بسمه تعالی

دانشکده مهندسی کامپیوتر

درس برنامه‌نویسی وب

نیم‌سال اول ۱۴۰۵-۱۴۰۴

گزارش پروژه درس برنامه‌نویسی وب

اعضای گروه	
ابوالفضل شیخها	۴۰۱۱۰۶۱۴۷
صادق سرگران	۴۰۱۱۰۶۰۳۹
امیررضا سعیدی	۴۰۱۱۰۶۰۴۱

۸ اسفند ۱۴۰۴

فهرست مطالب

۱ مسئولیت‌ها و وظایف انجام‌شده توسط اعضای تیم

در این بخش، مسئولیت‌ها و خروجی‌های هر عضو تیم به‌صورت خلاصه و شفاف ارائه شده است.

۱.۱ خلاصهٔ تقسیم کار تیم

عضو تیم	تعداد کامیت	حوزهٔ اصلی مسئولیت
ابوالفضل شیخها	۵۸	طراحی و پیاده‌سازی هستهٔ بک‌اند + تست‌های یکپارچه
صادق سرگران	۵۱	زیرساخت اولیهٔ بک‌اند، سرویس‌های پیشرفته، و بخشی از تست‌ها
امیررضا سعیدی	۴۲	معماری و پیاده‌سازی فرانت‌اند و تست/مستندسازی آن

۲.۱ ابوالفضل شیخها

- طراحی و پیاده‌سازی بخش‌های کلیدی بک‌اند شامل accounts، cases، evidence، board، suspects و core.
- پیاده‌سازی RBAC در سطح سرویس‌ها (مدیریت نقش/دسترسی، احراز هویت چندشناسه‌ای، مدیریت دسترسی مبتنی بر نقش).
- راه‌اندازی زیرساخت اجرا و پیکربندی پروژه (Docker Compose، تنظیمات محیطی، اتصال Vite/Backend).
- توسعهٔ گستردهٔ تست‌های یکپارچه برای سناریوهای اصلی پروژه (از احراز هویت و جریان کیس تا اسکوپ‌های RBAC، داشبورد، جست‌وجو و اعلان‌ها).

۳.۱ صادق سرگران

- ایجاد ساختار اولیه پروژهٔ بک‌اند (اسکلت اپ‌ها، مدل‌ها، تنظیمات اولیه ادمین و RBAC پایه).
- توسعهٔ پیش‌نویس API برای اپ‌های accounts، evidence و suspects و تکمیل مسیرهای سرویس، سریالایزر و ویو.
- پیاده‌سازی سرویس‌های پیشرفتهٔ دامنه شامل trial، most-wanted، bounty tips، bail، dashboard aggregations و global search.
- مستندسازی Swagger برای مصرف فرانت‌اند و انجام Hardening روی سرویس‌ها (رفع ایراد مسیرها، ORM و خطاهای مرزی).
- توسعهٔ بخشی از تست‌های یکپارچه (به‌خصوص سناریوهای جریان Crime Scene، Evidence و Detective Board) و چند اصلاح UI/Route در فاز نهایی.

۴.۱ امیررضا سعیدی

- طراحی و پیاده‌سازی معماری فرانت‌اند با React/Vite شامل پوستهٔ برنامه، نقشهٔ مسیرها، نشست احراز هویت، لایهٔ ارتباط با API و مدیریت خطا.

- توسعهٔ صفحات اصلی موردنیاز پروژه شامل صفحهٔ خانه، داشبورد ماژولار، مدیریت پرونده و شکایت، مدیریت شواهد، افراد تحت تعقیب، گزارش‌گیری، برد کارآگاهی و پنل ادمین.
- پیاده‌سازی تجربهٔ کاربری نقش‌محور (RBAC در سطح UI)، بهبودهای واکنش‌گرا، حالت‌های بارگذاری/خالی/خطا و رفع باگ‌های اتصال فرانت به بک‌اند.
- توسعهٔ تست‌های فرانت‌اند و تهیهٔ مستندات فنی تصمیمات، ماتریس نیازمندی‌ها و گزارش ارزیابی نهایی تحویل فرانت‌اند.

۵.۱ جمع‌بندی اجرایی تقسیم مسئولیت

- بک‌اند و منطق دامنه عمدتا توسط ابوالفضل شیخها و صادق سرگران توسعه داده شد (با تمرکز بر سرویس‌ها و تست‌های یکپارچه).
- فرانت‌اند و یکپارچه‌سازی نهایی رابط کاربری عمدتا توسط امیررضا سعیدی انجام شد (با پوشش تست و مستندسازی فرانت).
- هر سه عضو در فاز پایانی در رفع باگ‌ها و هم‌ترازسازی Front/Back برای تحویل نهایی مشارکت داشته‌اند.

