———

### Software Requirement Specification (SRS)

Project Name: 3D Building Volume Visualization on Google Maps

Version: 1.0

Prepared by: [Your Name / Your Team]

Date: [DD/MM/YYYY]

⸻

1. Introduction

1.1 Purpose

ซอฟต์แวร์นี้มีวัตถุประสงค์เพื่อ แสดงภาพสามมิติของอาคารและสิ่งปลูกสร้าง บนแผนที่ Google Maps โดยใช้ OpenGL และภาษา C พร้อมคำนวณขนาดอาคารตามกฎระเบียบ FAR (Floor Area Ratio) และ OSR (Open Space Ratio) ของแต่ละพื้นที่

นอกจากนี้ยังมีฟีเจอร์ เปรียบเทียบภาพผังอาคารที่ดึงจากแผนที่กับภาพถ่ายจริงของอาคารจากมุมมองถนน เพื่อช่วยในการวิเคราะห์และวางแผนการพัฒนาอสังหาริมทรัพย์

1.2 Scope

-แสดงโมเดล 3D ของอาคาร ที่สามารถสร้างได้ตามกฎหมายผังเมือง

-รองรับ Google Maps API เพื่อแสดงแผนที่และข้อมูลผังเมือง

-ให้ผู้ใช้เลือกตำแหน่งบนแผนที่ เพื่อคำนวณและแสดงโมเดลอาคาร

-คำนวณ FAR และ OSR อัตโนมัติ และแสดงผลลัพธ์

-รองรับการเปรียบเทียบภาพผังอาคาร กับ ภาพอาคารจากถนนจริง (Street View หรือ ภาพถ่ายผู้ใช้)

-พัฒนาโดยใช้ OpenGL และภาษา C

⸻

2. Functional Requirements

2.1 3D Building Volume Visualization

-ใช้ OpenGL และภาษา C ในการสร้างโมเดลสามมิติ

-แสดง Volume Space ของอาคารตามค่า FAR และ OSR

-รองรับการหมุน ซูม และปรับมุมมองโมเดล

2.2 Google Maps Integration

-ดึงข้อมูลแผนที่จาก Google Maps API

-ซ้อนทับข้อมูล ผังเมือง (Zoning Map) บนแผนที่

-รองรับการค้นหาตำแหน่งที่ดินโดยใช้ พิกัด GPS หรือที่อยู่

2.3 Building Size Calculation

-รับค่าพื้นที่ดิน (Land Area) จากผู้ใช้

-คำนวณขนาดอาคารที่สร้างได้โดยอัตโนมัติ:

-พื้นที่อาคารรวม = พื้นที่ดิน × FAR

-พื้นที่ว่างรอบอาคาร = พื้นที่ดิน × OSR

-แสดงผลขนาดอาคารและพื้นที่ว่างตามกฎหมาย

2.4 Image Comparison System

-ให้ผู้ใช้ เลือกตำแหน่งอาคารจากแผนที่

-ดึง ภาพผังอาคารจากแผนที่ดาวเทียม (Satellite View)

-เปรียบเทียบกับ ภาพอาคารจากถนน (Street View หรือ ภาพถ่ายของผู้ใช้เอง)

2.5 Data Storage & Export

-บันทึกผลการคำนวณและโมเดลอาคาร

-รองรับการ Export ข้อมูลเป็นไฟล์ PDF หรือ CSV

⸻

3. Non-Functional Requirements

3.1 Performance

-ใช้ OpenGL เพื่อให้การเรนเดอร์โมเดล 3D ลื่นไหลและตอบสนองรวดเร็ว

-รองรับ GPU Acceleration เพื่อเพิ่มความเร็วการประมวลผล (optional)

3.2 Usability

-อินเทอร์เฟซใช้งานง่าย รองรับทั้งนักพัฒนาอสังหาริมทรัพย์และผู้ใช้ทั่วไป

-รองรับการแสดงผลบน แพลตฟอร์ม Windows / Linux / macOS

3.3 Security (Optional)

-ปกป้องข้อมูล ภาพถ่ายและตำแหน่งที่ดิน ของผู้ใช้

-ใช้การเข้ารหัสข้อมูลเมื่อจัดเก็บและส่งข้อมูล

⸻

4. Technology Stack

ส่วนของระบบ และเทคโนโลยีที่ใช้

1. Frontend (3D Visualization) OpenGL + C
2. Backend C/C++ (SQLite หรือ PostgreSQL)\*
3. Mapping APIs Google Maps API
4. Image Processing OpenCV (สำหรับเปรียบเทียบภาพ)\*

\*(ไม่จำเป็นต้องมี)

⸻

5. Deliverables

1. แอปพลิเคชันแสดงโมเดลอาคาร 3D บน Google Maps

2. ระบบคำนวณขนาดอาคารตามกฎหมายผังเมือง

3. ฟีเจอร์แสดงภาพเปรียบเทียบภาพอาคารจากแผนที่และภาพถ่ายจริง

4. ระบบบันทึกและส่งออกข้อมูล (PDF/CSV)

