Depannini API Documentation

Overview

The Depannini API provides backend services for a roadside assistance application. It allows clients to request assistance and assistants to respond to these requests. The system supports real-time communication through WebSockets and location tracking.

Base URL

```
https://api.depannini.com/
```

Authentication

Most endpoints require authentication using JWT (JSON Web Token).

Headers

```
Authorization: Bearer <your_token>
```

User Types

- **Client**: Users who request roadside assistance
- **Assistant**: Users who provide roadside assistance services

API Endpoints

Authentication

Register a New User

- **URL**: `/api/auth/register/`
- **Method**: `POST`
- **Authentication**: Not required
- **Request Body**:

```json
{
  "phone_number": "string",
  "email": "string",
  "name": "string",
  "password": "string",
  "password_confirm": "string",
  "user_type": "client|assistant",
  "service_type": "towing|repair",
  "vehicle_type": "string",
  "driving_license_cat": "c|b",
  "driving_license_num": "string",
  "driving_license_expiry": "dd:mm:yy",
  "vehicle_registration_num": "integer",
  "current_lat": "number",
  "current_lng": "number",
  "address": "string"
}
```

- **Notes**:
  - Fields required for assistants: `service_type`, `vehicle_type`, `current_lat`, `current_lng`, `driving_license_cat`, `driving_license_num`, `driving_license_expiry`, `vehicle_registration_num`
  - `driving_license_expiry` must be in the future

Email Login

- **URL**: `/api/auth/login/email/`
- **Method**: `POST`
- **Authentication**: Not required
- **Request Body**:

```json
{
  "email": "string",
  "password": "string"
}
```

- **Response**: Returns JWT access and refresh tokens

Phone Login

- **URL**: `/api/auth/login/phone/`

- **Method**: `POST`
- **Authentication**: Not required
- **Request Body**:

```json
{
  "phone_number": "string",
  "password": "string"
}
```

- **Response**: Returns JWT access and refresh tokens

Google Login

- **URL**: `/api/auth/login/google/`
- **Method**: `POST`
- **Authentication**: Not required
- **Request Body**:

```json
{
  "token": "string"
}
```

Email Verification

- **URL**: `/api/auth/verify-email/`
- **Method**: `POST`
- **Authentication**: Not required
- **Request Body**:

```json
{
  "email": "string",
  "code": "string"
}
```

Phone Verification

- **URL**: `/api/auth/verify-phone/`
- **Method**: `POST`
- **Authentication**: Not required

- **Request Body**:

```
json

{
  "phone_number": "string",
  "code": "string"
}
```

Password Reset Request

- **URL**: `/api/auth/password-reset/`
- **Method**: `POST`
- **Authentication**: Not required
- **Request Body**:

```
json

{
  "email": "string"
}
```

Password Reset Confirmation

- **URL**: `/api/auth/password-reset/confirm/`
- **Method**: `POST`
- **Authentication**: Not required
- **Request Body**:

```
json

{
  "email": "string",
  "code": "string",
  "new_password": "string",
  "new_password_confirm": "string"
}
```

Resend Verification Code

- **URL**: `/api/auth/resend-verification/`
- **Method**: `POST`
- **Authentication**: Not required

Refresh Token

- **URL**: `/api/auth/token/refresh/`
- **Method**: `POST`
- **Authentication**: Not required
- **Request Body**:

```json
{
  "refresh": "string"
}
```

- **Response**: Returns a new access token

Profile Management

Get User Profile

- **URL**: `/api/profile/user/`
- **Method**: `GET`
- **Authentication**: Required
- **Response**: User profile data

Get Assistant Profile

- **URL**: `/api/profile/assistant/`
- **Method**: `GET`
- **Authentication**: Required (Assistant only)
- **Response**: Assistant profile data

Update Location

- **URL**: `/api/profile/update-location/`
- **Method**: `POST`
- **Authentication**: Required
- **Request Body**:

```json
{
  "latitude": "number",
  "longitude": "number"
}
```

Update Assistant Status

- **URL**: `/api/profile/assistant-status/`
- **Method**: `POST`
- **Authentication**: Required (Assistant only)
- **Request Body**:

```json
{
  "is_active": "boolean"
}
```

List All Assistants

- **URL**: `/api/assistants/`
- **Method**: `GET`
- **Authentication**: Required
- **Response**: List of available assistants

Assistance Management

Request Assistance

- **URL**: `/api/assistance/request/`
- **Method**: `POST`
- **Authentication**: Required (Client only)
- **Request Body**:

```json
{
  "pickup": {
    "lat": "number",
    "lng": "number"
  },
  "dropoff": {
    "lat": "number",
    "lng": "number"
  }
}
```

- **Notes**:
  - The `dropoff` field is optional
  - This endpoint will notify nearby assistants via WebSockets

- **Response**:

```json
{
  "assistance": {
    "id": "integer",
    "status": "string",
    "client": {
      "id": "integer",
      "name": "string",
      "phone_number": "string"
    },
    "assistant": null,
    "pickupLocation": {
      "lat": "number",
      "lng": "number"
    },
    "dropoffLocation": {
      "lat": "number",
      "lng": "number"
    },
    "distance_km": "number",
    "total_price": "number",
    "createdAt": "string",
    "updatedAt": "string"
  },
  "nearby_assistants": [
    {
      "assistant_id": "integer",
      "distance": "number"
    }
  ]
}
```

