Travian Whispers API Documentation

This document provides a comprehensive overview of all API endpoints available in the Travian Whispers application. The API is organized into multiple sections based on functionality.

Table of Contents

Authentication API

API endpoints related to user authentication, registration, and password management.

Login

Endpoint: `/auth/login`

Method: `POST`

Description: Authenticates a user and creates a new session.

Parameters:

- `username`: Username or email address of the user.
- `password`: User's password.
- `remember`: Boolean indicating if the session should be remembered.

Response:

- Success: Redirects to appropriate dashboard based on role.
- Error: Redirects back to login page with error message.

Logout

Endpoint: `/auth/logout`

Method: `GET`

Description: Logs out the current user and clears the session.

Response: Redirects to login page.

Register

Endpoint: `/auth/register`

Method: `POST`

Description: Registers a new user account.

Parameters:

- `username`: Desired username.
- `email`: User's email address.
- `password`: User's password.
- `confirm_password`: Password confirmation.
- `terms`: Boolean indicating acceptance of terms and conditions.

Response:

- Success: Redirects to login page with success message.
- Error: Redirects back to registration page with error message.

Verify Email

Endpoint: `/auth/verify-email/<token>`

Method: `GET`

Description: Verifies a user's email address using the provided token.

URL Parameters:

- `token`: Email verification token.

Response:

- Success: Redirects to login page with success message.
- Error: Displays verification failed page.

Resend Verification Email

Endpoint: `/auth/resend-verification`

Method: `POST`

Description: Resends the verification email to the specified address.

Parameters:

- `email`: Email address to send verification to.

Response:

- Success: Redirects to login page with success message.
- Error: Redirects back to resend verification page with error message.

Forgot Password

Endpoint: `/auth/forgot-password`

Method: `POST`

Description: Initiates password reset process.

Parameters:

- `email`: User's email address.

Response:

- Success: Redirects to login page with info message.
- Error: Redirects back to forgot password page with error message.

Reset Password

Endpoint: `/auth/reset-password`

Method: `GET/POST`

Description: Resets user's password using provided token.

Query Parameters (GET):

- `token`: Password reset token.

Form Parameters (POST):

- `token`: Password reset token.
- `new_password`: New password.
- `confirm_password`: New password confirmation.

Response:

- Success: Redirects to login page with success message.
- Error: Redirects back to reset password page with error message.

User API

API endpoints for user dashboard functionality and user settings.

Dashboard

Endpoint: `/dashboard`

Method: `GET`

Description: Displays the user dashboard with personalized information.

Requirements: User authentication required.

Response: User dashboard page with account overview and feature summaries.

User Profile

Endpoint: `/dashboard/profile`

Method: `GET/POST`

Description: Allows viewing and updating user profile information.

Requirements: User authentication required.

Form Parameters (POST):

- `form_type`: 'profile' or 'password' to indicate which form is being submitted.
- `email`: User's email address (for profile update).
- `notification_email`: Boolean for email notification settings.
- `auto_renew`: Boolean for auto-renewal of subscription.
- `current_password`: Current password (for password change).
- `new_password`: New password (for password change).
- `confirm_password`: New password confirmation (for password change).

Response:

- Success: Profile page with success message.
- Error: Profile page with error message.

Travian Settings

Endpoint: `/dashboard/travian-settings`

Method: `GET/POST`

Description: Allows viewing and updating Travian account credentials.

Requirements: User authentication required.

Form Parameters (POST):

- `travian_username`: Travian account username.
- `travian_password`: Travian account password (masked if unchanged).
- `travian_server`: Travian server URL.

- `travian_tribe`: Selected Travian tribe.

Response:

- Success: Travian settings page with success message.
- Error: Travian settings page with error message.

Villages Management

Endpoint: `/dashboard/villages`

Method: `GET`

Description: Displays and manages user's villages data.

Requirements: User authentication required.

Response: Villages management page showing all registered villages.

Auto Farm Management

Endpoint: `/dashboard/auto-farm`

Method: `GET`

Description: Access auto-farm feature for resource farming automation.

Requirements:

- User authentication required.
- Active subscription with Auto-Farm feature.