۲ قواعد توسعه

در این پروژه، برای یکپارچگی کد و مدیریت بهتر همکاری تیمی، مجموعه‌ای از قواعد توسعه رعایت شده است.

۱.۲ قواعد نام‌گذاری و ساختار کد

- در بک‌اند، اپ‌ها با نام کوتاه و کوچک تعریف شده‌اند (مانند accounts، cases، evidence، board، suspects و core).
- در هر اپ بک‌اند، ساختار لایه‌ای ثابت استفاده شده است: models.py، serializers.py، services.py، views.py و urls.py.
- در فرانت‌اند، صفحه‌ها و کامپوننت‌ها با الگوی PascalCase نام‌گذاری شده‌اند (مثل HomePage.tsx و DetectiveBoardPage.tsx).
- هوک‌های سفارشی با پیشوند use و الگوی camelCase نوشته شده‌اند (مثل useCases.ts و useDashboardStats.ts).
- فایل‌های سبک‌دهی به صورت CSS Module و هم‌نام با کامپوننت نگهداری شده‌اند (مثل DashboardPage.module.css).
- لایه‌های api و types به صورت دامنه‌محور تفکیک شده‌اند (مثل api/cases.ts و types/cases.ts).

۲.۲ قواعد پیام کامیت

- قالب غالب پیام‌ها از الگوی type(scope): summary پیروی می‌کند.
- نوع‌های پرتکرار شامل feat، fix، test، docs، refactor و chore بوده‌اند.

- scope دقیقا محدوده تغییر را مشخص می‌کند؛ مانند frontend، backend، api، ui و detective-board.
- در موارد ادغام و بازگشت تغییرات، کامیت‌های Merge و Revert به صورت شفاف ثبت شده‌اند تا تاریخچه پروژه قابل ردیابی بماند.

۳.۲ قواعد تست و کنترل کیفیت

- تست‌های بک‌اند با pytest و الگوی استاندارد test_*.py پیاده‌سازی شده‌اند.
- تست‌های فرانت‌اند با Vitest و الگوی test.ts.* یا test.tsx.* نوشته شده‌اند.
- در فرانت‌اند، اسکریپت‌های استاندارد lint، test و build در package.json تعریف شده‌اند.
- TypeScript در حالت strict تنظیم شده و قواعدی مانند noUnusedLocals و noUnusedParameters فعال هستند.
- ESLint با پیکربندی recommended برای JavaScript/TypeScript و افزونه‌های react-hooks و react-refresh استفاده شده است.

۴.۲ قواعد مستندسازی

- مستندات فنی به صورت مرحله‌ای و شماره‌گذاری‌شده در پوشه md-files نگهداری شده‌اند.
- برای فرانت‌اند، اسناد تصمیم‌گیری و یادداشت‌های فنی در مسیر frontend/docs به صورت موضوع‌محور ثبت شده‌اند.
- گزارش‌ها و یادداشت‌های اصلاح باگ به شکلی نوشته شده‌اند که مسیر تصمیم تا نتیجه نهایی قابل پیگیری باشد.

۳ رویکرد مدیریت پروژه و نحوه تولید/تقسیم تسک‌ها

مدیریت پروژه به‌صورت سناریومحور و مرحله‌ای انجام شد؛ یعنی به‌جای تعریف تسک‌های کلی، وظایف از روی جریان‌های واقعی سیستم شکسته شدند و هر خروجی به یک سناریوی قابل پیاده‌سازی و قابل تست تبدیل شد.

۱.۳ روش تولید تسک‌ها

- مبنای تولید تسک‌ها، نیازمندی‌های سند پروژه (جریان‌ها، نقش‌ها، صفحات و معیارهای ارزیابی) بود.
- در فاز اول، تسک‌ها بر اساس دامنه‌های بک‌اند تعریف شدند: accounts، cases، evidence، board، suspects و core.
- در فاز دوم، تسک‌ها بر اساس صفحات و ماژول‌های فرانت‌اند تعریف شدند: Home، Dashboard، Cases، Evidence، Detective Board، Most Wanted، Reporting و Admin Panel.
- برای هر جریان پیچیده، تسک‌ها به گام‌های کوچک‌تر با خروجی مشخص شکسته شدند (نمونه: سناریوهای 4.x، 6.x، 7.x، 8.x، 9.x و 10.x در تست‌ها).

