On-Premise Large Language Models

Using Local LLMs to aid Java and Rust Software Development, Automated Testing, and Software Reviews

ELECOMP Capstone Design Project 2024-2025

Sponsoring Company:

Rite-Solutions, Inc.
One Corporate Place
Middletown, RI. 02842
http://www.rite-solutions.com

Company Overview:

Rite-Solutions is an award-winning Veteran-Owned Small Business (VOSB) headquartered in Middletown, Rhode Island, established since April 2000. We have a stable and deep corporate history in Rhode Island, with over 380 employees with core competencies including systems engineering, software development, information technology, and cyber engineering. The stability of our business is demonstrated by both our longevity and business backlog. We have achieved both state and national recognition; one of our founders, Joe Marino, is an active member of the RI Science and Technology Advisory Council, along with other trusted Rhode Island business and academic leaders.

Rite-Solutions is known for innovation and dedication to the information and decision support needs of our government and commercial customers, as well as our commitment to providing our customers what we are known for, namely *The Information Advantage®*. We have significant US Navy prime contracts in both warfare systems and business systems development and sustainment, as well as in Information Technology infrastructure support and cyber protection. In research and development efforts, we demonstrate a proclivity for our own inventive solutions but also for finding and working with non-traditional partners, including

academia and small commercial businesses, whose technologies we adapt to national research objectives in practical and meaningful ways that lead to productization.

Our corporate culture is a major part of our drive to both attract and retain people who make a difference. We take pride in our ethos of the F.E.W. (Friends Enjoying Work). Our ownership and management team work to create an open, creative, and stimulating environment. Our founders embraced the servant-leader concept from company inception, and it continues to guide our management team and approach. We recognize great ideas can and do come from anyone, not just the C-suite and thus facilitate ideation across the workforce in an engaging format. This approach to business nurtures the employer/employee relationship and moves our teams to a model where anyone who wants to can be part of the company's growth and contribute to our success. We have won numerous awards. Each year since 2021, we have been awarded the **Providence Business News "best place to work"** among large organizations (> 150 employees) and received extensive media coverage and academic interest about our culture, including as the subject of case studies by Harvard University and Stanford University, in employee motivation and idea generation techniques. In 2022 and 2023, we had a national organization – Great Places to Work - conduct a survey where our employees could anonymously rate Rite-Solutions. The results were astounding – **96% of our employees say Rite-Solutions is a great place to work** as compared to 57% at a typical U.S. based company.

Technical Director(s):

Name **John Sullivan**
Title Software Engineer
Email jsullivan@rite-solutions.com

Name **TBD**

Project Motivation:

To stay competitive in the marketplace, Rite-Solutions is continually improving its software development processes, practices, and tools to increase individual and team productivity while improving product quality, decreasing time to market at lower cost. Software development and test for our primary customers (i.e., DoD, US Navy, commercial customers) require that we adhere to stringent security requirements to protect intellectual property and/or classified information. As a result, solutions commonly found in the commercial marketplace may not be available for us to use since our development environments may require the use of highly protected, on-premises systems and resources.

Of particular interest to Rite-Solutions is the use of open source, Large Language Models (LLMs) that support the automated generation of software design, interfaces, code development, testing, and documentation. Past capstones projects supported by Rite-Solutions at URI and other academic institutions have used LLMs to support a variety of use cases. More specifically, one capstone project previously identified and demonstrated the use of an LLM to generate software locally on a workstation.

Based on above successes using LLMs, we propose using LLMs to address a new use case, namely the generation of Java and Rust software. Rite-Solutions develops and maintains a significant body of Java software used in legacy and on-going projects. Looking forward, Rite-Solutions envisions the use of more modern languages such as Rust to address software safety challenges associated with safety critical systems. As the later section on “Broader Implications for the Best Outcome” will describe, the rapidly expanding use of Rust is having a significant impact on addressing safety critical requirements as well as cybersecurity concerns in a wide variety of commercial and DoD systems and markets.

As noted previously, a key impediment to the automated generation of software and related digital artifacts in our environment is security. Our desired solution will need to host LLMs on-premise where our security policies and controls can protect proprietary and/or other sensitive software and digital assets.

Anticipated Best Outcome:

Project Details:

Rite-Solutions wants to focus efforts of this capstone on generating code in Java and Rust as well as provide additional capabilities and support to enhance our DevSecOps pipeline, namely: (1) the generation of unit tests as well as other types of tests such as functional tests, integration tests, interface tests, performance tests, and security tests; (2) support for peer code reviews; and (3) the generation of documentation such as design documentation, interface/API documentation, and various test plans.

