Old sheet (for reference) : <https://docs.google.com/document/d/1m8UQMkwmdDO8yvxt4lIF639KfVA2fMm0pLOJhGnpl64/edit>

# Template:

Please use the following template when proposing the projects:

1. Project Title:
   * Mentor Name:
   * Project Duration: (3 months, 6 months, 9 months, 1 year)
   * Number of people:
   * Project Description and Scope:
   * Possible Technologies involved:

# Projects:

## **Project 1 : gRPC connector for Siddhi**

**Mentor Name:** Mohan (mohan@wso2.com)

**Duration :** 6 Months

**Number of people** : 1 Person

**Description**

gRPC [1] is a modern, open source remote procedure call (RPC) framework that can run anywhere. It enables client and server applications to communicate transparently, and makes it easier to build connected systems. Siddhi [2] is the complex event processing library we use within WSO2 SP[3]. Siddhi is a SQL like language where you can write queries which can do real-time analysis. Siddhi engine contains numerous types of connectors which allow to connect with different event sources and sinks to consume and publish events. In this project, you are expected to build a Siddhi IO connector to both consume and publish events to source and sink.

**Deliverables**

IO connector (Source and Sink) for Siddhi

**Skills Needed**

gRPC, Siddhi, Java

[1] <https://grpc.io/>

[2] <https://wso2.github.io/siddhi/>

[3] <https://docs.wso2.com/display/SP430>

## **Project 2 : VSCode Plugin for Siddhi**

**Mentor Name** : Mohan (mohan@wso2.com)

**Duration** : 6 Months

**Number of people** : 1 Person

**Description**

Build a Visual Studio Code Plugin[1] for Siddhi[2] with syntax highlighting, code completion, error reporting. Siddhi is the complex event processing library we use within WSO2 SP[3]. Siddhi is a SQL like language where you can write queries which can do real-time analysis. You can get yourself familiar with Siddhi language through Siddhi samples[4] and test-cases[5]. Siddhi is based on antlr[6] and grammar file can be found here[7].

**Deliverables**

Visual Studio Code Plugin with syntax highlighting, code completion and error reporting

**Skills Needed**

Visual Studio Code Plugin development

[1] <https://code.visualstudio.com/api>

[2] <https://github.com/wso2/siddhi>

[3] <https://wso2.com/analytics-and-stream-processing/>

[4] <https://github.com/wso2/siddhi/tree/master/modules/siddhi-samples/quick-start-samples>

[5] <https://github.com/wso2/siddhi/tree/master/modules/siddhi-core/src/main/java/org/wso2/siddhi/core/query>

[6] <http://www.antlr.org/>

[7] <https://github.com/wso2/siddhi/blob/master/modules/siddhi-query-compiler/src/main/antlr4/org/wso2/siddhi/query/compiler/SiddhiQL.g4>

## **Project 3 : Siddhi CEP-*lite* Implementation (for better performance)**

**Mentor Name** : Suho (suho@wso2.com)

**Duration** : 6 Months

**Number of people** : 1 Person

**Description**

Siddhi [1] is a complex event processing library which supports processing millions of events per second. It is used as the core engine in WSO2 Stream Processor product [2] as well. Siddhi CEP engine is implemented using Java programming language and it can be used as the library with any Java-based implementation and solution. Over the recent past, we have seen that there is a huge hype in cloud-native solutions/applications and most/all of them are implemented using GO programming language [3]. Even Though, there is a huge chance for Siddhi CEP integrations on these cloud-native solutions we cannot achieve that because SIddhi implementation is on Java. Then we have decided to do a minimal implementation of Siddhi in the GO language.

**Deliverables**

Siddhi CEP (*lite*) in Go

**Skills Needed**

Siddhi, Go Lang, Java

[1] <https://github.com/wso2/siddhi>

[2] <https://wso2.com/analytics-and-stream-processing/>

[3] <https://golang.org/>

## Project 4: Kubernetes agent for Testgrid deployment clusters

**Mentor Name:** Kasun Gajasinghe (kasung@wso2.com)

**Duration :** 6 months

**Number of people** : 1 Person

**Description:**

In essence we need to develop the counterpart of tinkerer agent in AWS cloud for Kubernetes. As a product cluster deployment creator and tester, Testgrid need to be able to work within the deployment in order to capture metrics data. These metrics data involve download and archival of logs, cpu/memory consumptions etc.

After the successful completion of this project, we’ll bring the Grafana/Telegraf-based metrics dashboard into production for visualizing cpu and memory data of the production deployment.

**Deliverables:**

Kubernetes Tinkerer agent Java implementation

Integration of tinkerer agent into existing deployment scripts

A running proof-of-concept in production

**Technologies Involved:**

Core Java, Websocket, Kubernetes on Google Cloud, AWS, CI/CD

## Project 5: Bring TG deployment to local host

**Mentor Name:** Kasun Gajasinghe (kasung@wso2.com)

**Duration :** 3 months

**Number of people** : 1 Person

**Description:**

Currently, Testgrid deployment only lives in the cloud. There’s no way to run it in local machine. Since developer productivity is greater when they have the code in the local machine, we need to package testgrid better and then make it executable on local Linux instances.

For this, you’ll come up with a distribution structure that packages all components of testgrid - TG-java, Jenkins, and dashboard. We’ll use an in-memory db like H2 as the default database. Of course team is there to guide you.

**Deliverables:**

A locally runnable Testgrid distribution

**Technologies:**

Java, Groovy scripting, Jenkins, Maven

Linux, bit of ReactJS

## Project 6: Research and development of resiliency of long-running product deployments

**Mentor Name:**

**Duration :** 2 months

**Number of people** : 1 Person

**Description:**

Running stable production deployments over a long period of time is a significant feat. to achieve. We need to make sure there’s no memory-leaks, cpu spinning issues, high disk consumption etc.

While current WSO2 products are ready for that, we need to continuously verify them for each new feature. It is complex compared to a short-running tests that we do. We need to research and decide how to develop and maintain such a production deployment. We can initiate this with apim cluster long-running deployments.

Resiliency tools like Netflix’s [chaosmonkey](https://github.com/Netflix/chaosmonkey) is one such directive that provide fault-tolerance.

**Deliverables:**

A tool that makes verifying resiliency of long-running deployments possible.

A white paper on the research

**Technologies involved:**

## Project 7: Simplify cluster deployment complexity of the deployment synchronizer

**Mentor Name:** Ching ([ching@wso2.com](mailto:ching@wso2.com)), Kasun Gajasinghe(kasung@wso2.com)

**Duration:** 4-6 months

**Number of people** : 1 Person

**Description:**

The WSO2 product cluster deployment scripts has become overly complex due to the deployment synchronizer architecture.

We may go for two poc implementations, and then choose what matches best:

1. Amazon S3 (Simple Storage Service) based artifact synchronization - Since S3 is a service managed by AWS, the deployment overhead is minimal. This is however only for deployments on AWS cloud.   
   This seems like a better approach since EFS, NFS based artifact syncing has shown that there’s a lot of limitations in there.

S3 provides read-after-write consistency but with one caveat. For artifact updates, it only provide eventual consistency. More details: <https://codeburst.io/quick-explanation-of-the-s3-consistency-model-6c9f325e3f82>

1. RDBMS-based artifact synchronizer - Given that a db is already a part of the deployment, we can leverage that as the central location to keep artifacts. This is a generic enough solution.   
   With RDBMS, we may only use it for coordination. The data may still lie in file-system or some other medium optimized for storage.

We need a medium where Consistency and Availability aspects of the CAP theorem is fulfilled.

Providing consistency is an important aspect of this effort. Each node in the cluster should have the same artifacts (md5sum should match). We need better cluster coordination to make this possible. We could use the database as the coordination point for this purpose.

**Deliverables:**

One production recommended deployment synchronizer component

One PoC code for deployment synchronizer component

**Technologies involved:**

Java, SQL, AWS cloud

<https://docs.wso2.com/display/CLUSTER44x/Introduction+to+Deployment+Synchronizer>

## Project 9: Provide resiliency for Kubernetes builds in Testgrid

**Mentor Name:** Kasun Gajasinghe (kasung@wso2.com)

**Duration :** 3 months

**Number of people** : 1 Person

**Description:**

We need to implement resiliency for kubernetes builds we do in Testgrid.

