Report for Health & Fitness Tracker

Outline

Health & Fitness Tracker is a comprehensive fitness activity management web application. Visitors can browse and search public fitness activities without authentication. Registered users gain full access to record personal activities (running, cycling, swimming, gym, yoga, etc.) with detailed metrics including duration, distance, and calories burned. The application features advanced multi-criteria filtering by activity type, date ranges, duration, and calorie thresholds. Users can view personalized statistics, interactive Chart.js visualizations (type distribution pie charts and daily calorie line charts), and export filtered data to CSV format. Administrative users access audit logs for security monitoring, user management, and content moderation tools. The system provides a complete REST API with Bearer token authentication for third-party integrations, documented through an interactive API builder interface. Built with modern web technologies: Node.js runtime, Express.js framework for routing and middleware, EJS templating for server-side rendering, MySQL relational database for data persistence, and Chart.js library for client-side data visualization.

Architecture

```
[Browser] <--HTTPS--> [Express/Node.js]
  |                        |
  | EJS + Chart.js         +-- Session Middleware
  | fetch API              +-- Helmet (Security Headers)
  | CSRF tokens            +-- CSRF Protection
  |                        +-- Rate Limiting
  |                        +-- Static Files
  |                        +-- Route Modules:
  |                             ├── auth.js (register, login, password)
  |                             ├── main.js (activity CRUD, profile, admin)
  |                             ├── internal.js (charts, export, stats API)
  |                             └── api.js (REST API + Bearer Token)
  |                        |
  |                   [MySQL Driver]
  |                        |
  |                   [MySQL Database]
                      ├── users
                      ├── fitness_activities
                      ├── email_verifications
                      └── audit_logs
```

Two-tier Model-View-Controller architecture separating presentation, business logic, and data access. Application tier built on Express.js with layered middleware: session management (express-session), security hardening (Helmet HTTP headers, csurf CSRF protection), and rate limiting preventing brute force attacks. Four route modules handle distinct concerns: auth.js (registration/authentication), main.js (activity CRUD/admin dashboards), internal.js (session-protected chart/stats endpoints for frontend), api.js (public REST API with Bearer tokens). EJS templates render server-side HTML; Chart.js provides client visualizations via fetch API. Data tier uses MySQL with normalized tables (users, fitness_activities, email_verifications, audit_logs), foreign key constraints ensuring referential integrity, and parameterized queries preventing SQL injection.

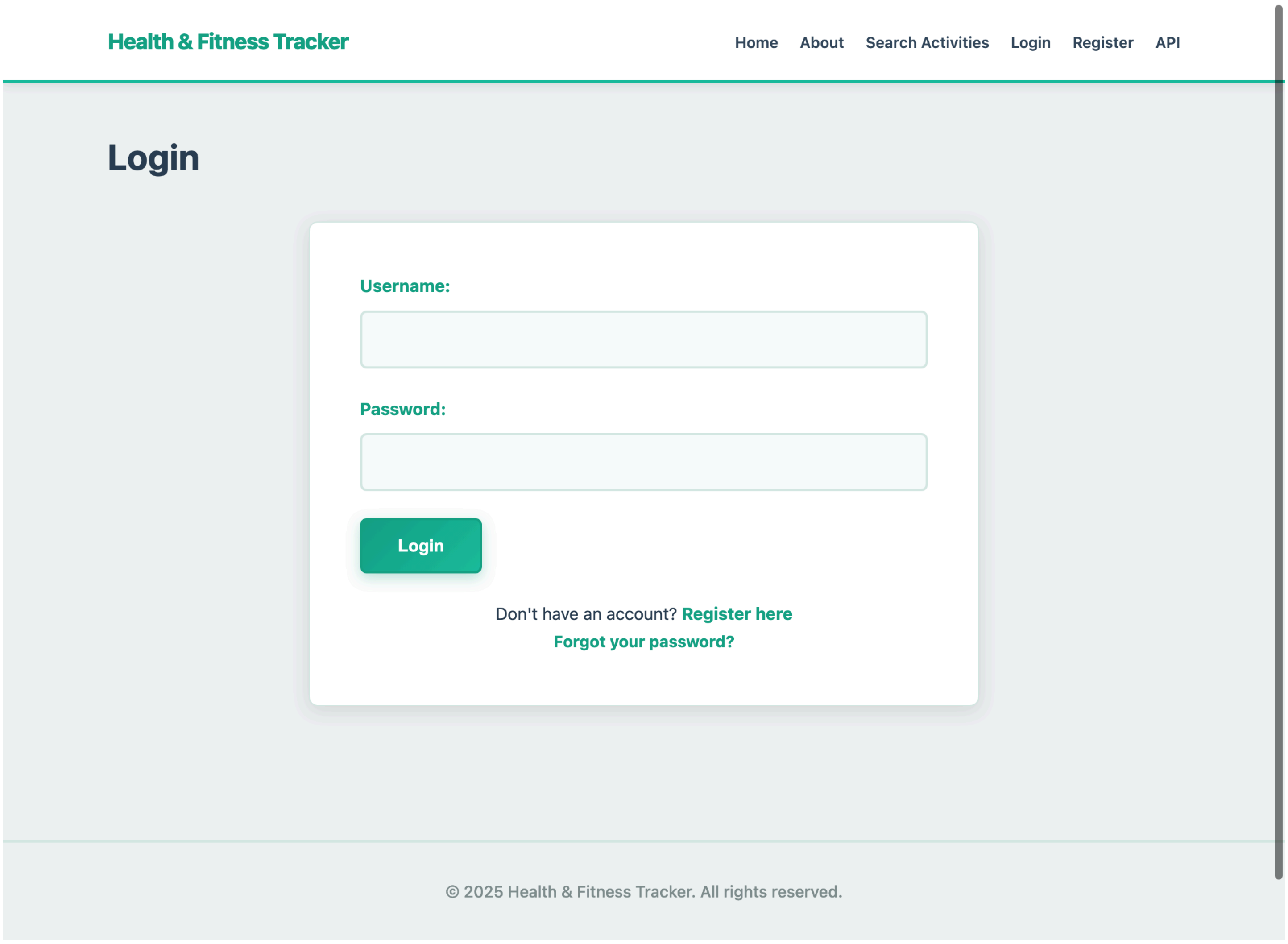
Data Model

Table	Key Columns	Purpose & Relationships
users	id, username, password (bcrypt), email, is_admin	Authentication. One-to-many with activities, verifications, logs
fitness_activities	id, user_id (FK), activity_type, activity_time, duration_minutes, distance_km, calories_burned, is_public	Activity records. Indexed on time, type, user_id
email_verifications	id, user_id (FK), new_email, verification_code, expires_at	Codes for email/password changes (30min TTL)
audit_logs	id, user_id (FK), event_type, resource_type, resource_id, changes (JSON), ip_address, created_at	Security tracking. Indexed on event_type, user_id

Four-table normalized MySQL schema. Users table stores bcrypt-hashed passwords (cost 10) and admin flags. Activities table uses is_public boolean for visibility control (public sharing vs private). Composite indexes on (activity_time, activity_type, user_id) accelerate filtered queries. Email verifications support secure workflows with time-limited codes (30min expiration). Audit logs use JSON data type for flexible change tracking, storing field snapshots with metadata (IP, user agent, paths). Foreign keys enforce referential integrity with CASCADE deletes preventing orphans. All queries use parameterized prepared statements eliminating SQL injection.

User Functionality

Application includes home, about, search/filter, data entry, and log viewing pages. Database initialization is performed directly via SQL scripts: run `create_db.sql` to drop/recreate the schema and `insert_test_data.sql` to insert seed users and activities. Default test credentials: `gold/smiths` (admin) and `testuser/smiths` (regular user).


Health & Fitness Tracker

Home About Search Activities Login Register API

Login

Username:

Password:

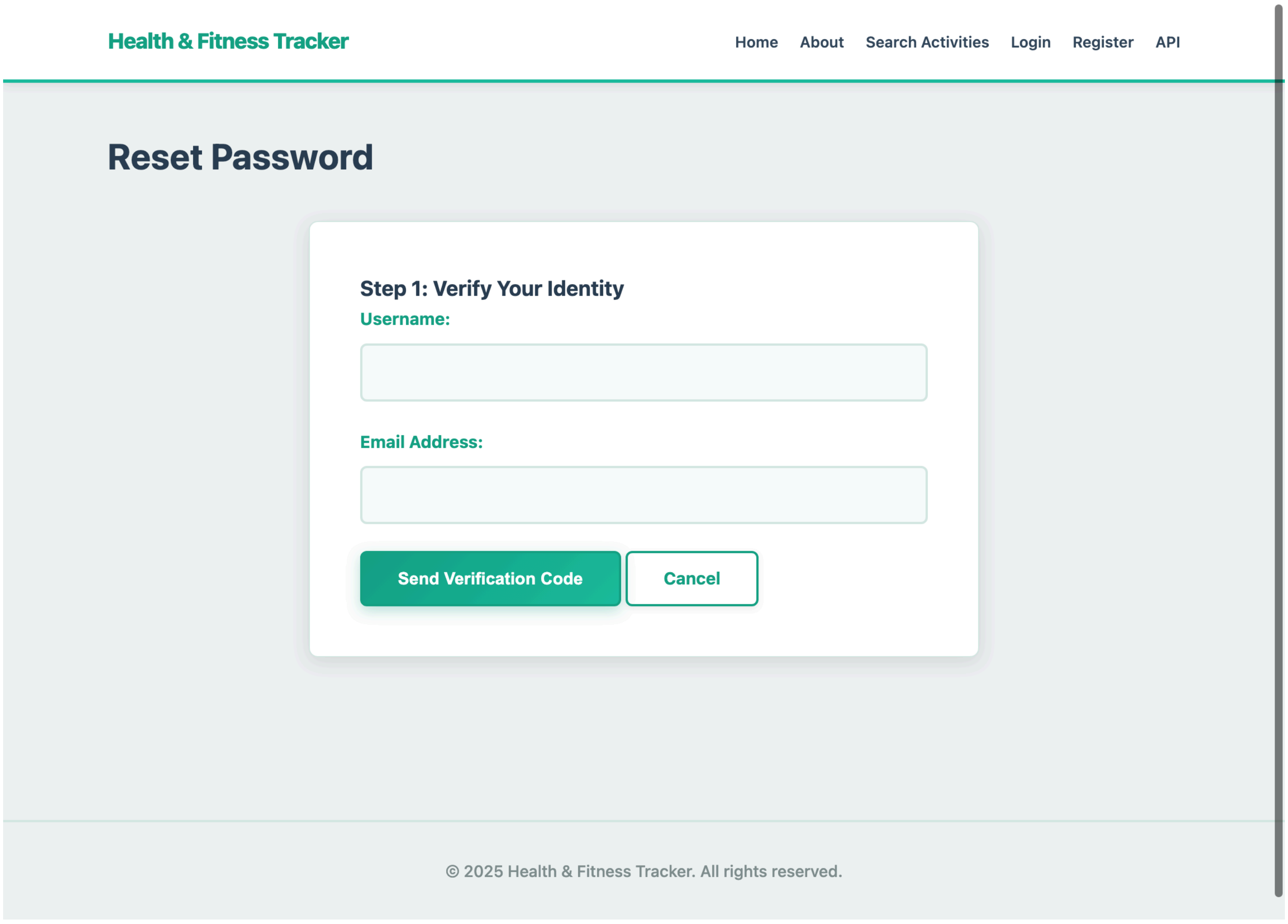
Login

Don't have an account? Register here

Forgot your password?

© 2025 Health & Fitness Tracker. All rights reserved.

Users can register, login, logout, and reset forgotten passwords via email verification codes (using Nodemailer). After login, users can modify profile information, change password, change email, or delete account with critical operations recorded in audit logs.



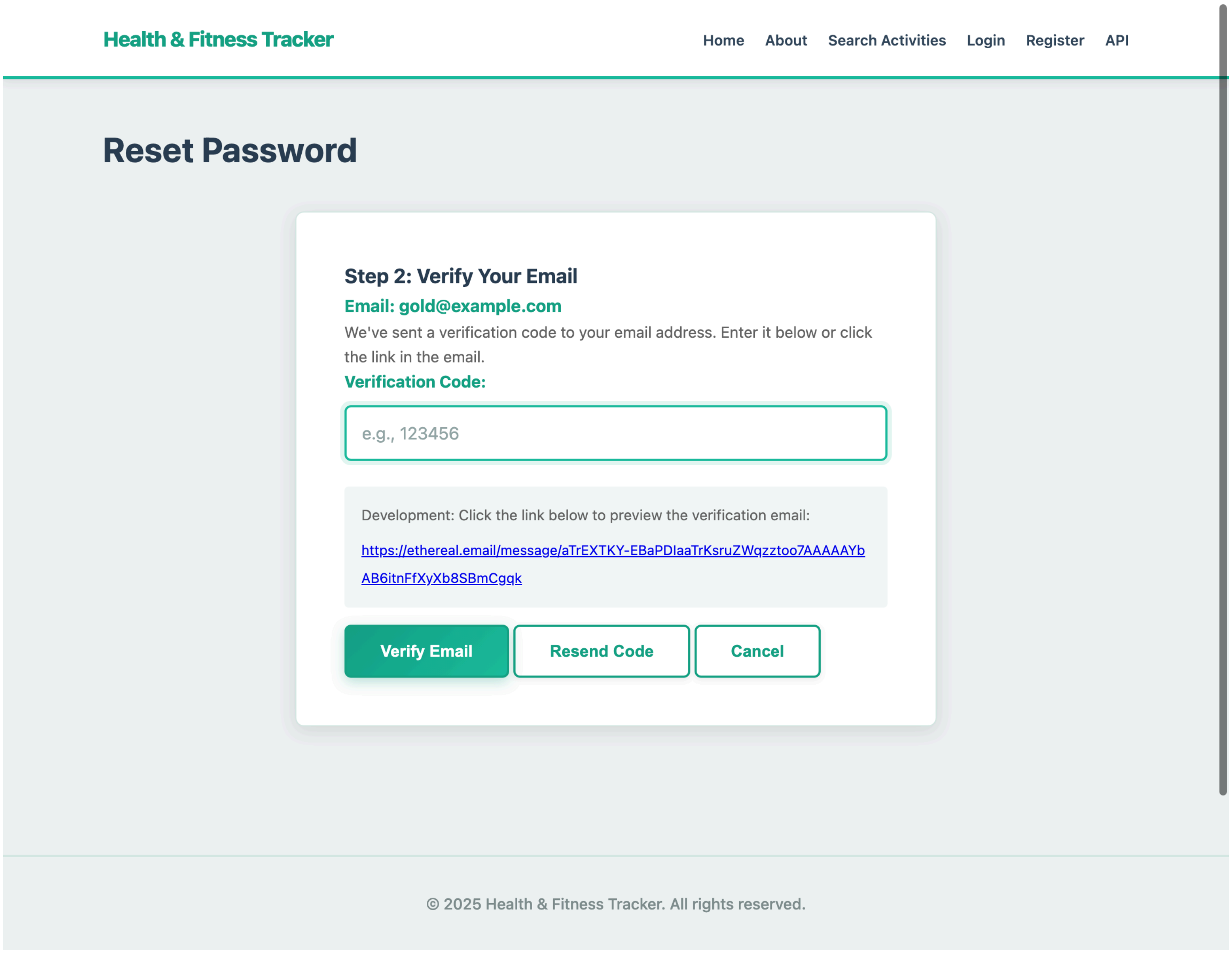
Health & Fitness Tracker

Reset Password

Step 1: Verify Your Identity

Username:

Email Address:



Reset Password

Step 2: Verify Your Email

Email: gold@example.com

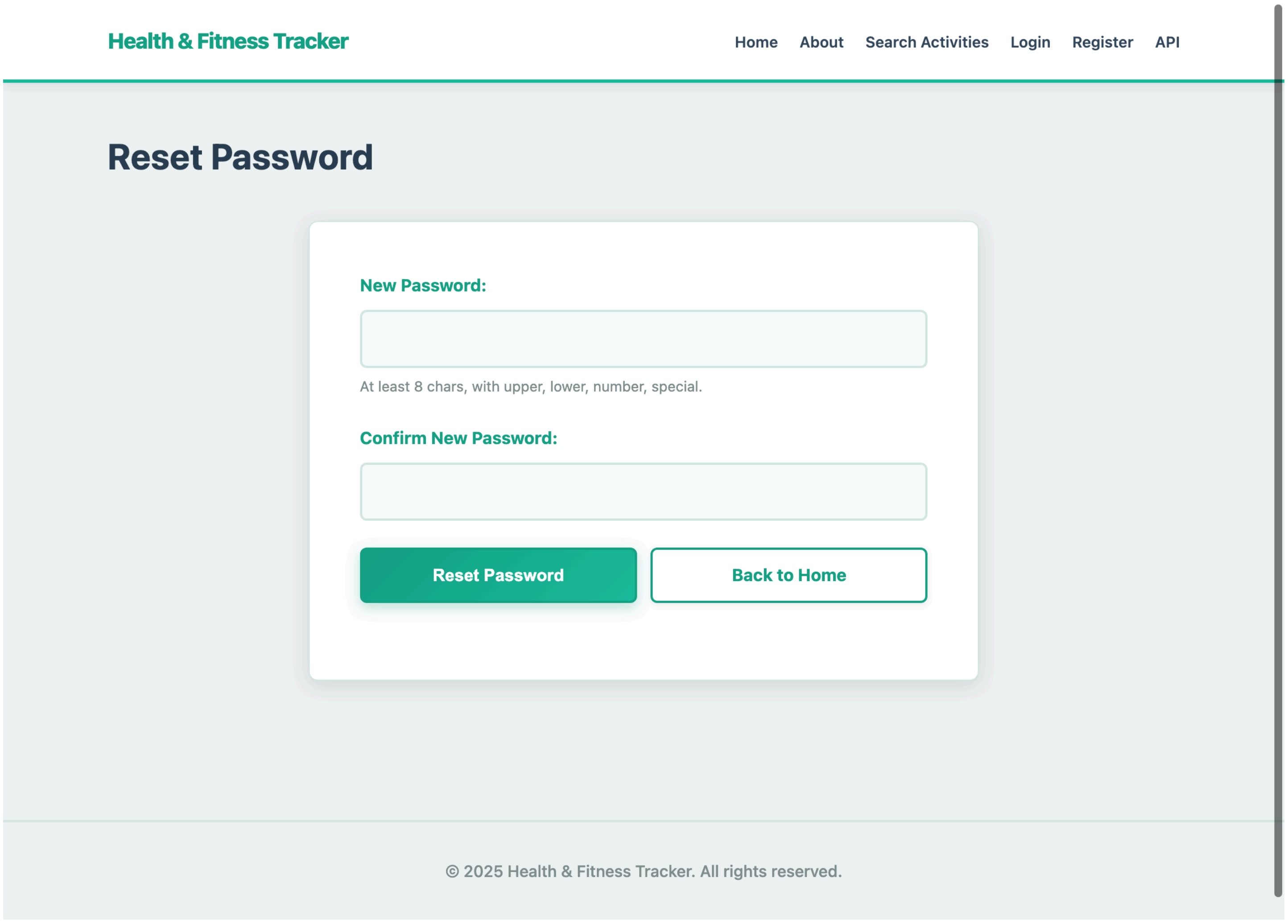
We've sent a verification code to your email address. Enter it below or click the link in the email.

