## Abstract
Hewlett Packard Enterprises has developed a system for external REST API documentation generated from XML source that is in sync with the current product source code. This documentation allows HPE’s customers and partners to integrate with the product’s functionality via their REST APIs. However, this system changes from release to release and are in need of clear documentation. The current project compares the API documentation source XML from one release to another and notes the changes by converting them to human-readable HTML. The system is currently written in Java but has become a challenge to maintain, while not meeting HPE’s definition of done.

Our project involved refactoring the Diff API system for optimization. We focused on making sure the code was well-written, maintainable, and well-tested. We worked to implement a series of unit tests to ensure the validity of the code’s execution. The project also added the possibility of being internationalized so steps were taken to make sure the appropriate string and libraries could be interchanged depending on the country.

The project’s impact will improving the existing system that HPE has put into place for Diff API changes. With the group’s work, the code is more manageable, modular, and ready for international use. This decreases the amount of maintenance the company needs to complete when using the system as well as giving them flexibility when updating the Diff API in the future.

---

## Results
The system HPE gave us already had the functionality of outputting the required REST API changes so the majority of the project was spent on refactoring the main class given to us. The group did work on unit testing at first to ensure we had the correct inputs and outputs. This was to make sure the main functionality remained intact while we optimized the code. Lastly, we added string constants that need to be called from a resource bundle for international use. Overall, these changes reduced the amount of code by a large margin. We began with around 3000+ lines and were able to make it manageable at around 600 lines. This also resulted in the 1 main class we were given to become split into 7 smaller classes and 1 class being created for the string constants.

### Deciding Language to Use
- Wrote new system in Java
- Existing code was Java
- Group had prior experience in language

### Splitting Main Class
- Created class sets to reduce and compare
- Compare functions consists of comparing APIs, method classes, query parameters, and request headers
- Reduce functions consists of reducing APIs, possible old APIs, and query parameters
- More modular

### String Constants Class
- Class for internationalization
- Replaced each hard-coded English string to constant
- Strings turned into message bundles for translation
- Load library for appropriate locale and language

---

## Summary
During the semester, our team worked with an HPE representative based in Fort Collins, Colorado. We communicated through emails and Skype sessions to make sure we all understood the tasks assigned for the Agile sprint we scheduled. This allowed us to make sure the project was in the correct stage of completion for the deadline. Our team found it easy to communicate even though we had to do remote meetings and send documents for approval through email. Our UTD mentor was included in these meetings and incorporated in the CC list for emails.

Along with communication over state lines, the group gained practice in using an Agile development model. Dividing the project into sprints allowed for more accurate progression benchmarks as well as flexibility when adding or removing tasks for each stage. The project gave us a better understanding of how the professional world works when developing software. Overall, this was a great experience and look into the computer science industry.

The team would like to thank our faculty advisor, Mehra Boranjany, and our HPE company sponsor, Terry Martin, for their guidance during this project.