گزارش پروژه درس برنامه‌نویسی وب
سامانه WEB-L.A. Noire

آرمان آریان پور
کوشا معینی
بهراد مظاهری

۷ اسفند ۱۴۰۴

۱ مقدمه

در این پروژه، ما به دنبال پیاده‌سازی یک سامانه جامع برای مدیریت فرآیندهای تحقیقاتی در یک اداره پلیس فرضی بودیم. سامانه WEB-L.A. Noire با هدف مکانیزه کردن ثبت پرونده، جمع‌آوری شواهد و مدیریت مظنونین طراحی شده است. از آنجایی که پروژه‌های این چنینی نیازمند دقت بالا در رعایت سلسله مراتب و امنیت داده‌ها هستند، ما از تکنولوژی‌های مدرنی همچون React برای بخش فرانت‌اند و Django برای بخش بک‌اند بهره بردیم. این گزارش، شرحی از فرآیند توسعه و نحوه پیاده‌سازی بخش‌های مختلف سامانه توسط اعضای تیم است.

۲ تقسیم وظایف و مسئولیت‌ها

با توجه به ماهیت پروژه و گستردگی بخش‌های آن، وظایف بر اساس بخش‌های کلیدی سامانه بین اعضای تیم تقسیم شد. اگرچه تمامی اعضا در حل چالش‌های فنی یکدیگر مشارکت داشتند، اما مسئولیت اصلی بخش‌های زیر بر عهده هر نفر بوده است:

۱.۲ آرمان آریان‌پور

آرمان مسئولیت بخش‌های پایه‌ای سیستم و مدیریت شواهد را بر عهده داشت. کارهای انجام شده توسط ایشان شامل موارد زیر است:

- احراز هویت (**Authentication**): پیاده‌سازی سیستم ورود و خروج با استفاده از JWT، مدیریت توکن‌ها در LocalStorage و طراحی کانتکست احراز هویت در فرانت‌اند برای حفظ وضعیت کاربر در تمام صفحات.
- ساخت پرونده (**Case Creation**): طراحی مدل‌های اولیه پرونده، ویوهای مربوط به ایجاد پرونده جدید توسط شاکیان و کارآموزان، و پیاده‌سازی سریالایزرهای مورد نیاز برای تبدیل داده‌ها.
- مدیریت شواهد (**Evidence**): پیاده‌سازی منطق چندریختی برای انواع مدارک (Witness، Biological، Vehicle) و قابلیت آپلود تصاویر و فایل‌های صوتی مربوط به شواهد هر پرونده.

۲.۲ کوشا معینی

کوشا بر روی گردش کار پرونده و فرآیندهای تحقیقاتی تمرکز داشت. فعالیت‌های ایشان عبارتند از:

- حل پرونده (**Case Solving**): پیاده‌سازی فرآیند تغییر وضعیت پرونده بر اساس تاییدیه مقام‌های ارشد (افسر و گروهبان) و مدیریت منطق مختومه کردن پرونده‌ها در دیتابیس.
- شناسایی مظنونین (**Suspect Identification**): طراحی بخش شناسایی و اضافه کردن مظنونین به پرونده، پیاده‌سازی منطق جستجو و فیلتر کردن مظنونین بر اساس ویژگی‌های ثبت شده.
- فرآیند محاکمه (**Verdict/Trial**): پیاده‌سازی بخش صدور حکم توسط قاضی، ثبت جزئیات محکومیت یا تبرئه و نهایی کردن وضعیت حقوقی مظنونین اصلی پرونده.

۳.۲ بهراد مظاهری

بهراد مسئولیت بخش‌های عملیاتی، وضعیت مظنونین و سیستم تشویقی را بر عهده داشت:

- وضعیت مظنونین (**Suspect Status**): مدیریت وضعیت لحظه‌ای مظنونین (تحت تعقیب، بازداشت، آزاد با وثیقه) و هماهنگ‌سازی آن با وضعیت کلی پرونده در هر مرحله.
- سیستم پاداش (**Rewards**): پیاده‌سازی بخش محاسبه و تخصیص پاداش برای پرونده‌های حل شده و نمایش رتبه‌بندی کاربران بر اساس امتیازات کسب شده از موفقیت در تحقیقات.
- زیرساخت و داکر: علاوه بر بخش‌های لاجیک، بهراد وظیفه داکریزه کردن سرویس‌ها، تنظیم فایل‌های docker-compose برای دیتابیس SQLite و سرور Vite، و مدیریت میرورهای لازم برای بیلد موفق پروژه را بر عهده داشت.

