استاد: دکتر ابریشمی
سهند اسماعیل‌زاده - کیان قاسمی - امیرحسین ملک‌محمدی ۴۰۱۱۱۰۶۷۵ - ۴۰۱۱۰۲۲۶۴ - ۴۰۱۱۰۶۵۷۷

دانشکده مهندسی کامپیوتر
طراحی وب
نیم‌سال اول ۱۴۰۴-۱۴۰۵

گزارش پروژه

مقدمه

این گزارش پیاده‌سازی پروژه la-noire-web-system است. هدف گزارش، ارائه یک تصویر جامع از سامانه نهایی، تشریح کامل امکانات ارائه‌شده در لایه‌های Back-end و Front-end، و ثبت نهایی تطابق پروژه با تمام نیازمندی‌های اعلام‌شده در چک‌پوینت اول و دوم است. قابلیت‌ها و منطق پیاده‌سازی تشریح شده است تا مشخص شود هر قابلیت دقیقاً در چه بخش‌هایی از ساختار پروژه ساخته شده و چگونه با سایر اجزا یکپارچه عمل می‌کند.

معماری سامانه

معماری پروژه به‌صورت Full-Stack و ماژولار طراحی و اجرا شده است. در لایه بک‌اند از Django + Django REST Framework استفاده شده و همه سرویس‌ها به‌صورت RESTful API در دسترس هستند. تنظیمات مرکزی در config/settings.py تعریف شده و در آن، اپ‌های دامنه‌ای، پیکربندی REST Framework، احراز هویت JWT، سیاست‌های CORS و مستندسازی OpenAPI به‌صورت یکپارچه مقداردهی شده‌اند. همچنین در config/urls.py مسیردهی اصلی انجام شده و هر دامنه از طریق include به urlconf اختصاصی خود متصل است تا تفکیک مسئولیت‌ها در سطح کد حفظ شود.

احراز هویت با SimpleJWT پیاده‌سازی شده و سرویس‌های صدور access/refresh token در کنار سرویس‌های پروفایل کاربر و مدیریت نشست عمل می‌کنند. به‌صورت کدنویسی‌شده، چرخه درخواست از URL router وارد کلاس‌های ViewSet/APIView می‌شود، سپس serializer داده ورودی را اعتبارسنجی می‌کند و در نهایت عملیات روی model layer انجام می‌گیرد. همین الگو در همه اپ‌ها ثابت نگه داشته شده و باعث می‌شود توسعه قابلیت جدید بدون تغییر معماری پایه ممکن باشد.

در فرانت‌اند، سامانه با React + Vite + Ant Design + Tailwind پیاده‌سازی شده و تمامی صفحات پروژه به‌صورت ماژولار و نقش‌محور ارائه شده‌اند. App.jsx مسیرهای اصلی را تعریف می‌کند، لایه Auth مسئول نگهداری وضعیت کاربر و کنترل دسترسی به مسیرهاست و ماژول‌های api/*.js به‌عنوان لایه ارتباطی با بک‌اند عمل می‌کنند. در این ساختار، هر صفحه صرفاً مسئول منطق نمایشی است و عملیات دریافت/ارسال داده در سرویس‌های API متمرکز شده است؛ بنابراین نگهداشت، توسعه و اشکال‌زدایی سیستم با هزینه پایین‌تری انجام می‌شود.

مدیریت کاربران، نقش‌ها و دسترسی‌ها

سامانه ثبت‌نام و ورود کاربران به‌صورت کامل پیاده‌سازی شده است. در سطح کد، مدل کاربر توسعه‌یافته در اپ accounts فیلدهای هویتی و ارتباطی را نگهداری می‌کند و محدودیت‌های یکتایی برای شناسه‌هایی مانند username، ایمیل، شماره تماس و کد ملی در مدل و لایه اعتبارسنجی سریالایزر اعمال می‌شود. این کار باعث می‌شود حتی در شرایط هم‌زمانی درخواست‌ها، داده تکراری وارد سامانه نشود. در کنار آن، serializerهای کاربری منطق اعتبارسنجی رمز عبور، یکپارچگی فیلدها و قالب داده ورودی را قبل از ذخیره نهایی کنترل می‌کنند.

