TreeHub API Documentation

This document provides comprehensive information about TreeHub's REST API endpoints.

🔗 Base URL

- **Development**: `http://localhost:3000/api`
- **Production**: `https://treehubusa.com/api`

🔐 Authentication

TreeHub uses NextAuth.js for authentication with session-based security.

Authentication Methods

1. **Email/Password**: Standard login with credentials
2. **OAuth**: Google and GitHub providers
3. **Magic Links**: Passwordless email authentication

Session Management

```
// Get current session
const session = await getServerSession(authOptions)

// Check authentication in API routes
if (!session) {
  return NextResponse.json({ error: 'Unauthorized' }, { status: 401 })
}
```

📋 API Endpoints

Authentication Endpoints

POST `/api/auth/signin`

Sign in with email and password.

Request Body:

```
{
  "email": "user@example.com",
  "password": "securepassword"
}
```

Response:

```
{
  "user": {
    "id": "user_id",
    "email": "user@example.com",
    "name": "John Doe",
    "role": "ADMIN"
  },
  "expires": "2024-01-01T00:00:00.000Z"
}
```

POST `/api/auth/signup`

Create a new user account.

Request Body:

```
{
  "email": "user@example.com",
  "password": "securepassword",
  "name": "John Doe",
  "company": "Tree Care Pro"
}
```

POST `/api/auth/signout`

Sign out the current user.

User Management

GET `/api/users/profile`

Get current user profile.

Response:

```
{
  "id": "user_id",
  "email": "user@example.com",
  "name": "John Doe",
  "company": "Tree Care Pro",
  "role": "ADMIN",
  "createdAt": "2024-01-01T00:00:00.000Z",
  "updatedAt": "2024-01-01T00:00:00.000Z"
}
```

PUT `/api/users/profile`

Update user profile.

Request Body:

```
{
  "name": "John Smith",
  "company": "Updated Tree Care",
  "phone": "+1234567890"
}
```

Client Management

GET `/api/clients`

Get all clients with pagination.

Query Parameters:
- `page` : Page number (default: 1)
- `limit` : Items per page (default: 10)
- `search` : Search term
- `sortBy` : Sort field (name, createdAt)
- `sortOrder` : Sort direction (asc, desc)

Response:


```
{
  "clients": [
    {
      "id": "client_id",
      "name": "ABC Property Management",
      "email": "contact@abcprop.com",
      "phone": "+1234567890",
      "address": "123 Main St, City, State 12345",
      "createdAt": "2024-01-01T00:00:00.000Z",
      "properties": 5,
      "totalJobs": 12
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 10,
    "total": 25,
    "pages": 3
  }
}
```


POST `/api/clients`

Create a new client.

Request Body:


```
{
  "name": "New Client",
  "email": "client@example.com",
  "phone": "+1234567890",
  "address": "123 Main St, City, State 12345",
  "notes": "Important client notes"
}
```


GET `/api/clients/[id]`

Get specific client details.

Response:

```
{
  "id": "client_id",
  "name": "ABC Property Management",
  "email": "contact@abcprop.com",
  "phone": "+1234567890",
  "address": "123 Main St, City, State 12345",
  "notes": "VIP client",
  "createdAt": "2024-01-01T00:00:00.000Z",
  "properties": [
    {
      "id": "property_id",
      "name": "Downtown Office Complex",
      "address": "456 Business Ave"
    }
  ],
  "jobs": [
    {
      "id": "job_id",
      "title": "Tree Pruning",
      "status": "COMPLETED",
      "scheduledDate": "2024-01-15T09:00:00.000Z"
    }
  ]
}
```


PUT `/api/clients/[id]`

Update client information.

DELETE `/api/clients/[id]`

Delete a client.

Property Management

GET `/api/properties`

Get all properties.

Query Parameters:

- `clientId` : Filter by client
- `page` : Page number
- `limit` : Items per page

Response:

```
{
  "properties": [
    {
      "id": "property_id",
      "name": "Downtown Office Complex",
      "address": "456 Business Ave, City, State",
      "clientId": "client_id",
      "client": {
        "name": "ABC Property Management"
      },
      "treeCount": 15,
      "lastInspection": "2024-01-01T00:00:00.000Z"
    }
  ]
}
```

POST `/api/properties`

Create a new property.

