# گزارش پروژه درس وب

آرمان آریان‌پور  کوشا معینی  بهراد مظاهری

# آرمان آریان‌پور

## مسئولیت‌ها و کارهای انجام شده

### فاز بک‌اِند (بخش‌های ۱، ۲ و ۳)

من زیرساخت احراز هویت و چرخه کامل پرونده‌سازی را طراحی کردم.

بخش ۱ — ثبت‌نام و ورود: سیستم ثبت‌نام با فیلدهای یکتای نام کاربری، کد ملی، ایمیل و شماره تماس پیاده‌سازی شد. کاربر می‌تواند با هر یک از این چهار شناسه وارد سیستم شود. نقش‌ها داینامیک بوده و مدیر می‌تواند هر نقش را به کاربران تخصیص دهد. احراز هویت با SimpleJWT و چرخش خودکار Refresh Token ایمن شد.

بخش ۲ — تشکیل پرونده: دو مسیر ایجاد پرونده: از طریق شکایت (با جریان تایید کارآموز → افسر پلیس) و از طریق مشاهده صحنه جرم. قانون ابطال پرونده پس از ۳ اطلاعات ناقص از شاکی پیاده‌سازی شد.

بخش ۳ — ثبت شواهد: چهار نوع مدل شاهد پیاده‌سازی شد: استشهاد شاهدان، شواهد زیستی (با فیلد نتیجه پزشکی)، وسایل نقلیه (پلاک یا شماره سریال به صورت منع‌الجمع) و مدارک شناسایی (ساختار کلید_مقدار).

### فاز فرانت‌اِند (بخش‌های ۱، ۲ و ۳)

- صفحه اصلی: معرفی اداره پلیس با سه آمار زنده (پرونده‌های حل‌شده، کارمندان، پرونده‌های فعال).
- صفحه ورود/ثبت‌نام: فرم چندمرحله‌ای با اعتبارسنجی کامل سمت کلاینت.
- داشبورد ماژولار: ماژول‌ها بر اساس نقش فیلتر می‌شوند (مثلاً کارآگاه تخته کارآگاه می‌بیند اما پزشک قانونی نمی‌بیند).

## قراردادهای توسعه

- نام‌گذاری: PascalCase برای مدل‌های جنگو؛ snake_case برای فیلدهای JSON و API.
- پیام‌های کامیت: الگوی Conventional Commits:

```
feat(auth): add JWT refresh token rotation
feat(case): implement 3-strike cancellation logic
feat(evidence): add vehicle plate/serial mutual exclusion
```

## نحوه مدیریت پروژه

وظایفم پیش‌نیاز بقیه بخش‌ها بودند. مدل‌های User، Case و Evidence را ابتدا به صورت stable تحویل دادم تا سایر اعضا بتوانند کار خود را شروع کنند. هماهنگی از طریق GitHub Issues و جلسات هفتگی انجام می‌شد.

## موجودیت‌های کلیدی سامانه

| موجودیت | دلیل وجود |
|---|---|
| CustomUser | شناسه‌های چندگانه یکتا لازم است که مدل پیش‌فرض جنگو پشتیبانی نمی‌کند |
| Role | نقش‌ها داینامیک هستند؛ مدیر باید بدون تغییر کد نقش اضافه/حذف کند |
| Case | واحد اصلی فعالیت؛ همه چیز به پرونده وابسته است |
| CaseRevision | نگهداری پیام‌های خطای کارآموز برای شفافیت فرآیند |
| EvidenceKeyValue | مدارک شناسایی تعداد فیلد مشخصی ندارند؛ ساختار JSON لازم است |

## نمونه کد تولید شده توسط هوش مصنوعی

بخش عمده‌ای از استایل‌دهی صفحه داشبورد با Tailwind:

```
<div className="min-h-screen bg-gray-950 text-gray-100
               flex flex-col items-start px-8 py-6">
  <header className="w-full flex justify-between items-center
                     border-b border-gray-700 pb-4 mb-6">
    <h1 className="text-2xl font-bold tracking-widest
                  uppercase text-amber-400">
      L.A. Noire - Police Department
    </h1>
    <button className="bg-red-700 hover:bg-red-600 text-white
                       px-4 py-2 rounded text-sm font-medium
                       transition-colors duration-150">
      Logout
    </button>
  </header>

  <main className="grid grid-cols-1 md:grid-cols-3
                   gap-6 w-full">
    <StatCard label="Active Cases" value={stats.active}
      className="bg-gray-800 border border-gray-700
                 rounded-lg p-5 shadow-lg" />
    <StatCard label="Solved Cases" value={stats.solved}
      className="bg-gray-800 border border-gray-700
                 rounded-lg p-5 shadow-lg" />
    <StatCard label="Officers"     value={stats.officers}
      className="bg-gray-800 border border-gray-700
                 rounded-lg p-5 shadow-lg" />
```

```
  </main>
</div>
```

