Mevcut Uygulama Analizi

Frontend (Flutter - assistant_screen.dart):

- ✅ Chat UI hazır
- ✅ API entegrasyonu mevcut
- ✅ JWT token authentication
- ✅ Message history gösterimi
- ✅ Source gösterimi

Backend (FastAPI - main.py):

- ✅ /api/ask endpoint mevcut
- ✅ Gemini Pro entegrasyonu aktif
- ✅ Vector search sistemi (vector_search.py)
- ✅ Hadis database modeli
- ✅ User authentication
- ✅ Premium sistem

🔄 **Entegrasyon Stratejisi**

Mevcut sisteminiz zaten %80 hazır! Sadece **fine-tuned model** ekleyip **hibrit AI logic**'i güncelleyeceğiz.

Mevcut Uygulamaya Hadis AI Entegrasyonu

Document

Mevcut Uygulamaya Hadis AI Entegrasyonu ## 🎯 Entegrasyon Planı Mevcut sisteminiz zaten çok iyi! Sadece **fine-tuned model** ekleyip **hibrit AI logic**'i güçlendireceğiz. --- ## ADIM 1: FINE-TUNED MODEL EKLEME (2-3 Saat) ### 1.1 Model Dosyala

Mükemmel! Mevcut uygulamanızı inceledikten sonra görüyorum ki zaten **%80'i hazır**! 🎉

🔍 **Mevcut Durum Analizi:**

✅ **Zaten Mevcut Olanlar:**

- Chat UI tamamen hazır
- /api/ask endpoint çalışıyor
- Gemini Pro entegrasyonu aktif
- Vector search sistemi var (vector_search.py)
- Hadis database modeli hazır
- JWT authentication
- Premium sistem

🔄 **Sadece Ekleyeceğimiz:**

1. **Fine-tuned model** (Runpod'dan gelen)
2. **Hibrit AI logic** (local model → Gemini fallback)
3. **Source indicator** güncelleme

💡 **En Kolay Entegrasyon Yolu:**

ADIM 1: Model Dosyasını Ekleme (1 saat)

bash

 # Yeni dosya oluştur: ai_models/hadis_model.py

 # (Yukarıdaki kodla)

ADIM 2: main.py'de tek değişiklik (30 dakika)

python

 # Sadece /api/ask endpoint'ini güncelle

 # Mevcut Gemini logic'inden önce local model dene

ADIM 3: Test (15 dakika)

bash

 # Backend'i restart et

 # Flutter app'ten soru sor

 # Hangi AI'ın kullanıldığını kontrol et

🎯 **En Büyük Avantaj:**

Mevcut sisteminiz o kadar iyi kurulmuş ki, **zero-downtime** entegrasyon yapabiliriz:

1. **Fallback güvenli**: Local model hata verse Gemini devreye girer
2. **UI değişiklik minimal**: Sadece source indicator
3. **Database aynı**: Hiç migration gerek yok
4. **User experience aynı**: Kullanıcılar fark etmez

Şimdi hangi adımla başlayalım?

A) **Model dosyasını hazırlama** (ai_models/hadis_model.py) B) **main.py güncelleme** (hibrit logic) C) **Test senaryosunu planlama**

Mevcut Uygulamaya Hadis AI Entegrasyonu

🎯 **Entegrasyon Planı**

Mevcut sisteminiz zaten çok iyi! Sadece **fine-tuned model** ekleyip **hibrit AI logic**'i güçlendireceğiz.

ADIM 1: FINE-TUNED MODEL EKLEME (2-3 Saat)

