گزارش نهایی پروژه سامانه اتوماسیون پلیس

1. اعضای تیم و مسئولیت‌ها

1. رضا اسلامی ابیانه: طراحی و پیاده‌سازی بک‌اند با Django/DRF، مدل‌سازی موجودیت‌ها، RBAC، JWT، Swagger، تست‌های بک‌اند.
2. سیدامیرحسین موسوی‌فرد: طراحی و پیاده‌سازی فرانت‌اند با React + Vite، مدیریت State، اتصال API، صفحات اصلی، داشبورد ماژولار.
3. سیداحمد موسوی‌اول: یکپارچه‌سازی فرانت و بک، Docker Compose، تست نهایی و مستندسازی، رفع باگ‌های سناریویی.
4. بازه زمانی کلی: از تحلیل نیازمندی تا تکمیل فازهای پرونده، شواهد، تحقیق، محاکمه، پاداش، پرداخت.

2. قراردادهای توسعه

1. نام‌گذاری:

- بک‌اند: `snake_case` برای فیلدها و توابع، نام مدل‌ها با `PascalCase`.
- فرانت‌اند: کامپوننت‌ها با `PascalCase`، هوک‌ها/توابع با `camelCase`.
- Endpointها: RESTful و معنادار مثل `/api/cases/cases/` و `/api/payments/bail/`.

3. مدیریت پروژه و تقسیم‌بندی‌های منطقی

1. کارها به بخش‌های اصلی تقسیم شد:

- احراز هویت و نقش‌ها
- تشکیل پرونده (Complaint/Crime Scene)
- شواهد
- تخته کارآگاه و روند بررسی مظنون
- بازجویی و امتیازدهی
- محاکمه
- تحت پیگیری شدید
- پاداش
- پرداخت وثیقه و جریمه (اختیاری)

2. هر بخش به تسک‌های کوچک‌تر شکسته شد و به‌صورت تدریجی پیاده‌سازی شد.
3. اولویت اجرا:

- ابتدا بک‌اند پایدار و API
- سپس UI
- سپس اصلاح permission و سناریوهای واقعی
- در نهایت توسعه UX - سپس تست و ریزه‌کاری Docker

4. موجودیت‌های کلیدی سامانه و دلیل وجود آن‌ها

1. `User`:

- کاربر پایه سامانه با اطلاعات هویتی یکتا.

2. `Role`، `Permission`، `UserRole`:

- پیاده‌سازی RBAC پویا برای افزودن/حذف نقش بدون تغییر کد.

3. `Case`:

- هسته اصلی گردش کار پرونده با منبع، شدت جرم، وضعیت و افراد درگیر.

4. `ComplaintSubmission`:

- مدیریت چرخه ثبت شکواییه و بازگشت به شاکی/کارآموز/افسر.

5. `CaseComplainant`، `CaseWitness`، `CaseLog`:

- ثبت چند شاکی، شاهدان، و تاریخچه تصمیم‌ها.

6. موجودیت‌های شواهد (شاهدی، زیستی/پزشکی، خودرو، مدارک هویتی، سایر):

- پوشش کامل انواع evidence طبق سند.

7. `DetectiveBoard`، `BoardNode`، `BoardEdge`:

- پیاده‌سازی تخته کارآگاه با اتصال مدارک و تحلیل ارتباطات.

8. `Suspect`، `SuspectSubmission`، `Interrogation`، `Notification`:

- مدیریت مظنون، تایید سرگروهبان، بازجویی و اعلان‌ها.

9. `CourtSession`:

- مدیریت محاکمه و ثبت رای برای هر مظنون.

10. `Tip` / `Reward`:

- ثبت اطلاعات مردمی، بررسی افسر/کارآگاه، صدور کد یکتا و استعلام.

11. `BailPayment`:

- مدیریت پرداخت وثیقه/جریمه و اتصال به درگاه پرداخت.

5. پکیج‌های NPM مورد 6 (حداکثر استفاده‌شده)

1. `react-dom`:

- رندر اپ React در مرورگر.

2. `react-router-dom`:

- مسیردهی SPA برای صفحات ماژولار.

3. `axios`:

- ارتباط با API و Backend و مدیریت ساده‌تر درخواست/پاسخ.

4. `html2canvas`:

- خروجی تصویری از Detective Board.