**Deliverables:**

**Technologies used:**

Java, Groovy scripting, Kubernetes

## Project 9: Self-servicing license manager implementation

**Mentor Name:** Maheshika (maheshika@wso2.com)

**Duration :** 3 months

**Number of people** : 1 Person

**Description:**

Current tool allows only a single designated user to generate the license. The current requirement is to reuse License Manager tool and implement a hosted application where team members can generate the license before a release.

**Deliverables:**

A tool with a UI which enable teams generate Licenses files when a product distribution is provided as an input.

**Skills Needed:**

Java, Maven and MySQL

## **Project 10 : gRPC support for microgateway**

**Mentor Name:** Rajith (rajithr@wso2.com)

**Duration :** 6 Months

**Number of people** : 1 Person

**Description**

gRPC[1] is a high-performance, lightweight communication framework designed for making traditional RPC calls, and developed by Google. The framework uses HTTP/2, the latest network transport protocol, primarily designed for low latency and multiplexing requests over a single TCP connection using streams. This makes gRPC amazingly fast and flexible compared to REST over HTTP/1.1.

Existing micro-gateway does not provide support for gRPC protocol.

In this project we need to support client to gateway and gateway to backend communication in GRPC and do the required transformations at the gateway. More details can be found in here[2]

**Deliverables**

First class grpc support at gateway

**Skills Needed**

gRPC, ballerina, Java

[1] <https://grpc.io/>

[2] <https://github.com/wso2/product-microgateway/issues/283>

## Project 11: Microgateway integration with Prometheus and Grafana (Observability for microgateway)

## **Mentor Name:** Rajith (rajithr@wso2.com)

**Duration : 6 months**

**Number of people** : 1 Person

**Description:**

Observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs. Monitoring, logging, and distributed tracing are key methods that reveal the internal state of the system to provide the observability. Microgateway should be fully observable by exposing itself via these three methods to various external systems allowing to monitor metrics such as request count and response time statistics, analyze logs, and perform distributed tracing. This is already available in ballerina level[1], we need to extend this for microgateway

**Deliverables:**

Graphana dashboards to track every request in microgateway,(response times, backend delay, authentication, delay, memeory, cpu etc)

**Skills Needed:**

Ballerina, promethieus

**[1] -** [**https://ballerina.io/learn/how-to-observe-ballerina-code/**](https://ballerina.io/learn/how-to-observe-ballerina-code/)

## Project 12: API Design Time Governance (Integrating API Publisher with apisecurity.io)

## **Mentor Name:** NuwanD (nuwand@wso2.com)

**Duration :** 3 months

**Number of people** : 1 Person

**Description:**

apisecurity.io provides a service which allows to validate an Open API Specification against a known set of rules (best practices). We need to integrate the API publisher with the services offered by apisecurity.io so that we can provide guidelines (recommendations) and enforce governance rules on the design of the API.

**Deliverables:**

A feature in API Manager which integrates apisecurity.io with the API publisher portal.

**Skills Needed:** Java, ReactJS, HTTP, OAS (Swagger), REST

## Project 13: Integrating the API gateway/microgateway with AWS Lambda

## **Mentor Name:** NuwanD (nuwand@wso2.com)

**Duration :** 4 months

**Number of people** : 1 Person

**Description:**

AWS Lambda provides a mechanism to execute functions as services on the AWS SaaS platform. Lambdas can only be invoked using the officially provided SDKs or through the AWS api gateway. We need to build a mechanism where the WSO2 API gateway and microgateway can expose a Lambda function as an API seamlessly when the gateway/microgateway is running on AWS.

**Deliverables:**

A feature on the API Manager (including the microgateway) to be able to expose a Lambda function as an API.

**Skills Needed:** Java, Ballerina, ReactJS, HTTP, AWS, Lambda, OAuth2.0

## Project 14: Building a standard for federated API marketplaces and gateways.

## **Mentor Name:** NuwanD (nuwand@wso2.com)

**Duration :** 4 months

**Number of people** : 1 Person

**Description:**

With the API space becoming immensely crowded, there are multiple API providers and API gateways in the market. There is however no common API standard that can be used as a general specification that can be accepted by any API provider for creating APIs. The OAS specification itself is quite limited and has lots of missing pieces in terms of defining an API on an API marketplace. This project is to come up with a standard, most probably as an extension to the Open API specification which can be adopted by any API marketplace solution.

**Deliverables:**

The initial version of the specification which can be released.

**Skills Needed:** OAS (Swagger), knowledge on API Gateway, knowledge on API marketplaces

## Project 15: Implementing Kafka based Distributed Coordination algorithm

* + Mentor Name: Srinath
  + Project Duration: (6 months)
  + Number of people: 1
  + Project Description and Scope: Distributed coordination is a hard problem and solved using algorithms such as [Paxos](https://medium.com/coinmonks/paxos-made-simple-3b83c05aac37) and [Raft](https://raft.github.io/) (e.g. Apache Zookeeper). The goal of the project is to explore a new algorithm based on Kafka. The project involves implementing the algorithm and comparing its performance to Paxos and Raft ( you do NOT have to implement Paxos or raft). The project can potentially lead to a peer-reviewed research paper.
  + Possible Technologies involved: Java, Zookeepr. <https://raft.github.io/>, <https://github.com/jaksa76/paxos>

## Project 16: Exploring effort of Server Architectures on Performance under different workloads

* + Mentor Name: Srinath
  + Project Duration: (6 months)
  + Number of people: 1
  + Project Description and Scope: When we build a server we choose IO model ( blocking, NIO, NIO2), concurrency (thread pool, actors, disrupter). We can combine these to build different server architectures. Furthermore, there is well-known architecture such as [SEDA](http://www.sosp.org/2001/papers/welsh.pdf) that combines blocking and non-blocking aspects. The goal of the project is to implement different workloads (e.g. CPU bound, IO-bound, memory bound) using each architecture and study performance behaviors of each architecture to identify when each architecture does well. It will give you a deep understanding of server architectures and performance. The project can lead to a peer-reviewed research paper.
  + Possible Technologies involved: Java, NIO, Also read <http://berb.github.io/diploma-thesis/original/042_serverarch.html>

## Project 17: Writing fuzzing based tester for Ballerina compiler

Fuzz based testing in gaining popularity as a way to find bugs automatically. Eg: google recently open sources one such tester <https://github.com/google/oss-fuzz>. This is especially important in compiler testing because it's impossible to test all possible programs by hand.

In this project we will generate possible source code and then run the compiler to see if it produces expected behaviour.

In first phase we will do black box fuzzing, eg see if compiler throws exceptions. After that we can research on how we can use greybox fuzzing.

Fuzzing is not widely used in insdety yet, so some research is needed to complete this task.

* + Mentor Name: Manuranga (manu@wso2.com)
  + Project Duration: (6 months)
  + Number of people: 1

Deliverables :

* Source code generator to create test inputs
* Test runner (or afl tool configuration)
* List of discovered bugs

Project 18: API Key implementation for API Manager

* + Mentor Name: HarshaK
  + Project Duration: (6 months)
  + Number of people: 1

Currently, API manager uses oauth2,basic and mutual ssl to authenticate and authorize API requests. This assures security and is good for dealing with apis that handle sensitive data. However APIs with less critical functionalities and can be exposed through API key authentication. Unlike access tokens used in oauth2, API keys do not have an expiry time or a scope associated with them. So basically an API key grants unrestricted asses (in time or scope) to the API.

Project 19: VSCode Plugin for Cellery

* + Mentor Name: Sinthuja
  + Project Duration: (6 months)
  + Number of people: 1

Cellery is a new project by WSO2 which focuses on build run and manage code-first composites on Kubernetes. And currently, the cells are supported in ballerina language. And with this project, we expect to create VSCode plugin that helps the developers to develop cellery project with following features.

* Initialize the cell file with default cellery skeleton.
* Support cellery and ballerina syntax (possibly extend ballerina plugin)
* Cell and component dependency diagram.

Project 20: Cellery Observability as Istio Mixer Plugin

* + Mentor Name: Sinthuja
  + Project Duration: (3 months)
  + Number of people: 1

Currently, cellery telemetry receiver implements the Mixer’s reporter interface. And the telemetry of the components are directly sent to cellery telemetry receiver (WSO2 SP). We need to make this as istio-mixer plugin, therefore the data will be received by all mixer plugins added to the environment in addition to SP plugin that is required for cellery observability.

