PegaWorld

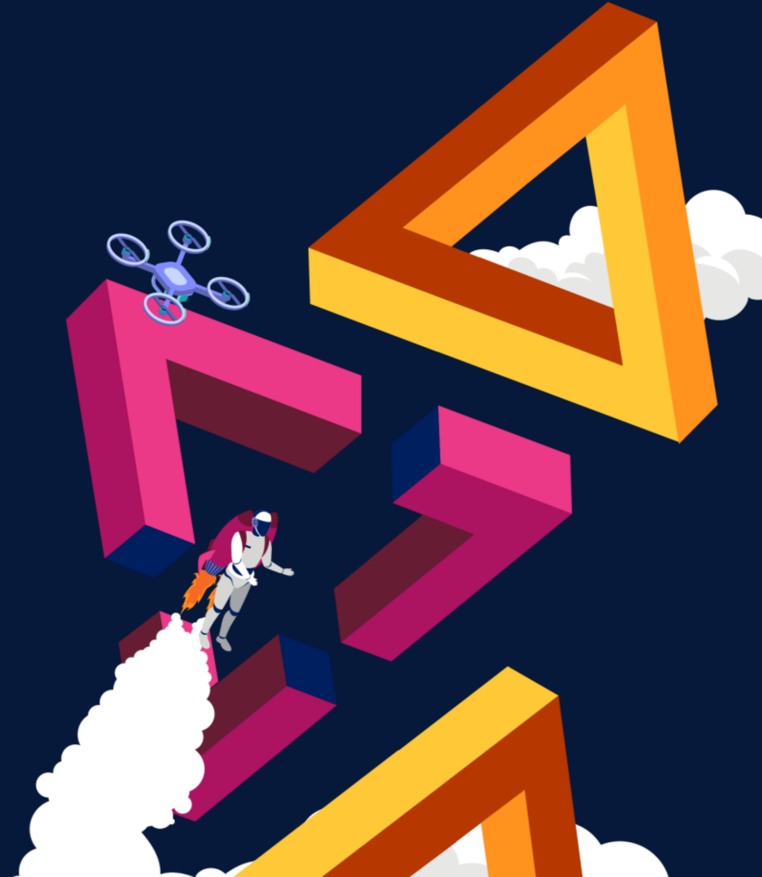
JUNE 1-3, 2025 | LAS VEGAS

Information shared during PegaWorld is not an offer or commitment by Pegasystems and does not create any legal obligation for Pegasystems, including to deliver any material, code, or functionality. The timing of the development and release of any features or functionality described about our products remains at our sole discretion. ©2025 Pegasystems Inc.

From Chaos to Control:

Testing & Releasing Dynamic Pega App with Smart Test Automation

Booking.com



Harneek Singh

- 1.5 year at booking
- In Netherlands ~ 8 years
- Engineering Manager test at Booking Customer Service
- Ensuring high quality delivery of Pega applications
- 14 years of experience in Quality
- Love for Tennis 🎾

Make it easier for everyone to experience the world.

Support every customer with any query the first time around.

Where did we begin? And now, where are we?

2019

Engagement with Pega to introduce a CRM

Continuously improve booking's customer experience
&
make every connection easy by leveraging Pega's abilities to optimize workflows and AI

2023

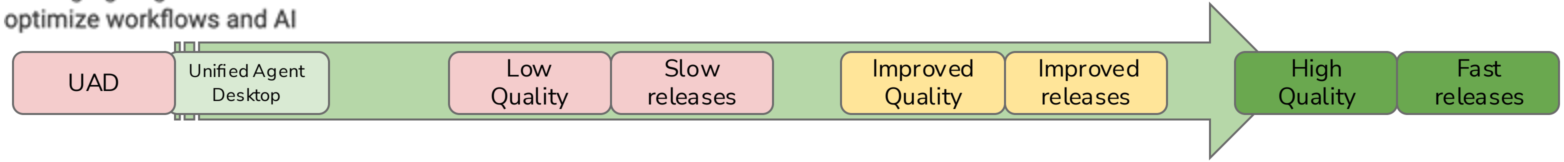
- Development in silos
- No proper regression testing
- Releases overwrite each other
- Manual sanity in isolation

2024

- Testers part of teams working closely and with design
- Bi-weekly releases
- Unit tests part of CI/CD
- Automated deployments

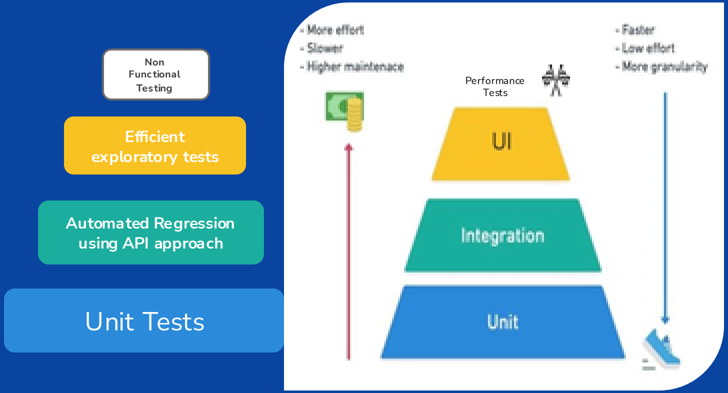
Now

- Automated sanity on all environments
- Automated Regression
- Effective performance tests



So what did we do, and how did we do it?

Sticking to the basic test pyramid helped us in solving our challenges and transitioning us from “chaos” to “control”



Non Functional Testing

Efficient exploratory tests

Automated Regression using API approach

Unit Tests

Automating Regression

Manual regression testing is an effort extensive task, automating regression enables us to release application faster.

Automating Sanity Testing

Sanity being precise but repetitive also consumes a lot of time.

