API Docs for MVP backend

Loretta Backend API Documentation

Version: 1.0
Base URL: `https://loretta-backend-dev-5oc2gjs2kq-el.a.run.app`
API Version: v1
Last Updated: December 2024

Table of Contents

User

email: loretta-dev@gmail.com

password: Test@123

Authentication

All ML service endpoints require authentication. You must sign in first to obtain an ID token.

Sign In

Endpoint: `POST /api/v1/auth/signin`

Description: Authenticate with email and password to receive an ID token for accessing protected endpoints.

Request:

```bash
curl -X POST https://loretta-backend-dev-5oc2gjs2kq-el.a.run.app/api/v1,
  -H "Content-Type: application/json" \
  -d '{
    "email": "user@example.com",
    "password": "your_password"
  }'
```

Request Body:

```
{
  "email": "string (required)",
  "password": "string (required)"
}
```

Response (200 OK):

```
{
  "success": true,
  "idToken": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjE...",
  "refreshToken": "AMf-vBxNrQ...",
  "expiresIn": "3600",
  "user": {
    "uid": "abc123def456",
    "email": "user@example.com",
    "displayName": "User Name",
    "role": "patient"
  }
}
```

Response Fields:

- `success` (boolean): Always `true` on successful authentication
- `idToken` (string): Firebase ID token - **Use this for Authorization header**
- `refreshToken` (string): Token for refreshing the ID token
- `expiresIn` (string): Token expiration time in seconds (typically 3600 = 1 hour)
- `user` (object): User information
  - `uid` (string): Unique user identifier
  - `email` (string): User email address
  - `displayName` (string): User display name
  - `role` (string): User role (must be "patient" for ML endpoints)

Error Responses:

400 Bad Request - Missing email or password:

```
{
  "error": "Email is required"
}
```

401 Unauthorized - Invalid credentials:

```
{
  "error": "Invalid email or password"
}
```

401 Unauthorized - Account disabled:

```
{
  "error": "This account has been disabled"
}
```

500 Internal Server Error:

```
{
  "error": "Internal server error"
}
```

Using the Token:

After receiving the `idToken`, include it in the `Authorization` header for all protected endpoints:

```
Authorization: Bearer YOUR_ID_TOKEN
```

Token Expiration: Tokens expire after 1 hour. You must sign in again to get a new token.

Questionnaire Endpoint

Endpoint: `GET /api/v1/ml/questionnaire`

Description: Retrieve the complete list of all features expected by the diabetes prediction model, including their descriptions, value types, and valid values.

Authorization: Required (Bearer token with `patient` role)

Request:

```
curl -X GET https://loretta-backend-dev-5oc2gjs2kq-el.a.run.app/api/v1/ml/questior
  -H "Authorization: Bearer YOUR_ID_TOKEN"
```

Response (200 OK):

```
{
  "questions": [
    {
      "ID": "RIDAGEYR",
      "Description": "Age in years at screening",
      "Value type": "Numerical",
      "Value description": {}
    },
    {
      "ID": "BPQ020",
      "Description": "Ever told you had high blood pressure",
      "Value type": "Categorical",
      "Value description": {
        "0.0": "No",
        "1.0": "Yes"
      }
    }
  ],
  "total_questions": 50
}
```

Response Fields:

- `questions` (array): List of feature information objects
    - `ID` (string): Feature identifier - **Use this in prediction requests**
    - `Description` (string): Human-readable description of the feature
    - `Value type` (string): Either `"Numerical"` or `"Categorical"`
    - `Value description` (object): For categorical features, maps numeric values to human-readable descriptions
- `total_questions` (integer): Total number of features expected by the model

Error Responses:

401 Unauthorized - Missing or invalid token:

```
{
  "error": "Unauthorized"
}
```

403 Forbidden - User doesn't have `patient` role:

```
{
  "error": "Forbidden"
}
```

503 Service Unavailable - ML service is unavailable:

```
{
  "error": "ML service is unavailable",
  "message": "Unable to fetch questionnaire at this time"
}
```

Use Cases:

- Build dynamic forms based on expected features
- Understand valid values for categorical features
- Validate feature IDs before making predictions
- Display user-friendly questions in your UI

Predict Endpoint

Endpoint: `POST /api/v1/ml/predict`

Description: Predict the probability of diabetes for a given set of health features.

