Technical Test

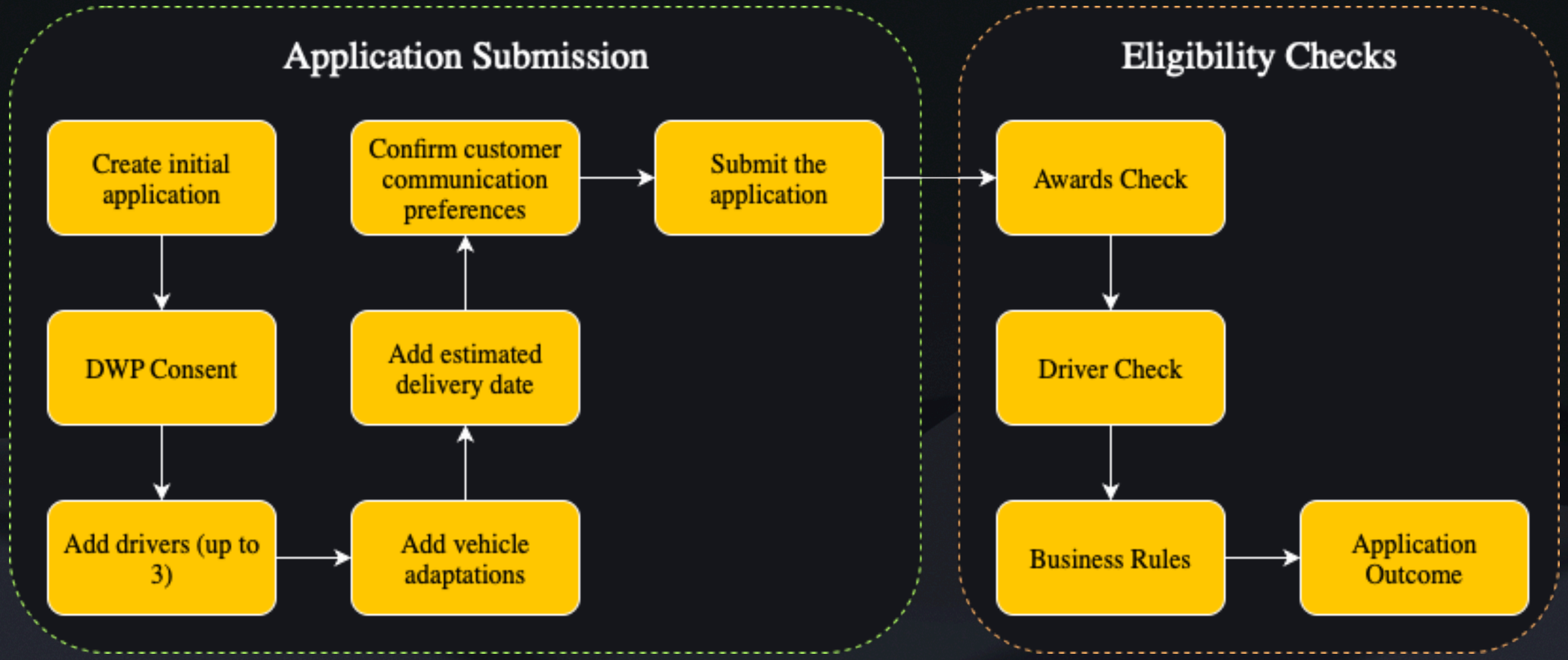
Motability Application Flow

James Wilcox

Task Overview

- Application that manages the process for users applying for the Motability Scheme
- The full application process may take a number of weeks
- The end customer is the scheme applicant who interacts with the process via a dealer acting on behalf of Motability
- The dealer is responsible for collecting all necessary information from the end customer and submitting the application
- Once the application is submitted, the system will carry out all necessary eligibility checks (including 3rd party systems e.g. DVLA)

