System Implemantation

Use Case Diagram dan Activity Diagram

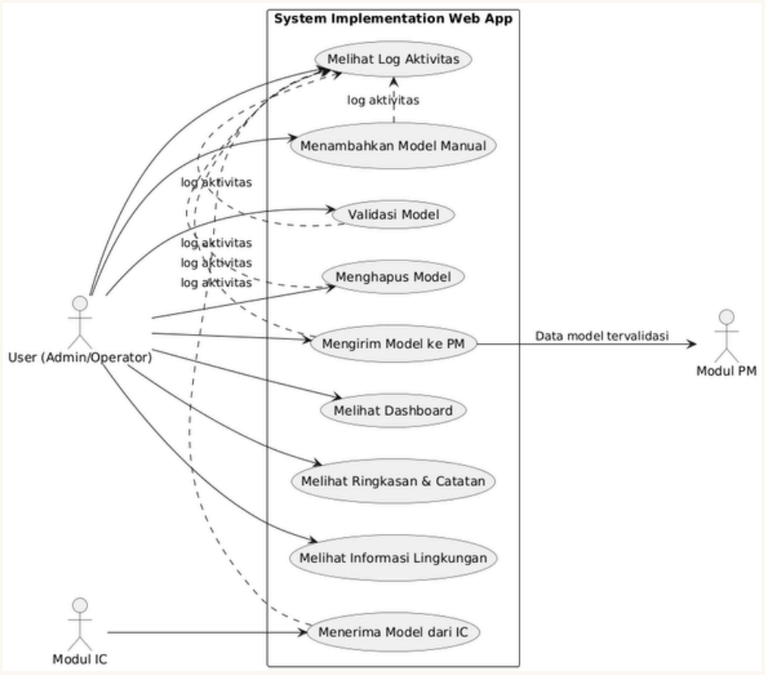
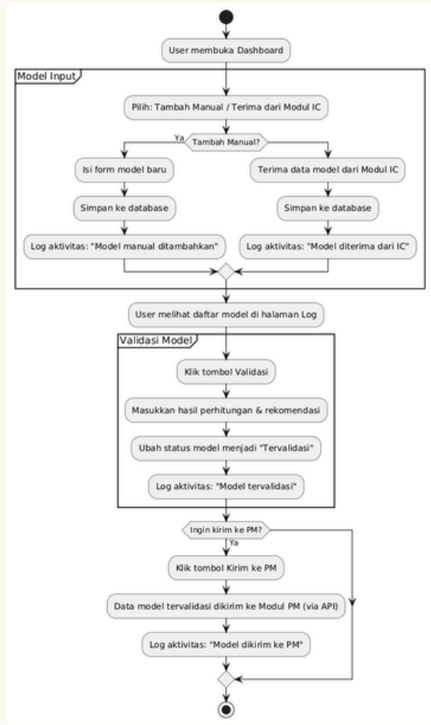
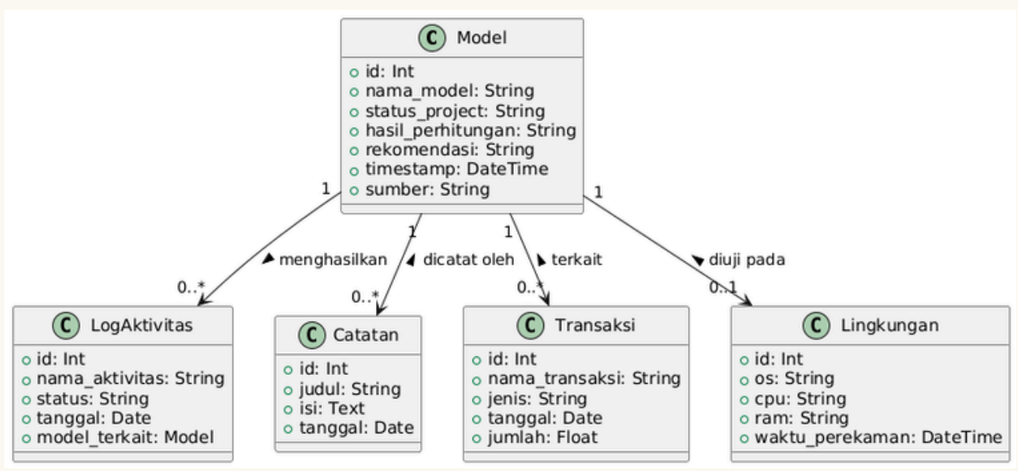