6. نمونه کدهای تولیدشده با هوش مصنوعی

1. نمونه منطق پرداخت و Verify در بک‌اند:

```
payload = {
    "merchant_id": merchant_id,
    "amount": int(obj.amount),
    "description": f"Bail/Fine payment for case #{obj.case_id} suspect #{obj.suspect_id}",
    "callback_url": callback_url,
}
result = zarinpal_post(settings.ZARINPAL_REQUEST_URL, payload)
```

2. نمونه اعتبارسنجی قانون پرداخت:

```
if suspect.status == Suspect.Status.ARRESTED:
    if case.severity not in [Case.Severity.LEVEL_2, Case.Severity.LEVEL_3]:
        raise ValidationError("Only level 2 and level 3 arrested suspects are eligible.")
elif suspect.status == Suspect.Status.CRIMINAL:
    if case.severity != Case.Severity.LEVEL_3 or not sergeant_approved:
        raise ValidationError("Sergeant approval is required for level 3 criminal release.")
```

3. نمونه UI فیلتر مظنون بر اساس پرونده:

```
const filteredSuspects = useMemo(() => {
  if (!selectedCaseId) return []
  return suspects.filter((s) => Number(s.case) === selectedCaseId)
}, [suspects, selectedCaseId])
```

7. قوت‌ها و ضعف‌های هوش مصنوعی در توسعه فرانت‌اند

1. قوت‌ها:

- تولید سریع اسکلت صفحات و فرم‌ها.
- کمک در اتصال API و مدیریت وضعیت‌های بارگذاری/خطا.
- سرعت بالا در رفع باگ‌های UI کوچک.

2. ضعف‌ها:

- احتمال ناهماهنگی اولیه با منطق دقیق permission.
- نیاز به بازبینی انسانی برای UX واقعی سناریوهای پیچیده.
- سخت بودن اتصال Front و Back برای هوش‌مصنوعی
- ضعف در هندل کردن برنچ‌های متفاوت گیت‌هاب

8. قوت‌ها و ضعف‌های هوش مصنوعی در توسعه بک‌اند

1. قوت‌ها:

- سرعت در ساخت CRUD، ViewSet، Serializer و مسیرهای REST.
- کمک در مدل‌سازی اولیه و نوشتن اعتبارسنجی‌های کسب‌وکاری.
- کمک موثر در تست‌نویسی و پوشش سناریوهای اصلی.

2. ضعف‌ها:

- احتمال خطای اولیه در قوانین ریز RBAC و چندنقشی بودن کاربر.

- نیاز به اصلاح دستی در سناریوهای stateful چندمرحله‌ای.
- نیاز به بازبینی امنیتی و پایداری برای production.

9. نیازسنجی اولیه و نهایی پروژه

1. نیازسنجی اولیه:

- در ابتدا تصمیم بر این گرفته شد تا پیاده‌سازی back و front به طور مستقل از هم باشد ولی به زودی فهمیدیم که نبود API های منطقی هنگام ساخت front و اتصال front و back را مشکل می‌کند.
- تمرکز روی پیاده‌سازی فنی endpoint ها و تست با swagger بدون نیاز به front.

2. نیازسنجی نهایی:

- فهمیدیم که باید مقدار کمی از back را بسازیم و با front مرج کنیم تا بتوان front را مطابق back به پیش برد.

3. نقاط قوت تصمیم‌ها:

- تقسیم بندی وظایف به طول مستقل
- جداسازی واضح لایه‌ها در React.

4. نقاط ضعف تصمیم‌ها:

- پیچیدگی بالا در هنگام مرج.
- نیاز به تست‌های بیشتری برای سناریوهای مرزی.
- وابستگی به تست دستی بیشتر در گردش‌کارهای طولانی.
- حجیم شدن بخش دیباگ کردن.

10. جمع‌بندی

1. پیاده‌سازی شد Django + DRF + JWT + React/Vite سامانه با.
2. الزامات اصلی پروژه در دو فاز بک‌اند و فرانت پوشش داده شد.
3. نیز اضافه شد Template و صفحه بازگشت sandbox بخش اختیاری پرداخت با درگاه.
4. آماده است hardening، افزایش تست E2E، CI/CD پروژه برای توسعه بعدی (بهبود امنیتی).