Response:

- Success: Auto-farm management page.
- Error: Redirect to subscription page with warning if feature not available.

Troop Trainer

Endpoint: `/dashboard/troop-trainer`

Method: `GET`

Description: Access troop training automation feature.

Requirements:

- User authentication required.
- Active subscription with Troop Trainer feature.

Response:

- Success: Troop trainer management page.
- Error: Redirect to subscription page with warning if feature not available.

Activity Logs

Endpoint: `/dashboard/activity-logs`

Method: `GET`

Description: Shows user's activity history.

Requirements: User authentication required.

Query Parameters:

- `page`: Page number for pagination.
- `per_page`: Number of items per page.
- `type`: Filter by activity type.
- `status`: Filter by activity status.

Response: Activity logs page with filtered logs and pagination.

Subscription Management

Endpoint: `/dashboard/subscription`

Method: `GET`

Description: Shows user's subscription status and available plans.

Requirements: User authentication required.

Response: Subscription management page with current plan and available options.

Support

Endpoint: `/dashboard/support`

Method: `GET`

Description: Access help and support resources.

Requirements: User authentication required.

Response: Support page with help resources and contact options.

IP Management

Endpoint: `/dashboard/ip/status`

Method: `GET`

Description: Shows IP status for user's proxy connections.

Requirements: User authentication required.

Response: IP status page with current assignments and statistics.

Endpoint: `/dashboard/ip/rotate`

Method: `POST`

Description: Rotates the user's IP address.

Requirements: User authentication required.

Response: Redirects to IP status page with success/error message.

Endpoint: `/dashboard/ip/rotate_identity`

Method: `POST`

Description: Rotates the user's entire digital identity (IP and browser session).

Requirements: User authentication required.

Response: Redirects to IP status page with success/error message.

Endpoint: `/dashboard/ip/session_info`

Method: `GET`

Description: Shows browser session information.

Requirements: User authentication required.

Response: Session info page with browser session details.

Endpoint: `/dashboard/ip/clear_session`

Method: `POST`

Description: Clears the user's browser session.

Requirements: User authentication required.

Response: Redirects to session info page with success/error message.

Admin API

API endpoints for the admin panel functionality.

Admin Dashboard

Endpoint: `/admin`

Method: `GET`

Description: Displays the admin dashboard with system statistics.

Requirements: Admin authentication required.

Response: Admin dashboard page with system overview.

Refresh Statistics

Endpoint: `/admin/refresh-stats`

Method: `GET`

Description: Refreshes admin dashboard statistics.

Requirements: Admin authentication required.

Response: JSON response with success status and message.

User Management

Endpoint: `/admin/users`

Method: `GET`

Description: Displays user management page with user listings.

Requirements: Admin authentication required.

Query Parameters:

- `page`: Page number for pagination.
- `per_page`: Number of items per page.
- `status`: Filter by user status (active/inactive).
- `role`: Filter by user role (admin/user).
- `q`: Search query for username or email.
- `sort`: Sort field and direction.

Response: User management page with filtered users and pagination.

Endpoint: `/admin/users/create`

Method: `GET/POST`

Description: Create a new user account.

Requirements: Admin authentication required.

Form Parameters (POST):

- `username`: New username.
- `email`: New user's email address.
- `password`: User's password.
- `role`: User role (admin/user).
- `isVerified`: Boolean indicating if email is pre-verified.
- `subscriptionStatus`: Subscription status.
- `subscriptionPlan`: Selected plan ID (if active).

Response:

- Success: Redirects to users list with success message.
- Error: User creation form with error message.

Endpoint: `/admin/users/edit/<user_id>`

Method: `GET/POST`

Description: Edit an existing user.

Requirements: Admin authentication required.

URL Parameters:

- `user_id`: ID of the user to edit.

Form Parameters (POST):

- `email`: User's email address.

- `role`: User role (admin/user).
- `status`: Account status (active/inactive).
- `new_password`: New password (if changing).
- `subscription_plan`: Selected plan ID (if changing).
- `billing_period`: Subscription billing period.

Response:

- Success: Redirects to users list with success message.
- Error: User edit form with error message.