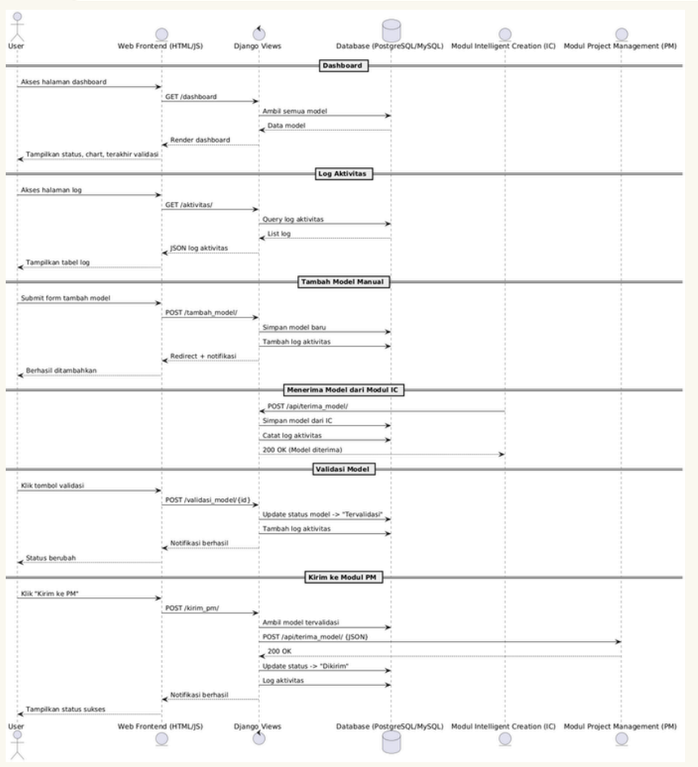
Diagram Use Case dan Activity pada gambar menggambarkan alur dan interaksi pengguna dengan sistem System Implementation Web App. Pada Use Case Diagram, aktor utama yaitu User (Admin/Operator) dapat melakukan berbagai aktivitas seperti melihat dan mencatat log aktivitas, mengelola model (menambah, memvalidasi, menghapus), mengirim model ke modul lain, serta melihat dashboard, ringkasan catatan, dan informasi lingkungan. Sementara itu, Activity Diagram menunjukkan alur proses pengguna dalam sistem, dimulai dari login, memilih modul yang akan dikelola, melakukan validasi atau penghapusan model, hingga mengirimkan data ke modul PM. Diagram ini membantu menggambarkan proses sistem secara terstruktur dan runtut untuk mendukung pengembangan aplikasi.