Verification Code:

e.g., 123456

Development: Click the link below to preview the verification email:

https://ethereal.email/message/aTrEXTKY-EBaPDIaaTrKsruZWqzztoo7AAAAYbAB6itnFfXyXb8SBmCgqk


Health & Fitness Tracker

Home About Search Activities Login Register API

Reset Password

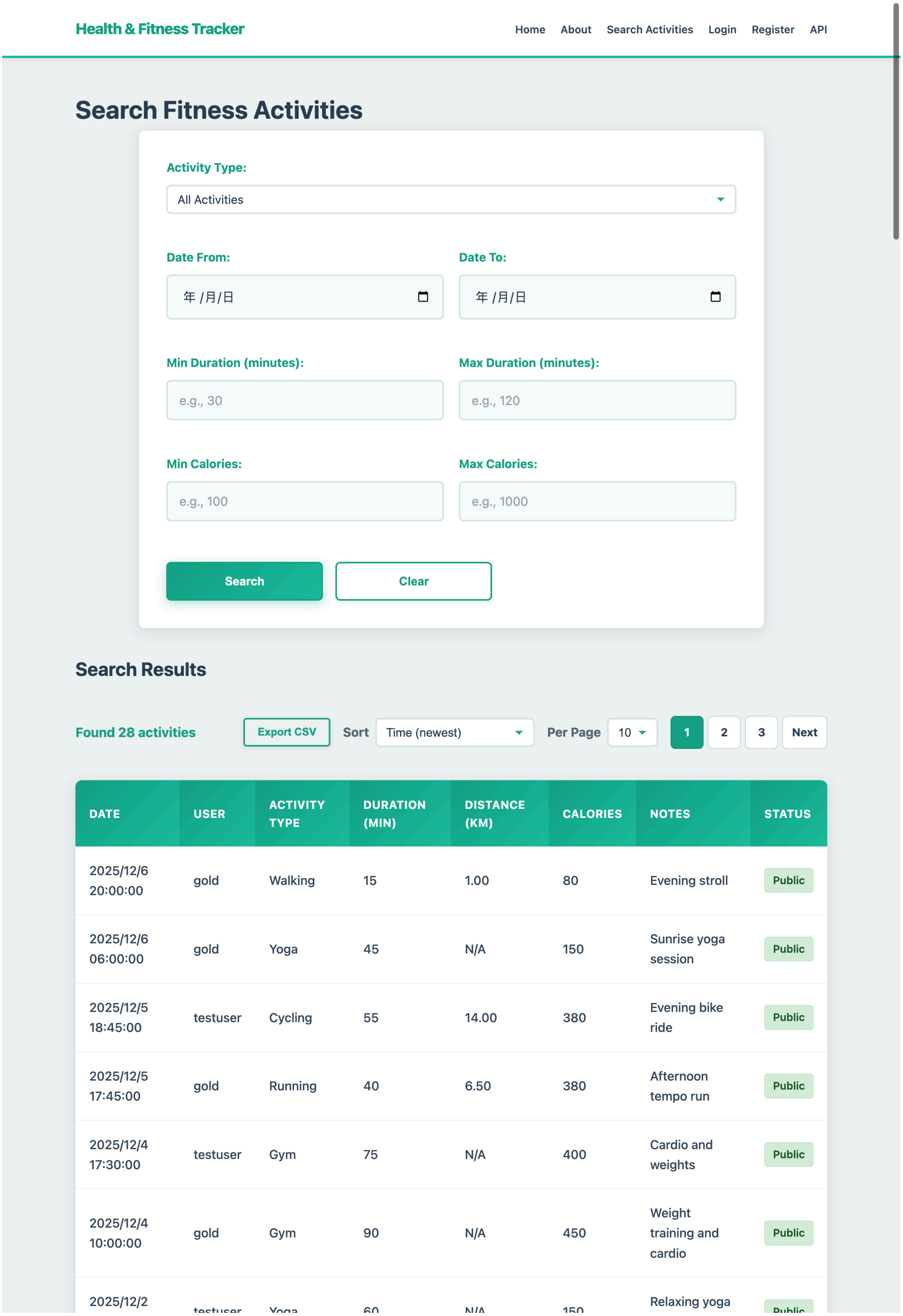
New Password:

At least 8 chars, with upper, lower, number, special.

Confirm New Password:

© 2025 Health & Fitness Tracker. All rights reserved.

Search activities page is open to all visitors without login. Supports filtering public activities by activity type, date range, duration range, and calorie range with sorting by date/calories/duration and 10/25/50/100 items per page. Filter conditions persist after page refresh and can be exported directly to CSV.


