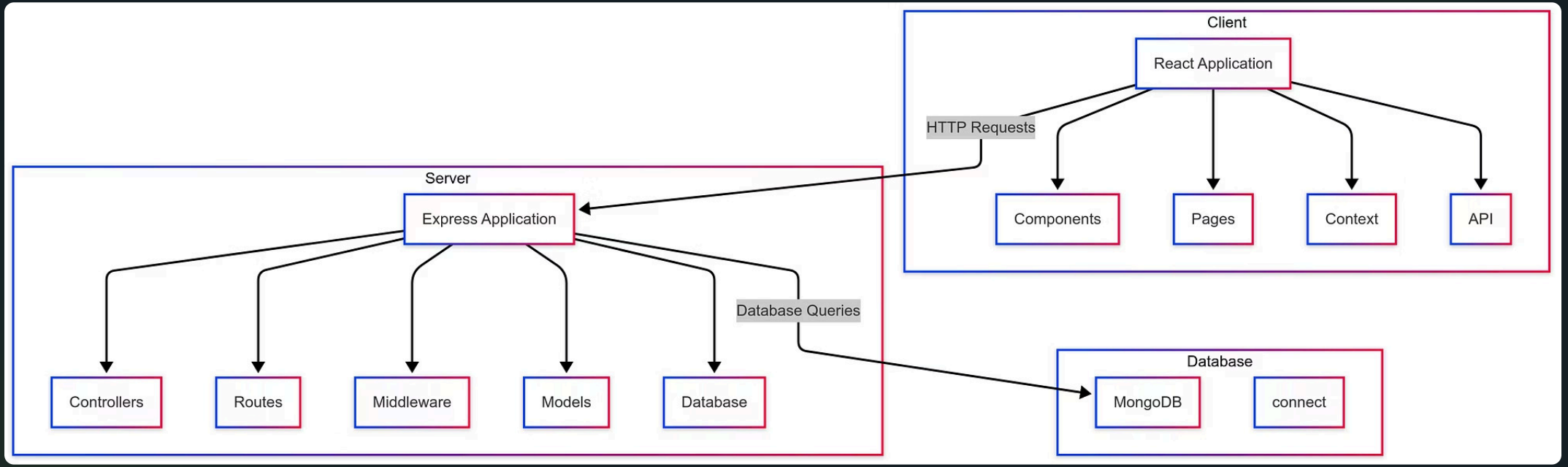
