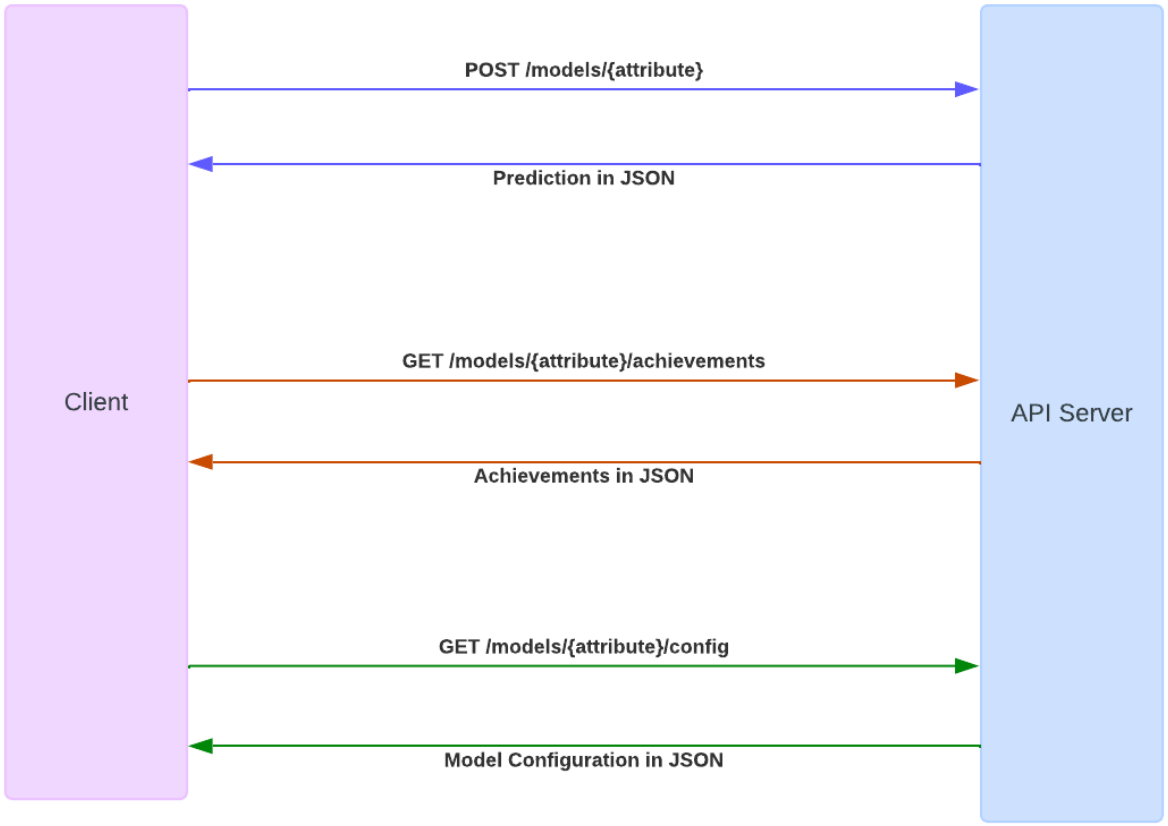
Endpoint Diagram of Flex Model API



The Flex Model API is currently setup with three endpoints for each attribute e.g. ‘grit’, ‘dependable’, ‘empathy’, etc. with no authentication.

Stateless API authentication methods

API Key Authentication

API Key authentication assigns a unique token to each client. Said client would then include the token in any request header or query parameter to access the API. The server verifies the token before processing the request.

1. Client sends request that includes API token
2. Server checks the validity of the token and the endpoint access it allows
3. Server processes the request

Pros:

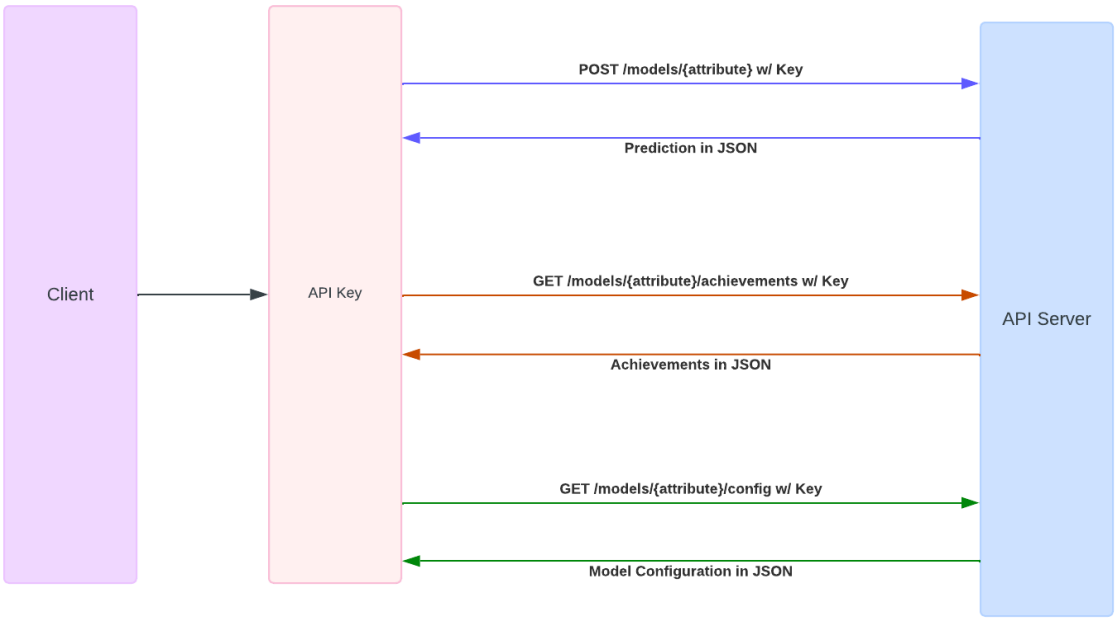
- Easier to implement
- Good for service-to-service communication
- Basic protection for model endpoints
- Access monitoring

Cons:

- No user identification, only identifies client
- Tokens don't expire by default, must be revoked manually
- How to issue different tokens for different services?
- All requests have the same level of authorization

Implementation Method:

Use os library to create an API key, store it in .env, and create a function to require api key and add decorators to each model call – super simple



Flex Model API with API Key Authentication.

JWT (JSON Web Token) Authentication

JWT is used to transmit securely-signed information between the client and server. The token has encoded user information and is signed using a secret key or public/private key pair.

1. Client sends API key (or credentials) to authenticate and receives a JWT
2. JWT is stored by the client and sent with every request
3. Server verifies JWT signature to check validity
4. Server processes the request

Pros:

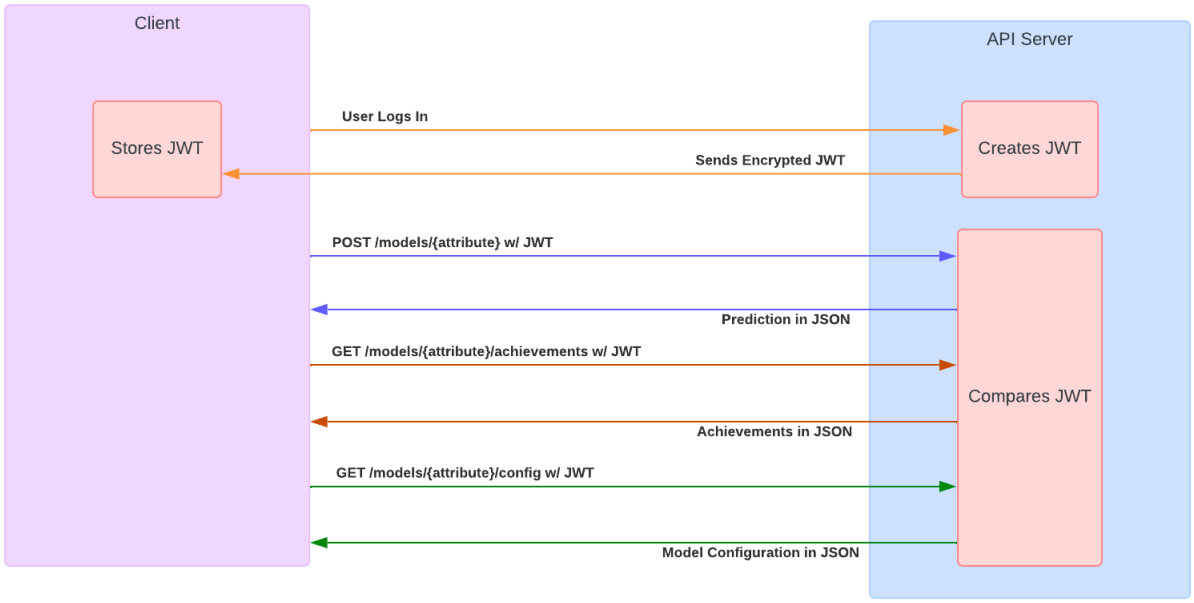
- JWT stores all information on the token, so there's no need to store sessions on the server
- Works well for scalable, distributed systems
- Can store user roles, permissions, or metadata.
- Has expiration times (refresh tokens can provide long-term access)
- Works well with single-page applications, mobile apps, and distributed systems
- Signed using HMAC or RSA, so the server can verify if the token has been tampered with