⸻

6. Constraints & Assumptions

* ซอฟต์แวร์นี้ต้องเชื่อมต่อ Google Maps API เพื่อแสดงข้อมูลผังเมือง
* ข้อมูล FAR และ OSR ของแต่ละพื้นที่ต้องเป็นข้อมูลที่ได้รับจากแหล่งที่ถูกต้อง
* ผู้ใช้ต้องมี GPU ที่รองรับ OpenGL เพื่อการแสดงผลที่ราบรื่น

⸻

7. Future Enhancements (Optional)

* รองรับการ สร้างแบบจำลองอาคาร 3D ที่ซับซ้อนยิ่งขึ้น
* พัฒนาฟังก์ชัน AI วิเคราะห์และปรับปรุงแบบจำลองอาคารอัตโนมัติ
* รองรับการ นำเข้าไฟล์ CAD หรือ BIM

⸻

8. Approval & Sign-Off

-ลูกค้า / ผู้ใช้ปลายทาง: [ชื่อ]

-ทีมพัฒนา: [ชื่อทีม]

-วันที่อนุมัติ: [DD/MM/YYYY]

### Design Document

Project Name:

3D Building Volume Visualization on Google Maps

Version:

1.0

Prepared by:

[Your Name / Your Team]

---

1. Introduction

1.1 Purpose

The design document outlines the architectural and technical details required to implement the 3D visualization of building volumes on Google Maps. It includes specifications for integrating Google Maps API, calculating building dimensions based on FAR and OSR regulations, and comparing satellite imagery with street-level views using image processing techniques.

1.2 Scope

This document defines the system architecture, database schema, algorithms, and user interface design necessary to achieve the functional and non-functional requirements outlined in the SRS.

---

2. System Architecture

2.1 Overview

The system follows a modular architecture with distinct layers:

* Frontend: Handles 3D visualization using OpenGL.
* Backend: Manages calculations, data storage, and API interactions using C/C++ with SQLite/PostgreSQL.
* Mapping Integration: Utilizes Google Maps API for map rendering and zoning overlays.
* Image Processing: Employs OpenCV for image comparison tasks.

2.2 Component Diagram

Component Description

1. 3D Visualization

Renders 3D models of buildings using OpenGL.

1. Calculation Engine

Computes FAR and OSR values based on user input.

1. Mapping Module

Integrates Google Maps API to display maps and zoning overlays.

1. Image Comparison

Render images for building analysis.

1. Data Storage (Optimal)

Stores user data and results in SQLite/PostgreSQL databases.

---

3. Detailed Design

3.1 Frontend Design

* Technology: OpenGL + C++.
* Features:
  + Interactive 3D model rendering (rotate, zoom, adjust views).
  + User-friendly interface for selecting map locations.
  + Design Approach:
  + Use OpenGL shaders to optimize rendering performance.
  + Implement event-driven programming for user interactions.

3.2 Backend Design

* Technology: C/C++ with SQLite or PostgreSQL.
* Features:
  + FAR/OSR calculations based on user input.
  + Export functionality (PDF/CSV).
  + Design Approach:
  + Develop modular functions for calculations (e.g., `calculateFAR`, `calculateOSR`).
  + Use RESTful APIs for communication between frontend and backend.

3.3 Mapping Integration

* Technology: Google Maps API.
* Features:
  + Display satellite maps with zoning overlays.
  + GPS-based location search.
  + Design Approach:
  + Integrate Google Maps SDK for seamless map rendering.
  + Use layers to overlay zoning data on maps.

3.4 Image Processing

* Technology: Open CV and Rendering Engine.
* Features:
  + Extract building outlines from satellite imagery.
  + Compare extracted outlines with street-level images.
* Design Approach
  + Implement edge detection algorithms (e.g., Canny Edge Detection).
  + Use image alignment techniques for comparison.

---

4. Database Design

4.1 Schema Design

Database (Optional)

Table Name /Fields/Description

* Users /UserID, Name, Email / Stores user information.
* Projects /ProjectID, UserID, LandArea, FAR, OSR / Stores project data and calculations
* Images /ImageID, ProjectID, Type (Satellite/Street)/Stores uploaded images for comparison

---

5. Algorithms

5.1 FAR Calculation

Building Area} = Land Area x FAR

5.2 OSR Calculation

Open Space Area = Land Area x OSR

5.3 Image Comparison

Steps:

1. Extract building outlines from satellite images using OpenCV edge detection methods.

2. Align satellite images with street-level images using feature matching techniques (e.g., ORB or SIFT).

3. Compute differences using pixel-by-pixel comparison.

---

6. Technology StackComponent and Technology Used

1. Frontend
   * OpenGL + C
2. Backend
   * C/C++ with SQLite/PostgreSQL
3. Mapping APIs
   * Google Maps API
4. Image Processing
   * OpenCV

---

7. Deliverables

1. Fully functional application for visualizing building models in 3D.

2. Automated calculation system for FAR and OSR values.

3. Image comparison module integrated with mapping features.

4. Data export functionality supporting PDF/CSV formats.

---

8. Constraints & Assumptions

- Requires active internet connection for Google Maps API integration.

- Users must have GPUs compatible with OpenGL for smooth rendering.

- Accurate FAR/OSR data must be sourced from reliable authorities.

---