۲.۳ روش تقسیم تسک‌ها بین اعضا

- تقسیم اصلی بر اساس تخصص انجام شد: بخش عمده منطق دامنه و سرویس‌های بک‌اند بین ابوالفضل شیخها و صادق سرگران و بخش عمده طراحی و پیاده‌سازی فرانت‌اند توسط امیررضا سعیدی.
- نقاط تماس مشترک (API Contract، مسیرها، مجوزها و مدیریت خطا) به‌صورت بین‌تیمی پیش رفت تا اتصال Front/Back بدون گلوگاه انجام شود.
- در مواردی که وابستگی بین تسک‌ها وجود داشت، ابتدا خروجی پایه تحویل می‌شد (مانند API Draft) و سپس تسک‌های تکمیلی (تست یکپارچه، بهینه‌سازی و Hardening) انجام می‌شد.

۳.۳ چرخه اجرا و پایش پیشرفت

- اجرای پروژه به‌صورت مرحله‌ای پیش رفت: ابتدا اسکلت و زیرساخت، سپس پیاده‌سازی سرویس‌ها/صفحات، سپس تست یکپارچه، و در پایان رفع باگ و پایدارسازی.
- پیشرفت کار با کامیت‌های کوچک و مکرر پایش شد؛ هر کامیت یک تغییر مشخص و قابل ردیابی را پوشش می‌داد.
- برای جلوگیری از انحراف، پس از هر دسته پیاده‌سازی، تست‌های مرتبط اضافه یا به‌روزرسانی شد تا رفتار نهایی با سناریوی تعریف‌شده منطبق بماند.
- مستندات مرحله‌ای در طول توسعه به‌روزرسانی شدند تا تصمیم‌ها و تغییر مسیرها قابل پیگیری باشند.

۴.۳ معیار اتمام تسک‌ها

- هر تسک زمانی کامل در نظر گرفته می‌شد که خروجی کد، تست مرتبط و مستند حداقلی آن آماده باشد.
- در تسک‌های فرانت‌اند، پوشش حالت‌های Loading/Empty/Error و واکنش‌گرایی صفحه جزو معیار اتمام بود.
- در تسک‌های بک‌اند، رعایت دسترسی نقش‌محور، مدیریت خطا و تطابق با سناریوهای تست یکپارچه جزو معیار اتمام بود.
- تسک‌هایی که در یکپارچه‌سازی نهایی ناسازگاری ایجاد می‌کردند، با کامیت‌های fix یا refactor بسته می‌شدند و سپس دوباره در مسیر اصلی ادغام می‌شدند.

۵.۳ جمع‌بندی مدیریتی

- رویکرد تیم مبتنی بر شکستن نیازمندی‌ها به سناریوهای قابل اجرا، تحویل تدریجی، و اعتبارسنجی مداوم با تست بوده است.
- این روش باعث شد ریسک تغییرات انتهای پروژه کمتر شود و هم‌راستایی بین نیازمندی، پیاده‌سازی و خروجی نهایی حفظ شود.

۴ موجودیت‌های کلیدی سیستم و دلیل وجود آن‌ها

مدل داده‌ای سیستم به‌صورت دامنه‌محور طراحی شده و در ۶ اپ اصلی پیاده‌سازی شده است: core، accounts، cases، evidence، suspects و board. در ادامه، موجودیت‌های کلیدی و منطق وجود هرکدام آمده است.

۱.۴ موجودیت‌های پایه و هویتی

- TimeStampedModel: مدل پایه انتزاعی برای ثبت created_at و updated_at در تمام جداول اصلی؛ هدف آن یکدست‌سازی زمان‌بندی رکوردها و حذف تکرار در مدل‌ها است.
- Role: موجودیت نقش پویا با permissions و hierarchy_level. دلیل وجود: پیاده‌سازی RBAC قابل مدیریت در زمان اجرا، بدون نیاز به تغییر کد.
- User: کاربر سفارشی سیستم با شناسه ملی، شماره موبایل، ایمیل یکتا و ارتباط ForeignKey به Role. دلیل وجود: پشتیبانی از ثبت‌نام/ورود چندشناسه‌ای و اعمال دسترسی مبتنی بر نقش.

