Based on the provided conditions, here's a revised ballpark estimate for the implementation of Automation Tests for the Shopping Cart and Checkout modules using Selenium WebDriver:

Ballpark Estimate: 80-160 man-work-days (based on the assumption that each test case requires approximately 0.5 to 2 days to automate, including test setup, execution, and maintenance)

Factors influencing the estimate:

Number of Test Cases: The range of 300-500 test cases already prepared can significantly impact the effort required for automation. The more test cases there are, the longer it may take to automate them all.

Complexity of the Pages: The complexity of React and Java/Tapestry pages can affect the automation effort. If the pages have intricate UI elements, dynamic content, or complex interactions, it may take longer to develop stable and reliable automation scripts.

Payment Types: The set of payment types adds complexity to the checkout process. Each payment type may require specific automation scenarios and validations, increasing the effort required for automation.

Lack of Existing Automation Tests: Starting automation from scratch without any existing automation framework or test scripts may require additional time for setting up the automation infrastructure, test environment, and creating reusable automation components.

Learning Curve: If the team is not already familiar with Selenium WebDriver, there may be a learning curve involved in understanding the tool, setting up the automation framework, and implementing best practices. This could influence the overall effort.

Risks:

Maintenance Effort: As the application evolves, maintenance of automation tests becomes crucial. Changes in UI, functionality, or business rules may require updates to existing test scripts, which could add to the overall effort.

Test Data Management: Proper management of test data, including various payment types, products, and user scenarios, is crucial for effective automation. Ensuring test data accuracy and maintaining it over time can be a challenge.

Test Stability and Reliability: Automating complex scenarios and UI interactions requires careful handling to ensure stability and reliability. Flaky tests or false positives/negatives can impact the trustworthiness of the automation suite.

Integration and Dependency Challenges: Integration with payment gateways, external services, or third-party APIs can introduce complexities and dependencies that need to be addressed during automation.