Project 21: Using Nanopass approach to introduce modular phases to ballerina compiler

* + Mentor Name: Manuranga (manu@wso2.com)
  + Project Duration: (6 months)
  + Number of people: 1

Please reffer to <https://www.cs.indiana.edu/~dyb/pubs/nano-jfp.pdf> and <https://www.cs.indiana.edu/~dyb/pubs/commercial-nanopass.pdf>

Idea is to develop a similar approach that can produce type verified compiler phases to Ballerina compiler.

Source -> {paser} -> {type chcker} -> ... -> {bir-gen} -> **your phase goes here** -> {jvm code gen} -> .class files

In this project we will come up with optimizations to compiler.

## Project 22 : Private Jet Mode for WSO2 API Manager with Kubernetes

## **Mentor Name:** Pubudu (pubudug@wso2.com)

**Duration :** 6 Months

**Number of people** : 1 Person

**Description**

Currently in the WSO2 API Manager when you deploy an API, it creates the Synapse artifact and it gets deployed in the Synapse Gateway. This is the basic approach. With the private jet mode, when you are publishing the API, you can select the Private Jet Mode and then publish the API. When this happens, this will create all the necessary artifacts which required to deploy the API Microgateway in Kubernetes. From the API manager you can check and manage all the gateways that are getting deployed.

**Deliverables**

* API Manager/K8s integration related artifacts and documentation
* Webinar
* Screencast

**Skills Needed**

Java, Kubernetes, React

[1] [https://kubernetes.io](https://kubernetes.io/)

## Project 23 : WSO2 API Manager integration with Prometheus

## **Mentor Name:** Pubudu (pubudug@wso2.com)

**Duration :** 6 Months

**Number of people** : 1 Person

**Description**

Currently in the WSO2 API Manager, the default analytic engine is WSO2 Stream Processor. API related analytics, alerts and other information are collected at the WSO2 SP. The objective of this project is to do an integration with Prometheus where you have the same capabilities that are in WSO2 SP. In addition to those existing features, we can add the Prometheus features as well. The Prometheus could run in a physical machine or in K8s.

**Deliverables**

* API Manager/Prometheus integration related artifacts and documentation
* Webinar
* Screencast

**Skills Needed**

Java, Prometheus

[1] [https://prometheus.io](https://prometheus.io/)

## **Project 24: Improve Resiliency aspect of EI production servers (product failure detection and auto recovery)**

**Mentor Name:** Isuru Udana (isuruu@wso2.com)

**Project Duration:** 3 months

**Number of people:** 1

**Project Description and Scope:**

When an issue occurs in a production server, to investigate the incident, we need to fetch several information from the server instance. At the moment we need to extract these information manually and sometimes we may not get required information before the server goes to a catastrophic situation. This project is about automatically extracting useful information from the server, when there are potential issues.

Following are the high level requirements

- Alerts on increase of resources

- Automatically generate required information for debugging (dumps)

- Handle/prevent catastrophic situations

- Correlate dumps with environment changes like request load and generate reports that can be used to analyse a problem

We have done a GSOC project on improving debugging capability of production ESB servers. In this project we are extending the functionality introduced in that project to improve the stability of the servers. This includes features like auto recovery from potential incidents. Improving server monitoring capabilities, etc.

**Possible Technologies involved:**

Java, JMX

## Project 25: gRPCsupport for Enterprise Integrator

**Mentor Name:** Kasun Indrasiri (kasun@wso2.com)

**Project Duration:** 3 months

**Number of people:** 1

**Project Description and Scope:**

gRPC is an inter-process communication protocol which is implemented on top of HTTP2. gRPC can use protocol buffers as both its Interface Definition Language (IDL) and as its underlying message interchange format.

This projects includes adding gRPC support for WSO2 EI to expose gRPC services as well as invoke gRPC services.

#### 

#### Deliverables

* Implementation of gRPC transport/inbound endpoint
* Documentation
* Unit/Integration tests

**Possible Technologies involved:**

gRPC, Protocol Buffers

WSO2 EI Fundamentals

#### 

[1] <https://grpc.io/>

[[2]](https://grpc.io/) <https://docs.wso2.com/display/EI640/WSO2+Enterprise+Integrator+Documentation>

[[3]](https://docs.wso2.com/display/EI640/WSO2+Enterprise+Integrator+Documentation) <https://docs.wso2.com/display/EI640/Custom+Inbound+Endpoint>

[[4]](https://docs.wso2.com/display/EI640/Custom+Inbound+Endpoint) <https://docs.wso2.com/display/EI640/ESB+Transports>

## Project 26 : Run WSO2 API Microgateway in Serverless mode

**Mentor Name:** Pubudu (pubudug@wso2.com)

**Duration :** 3 Months

**Number of people :** 1 Person

**Description**

Knative is Kubernetes-based platform to build, deploy, and manage modern serverless workloads. The objective of this project is to come up with relevant artifacts which required to deploy WSO2 API Microgateway in Serverless mode in KNative. With the completion of the project, white paper, webinar and a screencast can be done.

**Deliverables**

* Artifacts for deploying in Serverless mode
* White paper / Blog Post
* Webinar
* Screencast

**Skills Needed**

Java, Kubernetes, KNative

[1] [https://kubernetes.io](https://kubernetes.io/)

[2] <https://cloud.google.com/knative/>

## Project 27 : API Manager CRD for Kubernetes

**Mentor Name:** Pubudu (pubudug@wso2.com)

**Duration :** 6 Months

**Number of people :** 1 Person

**Description**

We have already created few custom resource definitions for API Manager and those would provide the functionality around API Management. The Objective of this project is to write a custom resource definition for API Manager and once you deploy the CRD, it would deploy the API Manager in Kubernetes. Based on the configurations provide within the CRD definition, the deployment changes.

**Deliverables**

* CRDs, controller and other related resources
* Webinar
* Screencast

**Skills Needed**

Go, Kubernetes

[1] [https://kubernetes.io](https://kubernetes.io/)

[2] <https://github.com/wso2/k8s-apim-operator>

[3] [https://golang.org](https://golang.org/)

## Project 28 : Update ReadyAPI! plugin for API Manager

**Mentor Name:** NuwanD (nuwand@wso2.com)

**Duration :** 3 Months

**Number of people :** 1 Person

**Description**

ReadyAPI! is an API testing tool by SmartBear. WSO2 has developed a plugin for ReadyAPI! so that it becomes possible to use ReadyAPI! easily to APIs deployed on the WSO2 API Manager. This project is to update the ReadyAPI! plugin for the WSO2 API Manager to be compatible with the latest version of WSO2 API Manager and the latest version of ReadyAPI! itself.

**Deliverables**

* New version of the ReadyAPI! plugin for WSO2 API Manager.

**Skills Needed**

Java, use of SoapUI/ReadyAPI!.

## Project 29 : OWASP Dependency Track Improvements

**Mentor Name**: Dulanja (dulanja@wso2.com)

**Project Duration**: 6 months

**Number of people**: 1

**Project Description and Scope**:

OWASP Dependency Track [1] will be utilized by WSO2 to monitor the 3rd party dependencies that are packed within the products, and identify the vulnerabilities of them by querying repositories such as the National Vulnerability Database (NVD). Currently there are multiple requirement gaps in the Dependency Track project that need to be fulfilled in order for us to effectively use it within WSO2 security processes. They are:

* Report generation - vulnerabilities by product, and etc.
* Migration of mitigation comments from existing product versions to the newer one.
* Adding custom maven repositories.
* Integration with JIRA to create tickets.

These feature gaps needs to be implemented in a generic manner (as much as possible) and ultimately contribute them to the Dependency Track project.

[1] <https://www.owasp.org/index.php/OWASP_Dependency_Track_Project>

**Possible Technologies involved**:

* Java, JavaScript

## Project 30 : Vulnerability Management System Improvements

**Mentor Name**: Ayoma (ayoma@wso2.com)

**Project Duration**: 6 months

**Number of people**: 1

**Project Description and Scope**:

OWASP DefectDojo is used as the centralized security vulnerability management system (VMS) in WSO2. This system is responsible for maintaining resolution and mitigation comments associated with security vulnerabilities getting reported by different security scanners (Veracode, Qualys WAS, ZAP, BurpSuite). In order to fully adopt OWASP DefectDojo to WSO2 security processes and provide difect access to the system to WSO2 engineers, it is required to make some custom alterations to the open source version. It is also required to make these alterations as generic as possible and contribute those as improvements to the open source project.