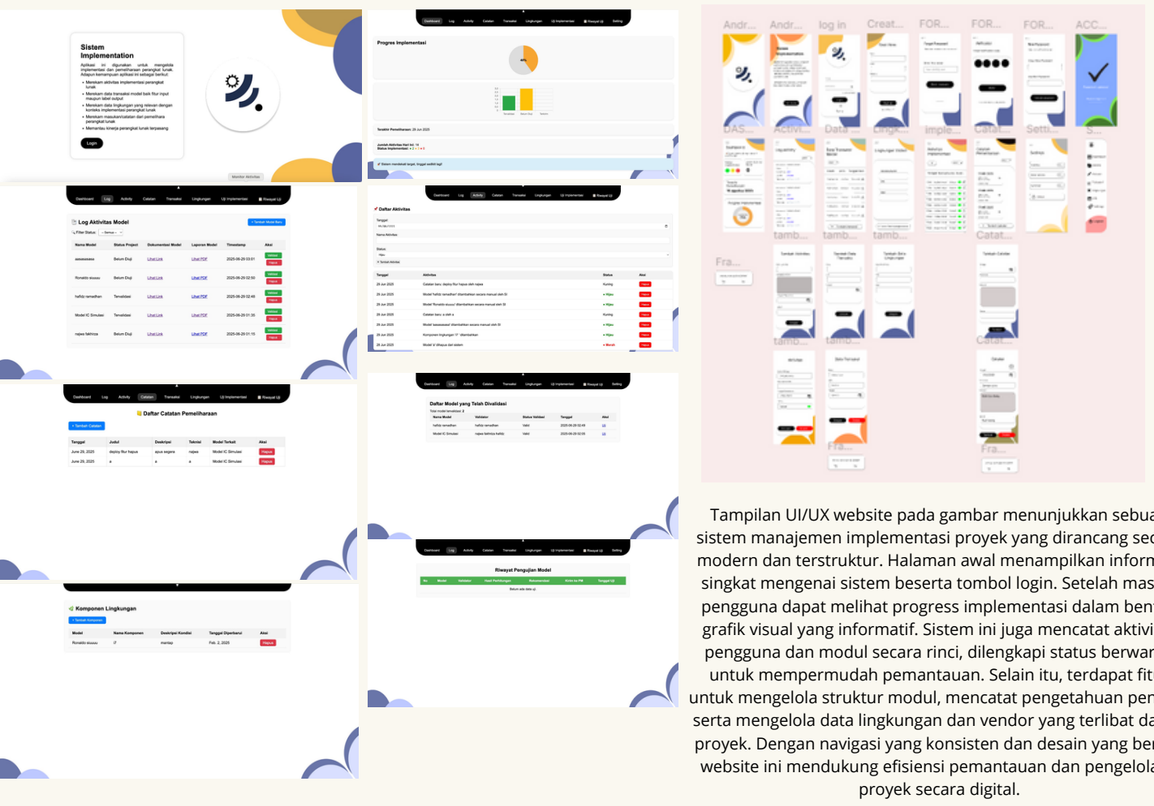
Class Diagram dan Sequence Diagram



Sistem ini dirancang untuk mengelola "Model" yang dapat berasal dari input manual atau Modul Intelligent Creation (IC). Fungsionalitas utamanya meliputi pengelolaan dashboard untuk melihat status model, pencatatan log aktivitas terperinci untuk setiap interaksi (seperti penambahan, validasi, dan pengiriman model), serta interaksi dengan Modul Project Management (PM) untuk model yang sudah divalidasi. Dari sisi struktur data, inti sistem adalah kelas Model yang memiliki atribut seperti nama, status proyek, hasil perhitungan, rekomendasi, dan sumber. Setiap Model dapat memiliki banyak LogAktivitas (mencatat setiap perubahan atau interaksi), Catatan (untuk informasi tambahan), dan Transaksi terkait, serta dapat diuji pada satu Lingkungan tertentu. Secara keseluruhan, sistem ini memfasilitasi siklus hidup model mulai dari pembuatan, validasi, hingga pengiriman ke modul lain, dengan pencatatan aktivitas yang komprehensif.



UI/UX Website dan mobile



Tampilan UI/UX website pada gambar menunjukkan sebuah sistem manajemen implementasi proyek yang dirancang secara modern dan terstruktur. Halaman awal menampilkan informasi singkat mengenai sistem beserta tombol login. Setelah masuk, pengguna dapat melihat progress implementasi dalam bentuk grafik visual yang informatif. Sistem ini juga mencatat aktivitas pengguna dan modul secara rinci, dilengkapi status berwarna untuk mempermudah pemantauan. Selain itu, terdapat fitur untuk mengelola struktur modul, mencatat pengetahuan penting, serta mengelola data lingkungan dan vendor yang terlibat dalam proyek. Dengan navigasi yang konsisten dan desain yang bersih, website ini mendukung efisiensi pemantauan dan pengelolaan proyek secara digital.