Rite-Solutions will work with the URI team to identify sample cases or problems of differing levels complexities so that LLM(s) will be used to generate Java and Rust code for the same problem. The generated code will be analyzed by analysis tools (e.g., SonarQube for Java, etc.). Metrics will be identified and collected to determine the efficacy of the proposed solution. For example, a common metric is Code Smells. These metrics will allow us to evaluate and compare a Java solution to a comparable solution developed in Rust.

Our primary stakeholders and end users for this system would be software and test engineers. Software engineers would be using the LLM(s) to generate modular software components (e.g., services, etc.) as well as code snippets to support peer code reviews. The generated software would include comments to aid understanding by software developers as well as aid in the automated generation of software documentation. It is expected that the LLM and the generated documentation could be used for software maintenance/sustainability efforts to enhance legacy software with new capabilities, address technical debt as well as support code peer reviews by providing recommended code snippets that address specific issues with new or legacy software. Ideally, the LLM should make recommendations to correct problems identified during peer reviews.

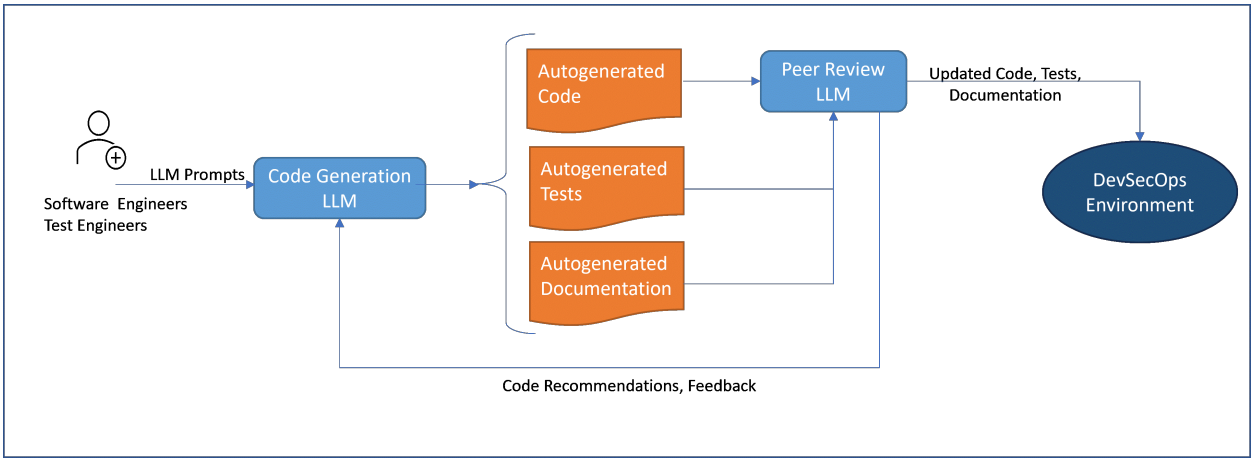
It is expected that the tasks identified in a later section will build upon previous capstone efforts to expedite capstone research and development efforts. As a result, Rite-Solutions will provide

the final report of previous capstone efforts associated with the use of LLMs for code generation to establish a foundation for efforts going forward.

Diagram

To provide a high level perspective, the following workflow diagram shows Software Engineers generating Prompt(s) ("known as Prompt Engineering") requesting the generation of Rust software, tests, and documentation. The generated software along with Prompt(s) from the Software Engineers is used to generate the unit tests and/or other tests as well as documentation.

Another LLM will be used to aid in the peer review of the generated code and tests. It identifies potential problems and makes recommendations to correct the identified issues. This information is passed back to the Code Generation LLM to regenerate code, tests, and documentation. In a production environment, the LLM generated software, unit tests, and documentation would be part of a project's Continuous Integration/Continuous Delivery workflow processes and DevSecOps environment.



Software/Computer Tasks:

The proposed approach should include the following tasks:

- LLM Identification. Identify current, pretrained open source LLMs to support Rust software generation. However, it may be that more than one LLM is required (e.g., one may be very strong in generating software but another is better at generating unit tests).
- Re-evaluation: Determine whether we should continue with last year's code generation LLM or select a new LLM for code generation. New LLMs have continued to emerge this

past year (e.g., Autocoder) although DeepSeek-Coder-V2 (https://the-decoder.com/deepseek-coder-v2-open-source-model-beats-gpt-4-and-claude-opus/#google_vignette) still seems to be highly regarded and competitive with proprietary models. An abridged evaluation will be performed using a Pugh matrix methodology and then build upon the evaluation criteria previously generated by a previous capstone.

