# LAPORAN PROYEK AKHIR - KOMPUTASI AWAN

## ITERASI 2 (FULL FEATURES )

**Judul Proyek: Sistem Inventaris Laboratorium Berbasis Cloud (LabInventory)**

**Mata Kuliah: Komputasi Awan (IF25-40201)**


| | |
|---|---|
| **Martua Kevin A.M.H.Lubis** | **122140119** |
| **Rachel Olivia Manullang** | **122140181** |
| **Christoper Benaya T** | **122140090** |
| **Arof Andestama** | **121140182** |
| **Ichza Auliya Gumilar** | **120140188** |
| **Fatkhan Aziez S** | **120140181** |

**INSTITUT TEKNOLOGI SUMATERA (ITERA)**

**FAKULTAS TEKNOLOGI INDUSTRI**

# DAFTAR ISI


**DOKUMENTASI TEKNIS** **4**

1. API SPECIFICATION (Backend API Endpoints) 4

Base URL 4

1.1 Health Check Endpoints 4

GET /healthz 4

GET /api/v1/healthz 4

1.2 Todo Management Endpoints 4

GET /api/v1/todo 4

POST /api/v1/todo 6

GET /api/v1/todo/{id} 7

PUT /api/v1/todo/{id} 8

POST /api/v1/todo/{id} 8

DELETE /api/v1/todo/{id} 10

1.3 CORS Configuration 10

2. DATABASE SCHEMA (Cloud SQL Structure) 11

Nama Database: todo_db 11

Tabel: todo 11

Contoh Data dalam Tabel 12

SQL Schema Definition 12

3. ARSITEKTUR SISTEM (Three-Tier Architecture) 13

3.1 Diagram Arsitektur 13

3.2 Alur Komunikasi Data 14

Alur Request (Create/Update/Delete): 14

Alur Response (Read): 15

3.3 Komponen Arsitektur Detail 15

Presentation Tier (Frontend) 15

Application Tier (Backend) 16

Data Tier (Storage) 16

3.4 Teknologi Stack 17

3.5 Fitur Keamanan 18

3.6 Strategi Caching 18

Auto-scaling Configuration: 19

Performance Optimization: 19

Backend Service Environment Variables 20

Testing Results (December 2025) 21

Core Web Vitals 22

Optimization Opportunities 22

6. DEPLOYMENT INFORMATION 22

Production Details 23

Container Images 23

Deployment Method 23
CI/CD Pipeline 23
7. MAINTENANCE & SUPPORT 24
Backup Strategy 24
Disaster Recovery 24
Cost Optimization 24
**Analisis Biaya & Efisiensi Cloud (Cost Analysis) 25**
**Laporan Security Audit 26**
1. Network & Infrastructure Security (Keamanan Jaringan) 26
2. Application & Data Security (Keamanan Aplikasi) 26
3. Access Control (Manajemen Akses) 27
**USER MANUAL**
**SISTEM INVENTARIS LAB IT 28**
1. Pendahuluan 28
1.1 Tujuan Dokumen 28
1.2 Ikhtisar Sistem 28
2. Akses Sistem 28
3. Dashboard Utama 28
4. Pengelolaan Inventaris 29

**DOKUMENTASI TEKNIS**

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

**Base URL**

```
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": "0001-01-01T00:00:00Z",
  "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**

| id | title | updated | completed | complete |
|---|---|---|---|---|
| 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;
```

