Xander Poalses
Presents

META CROWN

Your all encompassing
Clash Royale Stat Tracker

stats | View Card stats | View Leaderboard | Nerf Miner | Zap is Love | Hog Riiiiiider! | View my stats | Vi

Let us Explore the live website before we take a technical deep dive

Live link:
Metacrown.co.za

Live Website

META
CROWN

What is META CROWN

MetaCrown is a comprehensive web application that revolutionizes how Clash Royale players build, analyze, and optimize their decks

Why MetaCrown?

- Bridge the gap between casual gameplay and competitive strategy
- Transform data into actionable insights for better gameplay
- Make deck optimization accessible to every CR player

Core Purpose:

- Deck Building: Intuitive drag-and-drop interface for creating winning strategies
- Player Analytics: Real-time stats integration with official Clash Royale API
- Performance Tracking: Monitor your gameplay evolution and deck effectiveness
- Community Insights: Learn from top players and share your strategies

META CROWN

Live Website

MetaCrown uses MySQL to create an store user account date, as well as user saved decks.

Users can Create, Read, Update and Delete decks

User passwords are hashed for user digital security through Bycrypt, using *bycrypt.hash()* for encryption and *bycrypt.compare()* for authentication.

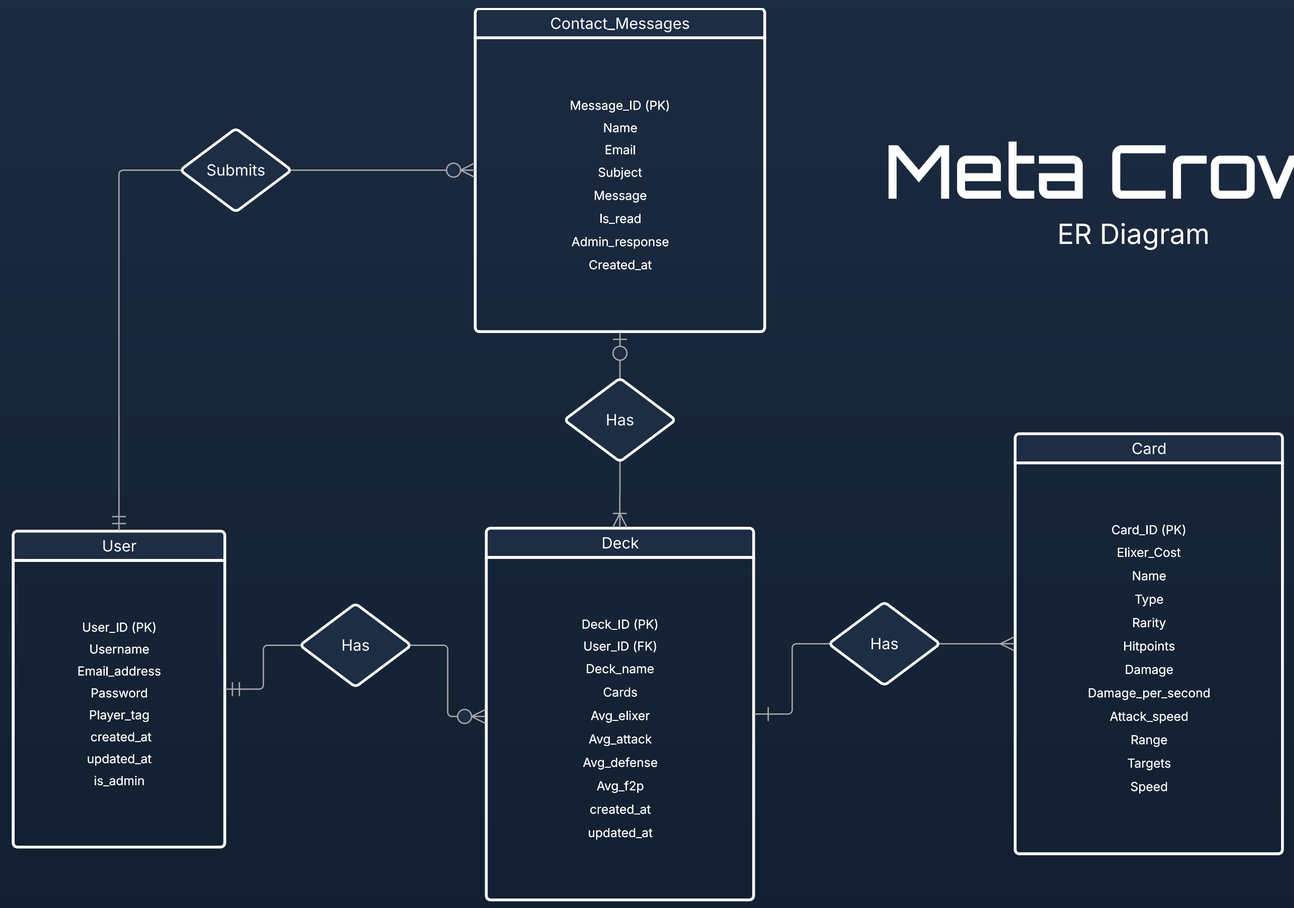
I selected bcrypt over standard hashing algorithms because it incorporates automatic salt generation, making the application resistant to rainbow table attacks through unique hash generation per password.

