Admin Wallet & Deposit Management Guide

Overview

This guide explains how to manually manage wallet balances and approve/reject deposits in the admin panel. The functionality is **already implemented** in the backend and accessible through the UI.

Features Available

☑ 1. Manual Wallet Balance Adjustment

Endpoint: `POST /api/admin/finance/wallet/:id/balance`

How it works:

- Admins can manually add or subtract funds from any user wallet
- Creates a transaction record automatically
- Sends email notification to the user
- Logs the action with "ADMIN" method in metadata

Request Body:

```
{
  "type": "ADD" | "SUBTRACT",
  "amount": 100.50
}
```

Backend Implementation:

- File: `backend/api/admin/finance/wallet/[id]/balance.post.ts`
- Creates transaction with type `INCOMING_TRANSFER` (ADD) or `OUTGOING_TRANSFER` (SUBTRACT)
- Transaction status is automatically set to `COMPLETED`
- Metadata includes `{ method: "ADMIN" }`

Transaction Created:

```
{
  userId: wallet.userId,
  walletId: wallet.id,
  amount: amount,
  type: "INCOMING_TRANSFER" | "OUTGOING_TRANSFER",
  status: "COMPLETED",
  metadata: {
    method: "ADMIN"
  },
  description: "Admin added/subtracted X CURRENCY to wallet"
}
```

☑ 2. Deposit Approval/Rejection

Endpoint: `PUT /api/admin/finance/transaction/:id`

How it works:

- View all PENDING deposits in Transaction Management page
- Edit a PENDING transaction to approve or reject it
- For DEPOSIT type transactions:
    - `status: "COMPLETED"` → Adds funds to wallet
    - `status: "REJECTED"` → No balance change (deposit rejected)

Supported Transaction Types:

- `DEPOSIT` - User deposits
- `WITHDRAW` - User withdrawals
- `FOREX_DEPOSIT` - Forex account deposits
- `FOREX_WITHDRAW` - Forex account withdrawals
- `ICO_CONTRIBUTION` - ICO contributions
- `INCOMING_TRANSFER` - Internal transfers

Backend Implementation:

- File: `backend/api/admin/finance/transaction/[id]/index.put.ts`
- Handles different transaction types with appropriate balance updates
- Sends email notifications to users
- Refunds on rejection when appropriate

DEPOSIT Flow:

```
// When status changes from PENDING to COMPLETED:
walletBalance += amount - fee;

// When status changes from PENDING to REJECTED:
// For DEPOSIT: No balance change (user never got the funds)
// For WITHDRAW: Balance refunded (user gets money back)
walletBalance += amount; // Refund
```

☑ 3. Transaction Statuses

All transactions support these statuses:

- `PENDING` - Awaiting admin approval
- `COMPLETED` - Approved and processed
- `REJECTED` - Rejected by admin

- `FAILED` - Processing failed
- `CANCELLED` - Cancelled by user/system
- `EXPIRED` - Transaction expired
- `REFUNDED` - Transaction refunded
- `FROZEN` - Transaction frozen
- `PROCESSING` - Currently processing
- `TIMEOUT` - Transaction timed out

UI Access

Wallet Management

Location: `/admin/finance/wallet`

Features:

- View all user wallets
- See balance and in-order amounts
- Filter by currency, type, user
- Enable/disable wallets

Current UI: `src/pages/admin/finance/wallet/index.tsx`

To manually adjust balance (via API):

```
curl -X POST /api/admin/finance/wallet/{wallet-id}/balance \
  -H "Content-Type: application/json" \
  -d '{"type": "ADD", "amount": 100}'
```

Transaction Management

Location: `/admin/finance/transaction`

Features:

- View all transactions
- Filter by type, status, user
- Edit PENDING transactions
- Change status to approve/reject

Current UI: `src/pages/admin/finance/transaction/index.tsx`

To approve a deposit:

1. Go to Transaction Management
2. Find the PENDING DEPOSIT transaction
3. Click "Edit"
4. Change `status` to `COMPLETED`

5. Optionally update `amount` or `fee`
6. Save

To reject a deposit:

1. Go to Transaction Management
2. Find the PENDING DEPOSIT transaction
3. Click "Edit"
4. Change `status` to `REJECTED`
5. Add rejection message in `metadata.message`
6. Save

Database Schema

Transaction Model

```
interface Transaction {
  id: string;
  userId: string;
  walletId: string;
  type: TransactionType;
  status: TransactionStatus;
  amount: number;
  fee?: number;
  description?: string;
  metadata?: any;
  referenceId?: string;
  createdAt: Date;
  updatedAt: Date;
}
```

Model Location: `types/models/transaction.d.ts`

Wallet Model

```
interface Wallet {
  id: string;
  userId: string;
  type: "FIAT" | "SPOT" | "ECO";
  currency: string;
  balance: number;
  inOrder: number;
  status: boolean;
  createdAt: Date;
  updatedAt: Date;
}
```

Implementation Examples

Example 1: Add Funds to User Wallet

Scenario: User reports missing deposit, admin wants to credit their account

API Call:

```
POST /api/admin/finance/wallet/abc-123-wallet-id/balance
{
  "type": "ADD",
  "amount": 500
}
```

Result:

- Wallet balance increases by 500
- Transaction created with type `INCOMING_TRANSFER`
- User receives email notification
- Transaction visible in user's history

Example 2: Approve Pending Deposit

Scenario: User deposited via bank transfer, admin confirms payment received

Steps:

1. Find transaction in Transaction Management
2. Verify deposit in bank account
3. Edit transaction:

```
PUT /api/admin/finance/transaction/txn-123
{
  "status": "COMPLETED",
  "amount": 1000,
  "fee": 0
}
```

Result:

- Wallet balance increases by 1000
- User receives confirmation email
- Transaction status shows COMPLETED

Example 3: Reject Fraudulent Deposit

Scenario: Suspicious deposit attempt detected

Steps:

1. Find transaction in Transaction Management
2. Edit transaction:

```
PUT /api/admin/finance/transaction/txn-456
{
  "status": "REJECTED",
  "metadata": {
    "message": "Suspicious activity detected. Please contact support."
  }
}
```

Result:

- No balance change
- User receives rejection email with message
- Transaction marked as REJECTED

Adding UI Buttons (Optional Enhancement)

If you want to add dedicated UI buttons for these actions:

Option 1: Add to Wallet Page (Quick Balance Adjustment)

Add a custom row action in `src/pages/admin/finance/wallet/index.tsx`:

```
// Use columnConfig actions array
{
  field: "balance",
  label: "Balance",
  type: "number",
  sortable: true,
  actions: [
    {
      icon: "mdi:cash-plus",
      color: "success",
      tooltip: "Add Funds",
      onClick: async (item) => {
        const amount = prompt("Enter amount to add:");
        if (amount) {
          await $fetch({
            url: `/api/admin/finance/wallet/${item.id}/balance`,
            method: "POST",
            body: { type: "ADD", amount: Number(amount) }
          });
        }
      }
    },
    {
      icon: "mdi:cash-minus",
      color: "danger",
      tooltip: "Subtract Funds",
      onClick: async (item) => {
        const amount = prompt("Enter amount to subtract:");
        if (amount) {
          await $fetch({
            url: `/api/admin/finance/wallet/${item.id}/balance`,
            method: "POST",
            body: { type: "SUBTRACT", amount: Number(amount) }
          });
        }
      }
    }
  ]
}
```

Option 2: Add to Transaction Page (Quick Approve/Reject)

Add quick action buttons in `src/pages/admin/finance/transaction/index.tsx`:

```
{
  field: "status",
  label: "Status",
  type: "select",
  options: statusOptions,
  sortable: true,
  actions: [
    {
      icon: "mdi:check-circle",
      color: "success",
      tooltip: "Approve",
      condition: (item) => item.status === "PENDING" && item.type ===
"DEPOSIT",
      onClick: async (item) => {
        await $fetch({
          url: `/api/admin/finance/transaction/${item.id}`,
          method: "PUT",
          body: { ...item, status: "COMPLETED" }
        });
      }
    },
    {
      icon: "mdi:close-circle",
      color: "danger",
      tooltip: "Reject",
      condition: (item) => item.status === "PENDING" && item.type ===
"DEPOSIT",
      onClick: async (item) => {
        const message = prompt("Rejection reason:");
        await $fetch({
          url: `/api/admin/finance/transaction/${item.id}`,
          method: "PUT",
          body: {
            ...item,
            status: "REJECTED",
            metadata: { ...item.metadata, message }
          }
        });
      }
    }
  ]
}
```