ورود کاربران با رمز عبور و یکی از شناسه‌های username، ایمیل، شماره تماس یا کد ملی پیاده‌سازی شده است. این رفتار در لایه authentication backend سفارشی‌سازی شده تا جست‌وجوی کاربر با چند شناسه به‌صورت یکدست انجام گیرد و سپس توکن JWT صادر شود. در بخش نقش و دسترسی، موجودیت‌های Role و ActionPermission به‌صورت دیتامحور تعریف شده‌اند و سیاست دسترسی در permission classهای اپ accounts و core اعمال می‌شود؛ به همین دلیل افزودن نقش جدید یا بازتعریف یک عملیات مجاز، فقط با تغییر داده و بدون اصلاح مسیرهای اصلی کد ممکن است.

در سطح API، مسیرهای مدیریت کاربر، نقش و مجوزها در accounts/urls.py سازمان‌دهی شده و در views/user.py پیاده‌سازی شده‌اند. در فرانت‌اند نیز صفحات احراز هویت و پنل مدیریت نقش‌ها درخواست‌ها را از طریق سرویس‌های api ارسال می‌کنند، پاسخ‌ها را در وضعیت محلی نگهداری می‌کنند و بر اساس نقش دریافتی، منوها و قابلیت‌های قابل مشاهده را تنظیم می‌کنند. این یکپارچگی بین بک‌اند و فرانت‌اند باعث شده کنترل دسترسی نه‌فقط در سطح رابط کاربری بلکه در سطح سرویس نیز تضمین شود.

چرخه کامل تشکیل پرونده

فرایند تشکیل پرونده از مسیر شکایت به‌صورت کامل پیاده‌سازی شده است. در کد، موجودیت‌های Complaint و Case در اپ cases/models تعریف شده‌اند و وضعیت‌های چرخه کاری به‌صورت فیلد status و قواعد انتقال وضعیت مدیریت می‌شوند. سریالایزرهای complaint.py و case.py علاوه بر اعتبارسنجی فیلدها، منطق نگاشت شکایت تأییدشده به پرونده را اجرا می‌کنند. در ویوها نیز عملیات تأیید، بازگشت با پیام اصلاحی، و ثبت تاریخچه تصمیم‌ها به‌صورت endpointهای مستقل پیاده‌سازی شده تا هر گام از فرایند قابل ردیابی باشد.

منطق بازگشت افسر به کارآموز و ابطال خودکار پس از سه بار ثبت داده ناقص، در لایه سرویس/ویو با شمارش دفعات بازگشت و به‌روزرسانی وضعیت شکایت انجام می‌شود. این منطق باعث شده رفتار سیستم کاملاً مطابق سناریوی عملیاتی پروژه باشد و از چرخه‌های بازگشت بی‌نهایت جلوگیری شود. در تست‌های اپ cases/tests نیز سناریوهای اصلی جریان شکایت، تغییر وضعیت‌ها و شرایط رد نهایی پوشش داده شده تا صحت این رفتار در نسخه‌های بعدی نیز حفظ شود.

مسیر تشکیل پرونده از صحنه جرم نیز با ساختار مشابه پیاده‌سازی شده است. اطلاعات رخداد، موقعیت، زمان، شرح اولیه و داده شاهدان در مدل‌های مرتبط ذخیره می‌شوند و سپس با توجه به نقش ایجادکننده، مسیر تأیید سلسله‌مراتبی اجرا می‌شود. در صورتی که ثبت توسط رئیس پلیس انجام شود، ویو مربوطه با دور زدن مراحل میانی و به‌روزرسانی مستقیم وضعیت، پرونده را به مرحله بررسی فعال منتقل می‌کند. این انعطاف در منطق کد باعث شده هر دو مسیر Complaint-driven و Crime-scene-driven در یک هسته دامنه‌ای واحد مدیریت شوند.

