Moozup - API Documentation

Version: 1.0
Backend: Node.js (Express) + PostgreSQL
Date: April 28, 2025

Table of Contents

---

1. Introduction

The Moozup Clone API provides a robust backend for managing users, events, and communities across a mobile app and a web-based admin panel. Built with **Node.js**, **Express**, and **PostgreSQL**, with **Supabase** for authentication and real-time synchronization, this API supports both user-facing mobile functionalities and administrative website tasks. This documentation outlines all endpoints, request/response formats, and best practices for integration.

---

2. Authentication

All APIs (except public ones) require authentication via **Supabase** JWT tokens.

- **Token Generation:** Obtain a token via `/auth/login` or `/auth/register`.
- **Token Usage:** Include the token in the `Authorization` header as `Bearer <token>`.

---

3. Headers

Header	Description
`Content-Type`	`application/json` for JSON payloads
`Authorization`	`Bearer <JWT_TOKEN>` for authenticated requests
`Accept-Language`	Optional, e.g., `en-US` for localization

---

4. Error Handling

Errors are returned in a standardized JSON format:

```
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "Detailed error message"
  }
}
```

Common error codes:

- `400`: Bad Request
- `401`: Unauthorized
- `403`: Forbidden
- `404`: Not Found
- `500`: Internal Server Error

---

5. API Grouping Overview

The APIs are grouped by platform (Mobile App and Website) and functionality.

Group	Mobile APIs	Website APIs
Authentication	5	5
Users	10	5
Events	15	10
Communities	15	8
Notifications	5	2
Chat / Messaging	10	5
Media Upload	5	3
Admin Management	5	5
Payments & Subscriptions	10	4
Settings & Miscellaneous	15	3
Total	**95**	**50**

---

6. Mobile App APIs (95 APIs)

6.1 Authentication APIs

Method	URL	Description
POST	`/auth/register`	Register a new user
POST	`/auth/login`	Login and obtain JWT token
POST	`/auth/logout`	Logout and invalidate token
POST	`/auth/refresh-token`	Refresh JWT token
POST	`/auth/reset-password`	Request password reset

6.2 User APIs

Method	URL	Description
GET	`/user/profile`	Get user profile
PUT	`/user/profile`	Update user profile
DELETE	`/user/delete`	Delete user account
GET	`/user/events`	Get user's joined/created events
GET	`/user/communities`	Get user's joined/created communities
POST	`/user/follow/:id`	Follow another user
POST	`/user/unfollow/:id`	Unfollow a user
GET	`/user/followers`	List user's followers
GET	`/user/following`	List users followed by the user
POST	`/user/report/:id`	Report a user

6.3 Event APIs

Method	URL	Description
POST	`/api/event/create`	Create a new event
GET	`/api/event/list`	Fetch list of all events
GET	`/api/event/details/:id`	Get detailed info about an event
PUT	`/api/event/update/:id`	Update an existing event
DELETE	`/api/event/delete/:id`	Delete an event
POST	`/api/event/join/:id`	Join an event
POST	`/api/event/leave/:id`	Leave an event
GET	`/api/event/my-events`	Get all events created by the user

POST	`/api/event/rsvp/:id`	RSVP to an event
GET	`/api/event/categories`	Fetch available event categories
GET	`/api/event/search`	Search events by keyword/location
POST	`/api/event/report/:id`	Report an inappropriate event
POST	`/api/event/invite`	Invite others to an event
GET	`/api/event/attendees/:id`	List attendees for an event
POST	`/api/event/upload-banner`	Upload event cover image/banner

Example — Create Event:

POST /api/event/create
Content-Type: application/json
Authorization: Bearer <JWT_TOKEN>

```
{
  "title": "Tech Meetup",
  "description": "Annual tech event.",
  "location": "New York",
  "date": "2025-08-01"
}
```

Response:

```
{
  "success": true,
  "message": "Event created successfully",
  "eventId": "xyz123"
}
```