۳ قراردادهای توسعه (قوانین خودمانی)

ما برای اینکه کدمون خیلی شلخته نشه یه سری قول و قرار با هم گذاشتیم:

۱. نام‌گذاری: توی پایتون همه چیز رو snake_case نوشتیم (مثل آدمیزاد) ولی توی جاوااسکریپت مجبور شدیم camelCase بنویسیم چون استانداردش اینطوری بود و ESLint گیر میداد.

۲. کامیت مسیج‌ها: سعی کردیم مسیج‌های کامیت رو درست بنویسیم که بفهمیم کی چه گندی زده. مثلا اگه باگ فیکس می‌کردیم می‌نوشتیم :fix و اگه فیچر جدید بود :feat. اما همونجور که پیشبینی میشد از یه جای بد فشار کار این موردو از بین برد و دیگر درست کامیت هارو نزدیم.

۳. برنچ‌ها: قرار بود روی برنچ‌های جدا کار کنیم و بعد مرج کنیم که تا حد خوبی رعایت شد ولی خب گاهی هم مستقیم روی مستر پوش کردیم که خدا ببخشه. و در این مورد بهراد مظاهری نقش پر رنگی رو ایفا کرد.

۴ موجودیت‌های سامانه (چیزهایی که ساختیم)

ما دیتابیس رو جوری طراحی کردیم که شبیه اداره پلیس واقعی باشه و بشه باهاش کار کرد. جدول‌های اصلیمون اینا هستن:

۱.۴ Case

این مهمترین جدول ماست. همه چیز به پرونده وصل میشه. فیلد status داره که نشون میده پرونده الان کجاست (دست سربازه؟ دست قاضیه؟). یه فیلد crime_level هم داره که نشون میده جرم چقدر سنگین بوده (خلاف سنگین یا دزدی معمولی).

۲.۴ Suspect

هر پرونده چند تا متهم داره. ما برای اینا عکس، مشخصات و وضعیت (دستگیر شده یا فراری) گذاشتیم. این بخش خیلی مهمه چون کاربر باهاش زیاد کار داره و باید بتونه متهم‌ها رو مدیریت کنه.

۳.۴ Evidence

این بخش رو خیلی روش کار کردیم. از ارث‌بری استفاده کردیم. یه کلاس پدر Evidence داریم و کلاس‌های فرزند مثل VehicleEvidence (ماشین) یا BiologicalEvidence (خون و DNA که هر کدوم فیلدهای خاص خودشون رو دارن.

۴.۴ Role

چون سیستم سلسله مراتب داره (افسر، کارآگاه، رئیس)، ما یه جدول نقش داریم که دسترسی‌ها رو کنترل می‌کنه. مثلا کارآموز نمی‌تونه پرونده رو ببنده یا گزارش نهایی بنویسه.

۵ پکیج‌های NPM (چیزایی که استفاده کردیم)

ما از پکیج‌های آماده استفاده کردیم که کارمون راحت بشه و نخوایم همه چی رو از صفر بنویسیم. اینا توی پروژمون هستن:

۱.۵ axios

به جای fetch از این استفاده کردیم. خیلی راحت‌تره به نظرمون. خودش جیسون رو پارس می‌کنه و ارورها رو بهتر نشون میده. ما یه interceptor نوشتیم که توکن رو بندازه تنگِ همه ریکوئست‌ها که نخوایم دستی بفرستیم.

۲.۵ react-router-dom

خب معلومه که برای رفتن به صفحه‌های مختلف بدون رفرش شدن از این استفاده کردیم. سایت ما SPA هست و اگه رفرش بشه حس خوبی نمیده. نسخه جدیدش یکم اذیت کرد ولی یادش گرفتیم.

۳.۵ lodash

جاوااسکریپت خودش تابع کم داره برای کار با آرایه. ما از lodash استفاده کردیم که راحت‌تر باشیم. مثلا جاهایی که می‌خواستیم دیبانس کنیم سرچ رو که وقتی کاربر تایپ میکنه هی ریکوئست نره سمت سرور.

۴.۵ html2canvas

این خیلی باحاله. کاربر می‌تونه از تخته مدارکش عکس بگیره. این پکیج کل HTML رو تبدیل می‌کنه به عکس PNG و دانلود می‌کنه. اینجوری کارآگاه میتونه گزارشش رو سیو کنه.

۵.۵ vitest

واسه تست نوشتن از این استفاده کردیم. خیلی سریع‌تر از Jest بود و با Vite خوب مچ میشد. ما تست‌ها رو نوشتیم که مطمئن بشیم لاجیک برنامه درسته (و البته نمره بگیریم).