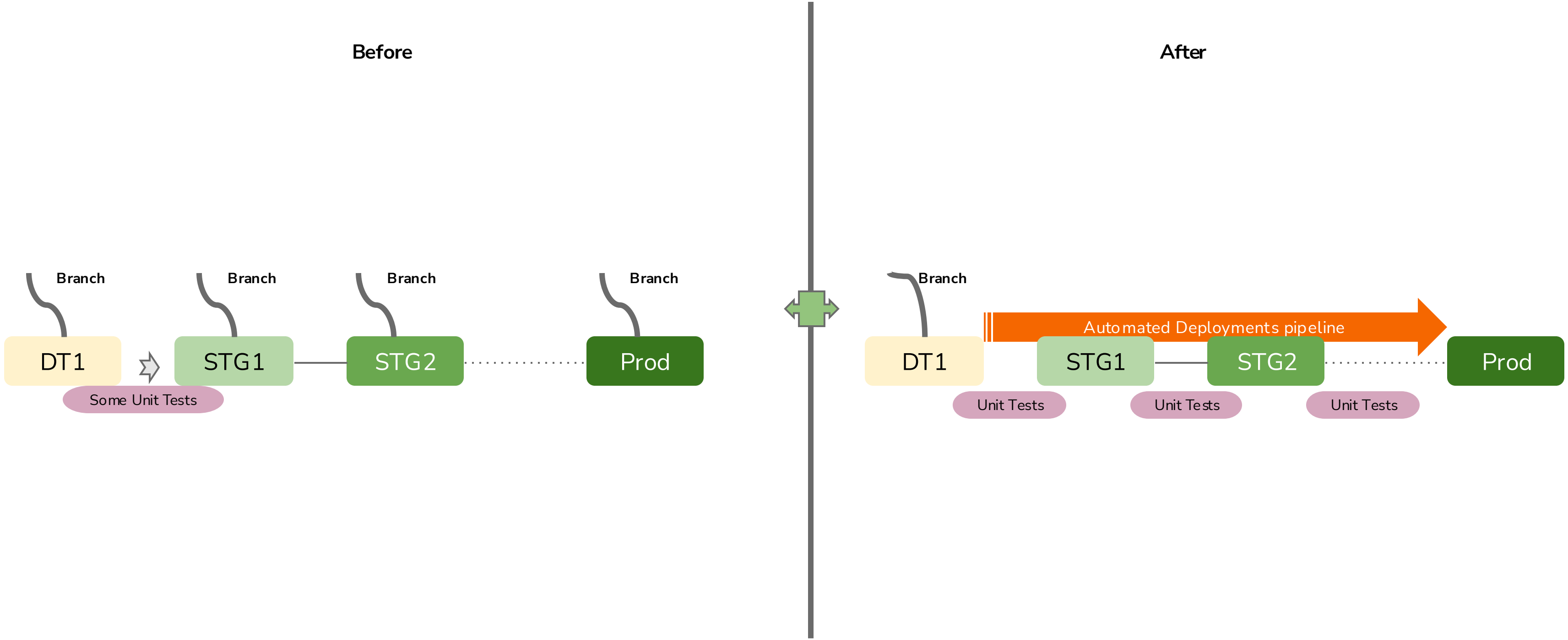
Focused And Faster Regression

When the regression is automated, testers can focus on other important factors like automating more and focused exploratory testing.

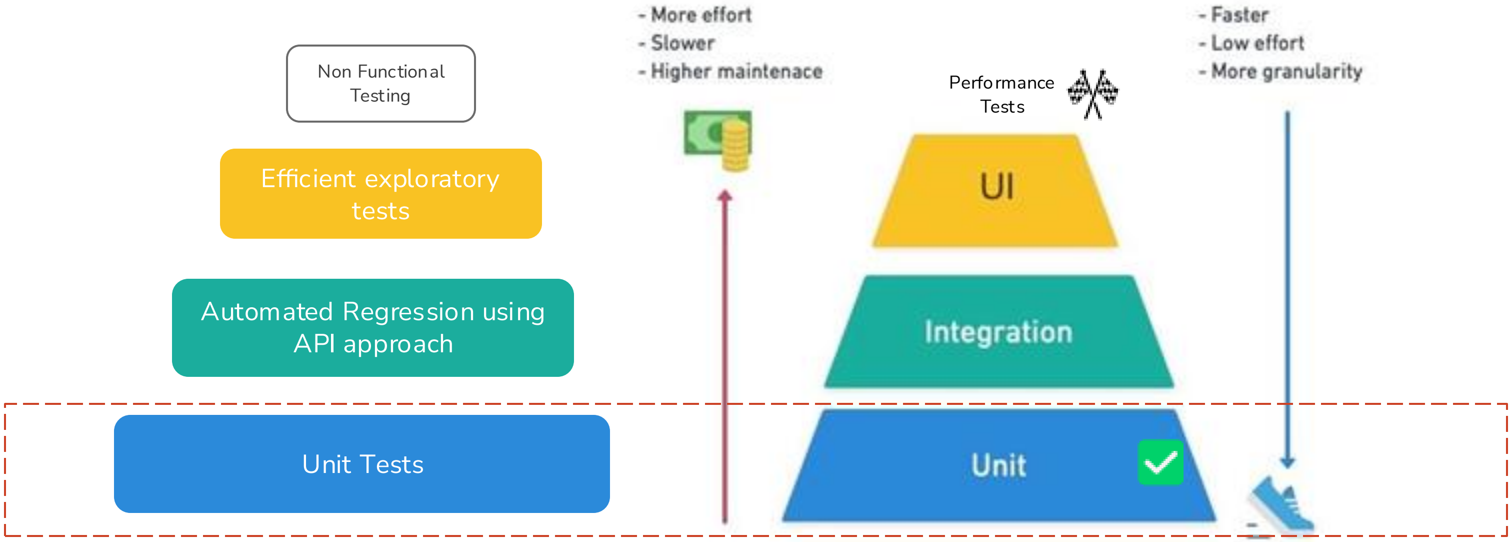
Making sure that the application performs “well”

With an application having a peak load of 12K agents, it is essential to ensure that the application performs good under load.

Unit tests are part of the deployment pipeline for every downstream environment & no long running branches



Making the base stronger



Non Functional Testing

Efficient exploratory tests

Automated Regression using API approach

Unit Tests

Testing At Scale, Quickly

Multiple teams and testers handle regression testing of the Pega application

Full regression testing takes five days to complete currently.

Need faster releases with more frequent regression testing cycles.

Automating the regression process to enable faster releases.

Challenges in automating a dynamically changing application like Pega CS UAD

Dynamically changing apps pose significant automation challenges.

Very frequently changing UI results in failures and ongoing maintenance.

Scaling automation for large systems is a complex task.

Teams adding new iterations of features make automation a tricky and unmaintainable task.