Cons:

- More complex than API keys
- Revocation requires token blacklist, or a short expiration time
- Much larger than API keys, so could impact network performance because of included payloads
- Requires secure storage of the secret key.

Implementation Method:

Use the Flask-JWT-Extended library and create a function in server.py to check JWT, and also set an expiration date for the tokens. Add decorators and jwt identity in each model function.

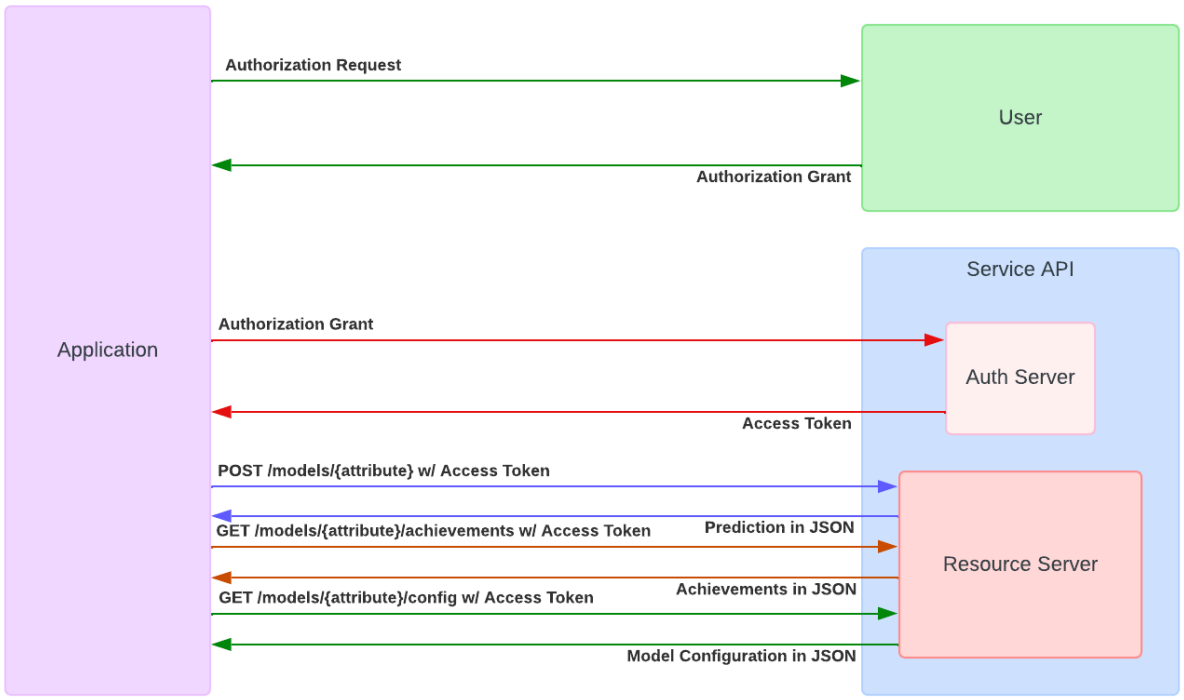


Flex Model API with JWT Authentication.

OAuth2 – Future Option (not stateless)

OAuth2 is a powerful authentication framework used by services like Google and GitHub. It's ideal when you need to allow **third-party access on behalf of users** (e.g., using Steam OpenID). OAuth2 might be useful if the API grows and requires user-specific permissions across multiple services.

Not required right now, as the users do not have to login to their steam.



Flex Model API with OAuth2

Other Recommended Implementations

Rate Limiting and Security Practices

Even with API Key or JWT Authentication, **rate limiting and secure transport** are crucial to prevent abuse and protect data.

- Implement Rate limiting to prevent abuse (e.g. 100 requests per hour per client). Use the Flask-Limiter library for setup.
- API key rotation should reduce risks from key leakage.
- Token blacklist should be created for manual revocation of tokens if JWT is used.