To answer the question whether automation is required for WODP project, or not, they answer is yes. There are several reasons, which seem to me as the main ones, specifically:

1. While WODP project itself, as part of my job, is limited to ~1 year of development and with only 1 release as a result – after being released, the project will enter CI/CD pipelines and will continue to operate with continuous delivery in mind until it reaches end of life state, or customer makes another decision which impacts its life cycle. In any case, however, it’s predicted that current project will lay the foundation for at least several years of continuous improvements. Not all, but most of the autotests created before the actual release of WODP products can, and should, be reused later. Framework itself, especially, can stay the same unless it’s decided by the customer to change it in the future. Keeping this in mind, automation might not seem like the best way to go, when looking at a project as a 1-year job, however it does bring value to the customer in the future, regardless of whether EPAM stays its offshore partner or not.
2. During the active development itself, due to the limited amount of time before the sign-off of the project, manual testers do not have time to perform smoke tests or regression passes on the scale which the project would require, considering the number of bugs, components, and services it uses. To save the time and allow manual testers to proceed with the active development changes, instead of focusing on repeated tests which, ideally, should ensure the daily stability of builds, tests are automated along or right after the manual testers finish writing the manual test cases. Afterwards, automated test suites can run before/after each deploy without the need to involve manual testing into it (with some minor exceptions which only negligently affect the amount of needed efforts).
3. Along with the dynamic features which are often changed or affected by others, the project also provides many features which, upon being implemented and tested by manual QA, will not be affected as much by improvements or other bugs. These features, which can be considered stable, can be automated quickly and with reduced need for maintenance of the automated tests. In the end it not only ensures the stability, but also catches the unlikely regression which can be easily missed by the manual QA engineers due to limited attention being paid to the stable features.

To answer the question of what should/could be automated on WODP project, and how, first I would like to mention the following:

1. Code should be surrounded with unit tests. The reason for it is obvious and lies in the fact that code should be tested on the unit level before the effort is spent on anything more significant.
2. Test cases related to API requests and responses can be quickly and relatively easily automated. For example, through Postman and collections, however it can also be done as a part of single framework (which I cannot go into the details yet, unfortunately). The reason for this is that they are more isolated than UI tests, however they do affect the E2E scenarios, AND, additionally, to ensure stability with manual efforts, would require constant regression passes by manual testers. Isolating them and automating them on their own would help to reduce the instability on the UI level, while at the same time keeping tests consistent and separated, due to dealing with separate API calls.
3. UI tests can and should be automated in the later parts of the project, but only in the case of static, unlikely-to-change logic. As an example, many of the pages and features which are provided by the project, are static in its logic. While the exact view of, for example, the available product and its features, might change from one instance to another, this can be circumvented by creating a test page / product / data which would comply with business requirements and at the same would be sufficient to cover the tests which require UI interactions. This approach would require careful test mapping, to ensure that the test coverage is sufficient but is not bloated, however in the end it significantly reduces the manual efforts, and quickly catches regression issues in the basic logic and/or functionality of components. This can be automated, for example, in combination of Selenium with BDD framework such as Cucumber.
4. Smoke tests, naturally, should be automated as well. Exactly how – would be defined by the scope of the smoke test.

Other, specific types of tests such as performance, localization, usability etc., in my opinion, should not be automated due to, mainly, one of two reasons:

* The test type is too ambiguous, such as in case with usability or accessibility
* The test type covers the area which is either too dynamic (such as localization, which can be changed anytime) or cannot be properly executed before the actual release (such as performance, which can be automated only right before or after the actual release)

As for the ROI, for the purpose of the exercise let’s imagine that this is the beginning of the project, and I must evaluate it. **Detailed ROI is attached to this email. Basic ROI is below.**

Note that for the detailed ROI, the percentage came out as 2351%, mainly because Canadian Tire net income is so large that it’s easy to see that there is benefit in automation.

**Basic ROI:**

Here I am again disregarding the possibility of support, since I do not know for how long the support will go on. Considering that support will not necessarily be carried out by EPAM, we cannot use the same values for two years – cost of manual testing or creation of automated test cases mainly lies within the first year, before the actual release.

Time scale: 1 year of development (**12 months**).

Cost of manual testing: 30 hours per week, for 12 months = **1440 man-hours**

TAF implementation – **120 man-hours** (considering the size of the project I think three weeks is more or less accurate)

Cost of creating automated test cases: 30 man-hours a week for 8 months of active development and 10 hours a week for the remaining 4 months = **1120 man-hours**

Cost of executing automated test cases and analyzing results: 10 man-hours for 12 months = **480 man-hours**

**ROI is: (1440 – (120 + 1120 + 480)) / (120 + 1120 + 480) = -0.16**

Conclusion: basic ROI calculator is not suitable for this case, since it does not take into an account amount of people, first of all, and future benefits of having automated test cases. Since there is no guarantee that EPAM will support the project for the next years, we cannot also confirm what is the amount of time which will be spent on either manual or automated testing. However, we can see that for the first year, even with the simplest calculator, the breakoff is almost at 0, which also means that even for the first year there is no big disadvantage for the automation testing – however it will go hand in hand with manual testing, which is, actually, exactly what happened and what could have been expected.