Three approaches to automate regression were evaluated

POC

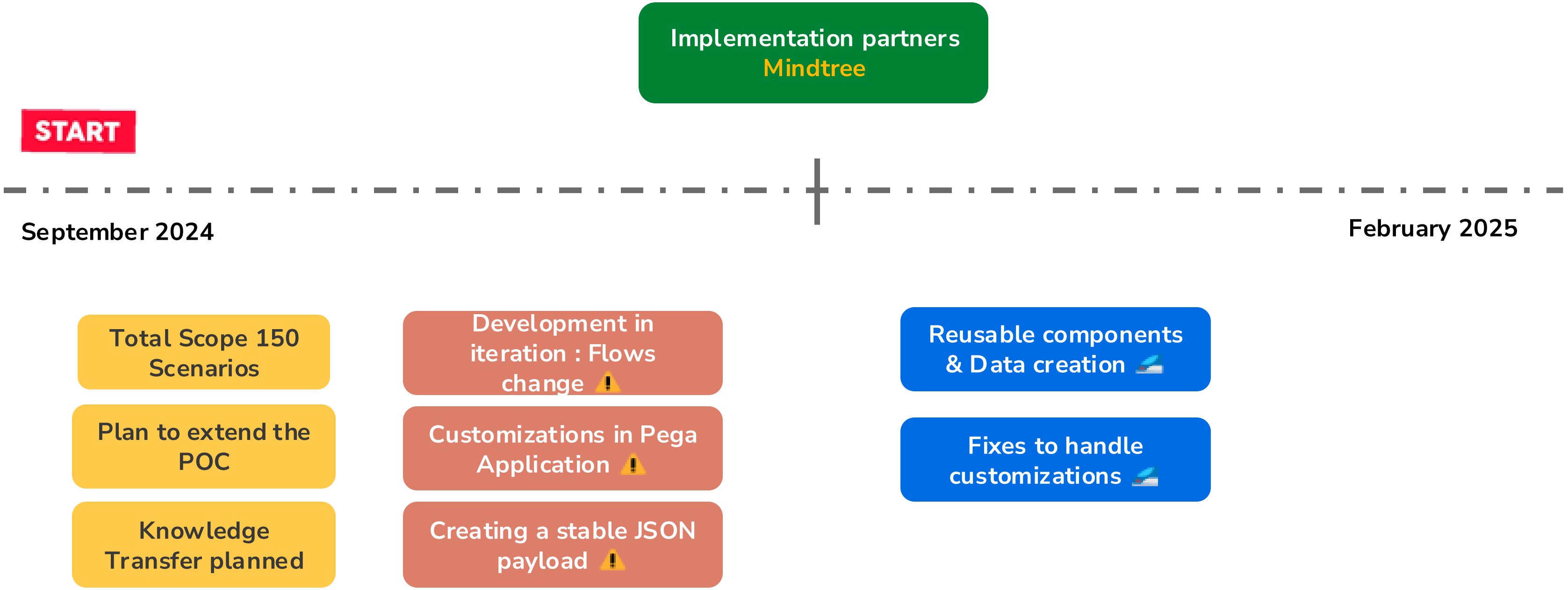
	1	2	3
OPTION	➔ API automation for everything that can be automated. ➔ **Manually** execute pending scenarios.	➔ Automate via API until the step it can be automated via API approach ➔ Switch to UI or vice versa	➔ Automate flows via API approach ➔ Execute some scenarios in UI approach for cross browser testing ➔ UI approach is focused on executing flows which are impossible to automate via API (not more than 10%) ➔ Focus on coverage rather than the number of scenarios.
	No Coverage	Complicated Hybrid	Coverage & stability Only API
DRAWBACK	➔ Limited automation coverage ➔ Still relying on manual regression ➔ Time-consuming regression	➔ Complicated to design such scenarios ➔ Switching from UI to API or vice versa may induce flakiness	➔ UI tests are subject to rapidly changing UI ➔ If not coded properly, might result in flakiness

Based on the results of POC we scored the options

Selected

Parameter	Option 1	Option 2	Option 3
Learning Curve	Steep	Medium	Low
Maintenance frequency	Low	Medium	Low
Maintenance ease	High	Medium	High
Execution speed	High	Medium	High
Failure rate	Medium	High	Low
Reusability	High	Medium	High