Fungsional dan non fungsional

✅ Kebutuhan Fungsional

1. Autentikasi Pengguna – Login dan logout sistem.
2. Manajemen Proyek – Tambah, ubah, hapus, dan lihat daftar proyek.
3. Manajemen Modul – Kelola modul dalam proyek (tambah/hapus).
4. Manajemen Aktivitas – Tambah aktivitas dengan status dan tenggat waktu.
5. Manajemen Kinerja – Input dan pantau kinerja aktivitas.
6. Manajemen Transaksi – Simpan dan atur data proyek dan aktivitas.
7. Manajemen Lingkungan – Simpan info teknis proyek (RAM, OS, CPU, dll).
8. UI/UX Web & Mobile – Tampilan responsif di berbagai perangkat.

⚙️ Kebutuhan Non-Fungsional

1. Kinerja – Respon cepat meskipun banyak pengguna dan data.
2. Kemudahan Penggunaan – Antarmuka mudah dipahami dan konsisten.
3. Portabilitas – Dapat diakses dari tablet, dan smartphone.

Codingan web dan mobile

API PROJECT

```python
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import (
    AktivitasViewSet, CatatanViewSet, TransaksiViewSet,
    LogAktivitasViewSet, LingkunganViewSet
)

router = DefaultRouter()
router.register(r'aktivitas', AktivitasViewSet)
router.register(r'catatan', CatatanViewSet)
router.register(r'transaksi', TransaksiViewSet)
router.register(r'log', LogAktivitasViewSet)
router.register(r'lingkungan', LingkunganViewSet)

urlpatterns = [
    path('api/', include(router.urls)),
]
```

BackEnd

```python
class AktivitasViewSet(viewsets.ModelViewSet):
    queryset = Aktivitas.objects.all()
    serializer_class = AktivitasSerializer

class CatatanViewSet(viewsets.ModelViewSet):
    queryset = Catatan.objects.all()
    serializer_class = CatatanSerializer

class TransaksiViewSet(viewsets.ModelViewSet):
    queryset = Transaksi.objects.all()
    serializer_class = TransaksiSerializer

class LogAktivitasViewSet(viewsets.ModelViewSet):
    queryset = LogAktivitas.objects.all()
    serializer_class = LogAktivitasSerializer

class LingkunganViewSet(viewsets.ModelViewSet):
    queryset = Lingkungan.objects.all()
    serializer_class = LingkunganSerializer


class SystemImplementationAPIView(APIView):
    parser_classes = [MultiPartParser, FormParser, JSONParser]

    def get(self, request):
        data = SystemImplementation.objects.all().order_by('-timestamp')
        serializer = SystemImplementationSerializer(data, many=True)
        return Response(serializer.data)

    def post(self, request):
        file_laporan = request.FILES.get('laporan_model')
        data = request.data

        payload = {
            "no": data.get("no"),
            "nama_model": data.get("nama_model"),
            "status_project": data.get("status_project"),
```

Kode ini merupakan bagian dari backend aplikasi menggunakan Django REST Framework. Beberapa ModelViewSetdisediakan untuk menangani operasi CRUD pada model seperti Aktivitas, Catatan, Transaksi, LogAktivitas, dan Lingkungan, masing-masing menggunakan serializer yang sesuai.