Endpoint: `/admin/users/delete/<user_id>`

Method: `POST`

Description: Delete a user account.

Requirements: Admin authentication required.

URL Parameters:

- `user_id`: ID of the user to delete.

Response: Redirects to users list with success/error message.

Endpoint: `/admin/user/<user_id>`

Method: `GET`

Description: Get user details as JSON.

Requirements: Admin authentication required.

URL Parameters:

- `user_id`: ID of the user to view.

Response: JSON with user details or error message.

Subscription Plan Management

Endpoint: `/admin/subscriptions`

Method: `GET`

Description: Displays subscription plans management page.

Requirements: Admin authentication required.

Response: Subscription plans management page with existing plans.

Endpoint: `/admin/subscriptions/create`

Method: `GET/POST`

Description: Create a new subscription plan.

Requirements: Admin authentication required.

Form Parameters (POST):

- `planName`: Plan name.
- `planDescription`: Plan description.
- `planPrice`: Monthly price.
- `yearlyPrice`: Yearly price.
- `featureAutoFarm`: Boolean for auto-farm feature inclusion.
- `featureTrainer`: Boolean for troop trainer feature inclusion.
- `featureNotification`: Boolean for notification feature inclusion.
- `featureAdvanced`: Boolean for advanced features inclusion.
- `maxVillages`: Maximum villages allowed.
- `maxTasks`: Maximum concurrent tasks allowed.

Response:

- Success: Redirects to subscriptions list with success message.
- Error: Plan creation form with error message.

Endpoint: `/admin/subscriptions/edit/<plan_id>`

Method: `GET/POST`

Description: Edit an existing subscription plan.

Requirements: Admin authentication required.

URL Parameters:

- `plan_id`: ID of the plan to edit.

Form Parameters (POST):

- Similar to the create endpoint, with all plan details.

Response:

- Success: Redirects to subscriptions list with success message.
- Error: Plan edit form with error message.

Endpoint: `/admin/subscriptions/delete/<plan_id>`

Method: `POST`

Description: Delete a subscription plan.

Requirements: Admin authentication required.

URL Parameters:

- `plan_id`: ID of the plan to delete.

Response: Redirects to subscriptions list with success/error message.

Transaction Management

Endpoint: `/admin/transactions`

Method: `GET`

Description: Displays transaction history and management page.

Requirements: Admin authentication required.

Query Parameters:

- `page`: Page number for pagination.
- `status`: Filter by transaction status.
- `plan`: Filter by subscription plan.
- `date_from`: Filter by date range start.
- `date_to`: Filter by date range end.
- `q`: Search query for transaction ID or username.

Response: Transactions management page with filtered transactions.

Endpoint: `/admin/transactions/<transaction_id>`

Method: `GET`

Description: View transaction details.

Requirements: Admin authentication required.

URL Parameters:

- `transaction_id`: ID of the transaction to view.

Response: Transaction details page.

Endpoint: `/admin/transactions/update-status/<transaction_id>`

Method: `POST`

Description: Update transaction status.

Requirements: Admin authentication required.

URL Parameters:

- `transaction_id`: ID of the transaction to update.

Form Parameters:

- `status`: New status (completed/pending/failed/refunded).

Response: Redirects to transaction details with success/error message.

System Maintenance

Endpoint: `/admin/maintenance`

Method: `GET`

Description: Displays system maintenance page.

Requirements: Admin authentication required.

Response: Maintenance page with backup and system options.

Endpoint: `/admin/update-maintenance`

Method: `POST`

Description: Update maintenance mode settings.

Requirements: Admin authentication required.

JSON Parameters:

- `enabled`: Boolean to enable/disable maintenance mode.
- `message`: Maintenance message to display.
- `duration`: Expected duration of maintenance.

Response: JSON response with success status and message.

Endpoint: `/admin/generate-report`

Method: `POST`

Description: Generate system reports.

Requirements: Admin authentication required.

Form Parameters:

- `report_type`: Type of report to generate.
- `date_range`: Date range for the report.
- `report_format`: Output format (PDF/CSV/Excel).

Response: Redirects to admin dashboard with success/error message.