Journey of creating API based automated framework : Start



Implementation partners
Mindtree

START

September 2024

February 2025

Total Scope 150 Scenarios

Plan to extend the POC

Knowledge Transfer planned

Development in iteration : Flows change ⚠️

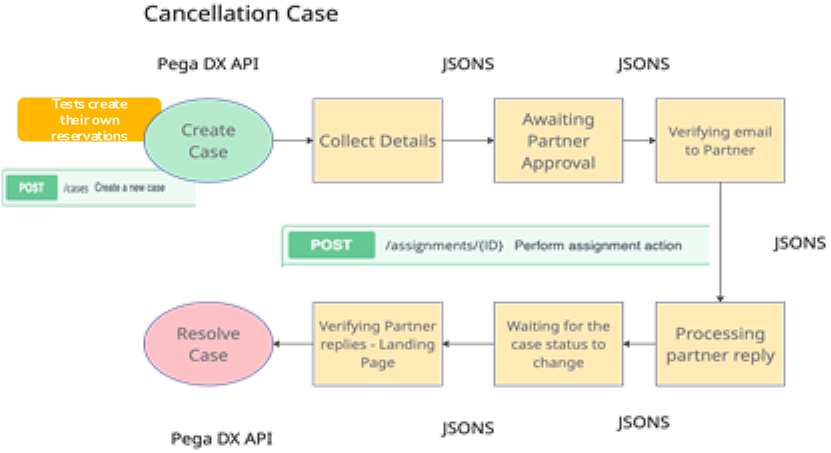
Customizations in Pega Application ⚠️

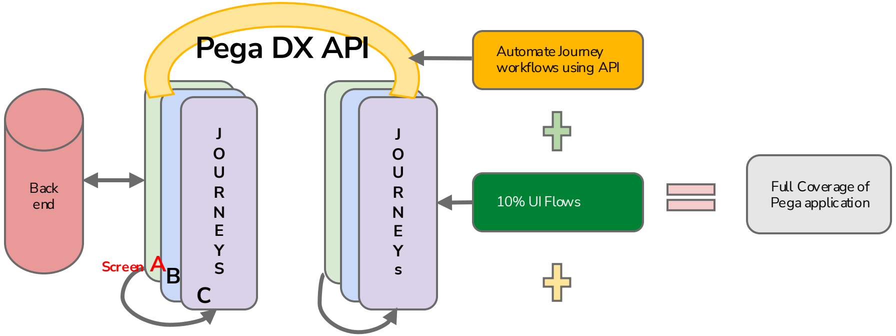
Creating a stable JSON payload ⚠️

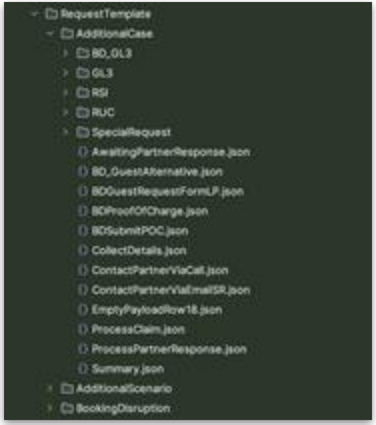
Reusable components & Data creation 👟

Fixes to handle customizations 👟

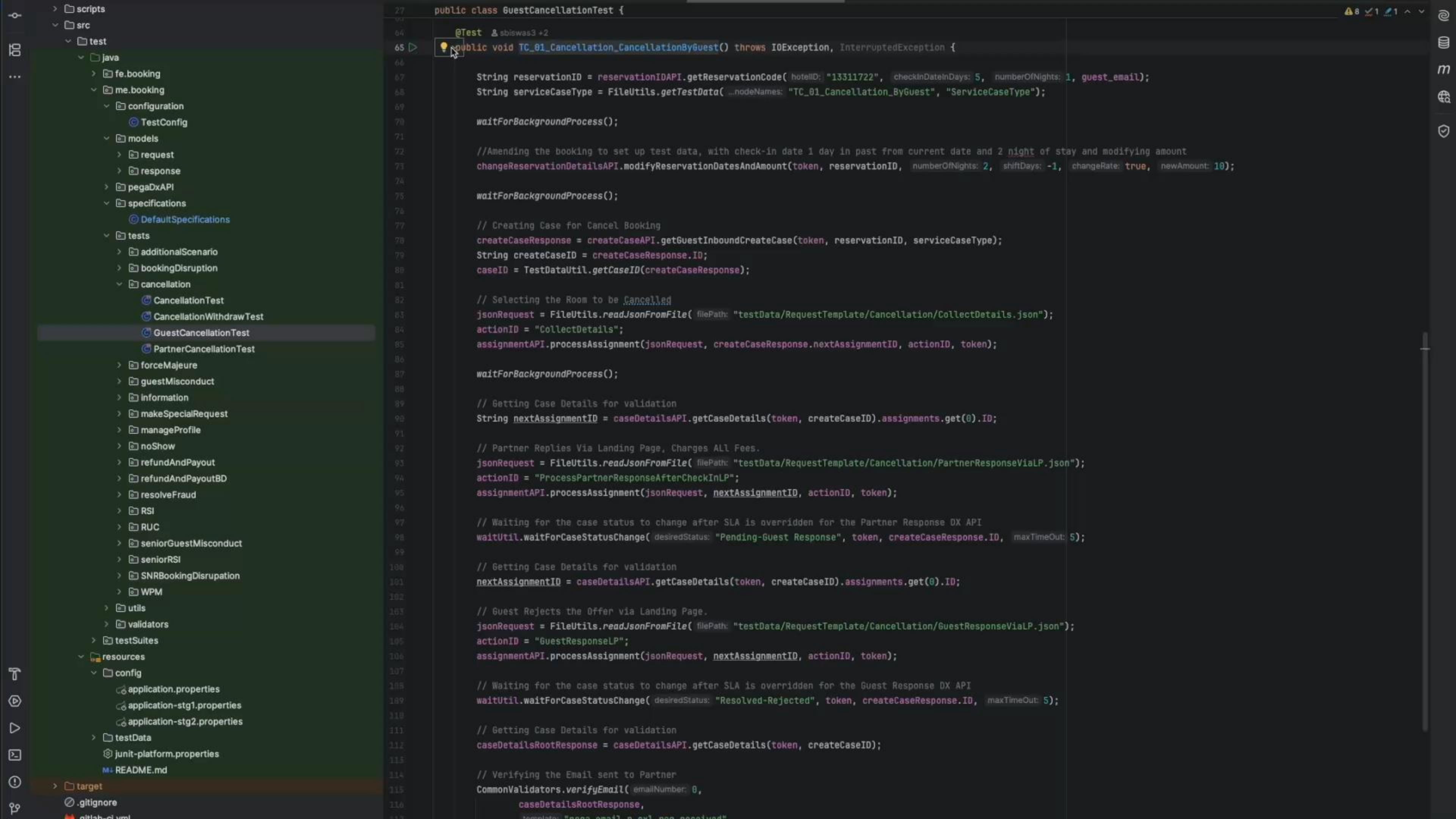
Deconstructing the API Approach


Cancellation Case
Pega DX API
JSONS
JSONS
Tests create their own reservations
Create Case
POST /cases Create a new case
Collect Details
Awaiting Partner Approval
Verifying email to Partner
POST /assignments/{ID} Perform assignment action
JSONS
Resolve Case
Verifying Partner replies - Landing Page
Waiting for the case status to change
Processing partner reply
Pega DX API
JSONS
JSONS