۶.۵ eslint

این رو نصب کردیم که کدمون قرمز بشه وقتی چرت و پرت می‌نویسیم. کمک کرد که غلط‌های املایی و متغیرهای تعریف نشده رو پیدا کنیم و کدمون تمیزتر بشه.

۶ کدهای پروژه

در این بخش، تکه‌هایی از کدهایی که در طول پروژه با همفکری و گاهی با نگاهی به کدهای آماده یا کمک هوش مصنوعی زدیم را آورده‌ایم. هدف این بود که نشان دهیم چطور قطعات مختلف را کنار هم چیدیم.

۱.۶ استایل‌های بخش منو (Sidebar.css)

این بخش مربوط به ظاهر دکمه ورود به منو است که سعی کردیم با افکت‌های ساده، آن را از حالت خشک خارج کنیم.

```
.hamburger-btn {
  position: fixed;
  top: 20px;
  right: 20px;
  background: var(--bg-card);
  border: 1px solid var(--primary);
  border-radius: 12px;
  transition: all 0.4s cubic-bezier(0.19, 1, 0.22, 1);
  box-shadow: 0 10px 30px rgba(0,0,0,0.5);
}

.hamburger-btn:hover {
  transform: scale(1.05);
  box-shadow: 0 15px 40px rgba(212, 175, 55, 0.2);
  border-color: #fff;
}
```

۲.۶ برقراری ارتباط با بک‌ئند (caseApi.ts)

برای اینکه لیست پرونده‌ها را بگیریم، سرویسی نوشتیم که هم آرایه‌های ساده و هم نتایج صفحه‌بندی شده را متوجه شود.

```
export const caseAPI = {
  listCases: async (): Promise<Case[]> => {
    const response = await api.get('/cases/');

    if (Array.isArray(response.data)) {
      return response.data;
    }
    return response.data?.results || [];
  },

  getCase: async (id: number): Promise<Case> => {
    const response = await api.get(`/cases/${id}/`);
    return response.data;
  }
```

```
};
```

۳.۶ مدیریت وضعیت کاربر (AuthContext)

اینجا وضعیت ورود کاربر و توکن‌ها را چک می‌کنیم تا اگر کسی لاگین نیست، به صفحات حساس دسترسی نداشته باشد.

```
export const AuthProvider = ({ children }: AuthProviderProps) => {
  const [user, setUser] = useState(null);
  const [token, setToken] = useState(localStorage.getItem('access_token'));

  useEffect(() => {
    const initAuth = async () => {
      const savedToken = localStorage.getItem('access_token');
      if (savedToken) {
        try {
          const userData = await authAPI.getMe();
          setUser(userData);
          setToken(savedToken);
        } catch (error) {
          localStorage.removeItem('access_token');
          setToken(null);
        }
      }
      setLoading(false);
    };
    initAuth();
  }, []);

  return <AuthContext.Provider value={{ user, token }}>{children}</
    AuthContext.Provider>;
};
```

۴.۶ مدل داده پرونده‌ها (models.py)

در سمت سرور، مدل اصلی پرونده را با وضعیت‌های مختلف تعریف کردیم.

```
class Case(models.Model):
    class Status(models.TextChoices):
        PENDING_TRAINEE = 'PT', '                '
        ACTIVE = 'AC', '     '
        SOLVED = 'SO', '    '

    title = models.CharField(max_length=255)
```

```
    status = models.CharField(max_length=2, choices=Status.choices, default
    ='PT')
    creator = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.
    PROTECT)
```

۵.۶ لاجیک investigation (views.py)

مثلا این تابع محاسبه پاداش رو به کمک هوش مصنوعی خیلی راحت و با تغییرات جزئی بدست اوردیم

```
def _reward_amount_for_suspect(suspect):
    if not suspect:
        return 0

    nc = (getattr(suspect, 'national_code', '') or '').strip()
    if not nc:
        # fallback
        if not suspect.case_id:
            return 0
        days = _pursuit_days(suspect)
        level = _crime_level_score(suspect.case.crime_level)
        return days * level * 20000000

    qs = Suspect.objects.select_related('case').filter(national_code=nc)

    max_lj = 0
    max_di = 0

    for s in qs:
        if s.case_id:
            di = _crime_level_score(s.case.crime_level)
            if di > max_di:
                max_di = di

        if s.case_id and _is_case_open(s.case):
            lj = _pursuit_days(s)
            if lj > max_lj:
                max_lj = lj

    score = max_lj * max_di
    return score * 20000000
```

۷ تجربه استفاده از هوش مصنوعی

در طول مسیر توسعه، هوش مصنوعی ابزار دست ما بود اما چالش‌های خاص خودش را هم داشت. تجربه ما در این بخش به شرح زیر است:

۱.۷ در بخش فرانت‌ئند (**Frontend**)

نقاط مثبت:

- سرعت در طراحی بصری: هوش مصنوعی در استایل‌دهی کلی و ساخت تم‌های CSS فوق‌العاده سریع است. برای اینکه ظاهر اولیه صفحات را بالا بیاوریم، کمک زیادی کرد و وقتمان برای کارهای تکراری تلف نشد.
- دقت در کدهای سرویس: در لایه services که مستقیماً با API درگیر بودیم، هوش مصنوعی در تولید کدهای مربوط به fetch کردن دیتا عالی عمل کرد. چون این بخش‌ها حساس هستند و یک اشتباه کوچک در تایپ یا آدرس‌دهی باعث از کار افتادن کل صفحه می‌شود، استفاده از هوش مصنوعی باعث شد از بروز اشتباهات ناشی از خستگی یا ضعف در تمرکز جلوگیری شود.

نقاط منفی:

- ناتوانی در تغییرات جزئی: برخلاف استایل‌دهی کلی، وقتی نیاز به یک تغییر خیلی ریز یا حساس در چیدمان داشتیم، هوش مصنوعی اصلاً متوجه منظور ما نمی‌شد. در واقع فهم مشکلات جزئی (مثل تداخل دو استایل در یک صفحه خاص) برایش سخت است و حتماً باید یک انسان کد را خط به خط بازبینی و اصلاح می‌کرد.

۲.۷ در بخش بک‌ئند (**Backend**)

نکات مثبت:

- اسکلت‌بندی سریع: برای ایجاد ساختار کلی مدل‌ها و ویوهای استاندارد جنگو، هوش مصنوعی خیلی خوب عمل می‌کند و باعث شد زمان بیشتری برای فکر کردن روی منطق اصلی پروژه داشته باشیم.

نکات منفی:

- ضعف در مدیریت دسترسی‌ها (**Permissions**): یکی از بدترین بخش‌های عملکرد هوش مصنوعی، قسمت مربوط به Permissionها بود. کدهایی که برای محدود کردن دسترسی کاربران مختلف پیشنهاد می‌داد اصلاً جالب نبود و اگر خودمان آن‌ها را بازنویسی نمی‌کردیم، امنیت کل سامانه زیر سؤال می‌رفت.
- ریزه کاری‌های منطقی: هوش مصنوعی معمولاً درک درستی از ارتباطات پیچیده بین موجودیت‌ها در دیتابیس ندارد و در پیاده‌سازی جزئیات خاص (مثل شرط‌های چند مرحله‌ای)، کدهای ناقصی تحویل می‌داد که نیاز به کنترل و مدیریت مستقیم ما داشت.

۸ نیازسنجی و ارزیابی نهایی

۱.۸ تحلیل نیازمندی‌ها

در ابتدای کار، هدف اصلی ما طراحی سامانه‌ای بود که فرآیند ثبت پرونده‌های جنایی را از حالت سنتی خارج کرده و به صورت دیجیتال مدیریت کند. نیازهای کلیدی شناسایی شده شامل موارد زیر بود:

- پیشگیری از مفقود شدن اطلاعات پرونده‌ها و مدارک.
- برقراری سلسله مراتب دقیق دسترسی (سرباز، کارآگاه، قاضی و غیره).
- شفاف‌سازی مراحل پیگیری پرونده و مشخص بودن مسئول هر مرحله.

۲.۸ ارزیابی محصول نهایی

خوشبختانه توانستیم اکثر قابلیت‌های پیش‌بینی شده را پیاده‌سازی کنیم. هرچند چالش‌های فنی زیادی در این مسیر وجود داشت. نقاط قوت تصمیمات تیم:

- استفاده از کانتینرهای Docker که باعث شد فرآیند بیلد و اجرای پروژه در هر محیطی بدون مشکل انجام شود.
- جداسازی کامل فرانت‌ئند و بک‌ئند، که امکان توسعه همزمان و مدیریت بهتر کدها را فراهم کرد.

نقاط ضعف و چالش‌ها:

- عدم پیاده‌سازی سیستم اعلانات ریل‌تایم (مانند استفاده از WebSockets)؛ در حال حاضر برای مشاهده پیام‌های جدید نیاز به رفرش صفحه است.
- تاخیر در شروع مستندسازی فنی که باعث شد در انتهای پروژه زمان زیادی صرف جمع‌آوری اطلاعات شود.
- وجود برخی محدودیت‌ها در رابط کاربری که در نسخه‌های بعدی قابل بهبود است.