Endpoint: `/admin/create-backup`

Method: `POST`

Description: Create a database backup.

Requirements: Admin authentication required.

Form Parameters:

- `backup_type`: Type of backup (full/users/transactions/subscriptions).
- `compress_backup`: Boolean to compress the backup.

Response: Redirects to settings page with success/error message.

Endpoint: `/admin/backups`

Method: `GET`

Description: View database backups.

Requirements: Admin authentication required.

Response: Backups page with list of available backups.

Endpoint: `/admin/restore-backup`

Method: `POST`

Description: Restore a database backup.

Requirements: Admin authentication required.

Form Parameters:

- `filename`: Backup filename to restore.

Response: Redirects to settings page with success/error message.

Endpoint: `/admin/delete-backup`

Method: `POST`

Description: Delete a database backup.

Requirements: Admin authentication required.

Form Parameters:

- `filename`: Backup filename to delete.

Response: Redirects to settings page with success/error message.

Endpoint: `/admin/download-backup/<filename>`

Method: `GET`

Description: Download a backup file.

Requirements: Admin authentication required.

URL Parameters:

- `filename`: Backup filename to download.

Response: File download or redirect with message.

System Logs

Endpoint: `/admin/logs`

Method: `GET`

Description: View system logs.

Requirements: Admin authentication required.

Query Parameters:

- `level`: Filter by log level.
- `user`: Filter by username.
- `date_from`: Filter by date range start.
- `date_to`: Filter by date range end.

Response: System logs page with filtered logs.

System Settings

Endpoint: `/admin/settings`

Method: `GET/POST`

Description: View and update system settings.

Requirements: Admin authentication required.

Query Parameters (GET):

- `tab`: Active settings tab.

Form Parameters (POST):

- `form_type`: Type of settings being updated (general/email/payment/security/backup).
- Various settings parameters depending on the form type.

Response: Settings page with appropriate tab and success/error messages.

Endpoint: `/admin/test-email`

Method: `POST`

Description: Test email configuration.

Requirements: Admin authentication required.

Form Parameters:

- `testEmailAddress`: Email address to send test to.

Response: Redirects to email settings with success/error message.

IP Management (Admin)

Endpoint: `/admin/ip/admin/ips`

Method: `GET`

Description: Admin IP management page.

Requirements: Admin authentication required.

Response: IP management page with all IPs and assignments.

Endpoint: `/admin/ip/admin/add_ip`

Method: `POST`

Description: Add a new IP to the pool.

Requirements: Admin authentication required.

Form Parameters:

- `ip`: IP address.
- `port`: Port number.
- `username`: Proxy username.
- `password`: Proxy password.
- `proxy_type`: Proxy type.

Response: Redirects to IP management page with success/error message.

Endpoint: `/admin/ip/admin/remove_ip/<ip_id>`

Method: `POST`

Description: Remove an IP from the pool.

Requirements: Admin authentication required.

URL Parameters:

- `ip_id`: ID of the IP to remove.

Response: Redirects to IP management page with success/error message.

Endpoint: `/admin/ip/admin/rotate_all`

Method: `POST`

Description: Rotate all IPs that have been used for too long.

Requirements: Admin authentication required.

Form Parameters:

- `max_age_hours`: Maximum age in hours before rotation.

Response: Redirects to IP management page with success/error message.

Public API

Public-facing API endpoints that don't require authentication.

Landing Page

Endpoint: `/`

Method: `GET`

Description: Displays the main landing page.

Response: Landing page with subscription plans and features.

About Page

Endpoint: `/about`

Method: `GET`

Description: Displays the about page.

Response: About page with information about the service.

Features Page

Endpoint: `/features`

Method: `GET`

Description: Displays the features page.

Response: Features page with detailed feature descriptions.

Pricing Page

Endpoint: `/pricing`

Method: `GET`

Description: Displays the pricing page.

Response: Pricing page with subscription plans and costs.

Contact Page

Endpoint: `/contact`

Method: `GET/POST`

Description: Displays contact form and processes submissions.

Form Parameters (POST):

- `name`: Contact name.
- `email`: Contact email.
- `subject`: Message subject.
- `message`: Message content.