Authorization: Required (Bearer token with `patient` role)

Request:

```
curl -X POST https://loretta-backend-dev-5oc2gjs2kq-el.a.run.app/api/v1/ml/predict
  -H "Authorization: Bearer YOUR_ID_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "features": [
      {"ID": "RIDAGEYR", "Value": "57"},
      {"ID": "BPQ020", "Value": "No"},
      {"ID": "SLQ300", "Value": "22:30"}
    ]
  }'
```

Request Body:

```
{
  "features": [
    {
      "ID": "string (required)",
      "Value": "string (required)"
    }
  ]
}
```

Request Fields:

- `features` (array, required): Array of feature input objects
  - `ID` (string, required): Feature identifier (must match IDs from `/questionnaire`)
  - `Value` (string, required): Feature value as a string. Can be:
    - Numeric string: `"57"`, `"125.0"`, `"3.02"`
    - Human-readable description: `"No"`, `"Yes"`, `"Very good"` (case-insensitive)
    - Time format (HH:MM): `"22:30"`, `"06:45"` (for time features only)
    - Empty string `""`: Treated as null and will be imputed

Important Notes:

- You don't need to include all 50 features in the request
- Missing features will be automatically set to `null` and imputed using training data statistics
- Empty strings are treated as `null` and will be imputed
- For categorical features, you can use either numeric strings or human-readable descriptions
- Case-insensitive matching is supported for categorical descriptions

- For time features (SLQ300, SLQ310, SLQ320, SLQ330), use HH:MM format (e.g., "22:30", "06:45")

Response (200 OK):

```
{
  "diabetes_probability": 0.234,
  "risk_level": "Low"
}
```

Response Fields:

- `diabetes_probability` (float): Probability of having diabetes (0.0 to 1.0)
- `risk_level` (string): Risk categorization based on probability:
  - `"Low"` : probability < 0.4
  - `"Medium"` : 0.4 ≤ probability < 0.7
  - `"High"` : probability ≥ 0.7

Error Responses:

400 Bad Request - Validation errors:

```
{
  "error": "Validation failed",
  "message": "Feature 'SLQ300': Invalid time format '7:00'. Expected HH:MM format
}
```

401 Unauthorized - Missing or invalid token:

```
{
  "error": "Unauthorized"
}
```

403 Forbidden - User doesn't have `patient` role:

```
{
  "error": "Forbidden"
}
```

503 Service Unavailable - ML service is unavailable:

```
{
  "error": "ML service is unavailable",
  "message": "Unable to process prediction at this time"
}
```

Complete Feature Reference

Feature Value Formats

Numerical Features

For numerical features, provide numeric values as strings:

```
{"ID": "RIDAGEYR", "Value": "57"}
{"ID": "WHD020", "Value": "125.0"}
{"ID": "INDFMPIR", "Value": "3.02"}
```

Categorical Features

For categorical features, you can use either:

Option 1: Numeric String (as used in training data)

```
{"ID": "BPQ020", "Value": "0.0"}  // No
{"ID": "BPQ020", "Value": "1.0"}  // Yes
```

Option 2: Human-Readable Description (case-insensitive)

```
{"ID": "BPQ020", "Value": "No"}
{"ID": "BPQ020", "Value": "yes"}  // Case-insensitive
{"ID": "HUQ010", "Value": "Very good"}
```

Time Features

For time features (SLQ300, SLQ310, SLQ320, SLQ330), use **HH:MM format**:

```
{"ID": "SLQ300", "Value": "22:30"}  // Sleep time on weekdays
{"ID": "SLQ310", "Value": "06:45"}  // Wake time on weekdays
{"ID": "SLQ320", "Value": "23:00"}  // Sleep time on weekends
{"ID": "SLQ330", "Value": "08:00"}  // Wake time on weekends
```

Time Format Requirements:

- Format: `HH:MM` (24-hour format)
- Hours: `00` to `23` (must be 2 digits with leading zero)
- Minutes: `00` to `59` (must be 2 digits with leading zero)
- Valid examples: `"00:00"`, `"06:45"`, `"22:30"`, `"23:59"`
- Invalid examples: `"7:00"` (missing leading zero), `"25:00"` (hour out of range), `"12:60"` (minute out of range)

Note: The API handles sin/cos conversion internally. You only need to provide the time in HH:MM format.