Health & Fitness Tracker

Home About Search Activities Login Register API

Search Fitness Activities

Activity Type: All Activities

Date From: 年 /月/日

Date To: 年 /月/日

Min Duration (minutes): e.g., 30

Max Duration (minutes): e.g., 120

Min Calories: e.g., 100

Max Calories: e.g., 1000

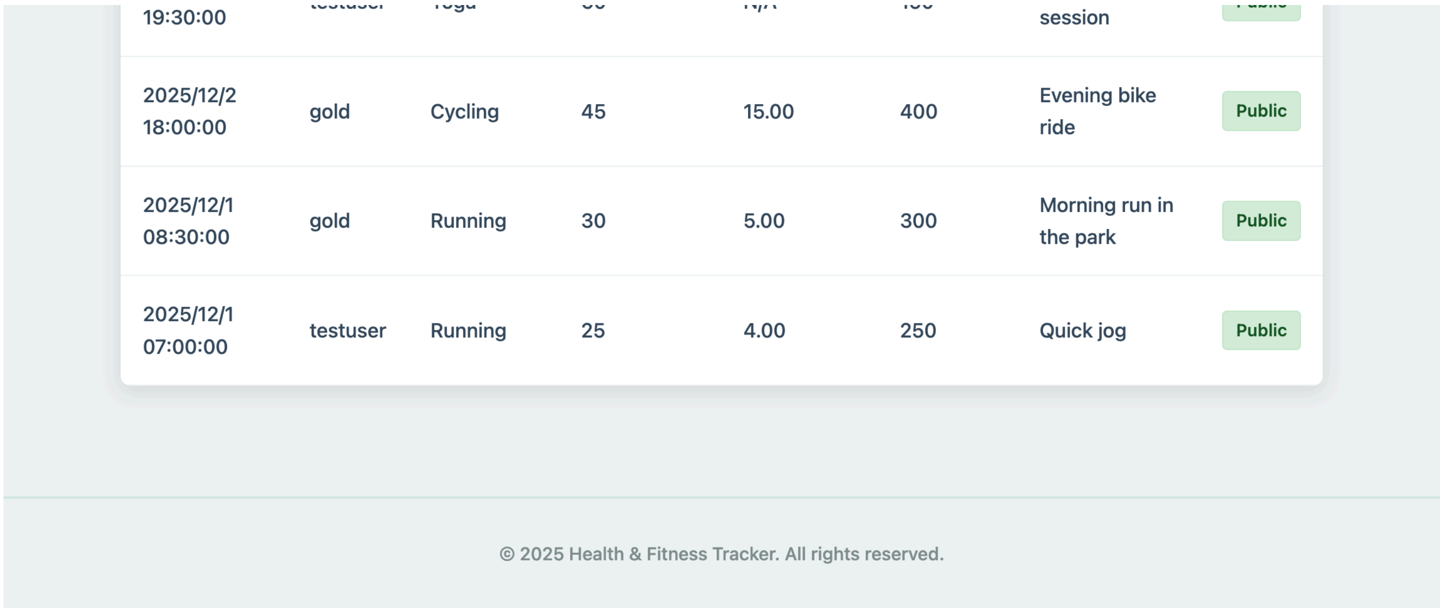
Search Clear

Search Results

Found 28 activities

Export CSV Sort Time (newest) Per Page 10 1 2 3 Next

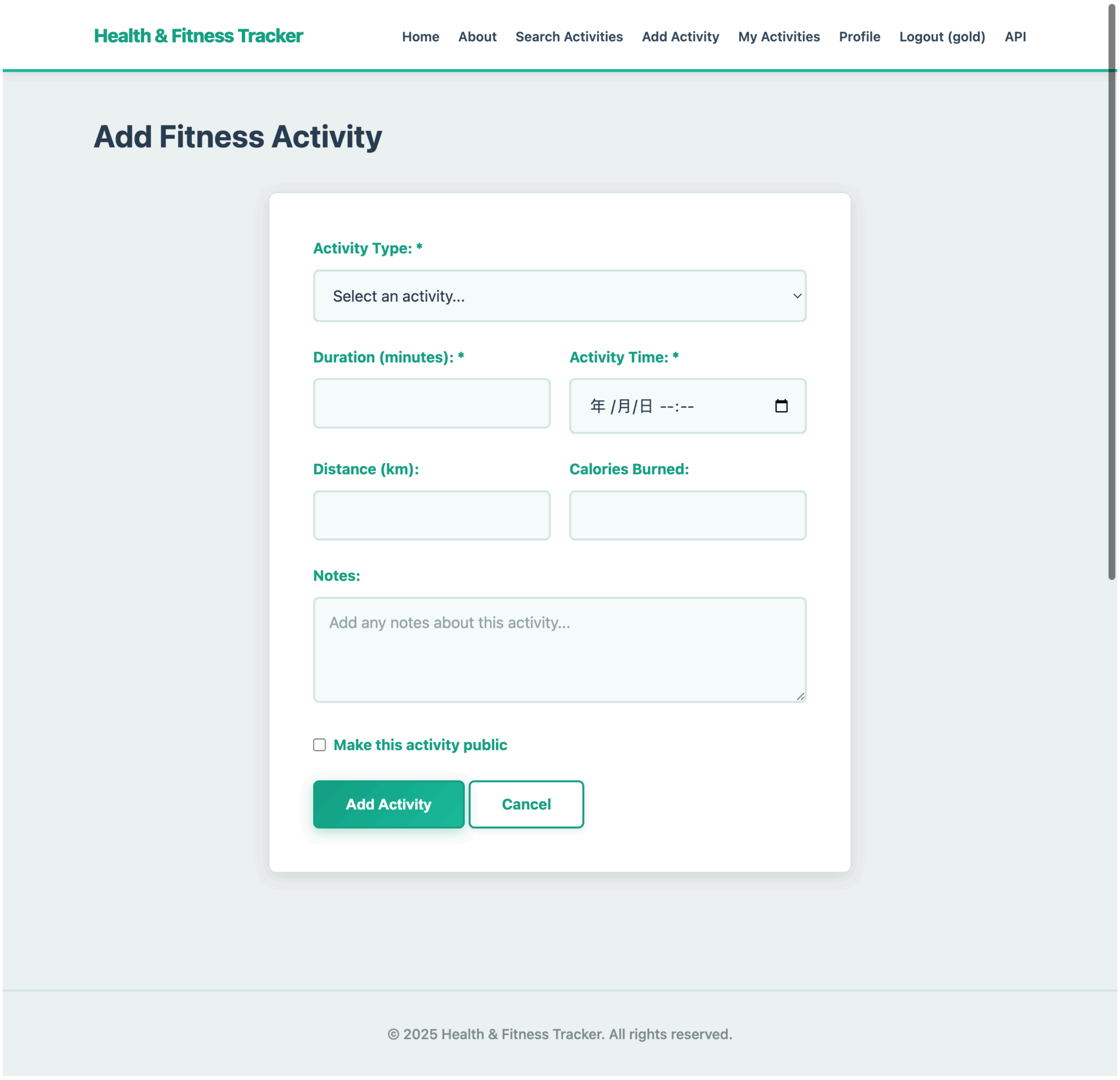
DATE	USER	ACTIVITY TYPE	DURATION (MIN)	DISTANCE (KM)	CALORIES	NOTES	STATUS
2025/12/6 20:00:00	gold	Walking	15	1.00	80	Evening stroll	Public
2025/12/6 06:00:00	gold	Yoga	45	N/A	150	Sunrise yoga session	Public
2025/12/5 18:45:00	testuser	Cycling	55	14.00	380	Evening bike ride	Public
2025/12/5 17:45:00	gold	Running	40	6.50	380	Afternoon tempo run	Public
2025/12/4 17:30:00	testuser	Gym	75	N/A	400	Cardio and weights	Public
2025/12/4 10:00:00	gold	Gym	90	N/A	450	Weight training and cardio	Public
2025/12/2	testuser	Yoga	60	N/A	150	Relaxing yoga	Public



19:30:00						session	
2025/12/2 18:00:00	gold	Cycling	45	15.00	400	Evening bike ride	Public
2025/12/1 08:30:00	gold	Running	30	5.00	300	Morning run in the park	Public
2025/12/1 07:00:00	testuser	Running	25	4.00	250	Quick jog	Public

© 2025 Health & Fitness Tracker. All rights reserved.

Users can create, edit, and delete their own activity records including type, time, duration, distance, calories, notes, and public visibility flag.


Health & Fitness Tracker

Home About Search Activities Add Activity My Activities Profile Logout (gold) API

Add Fitness Activity

Activity Type: *

Select an activity...

Duration (minutes): *

Activity Time: *

年 /月/日 --:--

Distance (km):

Calories Burned:

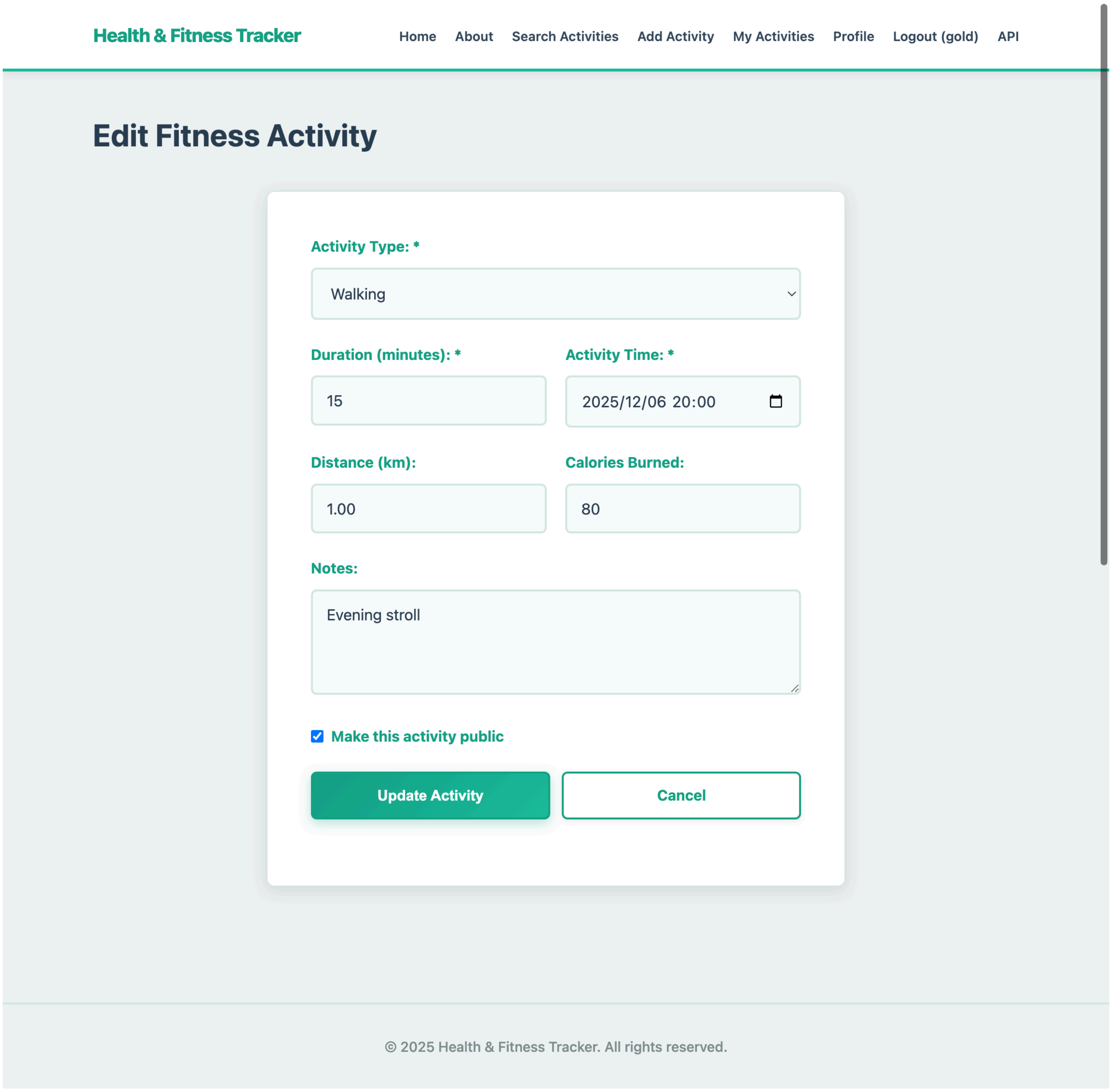
Notes:

Add any notes about this activity...

☐ Make this activity public

Add Activity Cancel

© 2025 Health & Fitness Tracker. All rights reserved.