Following are the identified alterations and tasks that needs to be completed:

* Using an already existing SAML extension (ex: django-saml2-auth) to SSO into DefectDojo. Thereby allowing engineers to SSO into the system based on the WSO2 login.
* Improve DefectDojo to support adding multiple notes with "note-types", so that we can store certain information about issues separately (ex: use case, impact, resolution comments). Currently, this information is stored concatenated into a single note, separated by a delimiter. This change should be contributed to the original open source project
* Ability to store and map all the "resolution types" WSO2 use (ex: already mitigated, not a threat, not applicable), as a "note-type".
* Since using a spreadsheet can be more efficient in analyzing large number of items, WSO2 currently has the CSV import and export ability added as a customization. Instead of the current CSV, it is required to introduce exporting and importing from a shared Google Drive location, using Google APIs.
* Currently CSV report generator is doing following customized tasks which need to be moved into report-parser (to avoid having to do an export to trigger these tasks):
  + Delicate issue identification
  + Separately tagging utility functions
* Prepare a migration script for the current VMS data set and update the VMS to the latest DefectDojo version.

**Possible Technologies involved**:

* Python, Django Framework
* Google Drive API, Google Sheet API
* SAML SSO
* JavaScript, HTML

## Project 31 : OIDC Federated Logout for WSO2 Identity Server

**Mentor Name**: Darshana (darshana@wso2.com)

**Project Duration**: 4 months

**Number of people**: 2

#### **Description**

OpenID Connect (OIDC) [1] is a standard authentication protocol which is widely used by modern applications. Its based on the OAuth 2.0 and it defines three logout mechanisms [2][3][4].

WSO2 Identity Server (WSO2 IS) [5] is an open source Identity and Access Management product. WSO2 IS supports identity federation [6] which allows client applications to login through the external (secondary) identity server. As per WSO2 IS architecture [7], communication with the external identity provider handled by the outbound authentication connectors and WSO2 IS has the support of several outbound authentication connectors including SAML 2.0, OIDC, PassiveSTS as well as many social connectors.

With this project, it’s expected to improve the OIDC connector [8] to introduce logout capability by implementing one of standard logout mechanisms defined in OIDC specifications. Implementation will require candidate to get familiar with selected logout specification and implement it to existing WSO2 IS code base. If time permits, other logout mechanisms too can be investigated.

[1] https://openid.net/specs/openid-connect-core-1\_0.html

[2] https://openid.net/specs/openid-connect-session-1\_0.html

[3] https://openid.net/specs/openid-connect-backchannel-1\_0.html

[4] https://openid.net/specs/openid-connect-frontchannel-1\_0.html

[5] https://wso2.com/identity-and-access-management/

[6] https://docs.wso2.com/display/IS570/Identity+Federation

[7] https://docs.wso2.com/display/IS570/Architecture

[8] https://github.com/wso2-extensions/identity-outbound-auth-oidc

#### **Deliverables**

* Implementation of OIDC federation logout
* Automation tests
* Documentation

#### **Skills Needed**

* Skill to read, understand and translate specifications it to working code
* Java, Maven, Git

## **Project 32 : Artificial Intelligence Driven Anomaly Detection for APIs**

**Mentor Name:** Sanjeewa Malalgoda (sanjeewa@wso2.com)

**Duration :** 6 Months

**Number of people :** 1 Person

**Description**

Each API has its own access patterns and users, which makes it hard to detect a specific pattern by analyzing large volumes of data manually or by using static policies. Unsupervised machine learning and artificial intelligence (AI) can be used to augment API security and efficiently detect security threats. With this project we will research and implement AI based threat detection solution for APIs. Implementation may have 3 parts 01. Security enforcement point 02.Decision point 03.Smart Engine.

**Deliverables**

* Security enforcement point, Decision point, Smart Engine.

**Skills Needed**

Java, Tensorflow, Markup models

## **Project 33: Fine Grained Access control for GraphQL APIs**

## **Mentor Name:** Sanjeewa Malalgoda (sanjeewa@wso2.com)

**Project Duration:** 6 months

**Number of people:** 1

**Project Description and Scope:**

GraphQL’s [2] power comes from a simple idea — instead of defining the structure of responses on the server, the flexibility is given to the client. Each request specifies what fields and relationships it wants to get back, and GraphQL will construct a response tailored for this particular request. The benefit: only one round-trip is needed to fetch all the complex data that might otherwise span multiple REST endpoints, and at the same time only return the data that are actually needed and nothing more. As of today GraphQL is just exposing web services over standard Get and Post. But what is missing today is fine grained access control for GraphQL based APIs. Most of the integration vendors or API Management vendors have not address this problem. For an example GraphQL query complexity, query execution time, concurrent access etc need to control and govern when we expose them as APIs to outside. This query analysis will have 2 parts. 01. Static query analysis in gateway level 02. Dynamic query analysis.

Also another challenge is we cannot apply standard rate limiting policies, authentication authorization etc to GraphQL APIs because we do not have a way to go to query or mutation level with current API support.

#### **Deliverables**

* Research about fine grained resource access control.
* Static and dynamic query validator.
* Documentation
* Unit/Integration tests

**Possible Technologies involved:**

GraphQL, fine grained access control for resources, policy definition languages.

[1] <https://sensedia.com/en/blog/apis-en/graphql-bff-pattern/>

## Project 34: Machine Learning based Auto-tuning Framework for WSO2 EI

## **Mentor Name:** malithj (malithj@wso2.com)

**Duration :** 6 months

**Number of people** : 1 Person

**Description:** Design of self-adaptive systems is a main research at WSO2. WSO2 EI has many configuration parameters and performance of these parameters will have a significant impact on the performance. The optimal parameter values to achieve the best performance will depend on the traffic conditions and the use case. In this project you will use machine learning techniques to tune configuration parameters WSO2 EI **online** (in order to achieve better performance). In particular you will use Bayesian and Reinforcement learning to compute the optimal configuration parameters.

**Deliverables:**

Performance auto-tuning framework for WSO2 EI

**Skills Needed:** programming skills (Java and Python preferred) , Have an interest in doing systems research (concurrency models, server architectures, etc.), have some exposure to machine learning

**[1]** [**https://bit.ly/2x03dHh**](https://bit.ly/2x03dHh)

**[2]** [**https://drive.google.com/file/d/18S\_DyGuWfULriYlqfXKkz7ycqx6RQn32/view**](https://drive.google.com/file/d/18S_DyGuWfULriYlqfXKkz7ycqx6RQn32/view)

**[3]** [**https://docs.wso2.com/display/EI650**](https://docs.wso2.com/display/EI650)

## Project 35: Dynamic Auto-scaling Policies

## **Mentor Name:** malithj (malithj@wso2.com)

**Duration :** 6 months

**Number of people** : 1 Person

**Description:** In Cloud computing, auto scaling ensures performance guarantees to customers while making efficient use of resources and keeping operational costs low. Although this is a valuable feature devising an auto-scaling policy which provides performance guarantees is extremely challenging task. In this project you will first evaluate different auto-scaling policies under different workload patterns and then develop a set of novel auto-scaling policies that can address issues in existing auto-scaling policies.

**Deliverables:**

Evaluation of auto-scaling policies

Develop a set of novel auto-scaling policies

**Skills Needed:** programming skills (Java preferred) , Have an interest in doing systems research (scalability, performance modelling, building systems, workload characterization), have some exposure to machine learning or willingness to learn

**[1]** [**http://www.buyya.com/papers/AutoScaleWebAppClouds-ACMCS.pdf**](http://www.buyya.com/papers/AutoScaleWebAppClouds-ACMCS.pdf)

**[2]** [**https://prismmodelchecker.org/papers/ccgrid17.pdf**](https://prismmodelchecker.org/papers/ccgrid17.pdf)

## Project 36: Distributed cache revocation at the API Gateway

## **Mentor Name:** NuwanD (nuwand@wso2.com)

**Duration :** 4 months

**Number of people** : 1 Person

**Description:** When a token (key) is cached in a cluster of WSO2 API Gateways it is required for a distributed computing solution to remove a particular cache entry from each node in the cluster. The current solution used in the gateway is based on something called “axis2 clustering” which is a multicast based communication mechanism that informs every node in the cluster to remove their respective cache entries. This project is to replace the clustering based mechanism and switch to a pub-sub (publish-subscribe) mechanism to remove the cache entries. The proposed solution is something that is already installed implemented on the WSO2 API Microgateway. This project is to implement the same solution on the regular API gateway.