All Features

Demographics

RIDAGEYR - Age in years at screening

- **Type:** Numerical
- **Example:** `"57"`, `"45"`, `"30"`

RIDRETH3 - Race/Hispanic origin w/ NH Asian

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"Mexican American"`
  - `"1.0"` or `"Other Hispanic"`
  - `"2.0"` or `"Non-Hispanic White"`
  - `"3.0"` or `"Non-Hispanic Black"`
  - `"4.0"` or `"Non-Hispanic Asian"`
  - `"5.0"` or `"Other Race - Including Multi-Racial"`

DMDEDUC2 - Education level - Adults 20+

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"Less than 9th grade"`
  - `"1.0"` or `"9-11th grade (Includes 12th grade with no diploma)"`
  - `"2.0"` or `"High school graduate/GED or equivalent"`
  - `"3.0"` or `"Some college or AA degree"`
  - `"4.0"` or `"College graduate or above"`

DMDMARTZ - Marital status

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"Married/Living with partner"`
  - `"1.0"` or `"Widowed/Divorced/Separated"`
  - `"2.0"` or `"Never married"`

DMDHHSIZ - Total number of people in the Household

- **Type:** Categorical
- **Example:** `"1"` , `"2"` , `"3"` , `"4"`

INDFMPIR - Ratio of family income to poverty

- **Type:** Numerical
- **Example:** `"1.5"` , `"2.0"` , `"3.5"`

INDFMMPI - Family monthly poverty level index

- **Type:** Numerical
- **Example:** `"1.2"` , `"2.5"` , `"3.0"`

INQ300 - Family has savings more than $20,000

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"No"`
  - `"1.0"` or `"Yes"`

Health Conditions

BPQ020 - Ever told you had high blood pressure

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"No"`
  - `"1.0"` or `"Yes"`

BPQ080 - Doctor told you - high cholesterol level

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"No"`
  - `"1.0"` or `"Yes"`

DIQ160 - Ever told you have prediabetes

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"No"`
  - `"1.0"` or `"Yes"`

DIQ180 - Had blood tested past three years

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"No"`
  - `"1.0"` or `"Yes"`

MCQ160A - Doctor ever said you had arthritis

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"No"`
  - `"1.0"` or `"Yes"`

MCQ160B - Ever told had congestive heart failure

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"No"`
  - `"1.0"` or `"Yes"`

MCQ160C - Ever told you had coronary heart disease

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"No"`
  - `"1.0"` or `"Yes"`

MCQ160E - Ever told you had heart attack

- **Type:** Categorical

- **Valid Values:**
    - `"0.0"` or `"No"`
    - `"1.0"` or `"Yes"`

KIQ022 - Ever told you had weak/failing kidneys?

- **Type:** Categorical
- **Valid Values:**
    - `"0.0"` or `"No"`
    - `"1.0"` or `"Yes"`

MCQ560 - Ever had gallbladder surgery?

- **Type:** Categorical
- **Valid Values:**
    - `"0.0"` or `"No"`
    - `"1.0"` or `"Yes"`

General Health

HUQ010 - General health condition

- **Type:** Categorical
- **Valid Values:**
    - `"0.0"` or `"Excellent"`
    - `"1.0"` or `"Very good"`
    - `"2.0"` or `"Good"`
    - `"3.0"` or `"Fair"`
    - `"4.0"` or `"Poor"`

HUQ030 - Routine place to go for healthcare

- **Type:** Categorical
- **Valid Values:**
    - `"0.0"` or `"Yes"`
    - `"1.0"` or `"There is no place"`
    - `"2.0"` or `"There is more than one place"`

HUQ055 - Past 12 months had video conf w/Dr?

- **Type:** Categorical
- **Valid Values:**
    - `"0.0"` or `"No"`
    - `"1.0"` or `"Yes"`

DPQ030 - Trouble sleeping or sleeping too much

- **Type:** Categorical
- **Valid Values:**
    - `"0.0"` or `"Not at all"`

  - `"1.0"` or `"Several days"`
  - `"2.0"` or `"More than half the days"`
  - `"3.0"` or `"Nearly every day"`

Physical Measurements

WHD010 - Current self-reported height (inches)

- **Type:** Numerical
- **Example:** `"64"` , `"68"` , `"72"`

WHD020 - Current self-reported weight (pounds)

- **Type:** Numerical
- **Example:** `"125.0"` , `"150.0"` , `"180.0"`

WHD050 - Self-reported weight - 1 yr ago (pounds)