۲.۴ موجودیت‌های پرونده (Case Domain)

- Case: موجودیت مرکزی سیستم که کل چرخه پرونده را نگه می‌دارد (نوع ایجاد، سطح جرم، وضعیت، نیروهای منصوب‌شده و اطلاعات رخداد). دلیل وجود: تمام جریان‌های سیستم (شواهد، مظنون، بازجویی، دادگاه، برد کارآگاهی) به پرونده متصل هستند.
- CaseComplainant: جدول واسط بین پرونده و شاکی با داده‌های تکمیلی مانند is_primary و وضعیت تایید. دلیل وجود: یک پرونده می‌تواند چند شاکی داشته باشد و هر شاکی وضعیت بررسی مستقل دارد.
- CaseWitness: ثبت شاهدان غیرسیستمی با نام، تلفن و شناسه ملی. دلیل وجود: شاهد لزوما کاربر ثبت‌نام‌شده نیست و باید مستقل از User ذخیره شود.
- CaseStatusLog: تاریخچه تغییر وضعیت پرونده شامل وضعیت قبلی/جدید، تغییردهنده و پیام. دلیل وجود: نگهداری ردپای کامل تصمیم‌ها (تایید، رد، ارجاع و ...).

۳.۴ موجودیت‌های شواهد (Evidence Domain)

- Evidence (پایه): شواهد عمومی متصل به پرونده با نوع شاهد، عنوان، توضیح و ثبت‌کننده. دلیل وجود: یک هسته مشترک برای همه انواع شواهد.
- TestimonyEvidence، BiologicalEvidence، VehicleEvidence، IdentityEvidence: زیرنوع‌های تخصصی شواهد بر پایه multi-table inheritance. دلیل وجود: ترکیب «هسته مشترک» با «فیلدهای تخصصی هر نوع».
- نکته ساختاری مهم: در VehicleEvidence قید دیتابیسی CheckConstraint تعریف شده تا فقط یکی از پلاک یا شماره سریال ثبت شود (قاعده XOR).
- EvidenceFile: فایل‌های متصل به هر مدرک (image/video/audio/document). دلیل وجود: ذخیره و مدیریت رسانه مستقل از فیلدهای متنی مدرک.
- EvidenceCustodyLog: زنجیره تحویل و نگهداری مدرک (تحویل، انتقال، بررسی، امحا و ...). دلیل وجود: ثبت مسیر دست‌به‌دست‌شدن مدرک برای قابلیت پیگیری حقوقی/عملیاتی.

۴.۴ موجودیت‌های مظنون و دادرسی (Suspects Domain)

- Suspect: رکورد مظنون در بستر یک پرونده با وضعیت کامل چرخه (تحت تعقیب تا حکم نهایی). دلیل وجود: یک فرد می‌تواند در چند پرونده رکورد مستقل داشته باشد. همچنین منطق Most Wanted بر مبنای تجمیع رکوردها با شناسه ملی انجام می‌شود.
- Warrant: حکم جلب متصل به مظنون با وضعیت و اولویت. دلیل وجود: کنترل رسمی مرحله جلب در جریان حل پرونده.
- Interrogation: جلسه بازجویی با امتیازدهی 1..10 توسط کارآگاه و گروهبان. دلیل وجود: ثبت ساختاری خروجی بازجویی و ورودی تصمیم‌گیری مرحله بعد.
- Trial: رکورد دادرسی شامل قاضی، حکم و جزئیات مجازات. دلیل وجود: ثبت نتیجه نهایی قضایی در اتصال مستقیم با پرونده و مظنون.
- BountyTip: گزارش مردمی با چرخه بررسی چندمرحله‌ای (افسر، سپس کارآگاه) و کد یکتای دریافت پاداش. دلیل وجود: پیاده‌سازی جریان جایزه برای اطلاعات مردمی.
- SuspectStatusLog: تاریخچه تغییر وضعیت مظنون. دلیل وجود: قابلیت ردگیری تصمیم‌های عملیاتی روی هر مظنون.

۵.۴ موجودیت‌های برد کارآگاهی (Board Domain)

- DetectiveBoard: برد اختصاصی هر پرونده با رابطه OneToOne به Case. دلیل وجود: هر پرونده یک فضای تحلیل بصری یکتا داشته باشد.
- BoardNote: یادداشت تحلیلی کارآگاه روی برد. دلیل وجود: ثبت تحلیل‌های متنی که لزوما «شاهد» رسمی نیستند.
- BoardItem: آیتم قابل جابه‌جایی با مختصات position_x و position_y و ارتباط GenericForeignKey به موجودیت مقصد. دلیل وجود: امکان پین کردن انواع مختلف محتوا (مثل Evidence و BoardNote) در یک ساختار واحد.
- BoardConnection: اتصال بین دو BoardItem با برچسب اختیاری. دلیل وجود: مدل‌سازی خط‌های ارتباطی روی برد برای نمایش رابطه بین آیتم‌ها.