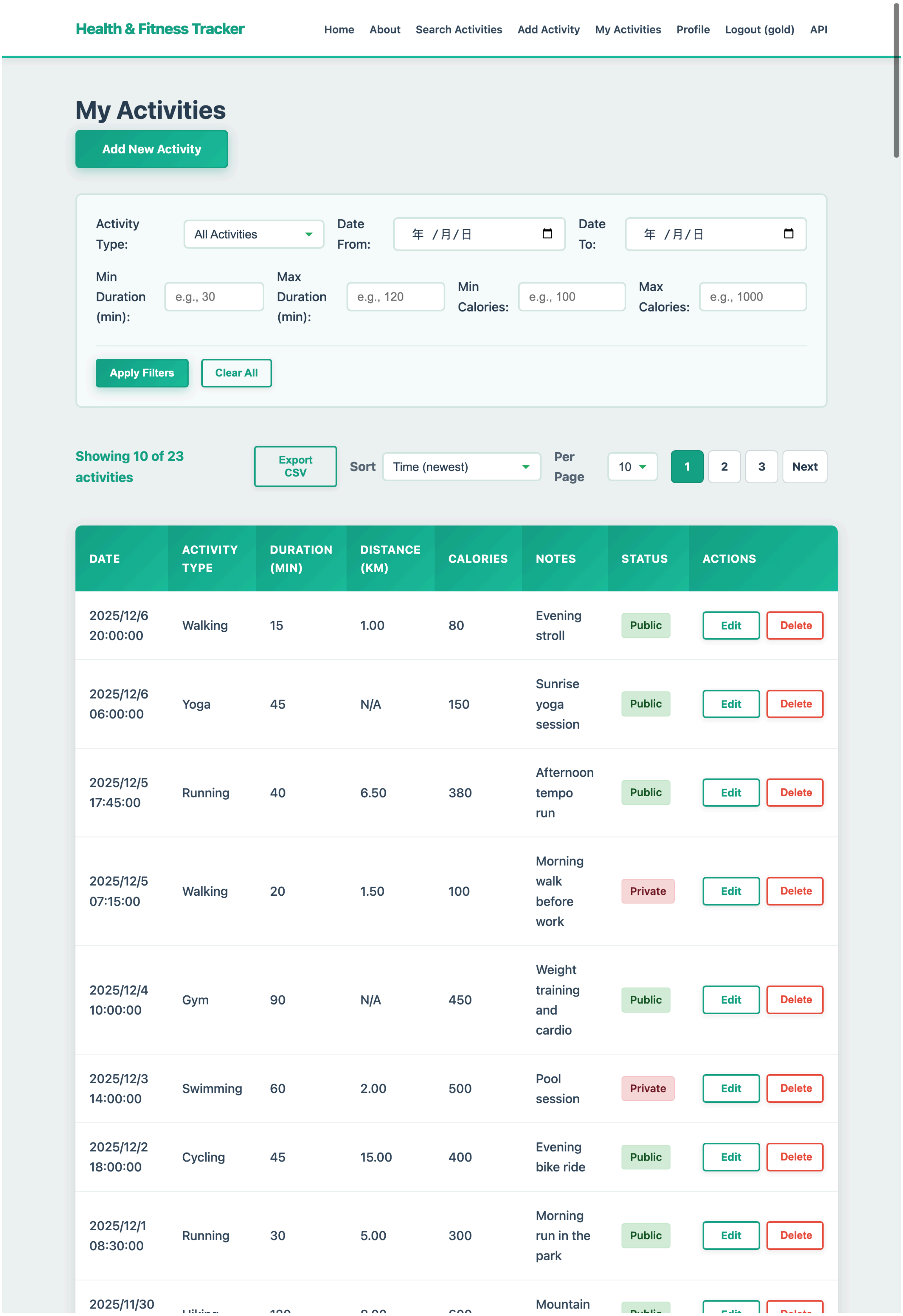


Edit Fitness Activity

Activity Type: *	Walking
Duration (minutes): *	15
Activity Time: *	2025/12/06 20:00
Distance (km):	1.00
Calories Burned:	80
Notes:	Evening stroll
✓ Make this activity public	

© 2025 Health & Fitness Tracker. All rights reserved.

On My Activities page, users can view personal activity list with same pagination, filtering, and sorting features as search page, with ability to edit or delete existing activities. Page aggregates current filtered data showing activity count, total duration, total distance, total calories, and max/min/average intensity, accompanied by doughnut chart (type distribution) and line chart (daily calories).

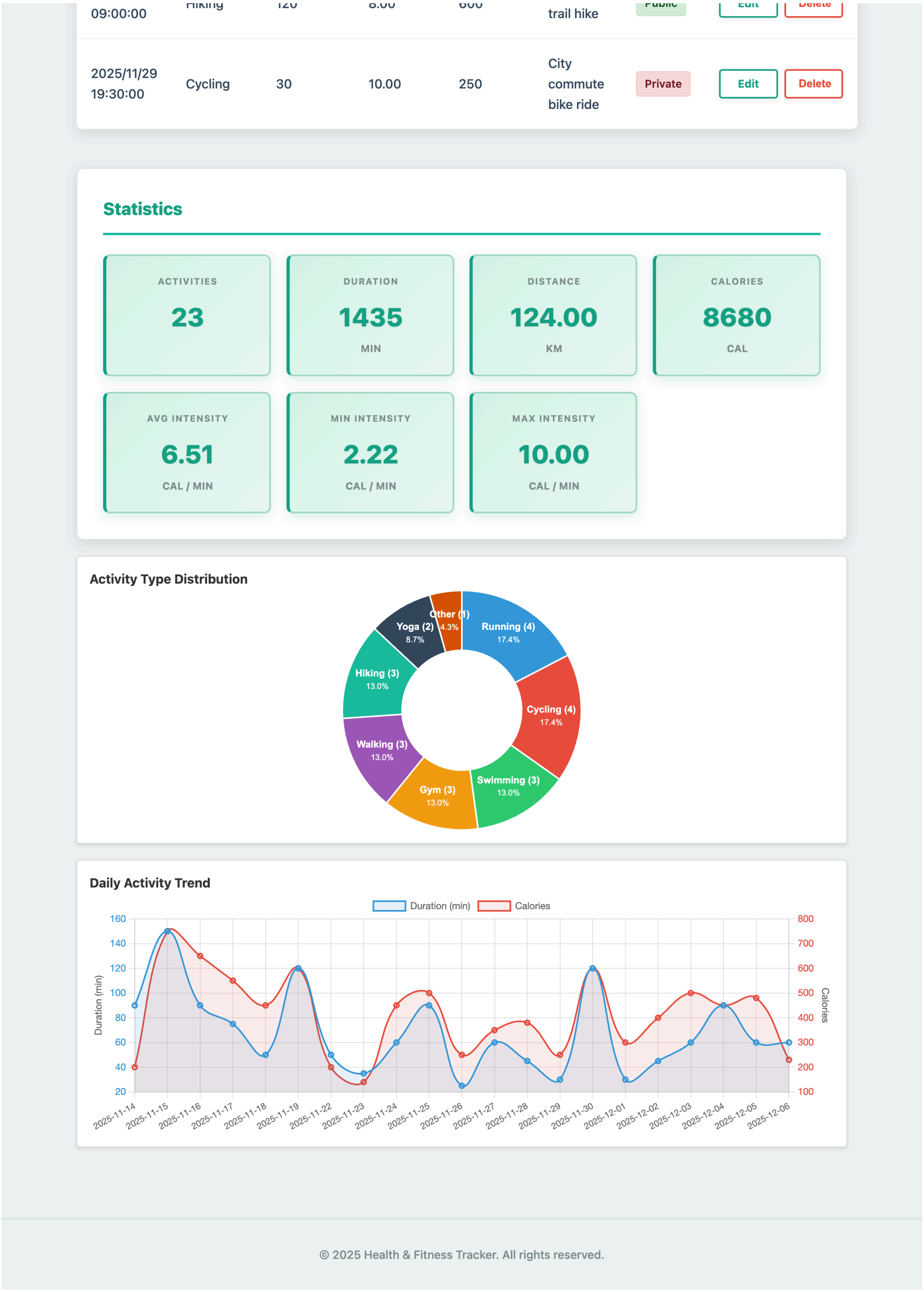


My Activities

Activity Type:	All Activities	Date From:	年 /月/日	Date To:	年 /月/日		
Min Duration (min):	e.g., 30	Max Duration (min):	e.g., 120	Min Calories:	e.g., 100	Max Calories:	e.g., 1000

Showing 10 of 23 activities

DATE	ACTIVITY TYPE	DURATION (MIN)	DISTANCE (KM)	CALORIES	NOTES	STATUS	ACTIONS
2025/12/6 20:00:00	Walking	15	1.00	80	Evening stroll	Public	Edit Delete
2025/12/6 06:00:00	Yoga	45	N/A	150	Sunrise yoga session	Public	Edit Delete
2025/12/5 17:45:00	Running	40	6.50	380	Afternoon tempo run	Public	Edit Delete
2025/12/5 07:15:00	Walking	20	1.50	100	Morning walk before work	Private	Edit Delete
2025/12/4 10:00:00	Gym	90	N/A	450	Weight training and cardio	Public	Edit Delete
2025/12/3 14:00:00	Swimming	60	2.00	500	Pool session	Private	Edit Delete
2025/12/2 18:00:00	Cycling	45	15.00	400	Evening bike ride	Public	Edit Delete
2025/12/1 08:30:00	Running	30	5.00	300	Morning run in the park	Public	Edit Delete
2025/11/30					Mountain		