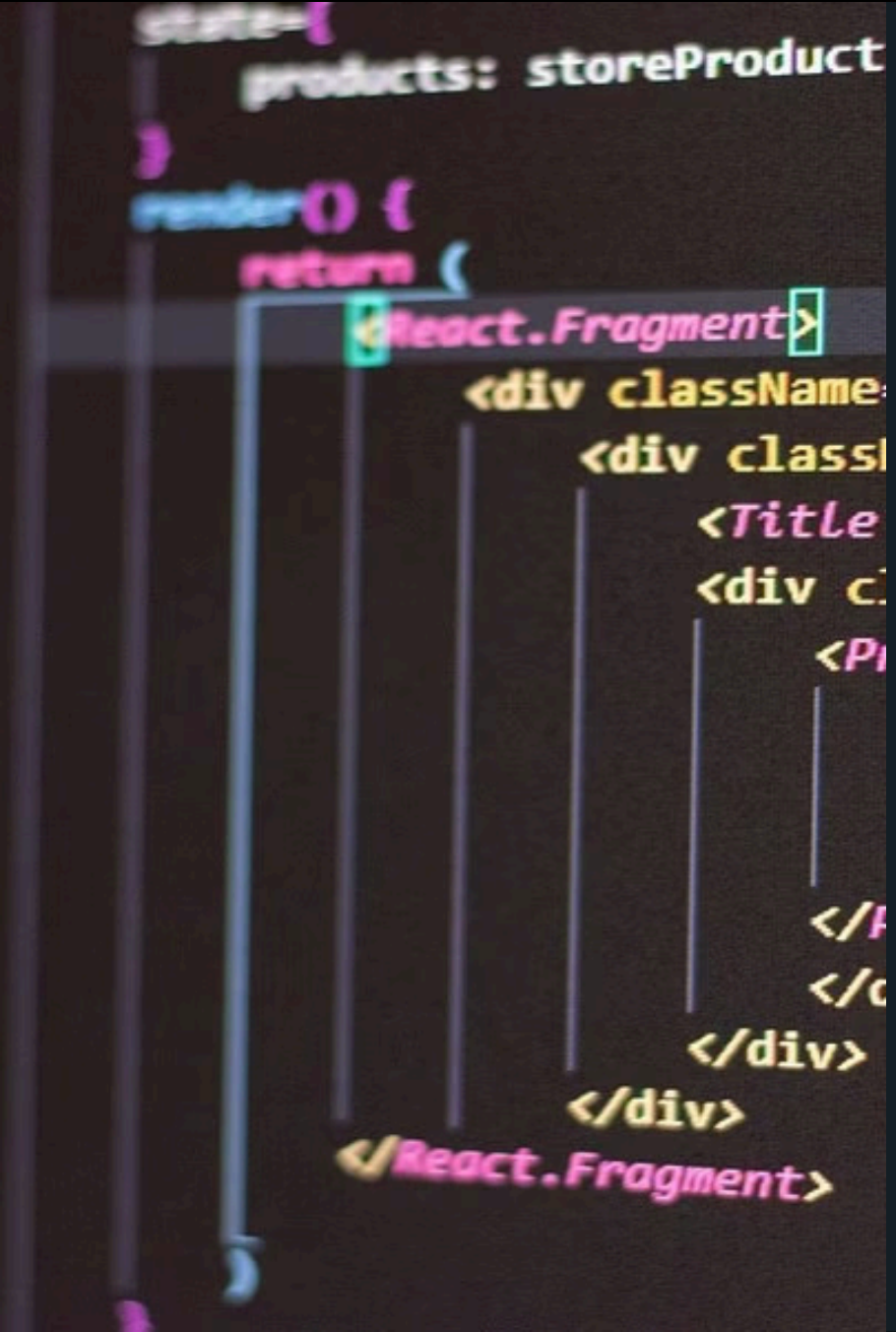
# University Explorer