6.4 Community APIs

Method	URL	Description

POST	`/api/community/create`	Create a new community
GET	`/api/community/list`	Get all communities
GET	`/api/community/details/:id`	Get details about a community
PUT	`/api/community/update/:id`	Update a community
DELETE	`/api/community/delete/:id`	Delete a community
POST	`/api/community/join/:id`	Join a community
POST	`/api/community/leave/:id`	Leave a community
GET	`/api/community/members/:id`	List all members of a community
GET	`/api/community/my-communities`	Get communities created by the user
POST	`/api/community/invite`	Invite users to a community
GET	`/api/community/categories`	Fetch community categories
POST	`/api/community/post/create`	Create a post in a community
GET	`/api/community/post/list/:communityId`	List posts inside a community
POST	`/api/community/post/like/:postId`	Like a community post
POST	`/api/community/post/comment/:postId`	Comment on a community post

6.5 Chat and Notifications APIs

Method	URL	Description
POST	`/chat/send`	Send a chat message
GET	`/chat/conversations`	List user's conversations
GET	`/chat/messages/:conversationId`	Get messages in a conversation
POST	`/chat/group/create`	Create a group chat
DELETE	`/chat/message/delete/:id`	Delete a chat message
GET	`/notifications/list`	List user notifications
POST	`/notifications/mark-read`	Mark notification as read
POST	`/notifications/settings`	Update notification preferences
GET	`/notifications/unread-count`	Get count of unread notifications
DELETE	`/notifications/clear`	Clear all notifications

6.6 Media Upload APIs

Method	URL	Description
POST	`/media/upload/profile`	Upload user profile picture
POST	`/media/upload/event`	Upload event media
POST	`/media/upload/community`	Upload community media
DELETE	`/media/delete/:id`	Delete uploaded media
GET	`/media/list/:type`	List media by type (profile/event/community)

6.7 Payment and Subscription APIs

Method	URL	Description
POST	`/payment/subscribe`	Subscribe to a premium plan
POST	`/payment/cancel-subscription`	Cancel subscription
GET	`/payment/subscription`	Get subscription details
POST	`/payment/event-ticket/:id`	Purchase event ticket
GET	`/payment/history`	Get payment history
POST	`/payment/refund/:id`	Request refund for a payment
GET	`/payment/plans`	List available subscription plans
POST	`/payment/update-method`	Update payment method
POST	`/payment/verify`	Verify payment status
GET	`/payment/invoices`	List user invoices

6.8 Miscellaneous APIs

Method	URL	Description
GET	`/settings/preferences`	Get user preferences
PUT	`/settings/preferences`	Update user preferences
GET	`/faq`	Get FAQ content
POST	`/feedback`	Submit user feedback
GET	`/analytics/user`	Get user analytics (e.g., engagement)
GET	`/search/global`	Global search (users/events/communities)
POST	`/report/abuse`	Report abuse (general)
GET	`/terms`	Get terms of service

GET	`/privacy`	Get privacy policy
POST	`/contact`	Submit contact form
GET	`/categories/global`	Get global categories
POST	`/survey/submit`	Submit survey response
GET	`/survey/list`	List available surveys
POST	`/bookmark/add/:type/:id`	Bookmark an event/community
GET	`/bookmark/list`	List user bookmarks

---

7. Website APIs (50 APIs)

Admin Event and Community Management APIs

Method	URL	Description
GET	`/admin/event/list`	List all events
PUT	`/admin/event/approve/:id`	Approve an event
DELETE	`/admin/event/delete/:id`	Delete an event
GET	`/admin/community/list`	List all communities
PUT	`/admin/community/approve/:id`	Approve a community
DELETE	`/admin/community/delete/:id`	Delete a community
GET	`/admin/event/reports`	List reported events
GET	`/admin/community/reports`	List reported communities

PUT	`/admin/event/feature/:id`	Feature an event
PUT	`/admin/community/feature/:id`	Feature a community

Admin User Management APIs

Method	URL	Description
GET	`/admin/user/list`	List all users
PUT	`/admin/user/ban/:id`	Ban a user
PUT	`/admin/user/unban/:id`	Unban a user
GET	`/admin/user/reports`	List reported users
DELETE	`/admin/user/delete/:id`	Delete a user account

Website User APIs

Method	URL	Description
GET	`/user/profile`	Get user profile
PUT	`/user/profile`	Update user profile
GET	`/user/events`	Get user’s joined/created events
GET	`/user/communities`	Get user’s joined/created communities
POST	`/user/report/:id`	Report a user