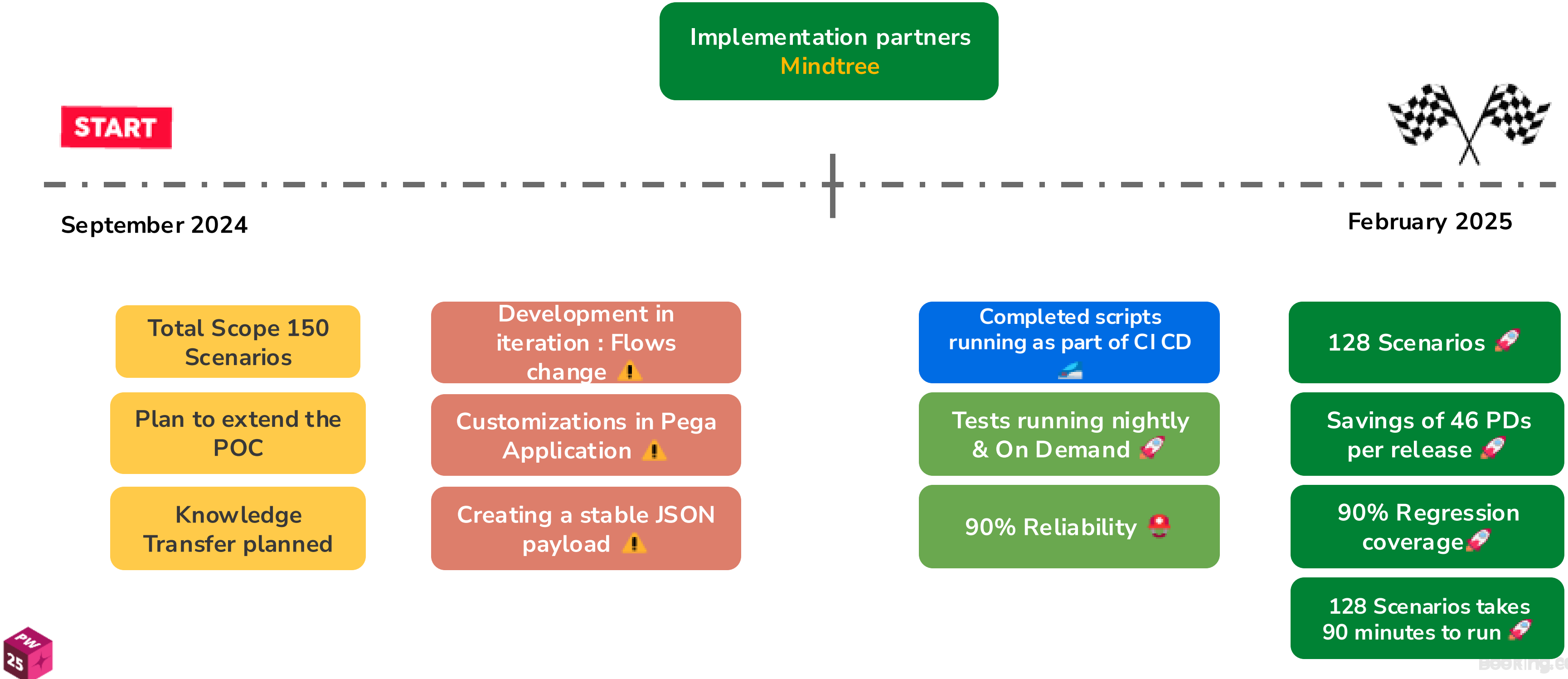
```
⌄ 🗀 RequestTemplate
  ⌄ 🗀 AdditionalCase
    › 🗀 BD_GL3
    › 🗀 GL3
    › 🗀 RSI
    › 🗀 RUC
    › 🗀 SpecialRequest
      {} AwaitingPartnerResponse.json
      {} BD_GuestAlternative.json
      {} BDGuestRequestFormLP.json
      {} BDProofOfCharge.json
      {} BDSubmitPOC.json
      {} CollectDetails.json
      {} ContactPartnerViaCall.json
      {} ContactPartnerViaEmailSR.json
      {} EmptyPayloadRow18.json
      {} ProcessClaim.json
      {} ProcessPartnerResponse.json
      {} Summary.json
  › 🗀 AdditionalScenario
  › 🗀 BookingDisruption
```



```
> scripts
v src
  v test
    v java
      > fe.booking
      v me.booking
        v configuration
            TestConfig
        v models
          > request
          > response
        > pegaDxAPI
        v specifications
            DefaultSpecifications
        v tests
          > additionalScenario
          > bookingDisruption
          v cancellation
              CancellationTest
              CancellationWithdrawTest
              GuestCancellationTest
              PartnerCancellationTest
          > forceMajeure
          > guestMisconduct
          > information
          > makeSpecialRequest
          > manageProfile
          > noShow
          > refundAndPayout
          > refundAndPayoutBD
          > resolveFraud
          > RSI
          > RUC
          > seniorGuestMisconduct
          > seniorRSI
          > SNRBookingDisrupation
          > WPM
        > utils
        > validators
      > testSuites
    v resources
      v config
          application.properties
          application-stg1.properties
          application-stg2.properties
      > testData
        junit-platform.properties
      README.md
> target
  .gitignore
```

```java
public class GuestCancellationTest {

    @Test  sbiswas3 +2
    public void TC_01_Cancellation_CancellationByGuest() throws IOException, InterruptedException {

        String reservationID = reservationIDAPI.getReservationCode( hotelID: "13311722",  checkInDateInDays: 5,  numberOfNights: 1, guest_email);
        String serviceCaseType = FileUtils.getTestData( ...nodeNames: "TC_01_Cancellation_ByGuest", "ServiceCaseType");

        waitForBackgroundProcess();

        //Amending the booking to set up test data, with check-in date 1 day in past from current date and 2 night of stay and modifying amount
        changeReservationDetailsAPI.modifyReservationDatesAndAmount(token, reservationID,  numberOfNights: 2,  shiftDays: -1,  changeRate: true,  newAmount: 10);

        waitForBackgroundProcess();

        // Creating Case for Cancel Booking
        createCaseResponse = createCaseAPI.getGuestInboundCreateCase(token, reservationID, serviceCaseType);
        String createCaseID = createCaseResponse.ID;
        caseID = TestDataUtil.getCaseID(createCaseResponse);

        // Selecting the Room to be Cancelled
        jsonRequest = FileUtils.readJsonFromFile( filePath: "testData/RequestTemplate/Cancellation/CollectDetails.json");
        actionID = "CollectDetails";
        assignmentAPI.processAssignment(jsonRequest, createCaseResponse.nextAssignmentID, actionID, token);

        waitForBackgroundProcess();

        // Getting Case Details for validation
        String nextAssignmentID = caseDetailsAPI.getCaseDetails(token, createCaseID).assignments.get(0).ID;

        // Partner Replies Via Landing Page, Charges ALL Fees.
        jsonRequest = FileUtils.readJsonFromFile( filePath: "testData/RequestTemplate/Cancellation/PartnerResponseViaLP.json");
        actionID = "ProcessPartnerResponseAfterCheckInLP";
        assignmentAPI.processAssignment(jsonRequest, nextAssignmentID, actionID, token);

        // Waiting for the case status to change after SLA is overridden for the Partner Response DX API
        waitUtil.waitForCaseStatusChange( desiredStatus: "Pending-Guest Response", token, createCaseResponse.ID,  maxTimeOut: 5);

        // Getting Case Details for validation
        nextAssignmentID = caseDetailsAPI.getCaseDetails(token, createCaseID).assignments.get(0).ID;

        // Guest Rejects the Offer via Landing Page.
        jsonRequest = FileUtils.readJsonFromFile( filePath: "testData/RequestTemplate/Cancellation/GuestResponseViaLP.json");
        actionID = "GuestResponseLP";
        assignmentAPI.processAssignment(jsonRequest, nextAssignmentID, actionID, token);

        // Waiting for the case status to change after SLA is overridden for the Guest Response DX API
        waitUtil.waitForCaseStatusChange( desiredStatus: "Resolved-Rejected", token, createCaseResponse.ID,  maxTimeOut: 5);

        // Getting Case Details for validation
        caseDetailsRootResponse = caseDetailsAPI.getCaseDetails(token, createCaseID);

        // Verifying the Email sent to Partner
        CommonValidators.verifyEmail( emailNumber: 0,
                caseDetailsRootResponse,
```

