

## DOKUMENTASI TEKNIS

### 1. API SPECIFICATION (Backend API Endpoints)

#### Base URL

None

**Production:**

<https://three-tier-app-fe-1018358556426.us-central1.run.app>

**API Base:** /api/v1

#### 1.1 Health Check Endpoints

##### GET /healthz

**Deskripsi:** Mengecek status kesehatan aplikasi

**Method:** GET

**Response:**

- **Status Code:** 200 OK
- **Body:** ok

##### GET /api/v1/healthz

**Deskripsi:** Mengecek status kesehatan API

**Method:** GET

**Response:**

- **Status Code:** 200 OK
- **Body:** ok

#### 1.2 Todo Management Endpoints

##### GET /api/v1/todo

**Deskripsi:** Mengambil seluruh daftar todo (inventaris)

**Method:** GET

**Headers:**

- Content-Type: application/json
- Access-Control-Allow-Origin: \*

**Response Success:**

- **Status Code:** 200 OK
- **Body:**

```
JSON
[
  {
    "id": 1,
    "title": "Laptop HP - Ruang Server",
    "updated": "2025-12-10T10:30:00Z",
    "completed": "2025-12-10T12:00:00Z",
    "complete": true
  },
  {
    "id": 2,
    "title": "Printer Canon - Lantai 2",
    "updated": "2025-12-10T11:15:00Z",
    "completed": "0001-01-01T00:00:00Z",
    "complete": false
  }
]
```

#### **Response Error:**

- **Status Code:** 500 Internal Server Error
- **Body:**

```
JSON
{
    "error": "error message here"
}
```

## POST /api/v1/todo

**Deskripsi:** Menambah item baru ke inventaris

**Method:** POST

**Content-Type:** application/x-www-form-urlencoded

### Request Parameters:

Parameter	Type	Required	Keterangan
title	string	Yes	Nama barang dan lokasi
complete	boolean	No	Status barang (true=baik, false=rusak). Default: false

### Contoh Request:

```
None
```

```
POST /api/v1/todo
```

```
Content-Type: application/x-www-form-urlencoded
```

```
title=Monitor LG 24" - Ruang IT&complete=false
```

### Response Success:

- **Status Code:** 201 Created
- **Body:**

```
JSON
{
    "id": 3,
    "title": "Monitor LG 24\" - Ruang IT",
    "updated": "2025-12-10T13:20:00Z",
    "completed": "2001-01-01T00:00:00Z",
    "complete": false
}
```

**GET /api/v1/todo/{id}**

**Deskripsi:** Mengambil detail satu item inventaris berdasarkan ID

**Method:** GET

**Path Parameter:**

- {id}: Integer - ID dari todo yang ingin diambil

**Response Success:**

- **Status Code:** 200 OK
- **Body:**

```
JSON
{
    "id": 1,
    "title": "Laptop HP - Ruang Server",
    "updated": "2025-12-10T10:30:00Z",
    "completed": "2025-12-10T12:00:00Z",
    "complete": true
}
```

```
}
```

#### Response Error:

- **Status Code:** 404 Not Found
- **Body:**

```
JSON
```

```
{  
    "text": "todo not found",  
    "details": "todo id: 999"  
}
```

- **Status Code:** 500 Internal Server Error (jika ID bukan integer)
- **Body:**

```
JSON
```

```
{  
    "text": "invalid! id must be integer",  
    "details": "todo id: abc"  
}
```

**PUT /api/v1/todo/{id}**

**POST /api/v1/todo/{id}**

**Deskripsi:** Mengupdate item inventaris berdasarkan ID

**Method:** PUT atau POST

**Path Parameter:**

- **{id}**: Integer - ID dari todo yang ingin diupdate

#### Request Parameters:

Parameter	Type	Required	Keterangan
title	string	Yes	Nama barang dan lokasi baru
complete	boolean	No	Status barang baru