ثبت و مدیریت کامل شواهد

ماژول شواهد کل نیازمندی‌های سند را پوشش می‌دهد. در لایه مدل، انواع شواهد به‌صورت موجودیت‌های تفکیک‌شده اما مرتبط با پرونده طراحی شده‌اند تا هم اشتراکات داده‌ای حفظ شود و هم ویژگی‌های اختصاصی هر نوع مدرک قابل ثبت باشد. برای مثال، همه انواع شواهد دارای شناسه پرونده، ثبت‌کننده، زمان ثبت، عنوان و توضیح هستند، اما برای شواهد زیستی یا وسایل نقلیه فیلدهای تخصصی جداگانه تعریف شده است. این طراحی باعث می‌شود کوئری‌های گزارش‌گیری ساده بمانند و در عین حال جزئیات تخصصی هر مدرک از بین نرود.

در سریالایزرهای evidence.py، اعتبارسنجی‌های نوع‌محور انجام می‌شود؛ یعنی بسته به نوع مدرک، قواعد لازم روی ورودی اعمال می‌گردد. برای شواهد زیستی، فیلدهای مرتبط با تأیید پزشکی قانونی و نتیجه پیگیری کنترل می‌شوند؛ برای وسایل نقلیه، الگوهای پلاک و شماره سریال اعتبارسنجی می‌شوند؛ و برای مدارک شناسایی، ساختار key-value به‌صورت پویا پذیرفته و ذخیره می‌شود. این منطق در ویوها به‌گونه‌ای پیاده‌سازی شده که کاربر از یک جریان یکپارچه برای ثبت استفاده کند ولی سیستم در پشت صحنه قواعد تخصصی هر دسته را اعمال نماید.

در فرانت‌اند، فرم‌های ثبت مدرک به‌صورت پویا بر اساس نوع شواهد فیلدهای لازم را نمایش می‌دهند و با ارسال داده استاندارد به API، نتیجه ثبت را در همان صفحه بازتاب می‌دهند. صفحات مشاهده شواهد نیز با واکشی endpointهای جزئیات، امکان مرور، ویرایش و بررسی را فراهم می‌کنند. بدین ترتیب زنجیره کامل ایجاد، بازیابی و تحلیل شواهد در هر دو لایه نرم‌افزار پیاده‌سازی و یکپارچه شده است.

حل پرونده، تخته کارآگاه و اعلان‌ها

قابلیت Detective Board به‌طور کامل پیاده‌سازی شده است. در سطح کد، داده‌های گره‌ها و ارتباط‌ها در ساختارهایی ذخیره می‌شوند که به پرونده و شواهد متصل هستند؛ بنابراین هر تخته، نمایش گرافیِ قابل بازسازی از وضعیت تحلیلی همان پرونده است. در فرانت‌اند، کامپوننت تخته با مدیریت وضعیت محلی، عملیات افزودن گره، جابه‌جایی با Drag-and-Drop، اتصال اقلام و حذف یا اصلاح لینک‌ها را انجام می‌دهد. هر تغییر کاربر به‌صورت رویداد به سرویس API ارسال می‌شود تا حالت نهایی تخته در بک‌اند پایدار بماند.

برای تولید خروجی تصویری تخته، منطق کپچر رابط کاربری در صفحه تخته پیاده‌سازی شده و تصویر نهایی به‌عنوان ضمیمه گزارش پرونده قابل ذخیره و استفاده است. این بخش از کد باعث می‌شود نتیجه تحلیل کارآگاه فقط یک نمای موقت روی صفحه نباشد و بتوان آن را در مستندات رسمی پرونده نیز ثبت کرد. همچنین بازیابی دوباره تخته در مراجعات بعدی، از طریق واکشی داده ذخیره‌شده انجام می‌شود و پیوستگی فرایند تحلیل حفظ می‌گردد.