API MOBILE

```kotlin
interface ApiService {

    1 Usage
    @POST( value = "login/")
    suspend fun login(
        @Body request: LoginRequest
    ): Response<LoginResponse>

    1 Usage
    @GET( value = "api/system-implementation/")
    suspend fun getDashboardItems(
        @Header( value = "Authorization") token: String
    ): Response<List<DashboardItem>>

    1 Usage
    @GET( value = "api/logactivity/")
    suspend fun getLogActivity(
        @Header( value = "Authorization") token: String
    ): Response<List<LogActivity>>

    1 Usage
    @GET( value = "api/get-model-dari-ic/")
    suspend fun getModelDariIC(
        @Header( value = "Authorization") token: String
    ): Response<List<ModelDariIC>>
```

MainActivity

```kotlin
package com.example.najwahafidz

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.core.view.WindowCompat
import com.example.najwahafidz.ui.AppNavHost
import com.example.najwahafidz.ui.theme.NajwahafidzTheme

1 Usage
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Optional: membuat konten aplikasi di-render di bawah status bar (
        WindowCompat.setDecorFitsSystemWindows(window, decorFitsSystemWindows

        setContent {
            NajwahafidzTheme {
                // Ini akan memulai navigasi: Splash → Landing → Login → Mai
                AppNavHost()
            }
        }
    }
```

Sarializers

```python
from rest_framework import serializers
from .models import Aktivitas, Catatan, Transaksi, LogAktivitas, Lingkungan,SystemImplementation,ManagementsSI

class AktivitasSerializer(serializers.ModelSerializer):
    class Meta:
        model = Aktivitas
        fields = '__all__'

class CatatanSerializer(serializers.ModelSerializer):
    class Meta:
        model = Catatan
        fields = '__all__'

#ke ridho
class ManagementsSISerializer(serializers.ModelSerializer):
    class Meta:
        model = ManagementsSI
        fields = '_all_'

class TransaksiSerializer(serializers.ModelSerializer):
    class Meta:
        model = Transaksi
        fields = '__all__'

class LogAktivitasSerializer(serializers.ModelSerializer):
    class Meta:
        model = LogAktivitas
        fields = '__all__'

class LingkunganSerializer(serializers.ModelSerializer):
    class Meta:
        model = Lingkungan
        fields = '__all__'
```

View

```python
@csrf_exempt
def login_view(request):
    if request.method == 'POST':
        data = json.loads(request.body.decode('utf-8'))
        username = data.get('username')
        password = data.get('password')

        user = authenticate(username=username, password=password)
        if user:
            token, _ = Token.objects.get_or_create(user=user)
            return JsonResponse({
                'success': True,
                'token': token.key,
                'username': user.username,
                'email': user.email,
            })
        else:
            return JsonResponse({'success': False, 'error': 'Invalid credentials'}, status=401)

    return JsonResponse({'error': 'Only POST allowed'}, status=405)

def ringkasan_hasil_uji(request):
    from .models import HasilUjiModel
    hasil_list = HasilUjiModel.objects.all().order_by('-tanggal_uji')
    return render(request, 'myapp/ringkasan_uji.html', {'hasil_list': hasil_list})


def signup_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
```

Model

```python
from django.db import models


class Catatan(models.Model):
    model_terkait = models.ForeignKey('SystemImplementation', on_delete=models.CASCADE, null=True, blank=True)
    tanggal = models.DateField(auto_now_add=True)
    judul = models.CharField(max_length=255)
    deskripsi = models.TextField()
    teknisi = models.CharField(max_length=255)

    def __str__(self):
        return f"{self.judul} - {self.tanggal}"

class Aktivitas(models.Model):
    STATUS_CHOICES = [
        ('Hijau', 'Hijau'),
        ('Oranye', 'Oranye'),
        ('Merah', 'Merah'),
    ]
    tanggal = models.DateField()
    nama_aktivitas = models.CharField(max_length=255)
    status = models.CharField(max_length=50, choices=STATUS_CHOICES)

    def __str__(self):
        return f"{self.nama_aktivitas} - {self.status}"


class Transaksi(models.Model):
    model = models.CharField(max_length=100)
    jenis = models.CharField(max_length=100)
```