Request Body:

```
{
  "name": "New Property",
  "address": "789 Property Lane",
  "clientId": "client_id",
  "coordinates": {
    "lat": 40.7128,
    "lng": -74.0060
  },
  "notes": "Property specific notes"
}
```

GET `/api/properties/[id]`

Get property details with tree inventory.

Response:

```
{
  "id": "property_id",
  "name": "Downtown Office Complex",
  "address": "456 Business Ave",
  "coordinates": {
    "lat": 40.7128,
    "lng": -74.0060
  },
  "client": {
    "id": "client_id",
    "name": "ABC Property Management"
  },
  "trees": [
    {
      "id": "tree_id",
      "species": "Oak",
      "diameter": 24,
      "height": 45,
      "condition": "GOOD",
      "coordinates": {
        "lat": 40.7129,
        "lng": -74.0061
      }
    }
  ],
  "jobs": [
    {
      "id": "job_id",
      "title": "Annual Inspection",
      "status": "SCHEDULED"
    }
  ]
}
```


Tree Inventory

GET `/api/trees`

Get tree inventory.

Query Parameters:

- `propertyId` : Filter by property
- `species` : Filter by species
- `condition` : Filter by condition

POST `/api/trees`

Add a new tree to inventory.

Request Body:

```
{
  "propertyId": "property_id",
  "species": "Maple",
  "diameter": 18,
  "height": 35,
  "condition": "GOOD",
  "coordinates": {
    "lat": 40.7130,
    "lng": -74.0062
  },
  "notes": "Healthy specimen",
  "photos": ["photo1.jpg", "photo2.jpg"]
}
```

PUT `/api/trees/[id]`

Update tree information.

DELETE `/api/trees/[id]`

Remove tree from inventory.

Job Management

GET `/api/jobs`

Get all jobs with filtering.

Query Parameters:
- `status` : Filter by status (SCHEDULED, IN_PROGRESS, COMPLETED)
- `clientId` : Filter by client
- `propertyId` : Filter by property
- `startDate` : Filter by date range
- `endDate` : Filter by date range

Response:

```
{
  "jobs": [
    {
      "id": "job_id",
      "title": "Tree Pruning Service",
      "description": "Prune oak trees in front yard",
      "status": "SCHEDULED",
      "priority": "MEDIUM",
      "scheduledDate": "2024-01-15T09:00:00.000Z",
      "estimatedDuration": 240,
      "client": {
        "name": "ABC Property Management"
      },
      "property": {
        "name": "Downtown Office Complex"
      },
      "assignedCrew": [
        {
          "id": "user_id",
          "name": "John Smith"
        }
      ]
    }
  ]
}
```


POST `/api/jobs`

Create a new job.

Request Body:


```
{
  "title": "Emergency Tree Removal",
  "description": "Remove fallen tree blocking driveway",
  "clientId": "client_id",
  "propertyId": "property_id",
  "scheduledDate": "2024-01-20T08:00:00.000Z",
  "estimatedDuration": 480,
  "priority": "HIGH",
  "assignedCrew": ["user_id_1", "user_id_2"],
  "equipment": ["chainsaw", "chipper", "crane"]
}
```


GET `/api/jobs/[id]`

Get detailed job information.

PUT `/api/jobs/[id]`

Update job details.

PUT `/api/jobs/[id]/status`

Update job status.

Request Body:

```
{
  "status": "IN_PROGRESS",
  "notes": "Started work at 9:00 AM"
}
```

Estimate Management

GET `/api/estimates`

Get all estimates.

POST `/api/estimates`

Create a new estimate.

Request Body:

```
{
  "clientId": "client_id",
  "propertyId": "property_id",
  "title": "Tree Pruning Estimate",
  "description": "Comprehensive pruning service",
  "items": [
    {
      "description": "Tree pruning (per tree)",
      "quantity": 5,
      "unitPrice": 150.00,
      "total": 750.00
    },
    {
      "description": "Debris removal",
      "quantity": 1,
      "unitPrice": 200.00,
      "total": 200.00
    }
  ],
  "subtotal": 950.00,
  "tax": 76.00,
  "total": 1026.00,
  "validUntil": "2024-02-15T00:00:00.000Z"
}
```

GET `/api/estimates/[id]`

Get estimate details.

