Improvement Plan:

1. Deep Understanding of JavaScript and TypeScript

- Step 1: Enroll in online courses or webinars for JavaScript and TypeScript.

- Step 2: Study and practice JS and TS through coding exercises and projects.

- Step 3: Implement the same task both in JS and TS, learn differences and similarities.

2. Test Automation Development in JS Ecosystem - According to my CV I have more then 4 years of experience in automation on TS and 3 years in JS. But of course I’m continue learning:

**[Automated Testing Mentoring with JS: Basic [self-study]](https://learn.epam.com/detailsPage?id=ba41231e-de87-46da-8bc9-c71fc69e1c26)**

**[TypeScript In-Depth #34](https://learn.epam.com/detailsPage?id=22634132-63c1-4a96-87f8-44f82526e406)**

- Step 1: Read tutorials and watch videos on Test Automation Development using JS.

- Step 2: Practice creating test automations in JS.

3. Setting Up Framework from Scratch - - I’m sorry that I couldn’t share with you my experience in it. But based only on Epam projects:

1. Thin Monitor – in 2 projects was set up **Selenium Framework from Scratch**
2. LSEG - was set up **Cypress Framework from Scratch**
3. Merck – is set up **PlayWright Framework from Scratch**

- Step 1: Investigate best practices for framework setup.

- Step 2: Reach out to colleagues for assistance in creating a solution from scratch, or study previously built frameworks.

- Step 3: Attempt to build a simple test automation framework from scratch.

4. Understanding UI Automation Tools - I’m sorry that I couldn’t share with you my rich experience in all popular UI Tools. You can check my CV with real project implementation from the start to the CI:

* **WebDriver IO**: 6 years
* **Selenium**: 2 years
* **Cypress**: 1 year
* **Playwright**: 1 year

- Step 1: Investigate the most commonly used UI automation tools.

- Step 2: Compare functionalities, limitations, and use cases for each tool.

- Step 3: Practice using different tools on a project.

5. Clear Understanding of BDD - I’m sorry that I couldn’t share with you my rich experience in the BDD implementation and popularization. During 12 years of my experience, I have implemented BDD (or/and Gerking in testing) in 70% of my projects, took part as a speaker in list of international and regional conferences about BDD in UI Automation.

- Step 1: Review the key concepts, advantages and disadvantages of BDD.

- Step 2: Apply BDD concepts to real-world projects.

6. Improve Communication Skills – thanks for your feedback, but it looks that my skills are enough for all projects (inside team and between the commands). My opinion based on huge list of feedback and list of courses that I’ve passed. But I will try to be best on it, as I see it could be improved based on our session.

- Step 1: Enroll in communication skills workshops or online courses.

- Step 2: Practice active listening and focused responses.

7. Understand Jenkins Pipeline Principles

[**DevOps Foundations: Continuous Delivery/Continuous Integration (2017)**](https://www.linkedin.com/learning/certificates/69a21ee19e38903d7256b599c31a5fbb5fda3417d720ee16c35b3f0d5c28ca50)

**[Continuous Performance Testing with Jenkins](https://learn.epam.com/detailsPage?id=101061c9-d3d4-4797-96a5-85351eb9ff31)**

[**Software Development Life Cycle (SDLC)**](https://www.linkedin.com/learning/certificates/1f015be13238b3791c3c7a8ea6bc8c7e25c70c721a64c080411a1bba2b8b87c3)

- Step 1: Revisit Jenkins documentation and guides.

- Step 2: Create a simple Jenkins declarative pipeline for practice.

### 8. Knowledge of Design Principles – was passed learning course, on projects ware implemented:

### **Open/closed principle.**

### **Dependency inversion principle**

### **Liskov substitution principle**

### **Single responsibility principle**

[Learning S.O.L.I.D. Programming Principles](https://www.linkedin.com/learning/learning-s-o-l-i-d-programming-principles/next-steps?autoSkip=true&resume=false&u=2113185)

- Step 1: Study and familiarize with design principles like SOLID through online resources.

- Step 2: Apply design principles to programming tasks for better understanding.

9. Improve Understanding of Design Patterns - was passed learning course, on previous projects ware implemented: such patterns as Factory, Singelton, Facade and Decorator patterns.

# [Design Patterns in Go for Object Oriented Programming](https://www.linkedin.com/learning/design-patterns-in-go-for-object-oriented-programming/what-are-design-patterns?autoSkip=true&resume=false&u=2113185)

[GRASP (General Responsibility Assignment Software Patterns)](https://habr.com/ru/articles/92570/)

- Step 1: Review design patterns and their applications.