در بخش اعلان‌ها، با وقوع رویدادهای کلیدی مانند ورود مدرک جدید یا ارجاع جدید، ویوهای بک‌اند رکورد Notification تولید می‌کنند و این رکوردها در داشبورد نقش‌محور نمایش داده می‌شوند. مسیرهای mark-as-read و تغییر وضعیت اعلان در API تعریف شده‌اند تا کاربر بتواند وضعیت پیام‌ها را مدیریت کند. یکپارچگی این لایه با چرخه پرونده، تصمیم‌گیری تیمی را سریع‌تر و خطای انسانی در پیگیری پرونده‌های فعال را کمتر کرده است.

شناسایی مظنونین، بازجویی و تصمیم سلسله‌مراتبی

پس از شناسایی و دستگیری مظنون، فرایند امتیازدهی گناهکاری به‌صورت کامل پیاده‌سازی شده است. در مدل‌های اپ investigation، موجودیت مظنون به پرونده متصل می‌شود و برای ارزیابی گناهکاری، رکوردهای جداگانه‌ای برای امتیاز کارآگاه و گروهبان نگهداری می‌گردد. ویوهای مربوط

به ثبت امتیاز، بازه معتبر ۱۰/۱ را بررسی می‌کنند و از ثبت چندباره غیرمجاز برای یک نقش در یک مرحله جلوگیری می‌شود. سپس داده‌ها در لایه تصمیم‌یار تجمیع شده و برای مشاهده در پنل کاپیتان آماده می‌گردد.

کاپیتان در endpoint تصمیم نهایی، علاوه بر امتیازها به شواهد و سوابق بازجویی دسترسی دارد و خروجی تصمیم را با توضیح ثبت می‌کند. برای پرونده‌های بحرانی، یک مرحله تکمیلی در منطق وضعیت تعریف شده که تصمیم کاپیتان را به رئیس پلیس ارجاع می‌دهد و تا قبل از تأیید نهایی، پرونده وارد وضعیت قطعی نمی‌شود. این مکانیزم در سطح state transition پیاده‌سازی شده و مانع دور زدن سطوح تصمیم‌گیری می‌گردد.

ماژول بازجویی نیز به‌صورت کامل با مدل‌های جلسه بازجویی، زمان شروع و پایان، وضعیت اجرا و خروجی جلسه ساخته شده است. سریالایزرهای بازجویی کنترل می‌کنند که جلسه بدون اتصال به مظنون و پرونده ایجاد نشود و همچنین تغییر وضعیت جلسه به‌ترتیب منطقی انجام گیرد. در فرانت‌اند، صفحه بازجویی با دریافت داده زمان‌بندی و خروجی‌ها، روند جلسات را به‌شکل قابل پیگیری نمایش می‌دهد و امکان ثبت سریع نتیجه هر جلسه را برای مأمور مسئول فراهم می‌کند.

محاکمه و ثبت رأی قضایی

بخش محاکمه به‌صورت کامل عملیاتی است. در لایه داده، موجودیت محاکمه و رأی نهایی به پرونده متصل می‌شوند تا هر رأی، تاریخچه کامل فرایند تحقیق تا دادرسی را همراه خود داشته باشد. در serializerهای این بخش، قواعدی مانند اجباری بودن نوع رأی، تاریخ صدور، و صحت فیلدهای مرتبط با مجازات اعمال می‌شود. این طراحی باعث می‌شود خروجی قضایی از نظر ساختاری استاندارد باشد و برای گزارش‌گیری رسمی به‌صورت مستقیم قابل استفاده باشد.

در لایه ویو، endpoint مشاهده پرونده قضایی، داده‌های تجمیع‌شده پرونده، شواهد، مظنونین، نتایج بازجویی و جمع‌بندی‌های قبلی را برای قاضی آماده می‌کند. پس از ثبت رأی، وضعیت پرونده در هسته دامنه‌ای به حالت نهایی منتقل می‌شود و دسترسی‌های مراحل قبل محدود می‌گردد تا از تغییرات ناسازگار پس از رأی جلوگیری شود. این منطق دقیقاً نشان می‌دهد که کد محاکمه صرفاً ثبت یک فرم نیست، بلکه نقطه بستن چرخه پرونده در کل سیستم است.