View Assistance Requests

- **URL**: `/api/assistance/view/`
- **Method**: `GET`
- **Authentication**: Required
- **Response**: List of assistance requests associated with the user
- **Notes**: Returns different data depending on user type (client or assistant)

View Specific Assistance Request

- **URL**: `/api/assistance/view/<assistance_id>/`

- **Method**: `GET`
- **Authentication**: Required
- **Response**: Details of the specified assistance request

Accept Assistance Request

- **URL**: `/api/assistance/accept/<assistance_id>/`
- **Method**: `PATCH`
- **Authentication**: Required (Assistant only)
- **Notes**: Updates the assistance status to 'accepted' and assigns the assistant
- **Response**: Updated assistance details

Update Assistance Status

- **URL**: `/api/assistance/update/<assistance_id>/`
- **Method**: `PATCH`
- **Authentication**: Required
- **Request Body**:

```json
{
  "status": "ongoing|completed|canceled"
}
```

- **Notes**:
  - Clients can only cancel an assistance
  - Assistants can update to ongoing, completed, or canceled
  - Cannot update an assistance that is already completed or canceled

WebSockets

The application uses WebSockets for real-time communication between clients and assistants.

Assistant Channel

- **URL**: `ws://your-domain.com/ws/assistant/<assistant_id>/`
- **Purpose**: Assistants connect to this channel to receive new assistance requests

Events Received

- **new_assistance_request**: When a new assistance request is created near the assistant

```json
{
  "type": "new_assistance_request",
  "assistance_id": "integer",
  "pickup_location": {
    "lat": "number",
    "lng": "number"
  },
  "client": {
    "id": "integer",
    "name": "string"
  }
}
```

Assistance Channel

- **URL**: `ws://your-domain.com/ws/assistance/<assistance_id>/`
- **Purpose**: Both clients and assistants connect to this channel to receive updates about a specific assistance request

Events Received

- **status**: When assistance status changes

```json
{
  "type": "status",
  "status": "accepted|ongoing|completed|canceled",
  "assistant": {
    "id": "integer",
    "name": "string"
  },
  "client": {
    "id": "integer",
    "name": "string"
  }
}
```

- **location**: When assistant location updates

```json
{
  "type": "location",
  "lat": "number",
  "lng": "number",
  "user_id": "integer"
}
```

- **chat**: For chat messages

```json
{
  "type": "chat",
  "message": "string",
  "sender": "string"
}
```

Events Sent

- **location update**: Clients or assistants can send location updates

```json
{
  "type": "location",
  "lat": "number",
  "lng": "number",
  "user_id": "integer"
}
```

Data Models

User

- **Fields**:
  - `id`: Integer (Primary Key)
  - `email`: String (Optional, Unique)
  - `name`: String
  - `phone_number`: String (Unique)
  - `profile_photo`: Image (Optional)
  - `address`: String (Optional)
  - `current_lat`: Float (Optional)
  - `current_lng`: Float (Optional)

- `user_type`: String ('client' or 'assistant')
- `service_type`: String ('towing' or 'repair') (Required for assistants)
- `vehicle_type`: String (Required for assistants)
- `is_active_assistant`: Boolean
- `driving_license_cat`: String ('c' or 'b') (Required for assistants)
- `driving_license_num`: String (Required for assistants)
- `driving_license_expiry`: Date (Required for assistants)
- `vehicle_registration_num`: Integer (Required for assistants)

Assistance

- **Fields**:
  - `id`: Integer (Primary Key)
  - `client`: ForeignKey (User)
  - `assistant`: ForeignKey (User, Optional)
  - `pickup_lat`: Float
  - `pickup_lng`: Float
  - `dropoff_lat`: Float (Optional)
  - `dropoff_lng`: Float (Optional)
  - `created_at`: DateTime
  - `updated_at`: DateTime
  - `status`: String ('requested', 'accepted', 'ongoing', 'completed', 'canceled')
  - `rating`: Integer
  - `distance_km`: Float
  - `total_price`: Decimal

Error Handling

The API returns standard HTTP status codes:

- `200`: OK - Request successful
- `201`: Created - Resource created successfully
- `202`: Accepted - Request has been accepted for processing
- `400`: Bad Request - Invalid request parameters
- `401`: Unauthorized - Authentication required

- `403`: Forbidden - User doesn't have permission
- `404`: Not Found - Resource not found
- `409`: Conflict - Request conflicts with server state
- `500`: Internal Server Error - Server error occurred

Example Workflows

Client Requesting Assistance

1. Client logs in using `/api/auth/login/phone/`
2. Client requests assistance using `/api/assistance/request/`
3. Client connects to WebSocket `ws://your-domain.com/ws/assistance/<assistance_id>/`
4. Client receives updates about assistance status and assistant location

Assistant Accepting Assistance

1. Assistant logs in using `/api/auth/login/phone/`
2. Assistant updates status to active using `/api/profile/assistant-status/`
3. Assistant connects to WebSocket `ws://your-domain.com/ws/assistant/<assistant_id>/`
4. Assistant receives notification about new assistance request
5. Assistant accepts request using `/api/assistance/accept/<assistance_id>/`
6. Assistant connects to WebSocket `ws://your-domain.com/ws/assistance/<assistance_id>/`
7. Assistant updates status to ongoing when arriving using `/api/assistance/update/<assistance_id>/`
8. Assistant periodically sends location updates via WebSocket
9. Assistant updates status to completed after service using `/api/assistance/update/<assistance_id>/`