۶.۴ موجودیت اطلاع‌رسانی

- Notification: اعلان کاربر با گیرنده مشخص و ارتباط GenericForeignKey به شی مبدا. دلیل وجود: یک مکانیزم اعلان عمومی که بتواند از هر دامنه‌ای (پرونده، شاهد، مظنون و ...) رویداد دریافت کند.

۷.۴ جمع‌بندی روابط کلیدی

- هسته ارتباطات سیستم Case است و موجودیت‌های Evidence، Suspect، Interrogation، Trial، BountyTip، Bail و DetectiveBoard مستقیم به آن متصل هستند.
- هسته هویتی سیستم User/Role است و نقش‌ها، دسترسی و بازیگران تمام جریان‌ها را کنترل می‌کند.
- برای رویدادها و تاریخچه، مدل‌های CaseStatusLog، SuspectStatusLog، EvidenceCustodyLog و Notification لایه پیگیری و شفافیت عملیاتی را فراهم می‌کنند.

۵ پکیج‌های NPM استفاده‌شده (حداکثر ۶ مورد)

در این پروژه، دقیقا ۶ پکیج اجرایی در فرانت‌اند استفاده شده است. کارکرد و توجیه هر مورد به صورت زیر است:

پکیج	کارکرد	دلیل استفاده
react	کتابخانه اصلی برای ساخت رابط کاربری مبتنی بر کامپوننت.	کل معماری فرانت‌اند پروژه بر پایه کامپوننت‌های React طراحی شده و پیاده‌سازی صفحات الزامی پروژه با این ساختار انجام شده است.
react-dom	اتصال کامپوننت‌های React به DOM مرورگر و مدیریت رندر.	برای نمایش واقعی خروجی رابط کاربری در مرورگر، وجود این لایه الزامی است و مکمل مستقیم react محسوب می‌شود.
react-router-dom	مدیریت مسیرها، ناوبری و تفکیک صفحه‌ها در برنامه تک‌صفحه‌ای.	پروژه دارای صفحه‌ها و جریان‌های متعدد (مانند Dashboard، Cases، Evidence، Reporting و Admin) است و مسیریابی نقش‌محور بدون این پکیج عملی نیست.
@tanstack/react-query	مدیریت داده‌های سمت سرور در فرانت‌اند (دریافت، کش، همگام‌سازی و mutation).	کاهش فراخوانی تکراری API، کنترل بهتر وضعیت‌های بارگذاری/خطا و یکپارچه‌سازی رفتار داده‌ها در ماژول‌های مختلف پروژه.
@xyflow/react	پیاده‌سازی برد تعاملی گره/یال با قابلیت جابه‌جایی آیتم‌ها و اتصال بین آن‌ها.	برای ساخت صفحه Detective Board (جریان تحلیل پرونده با آیتم‌های قابل اتصال) این پکیج هسته فنی اصلی رابط کاربری برد است.
html-to-image	تبدیل خروجی یک المان رابط کاربری به تصویر.	برای خروجی گرفتن از برد کارآگاهی به صورت تصویر و پشتیبانی از نیاز گزارش‌گیری/اشتراک‌گذاری وضعیت برد استفاده شده است.

۶ چند نمونه از کدهای تولیدشده با هوش مصنوعی

در این پروژه، از هوش مصنوعی برای تولید پیش‌نویس کد در بخش‌های مختلف استفاده شد و سپس خروجی‌ها به‌صورت دستی بازبینی، اصلاح و با تست‌های پروژه اعتبارسنجی شدند. چند نمونه واقعی از این کدها:

۱.۶ نمونه ۱: قید دیتابیسی XOR برای مدرک خودرو

فایل: backend/evidence/models.py

قطعه کد:

```
class Meta:
    constraints = [
        models.CheckConstraint(
            condition=(
                models.Q(license_plate="", serial_number__gt="")
                | models.Q(license_plate__gt="", serial_number="")
            ),
```

```
            name="vehicle_plate_xor_serial",
        ),
    ]

def save(self, *args, **kwargs):
    self.evidence_type = EvidenceType.VEHICLE
    super().save(*args, **kwargs)
```

توضیح: در مدل VehicleEvidence این قید تضمین می‌کند که از بین دو فیلد license_plate و serial_number دقیقا یکی مقدار داشته باشد. این کار جلوی ثبت داده ناسازگار را در سطح دیتابیس می‌گیرد و فقط به ولیدیشن فرانت‌اند وابسته نیست.