Response:

- GET: Contact form page.
- POST Success: Contact page with success message.
- POST Error: Contact page with error message.

FAQ Page

Endpoint: `/faq`

Method: `GET`

Description: Displays frequently asked questions.

Response: FAQ page with questions and answers.

Terms Page

Endpoint: `/terms`

Method: `GET`

Description: Displays terms and conditions.

Response: Terms and conditions page.

Privacy Policy Page

Endpoint: `/privacy`

Method: `GET`

Description: Displays privacy policy.

Response: Privacy policy page.

Subscription & Payment API

API endpoints related to subscription management and payment processing.

User Villages Update

Endpoint: `/api/user/villages/update`

Method: `POST`

Description: Updates user's villages.

Requirements: User authentication required.

JSON Parameters:

- `villages`: Array of village data.

Response: JSON response with success status, message, and updated villages data.

User Settings Update

Endpoint: `/api/user/settings/update`

Method: `POST`

Description: Updates user's settings.

Requirements: User authentication required.

JSON Parameters:

- `settings`: Object with settings to update.

Response: JSON response with success status, message, and updated settings data.

User Travian Credentials Update

Endpoint: `/api/user/travian-credentials/update`

Method: `POST`

Description: Updates user's Travian account credentials.

Requirements: User authentication required.

JSON Parameters:

- `travianCredentials`: Object with credentials to update.

Response: JSON response with success status and message.

Create Subscription Order

Endpoint: `/api/subscription/create-order`

Method: `POST`

Description: Creates a PayPal order for subscription payment.

Requirements: User authentication required.

JSON Parameters:

- `planId`: ID of the selected subscription plan.

Response: JSON response with success status, order ID, and approval URL.

Process Payment

Endpoint: `/api/subscription/process-payment`

Method: `POST`

Description: Processes a successful PayPal payment.

Requirements: User authentication required.

JSON Parameters:

- `orderId`: PayPal order ID to process.

Response: JSON response with success status and message.

Cancel Subscription

Endpoint: `/api/subscription/cancel`

Method: `POST`

Description: Cancels a user's active subscription.

Requirements: User authentication required.

Response: JSON response with success status and message.

PayPal Webhook

Endpoint: `/api/webhooks/paypal`

Method: `POST`

Description: Webhook endpoint for PayPal payment notifications.

JSON Parameters: PayPal event data.

Headers:

- Webhook signature headers for verification.

Response: JSON response with success status and message.

Automation Features API

API endpoints specific to the automation features of the application.

User Profile API

Endpoint: `/api/user/profile`

Method: `GET`

Description: Gets user profile data.

Requirements: User authentication required.

Response: JSON response with user profile information.

User Villages API

Endpoint: `/api/user/villages`

Method: `GET`

Description: Gets user's villages data.

Requirements: User authentication required.

Response: JSON response with villages data.

System API

API endpoints for system operations and administration.

Admin Create Backup API

Endpoint: `/admin/admin/create-backup`

Method: `POST`

Description: Creates a database backup via API.

Requirements: Admin authentication required.

JSON Parameters:

- `backup_type`: Type of backup.
- `compress_backup`: Boolean to compress backup.

Response: JSON response with success status, filename, and message.

Admin Update Maintenance API

Endpoint: `/admin/admin/update-maintenance`

Method: `POST`

Description: Updates maintenance mode settings via API.

Requirements: Admin authentication required.

JSON Parameters:

- `enabled`: Boolean to enable/disable maintenance mode.
- `message`: Maintenance message to display.
- `duration`: Expected duration of maintenance.

Response: JSON response with success status and message.

Notes

1. All admin routes are protected by the `@admin_required` decorator which verifies both user authentication and admin role.
2. All user dashboard routes are protected by the `@login_required` decorator which verifies user authentication.
3. Some features (like Auto-Farm and Troop Trainer) require an active subscription with appropriate features enabled.
4. Form validation is generally performed both client-side (with JavaScript) and server-side.
5. The API returns appropriate HTTP status codes for different scenarios.
6. Many endpoints support JSON response format for API usage as well as HTML templates for browser usage.