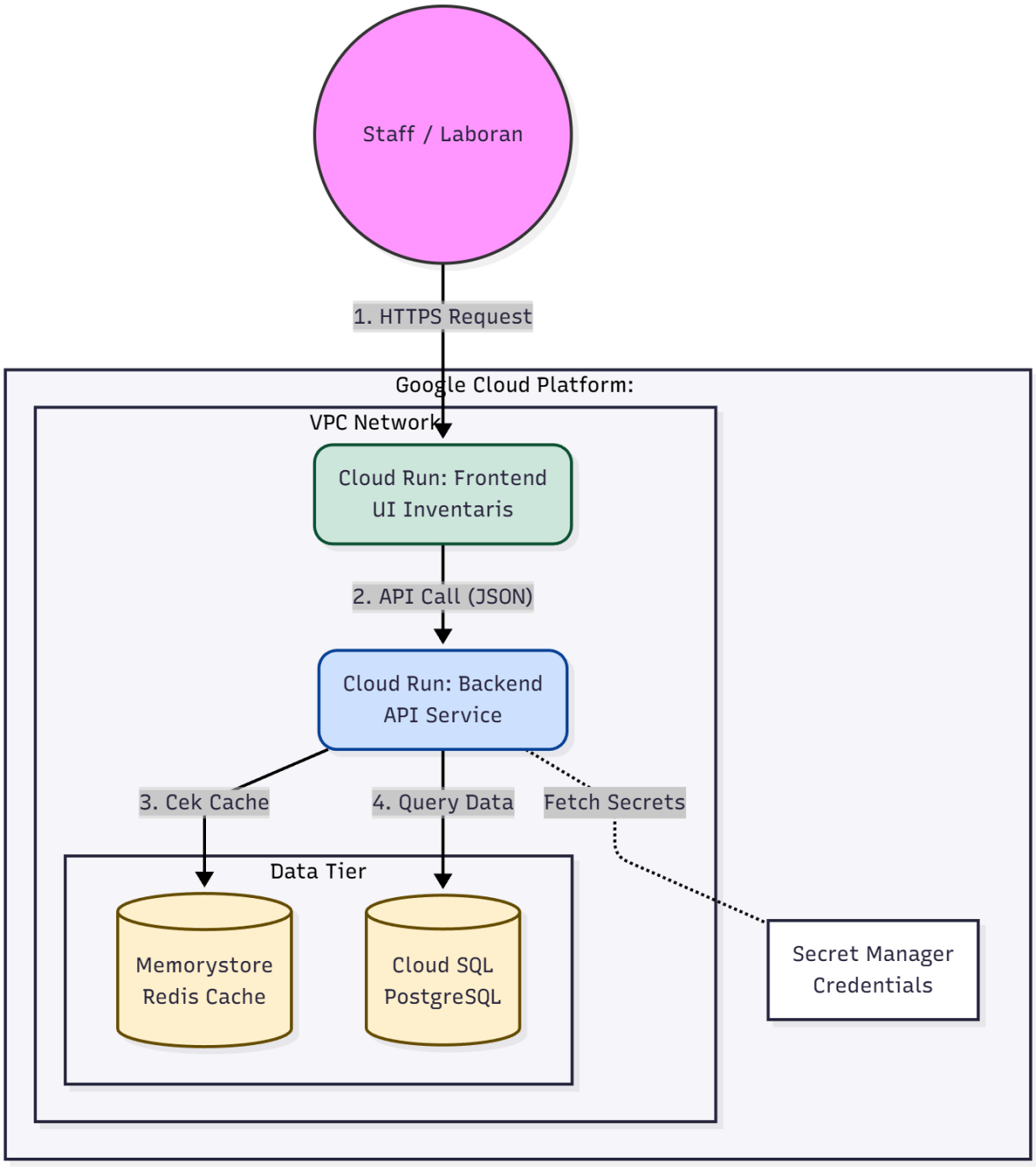
```sql
CREATE TABLE IF NOT EXISTS todo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    updated DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
            ON UPDATE CURRENT_TIMESTAMP,
    completed DATETIME NULL DEFAULT NULL,

    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 |

| | |
|---|---|
| **Features** | Form input, Display list, Search, Filter |
| **Performance** | 98/100 (Lighthouse Score) |
| **Accessibility** | 89/100 |

**Application Tier (Backend)**

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

**Data Tier (Storage)**

| Komponen | Detail |
|---|---|

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

**3.4 Teknologi Stack**

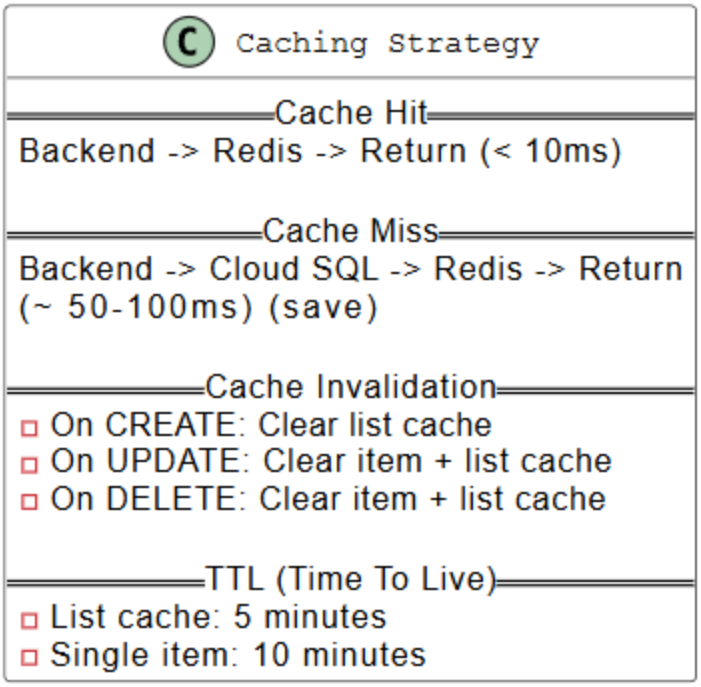
| Layer | Service/Technology | Purpose |
|---|---|---|
| **User Layer** | Web Browser | Access point untuk staff/laboran |
| **Frontend** | Cloud Run + Container | Serverless hosting untuk UI |
| **Backend** | Cloud Run + Golang | RESTful API service |
| **Cache** | Memorystore (Redis) | Performance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

| | | |
|---|---|---|
| **Network** | VPC Network | Secure isolated network |
| **Container** | Docker | Application containerization |