۲.۶ نمونه ۲: پشتیبانی از آیتم‌های چندنوعی روی برد کارآگاهی

فایل: backend/board/models.py

قطعه کد:

```
content_type = models.ForeignKey(
    ContentType,
    on_delete=models.CASCADE,
    verbose_name="Content Type",
)
object_id = models.PositiveIntegerField(
    verbose_name="Object ID",
)
content_object = GenericForeignKey("content_type", "object_id")

position_x = models.FloatField(
    default=0.0,
    verbose_name="X Coordinate",
)
position_y = models.FloatField(
    default=0.0,
    verbose_name="Y Coordinate",
)
```

توضیح: با این ساختار، BoardItem می‌تواند به انواع مختلف موجودیت (مثل Evidence یا BoardNote) وصل شود. در نتیجه، بدون ساخت جدول جدا برای هر نوع آیتم، برد کارآگاهی توسعه‌پذیر و یکپارچه باقی می‌ماند.

۳.۶ نمونه ۳: مدیریت صحیح **Content-Type** در آپلود فایل

فایل: frontend/src/api/client.ts

قطعه کد:

```
const url = `${BASE_URL}${path}`;
const headers = new Headers(options.headers);

if (accessToken) {
  headers.set("Authorization", `Bearer ${accessToken}`);
}

if (
```

```
  options.body &&
  !(options.body instanceof FormData) &&
  !headers.has("Content-Type")
) {
  headers.set("Content-Type", "application/json");
}

try {
  const res = await fetch(url, { ...options, headers });
  // ...
} catch {
  // ...
}
```

توضیح: این منطق باعث می‌شود برای درخواست‌های FormData هدر Content-Type به‌صورت دستی تنظیم نشود و مرورگر خودش boundary را ست کند. نتیجه این است که آپلود فایل در مسیرهای شواهد بدون خطای multipart عمل می‌کند.

۴.۶ نمونه ۴: نرمال‌سازی خطاهای **API** در کل فرانت‌اند

فایل: frontend/src/api/client.ts

قطعه کد:

```
function normaliseError(body: unknown, status: number): ApiError {
  if (body && typeof body === "object") {
    const obj = body as Record<string, unknown>;

    if (typeof obj.detail === "string") {
      return { message: obj.detail };
    }

    if (typeof obj.message === "string") {
      return { message: obj.message };
    }
  }

  return { message: `Request failed (HTTP ${status})` };
}
```

توضیح: این تابع خطاهای با ساختارهای متفاوت بک‌اند (مثل detail، field errors و message) را به یک فرمت واحد تبدیل می‌کند. به همین دلیل، تمام فرم‌ها و صفحه‌ها پیام خطا را به‌صورت سازگار و قابل‌نمایش دریافت می‌کنند.

۵.۶ نمونه ۵: محاسبه امتیاز **Most Wanted**

فایل: backend/suspects/models.py

قطعه کد:

```
related = Suspect.objects.filter(
    national_id=self.national_id,
).select_related("case")

max_days = 0
```

```
    max_degree = 0
    for record in related:
        if record.case.is_open:
            max_days = max(max_days, record.days_wanted)
        max_degree = max(max_degree, record.case.crime_level)

    return max_days * max_degree
```

توضیح: در ویژگی most_wanted_score، امتیاز هر مظنون با بیشینه روزهای تحت تعقیب و شدت جرم محاسبه می‌شود. این خروجی مبنای رتبه‌بندی مظنونان در صفحه Most Wanted و همچنین محاسبه پاداش قرار می‌گیرد.

۷ نقاط قوت و ضعف هوش مصنوعی در توسعه فرانت‌اند

۱.۷ نقاط قوت

- سرعت بالا در تولید اسکلت اولیه: در شروع فاز فرانت‌اند، تولید سریع پوسته برنامه، مسیرها، الگوی صفحه‌ها و ساختار پوشه‌ها با کمک AI باعث شد تیم زودتر وارد فاز پیاده‌سازی قابلیت‌های اصلی شود.
- شتاب‌دهی در تولید کد تکرارشونده: در بخش‌هایی مانند لایه API Client، Route Guards، Error Handling و فرم‌های مشابه، AI زمان نوشتن کدهای تکراری را کم کرد و سرعت توسعه را بالا برد.
- کمک موثر در نوشتن تست‌های فرانت‌اند: AI در تولید پیش‌نویس تست برای سناریوهای رایج (ورود، خطا، رندر لیست، token storage) مفید بود و باعث شد پوشش تست پروژه سریع‌تر تکمیل شود.
- بهبود کیفیت مستندسازی فنی: تهیه پیش‌نویس یادداشت‌های فنی، ماتریس نیازمندی‌ها و گزارش‌های مرحله‌ای با کمک AI سریع‌تر انجام شد و ثبت تصمیم‌ها در طول پروژه منظم‌تر پیش رفت.