META CROWN

Live Website

Meta Crown

ER Diagram



Xander Poalses

241322

DV200

Summative Assessment

META CROWN

Live Website

CRUD Functionality

Create

User Registration:

Frontend: Multi-step form validation (email format, password strength)

Backend: POST /api/auth/signup

Security: bcrypt password hashing with salt generation

Database: Insert new user record with encrypted credentials

Deck Creation:

Frontend: Drag-and-drop deck builder interface

Backend: POST /api/decks

Validation: 8-card deck requirement, card availability checks

Database: Insert deck with foreign key relationships to user and cards

Read

User Authentication:

Frontend: Login form with credential validation

Backend: POST /api/auth/login

Process: Email lookup → bcrypt comparison → session establishment

Database: Query user table with indexed email lookups

Data Retrieval:

User Decks: GET /api/decks/user/:id - Fetch all user's decks

Player Stats: GET /api/player/:tag - Real-time CR API integration

Leaderboards: GET /api/leaderboards - Aggregated performance data

Update

Profile Management:

Frontend: Editable profile forms with real-time validation

Backend: PUT /api/users/:id

Security: Authorization checks ensure users modify only their data

Database: Conditional updates with optimistic locking

Deck Modifications:

Frontend: Live deck editor with card swapping and deck renaming

Backend: PUT /api/decks/:id

Validation: Deck composition rules and user ownership verification

Delete

Data Cleanup:

Deck Deletion: DELETE /api/decks/:id

Account Removal: DELETE /api/users/:id

Security: Authorization validation via user_id matching

Database: Cascade delete operations to maintain referential integrity

META CROWN

Live Website

API Intergration and External Data Flow

API: Official Clash Royale Public API

Authentication System

Bearer Token Authentication: Secure API access with official CR developer token

Header Implementation: *Authorization: Bearer <token>* for all API requests

Token Management: Secure storage and rotation for production environment

API Endpoints Utilized

Player Data: */v1/players/<playerTag>* - Real-time player statistics

Card Information: */v1/cards* - Complete card database with metadata

Clan Data: */v1/clans/<clanTag>* - Clan statistics and member information

Tournament Data: */v1/tournaments* - Active tournament information

Data flow

Frontend -

User requests player data

```
fetch('/api/player/2PP')
```

Backend -

Server acts as secure proxy to CR API

```
app.get('/api/player/:tag', async (req, res) => {
  const response = await fetch(`https://api.clashroyale.com/v1/players/%23${tag}`, {
    headers: { 'Authorization': `Bearer ${CR_API_TOKEN}` }
  });
});
```

META CROWN

Live Website

Frontend Architecture

React 19.1.1

Component Hierarchy

App.js: Root component with routing configuration

Pages: Dashboard, DeckCentre, Profile, Leaderboard (feature modules)

Components: NavBar, Footer, DeckComponent (reusable UI elements)