در فرانت‌اند، صفحه محاکمه داده‌های قضایی را به‌شکل یک نمای یکپارچه نمایش می‌دهد و کنترل‌های لازم برای ثبت رأی یا ویرایش توضیحات را ارائه می‌کند. پیام‌های موفقیت و خطا بر اساس پاسخ API مدیریت می‌شوند و پس از ثبت نهایی، کاربر به نمای وضعیت پرونده هدایت می‌شود. نتیجه این پیاده‌سازی، یک جریان شفاف و قابل استناد از مرحله بررسی تا صدور رأی است.

وضعیت مظنونین، تعقیب شدید و فرمول‌های رسمی پروژه

ماژول وضعیت مظنونین و صفحه Intensive Pursuit مطابق دقیق فرمول‌های پروژه پیاده‌سازی شده است. در بک‌اند، تاریخ شروع تعقیب و سطح خطر برای هر مظنون ذخیره می‌شود و یک منطق محاسباتی روی داده‌ها اجرا می‌گردد تا اگر مدت تعقیب از یک ماه عبور کند، وضعیت intensive pursuit به‌صورت خودکار فعال شود. این منطق در سطح query/business logic پیاده‌سازی شده و خروجی آن از طریق endpoint عمومی در اختیار فرانت‌اند قرار می‌گیرد.

فرمول رتبه‌بندی و فرمول پاداش دقیقاً طبق مستند پروژه در کد پیاده‌سازی و هم‌زمان در رابط کاربری نمایش داده شده‌اند:

$$\text{ranking} = \max(L_j) \times \max(D_i)$$

$$\text{reward} = \max(L_j) \times \max(D_i) \times ۲۰{,}۰۰۰{,}۰۰۰$$

در پیاده‌سازی بک‌اند، مقادیر L_j و D_i از داده پرونده/مظنون استخراج و سپس در سرویس محاسبه ترکیب می‌شوند. مقدار نهایی در پاسخ API برگردانده می‌شود تا معیار رسمی در همه کلاینت‌ها یکسان باشد. در فرانت‌اند نیز همین مقدار رسمی رندر می‌شود و در صفحه پیگیری شدید، کاربران می‌توانند رتبه و پاداش متناظر هر مظنون را به‌صورت آنی مشاهده کنند. به این ترتیب هم انطباق با فرمول پروژه تضمین شده و هم از اختلاف محاسباتی بین لایه‌ها جلوگیری شده است.

سامانه پاداش، کد یکتا و پرداخت

سامانه پاداش به‌صورت کامل پیاده‌سازی شده است. در سطح مدل، درخواست پاداش، وضعیت بررسی، کد یکتا، مبلغ و وضعیت پرداخت در موجودیت‌های اپ rewards نگهداری می‌شوند. در سطح ویو، یک جریان چندمرحله‌ای تعریف شده است: ثبت اولیه توسط کاربر عادی، بررسی افسر پلیس، تأیید نهایی کارآگاه و صدور کد یکتا. تولید کد یکتا در لایه بک‌اند انجام می‌شود تا یکتایی و امنیت فرآیند در سطح سرور تضمین شود و امکان دست‌کاری سمت کلاینت وجود نداشته باشد.

سرویس استعلام پاداش با دریافت کد ملی و کد یکتا پیاده‌سازی شده و پس از اعتبارسنجی تطابق اطلاعات، مبلغ و وضعیت پرونده پاداش را بازمی‌گرداند. همچنین مسیر ثبت دریافت پاداش، تغییر وضعیت از issued به received را با ثبت زمان و کاربر مسئول انجام می‌دهد. در فرانت‌اند، فرم‌های ثبت و استعلام به سرویس‌های api/reward متصل‌اند و نتیجه فرایند را با پیام‌های مرحله‌ای برای کاربر نمایش می‌دهند.

