گزارش پروژه

1. مسئولیت‌ها و کارهای انجام‌شده توسط هر عضو

در این بخش توضیح دهید که هر عضو تیم چه وظایفی بر عهده داشته است و چه بخش‌هایی را توسعه داده است.

- **عضو ۱ – نام: ملیکا علیزاده**
  توسعه فرانت‌اند بخش‌های تشکیل پرونده، ثبت و بررسی شواهد، حل پرونده، شناسایی و بررسی مظنون‌ها و تعیین مجازات؛ تست و رفع viewهای ایراد آنها؛ کامل کردن فیلدهای مدل‌های بک‌اند در فلوهای ذکرشده؛ توسعه بک‌اند بخش‌های پاداش و مجازات؛ توسعه تعدادی از در بخش‌های مربوط به تشکیل و حل پرونده.

- **عضو ۲ – نام: ثمین اکبری**
  ایجاد مدل‌ها و اپ‌های پروژه و مدل‌سازی ابتدایی پروژه؛ توسعه بک‌اند بخش‌های تشکیل پرونده، ثبت و بررسی شواهد، حل پرونده، شناسایی و بررسی مظنون‌ها و پاداش؛ نوشتن تست و رفع ایراد فلوهای بک‌اند.

- **عضو ۳ – نام: معین آعلی**
  توسعه بک‌اند بخش‌های ثبت‌نام، ورود و درگاه پرداخت؛ توسعه فرانت‌اند بخش‌های درگاه پرداخت، تخته کاراگاه، نمایش متهمان تحت تعقیب شدید و ثبت اطلاعات و دریافت پاداش؛ نوشتن تست برای فرانت‌اند.

---

2. قرارداد‌های توسعه (Naming Conventions، قالب پیام‌های Commit و ...)

استفاده PascalCase و برای نام مدل‌ها و کلاس‌ها از snake_case در پروژه بک‌اند، برای نام اپ‌ها، فایل‌ها، فولدرها، فیلدها و نام توابع از شده‌است.
برای توابع و متغیرها از الگوی kebab-case، ها از الگویcomponent type در پروژه فرانت‌اند، برای نام‌گذاری فایل‌ها و استفاده شده‌است PascalCase و برای کامپوننت‌ها از الگوی camelCase.
پیام‌های کامیت بر اساس الگوی Conventional Commits نوشته شدند، مانند:
`feat: Add user login API` یا `fix: Correct navbar alignment`.

---

3. نحوه مدیریت پروژه (چگونگی تولید و تقسیم وظایف)

ابتدا داک پروژه به طور کامل چند مطالعه شد و با توجه به آن بخش‌های اصلی مورد نیاز پروژه، شامل ماژول‌های بک‌اند و فرانت‌اند مشخص شد. سپس با شکستن هر یک از این بخش‌های بزرگ به تسک‌های کوچک‌تر، با توجه به میزان تسلط و حجم تسک‌های دیگر بین اعضای تیم تقسیم شد.
هر تسک پس از انجام شدن با توسعه‌دهنده فرانت‌اند یا بک‌اند متناظر هماهنگ شده و تست‌های لازم انجام می‌شد.

4. موجودیت‌های کلیدی سامانه و دلیل وجود آن‌ها

موجودیت *User* برای مدیریت کاربران
موجودیت *Role* برای مدیریت نقش‌های کاربران
موجودیت *Case* برای ذخیره اطلاعات مربوط به پرونده‌ها و حل آنها
موجودیت *Complaint* برای ثبت و ذخیره شکایات
موجودیت *Crime* برای ذخیره جرم مربوط به هر پرونده
موجودیت *Suspect* برای ثبت و مدیریت مظنون‌ها
موجودیت *Interrogation* برای ثبت و بررسی بازجویی متهمان
موجودیت *Punishment* برای ثبت و ذخیره مجازات‌ها
موجودیت *Evidence* برای ثبت و بررسی شواهد که توسط کاربران عادی یا پلیس وارد می‌شود
موجودیت *Report* برای ثبت گزارشات کاربران در مورد مظنون‌ها و پرونده‌ها
موجودیت *Reward* برای مدیریت پاداش کاربران
موجودیت *Transaction* برای مدیریت درگاه پرداخت

---

5. حداکثر ۶ پکیج NPM استفاده‌شده در پروژه + خلاصه کارکرد و دلیل استفاده

1. **Jest** — برای تست واحد و یکپارچه؛ رانر سریع و سازگار با TypeScript.
2. **Zod** — برای validation و type-safe schema؛ هم در runtime و هم در TypeScript.
3. **Zustand** — برای state management کلاینت (auth، detective board)؛ سبک و بدون boilerplate.
4. **@xyflow/react** — برای ساخت گراف و flowchart تعاملی؛ مثل detective board و نمودار پرونده‌ها.
5. **@tanstack/react-query** — برای مدیریت داده‌های سرور، کش و درخواست‌های async؛ کاهش boilerplate و بهبود UX.
6. **React Hook Form** — برای مدیریت فرم‌ها؛ عملکرد بالا، validation یکپارچه با Zod.