The University Explorer is a comprehensive web application built with a modern tech stack, featuring a React.js frontend and Node.js/Express backend. The system provides an intuitive interface for exploring universities, programs, and educational opportunities while offering robust administrative capabilities for content management.

# High Level System Architecture






# Frontend Technology Stack

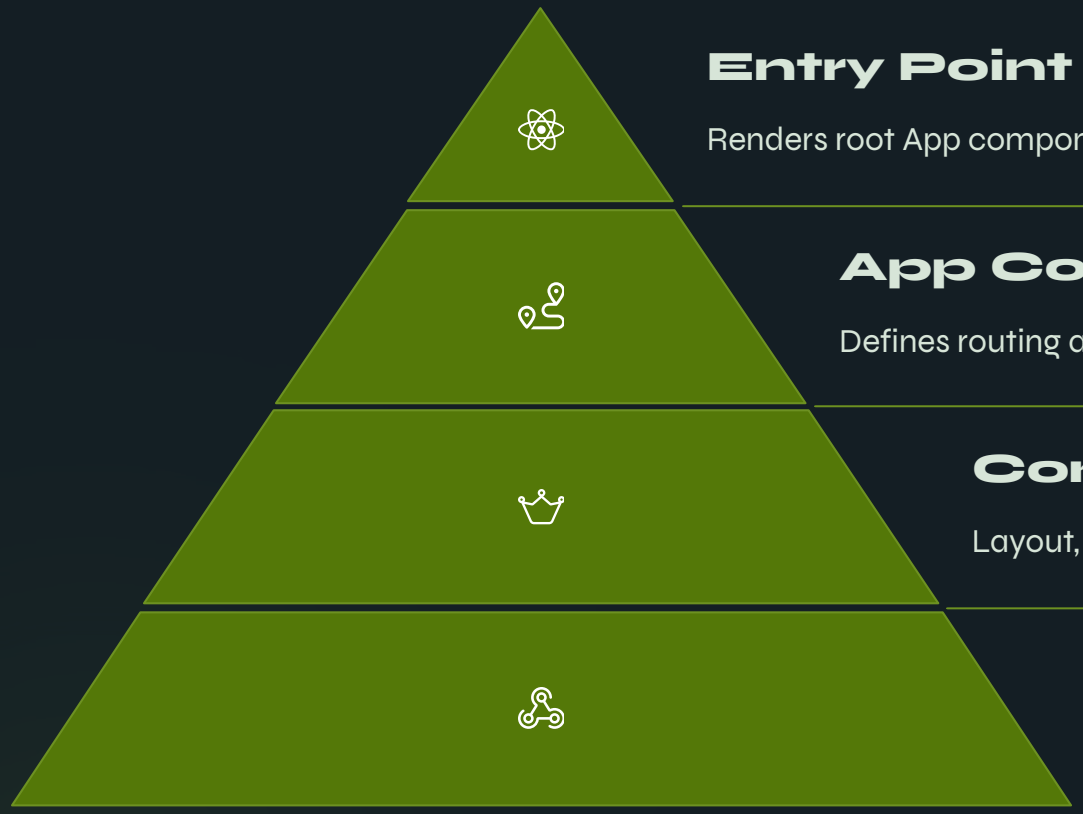
## Core Technologies

- React.js (v19.0.0)
- React Router (v7.4.0)
- Axios
- Reactstrap/Bootstrap 5
- JWT-decode