| **CI/CD** | 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




**3.7 Scalability & Performance**

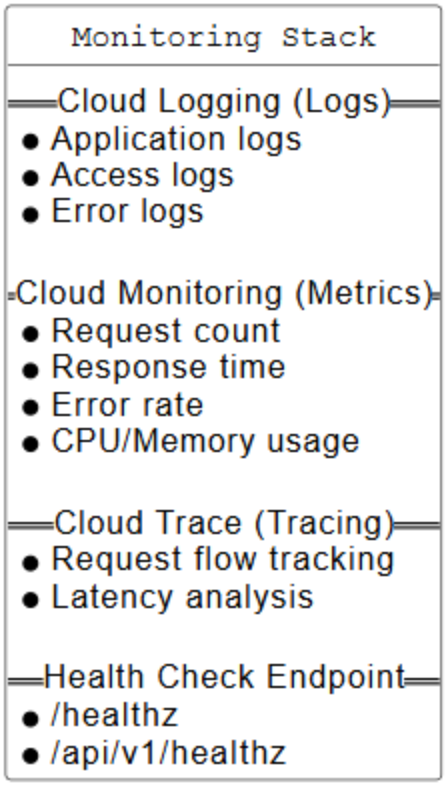
**Auto-scaling Configuration:**

- **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 | ✅ Excellent |
| **Accessibility** | 89/100 | 89/100 | >80 | ✅ Good |
| **Best Practices** | 100/100 | 100/100 | 100 | ✅ Perfect |
| **SEO** | 90/100 | 90/100 | >85 | ✅ Excellent |

**Core Web Vitals**

| Metric | Value | Threshold | Status |
|---|---|---|---|
| **First Contentful Paint (FCP)** | 1.9s | <2.5s | ✅ Good |
| **Largest Contentful Paint (LCP)** | 1.9s | <2.5s | ✅ Good |
| **Total Blocking Time (TBT)** | 20-50ms | <300ms | ✅ Excellent |
| **Cumulative Layout Shift (CLS)** | 0 | <0.1 | ✅ Perfect |
| **Speed Index (SI)** | 1.9s | <3.4s | ✅ Good |

**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 |
|---|---|
| **Production URL** | https://three-tier-app-fe-1018358556426.us-central1.run.app |
| **Region** | us-central1 (Iowa, USA) |
| **Project ID** | your-project-id |
| **VPC Network** | 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**

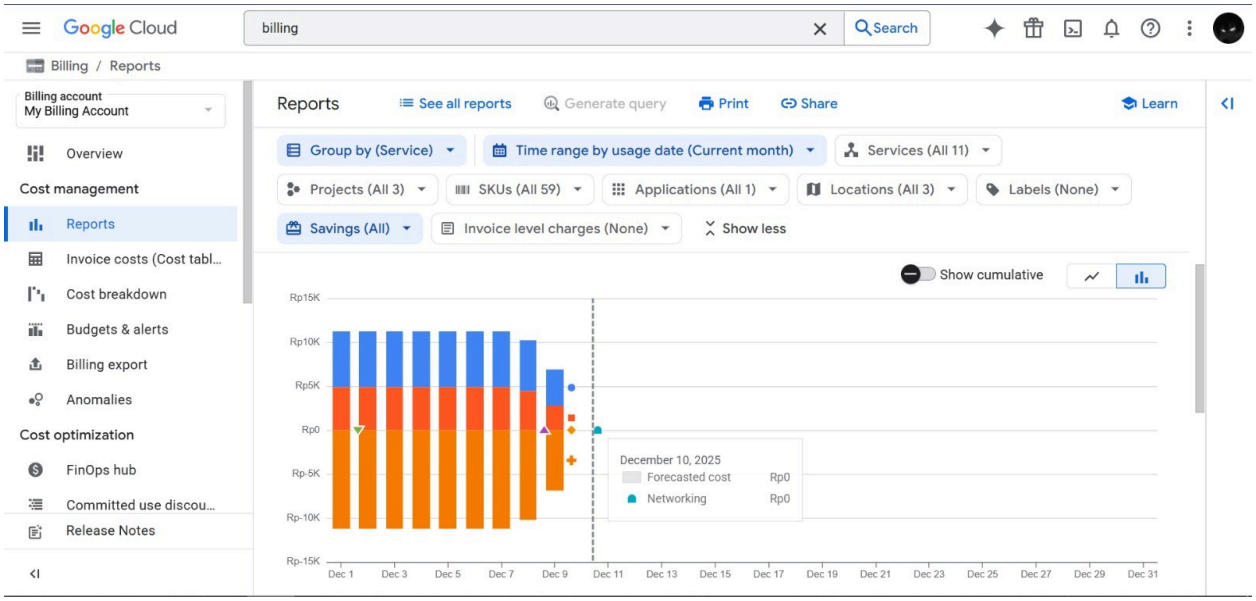
- **RTO (Recovery Time Objective)**: < 1 hour