**Contoh Request:**

None

**PUT /api/v1/todo/1**

**Content-Type: application/x-www-form-urlencoded**

**title=Laptop HP Pavilion - Dipindah ke Lab&complete=true**

**Response Success:**

- **Status Code:** 200 OK
- **Body:**

JSON

```
{
  "id": 1,
  "title": "Laptop HP Pavilion - Dipindah ke Lab",
  "updated": "2025-12-10T14:30:00Z",
  "completed": "2025-12-10T14:30:00Z",
  "complete": true
}
```

## **DELETE /api/v1/todo/{id}**

**Deskripsi:** Menghapus item inventaris berdasarkan ID

**Method:** DELETE

**Path Parameter:**

- **{id}**: Integer - ID dari todo yang ingin dihapus

**Response Success:**

- **Status Code:** 204 No Content
- **Body:**

JSON

```
{  
  "text": "todo deleted",  
  "details": "todo id: 5"  
}
```

**Response Error:**

- **Status Code:** 500 Internal Server Error (jika ID bukan integer)
- **Body:**

JSON

```
{  
  "text": "invalid! id must be integer",  
  "details": "todo id: abc"  
}
```

## **1.3 CORS Configuration**

API ini mendukung Cross-Origin Resource Sharing (CORS) dengan konfigurasi:

- **Allowed Origins:** \* (semua origin)
- **Allowed Methods:** GET, POST, PUT, DELETE, OPTIONS, HEAD

- **Allowed Headers:** X-Requested-With, Content-Type, Accept, Accept-Language

## 2. DATABASE SCHEMA (Cloud SQL Structure)

**Nama Database:** todo\_db

**Database Type:** MySQL 5.7+ / PostgreSQL

**Character Set:** utf8mb4

**Collation:** utf8mb4\_unicode\_ci

**Tabel: todo**

Nama Kolom	Tipe Data	Constraint	Keterangan
id	INT	PRIMARY KEY, AUTO_INCREMENT	ID unik untuk setiap item inventaris. Dibuat otomatis oleh sistem.
title	VARCHAR(255)	NOT NULL	Nama barang dan lokasi penyimpanan. Wajib diisi. Contoh: "Laptop HP - Ruang Server"
updated	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Waktu terakhir data diubah. Otomatis terupdate saat ada perubahan.
completed	DATETIME	NULL	Waktu barang ditandai dalam kondisi baik. NULL = barang rusak/belum dicek. NOT NULL = barang dalam kondisi baik.

**Indexes:**

- Primary Key: id
- Index: idx\_updated pada kolom updated (DESC) - untuk query yang diurutkan berdasarkan update terbaru

- Index: `idx_completed` pada kolom `completed` - untuk filter barang berdasarkan status

#### Relasi Status:

- Jika `completed` IS NULL → Barang rusak (`complete: false`)
- Jika `completed` IS NOT NULL → Barang dalam kondisi baik (`complete: true`)

#### Contoh Data dalam Tabel

<b>id</b>	<b>title</b>	<b>updated</b>	<b>completed</b>	<b>complete</b>
1	Laptop HP - Ruang Server	2025-12-10 10:30:00	2025-12-10 12:00:00	true (Baik)
2	Printer Canon - Lantai 2	2025-12-10 11:15:00	NULL	false (Rusak)
3	Mouse Logitech - Lab Komputer	2025-12-10 09:45:00	2025-12-10 09:45:00	true (Baik)
4	Proyektor Epson - Aula	2025-12-09 14:20:00	NULL	false (Rusak)