6. چند نمونه کد تولید شده توسط هوش مصنوعی

```
def test_admin_can_assign_role_to_user(self):
    # Authenticate as admin
    self.client.force_authenticate(user=self.admin_user)

    # Prepare update payload
    url = reverse("user-detail")
    payload = {
        "role": self.detective_role.id,
    }

    response = self.client.patch(
        reverse("user-detail"),
```

```
        payload,
        format="json",
    )

    self.normal_user.refresh_from_db()

    self.assertEqual(response.status_code, status.HTTP_200_OK)
    self.assertEqual(self.normal_user.role.id, self.detective_role.id)


class RegisterView(generics.CreateAPIView):
    queryset = User.objects.all()
    serializer_class = RegisterSerializer
    permission_classes = [permissions.AllowAny]

    def post(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        user = serializer.save()

        refresh = RefreshToken.for_user(user)

        return Response(
            {
                "user": UserSerializer(user).data,
                "refresh": str(refresh),
                "access": str(refresh.access_token),
            },
            status=status.HTTP_201_CREATED,
        )


const refreshAccessToken = async (): Promise<string> => {
  const refresh = useAuthStore.getState().session?.refresh;
  if (!refresh) throw new Error("No refresh token");

  const { data } = await axios.post<{ access: string }>(
    `${API_URL}/auth/login/refresh/`,
    { refresh },
  );

  const session = useAuthStore.getState().session;
  if (session) {
    useAuthStore.getState().setSession({ ...session, access: data.access
  }
```

```
  return data.access;
};


const getIcon = (evidence: BaseEvidence) => {
  const e = evidence as unknown as Record<string, unknown>;
  if (e.transcription) return <HugeiconsIcon icon={ArchiveIcon} classNam
  if (e.vehicle_model) return <HugeiconsIcon icon={Car01Icon} className="
  if (e.images) return <HugeiconsIcon icon={InjectionIcon} className="siz
  if (e.owner_first_name) return <HugeiconsIcon icon={FingerPrintIcon} cl
  return <HugeiconsIcon icon={ArchiveIcon} className="size-4" />;
};
```

---

7. ضعف‌ها و قوت‌های هوش مصنوعی در توسعه فرانت‌اند

.نقاط قوت: ساخت سریع کامپوننت‌ها، صفحات و فرم‌ها
composition)، مهارت در الگوهای ری‌اکت (هوک‌ها
مفید برای رابط‌های کاربری تکراری (جدول‌ها، کارت‌ها، مودال‌ها)؛
API از روی اسکیماهای TypeScript کاربردی برای تولید تایپ‌های
مؤثر در پیاده‌سازی مسیریابی و ساختار لایه‌بندی؛
یکپارچه‌سازی کتابخانه‌ها (React Query، Zustand)
روی یک پایهٔ مستحکم خوب عمل می‌کند — زمانی که معماری اولیه مناسب باشد، به‌خوبی توسعه و گسترش می‌دهد؛
:نقاط ضعف
های غیرضروری اضافه کند؛abstraction ممکن است بیش‌ازحد مهندسی کند یا
ها را در فرم‌ها و اعتبارسنجی نادیده بگیردedge case ممکن است
ممکن است با سیستم طراحی یا قراردادهای موجود هماهنگ نباشد؛
ممکن است کدهای طولانی یا تکراری تولید کند؛
بارگذاری تنبل) بهینه عمل نکند؛ ،memoization) ممکن است برای بهینه‌سازی عملکرد
را به‌خوبی رعایت نکند (accessibility) ممکن است دسترس‌پذیری
وقتی پایهٔ اولیه ضعیف باشد عملکرد خوبی ندارد؛ کیفیت خروجی به‌شدت به فونداسیون پروژه وابسته است
.تمایل دارد نسخه‌های قدیمی کتابخانه‌ها را پیشنهاد دهد که منجر به مشکلات سازگاری و باگ می‌شود

---

8. ضعف‌ها و قوت‌های هوش مصنوعی در توسعه بک‌اند

خوب ai نقاط قوت: در معماری و مدل‌بندی ابتدای پروژه کمک کرد تا ساختار کلی پروژه برایمان مشخص شود. در نوشتن تست‌ها استفاده از است چون از دیدی غیر از دید خود برنامه‌نویس حالت‌های مختلف را بررسی می‌کند. تشخیص کارکردهای اصلی ماژول و نوشتن تست برای بسیار سریع عمل می‌کند و بدون آن مدت بیشتری طول می‌کشد ai محسوب می‌شود. در دیباگ کردن هم ai پوشش همه آنها یک نقطه قوت برای تا منبع یک مشکل پیدا شود. در نوشتن کلاس‌های ساده مانند ادمین‌ها، سریالایزرها و ویوهای ساده، سریع و دقیق است و باعث افزایش سرعت

.توسعه می‌شود

نقاط ضعف: هوش مصنوعی نمی‌توانست به طور دقیق فلوها را شناسایی کند و بعضاً فیلدهایی برای مدل‌ها پیشنهاد می‌داد که لازم نبود و تکراری بود. همچنین پروژه را به اپ‌های بسیار کوچکی شکسته بود که این کار صحیح نیست و هر اپ نباید فقط شامل یک مدل باشد. اکثر .کدهایی که توسط هوش مصنوعی نوشته می‌شد نیازمند بررسی و تغییرات قابل توجهی بود که با ساختار فعلی پروژه هماهنگ شود

9. نیازسنجی‌های ابتدایی و نهایی پروژه + قوت‌ها و ضعف‌های تصمیمات

در ابتدا دیدی که نسبت به پروژه و نیازمندی‌های آن داشتیم کوچکتر بود و دیدگاه کاملاً دقیقی نداشتیم؛ هر چه جلوتر رفتیم متوجه شدیم که بعضی نیازها به درستی تشخیص داده نشده‌اند و در بخش‌های توسعه داده شده باید تغییر ایجاد میکردیم. در مجموع تصمیماتی که در ابتدا گرفته بودیم تا حد خوبی درست بود و از همان تصمیمات تا پایان پروژه پیروی شد؛ بخش‌هایی که مشکل داشتند عمدتاً کوچک بودند و در جریان فلوها تشخص .داده شدند