۲.۷ نقاط ضعف

- احتمال تولید کد ظاهرا درست ولی ناسازگار با قرارداد واقعی API: در چند مورد، مسیرها یا شکل داده پیشنهادی AI با بک‌اند یکی نبود و در مرحله یکپارچه‌سازی نیاز به اصلاح دستی endpointها و request shape وجود داشت.
- ضعف در درک کامل رفتارهای پیچیده UI: در بخش‌هایی مثل Detective Board و مدیریت وضعیت گره/یال، خروجی اولیه AI در برخی حالت‌ها به رفتار ناپایدار (مثل حلقه به‌روزرسانی) منجر شد و نیاز به بازنویسی انسانی داشت.
- ایجاد اطمینان کاذب در موضوعات حساس مرورگر: نمونه مشخص آن مدیریت FormData و هدر Content-Type بود؛ اگر خروجی AI بدون بازبینی استفاده می‌شد، آپلود فایل به خطا می‌خورد.
- نیاز بالا به بازبینی نهایی انسانی: خروجی AI در سطح کدنویسی مفید بود، اما برای سازگاری با RBAC، تجربه کاربری، متن‌های نهایی و قیود پروژه، بازبینی و اصلاح دستی همچنان ضروری بود.

۸ نقاط قوت و ضعف هوش مصنوعی در توسعه بک‌اند

۱.۸ نقاط قوت

- سرعت بالا در تولید اسکلت دامنه‌ها: در فاز اولیه بک‌اند، تولید سریع مدل‌ها، serializerها، serviceها و viewها با کمک AI باعث شد دامنه‌های اصلی پروژه (cases/evidence/suspects/board) سریع‌تر عملیاتی شوند.
- کمک موثر در طراحی مدل‌های داده: AI در پیشنهاد ساختار رابطه‌ها (FK/OneToOne/GenericFK)، قیود دیتابیسی (مثل CheckConstraint) و تفکیک مدل پایه/زیرنوع‌ها نقش مفیدی داشت.
- شتاب‌دهی در نوشتن تست‌های یکپارچه: برای سناریوهای متعدد پروژه، AI در تولید پیش‌نویس تست‌ها و کیس‌های مرزی کمک کرد و مسیر پوشش‌دهی کامل‌تر سناریوها را کوتاه‌تر کرد.
- بهبود سرعت refactor و hardening: در مراحل پایانی، برای اصلاح مسیرها، یکدست‌سازی خطاها و کاهش کد تکراری، استفاده از AI سرعت اصلاحات را بالا برد.

۲.۸ نقاط ضعف

- ریسک ناهماهنگی با قواعد دامنه: در بعضی خروجی‌های اولیه، پیشنهادهای AI از نظر منطق کسب‌وکار دقیق نبود و با جریان واقعی پرونده/نقش‌ها کاملا هم‌راستا نبود.
- احتمال خطا در جزئیات حساس امنیت و دسترسی: در بک‌اند، کوچک‌ترین خطا در RBAC، وضعیت‌ها یا کنترل سطح دسترسی بحرانی است؛ بنابراین کد پیشنهادی AI بدون بازبینی انسانی قابل اتکا نبود.
- ضعف در درک اثرات تراکنشی و یکپارچگی داده: در برخی موارد، کد اولیه نیاز به اصلاح دستی داشت تا از نظر تراکنش، ترتیب تغییر وضعیت، و عدم تولید داده ناسازگار مطمئن شود.
- نیاز قطعی به تست و بازبینی عمیق: حتی وقتی کد تولیدشده ظاهرا درست بود، بدون اجرای تست یکپارچه و مرور دقیق، امکان بروز خطاهای رفتاری در سناریوهای واقعی وجود داشت.