فرایند وثیقه و جریمه برای جرایم سطوح ۲ و ۳ نیز در همین دامنه به‌صورت کدنویسی‌شده مدیریت می‌شود. مبلغ پایه توسط گروهبان تعیین می‌شود، شرط تأیید اضافه برای سطح ۳ در منطق وضعیت لحاظ می‌گردد و پس از پرداخت، پاسخ درگاه در callback endpoint پردازش می‌شود. صفحه بازگشت از درگاه پرداخت در فرانت‌اند نتیجه تراکنش را از بک‌اند واکشی می‌کند و وضعیت نهایی را شفاف به کاربر اعلام می‌نماید؛ بنابراین حلقه کامل «تعریف بدهی، پرداخت، ثبت قطعی» در سامانه بسته می‌شود.

صفحات موردنیاز فرانت‌اند و تجربه کاربری

تمام صفحات الزام‌شده در مستند پروژه پیاده‌سازی شده‌اند. در سطح کد فرانت‌اند، مسیرها در App.jsx تعریف شده و هر مسیر به صفحه دامنه‌ای متناظر در شاخه src/pages متصل است. داشبورد ماژولار نقش‌محور با تکیه بر داده کاربر احرازشده رندر می‌شود و منوها/اکشن‌ها به‌صورت شرطی نمایش داده می‌شوند. این کنترل نمایشی با کنترل دسترسی بک‌اند هم‌راستا است تا امنیت فقط به UI hiding محدود نشود.

صفحه اصلی با واکشی آمار کلیدی از API و نمایش کارت‌های خلاصه پرونده‌ها پیاده‌سازی شده است. صفحه ورود و ثبت‌نام نیز با مدیریت فرم، اعتبارسنجی سمت کلاینت و نگهداری امن توکن در چرخه نشست کار می‌کند. صفحات تخصصی مانند تخته کارآگاه، پیگیری شدید، وضعیت پرونده‌ها و شکایات، گزارش‌گیری، و ثبت/بررسی مدارک هرکدام دارای ماژول داده اختصاصی هستند و در عین حال از الگوی مشترک درخواست، مدیریت خطا و بارگذاری پیروی می‌کنند.

پنل ادمین اختصاصی مستقل از Django Admin نیز پیاده‌سازی شده و مدیریت کاربران، نقش‌ها، پرونده‌ها، شکایات و پاداش‌ها را در رابط یکپارچه فراهم می‌کند. در همه صفحات، حالت‌های loading، empty state و skeleton برای بهبود تجربه کاربری پیاده‌سازی شده‌اند و با استفاده از Tailwind و Ant Design، رابط کاربری در اندازه‌های مختلف نمایش responsive و پایدار باقی می‌ماند. نتیجه این لایه، یک تجربه کاربری منسجم و عملیاتی برای تمام نقش‌های سامانه است.

مستندسازی، تست و استقرار

مستندات Swagger به‌صورت کامل و قابل اتکا آماده شده‌اند و همه endpointهای اصلی پروژه همراه با نمونه درخواست/پاسخ و توضیحات لازم در دسترس هستند. این مستندات با drf-spectacular تولید می‌شوند و با توجه به serializerها، نوع فیلدها، پارامترها و کدهای پاسخ به‌صورت خودکار همگام می‌مانند. بنابراین هر تغییر در لایه API بلافاصله در سند فنی منعکس می‌شود و تیم توسعه و ارائه از یک مرجع واحد استفاده می‌کنند.

در بخش آزمون، سناریوهای اصلی در اپ‌های مختلف بک‌اند پوشش داده شده‌اند و جریان‌های حیاتی مانند احراز هویت، دسترسی نقش‌محور، تشکیل پرونده، ثبت شواهد و چرخه پاداش اعتبارسنجی می‌شوند. ساختار تست‌ها به‌گونه‌ای است که هم رفتار endpoint و هم منطق تغییر وضعیت‌ها کنترل شود. در فرانت‌اند نیز تست جریان‌های کلیدی تعامل کاربر با API پیاده‌سازی شده تا سازگاری بین پاسخ سرویس و رفتار رابط کاربری در نسخه‌های مختلف حفظ گردد.