- Evaluate the identified LLMs using a Pugh matrix; the evaluation criteria and weights should be jointly developed with the Rite-Solutions mentors.
- Document and present to Rite-Solutions the recommended LLM(s) to be used to support Java and Rust code generation and test generation, software documentation generation, and support peer reviews.
- Install and configure the recommended LLM(s); it would be preferred if the LLM(s) could be installed/delivered in a Docker container. Generate user documentation needed to install/configure the LLMs. It is expected that delivery of results, documentations (e.g., installation/configuration documentation, etc.) and software will be provided at the end of each semester.
- Develop a test approach to evaluate the LLM(s). Since LLMs are known to generate code that does not work, recommend an approach that will aid in correcting problems with generated code. One approach to consider is the use of micro agents. (https://github.com/BuilderIO/micro-agent)
- Identify and document detailed tests to be performed. Rite-Solutions will work with URI to identify sample functions needed (e.g., generate software to translate a data item from one format to another, etc.) to support test and evaluation.
- Identify and document metrics that need to be collected to evaluate the LLM(s).
- Perform in-depth tests of the LLM(s) and collect metrics. Evaluate the generated Rust software, unit tests, comments, etc., to determine if they work (i.e., generate the correct software). If the generated code/unit tests do not work, correct the generated software and track the lines of code that were needed to be added, deleted, or correct. Collect and document test results.
- Generate preliminary report that describes what was performed, lessons learned, challenges, recommendations for Rite-Solutions review; incorporate review comments, generate final report.

Composition of Team:

1 Computer Software Engineer; 1 Data Science Engineer

Skills Required:

Software/Data Engineering Skills Required:

- Software design, development, integration, and test
- Ability to understand, evaluate, and test LLMs.
- Ability to identify and install open-source software solutions.
- Ability to generate and/or update scripts to support activities (e.g., build, installation, test, etc.)
- Ability to generate user manual, installation documentation, design documentation.
- Experience with Linux; understanding containers would be desirable.
- Experience with Rust and/or object-oriented computer languages would be desirable

Anticipated Best Outcome's Impact on Company's Business, and Economic Impact

Rite-Solutions' is an early adopter for new technologies that increase productivity, enhance our DevSecOps pipelines, and improve the quality of the products and services we provide to our commercial and DoD customers in less time and at lower cost. Based on our internal research and development investments in conjunction with work performed by capstone projects at URI and other academic institutions, we believe the approached technologies identified in this capstone promises to address these needs.

Our anticipated best output would be a solution that could be run on a laptop that utilizes open source LLMs to:

1. Generate both Java and Rust software for the proposed problems.
2. Generate tests - unit tests (as minimum but additional tests would be desirable such as functional test, interface tests, integration tests, system tests performance tests)
3. Generate documentation – software design documentation, interface documentation, test documentation.
4. Support peer reviews – identify potential coding problems as well as recommend potential solutions.

Broader Implications of the Best Outcome on the Company's Industry:

Our focus on the use of open source LLMs to generate java and Rust software would be a significant step toward improving the software safety and cybersecurity for multiple domains and systems, such as Government/DoD systems, embedded systems, medical devices, Internet of Things (IoT) environments, financial and banking systems, system applications (e.g., critical components of operating systems), communications systems, and mobile devices. This approach provides the opportunity to accelerate development, test, and integration of Java in legacy and ongoing/new efforts. Looking forward, Rust provides the opportunity to use a more modern language to better address software safety challenges. Recent government studies have identified Rust as language that addresses memory safety problems and recommends its use in new development as well as in legacy systems.

Although Rust is highly regarded as a significant solution to software safety issues, one challenge is training software developers to use Rust. The community of Rust developers is large (estimated to 2.8 million). However, the demand for Rust developers has continued to grow yearly. Our proposed approach of using LLMs as a knowledgeable assistant will aid the generation of Rust software and automated tests as well as assist in peer reviews to enhance quality of the software under development. Of equal importance, this approach should reduce the DevSecOps pipeline time thereby aiding the deployment of memory safety applications in safety critical and cybersecurity applications and systems in less time.