Utilities: API handlers, authentication helpers, validation functions

Styling and Responsiveness

CSS Modules: Scoped styling for component isolation

Responsive Design: Breakpoints used to accommodate for modern screen sizes

Smart Mobile Detection: Orientation-based overlay system for mobile detection

State Management

Local State: useState hooks for component-specific data

Session Storage: User authentication persistence

Context API: Global user state and authentication status

Effect Hooks: useEffect for API calls and lifecycle management

META CROWN

Live Website

Backend Architecture

Node.js Express 4.21.2

RESTful API Design

```
// Route Structure
app.use('/api/auth', authRoutes);       // Authentication endpoints
app.use('/api/users', userRoutes);      // User CRUD operations
app.use('/api/decks', deckRoutes);      // Deck management
app.use('/api/cards', cardRoutes);      // Card data endpoints
app.use('/api/player', playerRoutes);  // CR API proxy
```

META CROWN

Live Website

Request Flow

React Component

Fetch API

Express Route

Sequelize Query

MySQL Database

CR API Call

Response Processing

JSON Return

META CROWN

Live Website

Deployment

Cpanel

Why cPanel Hosting:

Simplicity: User-friendly interface for file management and deployment

Node.js Support: Built-in Node.js 22.18.0 runtime environment

MySQL Integration: Direct database management with phpMyAdmin

Custom Domain: Professional branding with SSL certificate support

Cost-Effective: Affordable hosting solution for full-stack applications

Security Considerations:

Hardcoded Credentials: Secure production-specific database configuration

CORS Configuration: app.use(cors()) for cross-origin requests

SSL Certificate: HTTPS enabled for secure data transmission

API Token Security: Environment-based Clash Royale API key management

META CROWN

Live Website

Deployment

Cpanel Comparrison vs AWS/Azure/Google Cloud Platform (GCP)

Criteria	cPanel Node.js	AWS/Azure/GCP	MetaCrown Choice
🚀 **Initial Setup**	Immediate deployment	Complex configuration required	✅ **Rapid deployment advantage**
📈 **Scaling Options**	Limited vertical scaling	Horizontal + Vertical scaling	✅ **Current traffic sufficient**
💰 **Cost Structure**	Fixed monthly fee	Usage-based pricing	✅ **Predictable budgeting**
🛠️ **DevOps Complexity**	Minimal management	High technical overhead	✅ **Small team advantage**
🌐 **Domain Integration**	Native DNS management	External DNS configuration	✅ **Existing registration**
🔧 **Maintenance**	Web-based interface	Command-line/API management	✅ **User-friendly management**

META CROWN

Live Website

SEO Web Optimisation

Professional Web Presence:

Google Analytics Integration: Real-time user behavior tracking

Search Console Verification: Indexed sitemap and performance monitoring

Open Graph Meta Tags: Rich social media link previews

Structured Data: Search engine optimization for gaming content

```
<!-- Open Graph / Social Media Meta Tags -->
<meta property="og:type" content="website" />
<meta property="og:title" content="MetaCrown - Clash Royale Deck Builder & Analytics" />    "Royale": Unknown word.
<meta property="og:description" content="Build and analyze Clash Royale decks with real-time player stats and competitive insights." />
<meta property="og:url" content="https://metacrown.co.za" />
<meta property="og:site_name" content="MetaCrown" />
<meta property="og:image" content="https://metacrown.co.za/crown.png" />      You, yesterday • imporved sitemaps and meta tags
```

META CROWN

Live Website

Technical Challenges

Challenge #1 Smart Mobile Detection System

Challenge: Traditional pixel-based breakpoints failed on modern high-resolution devices

Solution: Orientation-based detection for true mobile usage patterns

Challenge #2 User IP Whitelisting

Challenge: The Clash Royale API requires you to whitelist your IP address which is an issue because I cannot whitelist every user's IP address

Solution: I had to ask my hosting company (SA Gateway) to create a shared IP Address for me so I could whitelist the shared IP address

Challenge #3 API Rate Limiting & Error Handling

Challenge: Clash Royale API has strict rate limits (10 requests/second) and can return various error states

Solution: Implemented intelligent caching, request queuing, and graceful degradation with fallback data

Challenge #4 Production vs Development Environment Configuration

Challenge: Managing different database connections, API endpoints, and build configurations

Solution: Separate server files (server.js vs production-server.js) with environment-specific hardcoded credentials

META CROWN

Live Website

Thank You

Live Website

stats View Card stats View Leaderboard Nerf Miner Zap is Love Hog Riiiiiider! View my stats