Tusenfryd Backend Technical Documentation

Overview

This document describes the REST API, data models, and project structure for the Tusenfryd backend system.

Arkitektur og datamodell

Systemarkitektur

- **server.js**: Initialiserer Express-app, konfigurerer middleware, ruter og databaseforbindelse.
- **routes/**: Definerer HTTP-endepunkter og videresender forespørsler til kontrollerne (`visitor.js`, `admin.js`, `api.js`).
- **controllers/**: Implementerer forretningslogikk for hver rute, håndterer inputvalidering, databehandling og svar.
- **models/**: Mongoose-skjemaer og metoder for datamodeller (`Attraction`, `Reservation`, `ParkSetting`, `User`).
- **middleware/**: Autentisering (`auth.js`), validering (`validation.js`), feilhåndtering (`errorHandler.js`), filopplasting (`upload.js`).
- **views/** og **public/**: EJS-malene og statiske ressurser (CSS, JavaScript, bilder).

Datamodell

- **Attraction**: Representerer en attraksjon med felter som navn, beskrivelse, status, kapasitet, gjeldende kø, åpningstider, kategori og beregnet ventetid (`methods.calculateWaitTime`).
- **Reservation**: Lagrer reservasjoner med referanse til `Attraction`, gjestedetaljer, antall gjester, automatisk køposisjon (`pre('save')`) og estimert tid, samt status (active, completed, cancelled, expired).
- **ParkSetting**: Nøkkel-/verdi-innstillinger for parken (f.eks. åpningstider, notifikasjoner, generelle og nødinnstillinger) med statiske metoder (`getSetting`, `setSetting`).
- **User**: Administrator- og lederkontoer med brukernavn, e-post, rolle, hash av passord (bcrypt) og siste påloggingstid.

Data Models

Attraction

- _id: ObjectId

- name: String (required)
- description: String (required)
- status: enum(`open`, `closed`, `maintenance`)
- waitTime: Number
- capacity: Number
- currentQueue: Number
- openingHours: { start: String, end: String }
- category: enum(`roller-coaster`, `family`, `thrill`, `water`, `children`)

Reservation

- _id: ObjectId
- attraction: ObjectId (ref Attraction)
- guestName: String
- guestEmail: String
- guestPhone: String
- numberOfGuests: Number
- queuePosition: Number
- reservationTime: Date
- estimatedTime: Date
- status: enum(`active`, `completed`, `cancelled`, `expired`)

ParkSetting

- _id: ObjectId
- key: String (unique)
- value: Mixed
- description: String
- category: enum(`hours`, `notification`, `general`, `emergency`)
- isActive: Boolean

User

- _id: ObjectId
- username: String
- password: String (hashed)
- email: String
- firstName: String
- lastName: String

- role: enum(`admin`, `manager`)
- isActive: Boolean
- lastLogin: Date

REST API Endpoints

Public API Endpoints

GET /api/attractions

Retrieve all attractions. Supports optional query parameters:

- `status` (string): `open`, `closed`, `maintenance`
- `category` (string): `roller-coaster`, `family`, `thrill`, `water`, `children`

Example Request:

```
GET http://localhost:3000/api/attractions?status=open&category=water
```

Example Response:

```json
{
  "success": true,
  "data": [
    {
      "_id": "60f8a1b2c3d4e5f6a7b8c9d0",
      "name": "Extreme Slide",
      "description": "A thrilling water slide...",
      "status": "open",
      "waitTime": 20,
      "capacity": 30,
      "currentQueue": 15,
      "openingHours": { "start": "10:00", "end": "22:00" },
      "category": "water"
    }
  ]
}
```

GET /api/attractions/:id

Retrieve details of a single attraction, including updated wait time and current queue count.

Example Request:

```
GET http://localhost:3000/api/attractions/60f8a1b2c3d4e5f6a7b8c9d0
```

Example Response:

```
{
  "success": true,
  "data": {
    "attraction": {
      "_id": "60f8a1b2c3d4e5f6a7b8c9d0",
      "name": "Extreme Slide",
      "description": "A thrilling water slide...",
      "status": "open",
      "waitTime": 18,
      "capacity": 30,
      "currentQueue": 12,
      "openingHours": { "start": "10:00", "end": "22:00" },
      "category": "water"
    },
    "queueCount": 12
  }
}
```

POST /api/reservations

Create a new reservation for an attraction. Body fields:

- `attraction` (ObjectId)
- `guestName` (string)
- `guestEmail` (string)
- `guestPhone` (string, optional)
- `numberOfGuests` (number)
- `notes` (string, optional)

Example Request:

```
POST http://localhost:3000/api/reservations
{
  "attraction": "60f8a1b2c3d4e5f6a7b8c9d0",
  "guestName": "Jane Doe",
  "guestEmail": "jane@example.com",
  "numberOfGuests": 2,
  "notes": "Prefer front seats"
}
```

Example Response (201 Created):

```
{
  "success": true,
  "data": {
    "_id": "70g9b2c3d4e5f6a7b8c9d0e1",
    "attraction": {
      "_id": "60f8a1b2c3d4e5f6a7b8c9d0",
      "name": "Extreme Slide",
      "status": "open",
      "waitTime": 18
    },
    "guestName": "Jane Doe",
    "guestEmail": "jane@example.com",
    "numberOfGuests": 2,
    "queuePosition": 3,
    "reservationTime": "2025-06-04T09:15:00.000Z",
    "estimatedTime": "2025-06-04T09:33:00.000Z",
    "status": "active"
  }
}
```

GET /api/reservations/check

Retrieve active reservations by guest email.

Query Parameters:

- `email` (string, required)

Example Request:

```
GET http://localhost:3000/api/reservations/check?email=jane@example.com
```

Example Response:

```
{
  "success": true,
  "data": [
    {
      "_id": "70g9b2c3d4e5f6a7b8c9d0e1",
      "attraction": { "_id": "60f8a1b2c3d4e5f6a7b8c9d0", "name": "Extreme Slide" },
      "guestName": "Jane Doe",
      "status": "active",
      "reservationTime": "2025-06-04T09:15:00.000Z",
      "estimatedTime": "2025-06-04T09:33:00.000Z"
    }
  ]
}
```

Admin API Endpoints (JWT required)

All endpoints below require `Authorization: Bearer <token>` header.

GET /api/admin/attractions

Fetch all attractions (admin view).

Example Response:

```
{
  "success": true,
  "data": [ /* same format as GET /api/attractions */ ]
}
```

POST /api/admin/attractions

Create a new attraction (admin only). Body fields same as Attraction model.

Example Request & Response follow POST /api/reservations style (201 Created).

PUT /api/admin/attractions/:id

Update an existing attraction. Body fields same as creation.

Example Request:

```
PUT http://localhost:3000/api/admin/attractions/60f8a1b2c3d4e5f6a7b8c9d0
{
  "status": "maintenance",
  "description": "Under scheduled maintenance"
}
```

Example Response:

```
{
  "success": true,
  "data": { /* updated attraction object */ }
}
```

DELETE /api/admin/attractions/:id

Delete an attraction and cancel its active reservations.

Example Response:

```
{
  "success": true,
  "message": "Attraction deleted successfully"
}
```

GET /api/admin/reservations

List all reservations (any status).

Example Response:

```
{
  "success": true,
  "data": [ /* array of reservation objects with attraction populated */ ]
}
```

PATCH /api/admin/reservations/:id

Update reservation status (`active` , `completed` , `cancelled`).

Example Request:

```
PATCH http://localhost:3000/api/admin/reservations/70g9b2c3d4e5f6a7b8c9d0e1
{
  "status": "completed"
}
```

Example Response:

```
{
  "success": true,
  "data": { /* updated reservation object */ }
}
```

Internal Admin API Endpoints (cookie auth required)

These endpoints are mounted under `/admin/api` and require valid session cookie (via `auth` middleware).

GET /admin/api/attraction-status

Fetch name, status, waitTime, and currentQueue for all attractions.

Example Response:

```json
[
  { "name": "Roller Coaster", "status": "open", "waitTime": 15,
"currentQueue": 10 },
  { "name": "Extreme Slide", "status": "open", "waitTime": 20,
"currentQueue": 15 }
]
```

GET /admin/api/reservation-counts

Fetch counts of reservations by status.

Example Response:

```json
{
  "active": 5,
  "pending": 2,
  "completed": 10,
  "cancelled": 1
}
```

Error Handling

All errors return JSON:

```json
{ "success": false, "error": "Error message" }
```

404 handler for API routes responds with status 404 and error JSON.

Project Structure

```
/controllers
  adminController.js
  apiController.js
  visitorController.js
/models
  Attraction.js
  Reservation.js
  ParkSetting.js
  User.js
/routes
  admin.js
  api.js
  visitor.js
/middleware
  errorHandler.js
  upload.js
/server.js
```