Validator منع‌الجمع پلاک و سریال در بک‌اند:

```
def clean(self):
    if self.plate_number and self.serial_number:
        raise ValidationError(
            "Cannot have both plate and serial number."
        )
```

Interceptor خودکار برای Token: Refresh

```
api.interceptors.response.use(
  res => res,
  async err => {
    const original = err.config;
    if (err.response?.status === 401 && !original._retry) {
      original._retry = true;
      const { data } = await axios.post('/auth/token/refresh/',
        { refresh: localStorage.getItem('refresh') });
      api.defaults.headers.common['Authorization'] =
        'Bearer ' + data.access;
      return api(original);
    }
    return Promise.reject(err);
  }
);
```

## ضعف‌ها و قوت‌های هوش مصنوعی در فرانت‌اِند

قوت: سرعت نوشتن CSS و کلیات صفحات فرانت را به شدت بالا برد. ساختار اولیه صفحات، لایوت گرید و رنگ‌بندی کلی را در چند دقیقه آماده می‌کرد که در برنامه‌نویسی معمولی ساعت‌ها وقت می‌گرفت.

ضعف: ضعف شدید در جزئیات و یکدست نبودن خروجی. مثلاً با دستور «همه دکمه‌ها را قرمز کن» فقط برخی دکمه‌ها تغییر می‌کردند و بقیه دست‌نخورده می‌ماندند. سلیقه بصری ضعیفی داشت و بدون راهنمایی دقیق، ظاهر یکدستی تولید نمی‌کرد.

## ضعف‌ها و قوت‌های هوش مصنوعی در بک‌اِند

قوت: ایده‌های خوبی برای ساختار اولیه پروژه داشت؛ از جمله پیشنهاد نحوه تنظیم URLها و وصل کردن بخش‌های مختلف کد از طریق مسیریابی و API endpointهای منطقی.

ضعف: مشکلات زیادی در برقراری روابط نقش‌ها داشت. تعریف مدل داینامیک Role و تخصیص آن به CustomUser را بارها اشتباه پیاده می‌کرد و کدهای تولیدشده اغلب با معماری Permission موجود در DRF سازگار نبودند.

# کوشا معینی

## مسئولیت‌ها و کارهای انجام شده

### فاز بک‌اِند (بخش‌های ۴، ۵ و ۶)

من هسته تعاملی پروژه را پیاده‌سازی کردم؛ از محیط بصری کارآگاه تا جریان بازجویی و دادگاه.

بخش ۴ — حل پرونده (تخته کارآگاه): دو موجودیت اصلی تعریف شد: BoardItem برای ذخیره مختصات x,y هر مدرک روی تخته و BoardLink برای ذخیره جفت‌های متصل. مکانیزم اعلان با Django Signal پیاده‌سازی شد تا کارآگاه بلافاصله پس از ثبت مدرک جدید نوتیف دریافت کند. API PATCH برای بروزرسانی مختصات پس از هر drag نیز طراحی شد.

بخش ۵ — بازجویی: سیستم امتیازدهی دوگانه: گروهبان و کارآگاه هر کدام مستقل عدد ۱ تا ۱۰ می‌دهند و میانگین به کاپیتان ارسال می‌شود. برای جرایم با درجه بحرانی، تایید نهایی رئیس پلیس هم الزامی است. مدل InterrogationScore برای جدا نگه داشتن نظر هر مقام طراحی شد.

بخش ۶ — محاکمه: مدل Trial یک Snapshot کامل از پرونده در لحظه محاکمه نگه می‌دارد تا اگر مدارک بعداً ویرایش شدند، رای قاضی بر اساس اطلاعات همان لحظه باشد. فیلد verdict و punishment نظر نهایی دادگاه را ثبت می‌کنند.

### فاز فرانت‌اِند (بخش‌های ۴، ۵ و ۶)

تخته کارآگاه: هر مدرک به صورت کارت Draggable روی یک canvas div قرار می‌گیرد. پس از رها کردن کارت، مختصات جدید با API PATCH ذخیره می‌شود. خطوط قرمز با xarrows بین کارت‌های متصل رسم می‌شوند. دکمه خروجی با html-to-image کل تخته را به PNG تبدیل و دانلود می‌کند تا قاضی بتواند آن را به پرونده ضمیمه کند.