**Deliverables:**

A feature on the WSO2 API gateway that is capable of revoking cache entries based on a pub-sub model.

**Skills Needed:** Java, message brokers, JMS

## **Project 37 : Capacity Forecasting for Enterprise Integrator**

**Mentor Name** : Maninda Edirisooriya (maninda@wso2.com)

**Duration** : 6 Months

**Number of people** : 1 Person

**Description**

Run some scenarios with Enterprise Integrator (EI) and collect capacity data like CPU and memory consumption. Research in machine learning domain for best potential models for forecasting capacity. (Time Series Forecasting models) Build models with collected data and test for accuracy. Implement a notification system for running out of future capacity.

**Deliverables**

* Artifacts for running scenarios in EI for collecting data like scripts.
* Data collected for common scenarios in EI.
* Machine Learning code for generating models.
* Notification system code for capacity overrun.

**Skills Needed**

Basic Machine Learning knowledge including Deep Neural Networks are required. Python programming skills are good to have. Will be required to develop with Tensorflow.

## Proposal 38: Redis Caching support for Enterprise Integrator

**Mentor Name:** Asanka Abeyweera **(**asankaab@wso2.com)

**Project Duration:** 3 months

**Number of people:** 1

**Project Description and Scope:**

Redis is an open source, in-memory data structure store used as a database, a cache and a message broker. This project is about implementing a way that WSO2 EI can leverage Redis as the caching layer to cache the responses.

#### 

**Deliverables**

* Implementation of Redis cache for EI
* Documentation
* Unit/Integration tests

#### 

**Possible Technologies involved:**

1. Java, WSO2 EI, Redis, Caching techniques
2. <https://docs.wso2.com/display/EI640/Cache+Mediator>
3. <https://docs.wso2.com/display/EI640/WSO2+Enterprise+Integrator+Documentation>
4. <https://redis.io/>

Proposal 39: Tracing Support for WSO2 EI with CNCF Jaeger

**Mentor Name:** Sajith Ravindra (sajithr@wso2.com)

**Project Duration:** 6 months

**Number of people:** 1

**Project Description and Scope:**

Jaeger is an open source distributed tracing system which operates on open tracing standards. WSO2 EI is one of the well known, open source integration products available in the market. EI lets users deploy services, APIs, etc, messages received by such services are passed through 1 or more ’sequences’, a sequence is a set of constructs referred to as ‘mediators’ connected with each other, each performing different actions. These actions include message transformation, calling external services, creating XML payloads, etc. Enabling tracing for EI will let users clearly observe the execution flow and message flow through different mediators of a service, identify performance bottlenecks, troubleshoot problems, etc. The scope of this project is to use Jaeger to provide tracing for WSO2 EI.

**Deliverables**

* Open Tracing/Jaeger Implementation for WSO2 EI
* Article on Usage of Tracing with WSO2 EI with a demo

**Possible Technologies involved:**

* Java
* WSO2 EI
* OpenTracing : <https://opentracing.io/>
* Apache Synapse : <https://en.wikipedia.org/wiki/Apache_Synapse>
* CNCF Jaeger : <https://www.jaegertracing.io/>

## Proposal 40: First class support for Streaming integration in Micro Integrator

**Mentor Name:** Sajith Ravindra (sajithr@wso2.com)

**Project Duration:** 6 months

**Number of people:** 1

**Project Description and Scope:**

Streaming Integrator which is based on WSO2 Siddhi can consume streaming data, process them and derive useful insights and execute lightweight integration scenarios. However, it’s not capable of carrying out heavy duty integration scenarios such as service chaining, calling multiple endpoints, etc. Micro integrator is the cloud-native version of WSO2 EI, it’s capable of implementing complex enterprise integration scenarios and is not designed to handle streaming data. Modern enterprises have to consume streaming data more often and require to take complex actions by processing this data, which demand both streaming and integration capabilities in a single pipeline of execution. The goal of this project is to provide a mechanism for Streaming Integrator and Micro Integrator to work with each other in a seamless manner by making calls to each other in a transparent manner.

#### **Deliverables**

* Implement support to make calls to Micro Integrator from Streaming Integrator and get the feedback
* Implement support to make calls to Streaming Integrator from Micro Integrator and get the feedback
* Whitepaper on the title “Enterprise Integration empowered by Streaming Data”

**Possible Technologies**

* Java
* WSO2 EI
* CNCF gRPC : <https://grpc.io/>
* Apache Synapse : <https://en.wikipedia.org/wiki/Apache_Synapse>
* WSO2 Siddhi : [https://github.com/siddhi-io](https://redis.io/)

## **Project 41 : Implement a NATS connector for (Micro) Integrator**

**Mentor Name** : Hasitha Abeykoon ([hasithah@wso2.com](mailto:hasithah@wso2.com))

**Duration** : 6 months

**Number of people** : 1 Person

**Description:**

NATS is a modern messaging system that facilitates high performance messaging and eventing capabilities. In micro service architecture it is a common pattern in integrate micro services using asynchronous events. To cater this, WSO2 Micro Integrator needs to have the capability to integrate with NATS platform seamlessly, so that the micro service developers can only focus on the business logic. This project is on developing a NATS Connector for WSO2 MI.

**Deliverables:**

* Connector for sending out messages to NATS messaging platform
* Inbound endpoint to listen and receive messages from NATS
* Documentation
* Integration Tests

**Possible Technologies involved**

<https://nats-io.github.io/docs/>

<https://docs.wso2.com/display/EI650/WSO2+Enterprise+Integrator+Documentation>

## **Project 42 : Attaching and Configuring Policies for APIs with API Publisher**

**Mentor Name** : Chanaka Jayasena ([chanaka@wso2.com](mailto:chanaka@wso2.com)), Malintha Amarasinghe ( [malinthaa@wso2.com](mailto:malinthaa@wso2.com))

**Duration** : 6 months

**Number of people** : 1 Person

**Description:**

1st Phase:

Support for single mediation policy designing and uploading to the backend via the UI. The backend already has support for executing single policy for each In, Out, and Fault flow of API requests. We need to add drag and drop support for designing the mediation flow. Support for validating the policies via backend service.

2nd Phase:

Extend the above for multiple sequences. This will involve both backend and UI changes.

**Possible Technologies involved:**

* Java, CXF, OpenAPI, Apache Synapse
* React, Webpack, Babel, ES6, JSS, CSS, HTML5

**Deliverables**

* **Phase 1:** Implementing support to design and engage mediation policies (In, Out, Fault flows) through publisher UI.
* **Phase 2:** Implementing support to design and engage multiple mediation policies (In, Out, Fault flows) through publisher UI.
* Documentation
* Integration Tests

**References**

<https://docs.wso2.com/display/AM260/Convert+a+JSON+Message+to+SOAP+and+SOAP+to+JSON>

## **Project 43: GraphQL support for Ballerina Integrator**

**Mentor Name:** Asitha Nanayakkara (asitha@wso2.com)

**Project Duration:** 3 months

**Number of people:** 1

**Project Description and Scope:**

GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.

This project is about designing and implementing a mechanism to expose and consume GraphQL based APIs using Ballerina.

Deliverables

Implementation of GraphQL API support in Ballerina

Documentation

Unit/Integration tests

#### 

**Possible Technologies involved:**

GraphQL, WSO2 EI Fundamentals

#### 

[1] <https://graphql.org/>

[[2] https://ballerina.io/learn/by-example/](https://graphql.org/)

## 

## **Project 44: Siddhi monitoring using Grafana**

**Mentor Name:** Ramindu De Silva (ramindus@wso2.com)

**Project Duration:** 6 months

**Number of people:** 1

**Project Description and Scope:**

The users need to check whether the SP nodes in which their Siddhi applications are deployed are working as expected, or whether any Siddhi application requires performance improvements. Siddhi should be fully observable by exposing values of throughput, latency, etc of siddhi components using prometheus metrics and should use Grafana dashboard to visualize the information.