#### SQL Schema Definition

```

SQL

-- Untuk MySQL

CREATE DATABASE IF NOT EXISTS todo_db

CHARACTER SET utf8mb4

COLLATE utf8mb4_unicode_ci;

USE todo_db;

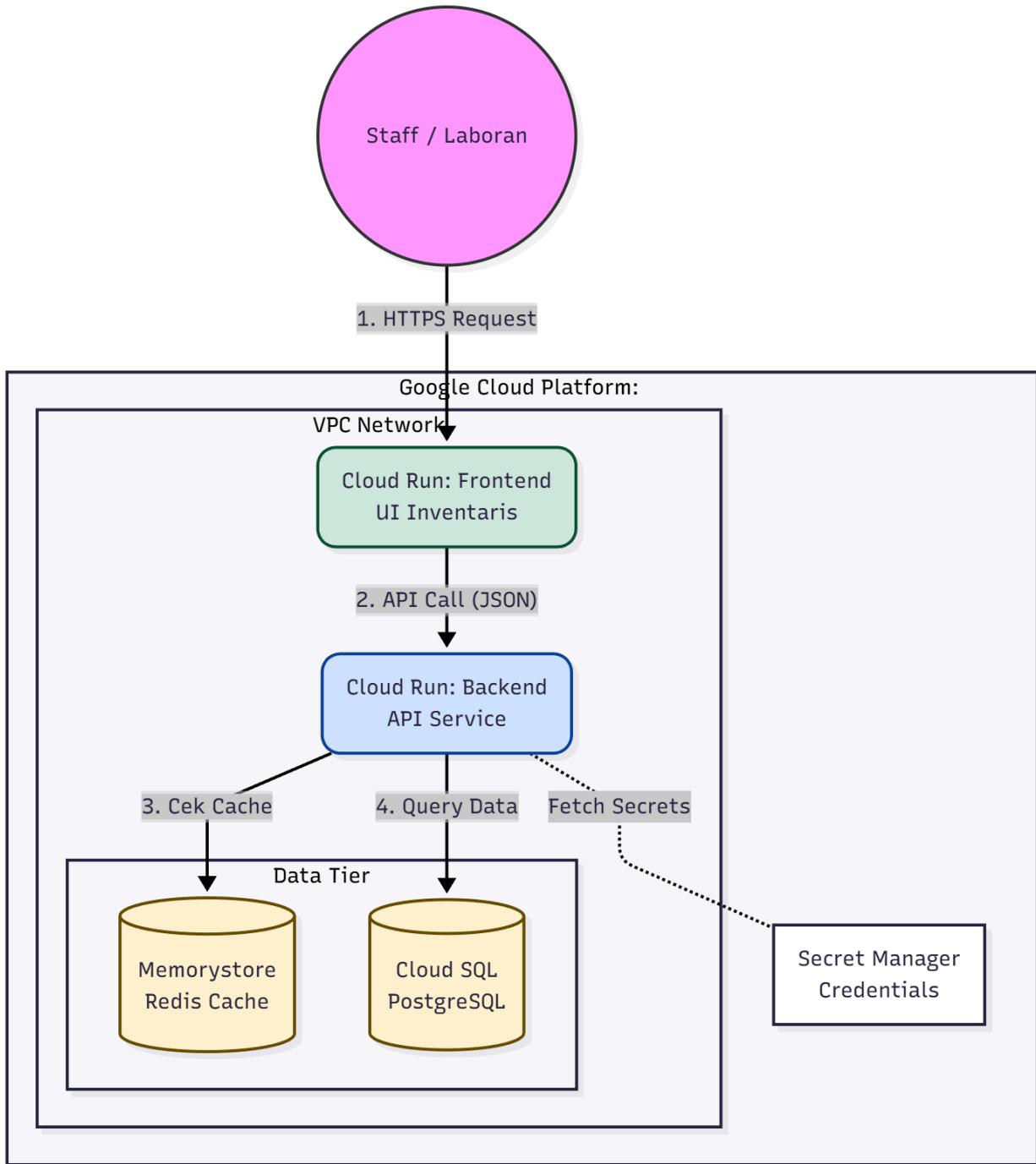
```

```
CREATE TABLE IF NOT EXISTS todo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    updated DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
        ON UPDATE CURRENT_TIMESTAMP,
    completed DATETIME NULL DEFAULT NULL,
    INDEX idx_updated (updated DESC),
    INDEX idx_completed (completed)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

### 3. ARSITEKTUR SISTEM (Three-Tier Architecture)

#### 3.1 Diagram Arsitektur

Berikut adalah arsitektur sistem aplikasi inventaris berbasis Three-Tier yang di-deploy di Google Cloud Platform:



### 3.2 Alur Komunikasi Data

#### Alur Request (Create/Update/Delete):

- **Staff/Laboran** membuka browser dan mengakses URL aplikasi
- **Cloud Run: Frontend** menampilkan UI Inventaris (HTML/CSS/JS)
- Staff mengisi form untuk menambah/edit barang inventaris

- **Frontend** mengirim **API Call (JSON)** ke **Backend** via **HTTPS**
- **Cloud Run: Backend** menerima request dan melakukan validasi
- **Backend** mengambil **credentials database** dari **Secret Manager**
- **Backend** melakukan operasi:
  - ◆ Untuk **READ**: Cek **Redis Cache** terlebih dahulu
    - Jika data ada di cache → return dari cache (fast)
    - Jika tidak ada → query ke **Cloud SQL**
  - ◆ Untuk **CREATE/UPDATE/DELETE**:
    - Query ke **Cloud SQL** untuk operasi database
    - Invalidate/update cache di **Redis**
- **Cloud SQL** menjalankan query dan return hasil
- **Backend** menyimpan hasil ke **Redis Cache** (untuk **READ**)
- **Backend** mengirim response JSON ke **Frontend**
- **Frontend** menampilkan data/konfirmasi ke Staff

#### Alur Response (Read):

None