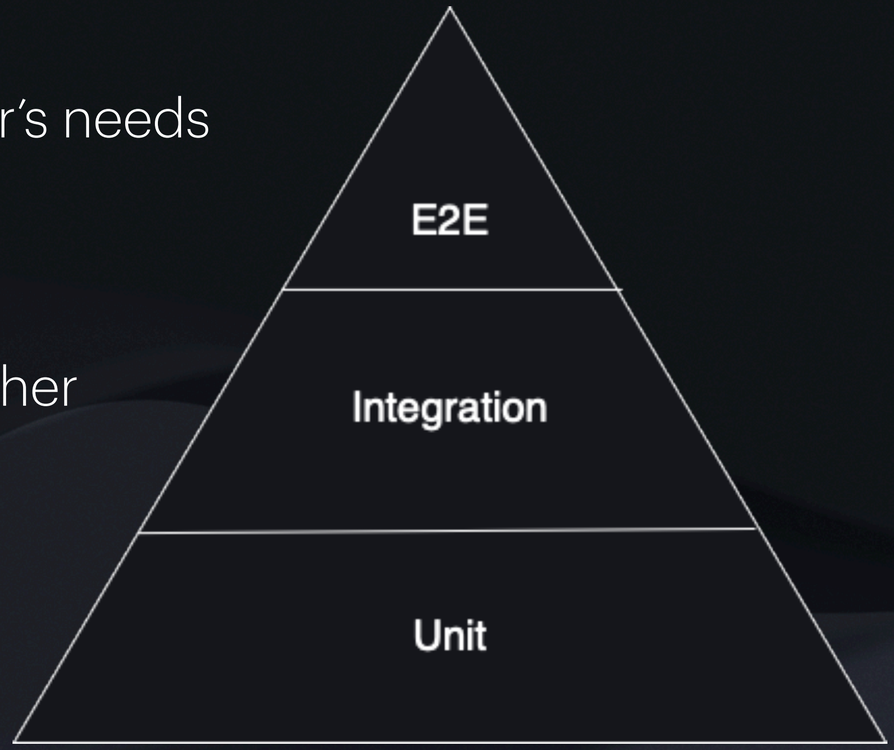
Application Flow



Test Approach

Test Pyramid

- E2E (End-To-End) Testing
  - Ensure the system operates correctly and meets the user’s needs
- Integration Testing
  - Ensure distinct system components work correctly together
- Unit Testing
  - Ensure each system component operates correctly in isolation



Application Submission

Application Submission

Functional Testing

- API Testing
  - Test possible response codes e.g. 200 OK, 400 Bad Request, 401 Unauthorised
  - Test validation of payload e.g. incorrectly formatted data, missing fields
- UI Testing
  - Component testing e.g. text input, date picker, button
  - Test validation of form fields e.g. client-side validation, phone number, national insurance number
- Integration Testing (DVLA system)
  - During development, use a mock. Then conduct system integration testing in pre-production/staging

Application Submission

Non-Functional Testing

- Performance Testing
  - Ensure the performance testing environment is representative of production
  - Monitoring:
    - System-under-test resources (CPU, memory, network, disk)
    - API response times & response codes
    - Load generator(s) resources (CPU, memory, network)
  - System observability (e.g. metrics, logs, traces) are key to generating actionable insights
- Also consider other NFT types e.g. accessibility testing, usability testing, and security testing

Application Submission

Performance Testing

- Load testing = Run the system at the anticipated production load
  - Useful to prove the system can support the expected production load
- Stress testing = Slowly ramp up the system load to above the anticipated production load
  - Useful to understand how the system load impacts the performance of the system
- Spike testing = Run the system at the anticipated production load with short duration load spikes
  - Useful to prove the system can support brief load spikes and understand how quickly it can recover.
- Soak testing = Run the system at the anticipated production load for an extended duration
  - Useful to detect any performance degradation over time e.g. memory leaks, database performance

Eligibility Checks

Eligibility Checks

Key Test Types

- Equivalence Partitioning (EP)
  - Separate the input data into partitions that equate to boolean states (true/false). Only one value in each equivalence partition needs to be tested, reducing the test scope e.g. 10 < Foo < 50
- Boundary Value Analysis (BVA)
  - Design the test data to specifically target the boundaries between success and failure states, identified by equivalence partitioning. This is useful for detecting “out-by-one” errors. Can use either 2-value of 3-value BVA
- Decision Table Testing
  - Useful for representing how different combinations of input conditions will impact the result

	<=10	11-49	>=50
Value	8	32	61
Foo	FALSE	TRUE	FALSE

Boundary	2-value	3-value
10	10,11	9,10,11
50	49,50	49,50,51

Foo	Bar	Result
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Eligibility Checks

Business Rules

Key	Title	Description	Recommended Test Type
BR1	DRIVER_WITH_AUTOMATIC_LICENCE_CANNOT_DRIVE_MANUAL_VEHICLE	The rule will fail if the vehicle has manual transmission and the driver has an automatic licence.	Decision table testing: Transmission type (Manual/Automatic) License Type (Manual/Automatic)
BR2	AT_LEAST_ONE_DRIVER_WITH_FULL_LICENCE	The rule will pass if there is at least one driver with a full entitlement (i.e. not provisional), if not it will fail.	EP & BVA: Count(full_license) >= 1 Test cases: [0, 1, 2] with full license
BR3	DRIVER_IS_DISQUALIFIED	The rule will fail if the driver is disqualified at the moment.	Boolean test: !isDisqualified(driver) Test both conditions