- **RPO (Recovery Point Objective)**: < 15 minutes
- **HA Configuration**: Multi-zone deployment

**Cost Optimization**

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

**Analisis Biaya & Efisiensi Cloud (Cost Analysis)**

**Gambar:** Grafik Billing (Cost Analysis)




**1. Analisis Grafik Penggunaan (Usage vs Credits)** Berdasarkan grafik *Billing Reports* per tanggal 10 Desember 2025, terlihat struktur biaya sebagai berikut:

- **Total Usage Cost (Bar Biru & Merah):** Infrastruktur memunculkan biaya kotor (gross cost) sekitar **Rp 10.000 - Rp 12.000 per hari**. Biaya ini berasal dari operasional *Cloud SQL* (Database) dan *Compute* (Cloud Run).
- **Discounts/Credits (Bar Oranye):** Grafik menunjukkan nilai negatif (ke bawah) yang menyeimbangkan biaya penggunaan secara simetris. Ini merepresentasikan **Promotional Credits** atau **Free Tier** dari Google Cloud Platform yang secara otomatis memotong tagihan.
- **Net Cost (Titik Garis Rp 0):** Hasil akhirnya (Net Cost) berada di garis **Rp 0**. Artinya, tidak ada biaya tunai yang harus dibayarkan.

**2. Alasan Efisiensi Biaya (Why Low Cost?)** Rendahnya biaya operasional ini dicapai berkat penerapan arsitektur **Serverless** menggunakan **Google Cloud Run**:

- **Mekanisme Scale-to-Zero:** Berbeda dengan *Virtual Machine* (VM) tradisional yang menyedot biaya 24 jam non-stop meskipun tidak ada pengguna, Cloud Run memiliki fitur *Scale-to-Zero*.
- **Pay-as-you-go:** Server Frontend dan Backend kami otomatis "mati" (idle) saat tidak ada traffic/request masuk. Google hanya menagih biaya hitungan detik saat server memproses permintaan user.

- **Optimasi Resource:** Karena aplikasi ini masih dalam tahap MVP (Iterasi 1 & 2), penggunaan CPU dan Memory masih sangat rendah sehingga sepenuhnya tertutup oleh kuota *Free Tier* bulanan.

**Laporan Security Audit**

Status Keamanan: Baseline Security (MVP Level)

Tanggal Audit: [10 Desember 2025]

Berikut adalah hasil analisis keamanan pada infrastruktur Three-Tier Google Cloud Platform yang telah dibangun:

**1. Network & Infrastructure Security (Keamanan Jaringan)**

| **Kontrol Keamanan** | **Status** | **Keterangan Teknis** |
|---|---|---|
| **HTTPS/TLS Encryption** | ✅ **Aman** | Seluruh trafik dari User ke Frontend dan Backend dienkripsi secara otomatis menggunakan sertifikat SSL/TLS yang dikelola oleh Google Cloud Run. Data tidak dapat disadap di tengah jalan (*Man-in-the-Middle Protection*). |