## Development Tools

- Create React App
- React Scripts
- ESLint
- Testing Library

# Frontend Architecture



## Entry Point (index.js)

Renders root App component with Bootstrap styling

## App Component (App.js)

Defines routing and wraps app in authentication context

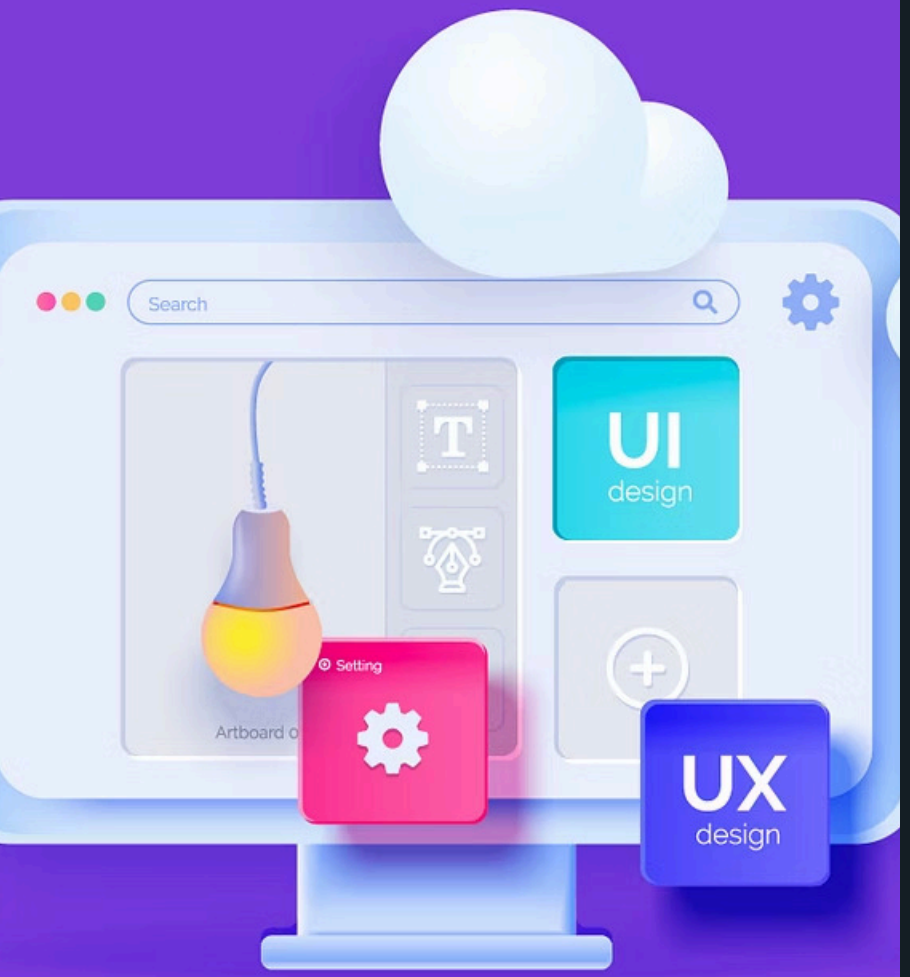
## Component Organization

Layout, Authentication, and Page Components

## API Layer (api.js)

Centralizes backend communication

The frontend implements a hierarchical component structure with clear separation of concerns. Components are organized by function, with layout components handling the overall structure, authentication components managing user access, and page components delivering specific content areas.




# Frontend Features

## Authentication System

- Token Storage in localStorage
- Automatic Token Handling with Axios interceptors
- Session Management with token retrieval on page refresh
- Protected Routes with authentication verification

## University Exploration

- Searching by university name
- Filtering by location and program offerings
- Sorting by ranking or name
- Favoriting universities

## Admin Dashboard

- Dashboard Overview with entity stats
- Entity Management with CRUD operations
- Form Validation for creating and editing entities

## User Interface Design

- Responsive Grid System
- Card-Based UI
- Consistent Navigation
- Visual Feedback
- Iconography

# Backend Technology Stack

### Node.js

JavaScript runtime environment for server-side execution

### Express.js

Web application framework for handling HTTP requests, routing, and middleware

### MongoDB

NoSQL database for data storage

### Mongoose

ODM library providing schema validation and relationship modeling

The backend also implements robust security features with JWT for authentication, bcryptjs for password hashing, helmet for security headers, express-rate-limit to prevent attacks, express-validator for data validation, and express-mongo-sanitize to prevent NoSQL injection.

# Backend Architecture

## Entry Point (server.js)

Configures Express, connects to MongoDB, sets up middleware and routes

## Models (Mongoose Schemas)

Defines structure and validation for University, Program, User, and Review

## Controllers

Implements business logic for each route with CRUD operations

## Routes

Defines API endpoints and connects them to controllers

The backend follows a modular architecture with clear separation of concerns. The server.js entry point configures the application, while models define data structures, controllers implement business logic, and routes expose API endpoints.