Security Considerations

1. **Permissions:** All endpoints require `"Access Wallet Management"` or `"Access Transaction Management"` permission
2. **Authentication:** Admin must be logged in
3. **Audit Trail:** All manual adjustments are logged in transactions with metadata
4. **Email Notifications:** Users are notified of all balance changes
5. **Validation:**
   - Only PENDING transactions can be edited
   - Balance cannot go negative
   - Amount must be positive

Email Notifications

Balance Adjustment Email

- Sent via `sendWalletBalanceUpdateEmail()`
- Includes: action type (added/subtracted), amount, new balance

Transaction Status Email

- Sent via `sendTransactionStatusUpdateEmail()`
- Includes: transaction details, status, rejection message (if any)

Withdrawal Confirmation Email

- Sent via `sendSpotWalletWithdrawalConfirmationEmail()`
- Includes: withdrawal details, status

Common Use Cases

Use Case 1: User reports missing deposit

1. Verify deposit in payment gateway

2. Find user's wallet in Wallet Management
3. Use API to add funds: `POST /api/admin/finance/wallet/:id/balance`
4. User receives email confirmation

Use Case 2: Manual fiat deposit approval

1. User initiates bank transfer
2. Transaction created with status PENDING
3. Admin verifies bank transfer received
4. Admin edits transaction, sets status to COMPLETED
5. User's balance updated automatically

Use Case 3: Suspend suspicious account

1. Find user's wallets in Wallet Management
2. Use status toggle to disable wallets
3. Reject any pending transactions
4. Investigate further

Use Case 4: Bonus/Promotional credit

1. Find user's wallet
2. Use API to add bonus: `POST /api/admin/finance/wallet/:id/balance` with type "ADD"
3. Transaction recorded as INCOMING_TRANSFER with method "ADMIN"
4. User can see bonus in transaction history

Monitoring & Reports

View Transaction History

- Navigate to `/admin/finance/transaction`
- Filter by:
  - Type: DEPOSIT, WITHDRAW, etc.
  - Status: PENDING, COMPLETED, REJECTED
  - User: Search by email or name
  - Date range

View Wallet Balances

- Navigate to `/admin/finance/wallet`
- See all user balances
- Export to CSV for accounting

Analytics

- Both pages have analytics view (`/analysis`)
- View charts and trends
- Monitor pending transactions
- Track manual adjustments

Troubleshooting

Issue: Balance not updating after approval

Solution: Check transaction status in database, ensure it's COMPLETED not PENDING

Issue: User not receiving email

Solution: Check email settings, verify user's email address is valid

Issue: Cannot edit transaction

Solution: Only PENDING transactions can be edited. Check current status.

Issue: Error "Insufficient balance"

Solution: For SUBTRACT operations, ensure wallet has enough balance

API Reference Summary

Endpoint	Method	Purpose	Permission Required
`/api/admin/finance/wallet/:id/balance`	POST	Add/Subtract wallet balance	Access Wallet Management
`/api/admin/finance/transaction/:id`	PUT	Approve/Reject transaction	Access Transaction Management
`/api/admin/finance/wallet/:id/withdraw/approve`	POST	Approve withdrawal	Access Wallet Management
`/api/admin/finance/wallet/:id/withdraw/reject`	POST	Reject withdrawal	Access Wallet Management
`/api/admin/finance/wallet`	GET	List all wallets	Access Wallet Management
`/api/admin/finance/transaction`	GET	List all transactions	Access Transaction Management

Conclusion

The admin wallet and deposit management functionality is fully implemented. Admins can:

- [x] Manually adjust any wallet balance via API
- [x] Approve/reject deposits by editing transactions
- [x] View complete transaction history
- [x] Receive audit trails for all actions
- [x] Users are automatically notified of all changes

For UI enhancements, the `actions` array in column config provides a simple way to add quick-action buttons without modifying core components.