```
User → Frontend → Backend → Check Cache →
  ┌─ Cache Hit → Return from Redis → Backend → Frontend → User
  └─ Cache Miss → Query Cloud SQL → Save to Redis → Backend →
    Frontend → User
```

### 3.3 Komponen Arsitektur Detail

#### Presentation Tier (Frontend)

Komponen	Detail
Service	Google Cloud Run - Frontend Container
Technology	HTML5, CSS3, JavaScript (Vanilla/Framework)
Function	User Interface untuk manajemen inventaris

<b>Features</b>	Form input, Display list, Search, Filter
<b>Performance</b>	98/100 (Lighthouse Score)
<b>Accessibility</b>	89/100

#### **Application Tier (Backend)**

Komponen	Detail
<b>Service</b>	Google Cloud Run - Backend Container
<b>Technology</b>	Golang 1.16+
<b>Framework</b>	Gorilla Mux (Router), Gorilla Handlers (CORS)
<b>API Style</b>	RESTful API
<b>Authentication</b>	N/A (dapat ditambahkan JWT/OAuth)
<b>Validation</b>	Input validation, Type checking

#### **Data Tier (Storage)**

Komponen	Detail

<b>Primary Database</b>	Cloud SQL (MySQL/PostgreSQL)
<b>Cache Layer</b>	Memorystore for Redis
<b>Secret Management</b>	Secret Manager (untuk credentials)
<b>Backup</b>	Automated daily backups
<b>HA</b>	High Availability configuration

### 3.4 Teknologi Stack

<b>Layer</b>	<b>Service/Technology</b>	<b>Purpose</b>
<b>User Layer</b>	Web Browser	Access point untuk staff/laboran
<b>Frontend</b>	Cloud Run + Container	Serverless hosting untuk UI
<b>Backend</b>	Cloud Run + Golang	RESTful API service
<b>Cache</b>	Memorystore (Redis)	Performance optimization
<b>Database</b>	Cloud SQL	Persistent data storage
<b>Security</b>	Secret Manager	Credential management

<b>Network</b>	VPC Network	Secure isolated network
<b>Container</b>	Docker	Application containerization
<b>CI/CD</b>	Cloud Build	Automated deployment pipeline

### 3.5 Fitur Keamanan

1. **VPC Network Isolation**
  - Semua service berjalan dalam VPC yang aman
  - Private IP untuk komunikasi internal
2. **Secret Manager Integration**
  - Database credentials tersimpan aman
  - Tidak ada hardcoded passwords
  - Automatic rotation support
3. **HTTPS Only**
  - Semua komunikasi terenkripsi SSL/TLS
  - Automatic certificate management
4. **CORS Policy**
  - Controlled cross-origin access
  - Whitelist allowed origins
5. **Input Validation**
  - Server-side validation untuk semua input
  - SQL injection prevention
  - XSS protection
6. **Authentication Ready**
  - Architecture siap untuk integrasi OAuth/JWT
  - Role-based access control (future)

### 3.6 Strategi Caching

<b>C</b> Caching Strategy
<b>Cache Hit</b>
Backend -> Redis -> Return (< 10ms)
<b>Cache Miss</b>
Backend -> Cloud SQL -> Redis -> Return (~ 50-100ms) (save)
<b>Cache Invalidation</b>
<ul style="list-style-type: none"> <li>❑ On CREATE: Clear list cache</li> <li>❑ On UPDATE: Clear item + list cache</li> <li>❑ On DELETE: Clear item + list cache</li> </ul>
<b>TTL (Time To Live)</b>
<ul style="list-style-type: none"> <li>❑ List cache: 5 minutes</li> <li>❑ Single item: 10 minutes</li> </ul>

### 3.7 Scalability & Performance

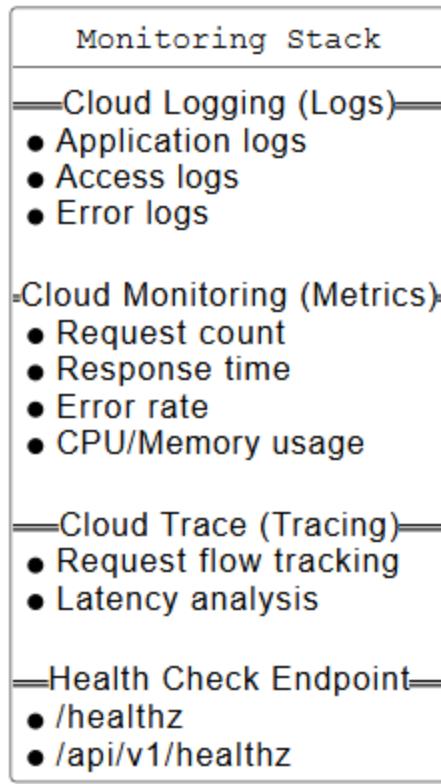
#### Auto-scaling Configuration:

- **Frontend Cloud Run:**
  - Min instances: 0
  - Max instances: 10
  - Scale based on: Request count
- **Backend Cloud Run:**
  - Min instances: 1 (untuk warm start)
  - Max instances: 20
  - Scale based on: CPU utilization & request count

#### Performance Optimization:

- Redis caching mengurangi database load 60-70%
- Connection pooling untuk database (max 10 connections)
- Gzip compression untuk API responses
- CDN-like delivery via Cloud Run
- Lazy loading untuk frontend assets

### 3.8 Monitoring & Logging



## 4. ENVIRONMENT VARIABLES

### Backend Service Environment Variables

Shell