# API Endpoints

## Authentication APIs

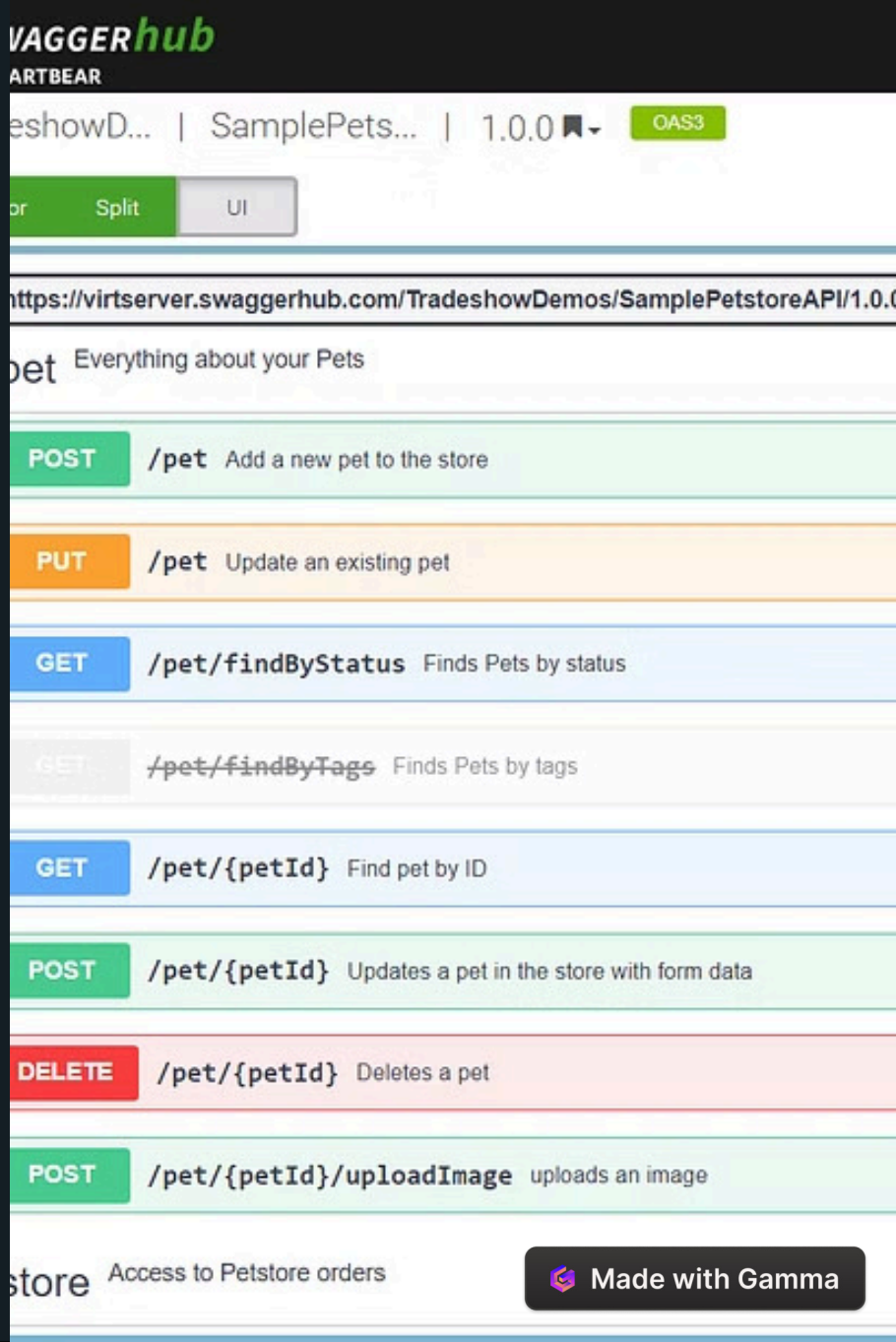
- POST /api/auth/register: User registration
- POST /api/auth/login: User authentication
- GET /api/auth/me: Get current user profile

## University APIs

- GET /api/universities: List universities with filtering, sorting, and pagination
- GET /api/universities/:id: Get university details with programs
- POST /api/universities: Create university (admin only)
- PUT /api/universities/:id: Update university (admin only)
- DELETE /api/universities/:id: Delete university (admin only)

## Program & User APIs

- GET /api/programs: List programs with filtering and sorting
- GET /api/user/profile: Get user profile
- GET /api/user/favorites: Get user's favorite universities
- POST /api/reviews/:universityId: Add a review for a university



# Security Features

## Authentication

JWT-based authentication with token expiration, HTTP-only cookies for token storage, and password hashing with bcrypt and salting

## Data Validation

Input validation with express-validator and MongoDB sanitization to prevent injection attacks

## Authorization

Role-based access control (admin/user) and route protection with middleware

## API Security

Rate limiting to prevent brute force attacks, security headers with Helmet.js, and CORS configuration

# Future Expansion Areas

## Advanced Search

More sophisticated filtering and comparison tools to enhance the university exploration experience

## User Settings

Personalization options and preferences to customize the application experience

## Analytics Dashboard

Data visualization for administration to gain insights into system usage and trends

## Internationalization

Multi-language support to make the application accessible to a global audience

## Accessibility Enhancements

ARIA attributes and keyboard navigation to improve usability for all users