- **Type:** Numerical
- **Example:** `"120.0"` , `"145.0"` , `"175.0"`

Lifestyle & Activity

ALQ121 - Past 12 mos how often drink alc bev

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"Never in the last year"`
  - `"1.0"` or `"Every day"`
  - `"2.0"` or `"Nearly every day"`
  - `"3.0"` or `"3 to 4 times a week"`
  - `"4.0"` or `"2 times a week"`
  - `"5.0"` or `"Once a week"`
  - `"6.0"` or `"2 to 3 times a month"`
  - `"7.0"` or `"Once a month"`
  - `"8.0"` or `"7 to 11 times in the last year"`
  - `"9.0"` or `"3 to 6 times in the last year"`
  - `"10.0"` or `"1 to 2 times in the last year"`

PAD790 - Hour moderate LTPA/week

- **Type:** Numerical
- **Example:** `"2.5"` , `"5.0"` , `"10.0"`

PAD680 - Sedentary activity (hr/day)

- **Type:** Numerical
- **Example:** `"4.0"` , `"6.0"` , `"8.0"`

Sleep

SLD012 - Sleep hours - weekdays or workdays

- **Type:** Numerical
- **Example:** `"7.0"` , `"7.5"` , `"8.0"`

SLD013 - Sleep hours - weekends

- **Type:** Numerical
- **Example:** `"8.0"` , `"9.0"` , `"10.0"`

SLQ300 - Usual sleep time on weekdays or workdays

- **Type:** Time (HH:MM format)
- **Format:** `HH:MM` (24-hour format)
- **Example:** `"22:00"` , `"22:30"` , `"23:00"`
- **Note:** Hours must be 00-23, minutes must be 00-59. Must include leading zeros.

SLQ310 - Usual wake time on weekdays or workdays

- **Type:** Time (HH:MM format)
- **Format:** `HH:MM` (24-hour format)
- **Example:** `"06:00"` , `"06:45"` , `"07:30"`
- **Note:** Hours must be 00-23, minutes must be 00-59. Must include leading zeros.

SLQ320 - Usual sleep time on weekends

- **Type:** Time (HH:MM format)
- **Format:** `HH:MM` (24-hour format)
- **Example:** `"23:00"` , `"23:30"` , `"00:00"`
- **Note:** Hours must be 00-23, minutes must be 00-59. Must include leading zeros.

SLQ330 - Usual wake time on weekends

- **Type:** Time (HH:MM format)
- **Format:** `HH:MM` (24-hour format)
- **Example:** `"08:00"` , `"08:30"` , `"09:00"`
- **Note:** Hours must be 00-23, minutes must be 00-59. Must include leading zeros.

Medications

RXQ510 - Dr told to take daily low-dose aspirin?

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"No"`
  - `"1.0"` or `"Yes"`

RXQ033 - Taken prescription medicine, past month

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"No"`
  - `"1.0"` or `"Yes"`

Balance & Mobility

BAQ321C - Past 12 months, problems with unsteady

- **Type:** Categorical
- **Valid Values:**

  - `"0.0"` or `"No"`
  - `"1.0"` or `"Yes"`

BAQ530 - Past 5 years, how many times fallen?

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"Never"`
  - `"1.0"` or `"1 or 2 times"`
  - `"2.0"` or `"3 to 4 times"`
  - `"3.0"` or `"About every year"`
  - `"4.0"` or `"About every month"`
  - `"5.0"` or `"About every week"`
  - `"6.0"` or `"Daily or constantly"`

Hearing

AUQ054 - General condition of hearing

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"Excellent"`
  - `"1.0"` or `"Good"`
  - `"2.0"` or `"A little trouble"`
  - `"3.0"` or `"Moderate hearing trouble"`
  - `"4.0"` or `"A lot of trouble"`
  - `"5.0"` or `"Deaf"`

Oral Health

OHQ845 - Rate the health of your teeth and gums

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"Excellent"`
  - `"1.0"` or `"Very good"`
  - `"2.0"` or `"Good"`
  - `"3.0"` or `"Fair"`
  - `"4.0"` or `"Poor"`

OHQ620 - How often last yr. had aching in mouth?

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"Very often"`
  - `"1.0"` or `"Fairly often"`
  - `"2.0"` or `"Occasionally"`
  - `"3.0"` or `"Hardly ever"`
  - `"4.0"` or `"Never"`

OHQ630 - How often felt bad because of mouth?