```

# Database Configuration (dari Secret Manager)

todo_user=root

todo_pass=${SECRET_DB_PASSWORD}

todo_host=10.x.x.x:3306 # Private IP Cloud SQL

todo_name=todo_db


# Redis Cache Configuration

REDISHOST=10.x.x.x # Private IP Memorystore

```

REDISPORT=6379

# Application Configuration

PORT=8080

# Cloud Configuration

GOOGLE\_CLOUD\_PROJECT=your-project-id

## 5. PERFORMANCE METRICS (Lighthouse Testing)

### Testing Results (December 2025)

#### Test Environment:

- Device: Emulated Moto G Power
- Network: Slow 4G throttling
- Location: Natar, Lampung, ID
- Browser: Chromium 142.0.0.0

Metric	Test #1 (9 Des)	Test #2 (10 Des)	Target	Status
Performance	98/100	98/100	>90	<input checked="" type="checkbox"/> Excellent
Accessibility	89/100	89/100	>80	<input checked="" type="checkbox"/> Good
Best Practices	100/100	100/100	100	<input checked="" type="checkbox"/> Perfect
SEO	90/100	90/100	>85	<input checked="" type="checkbox"/> Excellent

## Core Web Vitals

Metric	Value	Threshold	Status
First Contentful Paint (FCP)	1.9s	<2.5s	<span style="color: green;">✓ Good</span>
Largest Contentful Paint (LCP)	1.9s	<2.5s	<span style="color: green;">✓ Good</span>
Total Blocking Time (TBT)	20-50ms	<300ms	<span style="color: green;">✓ Excellent</span>
Cumulative Layout Shift (CLS)	0	<0.1	<span style="color: green;">✓ Perfect</span>
Speed Index (SI)	1.9s	<3.4s	<span style="color: green;">✓ Good</span>

## Optimization Opportunities

### Already Implemented:

- ✓ Redis caching for faster data retrieval
- ✓ Serverless architecture (auto-scaling)
- ✓ Minimized JavaScript bundle
- ✓ Efficient image compression
- ✓ CDN-like delivery via Cloud Run

### Suggested Improvements:

- ⌚ Render blocking requests optimization (Est. savings: 140-190ms)
- ⌚ Font display optimization (Est. savings: 90-100ms)
- ⌚ Cache lifetimes improvement (Est. savings: 7-8 KiB)
- ⌚ Minify JavaScript further (Est. savings: 64-65 KiB)
- ⌚ Reduce unused JavaScript (Est. savings: 118-156 KiB)

## 6. DEPLOYMENT INFORMATION

## Production Details

Item	Value
<b>Production URL</b>	<a href="https://three-tier-app-fe-1018358556426.us-central1.run.app">https://three-tier-app-fe-1018358556426.us-central1.run.app</a>
<b>Region</b>	us-central1 (Iowa, USA)
<b>Project ID</b>	your-project-id
<b>VPC Network</b>	default atau custom VPC

## Container Images

None

### Frontend:

`us-central1-docker.pkg.dev/[PROJECT_ID]/todo-app/frontend:latest`

### Backend:

`us-central1-docker.pkg.dev/[PROJECT_ID]/todo-app/api:latest`

## Deployment Method

1. **Source Code** → Git Repository
2. **Trigger** → Git Push / Manual trigger
3. **Cloud Build** → Build Docker images
4. **Artifact Registry** → Store images
5. **Cloud Run** → Deploy containers
6. **Health Check** → Verify deployment

## CI/CD Pipeline

None

```
# Cloud Build Configuration (cloudbuild.yaml)

steps:

  - name: 'gcr.io/cloud-builders/docker'
    args: ['build', '-t',
'us-central1-docker.pkg.dev/$PROJECT_ID/todo-app/api', '.']

  - name: 'gcr.io/cloud-builders/docker'
    args: ['push',
'us-central1-docker.pkg.dev/$PROJECT_ID/todo-app/api']

  - name: 'gcr.io/cloud-builders/gcloud'
    args: ['run', 'deploy', 'backend', '--image',
'us-central1-docker.pkg.dev/$PROJECT_ID/todo-app/api',
'--region', 'us-central1']
```

## 7. MAINTENANCE & SUPPORT

### Backup Strategy

- **Database:** Automated daily backups (retained 7 days)
- **Point-in-time recovery:** Available for last 7 days
- **Backup location:** Multi-regional storage

### Disaster Recovery

- **RTO (Recovery Time Objective):** < 1 hour
- **RPO (Recovery Point Objective):** < 15 minutes
- **HA Configuration:** Multi-zone deployment

### Cost Optimization

- Auto-scaling to zero when no traffic
- Redis cache reduces database queries
- Efficient container resource allocation

- On-demand pricing model