صفحه Wanted: Most لیست مظنونین با عکس، نام، مبلغ پاداش و امتیاز رتبه‌بندی نمایش داده می‌شود. مرتب‌سازی بر اساس فرمول امتیاز است و تازه‌واردهای لیست با برچسب ویژه مشخص می‌شوند.

صفحه وضعیت پرونده‌ها: بر اساس نقش کاربر، دکمه‌های تایید، رد یا ارسال پیام خطا به کارآموز نمایش داده می‌شود. کارآموز پیغام خطا دریافت می‌کند، افسر می‌تواند تایید یا رد کند و کارآگاه وضعیت کلی را مشاهده می‌کند.

## قراردادهای توسعه

- نام‌گذاری: PascalCase برای کامپونت‌ها؛ camelCase برای توابع.
- پیام‌های کامیت:

```
feat(board): add drag-and-drop card positioning
feat(board): implement red line connections with xarrows
fix(trial): fix judge verdict submission endpoint
```

## نحوه مدیریت پروژه

هر بخش را روی یک برنچ جداگانه توسعه دادم. تخته کارآگاه را ابتدا با داده‌های mock ساختم و سپس به API متصل کردم. API contract مشترک را هم در یک فایل جداگانه نگه داشتیم.

## موجودیت‌های کلیدی سامانه

| موجودیت | دلیل وجود |
|---|---|
| BoardItem | ذخیره مختصات x,y مدارک روی تخته برای بازسازی بین جلسات |
| BoardLink | ذخیره جفت‌های متصل مدارک برای رسم خطوط در بار بعدی |
| InterrogationScore | نظرات گروهبان و کارآگاه مستقل هستند؛ یک مدل واحد نمی‌توان داشت |
| Trial | Snapshot کامل پرونده در لحظه محاکمه؛ مدارک ممکن است بعداً تغییر کنند |
| Notification | کارآگاه باید بلافاصله از مدارک جدید مطلع شود |

## نمونه کد تولید شده توسط هوش مصنوعی

ذخیره مختصات کارت پس از **drag**:

```
const handleDragStop = (id, data) => {
  setBoardItems(prev =>
    prev.map(item =>
      item.id === id ? { ...item, x: data.x, y: data.y } : item
    )
  );
  api.patch(`/board-items/${id}/`, { x: data.x, y: data.y });
};
```

خروجی تصویری از تخته:

```
const exportBoard = () => {
  htmlToImage.toPng(boardRef.current, { quality: 0.95 })
    .then(dataUrl => {
      const link = document.createElement('a');
      link.download = `case-board-${caseId}.png`;
      link.href = dataUrl;
      link.click();
    });
};
```

## ضعف‌ها و قوت‌های هوش مصنوعی در فرانت‌اِند

قوت‌ها: پیشنهاد پکیج xarrows که خودم نمی‌شناختم؛ تولید سریع منطق Drag-and-Drop.

ضعف‌ها: در مدیریت تداخل z-index خطوط و کارت‌ها اشتباه می‌کرد؛ کد خروجی تصویری را با کتابخانه‌های منسوخ می‌نوشت.

## ضعف‌ها و قوت‌های هوش مصنوعی در بک‌اِند

قوت‌ها: طراحی مدل Trial با تمام relationهای لازم؛ نوشتن سریع Signal برای اعلان به کارآگاه.

ضعف‌ها: Snapshot پرونده برای قاضی را عمق کافی نمی‌داد؛ پیاده‌سازی رتبه‌بندی بازجویی را ساده‌انگارانه می‌نوشت.

# بهراد مظاهری

## مسئولیت‌ها و کارهای انجام شده

### فاز بک‌اِند (بخش‌های ۷، ۸ و ۹)

من بخش‌های انتهایی چرخه تجاری را پیاده‌سازی کردم و همچنین مسئول استقرار کل سیستم بودم.
بخش ۷ — وضعیت مظنونین (Most Wanted): فرمول رتبه‌بندی مظنونین به صورت زیر تعریف شد:

$$Score = \max(L_j) \times \max(D_i)$$

که $L_j$ بیشترین روزهای تعقیب در یک پرونده باز و $D_i$ بالاترین درجه جرم ارتکابی است. فرمول پاداش ریالی: $\max(L_j) \times \max(D_i) \times 20{,}000{,}000$ ریال. محاسبه روزانه با Celery Beat انجام می‌شود و نتیجه در مدل WantedScore کش می‌شود تا درخواست‌های صفحه عمومی بدون بار محاسباتی پاسخ بگیرند.
بخش ۸ — گزارش مردمی و پاداش: جریان گزارش مردمی: شهروند گزارش می‌دهد، افسر بررسی می‌کند، کارآگاه تایید نهایی می‌دهد. پس از تایید، یک UUID منحصربه‌فرد (RewardToken) صادر می‌شود. شهروند می‌تواند با وارد کردن کد ملی و این شناسه، وضعیت و مبلغ پاداشش را استعلام بگیرد.
بخش ۹ — وثیقه (optional): مدل BailPayment طراحی شد اما پیاده‌سازی کامل به دلیل محدودیت زمانی به فاز بعدی موکول شد. API پایه برای ثبت وثیقه موجود است.