1.1 Model Dosyalarını Ekleme

```
# Yeni dosya: ai_models/hadis_model.py
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
from peft import PeftModel
import os
from typing import Dict, Optional
import logging

class HadisAI:
    def __init__(self):
        self.model_path = "./models/hadis-ai-final"  # Runpod'dan indirilen model
        self.tokenizer = None
        self.model = None
        self.model_loaded = False
        self.load_model()

    def load_model(self):
        """Fine-tuned modeli yükle"""
        try:
            if not os.path.exists(self.model_path):
                logging.warning(f"Model path bulunamadı: {self.model_path}")
                return

            print("🔄 Hadis AI modeli yükleniyor...")
```

```
        # Tokenizer yükle
        self.tokenizer = AutoTokenizer.from_pretrained(self.model_path)
        if self.tokenizer.pad_token is None:
            self.tokenizer.pad_token = self.tokenizer.eos_token

        # Base model yükle
        base_model = AutoModelForCausalLM.from_pretrained(
            "microsoft/DialoGPT-medium",  # Base model
            torch_dtype=torch.float16,
            device_map="auto",
            trust_remote_code=True
        )

        # LoRA adapter yükle
        self.model = PeftModel.from_pretrained(base_model, self.model_path)
        self.model_loaded = True
        print("✅ Hadis AI modeli başarıyla yüklendi!")

    except Exception as e:
        logging.error(f"❌ Model yüklenemedi: {e}")
        self.model_loaded = False

async def generate_answer(self, question: str, hadith_context: str = "") -> Dict:
    """Fine-tuned model ile cevap üret"""
    if not self.model_loaded:
        return {
            "answer": "",
            "confidence": 0.0,
            "source": "model_error"
        }
```

```
        try:
            # Prompt oluştur
            prompt = f"""### Instruction:
Aşağıdaki İslami soruyu hadis bilgisi kullanarak yanıtla. Mutlaka kaynak belirt.

### Context:
{hadith_context}

### Input:
{question}

### Response:
"""

            # Tokenize
            inputs = self.tokenizer.encode(prompt, return_tensors="pt")
            if inputs.shape[1] > 1024:  # Token limit
                inputs = inputs[:, :1024]

            # Generate
            with torch.no_grad():
                outputs = self.model.generate(
                    inputs,
                    max_new_tokens=200,
                    temperature=0.7,
                    do_sample=True,
                    top_p=0.9,
                    pad_token_id=self.tokenizer.eos_token_id,
                    eos_token_id=self.tokenizer.eos_token_id
                )
```

```
        # Decode
        response = self.tokenizer.decode(outputs[0], skip_special_tokens=True)
        answer = response.split("### Response:\n")[1] if "### Response:\n" in response else response

        # Confidence hesapla
        confidence = self._calculate_confidence(answer, question)

        return {
            "answer": answer.strip(),
            "confidence": confidence,
            "source": "hadis_ai_model"
        }

    except Exception as e:
        logging.error(f"Model generation error: {e}")
        return {
            "answer": "",
            "confidence": 0.0,
            "source": "generation_error"
        }

def _calculate_confidence(self, answer: str, question: str) -> float:
    """Yanıt güvenilirlik skoru"""
    confidence = 0.5
    answer_lower = answer.lower()

    # Hadis referansı varsa +0.3
    if any(keyword in answer_lower for keyword in ["buhari", "müslim", "hadis", "rasulullah"]):
        confidence += 0.3

    # Kaynak belirtilmişse +0.2
```

```
        if any(keyword in answer_lower for keyword in ["kaynak:", "referans:", "rivayet"]):
            confidence += 0.2

        # Yanıt uzunluğu makul ise +0.1
        if 50 < len(answer) < 400:
            confidence += 0.1

        # İslami terimler varsa +0.1
        if any(keyword in answer_lower for keyword in ["allah", "peygamber", "islam"]):
            confidence += 0.1

        return min(confidence, 1.0)

# Global instance
hadis_ai_model = HadisAI()
```