در استقرار، سرویس‌های Back-end و Front-end به‌همراه وابستگی پایگاه‌داده PostgreSQL در قالب docker-compose یکپارچه شده‌اند. این ساختار اجرای محیط توسعه و ارائه را استاندارد می‌کند و با حداقل پیش‌نیاز، نسخه قابل اجرا را در اختیار تیم قرار می‌دهد. تنظیمات لازم برای اتصال سرویس‌ها، پورت‌ها و متغیرهای محیطی در فایل‌های پیکربندی لحاظ شده و چرخه build/run سامانه به‌صورت پایدار قابل تکرار است.

راهنمای تست فرانت‌اند و بک‌اند

تست فرانت‌اند

برای تست فرانت‌اند از ترکیب Vitest، React Testing Library و Playwright استفاده شده است. این ابزارها امکان تست واحد، تست یکپارچگی و تست سرتاسر را فراهم می‌کنند.

ابزارهای تست فرانت‌اند:

- Vitest: موتور اجرای تست سریع و سازگار با Vite
- React Testing Library: تست کامپوننت‌های React با تمرکز بر رفتار کاربر
- @testing-library/user-event: شبیه‌سازی تعاملات کاربر
- Playwright: تست سرتاسر در مرورگر واقعی
- jsdom: محیط DOM برای اجرای تست‌ها

دستورات اجرای تست فرانت‌اند: پیش از اجرای تست‌ها، وابستگی‌ها را نصب کنید:

```
cd frontend
npm install
```

برای اجرای تست‌های واحد و یکپارچگی:

```
npm test

npm test -- --watch

npm run test:ui

npm run test:coverage
```

برای اجرای تست‌های سرتاسر با Playwright:

```
npx playwright install

npm run test:e2e

npm run test:e2e:ui

npx playwright test --headed

npx playwright test e2e/app.spec.js
```

تست بک‌اند

برای تست بک‌اند از فریمورک تست Django و Django REST Framework استفاده شده است. تست‌ها با pytest نیز سازگار هستند.
دستورات اجرای تست بک‌اند: ابتدا محیط مجازی را فعال کنید و به دایرکتوری بک‌اند بروید:

```
venv\Scripts\activate
cd backend
```

اجرای تست‌ها به همراه مثال:

```
python manage.py test

python manage.py test accounts

python manage.py test accounts.tests.test_auth

python manage.py test --verbosity=2

python manage.py test --keepdb

python manage.py test --parallel
```

اجرای تست با pytest:

```
pip install pytest pytest-django

pytest

pytest --cov=. --cov-report=html

pytest accounts/tests/test_auth.py::TestUserLogin
```

پوشش تست‌های بک‌اند: تست‌های بک‌اند موارد زیر را پوشش می‌دهند:

- احراز هویت و مجوزدهی (JWT token، نقش‌های کاربر، دسترسی‌ها)
- جریان تشکیل پرونده (شکایت، صحنه جرم، تأیید چندمرحله‌ای)
- ثبت و مدیریت شواهد (انواع مختلف شواهد، اعتبارسنجی)
- تخته کارآگاه (ذخیره و بازیابی گراف ارتباطات)
- شناسایی مظنونین و بازجویی (امتیازدهی، تصمیم‌گیری سلسله‌مراتبی)
- محاکمه و رأی قضایی (ثبت رأی، تغییر وضعیت پرونده)
- سامانه پاداش (ثبت، بررسی، صدور کد یکتا، استعلام)
- محاسبات رتبه‌بندی و پاداش (صحت فرمول‌های پروژه)
- API endpoints (درخواست‌ها، پاسخ‌ها، کدهای وضعیت)

نتیجه‌گیری نهایی

پروژه L.A. Noire Web System به‌صورت کامل پیاده‌سازی شده و تمامی نیازمندی‌های اعلام‌شده در مستند درس در دو چک‌پوینت اول و دوم را پوشش می‌دهد. در این گزارش، قابلیت و منطق پیاده‌سازی ارائه شد.