Journey of creating API based automated framework: In the end



Implementation partners
Mindtree

START

September 2024

February 2025

Total Scope 150 Scenarios

Plan to extend the POC

Knowledge Transfer planned

Development in iteration : Flows change ⚠️

Customizations in Pega Application ⚠️

Creating a stable JSON payload ⚠️

Completed scripts running as part of CI CD 🚄

Tests running nightly & On Demand 🚀

90% Reliability 🚨

128 Scenarios 🚀

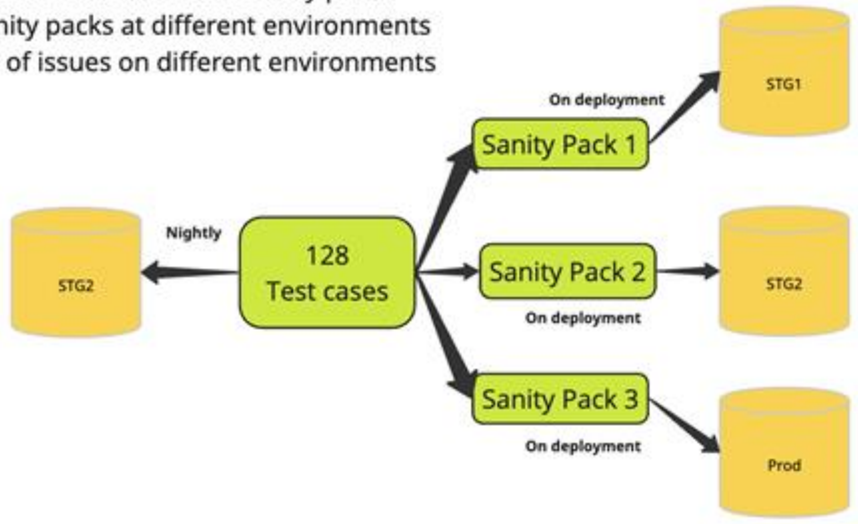
Savings of 46 PDs per release 🚀

90% Regression coverage🚀

128 Scenarios takes 90 minutes to run 🚀

Fortifying our development lifecycle: Automated Sanity

- Slice the automated test cases into sanity packs
- Execute the sanity packs at different environments
- Early detection of issues on different environments



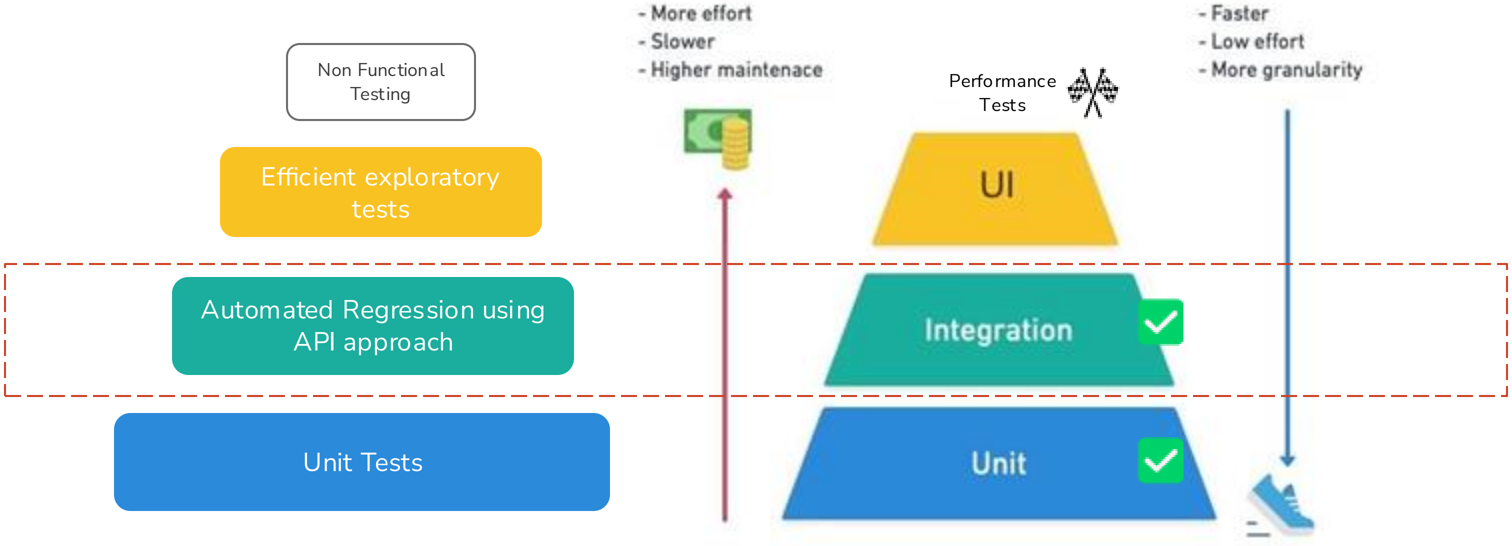
Automated Health check on production

Tests triggered by deployment

Sanity tests running on all downstream environments

Release anytime

Making the integration layer stronger



Non Functional Testing

Efficient exploratory tests

Automated Regression using API approach

Unit Tests

Regression focused on new features and exploratory checks

Since the regression is automated testers are focused on

Regression testing new features

Ensuring that new features that are recently added to the release are working with high quality