- **Type:** Categorical
- **Valid Values:**
    - `"0.0"` or `"Very often"`
    - `"1.0"` or `"Fairly often"`
    - `"2.0"` or `"Occasionally"`
    - `"3.0"` or `"Hardly ever"`
    - `"4.0"` or `"Never"`

OHQ660 - Last yr avoid some food because of mouth

- **Type:** Categorical
- **Valid Values:**
    - `"0.0"` or `"Very often"`
    - `"1.0"` or `"Fairly often"`
    - `"2.0"` or `"Occasionally"`
    - `"3.0"` or `"Hardly ever"`
    - `"4.0"` or `"Never"`

OHQ670 - Last yr couldn't eat because of mouth

- **Type:** Categorical
- **Valid Values:**
    - `"0.0"` or `"Very often"`
    - `"1.0"` or `"Fairly often"`
    - `"2.0"` or `"Occasionally"`
    - `"3.0"` or `"Hardly ever"`
    - `"4.0"` or `"Never"`

Housing

HOD051 - Number of rooms in home

- **Type:** Categorical
- **Valid Values:**
    - `"0.0"` or `"1"`
    - `"1.0"` or `"2"`
    - `"2.0"` or `"3"`
    - `"3.0"` or `"4"`
    - `"4.0"` or `"5"`
    - `"5.0"` or `"6"`
    - `"6.0"` or `"7"`
    - `"7.0"` or `"8"`
    - `"8.0"` or `"9"`
    - `"9.0"` or `"10"`
    - `"10.0"` or `"11"`
    - `"11.0"` or `"12 or more"`

Employment

OCD150 - Type of work done last week

- **Type:** Categorical
- **Valid Values:**
  - `"0.0"` or `"Working at a job or business"`
  - `"1.0"` or `"With a job or business but not at work"`
  - `"2.0"` or `"Looking for work"`
  - `"3.0"` or `"Not working at a job or business"`

Error Handling

Common Error Responses

400 Bad Request - Validation errors:

```
{
  "error": "Validation failed",
  "message": "Feature 'SLQ300': Invalid time format '7:00'. Expected HH:MM format
}
```

401 Unauthorized - Missing or invalid token:

```
{
  "error": "Unauthorized"
}
```

403 Forbidden - User doesn't have required role:

```
{
  "error": "Forbidden"
}
```

503 Service Unavailable - ML service is unavailable:

```
{
  "error": "ML service is unavailable",
  "message": "Unable to process prediction at this time"
}
```

Time Validation Errors

Invalid time format:

```
{
  "error": "Validation failed",
  "message": "Feature 'SLQ300': Invalid time format '7:00'. Expected HH:MM format
}
```

Hour out of range:

```
{
  "error": "Validation failed",
  "message": "Feature 'SLQ300': Invalid time '25:00': hour must be between 00 and
}
```

Minute out of range:

```
{
  "error": "Validation failed",
  "message": "Feature 'SLQ300': Invalid time '12:60': minute must be between 00 ar
}
```

Code Examples

Python Example

```python
import requests

# Base URL
BASE_URL = "https://loretta-backend-dev-5oc2gjs2kq-el.a.run.app"

# Step 1: Sign in
signin_response = requests.post(
    f"{BASE_URL}/api/v1/auth/signin",
    json={
        "email": "user@example.com",
        "password": "your_password"
    }
)

if signin_response.status_code != 200:
    print(f"Sign-in failed: {signin_response.json()}")
    exit(1)

auth_data = signin_response.json()
id_token = auth_data["idToken"]
```

```
# Step 2: Get questionnaire
headers = {"Authorization": f"Bearer {id_token}"}
questionnaire_response = requests.get(
    f"{BASE_URL}/api/v1/ml/questionnaire",
    headers=headers
)

if questionnaire_response.status_code == 200:
    questionnaire = questionnaire_response.json()
    print(f"Total features: {questionnaire['total_questions']}")

# Step 3: Make prediction
prediction_data = {
    "features": [
        {"ID": "RIDAGEYR", "Value": "57"},
        {"ID": "BPQ020", "Value": "No"},
        {"ID": "DIQ180", "Value": "Yes"},
        {"ID": "HUQ010", "Value": "Very good"},
        {"ID": "WHD020", "Value": "125.0"},
        {"ID": "SLQ300", "Value": "22:30"},
        {"ID": "SLQ310", "Value": "06:45"},
        {"ID": "SLQ320", "Value": "23:00"},
        {"ID": "SLQ330", "Value": "08:00"}
    ]
}

prediction_response = requests.post(
    f"{BASE_URL}/api/v1/ml/predict",
    headers=headers,
    json=prediction_data
)

if prediction_response.status_code == 200:
    result = prediction_response.json()
    print(f"Diabetes Probability: {result['diabetes_probability']:.2%}")
    print(f"Risk Level: {result['risk_level']}")
else:
    print(f"Prediction failed: {prediction_response.json()}")
```