Deliverables

Grafana dashboards

Integrating prometheus into Siddhi

#### 

**Possible Technologies involved:**

Java, Siddhi, Grafana, Prometheus

## **Project 45 : AI based Recommendation System for API Store**

**Mentor Name:** Fazlan Nazeem ([fazlann@wso2.com](mailto:fazlann@wso2.com)) , Sanjeewa Malalgoda (sanjeewa@wso2.com)

**Duration :** 6 Months

**Number of people :** 1 Person

**Description**

WSO2 API Manager Store Portal is a marketplace for APIs. Developers can login to the portal to choose and subscribe which APIs to leverage in their applications. Currently, the developer has to pick the APIs of interest by browsing through the APIs or searching with a particular information associated with the API.

This project is to introduce an AI based recommendation system (collaborative filtering) to recommend APIs which can be beneficial for the developer. The recommendation system has to build models with API subscription data and suggest similar APIs while the developer is browsing through the portal.

**Deliverables**

* Recommendation System, UI implementation , Documentation

**Skills Needed**

Java, Tensorflow, ReactJs

[1] <https://cloud.google.com/solutions/machine-learning/recommendation-system-tensorflow-overview>

## **Project 46 : Lightweight Siddhi Runner in Java**

**Mentor Name** : Suho (suho@wso2.com)

**Duration** : 6 Months

**Number of people** : 1 Person

**Description**

Siddhi [1] is a complex event processing library which supports processing millions of events per second. It is used as the core engine in WSO2 Stream Processor product [2] as well. Siddhi CEP engine is implemented using Java programming language and it can be used as the library with any Java-based implementation and solution. We have built the Siddhi runner implementation [3] using the OSGi framework. At the moment, the startup time of the runner varies from 5 sec to 15 sec based on the OSGi bundle loading order in different OSs. We are looking at a way to move away from OSGI, in which it results in quick start-up time; this will be highly helpful to position Siddhi as the Cloud Native Stream Processor.

**Deliverables**

Siddhi Runner Impl in Java

Integration Tests

**Skills Needed**

Siddhi, Java, OSGI

[1] <https://github.com/wso2/siddhi>

[2] <https://wso2.com/analytics-and-stream-processing/>

[3] <https://github.com/siddhi-io/distribution>

## **Project 47: Implement Language Server (LSP) on Adaptive Authentication Scripts for WSO2 Identity Server**

**Mentor Name**: ruwana@wso2.com

**Duration**: 5 Months

**Number of people** : 1 Person

**Description**

Currently, IAM is very short on developer friendliness, in contrast to strong “Identity Administrator” capabilities. We need to improve the “developer” capabilities in IAM.

Implement LSP in identity server, targeting developer. Starts with adaptive authentication script.

The developer should be able to work on adaptive authentication scripts with VS Code IDE.

**Deliverables**

* Implement Language Server (LSP) on Adaptive Authentication scripts. Add intellisense, code completion
* Add IntelliJ Idea, VS-Code support for authentication script
* Screencasts/Webinar/Blogs
* Documentation

**Skills Needed**

* Java/ Javascript

[1] <https://langserver.org/>

## **Project 48: Git Integration on Authentication Flow Artifacts (SP, IdP)**

**Mentor Name**: ruwana@wso2.com

**Duration**: 4 Months

**Number of people**: 1 Person

**Description**

Currently, IAM is very short on developer friendliness, in contrast to strong “Identity Administrator” capabilities. We need to improve the “developer” capabilities in IAM.

Each tenant in the Identity Server should be able to configure his own Git repository URL, Access Token, and branch name.

The Identity Server will poll the Git Repository for updates in the artifacts and load them to the current runtime.

Errors occurred while loading should be logged per tenant with tenant scoped error log, which can be visible by the developers/admin of the tenant.

There should be one master node which can do the Git Pull. Master election mechanism needs to be created. Current Hazelcast election can be reused.

**Deliverables**

* Git integration on Authentication flow artifacts (SP, IdP) via IDE (Already present)
* Load the SP by Identity server via Git Repositories
* Load the IdP by Identity server via Git Repositories
* Load the “User Store” by Identity server via Git Repositories
* Screencasts / Webinar / Blogs
* Documentation

**Skills Needed**

* Java, Visual Studio Code IDE

[1] <https://github.com/wso2-extensions/identity-fetch-remote>

## **Project 49: Debugger for Identity Server Authentication Flow**

**Mentor Name**: isura@wso2.com

**Duration**: 5 Months

**Number of people**: 1 Person

**Description**

Currently IAM is very short on developer friendliness, in contrast to strong “Identity Administrator” capabilities. We need to improve the “developer” capabilities in IAM.

Debug support is necessary for developers to see what is going on the Identity Server when performing the authentication flow and authorization flow.

**Deliverables**

* Develop plugin for embedded mode of Identity Server in for IntelliJ Idea in debug mode, and VS-Code to run and test the IAM
* Debug support for Authentication flow
* Screencasts / Webinar / Blogs
* Documentation

**Skills Needed**

* Java

[1] <https://docs.oracle.com/javase/8/docs/technotes/guides/jpda/index.html>

## **Project 50: Support for OAuth Device Flow in WSO2 Identity Server**

**Mentor Name**: omindu@wso2.com

**Duration**: 3 Months

**Number of people**: 1 Person

**Description**

OAuth Device Flow formally known as “OAuth 2.0 Device Authorization Grant”[1][2] is an OAuth 2.0 extension that enables devices with no browser (e.i IoT devices) or limited input capability (ex: smart tv) to obtain user authorization to access protected resources.

The purpose of the project is to implement OAuth device flow support to WSO2 Identity Server. A quick read up on the device flow can be found at [3] and a sample implementation can be found at [4].

**Deliverables**

* WSO2 IS compliance for OAuth device flow
* Unit/Integration Tests
* Sample application to demonstrate the flow
* Documentation
* Article/Blogs/Screencasts

**Skills Needed**

* Skill to read, understand and translate specifications into working code
* Java, Maven, Git

[1] - <https://tools.ietf.org/html/draft-ietf-oauth-device-flow>

[2] - <https://oauth.net/2/device-flow>

[3] - <https://alexbilbie.com/2016/04/oauth-2-device-flow-grant>

[4] - <https://developer.okta.com/blog/2019/02/19/add-oauth-device-flow-to-any-server>

## **Project 51: Support for OpenID Client Initiated Backchannel Authentication Specification**

**Mentor Name**: farasatha@wso2.com

**Duration**: 4 Months

**Number of people**: 1 Person

**Description**

OpenID Connect Client Initiated Backchannel Authentication Flow is an authentication flow like OpenID Connect. However, unlike OpenID Connect, there is direct Relying Party to OpenID Provider communication without redirects through the user's browser. This specification allows a Relying Party that knows the user's identifier to obtain tokens from the OpenID Provider. The user consent is given at the user's Authentication Device mediated by the OpenID Provider.

**Deliverables**

* WSO2 IS compliance for CIBA Specification
* Unit/Integration Tests
* Sample application to demonstrate the flow
* Documentation
* Article/Blogs/Screencasts

**Skills Needed**

* Skill to read, understand and translate specifications into working code
* Java, Maven, Git

[1] - <https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0-02.html>

## **Project 52: Incommon Federation Implementation for WSO2 Identity Server**

**Mentor Name**: malithim@wso2.com

**Duration**: 4 Months

**Number of people**: 1 Person

**Description**

InCommon federation provides a common framework for trusted shared management of access to online resources. With this framework, Identity Providers can provide Single Sign On capabilities and privacy protection for their users. Thus, online Service Providers no longer need to maintain user accounts. Objective of this project is to implement Incommon federation support for WSO2 Identity Server.

A high level break down of possible tasks to complete would be,

Understand InCommon Federation compliance requirements and SAML protocol

Understand the current WSO2 Identity Server SAML and SAML extension processing implementations

Identify the gaps in the WSO2 IS implementation to support InCommon Federation.

Implement fixes to bridge the gaps in the WSO2 IS implementation to be compliant with InCommon Federation.