09:00:00	Hiking	120	8.00	600	trail hike	Public	Edit Delete
2025/11/29 19:30:00	Cycling	30	10.00	250	City commute bike ride	Private	Edit Delete

Statistics

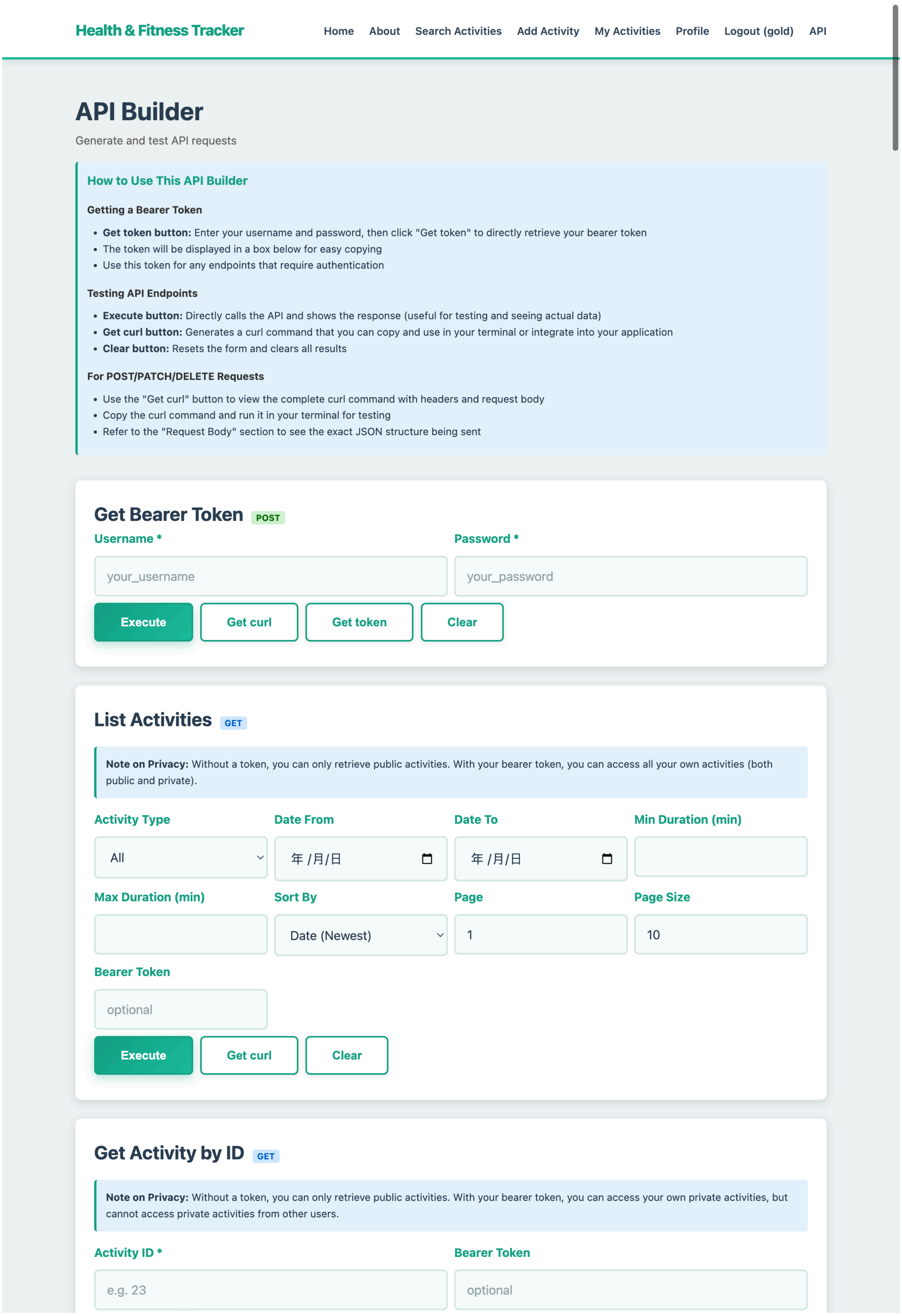
ACTIVITIES 23	DURATION 1435 MIN	DISTANCE 124.00 KM	CALORIES 8680 CAL
AVG INTENSITY 6.51 CAL / MIN	MIN INTENSITY 2.22 CAL / MIN	MAX INTENSITY 10.00 CAL / MIN	

Activity Type Distribution

Daily Activity Trend

© 2025 Health & Fitness Tracker. All rights reserved.

Self-service API documentation page at `/api-builder` lists all available endpoints for quick testing and integration.



API Builder

Generate and test API requests

How to Use This API Builder

Getting a Bearer Token

- **Get token button:** Enter your username and password, then click "Get token" to directly retrieve your bearer token
- The token will be displayed in a box below for easy copying
- Use this token for any endpoints that require authentication

Testing API Endpoints

- **Execute button:** Directly calls the API and shows the response (useful for testing and seeing actual data)
- **Get curl button:** Generates a curl command that you can copy and use in your terminal or integrate into your application
- **Clear button:** Resets the form and clears all results

For POST/PATCH/DELETE Requests

- Use the "Get curl" button to view the complete curl command with headers and request body
- Copy the curl command and run it in your terminal for testing
- Refer to the "Request Body" section to see the exact JSON structure being sent

Get Bearer Token POST

Username *	your_username
Password *	your_password

List Activities GET

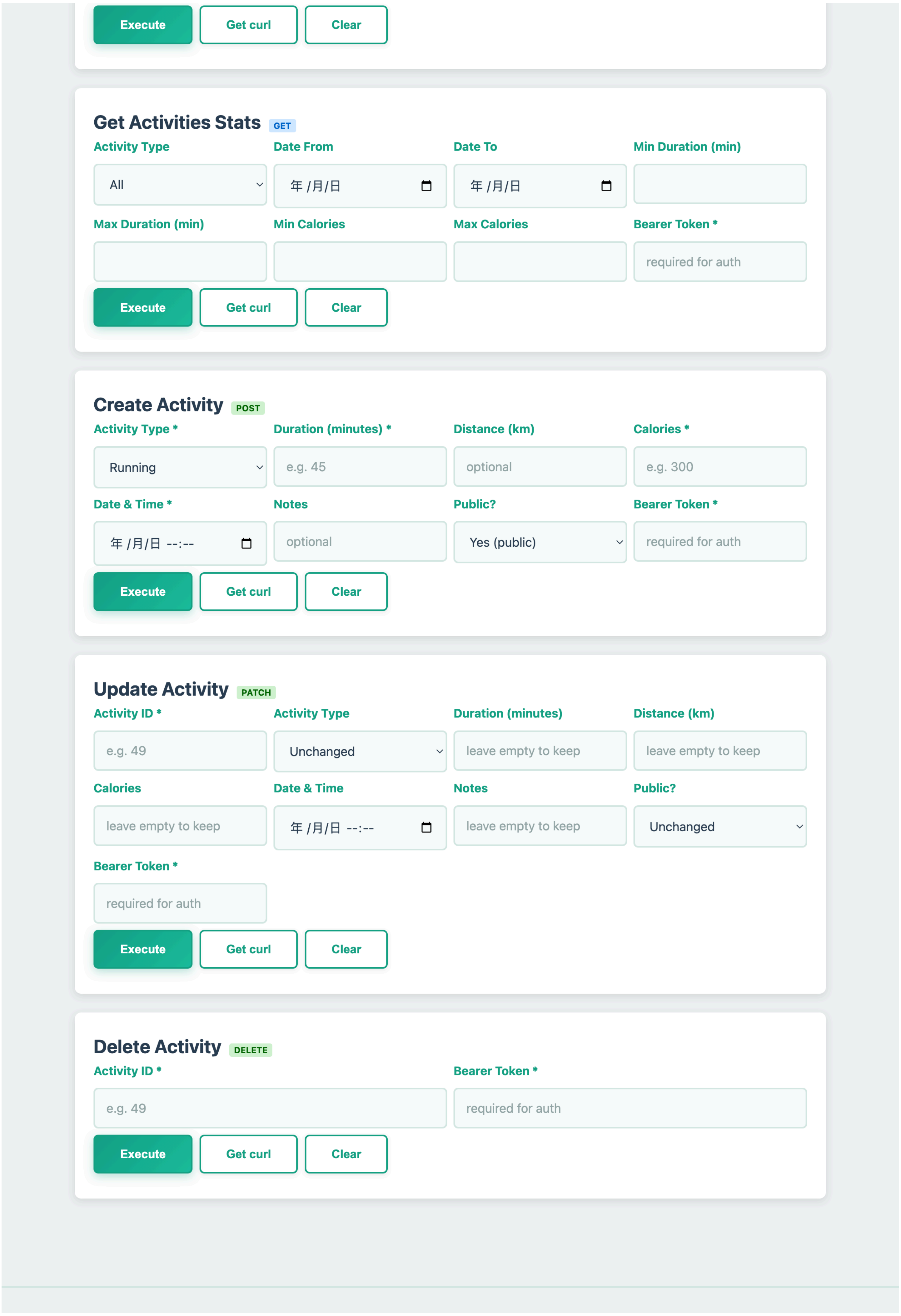
Note on Privacy: Without a token, you can only retrieve public activities. With your bearer token, you can access all your own activities (both public and private).