### فاز فرانت‌اِند (بخش‌های ۷ و ۸)

پنل گزارش‌گیری کلی: صفحه‌ای مخصوص قاضی، کاپیتان و رئیس پلیس که تمام پرونده‌های سیستم را با فیلترهای داینامیک (بازه تاریخ، وضعیت پرونده، نام کارآگاه) نمایش می‌دهد. جزئیات کامل هر پرونده شامل شواهد، مظنونین، اعضای پلیس دخیل و تاریخچه تغییرات قابل مشاهده است.
فرم داینامیک ثبت مدرک: فرمی که فیلدهایش بر اساس نوع مدرک انتخابی تغییر می‌کند: برای مدرک زیستی فیلد نتیجه آزمایش پزشکی ظاهر می‌شود، برای وسیله نقلیه پلاک یا سریال، و برای مدارک شناسایی یک key-value editor داینامیک. این الگو با useWatch از react-hook-form بدون رندر بیهوده پیاده‌سازی شد.
صفحه استعلام پاداش: فرمی عمومی (بدون نیاز به لاگین) که شهروند کد ملی و شناسه یکتا وارد می‌کند و وضعیت پاداش را مشاهده می‌کند.

## قراردادهای توسعه

- نام‌گذاری سرویس‌های **Docker**: lowercase kebab-case (wp-backend، wp-db)
- پیام‌های کامیت:

```
feat(reward): add unique ID generation for witness
feat(wanted): implement score ranking formula
chore(docker): add nginx reverse proxy config
```

## نحوه مدیریت پروژه

وظایفم در انتهای چرخه تجاری قرار داشت. با seed data سایر اعضا تست می‌کردم. Dockerfile هر سرویس توسط عضو مربوطه نوشته شد و من آن‌ها را در docker-compose.yml ادغام کردم.

## موجودیت‌های کلیدی سامانه

| موجودیت | دلیل وجود |
|---|---|
| WantedScore | Cache امتیاز برای جلوگیری از محاسبات سنگین در هر request |
| CitizenReport | از گزارش مردمی تا تایید پلیس چند مرحله دارد؛ نیاز به موجودیت مستقل |
| RewardToken | شناسه یکتایی که شهروند با آن پاداش می‌گیرد |
| RewardClaim | تاریخچه پرداخت‌ها برای شفافیت مالی |

## نمونه کد تولید شده توسط هوش مصنوعی

فرمول محاسبه پاداش با **ORM** جنگو:

```
def calculate_reward(suspect):
    open_cases = suspect.cases.filter(status='OPEN')
    max_days = open_cases.annotate(
        days=ExpressionWrapper(
            Now() - F('opened_at'),
            output_field=DurationField()
        )
    ).aggregate(max_d=Max('days'))['max_d']
    max_severity = suspect.cases.aggregate(
        max_s=Max('crime_level'))['max_s'] or 1
    days_int = max_days.days if max_days else 0
    return days_int * max_severity * 20_000_000
```

فرم داینامیک ثبت مدرک در **React**:

```
const EvidenceForm = ({ caseId }) => {
  const { register, watch } = useForm();
  const evidenceType = watch('type');
  return (
    <form>
      <select {...register('type')}>
        <option value="biological"   <</option>
        <option value="vehicle"   <    </option>
      </select>
      {evidenceType === 'vehicle' && <VehicleFields />}
      {evidenceType === 'biological' && <BioFields />}
    </form>
  );
};
```

## ضعف‌ها و قوت‌های هوش مصنوعی در فرانت‌اِند

قوت‌ها: تولید سریع جداول گزارش با فیلترهای داینامیک؛ پیشنهاد الگوی Conditional Field Rendering در فرم مدارک.

ضعف‌ها: در محاسبه مبلغ پاداش سمت فرانت اشتباهات float precision داشت؛ پیشنهادهای قدیمی React Router v5 می‌داد.

