts access to the users page to authenticated admins.

Details:

- **Input:** None (via admin-header.php)
- **Process:**
  - Inherited from admin-header.php, which checks $_SESSION['admin_logged_in'].
- **Output:** Access to users page if authenticated.

5. Fetch All Users (admin-users.php)

Description: Retrieves all user data for the admin interface.

Details:

- **Input:** None (database query)
- **Process:**
  - Queries users table, ordered by pinned DESC and id ASC.
  - Fetches all columns for each user.
- **Output:** Result set of user records.

6. Display User Cards with Actions (admin-users.php)

Description: Renders a searchable list of users with management options.

Details:

- **Input:** User records result set
- **Process:**
  - Displays each user in a card with details (ID, name, email, balance, etc.).
  - Includes actions: Delete, Block/Unblock, Pin/Unpin.
  - Supports real-time search by ID, name, or email.
  - Highlights pinned users with a distinct style.
- **Output:** Responsive grid of user cards with interactive buttons.

7. Set Daily Reward for All Users (admin-users.php)

Description: Allows admin to update the daily reward for all users.

Details:

- **Input:** Reward amount from form
- **Process:**
  - Validates input (non-negative number).
  - Submits via AJAX to set-reward.php.
  - Updates displayed reward amounts on success.
- **Output:** Success/error message and updated UI.

8. Admin Transactions Fetch (admin-transactions.php)

Description: Retrieves all deposit and withdrawal transactions.

Details:

- **Input:** None (database queries)
- **Process:**
  - Deposits: Queries transactions where type = 'deposit', ordered by created_at DESC.
  - Withdrawals: Queries transactions where type = 'withdrawal', ordered by created_at DESC.
- **Output:** Two result sets for deposits and withdrawals.

9. Display Transactions with Actions (admin-transactions.php)

Description: Shows tables for deposits and withdrawals with approval/rejection options.

Details:

- **Input:** Deposit and withdrawal result sets
- **Process:**
  - Deposits table: Shows ID, email, amount, TxID, screenshot, status, date, and Approve/Reject buttons for pending transactions.
  - Withdrawals table: Shows ID, email, amount, fee, final amount, wallet address, status, date, and Approve/Reject buttons for pending transactions.
  - Includes a Clear History button for each table.
- **Output:** Responsive tables with interactive buttons.

10. Admin Notifications Form (admin-notifications.php)

Description: Allows admin to post new notifications with optional images.

Details:

- **Input:** Message text, optional image file
- **Process:**

    - Handles form submission with text and image upload.
    - Saves image to uploads/ directory with a unique name.
    - Inserts notification into notifications table.
- **Output:** Success/error message and updated notification list.

11. Display Notification History (admin-notifications.php)

Description: Shows a list of all posted notifications.

Details:

- **Input:** Notifications from notifications table
- **Process:**
    - Fetches all notifications, ordered by created_at DESC.
    - Displays each with message, optional image, and timestamp.
- **Output:** Responsive list of notification items.

12. Admin Login Authentication (admin-login.php)

Description: Handles admin login with hardcoded credentials.

Details:

- **Input:** Email and password from form
- **Process:**
  - Checks against fixed credentials (aitrustledgar@gmail.com, 1911785676337890).
  - Sets $_SESSION['admin_logged_in'] on success.
  - Redirects to admin-dashboard.php if already logged in or on successful login.
- **Output:** Login form or redirection to dashboard.

13. Approve Deposit Processing (approve-deposit.php)

Description: Approves a pending deposit and updates user data.

Details:

- **Input:** Transaction ID from POST
- **Process:**
  - Validates transaction (pending deposit).
  - Updates transactions status to completed.
  - Increases user's deposit_amount.
  - Awards 5% referral bonus to referrer if applicable, updating deposit_bonus and balance.
  - Records bonus transaction.
  - Uses database transaction for consistency.

- **Output:** JSON response with success/error message.

14. Approve Withdrawal Processing (approve-withdraw.php)

Description: Approves a pending withdrawal request.

Details:

- **Input:** Transaction ID from POST
- **Process:**
  - Validates transaction (pending withdrawal).
  - Updates transactions status to completed.
  - Calculates fee (10% for principal, 3% for profit) and final amount.
  - Logs process to debug.log.
- **Output:** JSON response with success message, fee, and final amount.

15. Reject Deposit Processing (reject-deposit.php)

Description: Rejects a pending deposit request.

Details:

- **Input:** Transaction ID from POST
- **Process:**
  - Validates transaction (pending deposit).
  - Updates transactions status to failed.

- **Output:** JSON response with success/error message.

16. Reject Withdrawal Processing (reject-withdraw.php)

Description: Rejects a pending withdrawal and refunds the amount.

Details:

- **Input:** Transaction ID from POST
- **Process:**
  - Validates transaction (pending withdrawal).
  - Updates transactions status to failed.
  - Refunds amount to user's balance and reduces withdrawal_amount.
  - Uses database transaction for consistency.
- **Output:** JSON response with success message and refunded amount.

1. Block/Unblock User Processing (blockuser.php)

Description: Allows the admin to block or unblock a user by updating their status in the database.

Details:

- **Input:** User ID and action (block or unblock) from POST

- **Process:**
  - Validates inputs: ensures userId is provided and action is either block or unblock.
  - Sets blocked column to 1 for block or 0 for unblock in the users table.
  - Uses prepared statement to update the user's status.
- **Output:** JSON response with success message (e.g., "User blocked successfully") or error message if the update fails.