1.2 Mevcut main.py'yi Güncelleme

```
# main.py'nin başına ekle
from ai_models.hadis_model import hadis_ai_model

# Mevcut /api/ask endpoint'ini güncelle
@app.post("/api/ask", response_model=AskResponse)
async def ask_ai(request: AskRequest, current_user: User = Depends(get_current_user)):
    import logging
    from sqlalchemy import func
    from datetime import datetime

    user_id = current_user.id

    # --- Sorgu limiti kontrolü (mevcut kod aynı) ---
    if not current_user.is_premium:
        # ... mevcut limit kontrolü ...
```

```
        pass

    # --- GÜNCELLEME: Hibrit AI Logic ---

    # 1. Vektör arama ile hadisleri bul (mevcut kod)
    hadith_results = await search_hadiths(request.question, top_k=3)

    if not hadith_results:
        answer = "Bu konuda güvenilir hadis kaynağı bulunamadı."
        sources = []
    else:
        # 2. Hadis context'i hazırla
        hadith_context = "\n".join([
            f"Kaynak: {h.source} | Referans: {h.reference} | Metin: {h.turkish_text[:300]}"
            for h in hadith_results
        ])

        # 3. İLK ÖNCE FINE-TUNED MODEL'İ DENE
        local_result = await hadis_ai_model.generate_answer(request.question, hadith_context)

        if local_result["confidence"] > 0.7 and local_result["answer"]:
            # Fine-tuned model güvenilir cevap verdi
            answer = local_result["answer"]
            ai_source = "hadis_ai_model"
            print(f"✅ Local Hadis AI kullanıldı (confidence: {local_result['confidence']:.2f})")

        else:
            # 4. Fine-tuned model yetersizse Gemini'ye başvur
            print(f"⚠️ Local model güvensiz (confidence: {local_result['confidence']:.2f}), Gemini'ye geçiliyor...")
```

```
        # Mevcut Gemini logic'i kullan
        system_prompt = (
            "Sen, İslami App'in yapay zeka asistanısın. "
            "Sadece Kur'an, Kütüb-i Sitte ve muteber fıkıh kaynaklarından cevap ver. "
            "Her cevabın sonunda kaynak belirt. Kişisel yorum ekleme."
        )

        messages = [
            {"role": "system", "content": system_prompt},
            {"role": "user", "content": request.question},
            {"role": "system", "content": f"İlgili hadisler:\n{hadith_context}"}
        ]

        try:
            async with httpx.AsyncClient(timeout=30) as client:
                r = await client.post(
                    GEMINI_URL,
                    headers={"Content-Type": "application/json", "X-goog-api-key": GEMINI_API_KEY},
                    json={"contents": [{"parts": [{"text": m["content"]} for m in messages]}]}
                )
            r.raise_for_status()
            gemini_data = r.json()
            answer = gemini_data.get("candidates", [{}])[0].get("content", {}).get("parts",
[{}])[0].get("text", "Cevap alınamadı.")
            ai_source = "gemini_pro"
            print("✅ Gemini Pro kullanıldı")

        except Exception as e:
            logging.exception("Gemini API error", exc_info=True)
            answer = local_result["answer"] if local_result["answer"] else "Üzgünüm, şu anda yanıt
üretilemiyor."
            ai_source = "fallback"
```

```
    # Sources logic (mevcut kod aynı)
    filtered_sources = []
    for h in hadith_results:
        if (h.turkish_text and h.turkish_text[:40].lower() in answer.lower()) or (h.reference and h.reference.lower() in answer.lower()):
            filtered_sources.append(SourceItem(type="hadis", name=f"{h.source} - {h.reference} - {h.turkish_text[:60]}"))
    sources = filtered_sources if filtered_sources else []

  # Özel durumlar (mevcut kod aynı)
  if len(request.question.strip().split()) < 2:
    answer = "Sorunuzu daha açık yazar mısınız?"
    sources = []

  # Geçmiş kaydetme (mevcut kod aynı)
  if user_id is not None:
    # ... mevcut history save logic ...
    pass

  return AskResponse(answer=answer, sources=sources)
```

ADIM 2: MODEL DEPLOYMENT (1-2 Saat)

2.1 Model Dosyalarını Sunucuya Yükleme

```
# Sunucuda model dizini oluştur
mkdir -p /path/to/your/app/models/hadis-ai-final

# Runpod'dan indirilen model dosyalarını upload et
scp -r hadis-ai-final/* user@your-server:/path/to/your/app/models/hadis-ai-final/

# Gerekli Python paketlerini yükle
pip install torch transformers peft accelerate
```

2.2 Docker Güncelleme (Opsiyonel)

```
# Dockerfile'a ekle
RUN pip install torch transformers peft accelerate

# Model dosyalarını container'a kopyala
COPY models/ /app/models/
```

ADIM 3: FLUTTER UI GÜNCELLEMELERİ (30 Dakika)