Eligibility Checks

Business Rules

Key	Title	Description	Recommended Test Type
BR4	DAG_V2_DRIVER_LICENCE_CATEGORY_GHI_ENDORSEMENT	The rule will fail if the driver licence has 4 or more endorsements in the last 4 years from categories G, H, and I combined.	EP & BVA: totalEndorsements < 4 where date < 4 years
BR5	DAG_V2_DRIVER_LICENCE_CATEGORY_G_ENDORSEMENT	The rule will fail if the driver has two or more endorsements in the last 4 years from category G and has a conviction date in the past 4 years.	EP & BVA: endorsementsCatG < 2 where date < 4 years && convictionDate=null || >4 years
BR6	DAG_V2_DRIVER_LICENCE_CATEGORY_F_ENDORSEMENT	The rule will fail if the driver has any endorsements in the last 4 years from category F and has a conviction date in the past 4 years.	EP & BVA: endorsementsCatF == 0 where date < 4 years && convictionDate=null || >4 years

Eligibility Checks

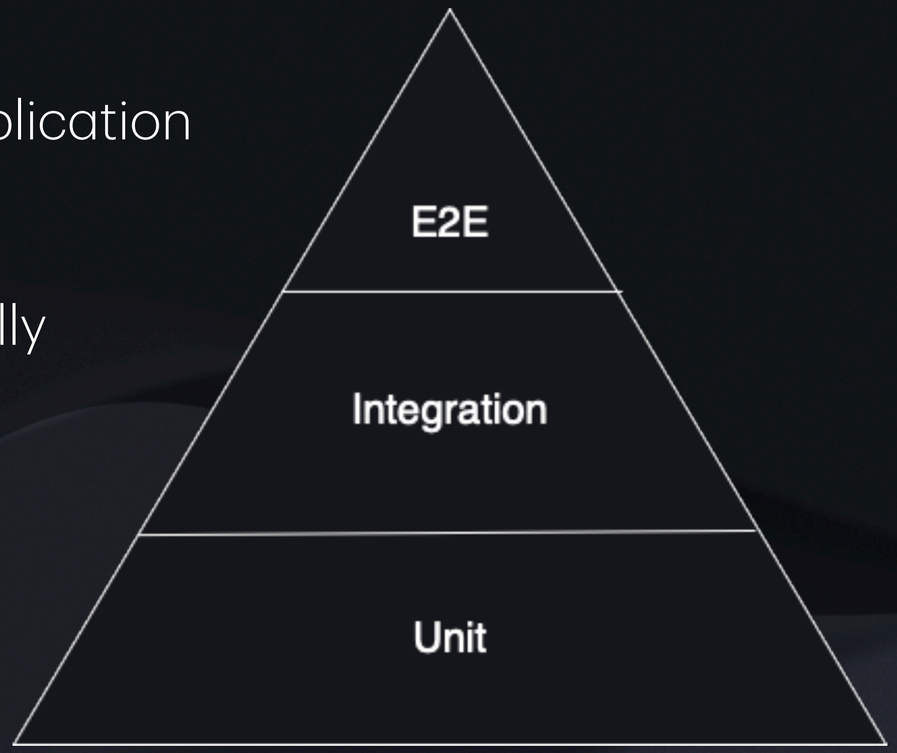
Non-Functional Testing

- “Typical peak traffic for these end eligibility end points is 12,000 calls per hour with up to 50 concurrent users. There is an agreed service level API response time of less than a second.”
- Load test = Consistent load of 12,000 requests per hour and up to 50 virtual users/threads
- Stress test = Incrementally ramp up the load to above the expected load e.g. ramping up by 2,000 requests per hour and 15 users every hour until 20,000 requests per hour and up to 75 users
- Spike test = Steady state load of 12,000 requests per hour and up to 50 users with periodic spikes e.g. 24,000 requests per hour and up to 100 users lasting 1 minute
- Soak test = Production-representative load pattern running for an extended duration e.g. run for 1 week and use monitoring data from production to model load pattern over the week
- For all test types, our success criteria is determined by the SLA API response time of < 1 second.

End-To-End Testing

End-To-End Testing

- Completes the test pyramid approach to testing
- Test the full application flow from submitting the initial application to receiving the application outcome.
- Also consider unhappy path scenarios e.g. user accidentally enters incorrect information or user is not eligible
- Relies on strong foundation of unit and integration testing
- Conduct exploratory testing to draw on the tester’s experience and knowledge of common issues



Implementation

Implementation

- Begin with unit testing (e.g. API, UI, performance, business rules)
  - Catch defects as early as possible in the development process
  - Can be run locally by developers and on commit to the repository via CI/CD
- Implement integration testing approach (e.g. DVLA system, databases)
  - Detect defects/breaking changes in how we interact with other components or external systems
  - Runs against each commit to the repository via CI/CD
- End-to-end testing approach
  - Runs periodically via CI/CD (e.g. nightly)
- Plan and execute manual performance testing experiments and exploratory sessions as areas of risk are identified