۹ تحلیل اولیه و نهایی نیازمندی‌های پروژه و ارزیابی تصمیم‌ها

۱.۹ تحلیل اولیه نیازمندی‌ها

- در تحلیل اولیه، نیازمندی‌ها به دو لایه اصلی شکسته شدند: لایه بک‌اند (مدل دامنه، جریان وضعیت‌ها، RBAC، سرویس‌ها) و لایه فرانت‌اند (۹ صفحه الزامی، جریان‌های نقش‌محور و قیود امتیازدهی CP2).
- تصمیم کلیدی اولیه این بود که ابتدا «اسکلت دامنه و قراردادها» تثبیت شود (تفکیک ۶ اپ اصلی، ساختار service/serializer/view، مدل‌های داده، مسیرها و دسترسی‌ها) و سپس پیاده‌سازی کامل منطق کسب‌وکار انجام شود.
- مزیت این تصمیم: معماری از ابتدا منسجم و توسعه‌پذیر شد.

  عیب این تصمیم: در فاز اولیه، بخشی از خروجی‌ها هنوز در سطح draft/stub بودند و تصویر عملیاتی کامل سیستم دیرتر شکل گرفت.

۲.۹ تحلیل نهایی نیازمندی‌ها

- در وضعیت نهایی، نیازمندی‌های هسته‌ای پروژه در هر دو لایه پیاده‌سازی شد: جریان‌های اصلی پرونده، شواهد، برد کارآگاهی، Most Wanted، Bounty، RBAC و صفحات الزامی فرانت‌اند در خروجی نهایی وجود دارند.
- در فرانت‌اند، ۹ صفحه اصلی مورد انتظار پروژه پیاده‌سازی شد و موارد امتیازمحور کلیدی مانند Responsive، Loading/Error States، تست فرانت‌اند و داکر نیز پوشش داده شد.
- در کنار این موارد، بخشی از مسیرهای فرعی به شکل حداقلی نگه داشته شدند (مانند صفحه‌های Suspects، Interrogations و Trial در سطح placeholder) تا تمرکز تحویل روی نیازمندی‌های اصلی و معیارهای امتیازدهی حفظ شود.
- در بک‌اند، تصمیم تمرکز بر تست‌های یکپارچه باعث شد رفتار سناریوهای اصلی (از احراز هویت و نقش‌ها تا جریان پرونده، شواهد و مظنون) قابل اتکا و قابل راستی‌آزمایی باشد.

۳.۹ تصمیم‌های اصلی و مزایا/معایب آن‌ها

- تصمیم: اولویت‌دادن به نیازمندی‌های امتیازمحور CP2 و جریان‌های اصلی.

  مزایا: تحویل پایدارتر، ریسک کمتر در دمو نهایی، پوشش کامل معیارهای اصلی ارزیابی.

  معایب: برخی صفحه‌های فرعی مانند Profile و Notifications و بخشی از UI سناریوهای Suspects، Interrogations و Trial به صورت حداقلی یا placeholder باقی ماندند.
- تصمیم: پذیرش «کد بک‌اند به‌عنوان مرجع نهایی قرارداد API» هنگام اختلاف مستند و اجرا.

  مزایا: رفع سریع ناسازگاری‌ها و جلوگیری از بن‌بست یکپارچه‌سازی.

  معایب: نیاز به همگام‌سازی مداوم مستندات با رفتار واقعی سرویس‌ها.

۴.۹ نتیجه تحلیل نیازمندی‌ها

- تحلیل اولیه باعث شد مرز دامنه‌ها و مسیر اجرای پروژه روشن شود.
- تحلیل نهایی نشان داد تصمیم‌های اجرایی تیم، با وجود برخی trade-offها، پروژه را به خروجی قابل دفاع و هم‌راستا با نیازمندی‌های اصلی رسانده است.

۱۰ جمع‌بندی نهایی

- در این پروژه، مسیر «تحلیل دقیق نیازمندی‌ها + تصمیم‌گیری مرحله‌ای + اعتبارسنجی با تست» باعث شد خروجی نهایی با نیازمندی‌های اصلی پروژه هم‌راستا باقی بماند.
- هوش مصنوعی در هر دو لایه فرانت‌اند و بک‌اند نقش شتاب‌دهنده داشت، اما کیفیت نهایی فقط با بازبینی انسانی، تست یکپارچه و کنترل دقیق قراردادها به دست آمد.
- جمع‌بندی عملی تیم این بود که موفقیت پروژه حاصل ترکیب سه عامل است: معماری درست، مدیریت تصمیم‌های فنی با trade-off شفاف و اجرای منظم تست و بازبینی.