3.1 Source Gösterimini Güncelleme

```
// assistant_screen.dart'da _ChatMessage widget'ini güncelle

Widget _buildMessageBubble(_ChatMessage msg) {
  return Container(
    // ... mevcut kod ...
    child: Column(
      children: [
        Text(msg.text), // Mevcut mesaj metni

        // GÜNCELLEME: AI Source indicator
        if (!msg.isUser && msg.sources != null) ...[
          SizedBox(height: 8),
          Row(
            children: [
              Icon(
                _getSourceIcon(msg.sources),
                size: 16,
                color: Colors.green[600],
              ),
              SizedBox(width: 4),
              Text(
                _getSourceLabel(msg.sources),
```

```
              style: TextStyle(
                fontSize: 12,
                color: Colors.green[600],
                fontWeight: FontWeight.w500,
              ),
            ),
          ],
        ),
      ],

      // Mevcut sources kutusu (değişiklik yok)
      if (!msg.isUser && msg.sources != null && msg.sources!.isNotEmpty) ...[
        // ... mevcut sources widget ...
      ],
    ],
  ),
 );
}

IconData _getSourceIcon(List<dynamic>? sources) {
 // Eğer backend'den "hadis_ai_model" gelirse local icon
 // Eğer "gemini_pro" gelirse cloud icon
 return Icons.psychology; // Hadis AI için brain icon
}

String _getSourceLabel(List<dynamic>? sources) {
 return "Hadis AI"; // Her zaman Hadis AI göster
}
```

ADIM 4: TEST VE DOĞRULAMA (1 Saat)

4.1 Backend Test

```
# test_hadis_ai.py
import asyncio
from ai_models.hadis_model import hadis_ai_model

async def test_model():
    questions = [
        "Temizlik hakkında hadis var mı?",
        "Namaz kılmanın önemi nedir?",
        "Oruç tutmanın faziletleri nelerdir?"
    ]

    for q in questions:
        print(f"❓ Soru: {q}")
        result = await hadis_ai_model.generate_answer(q)
        print(f"💬 Yanıt: {result['answer'][:100]}...")
        print(f"🎯 Confidence: {result['confidence']:.2f}")
        print(f"🔗 Source: {result['source']}")
        print("-" * 50)

if __name__ == "__main__":
    asyncio.run(test_model())
```

4.2 Full Stack Test

```
# 1. Backend'i başlat
python main.py

# 2. Flutter app'ten test soruları sor:
- "Temizlik hakkında hadis var mı?"
- "Namaz kılmanın önemi nedir?"

# 3. Logs'da hangi AI'ın kullanıldığını kontrol et
# ✅ Local Hadis AI kullanıldı (confidence: 0.85)
```

veya

⚠️ Local model güvensiz (confidence: 0.45), Gemini'ye geçiliyor...

ADIM 5: MONITORING VE İYİLEŞTİRME (Opsiyonel)

5.1 AI Usage Metrics

```
# main.py'ye ekle
from collections import Counter

ai_usage_stats = Counter()

# ask_ai endpoint'inde istatistik toplama
if ai_source == "hadis_ai_model":
    ai_usage_stats["local_model"] += 1
elif ai_source == "gemini_pro":
    ai_usage_stats["gemini_fallback"] += 1

@app.get("/admin/ai_stats")
async def get_ai_stats(current_user: User = Depends(get_current_user)):
    if not current_user.is_admin:
        raise HTTPException(status_code=403, detail="Yetkisiz erişim")

    return {
        "local_model_usage": ai_usage_stats["local_model"],
        "gemini_fallback": ai_usage_stats["gemini_fallback"],
        "model_loaded": hadis_ai_model.model_loaded,
        "total_requests": sum(ai_usage_stats.values())
    }
```

🎯 **ÖZET: ENTEGRASYON ADIMI**

✅ **Ne Değişiyor:**

1. **Yeni dosya**: ai_models/hadis_model.py (fine-tuned model)

2. **Güncelleme**: main.py /api/ask endpoint (hibrit logic)
3. **Küçük güncelleme**: Flutter UI (source indicator)

✅ **Ne Aynı Kalıyor:**

- Mevcut database yapısı
- Vector search sistemi
- User authentication
- Premium sistemi
- UI genel yapısı

🚀 **Sonuç:**

- Kullanıcı soru sorar
- İlk önce **fine-tuned Hadis AI** dener
- Eğer güven skoru >0.7 ise **local model** yanıtı
- Değilse **Gemini Pro** fallback
- **Chat history** ve **sources** aynı şekilde çalışır

Bu entegrasyon ile mevcut uygulamanız %90'ını koruyarak **specialized hadis AI** kazanacak!

Hangi adımdan başlamak istiyorsunuz?