| **Database Isolation** | ✅ **Aman** | Database Cloud SQL dikonfigurasi menggunakan *Private IP* dalam jaringan VPC internal. Database tidak memiliki IP Publik, sehingga tidak bisa diakses langsung dari internet terbuka (Anti-Brute Force). |
| **DDoS Protection** | ✅ **Aman** | Menggunakan infrastruktur Google Front End (GFE) yang secara bawaan memiliki mitigasi terhadap serangan DDoS dasar sebelum trafik mencapai server aplikasi. |

**2. Application & Data Security (Keamanan Aplikasi)**

| **Kontrol Keamanan** | **Status** | **Keterangan Teknis** |
|---|---|---|
| **Container Security** | ✅ **Aman** | Aplikasi berjalan di atas container (Docker) yang terisolasi. Jika satu service crash atau diserang, tidak akan langsung mempengaruhi service lain atau host system. |

| | | |
|---|---|---|
| **Input Validation** | ⚠️ **Parsial** | Frontend menggunakan validasi dasar (HTML5 required attributes) untuk mencegah input kosong. Namun, validasi sisi server (Backend sanitization) masih perlu ditingkatkan untuk mencegah *XSS/SQL Injection* tingkat lanjut. |
| **Secret Management** | ✅ **Aman** | Kredensial database tidak di-hardcode di dalam kodingan aplikasi, melainkan dikelola melalui *Environment Variables* di Cloud Run dan Terraform State yang dijaga kerahasiaannya. |