Activity Type	All
Date From	年 /月/日
Date To	年 /月/日
Min Duration (min)	
Max Duration (min)	
Sort By	Date (Newest)
Page	1
Page Size	10
Bearer Token	optional

Get Activity by ID GET

Note on Privacy: Without a token, you can only retrieve public activities. With your bearer token, you can access your own private activities, but cannot access private activities from other users.

Activity ID *	e.g. 23
Bearer Token	optional



Execute | Get curl | Clear

Get Activities Stats GET

Activity Type	Date From	Date To	Min Duration (min)
All	年 /月/日	年 /月/日	

Max Duration (min)	Min Calories	Max Calories	Bearer Token *
			required for auth

Execute | Get curl | Clear

Create Activity POST

Activity Type *	Duration (minutes) *	Distance (km)	Calories *
Running	e.g. 45	optional	e.g. 300

Date & Time *	Notes	Public?	Bearer Token *
年 /月/日 --:--	optional	Yes (public)	required for auth

Execute | Get curl | Clear

Update Activity PATCH

Activity ID *	Activity Type	Duration (minutes)	Distance (km)
e.g. 49	Unchanged	leave empty to keep	leave empty to keep

Calories	Date & Time	Notes	Public?
leave empty to keep	年 /月/日 --:--	leave empty to keep	Unchanged

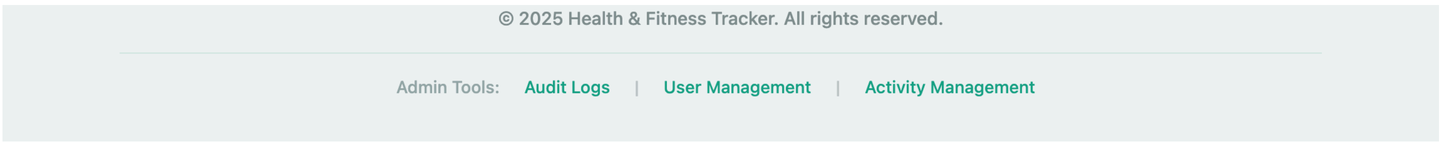
Bearer Token *
required for auth

Execute | Get curl | Clear

Delete Activity DELETE

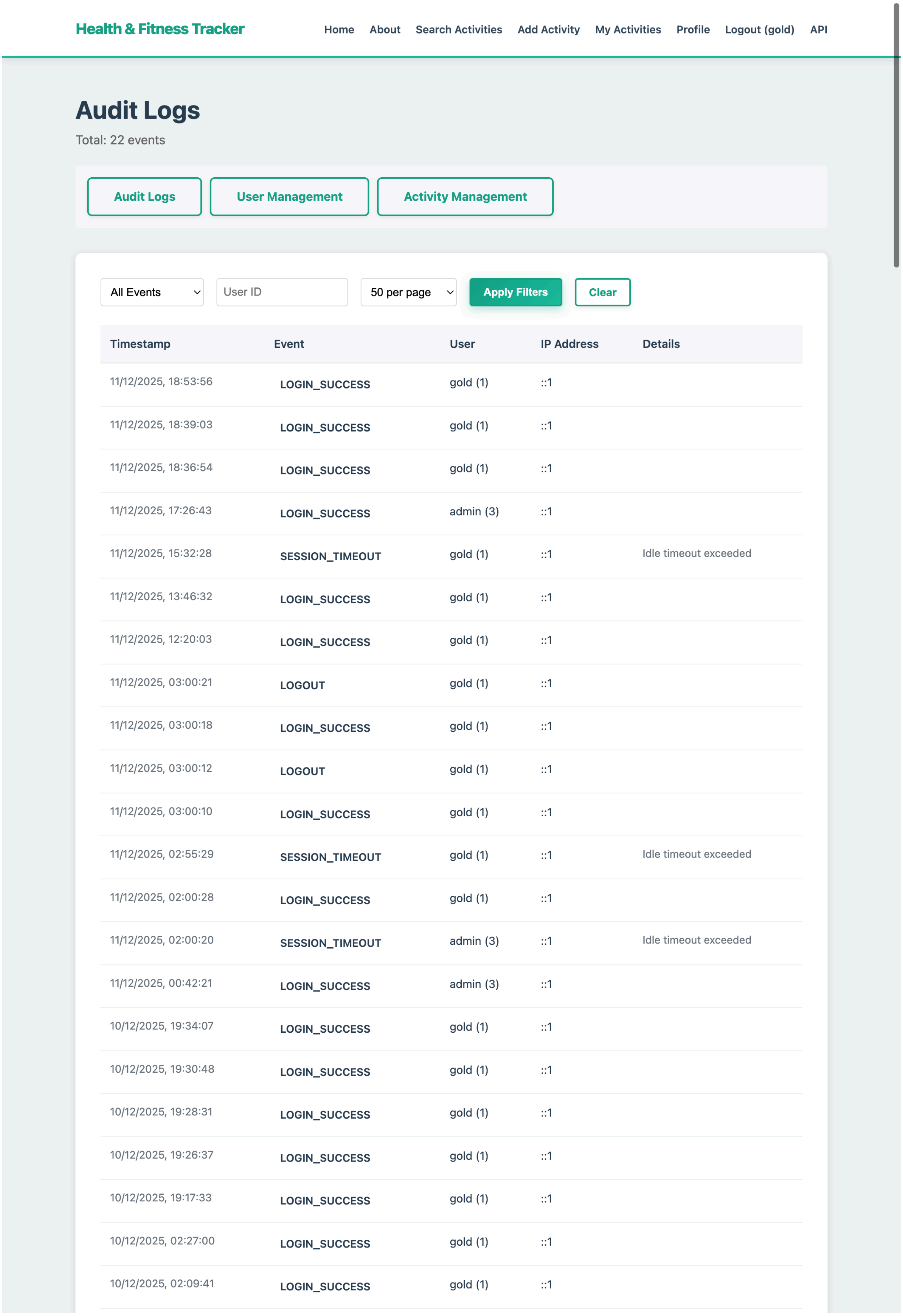
Activity ID *	Bearer Token *
e.g. 49	required for auth

Execute | Get curl | Clear


© 2025 Health & Fitness Tracker. All rights reserved.

Admin Tools: Audit Logs | User Management | Activity Management

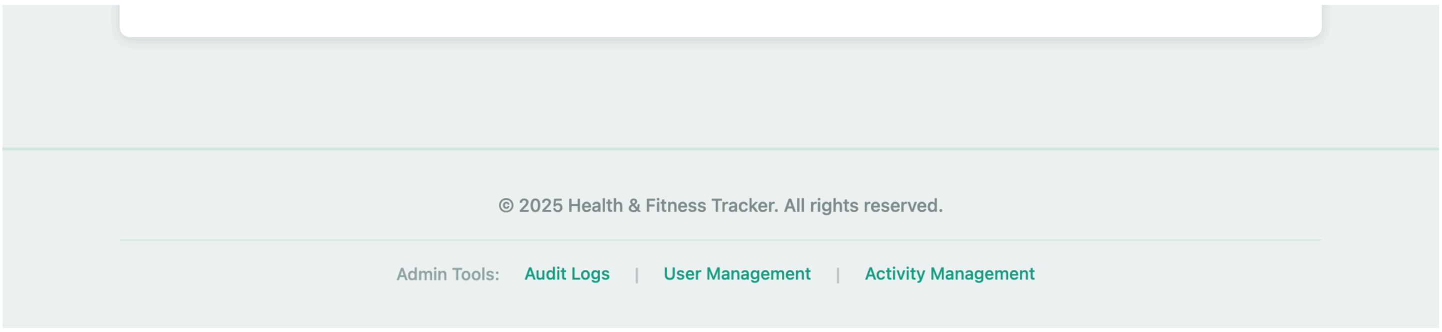
Admin Features: Administrators can access additional pages for system management. The audit logs page (`/admin/logs`) displays all user actions (login, registration, password change, email change, account deletion, activity operations) with filtering by event type and user ID, pagination (10-200 items per page), and sortable columns showing timestamp, user, event type, and affected resource. User management page (`/admin/users`) lists all registered users with deletion capability; activity management page (`/admin/activities`) displays all system activities across users with filtering and deletion tools, allowing admins to remove inappropriate public activities or manage content. All admin operations are logged in audit trail for accountability.