2. Pin/Unpin User Processing (pinuser.php)

Description: Enables the admin to pin or unpin a user, affecting their display order in the user list.

Details:

- **Input:** User ID and action (pin or unpin) from POST
- **Process:**
  - Validates inputs: ensures userId and action are provided and valid.
  - Updates pinned column to 1 for pin or 0 for unpin in the users table.
  - Uses prepared statement for secure database update.

- **Output:** JSON response with success message (e.g., "Pin successful") or error message if the request or database operation fails.

3. Set Daily Reward Processing (set-reward.php)

Description: Updates the daily reward for all users based on a percentage of their deposit amount.

Details:

- **Input:** Reward percentage from POST (reward_amount)
- **Process:**
  - Validates input: ensures reward_percentage is non-negative.
  - Checks if the admin has already set a reward today by comparing last_reward_date with the current date.
  - Calculates new daily_reward as deposit_amount * (reward_percentage / 100) for all users.
  - Resets last_reward_date for users and updates admin's last_reward_date to today.
  - Logs actions and errors for debugging.
- **Output:** JSON response with success message ("Daily reward percentage set successfully") or error message if already set today or update fails.

4. Delete User Processing (deleteuser.php)

Description: Permanently removes a user from the database.

Details:

- **Input:** User ID from GET (id)
- **Process:**
  - Validates input: ensures userId is provided.
  - Executes a DELETE query on the users table for the specified ID.
  - Uses prepared statement to prevent SQL injection.
- **Output:** Redirects to users.php on success or displays an error message ("Error deleting user") if the operation fails.

DB STRUCTURE

Table Name: users

Column Name	Data Type	Attributes	Null Allowed	Default Value	Extra
id	int	Primary Key	No	None	AUTO_INCREMENT
fullname	varchar(255)		No	None	None
email	varchar(255)		No	None	None
password	varchar(255)		No	None	None
email_verified	enum('yes', 'no')		Yes	'no'	None

created_at	datetime	Yes	CURRENT_TIMESTAMP	None
verification_code	varchar(255)	Yes	NULL	None
deposit_amount	decimal(10,2)	Yes	'0.00'	None
withdrawal_amount	decimal(10,2)	Yes	'0.00'	None
referral_count	int	Yes	'0'	None
active_referral_count	int	Yes	'0'	None
balance	float(10,2)	No	'0.00'	None
level	varchar(50)	No	'Bronze Star'	None
referral_code	varchar(10)	Yes	NULL	None
referred_by	varchar(10)	Yes	NULL	None
total_earning	decimal(10,2)	Yes	'0.00'	None
daily_reward	float(10,2)	No	'0.00'	None
last_reward_date	datetime	Yes	NULL	None
deposit_bonus	double	Yes	'0'	None
last_login	datetime	Yes	NULL	None
reset_token	varchar(255)	Yes	NULL	None
reset_token_expires	datetime	Yes	NULL	None
level_income	decimal(10,2)	Yes	'0.00'	None
blocked	tinyint(1)	Yes	'0'	None
pinned	tinyint	Yes	'0'	None

Table Name: transactions

Column Name	Data Type	Attributes	Null Allowed	Default Value	Extra
id	int	Primary Key	No	None	AUTO_INCREMENT
user_email	varchar(255)		No	None	None
amount	decimal(10,2)		No	None	None
wallet_address	varchar(255)		No	None	None
txid	varchar(255)		No	None	None
screenshot_path	varchar(255)		Yes	NULL	None
type	varchar(50)		No	'deposit'	None
status	enum('pending', 'completed', 'failed')		No	'pending'	None
created_at	timestamp		Yes	CURRENT_TIMESTAMP	None
bonus	decimal(10,2)		Yes	'0.00'	None
fee	decimal(10,2)		Yes	'0.00'	None
final_amount	decimal(10,2)		Yes	'0.00'	None
withdraw_type	varchar(10)		Yes	NULL	None

Table Name: notifications

Column Name	Data Type	Attributes	Null Allowed	Default Value	Extra

id	int	Primary Key	No	None	AUTO_INCREMENT
message	text	utf8mb4_0900_ai_ci collation	Yes	NULL	None
image_path	varchar(255)	utf8mb4_0900_ai_ci collation	Yes	NULL	None
created_at	timestamp		Yes	CURRENT_TIMESTAMP	DEFAULT_GENERATED

Table Name: levels

Column Name	Data Type	Attributes	Null Allowed	Default Value	Extra
id	int	Primary Key	No	None	AUTO_INCREMENT
level_name	varchar(50)	utf8mb4_0900_ai_ci collation	No	None	None
required_active_members	int		No	'0'	None
earning_percentage	decimal(5,2)		No	'0.00'	None
unlocked_by_default	tinyint(1)		Yes	'0'	None

Table Name: ranks_claim

Column Name	Data Type	Attributes	Null Allowed	Default Value	Extra
id	int	Primary Key	No	None	AUTO_INCREMENT
user_email	varchar(255)		No	None	None
rank_number	int		No	None	None
reward_amount	decimal(10,2)		No	None	None
claimed_at	datetime		No	None	None

Table Name: rewards

Column Name	Data Type	Attributes	Null Allowed	Default Value	Extra
id	int	Primary Key	No	None	AUTO_INCREMENT
email	varchar(255)	utf8mb4_0900_ai_ci collation	Yes	NULL	None
reward_amount	decimal(10,2)		Yes	NULL	None
status	enum('pending', 'approved')	utf8mb4_0900_ai_ci collation	Yes	'approved'	None
created_at	timestamp		Yes	CURRENT_TIMESTAMP	DEFAULT_GE