**3. Access Control (Manajemen Akses)**

| Kontrol Keamanan | Status | Keterangan Teknis |
|---|---|---|
| **Developer Access** | ✅ **Aman** | Akses ke Google Cloud Console dibatasi menggunakan *IAM (Identity and Access Management)*. Hanya anggota kelompok yang terdaftar yang memiliki hak *Owner/Editor* untuk mengubah infrastruktur. |
| **User Authentication** | ❌ **Future Work** | **(Risiko Diterima)** Saat ini aplikasi diset --allow-unauthenticated (Publik) untuk kebutuhan demo MVP. Siapapun yang memiliki link dapat mengakses fitur. Implementasi *Google Identity Aware Proxy (IAP)* atau *OAuth2* dijadwalkan untuk Iterasi 3. |

# USER MANUAL
# SISTEM INVENTARIS LAB IT

## 1. Pendahuluan

### 1.1 Tujuan Dokumen

Manual ini menjelaskan cara menggunakan Sistem Inventaris Lab IT untuk mengelola aset teknologi di laboratorium komputer. Sistem ini memungkinkan pelacakan, pemutakhiran, dan pengelolaan inventaris perangkat keras IT.

### 1.2 Ikhtisar Sistem

Sistem ini adalah aplikasi web yang menyediakan antarmuka untuk:

a. Melihat daftar inventaris perangkat IT

b. Menambahkan aset baru

c. Mengelola kondisi dan lokasi barang

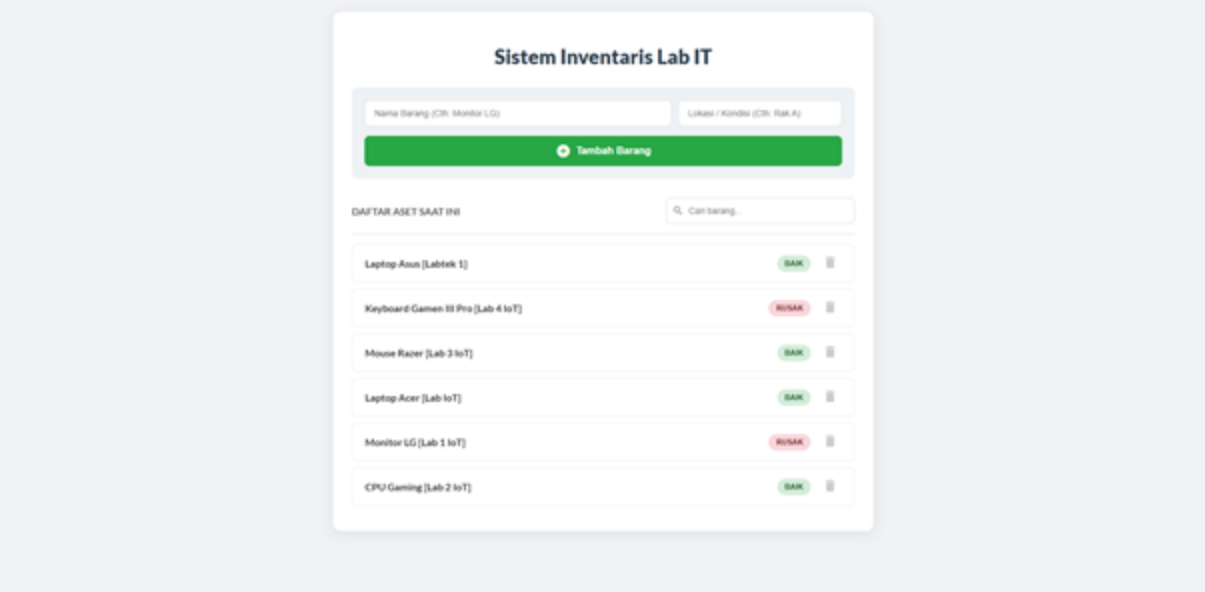