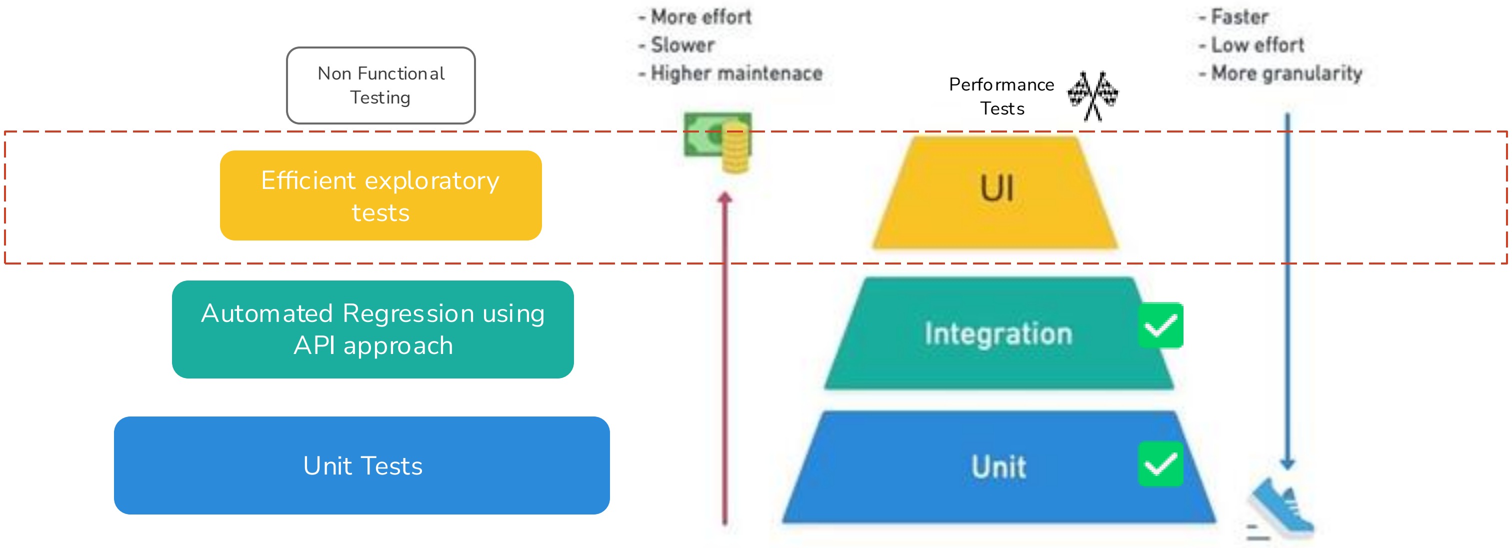
Exploratory testing the flows and UI

Testers are focused on doing a targeted time based exploratory checks on UI and workflows

End user testing perspective

Using the application like an end user, which is agent in our case and finding bugs that can degrade end user experience.

Regression focused



Non Functional Testing

Efficient exploratory tests

Automated Regression using API approach

Unit Tests

Performance Tests

Comparing the past to the present

Before

- Regression took **4 days for 23** testers
- Due to a shortage of time, regression was only done on important flows.
- After deployment STG2 it used to take 1 day to get feedback on build stability
- Not much focus was on UI and End to End flows

Now

- Regression takes **2 days for 23** testers
- Entire application is tested automatically
- The tests run **every night,** and by the time testers start work, we have the results of tests, and we can specify build stability.
- Regression is focused and testers can exploratory test Ui and end to end user flows

01

Works on constellation with minor changes

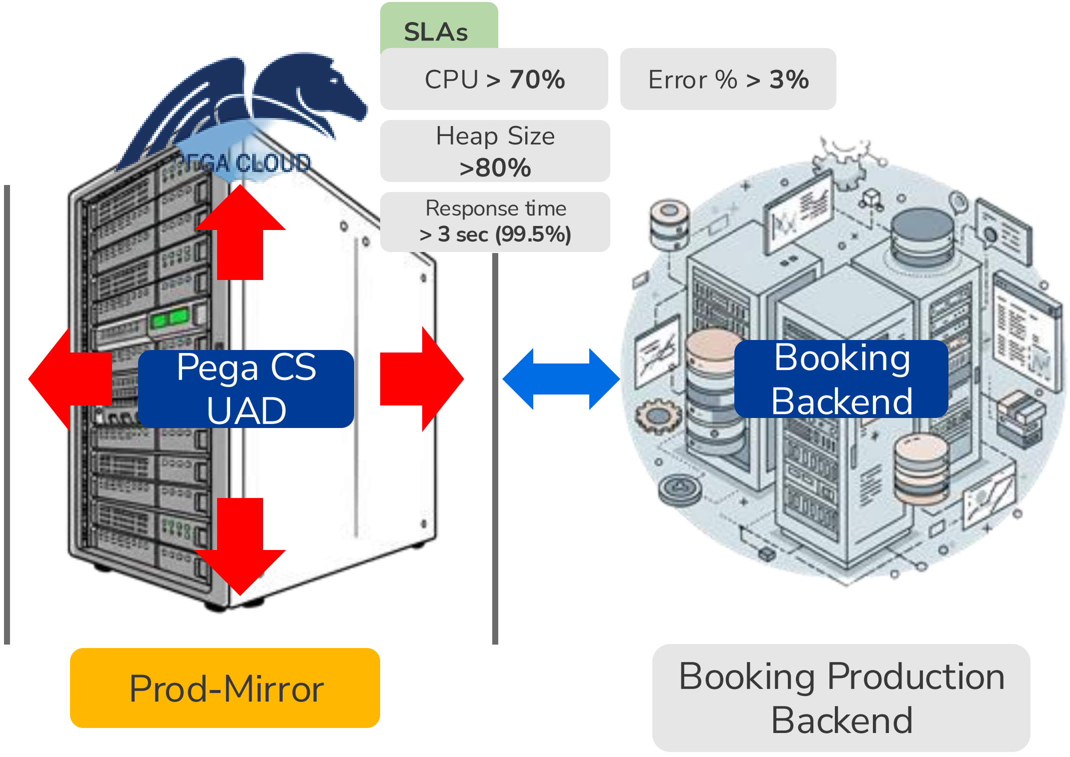
02

90% more reliable

03

Faster feedback

Ensure that Pega application is performing “well” under expected peak load



SLAs

CPU > **70%**

Error % > **3%**

Heap Size **>80%**

Response time **> 3 sec (99.5%)**

Prod-Mirror

Booking Production Backend

- Testing the Pega CS UAD in production is complex due to its connection to the booking backend.