**Deliverables**

* In built feature that makes WSO2 Identity Server compliance with InCommon Federation
* Screencasts / Webinar / Blogs
* Documentation

**Skills Needed**

* Skill to read, understand and translate specifications into working code
* Java, Maven, Git

[1] <https://www.incommon.org/federation/>

[2] <https://www.incommon.org/federation/metadata.html>

[3] <https://github.com/wso2-incubator/identity-incommon-manager/pull/1>

## **Project 53: Biometric Authenticator to Integrate with Mobile Devices**

**Mentor Name**: pulasthim@wso2.com

**Duration**: 3 Months

**Number of people**: 1 Person

**Description**

Authenticator to authenticate users using existing biometric hardware(preferably fingerprint) in Mobile devices.

**Deliverables**

* Mobile integrable biometric authenticator
* Unit/Integration tests
* A mobile application with an integrated biometric authenticator
* Documentation
* Article

**Skills Needed**

* Java
* Knowledge in mobile application development

[1] <https://docs.wso2.com/display/IS570/Writing+a+Custom+Local+Authenticator>

## **Project 54: OpenID RISC Profile of IETF Security Events**

**Mentor Name**: isharak@wso2.com

**Duration**: 4 Months

**Number of people**: 1 Person

**Description**

WSO2 Identity Server has an eventing framework[1] that publishes events for all important security events such as user credentials change, user account lock, token revoke etc.

RISC[2] (Risk and Incident Sharing and Coordination) Working Group, focuses on coming up with a standard to enable sharing important security events among service providers. The working group has come up several specs[3] to build a common standard on how the events will be represented and shared among different entities who are interested in security events. The main spec that describes the event sharing model is then OpenID RISC Profile of IETF Security Events spec[4].

In this project, we implement an event sharing model based on the “OpenID RISC Profile of IETF Security Events 1.0”[4] defined by RISC WG. We can reuse the eventing framework and implement the event sharing model on top of it.

**Deliverables**

* Implementation of OpenID RISC Profile of IETF Security Events 1.0 spec
* Unit / Integrations tests
* Building a sample to demonstrate a use case of the OpenID RISC Profile of IETF Security Events
* Screencasts / Webinar / Blogs
* Documentation explaining key concepts, use cases and how an external developer can consume the events shared

**Skills Needed**

* Skill to read, understand and translate specifications into working code
* Java

[1] <https://medium.com/@isurakarunaratne/wso2-identity-server-eventing-framework-32505bcc1600>

[2] <https://openid.net/wg/risc/>

[3] <https://bitbucket.org/openid/risc/src/master/>

[4] <https://openid.net/specs/openid-risc-profile-1_0-ID1.html>

## **Project 55: Developer SDKs for WSO2 Identity Server**

**Mentor Name**: ayesha@wso2.com

**Duration**: 2 Months

**Number of people**: 2 Persons

**Description**

The requirement of the project is to implement a set of SDKs (Software Development Kits) which can be used to integrate OpenID Connect based authentication with WSO2 Identity Server. For the project, it we are targeting to develop React, Android and IOS SDKs. AppAuth libraries by OpenID foundation [1] can be used as a base for the SDKs

**Deliverables**

* SDKs for React, Android, IOS, JS
* Samples
* Screencasts / Webinar / Blogs
* Documentation

**Skills Needed**

* Knowledge in React/Android/IOS/JS

[1] - [https://github.com/openid?utf8=%E2%9C%93&q=AppAuth&type=&language=](https://github.com/openid?utf8=✓&q=AppAuth&type=&language=)

[2] - <https://developer.okta.com/code/javascript/>

## **Project 56: Support for SAML2 Federated IdP Initiated Logout**

**Mentor Name**: thanuja@wso2.com

**Duration**: 4 Months

**Number of people**: 1 Person

**Description**

In the current Identity Server distribution, we have the support for SAML2 SP (Identity Server in this case) initiated federated IdP log out. The purpose of this project is to provide the capability to handle federated IdP initiated logout requests/responses.

**Deliverables**

* Identity Server should able to handle SAML2 logout request send by federated IdP and response back as a proper SP
* If required, terminate the related SP sessions connected to the Identity Server. (Can use cross-protocol logout feature)
* Unit / Integrations tests
* Screencasts / Webinar / Blogs
* Documentation

**Skills Needed**

* Skill to read, understand and translate specifications it to working code
* Java

[1] -<https://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf>

## **Project 57: Support OpenSAML 3.0 for WSO2 Identity Server**

**Mentor Name**: maduranga@wso2.com

**Duration**: 4 Months

**Number of people**: 1 Person

**Description**

OpenSAML2 reached its end of life 3 years back (July 31, 2016 [1]). Security fixes are no longer provided. So we need to upgrade it to the V3 immediately. Problem is there are multiple API changes in the V3 version. The purpose of this project is to upgrade and existing SAML2 related repos to OpenSAML3.

**Deliverables**

* Upgrade existing SAML2 related repos to support OpenSAML3
* Unit / Integrations tests.

**Skills Needed**

* Skill to read, understand and translate specifications it to working code
* Java

[1] - <https://wiki.shibboleth.net/confluence/display/OS30/Home>

## **Project 58: Implement a Financial API -Confirmation of Payee (for WSO2 Open Banking Solution)**

**Mentor Name:** amalka@wso2.com

**Project Duration:** 4 months

**Number of people:** 1

**Project Description and Scope:**

With the global trend of Open Banking concept, banks have started to expose bank customers’ account data to third party and do real time payments from customer account directly through APIs. Confirmation of Payee describes a specification about an endpoint to define by banks to identify the correct payee. This endpoint has defined from open banking uk specification to be consumed with payment API flow.

This project is about designing and implementing the API endpoint for confirmation of payee in WSO2 Open Banking solution with following the respective specification.

**Deliverables**

Implementation of Confirmation of Payee API in Open Banking solution

Mock backend to test the flow

Documentation

Unit/Integration tests

#### 

**Possible Technologies involved:**

Java , Open API definitions, OAuth2, OIDC, SQL , WSO2 API Manager, WSO2 Identity Server, Open Banking

**Skills Needed**

* Skill to read, understand and translate specifications into working code

[1] <https://www.wearepay.uk/confirmation-of-payee/>

[2] <https://openbanking.atlassian.net/wiki/spaces/DZ/overview>

## **Project 59: Third Party Provider [TPP] Aggregator for Financial APIs**

**Mentor Name:** [lalaji@wso2.com](mailto:lalaji@wso2.com)/ [amalka@wso2.com](mailto:amalka@wso2.com)

**Project Duration:** 6 months

**Number of people:** 2

**Project Description and Scope:**

With the global trend with Open Banking, banks have started to expose bank customers’ account/payment data to third parties [application developers] through APIs. There are different specifications released to define financial APIs and different banks use different spec based APIs. As a TPP ,it has to connect to multiple banks which exposed different API formats. Thus as a TPP, it has to have a solid framework to cater API data aggregation layer with proper integration points and storage mechanisms.

This project is about designing and implementing the Aggregator framework layer in WSO2 Open Banking solution as a new module.

**Deliverables**

Data aggregator module with proper integration points to available spec [ open banking UK and Berlin] and data storage mechanism

Documentation

Unit/Integration tests

**Possible Technologies involved:**

Java , Open API definitions, SQL , WSO2 API Manager, Enterprise Integrator

## **Project 60: Improvements for deployment automation in WSO2 Open Banking solution**

**Mentor Name:** [lalaji@wso2.com](mailto:lalaji@wso2.com) /[amalka@wso2.com](mailto:amalka@wso2.com)

**Project Duration:** 5 months

**Number of people:** 2

**Project Description and Scope:**

WSO2

**Project Description and Scope:**

**First 2 months -Enhancing the tryout setup of WSO2 Open Banking**

WSO2 Open Banking solution’s current demo website can be found from <https://openbanking.wso2.com/> .It has few limitations as current tryout setup is based on old API spec version and it doesn’t show how latest open banking UK, Berlin, AU specific APIs to tryout. And it’s not easy to update the demo setup upon spec version upgrades at the moment due to the underline Open Banking solution deployment in there is not a full automated one.

This project idea is to make the solution to be deployed with full automated scripts and then simplify user experience while trying out the demo setup.

**Next 2 months -Dockerize the main WSO2 OB deployment patterns**

WSO2 Open Banking solution need to be support dockerized setup.Currently we do have the basic docker files to create the solution.But need to improve the existing docker files and create docker files for main deployment patterns.