d. Fitur pencarian berdasarkan nama barang atau lokasi barang

## 2. Akses Sistem

Buka browser web dan akses URL sistem inventaris (https://three-tier-app-fe-1018358556426.us-central1.run.app/ ).

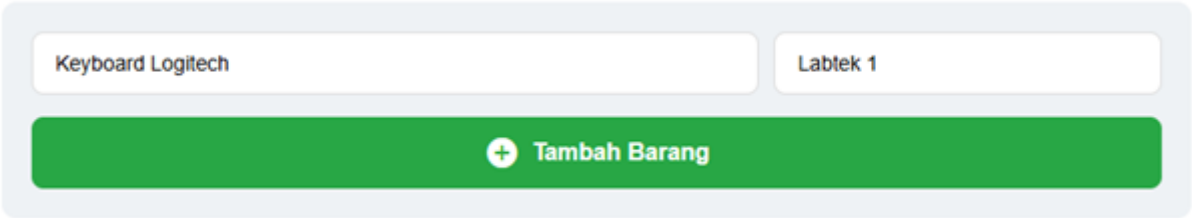
## 3. Dashboard Utama

Setelah mengakses URL, anda akan melihat dashboard seperti berikut.

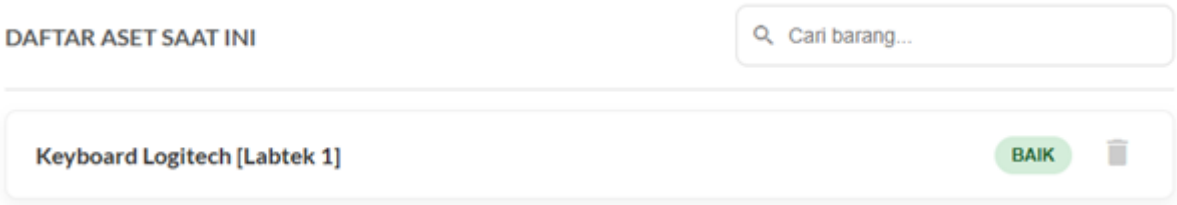



## 4. Pengelolaan Inventaris

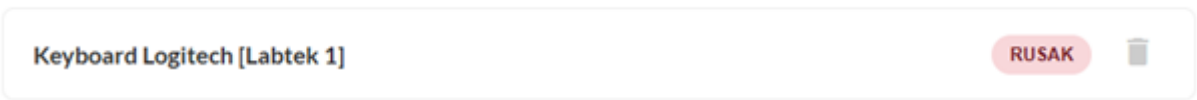
1. Untuk menambah barang baru, input nama barang pada kolom form dengan *placeholder* "Nama Barang" dan input lokasi dimana barang berada, lalu klik tombol "Tambah Barang"



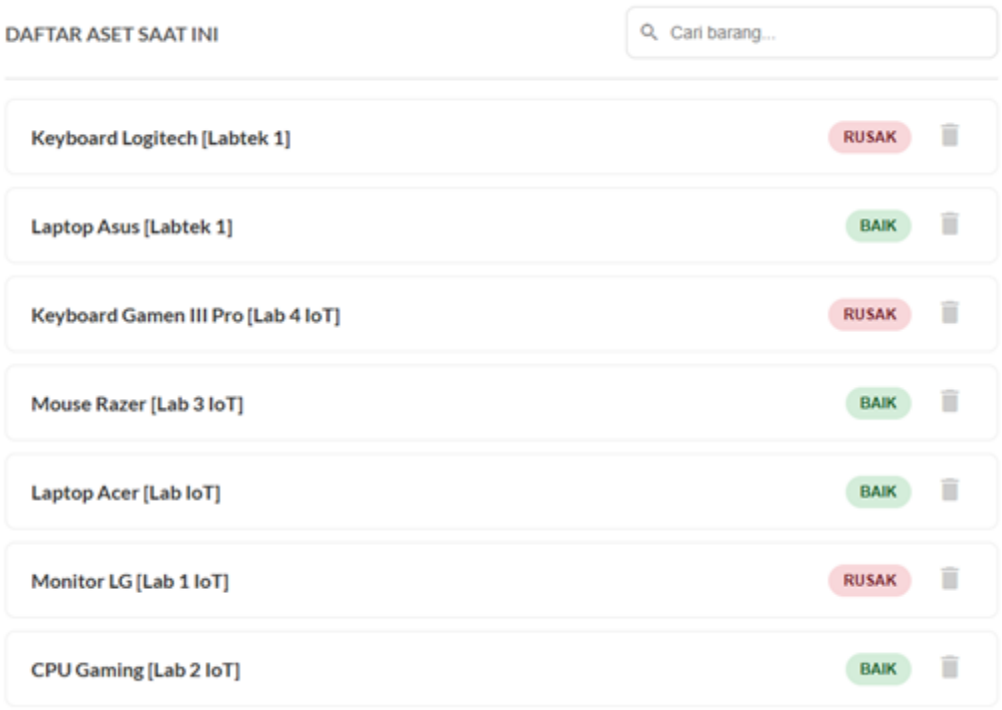

2. Setelah berhasil menambahkan barang, data akan muncul pada daftar aset

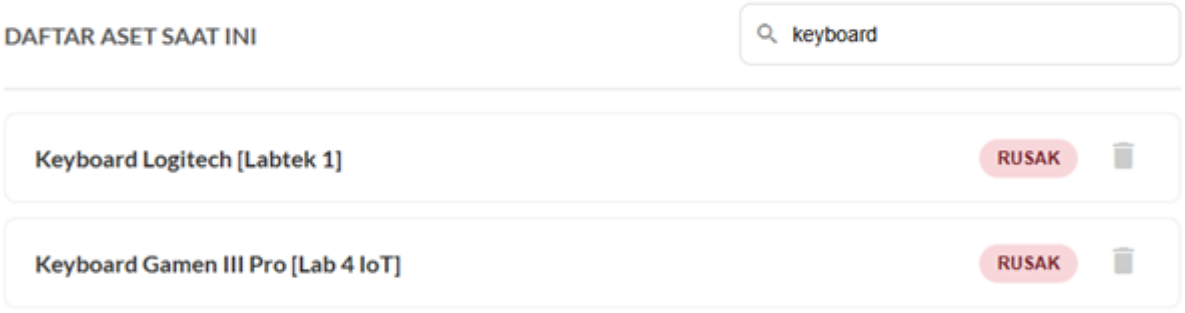


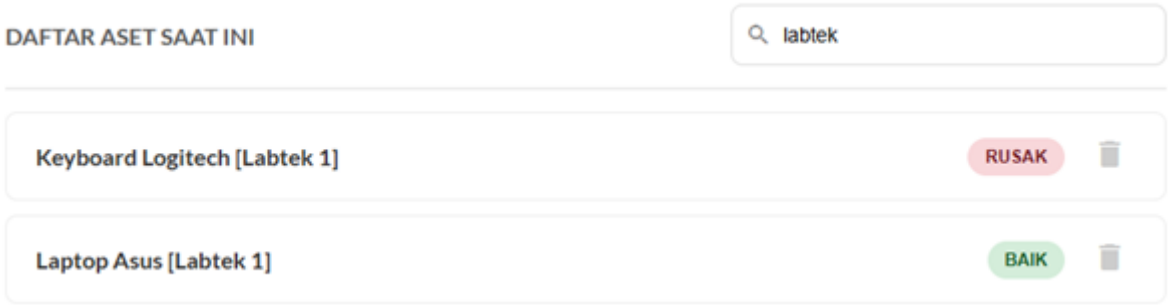

3. Untuk mengubah kondisi barang (Baik/Buruk) klik pada label barang saat ini (seperti pada contoh klik label baik) maka kondisi barang akan berubah



4. Untuk mencari barang, input data barang yang ingin dicari pada kolom pencarian dan format pencarian harus sesuai dengan nama barang atau lokasi barang



5. Untuk menghapus data barang, klik pada icon atau simbol tempat sampah, lalu akan muncul dialog dari browser dan klik OK