Drive-Ease – Expo Q&A

Concise questions and answers for expo presentations.

Generated: 17/11/2025, 12:36:04 am

Overview & Roles

Q: What is Drive-Ease?
A: A MERN application to digitize the driving license journey: registration, color vision test, learner test, supervisor-led road test (verification + evaluation), and license PDFs.

Q: Who are the main users?
A: Applicants and Supervisors (RTO Officers). Applicants complete tests; supervisors verify identity (photos + OTP) and evaluate road tests.

Technologies & Why

Q: Which technologies are used and for what?
A: React 18 (UI), react-router-dom 6 (routing), axios (HTTP). Express (API server), Mongoose 7 (MongoDB ODM), PDFKit (PDF generation), Multer (uploads), bcrypt/bcryptjs (security), validator (input), dotenv & cors (environment & CORS). MongoDB stores applications, tests, and statuses.

Q: Why MERN?
A: Fast SPA front-end with React; lightweight scalable REST API with Express; MongoDB fits evolving data (statuses, scores) and document-style updates.

Architecture & Integration

Q: How is the project structured?
A: Routes in backend/routes/*.js, models in backend/models/*.js, services in backend/services/*.js, React components in frontend/src/components/*.js. server.js mounts all routes.

Q: How are static images for signs served?
A: Express static route: GET /api/test-images from backend/data/'sign boards'.

Q: How do you backfill identity data?
A: A DigiLocker service stub fetches details to fill missing fields before generating licenses, ensuring PDFs carry proper identity.

Key Flows

- Registration & Payment
- Color Vision Test
- Learner Test (30 Q: 20 signs + 10 theory)
- Road Test: Verification (3 photos + OTP) and Evaluation (10 ratings)
- PDF Downloads: Learner License and Driving License

APIs & Logic

Q: How are learner test questions generated?
A: GET /api/learner-test/questions/:applicationNumber loads JSON, shuffles by category, returns 30 questions without answers.

Q: How is learner test scoring handled?
A: POST /api/learner-test/submit counts correct answers from attempted only; pass at >=21/30. Summary saved in Test collection.

Q: How does road test verification work?
A: Supervisor triggers OTP; CandidateVerification.js captures front/left/right photos. /verify-otp checks otpExpiry, sets roadTestStatus=verified, and stores verificationPhoto.

Q: How is road test evaluation handled?
A: Supervisor rates 10 parameters; total score out of 50; pass threshold 60%. /evaluate-candidate

stores questions, score, status, timestamps.

PDF Generation

Q: How is the Learner License PDF created?

A: GET /api/learner-test/download-license/:appNo uses PDFKit to render LL number, dates, identity info, and applicant photo when available.

Q: How is the Driving License PDF created?

A: GET /api/learner-test/download-driving-license/:appNo prioritizes verificationPhoto, then falls back to application photoData/photoPath. Includes consolidated test results and 20-year validity.

Frontend UX & Progress

Q: How do you compute progress?

A: Milestones: payment(20), color(35), learner pass(50), road scheduled(70), road failed(75), road passed(100). This is displayed in ApplicationDashboard.

Q: How do you ensure the camera does not stay on?

A: Programmatic capture starts the stream, waits ~1s, captures to canvas, and immediately stops all tracks per click.

Security & Edge Cases

Q: How do you secure OTP flow?

A: OTP has expiry; on verification we clear OTP and expiry, stamp verifiedAt, and only then allow evaluation.

Q: What if photos are missing during PDF generation?

A: Code renders a placeholder rectangle with "No Photo" to avoid errors; logs detail any read/parse issues.

Demo Script (2-3 minutes)

- Applicant: Login -> Color Vision -> Learner Test -> Download LL
- Supervisor: Login (supervisor/123456) -> Verify (photos + OTP) -> Evaluate (10 ratings)
- Applicant: Dashboard shows result -> Download Driving License

Future Enhancements

- JWT auth & RBAC for stricter access
- Scheduling + notifications and re-test workflows
- Analytics dashboards; Docker + CI/CD

Basic Tech Q&A

Q: What is the MERN stack?

A: MERN = MongoDB (database), Express (backend framework), React (frontend UI), Node.js (runtime). It enables full-stack JavaScript with JSON all the way.

Q: What is a REST API?

A: A stateless interface over HTTP using resources (nouns) and standard methods (GET, POST, PUT, PATCH, DELETE) with meaningful status codes.

Q: What is middleware in Express?

A: Functions that run between request and response (e.g., JSON parsing, CORS, authentication). Configured with app.use(...) or per-route.

Q: What is CORS and why is it needed?

A: Cross-Origin Resource Sharing controls which origins can call the API from browsers. We enable it via the cors package for the frontend dev server.

Q: What is Mongoose?

A: An ODM for MongoDB that provides schemas, models, validation, and convenient query APIs on top of the native driver.

Q: Schema vs Model in MongoDB/Mongoose?

A: Schema defines structure/validation of documents; Model is the compiled class that you use to create/query/update documents in a collection.

Q: PUT vs PATCH?

A: PUT replaces a resource entirely (idempotent); PATCH partially updates fields. We generally use PATCH/POST for targeted updates.

Q: What are promises and async/await?

A: Promises represent future results; async/await offers readable syntax to write asynchronous code without nested callbacks.

Q: What are React useState and useEffect?

A: useState manages component state; useEffect performs side-effects (e.g., fetching data on mount or reacting to state changes).

Q: What is React Router?

A: A client-side routing library that maps URL paths to components without full page reloads, improving SPA performance and UX.

Q: What is PDFKit?

A: A Node library to programmatically create PDFs: text, images, lines, and streaming output to clients efficiently.

Q: Why Base64 for images?

A: Base64 lets us send/store small images inline (e.g., verification photos). We also support filesystem paths for larger files.

Q: How do you handle passwords?

A: bcrypt/bcryptjs libraries are included for hashing. In this demo, supervisor uses fixed credentials for expo simplicity; hashing is recommended for production.

Q: What HTTP status codes do you use?

A: 200/201 for success, 400 for bad input, 403 for forbidden (e.g., not passed), 404 for not found, 500 for server errors.