Tic-Tac-Toe Game API

A RESTful API for managing tic-tac-toe games with user authentication and statistics tracking. Built with Express.js, TypeScript, Prisma, and JWT authentication with postgresql.

Features

- Play Tic-Tac-Toe against a computer opponent by sending api request to a stateless python game engine.
- Game Status: Check current game state and winner
- Game Statistics: Track game outcomes (wins, losses, draws)
- Protected routes for authenticated users with JWT tokens in the middleware
- Persistent storage with Prisma ORM and Postgresql
- Schema validation using Zod
- Type-safe request/response handling
- Input sanitization and validation with zod schemas

API Endpoints

POST `/auth/signup`

recived user details with email and password to save to dabatase

```
{
    "name": "full name",
    "email": "test@gmail.com",
    "password": "1234566"
}
```

Reponse

```
{
    "message": "user created successfully"
}

```


POST `/auth/login`

recived user email and password and generate authorization token

```json
{
    "email": "test@gmail.com",
    "password": "1234566"
}
```

```

Reponse

````json
 {
    "message": "Login successful",
    "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI5YzNlM2RmNC1jMTExLTQ4N
jUtODE1Ni03OTBiY2U5M2VlYWYiLCJlbWFpbCI6ImNsaW50b25uZ290dGFAZ21haWwuY29tIiwi
bmFtZSI6IkNsaW50b24gTmdvdHRhIiwiaWF0IjoxNzU2NzA2Nzk3LCJleHAiOjE3NTY3MTAzOTd
9.zCzL4fD2NZVhbMfyEjyBuwpLpd7cZI3lk7prfYgiTEc",
    "user": {
        "user_id": "9c3e3df4-c111-4865-8156-790bce93eeaf",
        "email": "test@gmail.com",
        "name": "First Name"
    }
}

```


POST `/game/play`

Make a move in a tic-tac-toe game (requires authentication with bearer
token).

sample payload:

```json
{
    "current_player": "O",
    "state": [
        [-1, 1, 1],
        [0, 1, 0],
        [0, 0, -1]
    ]
}
```

sample response:

```

{
    "is_game_over": true,
    "board": [
        [-1, 1, 1],
        [0, 1, 0],
        [1, 0, -1]
    ],
    "winner": 1,
    "is_draw": false,
    "next_move": [2, 0],
```
```

```
    "status": "win"
}
```

GET `/game/stats`

get player game stats: wins, losses, draws (requires authentication with bearer token).

sample response:

```
{
    "message": "user stats fetched successully",
    "data": [
        {
            "id": "a0fa5a9c-dee3-4f6b-83fa-46c5cd03d6b1",
            "wins": 5,
            "losses": 1,
            "draws": 0,
            "userId": "9c3e3df4-c111-4865-8156-790bce93eeaf",
            "createdAt": "2025-08-31T14:59:56.785Z",
            "updatedAt": "2025-08-31T15:04:22.585Z"
        }
    ]
}
```

Board State Format

The game board is represented as a 3x3 array where:

- `0` = Empty cell
- `1` = X player
- `1` = O player

Installation

Clone the repository

```
git clone git@github.com:clintonngotta/Tic-Tac-Toe-Node-API.git
cd Tic-Tac-Toe-Node-API
npm install
```

Environment and DB Configuration

Create a .env file in the root directory and edit env.exmplae

```
# Generate Prisma client
npx prisma generate
```

```
# create tables
npx prisma migrate dev --name init

# Run database migrations
npx prisma migrate
```

Run the project

```
npm run dev
```

Authentication

This API uses JWT (JSON Web Tokens) for authentication. Protected routes require a valid JWT token in the Authorization header:

```
Authorization: Bearer <jwt-token>
```

Error handling and HTTP status codes:

Error Handling

The API returns consistent error responses:

```
{
    "message": "Error description",
    "error": "Detailed error information (in development)"
}
```

HTTP status codes

- `200` - Success
- `400` - Bad Request (validation errors)
- `401` - Unauthorized (missing/invalid token)
- `403` - Forbidden (expired token)
- `500` - Internal Server Error