- Step 2: Apply design patterns in programming projects.

10. Understanding The Role of a Lead Engineer - **:** I think that this point is a result of miss understanding during assessment and is not 100% valid. I have finished a list of complex Epam courses for RM and leaders that mange this. For example:

[**Project Leadership**](https://www.linkedin.com/checkpoint/enterprise/login/2113185?pathWildcard=2113185&application=learning&dApp=53239054&redirect=https%3A%2F%2Fwww%2Elinkedin%2Ecom%2Flearning%2Fproject-leadership%3FdApp%3D53239054%26leis%3DLAA%26u%3D2113185)

**[Building Your Team](https://www.linkedin.com/checkpoint/enterprise/login/2113185?pathWildcard=2113185&application=learning&dApp=53239054&redirect=https%3A%2F%2Fwww%2Elinkedin%2Ecom%2Flearning%2Fbuilding-your-team%3FdApp%3D53239054%26leis%3DLAA%26u%3D2113185)**

**[Leadership Foundations: Leadership Styles and Models](https://www.linkedin.com/checkpoint/enterprise/login/2113185?pathWildcard=2113185&application=learning&dApp=53239054&redirect=https%3A%2F%2Fwww%2Elinkedin%2Ecom%2Flearning%2Fleadership-foundations-leadership-styles-and-models%3FdApp%3D53239054%26leis%3DLAA%26u%3D2113185)**

**[RM School 33.14: Performance Coaching and Actionable Feedback](https://learn.epam.com/detailsPage?id=e80239fa-b8c6-49d5-92e3-4901374fa8ba)**

**[TLG31.9: Leading Agile Teams: Team, Culture, Performance](https://learn.epam.com/detailsPage?id=de23d556-b569-4f3a-854b-dd6981071da1)**

- Step 1: Enroll in leadership training.

- Step 2: Understand the distinction between a lead role and the skills of a lead engineer.

11. Conflict Resolution and Motivation Skills**:** I think that this point is a result of miss understanding during assessment and is not 100% valid. I have finished a list of complex Epam courses for RM and leaders that mange this. For example:

**[Automated Testing Global Mentoring Program: Expert Q1Q2 2023 #1](https://learn.epam.com/detailsPage?id=1d0eae60-17a2-4027-8760-0d7134dba8a7)**

**[Conflict Resolution Foundations](https://www.linkedin.com/checkpoint/enterprise/login/2113185?pathWildcard=2113185&application=learning&dApp=53239054&redirect=https%3A%2F%2Fwww%2Elinkedin%2Ecom%2Flearning%2Fconflict-resolution-foundations-4%3FdApp%3D53239054%26leis%3DLAA%26u%3D2113185)**

**[Project Leadership](https://www.linkedin.com/checkpoint/enterprise/login/2113185?pathWildcard=2113185&application=learning&dApp=53239054&redirect=https%3A%2F%2Fwww%2Elinkedin%2Ecom%2Flearning%2Fproject-leadership%3FdApp%3D53239054%26leis%3DLAA%26u%3D2113185)**

And I don’t have any such types of issues on my projects.

- Step 1: Learn techniques for conflict resolution and employee motivation.

12. Understanding Test Automation Risks – **refresh knowledge:** <https://www.youtube.com/watch?v=uc3GvFnPwDA>

<https://www.youtube.com/watch?v=6O0_Un5vylQ>

Sources:

[Automation QA Use Cases](https://kb.epam.com/display/EPMVNHR/Automation+QA+Use+Cases)

[Quality Risks management](https://kb.epam.com/display/EPMQAOPTMO/Quality+Risks+management)

[QA Automation Challenges and Observations](https://kb.epam.com/display/GDOKB/QA+Automation+Challenges+and+Observations)

- Step 1: Research common risks associated with test automation and ways to mitigate them.

13. Experience with Metrics – **done, updated in the project test plan.**

- Step 1: Research test automation metrics and reporting.

- Step 2: Implement metrics in an ongoing project for hands-on experience.

14. Understanding Different Test Types:

**Refresh knowledge at:** [**https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing**](https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing)[**https://www.geeksforgeeks.org/types-software-testing/**](https://www.geeksforgeeks.org/types-software-testing/)

[**https://www.perfecto.io/resources/types-of-testing**](https://www.perfecto.io/resources/types-of-testing) **I have ISTQB certificate.**

- Step 1: Revisit testing fundamentals and learn about different test types.

Was implemented before in project test type:

- Step 2: Apply knowledge of testing types in practical scenarios.