Audit Logs

Total: 22 events

Timestamp	Event	User	IP Address	Details
11/12/2025, 18:53:56	LOGIN_SUCCESS	gold (1)	::1	
11/12/2025, 18:39:03	LOGIN_SUCCESS	gold (1)	::1	
11/12/2025, 18:36:54	LOGIN_SUCCESS	gold (1)	::1	
11/12/2025, 17:26:43	LOGIN_SUCCESS	admin (3)	::1	
11/12/2025, 15:32:28	SESSION_TIMEOUT	gold (1)	::1	Idle timeout exceeded
11/12/2025, 13:46:32	LOGIN_SUCCESS	gold (1)	::1	
11/12/2025, 12:20:03	LOGIN_SUCCESS	gold (1)	::1	
11/12/2025, 03:00:21	LOGOUT	gold (1)	::1	
11/12/2025, 03:00:18	LOGIN_SUCCESS	gold (1)	::1	
11/12/2025, 03:00:12	LOGOUT	gold (1)	::1	
11/12/2025, 03:00:10	LOGIN_SUCCESS	gold (1)	::1	
11/12/2025, 02:55:29	SESSION_TIMEOUT	gold (1)	::1	Idle timeout exceeded
11/12/2025, 02:00:28	LOGIN_SUCCESS	gold (1)	::1	
11/12/2025, 02:00:20	SESSION_TIMEOUT	admin (3)	::1	Idle timeout exceeded
11/12/2025, 00:42:21	LOGIN_SUCCESS	admin (3)	::1	
10/12/2025, 19:34:07	LOGIN_SUCCESS	gold (1)	::1	
10/12/2025, 19:30:48	LOGIN_SUCCESS	gold (1)	::1	
10/12/2025, 19:28:31	LOGIN_SUCCESS	gold (1)	::1	
10/12/2025, 19:26:37	LOGIN_SUCCESS	gold (1)	::1	
10/12/2025, 19:17:33	LOGIN_SUCCESS	gold (1)	::1	
10/12/2025, 02:27:00	LOGIN_SUCCESS	gold (1)	::1	
10/12/2025, 02:09:41	LOGIN_SUCCESS	gold (1)	::1	


© 2025 Health & Fitness Tracker. All rights reserved.

Advanced Techniques

Security Baseline: Bcrypt password hashing, express-session for login state, CSRF tokens on all forms/AJAX, Helmet security headers. Comprehensive express-validator input validation on all submissions (registration, login, password/email changes, account operations, activity CRUD) with trim, type coercion, range checking, whitelist validation. Sensitive operations logged to `audit_logs`. Rate limiting on login/registration endpoints prevents brute force attacks.

```
// middleware/rate-limit.js - Login rate limiting
const loginLimiter = rateLimit({
    windowMs: 15 * 60 * 1000,  // 15 minutes
    max: 5,  // 5 attempts
    handler: (req, res) => {
        res.status(429).render('error', {
            message: 'Too many failed login attempts. Please try again later.'
        });
    }
});
```

Email Verification Workflows: Forgot password, email change, account deletion use Nodemailer to send verification codes (Ethereal in dev). Frontend JavaScript controls multi-step flow, backend session stores codes/temporary data. All critical operations audit-logged.

```
// utils/email-service.js - Send verification code via Nodemailer
async function sendPasswordResetEmail(to, verificationCode) {
    const transporter = nodemailer.createTransport({
        host: 'smtp.ethereal.email',
        port: 587,
        auth: { user: testAccount.user, pass: testAccount.pass }
    });

    await transporter.sendMail({
        from: '"Health Tracker" <noreply@healthtracker.com>',
        to: to,
        subject: 'Password Reset Verification',
        html: `<p>Your verification code: <code>${verificationCode}</code></p>`
    });
}
```

Search & Filter Engine: Shared filter builder `utils/filter-helper.js` constructs parameterized SQL queries (type, date range, duration, calories) preventing injection. GET parameters validated (ISO8601 dates, numeric ranges, whitelist for types/sort). Public search enforces `is_public=1`. Pagination, sorting, CSV export operate on filtered results with persistent URL query state.

```
// utils/filter-helper.js - Reusable filter builder
function addActivityFilters(baseWhere, baseParams, filters) {
    let whereClause = baseWhere;
    const params = [...baseParams];

    if (filters.activity_type && filters.activity_type !== 'all') {
        whereClause += (baseWhere ? ' AND ' : ' ') + 'activity_type = ?';
        params.push(filters.activity_type);
    }
    if (filters.date_from) {
        whereClause += (baseWhere ? ' AND ' : ' ') + 'activity_time >= ?';
        params.push(filters.date_from);
    }
    // Additional filters: date_to, duration_min/max, calories_min/max
    return { whereClause, params };
}
```

Activity CRUD Validation: Express-validator enforces required fields (type from predefined set, duration ≥1 min), numeric ranges (distance, calories), text sanitization (trim, max 1000 chars notes). Current user bound to writes/updates. `is_public` flag controls search visibility.

```
// routes/main.js - Activity input validation example
router.post('/add-activity', [
    body('activity_type')
        .notEmpty().withMessage('Activity type is required')
        .trim()
        .isIn(['Running', 'Cycling', 'Swimming', 'Gym', 'Yoga', 'Walking', 'Hiking', '
        .withMessage('Invalid activity type'),
    body('duration_minutes')
        .notEmpty().withMessage('Duration is required')
        .isInt({ min: 1 }).withMessage('Duration must be at least 1 minute'),
    body('distance_km')
        .optional({ checkFalsy: true })
        .isFloat({ min: 0 }).withMessage('Distance must be a positive number'),
    body('calories_burned')
        .optional({ checkFalsy: true })
        .isInt({ min: 0 }).withMessage('Calories must be a non-negative number'),
    body('notes')
        .optional({ checkFalsy: true })
        .trim()
        .isLength({ max: 1000 }).withMessage('Notes must be under 1000 characters')
], async (req, res) => { /* ... */ });
```

Stats & Chart Alignment: My Activities table, aggregated statistics (count, duration, distance, calories, intensity), and Chart.js visualizations share same filtered dataset. Chart endpoints `/internal/activities/charts/*` read URL query parameters for consistent aggregates. Session-protected internal endpoints (stats/charts) are implementation details, not public API.

```
// public/js/modules/activity/charts.js - Apply filters to charts
const urlParams = new URLSearchParams(window.location.search);
const filterQuery = urlParams.toString();
const typeUrl = '/internal/activities/charts/type-distribution' +
                (filterQuery ? '?' + filterQuery : '');
fetch(typeUrl).then(res => res.json()).then(renderChart);
```

REST API & Bearer Tokens: `/api-builder` provides interactive testing with Bearer auth (`POST /api/auth/token`), activity list (`GET /api/activities`), single query (`GET /api/activities/:id`), stats (`GET /api/activities/stats`), create/update/delete. Express-validator validates request bodies and route handlers validate query parameters (whitelist types, numeric ranges, ISO8601 dates, sort orders). Endpoints generate curl commands for testing. Independent rate limiting prevents abuse.

```
// routes/api.js — Bearer token issuance (simplified excerpt)
router.post('/auth/token', apiTokenLimiter, async (req, res) => {
    const { username, password } = req.body || {};
    if (!username || !password) {
        return res.status(400).json({ success: false, error: 'Username and password ar
    }
    const [rows] = await db.query('SELECT id, username, password FROM users WHERE user
    if (!rows?.length) return res.status(401).json({ success: false, error: 'Invalid c
    const user = rows[0];
    const ok = await bcrypt.compare(password, user.password);
    if (!ok) return res.status(401).json({ success: false, error: 'Invalid credentials

    const secret = process.env.API_TOKEN_SECRET || process.env.SESSION_SECRET || 'api-
    const token = createToken({ uid: user.id, username: user.username }, secret, 3600)
    return res.json({ success: true, token, token_type: 'Bearer', expires_in: 3600 });
});
```

Admin Tools: Admin login required (use `gold/smiths`). `/admin/logs` shows audit trail (login, registration, password/email changes, deletions, activity ops) with filtering (event type, user ID), validated query parameters, pagination (10-200 items), timestamp sorting. `/admin/users` lists all users, `/admin/activities` shows all activities with delete capability. `requireAdmin` middleware enforces access control. All admin deletions auto-logged to audit_logs with accountability.

```
// routes/main.js - Admin logs with query validation
router.get('/admin/logs', requireAdmin, async (req, res) => {
    const validEventTypes = ['REGISTER', 'LOGIN_SUCCESS', 'LOGIN_FAILURE', 'PASSWORD_CH
    const eventTypeValue = req.query.event_type && validEventTypes.includes(req.query.

    const userIdValue = req.query.user_id ? parseInt(req.query.user_id, 10) : null;
    if (userIdValue !== null && Number.isNaN(userIdValue)) {
        return res.render('logs', { logs: [], error: 'Invalid user_id parameter' });
    }

    // Build WHERE clause with validated parameters
    let whereClause = '';
    const params = [];
    if (eventTypeValue) {
        whereClause = 'WHERE event_type = ?';
        params.push(eventTypeValue);
    }
    if (userIdValue) {
        whereClause += whereClause ? ' AND user_id = ?' : 'WHERE user_id = ?';
        params.push(userIdValue);
    }

    // Fetch and display logs...
});
```

AI Declaration

GitHub Copilot was used as an assistance tool during development. In requirements analysis phase, AI helped organize feature lists and database design ideas; during coding phase provided code completion and syntax suggestions; in debugging phase assisted with identifying problem causes; during refactoring provided code optimization directions; in documentation writing polished expression. All architectural design, feature implementation, and technology selection completed independently by developer, with AI-generated content reviewed, tested, and modified before integration.