Website Event APIs

Method	URL	Description
GET	`/api/event/list`	Fetch list of all events
GET	`/api/event/details/:id`	Get detailed info about an event

POST	`/api/event/join/:id`	Join an event
POST	`/api/event/leave/:id`	Leave an event
GET	`/api/event/my-events`	Get all events created by the user
POST	`/api/event/rsvp/:id`	RSVP to an event
GET	`/api/event/categories`	Fetch available event categories
GET	`/api/event/search`	Search events by keyword/location
POST	`/api/event/report/:id`	Report an inappropriate event
GET	`/api/event/attendees/:id`	List attendees for an event

Website Community APIs

Method	URL	Description
GET	`/api/community/list`	Get all communities
GET	`/api/community/details/:id`	Get details about a community
POST	`/api/community/join/:id`	Join a community
POST	`/api/community/leave/:id`	Leave a community
GET	`/api/community/members/:id`	List all members of a community
GET	`/api/community/my-communities`	Get communities created by the user
POST	`/api/community/post/like/:postId`	Like a community post

POST	`/api/community/post/comment/:postId`	Comment on a community post

Website Notification APIs

Method	URL	Description
GET	`/notifications/list`	List user notifications
POST	`/notifications/mark-read`	Mark notification as read

Website Chat APIs

Method	URL	Description
POST	`/chat/send`	Send a chat message
GET	`/chat/conversations`	List user’s conversations
GET	`/chat/messages/:conversationId`	Get messages in a conversation
POST	`/chat/group/create`	Create a group chat
DELETE	`/chat/message/delete/:id`	Delete a chat message

Website Media Upload APIs

Method	URL	Description
POST	`/media/upload/profile`	Upload user profile picture
POST	`/media/upload/event`	Upload event media
DELETE	`/media/delete/:id`	Delete uploaded media

Website Payment APIs

Method	URL	Description
POST	`/payment/subscribe`	Subscribe to a premium plan

GET	`/payment/subscription`	Get subscription details
POST	`/payment/event-ticket/:id`	Purchase event ticket
GET	`/payment/history`	Get payment history

Website Miscellaneous APIs

Method	URL	Description
GET	`/settings/preferences`	Get user preferences
PUT	`/settings/preferences`	Update user preferences
POST	`/feedback`	Submit user feedback

---

8. Database Schema Overview

The database is built using **PostgreSQL** with **Supabase** for authentication. Key tables and relationships:

- **Users**: Stores user details (id, email, username, profile_picture, etc.).
- **Events**: Stores event details (id, title, description, location, date, creator_id, etc.).
- **Communities**: Stores community details (id, name, description, creator_id, etc.).
- **Posts**: Stores community posts (id, community_id, user_id, content, etc.).
- **Event_Attendees**: Many-to-many relationship between Users and Events.
- **Threads
- **Community_Members**: Many-to-many relationship between Users and Communities.
- **Notifications**: Stores user notifications (id, user_id, type, message, etc.).
- **Messages**: Stores chat messages (id, conversation_id, sender_id, content, etc.).

---

9. Security and Best Practices

- **Authentication:** Use Supabase JWT for secure authentication.
- **Input Validation:** Sanitize all inputs to prevent SQL injection and XSS.
- **Rate Limiting:** Implement rate limiting to prevent abuse.
- **HTTPS:** Enforce HTTPS for all API calls.
- **Data Encryption:** Encrypt sensitive data (e.g., passwords) at rest and in transit.
- **Error Logging:** Log errors securely without exposing sensitive information.

---

10. Deployment and Environment Details

- **Backend Hosting:** Vercel for serverless functions and API routes.
- **Database Hosting:** Supabase for PostgreSQL and authentication.
- **CI/CD:** Automated deployments via GitHub Actions and Vercel.
- **Environment Variables:**
    - `SUPABASE_URL`: Supabase project URL
    - `SUPABASE_KEY`: Supabase API key
    - `JWT_SECRET`: Secret for JWT token generation
    - `NODE_ENV`: Environment (development/production)

🚀 Final Comments

✅ This documentation provides a comprehensive overview of the Moozup Clone API for both mobile and website platforms.
✅ All APIs are grouped clearly, with detailed request/response examples for key endpoints.
✅ The database schema and security practices ensure a robust and scalable backend.