12K concurrent agents

15K agent base 2025

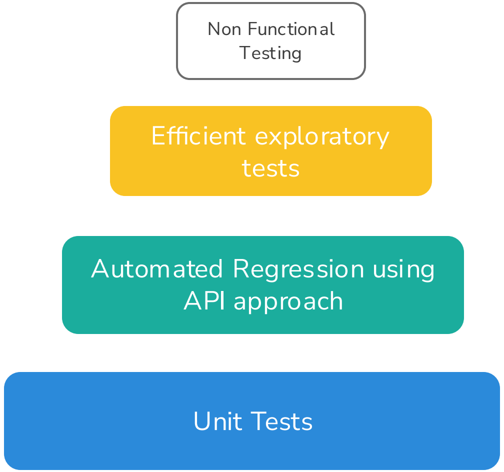
20 web nodes on production

10 batch nodes on production

Infra Performance Tests — **To size the infra for the expected load**

Application Performance tests — **Remove App performance bottlenecks**

Regression focused

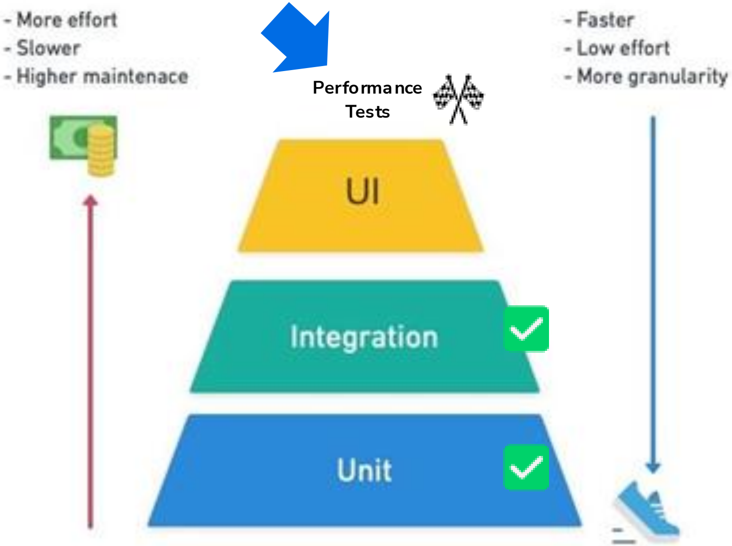


Non Functional Testing

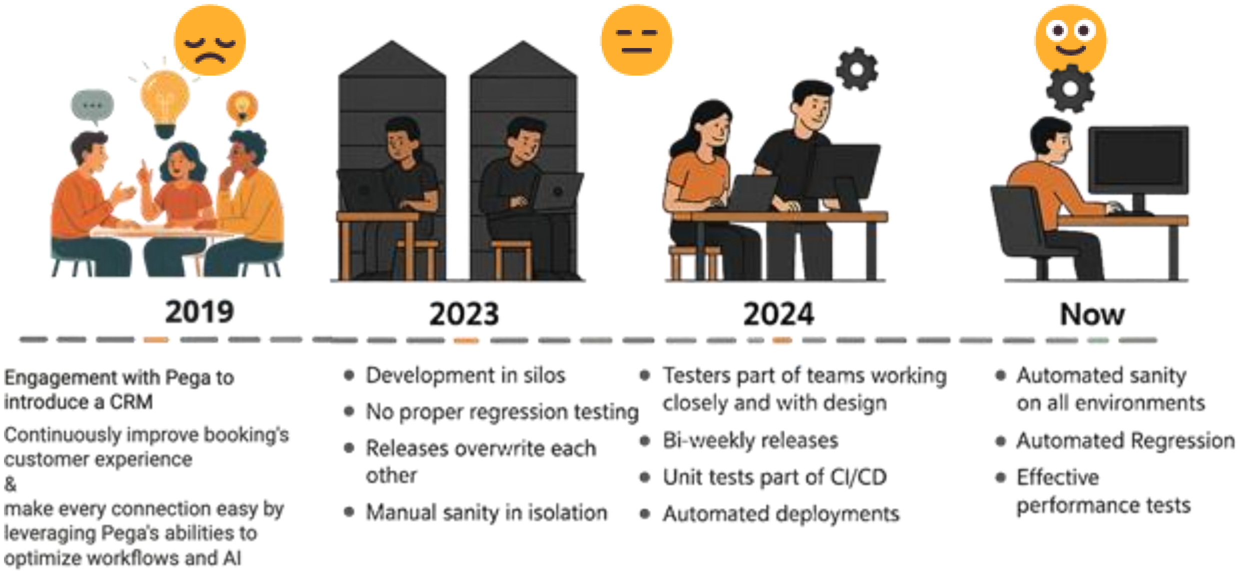
Efficient exploratory tests

Automated Regression using API approach

Unit Tests



Transition from Chaos to Control?



2019

Engagement with Pega to introduce a CRM

Continuously improve booking's customer experience
&
make every connection easy by leveraging Pega's abilities to optimize workflows and AI

2023

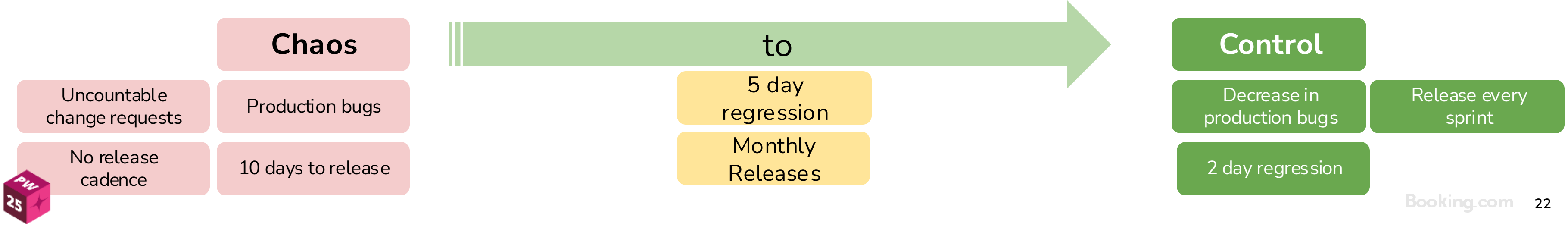
- Development in silos
- No proper regression testing
- Releases overwrite each other
- Manual sanity in isolation

2024

- Testers part of teams working closely and with design
- Bi-weekly releases
- Unit tests part of CI/CD
- Automated deployments

Now

- Automated sanity on all environments
- Automated Regression
- Effective performance tests



Chaos

Uncountable change requests

Production bugs

No release cadence

10 days to release

to

5 day regression

Monthly Releases

Control

Decrease in production bugs

Release every sprint

2 day regression

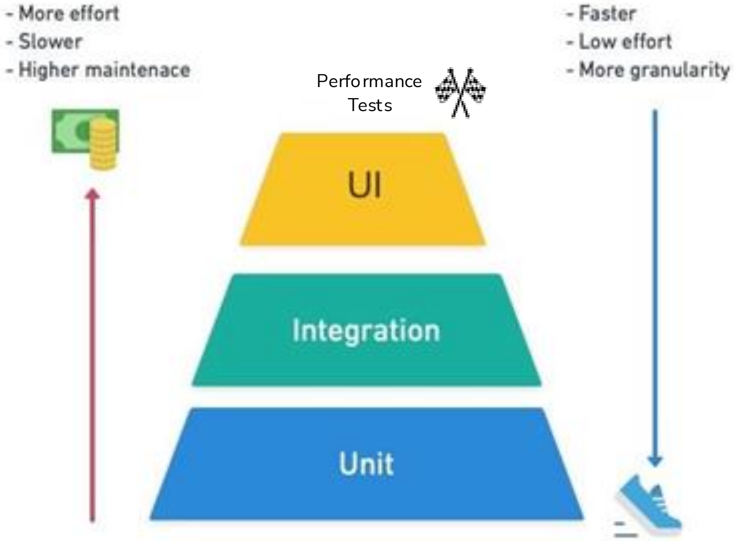
Focusing on the basics : The quality pyramid

Non Functional Testing

Efficient exploratory tests

Automated Regression using API approach

Unit Tests



Thank you!

Presenter Harneek Singh
harneek.singh@booking.com

Booking.com

PegaWorld

JUNE 1-3, 2025 | LAS VEGAS