## ضعف‌ها و قوت‌های هوش مصنوعی در بک‌اِند

قوت‌ها: طراحی Celery beat برای محاسبه روزانه؛ نوشتن کوئری‌های بهینه برای فرمول رتبه‌بندی با ORM جنگو.
ضعف‌ها: پیکربندی nginx در Docker به اشتباه timeout می‌گذاشت؛ برای محدودیت‌های شبکه ایران راه‌حلی نداشت.

# بخش مشترک —— تیم

## پکیج‌های NPM استفاده شده در پروژه

جدول زیر پکیج‌های اصلی مشترک کل پروژه را نشان می‌دهد.

| پکیج | کارکرد | توجیه انتخاب |
|---|---|---|
| react-hook-form | مدیریت فرم‌ها | کمترین رندر بیهوده؛ پشتیبانی از fieldArray برای فرم‌های داینامیک |
| axios | درخواست HTTP | امکان تعریف Interceptor برای Refresh Token خودکار در یک مکان |
| zustand | مدیریت State | سبک‌تر از Redux؛ بدون boilerplate؛ اشتراک State بین کامپوننت‌های دور از هم |
| react-draggable | Drag-and-Drop کارت‌ها | پیاده‌سازی ساده بدون نیاز به Canvas؛ مختصات دقیق در onStop |
| xarrows | رسم خطوط بین DOM | تنها پکیج بالغ برای رسم خطوط بین المان‌های معمولی بدون SVG دستی |
| tailwindcss | استایل‌دهی | توسعه سریع؛ dark mode آماده؛ هماهنگی تم Noir در کل پروژه |

## نیازسنجی ابتدایی و نهایی پروژه

### نیازسنجی ابتدایی

در ابتدای پروژه، سیستم به صورت یک اپلیکیشن ساده تعریف شده بود:

- ثبت‌نام و ورود با ایمیل و رمز عبور
- یک جدول کاربر با نقش‌های ثابت (پلیس، مدیر)
- فرم ساده ثبت پرونده جنایی
- لیست مجرمین تحت تعقیب

### نیازسنجی نهایی

پس از بررسی مستندات سیستم واقعی، نیازمندی‌ها به شکل زیر گسترش یافتند:

- احراز هویت چندشناسه‌ای (نام کاربری، کد ملی، ایمیل، شماره تماس) با JWT
- سیستم نقش داینامیک؛ مدیر بدون تغییر کد نقش جدید اضافه می‌کند
- دو مسیر تشکیل پرونده با جریان تایید کارآموز $\rightarrow$ افسر
- چهار نوع مدل شاهد با اعتبارسنجی‌های تخصصی
- تخته کارآگاه بصری با Drag-and-Drop و خطوط قرمز و خروجی PNG
- بازجویی با امتیازدهی مستقل چند مقام

- فرمول رتبه‌بندی ریاضی مظنونین با محاسبه روزانه
- سیستم گزارش مردمی و پاداش با شناسه یکتا
- استقرار کامل با Docker Compose و nginx

### قوت‌های تصمیمات گرفته شده

- نقش داینامیک: مدیر سیستم بدون دخالت توسعه‌دهنده نقش جدید اضافه یا حذف می‌کند؛ انعطاف بالا برای توسعه آینده.
- **JSONField** برای مدارک شناسایی: تعداد فیلدهای مدارک از پیش مشخص نیست؛ ساختار کلید–مقدار این مشکل را بدون migration جدید حل می‌کند.
- ذخیره مختصات در **DB**: تخته کارآگاه بین جلسات مختلف و کاربران مختلف یکسان باقی می‌ماند.
- **WantedScore Cache**: صفحه عمومی مظنونین بدون محاسبه سنگین در هر درخواست پاسخ می‌دهد.
- **Docker Compose**: محیط توسعه هر سه نفر یکسان شد و مشکل «روی ماشین من کار می‌کند» از بین رفت.

### ضعف‌های تصمیمات گرفته شده

- پیچیدگی جریان تایید پرونده: زنجیر کارآموز $\rightarrow$ افسر $\rightarrow$ کارآگاه بیشتر از برنامه زمان گرفت و edge caseهای آن کامل پوشش داده نشد.
- **Polling** به جای **WebSocket**: اعلان مدارک جدید به کارآگاه با polling پیاده شد که بار شبکه‌ای ایجاد می‌کند.
- **Celery** در استقرار: اضافه شدن Celery و Redis به docker-compose پیچیدگی deployment را بالا برد.
- بخش وثیقه ناتمام: بخش ۹ (پرداخت وثیقه) به دلیل محدودیت زمانی کامل پیاده‌سازی نشد.