**Deliverables**

* Develop a quick setup script for developers to consume WSO2 Open Banking solution
* Upgrade current OB demo website with created automation scripts to cater open banking uk,berlin,AU specific tryout sandbox environments.
* Maintenance & setup upgrade Documentation on the demo setup
* Dockerized setup for main OB deployment patterns

**Possible Technologies involved:**

Puppet,Docker, Docker Compose, Shell scripting , WSO2 API Manager WSO2 Identity Server fundamentals and WSO2 Open Banking solution on open banking uk, berlin,AU specifications specific flows

## **Project 61: GraphQL Integration for Enterprise Integrator**

**Mentor Name:** Nandika (nandika@wso2.com)

**Project Duration:** 3 months

**Number of people:** 1

**Project Description and Scope:**

GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.

This project is about designing and implement a mechanism to expose and consume GraphQL based APIs from EI.

Deliverables

Implementation of GraphQL API support for EI

Documentation

Unit/Integration tests

#### 

**Possible Technologies involved:**

GraphQL, WSO2 EI Fundamentals

#### 

[1] <https://graphql.org/>

[[2]](https://graphql.org/) <https://docs.wso2.com/display/EI640/WSO2+Enterprise+Integrator+Documentation>

## **Project 62:** **Log4j2 async logging support for log mediator**

**Mentor Name:** Isuru Udana

**Project Duration:** 3 months

**Number of people:** 1

**Project Description and Scope:**

Asynchronous Loggers are a new addition to Log4j 2 library. Their aim is to return from the call to Logger.log to the application as soon as possible. We can choose between making all Loggers asynchronous or using a mixture of synchronous and asynchronous Loggers. Making all Loggers asynchronous will give the best performance, while mixing gives you more flexibility.

Currently WSO2 Enterprise Integrator uses 1.x version of Log4j library.

This project is about Leverage log4j2 APIs in log mediator to do async logging, so that there is no performance impact on the mediation message processing when we have enabled logging.

**Possible Technologies involved:**

Java, WSO2 ESB, Log4j

[1] <https://logging.apache.org/log4j/2.x/manual/async.html>

[2] <https://docs.wso2.com/display/EI650/Log+Mediator>

## **Project 63:** **K8s developer workflow in EI tooling**

**Mentor Name:** Indika Sampath (indikas@wso2.com)

**Duration:** 3 Months

**Number of people:** 1 Person

**Project Description and Scope:**

A CRD call 'integration' define to deploy synapse project into the k8s cluster. The operator already implemented with basic features [1] using operator-sdk[2]. It required supportive feature from the EI tooling to build a Docker image with the Synapse project, push it to docker registry and generate CRD yaml file as an output of the build process [3].

**Possible Technologies involved:**

Kubernetes, Docker, Java

[1] <https://github.com/wso2/k8s-ei-operator>

[2] <https://github.com/operator-framework/operator-sdk>

[3] [https://github.com/wso2/k8s-ei-operator#developer-workflow](https://github.com/wso2/k8s-ei-operator" \l "developer-workflow)

### **Project 64: Maintenance dashboard and automation of incident reporting of WSO2 Public Cloud**

**Mentor Name:** Manjula Rathnayaka(manjular@wso2.com)

#### **Project Duration:** 3 months

#### **Number of people:** 1

#### **Project Description and Scope:**

WSO2 Cloud currently performs the incident handling process via google sheets and scripts. This includes incident investigating, resolving, and post mortem analysis for customers in different regions as well as cloud offerings. The project is to implement these functionalities as a web application and improve the incident handling process. Furthermore, a maintenance dashboard is required to show previous, current and future maintenance activities to the customers.

**Deliverables**

Implementation and deployment of the incident reporting and maintenance dashboard applications in the production environment.

#### **Possible Technologies involved**

Java, React, Amazon SES, some knowledge on OAuth2.0 and Microservices.

#### **References**

[1] - <https://wso2.com/cloud/>

[[2] -](https://wso2.com/cloud/) <https://aws.amazon.com/ses/getting-started/>

### 

### **Project 65: Disaster Recovery for WSO2 API Cloud**

#### **Mentor Name :** Sumedha Kodithuwakku (sumedhas@wso2.com)

#### **Duration :** 4 Months

#### **Number of people :** 1 Person

#### **Description**

WSO2 Cloud currently does not have a properly automated disaster recovery plan although all the necessary backups, data and configs are properly maintained. The project is to implement the Disaster recovery plan using those available resources. This will include defining the disaster recovery plan (helped by the team members) and the supporting tools/scripts which are required to restore or create a new deployment in case of a disaster.

**Deliverables**

A disaster recovery plan and necessary tools, scripts required for recovering WSO2 Cloud in case of a disaster

#### **Possible Technologies involved**

Scripting, AWS technologies (EC2, RDS, Route 53 etc), WSO2 products

#### **References**

[1] <https://searchdisasterrecovery.techtarget.com/definition/disaster-recovery>

[2] <https://www.networkworld.com/article/3248969/what-is-disaster-recovery-how-to-ensure-business-continuity.html>

[3] <https://aws.amazon.com/>

### **Project 66: WSO2 Public Cloud log collection, filtering and storing on different SaaS services using Fluentd**

**Mentor Name:** Manjula Rathnayaka(manjular@wso2.com)

#### **Project Duration:** 3 months

#### **Number of people:** 1

#### **Project Description and Scope:**

### WSO2 Cloud currently generates a huge amount of logs of different servers of API, Identity and Integration Clouds. These logs are required to be filtered per customer, per cloud service, etc using open source data collecting tool 'Fluentd'. Once these logs are filtered, they are published to Amazon CloudWatch, S3 and other log collecting cloud services based on customer preferences. This is partially implemented using log4j appenders but it has its own limitations. Hence the new approach is selected.

### **Deliverables**

### Fluentd high available deployment to filter cloud service logs of the load balancer, carbon servers and publish them to Amazon CloudWatch. This replaces the existing log monitoring and download solution in the production deployment.

#### **Possible Technologies involved**

### Amazon CloudWatch, Fluentd and Scripting

#### **References**

[1] - <https://wso2.com/cloud/>

[[2] -](https://wso2.com/cloud/) <https://www.fluentd.org/>

[[3] -](https://www.fluentd.org/) <https://aws.amazon.com/cloudwatch/>

## **Project 67:** **Fraud detection in WSO2 API Cloud**

**Mentor Name:** Amila Mahaarachchi (amilam@wso2.com)

**Duration:** 4 Months

**Number of people:** 1 Person

**Project Description and Scope:**

Currently, we do not have a way to detect if any unauthorized people are tampering with customers APIs. For example, if a customer’s OAuth application secrets are compromised, a hacker can use them without us knowing it. So, the idea of the project is to use AI to detect such misuses. We believe this can be done by looking at the IPs, user agent, geo location, time which the tokens are generated. This project will be done in conjunction with the WSO2 research team.

We plan to extend this to other scenarios later.

**Possible Technologies involved:**

WSO2 API Manager, Java, Tensorflow (possibly)

## **Project 68:** **Prometheus based traffic monitoring for WSO2 API Cloud**

**Mentor Name:** Amila Mahaarachchi (amilam@wso2.com)

**Duration:** 3 Months

**Number of people:** 1 Person

**Project Description and Scope:**

WSO2 API Cloud is a large deployment which spans across 6 regions in the world deployed on top of AWS. We are serving nearly 100 million API calls per month collectively throughout all these regios. We are blind at the moment on knowing how much traffic is handled by a region at a given time. Also, we do not have intel on the pattern of traffic handling in the history too.

Objective of the project is to tap into Nginx, get the traffic info and publish to Prometheus in a way we can solve the above problems.

**Possible Technologies involved:**

NginX, Prometheus, Nginx Lua scripting

## **Project 69:** **New billing vendor support for API monetization in API Cloud**

**Mentor Name:** Isuru Wimalasundara (isuruw@wso2.com)

**Duration:** 4 Months

**Number of people:** 1 Person

**Project Description and Scope:**

API Cloud supports API monetization [1]. I.e. Customers can charge their customers for consuming their APIs. Currently, this automatic charging is done using the billing vendor “Stripe”. But, Stripe is not supported in some countries. So, we need to support another billing vendor which is supported in more countries. This project is for that.

[1] https://docs.wso2.com/display/APICloud/API+Monetization

**Possible Technologies involved:**

WSO2 API Manager, Java, Paypal kind of a billing vendor which will be selected upon a research.