PUT `/api/estimates/[id]/status`

Update estimate status (DRAFT, SENT, ACCEPTED, REJECTED).

Invoice Management

GET `/api/invoices`

Get all invoices.

POST `/api/invoices`

Create invoice from completed job or accepted estimate.

GET `/api/invoices/[id]`

Get invoice details.

PUT `/api/invoices/[id]/payment`

Record payment for invoice.

Request Body:

```
{
  "amount": 1026.00,
  "method": "CHECK",
  "reference": "CHECK_12345",
  "paidAt": "2024-01-25T00:00:00.000Z"
}
```

File Upload

POST `/api/upload`

Upload files (images, documents).

Request: Multipart form data
- `file` : File to upload
- `type` : File type (image, document)
- `entityType` : Related entity (job, tree, property)
- `entityId` : Related entity ID

Response:

```
{
  "url": "https://a-z-animals.com/media/2023/03/shutterstock_1248290488.jpg",
  "filename": "tree-photo-001.jpg",
  "size": 1024000,
  "type": "image/jpeg"
}
```

Reports

GET `/api/reports/revenue`

Get revenue reports.

Query Parameters:
- `startDate` : Report start date
- `endDate` : Report end date
- `groupBy` : Group by (day, week, month, year)

GET `/api/reports/jobs`

Get job completion reports.

GET `/api/reports/clients`

Get client activity reports.

🔧 Error Handling

Error Response Format

```
{
  "error": "Error message",
  "code": "ERROR_CODE",
  "details": {
    "field": "Specific field error"
  }
}
```

HTTP Status Codes

- `200` : Success
- `201` : Created
- `400` : Bad Request
- `401` : Unauthorized
- `403` : Forbidden
- `404` : Not Found
- `422` : Validation Error
- `500` : Internal Server Error

Common Error Codes

- `UNAUTHORIZED` : User not authenticated
- `FORBIDDEN` : Insufficient permissions
- `VALIDATION_ERROR` : Input validation failed
- `NOT_FOUND` : Resource not found
- `DUPLICATE_ENTRY` : Resource already exists

📊 Rate Limiting

API endpoints are rate limited to prevent abuse:
- **Authenticated users**: 1000 requests per hour
- **Unauthenticated**: 100 requests per hour

Rate limit headers:
- `X-RateLimit-Limit` : Request limit
- `X-RateLimit-Remaining` : Remaining requests
- `X-RateLimit-Reset` : Reset timestamp

🔍 Filtering and Pagination

Query Parameters

- `page` : Page number (1-based)
- `limit` : Items per page (max 100)
- `sortBy` : Field to sort by
- `sortOrder` : `asc` or `desc`
- `search` : Search term

- `filter[field]`: Filter by field value

Response Format

```
{
  "data": [...],
  "pagination": {
    "page": 1,
    "limit": 10,
    "total": 100,
    "pages": 10
  }
}
```

🧪 Testing

API Testing

Use the included Postman collection or test with curl:

```
# Get all clients
curl -X GET "http://localhost:3000/api/clients" \
  -H "Cookie: next-auth.session-token=your-session-token"

# Create a new client
curl -X POST "http://localhost:3000/api/clients" \
  -H "Content-Type: application/json" \
  -H "Cookie: next-auth.session-token=your-session-token" \
  -d '{"name":"Test Client","email":"test@example.com"}'
```

Authentication Testing

```
# Sign in
curl -X POST "http://localhost:3000/api/auth/signin" \
  -H "Content-Type: application/json" \
  -d '{"email":"user@example.com","password":"password"}'
```

📚 SDK and Libraries

JavaScript/TypeScript SDK

```
import { TreeHubAPI } from '@treehub/sdk'

const api = new TreeHubAPI({
  baseURL: 'https://treehubusa.com/api',
  apiKey: 'your-api-key'
})

// Get clients
const clients = await api.clients.list()

// Create job
const job = await api.jobs.create({
  title: 'Tree Pruning',
  clientId: 'client_id'
})
```

📞 Support

For API support:
- **Documentation**: docs/API.md (API.md)
- **Issues**: GitHub Issues (https://github.com/yourusername/treehub-mobile-platform/issues)
- **Email**: api@treehubusa.com

API Version: 1.0.0
Last Updated: January 2025