JavaScript/TypeScript Example

```
const BASE_URL = "https://loretta-backend-dev-5oc2gjs2kq-el.a.run.app";

// Step 1: Sign in
async function signIn(email: string, password: string) {
  const response = await fetch(`${BASE_URL}/api/v1/auth/signin`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ email, password }),
  });

  if (!response.ok) {
    throw new Error(`Sign-in failed: ${await response.text()}`);
  }

  return await response.json();
}

// Step 2: Get questionnaire
async function getQuestionnaire(idToken: string) {
  const response = await fetch(`${BASE_URL}/api/v1/ml/questionnaire`, {
    headers: { Authorization: `Bearer ${idToken}` },
  });

  if (!response.ok) {
    throw new Error(`Failed to get questionnaire: ${await response.text()}`);
  }

  return await response.json();
}

// Step 3: Make prediction
async function predict(idToken: string, features: Array<{ID: string, Value: string
  const response = await fetch(`${BASE_URL}/api/v1/ml/predict`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Authorization: `Bearer ${idToken}`,
    },
    body: JSON.stringify({ features }),
  });

  if (!response.ok) {
    throw new Error(`Prediction failed: ${await response.text()}`);
  }

  return await response.json();
}

// Usage
```

```
(async () => {
  try {
    // Sign in
    const authData = await signIn("user@example.com", "your_password");
    const idToken = authData.idToken;

    // Get questionnaire
    const questionnaire = await getQuestionnaire(idToken);
    console.log(`Total features: ${questionnaire.total_questions}`);

    // Make prediction
    const result = await predict(idToken, [
      { ID: "RIDAGEYR", Value: "57" },
      { ID: "BPQ020", Value: "No" },
      { ID: "SLQ300", Value: "22:30" },
      { ID: "SLQ310", Value: "06:45" },
    ]);

    console.log(`Diabetes Probability: ${(result.diabetes_probability * 100).toFix
    console.log(`Risk Level: ${result.risk_level}`);
  } catch (error) {
    console.error("Error:", error);
  }
})();
```

cURL Examples

Sign In:

```
curl -X POST https://loretta-backend-dev-5oc2gjs2kq-el.a.run.app/api/v1/auth/signi
  -H "Content-Type: application/json" \
  -d '{
    "email": "user@example.com",
    "password": "your_password"
  }'
```

Get Questionnaire:

```
curl -X GET https://loretta-backend-dev-5oc2gjs2kq-el.a.run.app/api/v1/ml/questior
  -H "Authorization: Bearer YOUR_ID_TOKEN"
```

Make Prediction:

```
curl -X POST https://loretta-backend-dev-5oc2gjs2kq-el.a.run.app/api/v1/ml/predict
  -H "Authorization: Bearer YOUR_ID_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "features": [
      {"ID": "RIDAGEYR", "Value": "57"},
      {"ID": "BPQ020", "Value": "No"},
      {"ID": "SLQ300", "Value": "22:30"},
      {"ID": "SLQ310", "Value": "06:45"}
    ]
  }'
```

Best Practices

1. **Token Management:** Store the ID token securely and refresh it before expiration (1 hour)
2. **Error Handling:** Always check response status codes and handle errors appropriately
3. **Feature Validation:** Use the `/questionnaire` endpoint to validate feature IDs and values before making predictions
4. **Time Format:** Always use HH:MM format with leading zeros for time features (e.g., "06:45" not "6:45")
5. **Missing Values:** You can omit features or use empty strings - they will be imputed automatically
6. **Categorical Values:** Use human-readable descriptions for better code readability (case-insensitive)

Support

For issues or questions, please contact the development team or refer to the internal documentation.

Last Updated: December 2024

Activity All