**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**First Semester 2025-26**

**CSIW ZG628T DISSERTATION**

**Dissertation Outline**

**BITS ID No. 2021WA86438**

**Name of Student: Abid Rafique Khan**

**E-mail ID of the student: 2021wa86438@wilp.bits-pilani.ac.in**

**Name of Supervisor:** **Ayush Ranjan**

**Designation of Supervisor**: **Project Engineer**

**Qualification and Experience: BTech – 4.9 Years**

**E- mail ID of Supervisor:** [**ayush.ranjan1@wipro.com**](mailto:ayush.ranjan1@wipro.com)

**Title of Dissertation**: **FlexiFaaS - A Secure Serverless Function Deployment and Execution Platform for Cloud-Native Applications**

**Name of First Examiner: Mrugesh Mistry**

**Designation of First Examiner**: **Account Manager**

**Qualification and Experience: B.E. Comp – 12 Years**

**E- mail ID of First Examiner:** [**mrugesh.mistry@wipro.com**](mailto:mrugesh.mistry@wipro.com)

**Name of Second Examiner: Ajay Kumar**

**Designation of Second Examiner: Technical Manager**

**Qualification and Experience: BE – 20 years**

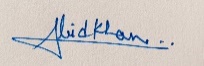
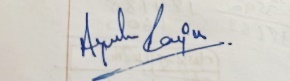
**E- mail ID of Second Examiner:** [**ajay.kumar@wipro.com**](mailto:ajay.kumar@wipro.com)

**Supervisor’s rating of the Technical Quality of this Dissertation Outline**

EXCELLENT / GOOD / FAIR/ POOR (Please specify): EXCELLENT

**Supervisor’s suggestions and remarks about the outline:**

The outline of the dissertation is thorough, well-organized, and technically robust. It manifestly shows exhaustive comprehension of scholarly concepts and practical applications within serverless and cloud-native design patterns. The intended methodology and outcomes are very pertinent and feasible within the project schedule. Overall, this is an outstanding outline and is recommended for approval and implementation to a great extent.

(Signature of Student) (Signature of Supervisor)

Date: 23/07/2025 Date: 25/07/2025

**Body of the outline**

**1. Discussion on the chosen topic**

**1.1 Purpose of the Work and Expected Outcome**

The aim of this project is to create a platform for securely and efficiently deploying serverless functions. It will integrate authentication, encryption, and DevOps practices into one system.

The project seeks to design and develop an engine for deploying serverless functions. This engine will allow users to upload, configure, deploy, and execute standalone functions smoothly. It is targeted at environments where developers or internal teams need to run backend logic without managing infrastructure manually.

Expected Outcome:

* A backend engine that can register, version, and deploy user-defined functions
* A secure middleware layer for handling encryption, authentication, and file safety checks
* An intuitive web frontend for uploading functions, monitoring, and controlling execution
* A complete CI/CD pipeline for automatic building, testing, and checking code quality
* Extensible support for containerization and optional deployment using Docker

When implemented in an enterprise environment, this project will help speed up the development of internal tools, automation scripts, and microservices orchestration. It will also ensure compliance with internal IT security policies.

**1.2 Literature Review**

Serverless computing is becoming more popular in modern application architecture. It allows developers to focus on writing code without managing infrastructure. Commercial services like AWS Lambda, Azure Functions, and Google Cloud Functions made this model well-known, but they often lack flexibility for internal, customizable workflows.

In research and open-source communities:

OpenFaaS provides Docker function deployments but is difficult to operate.

Knative needs extensive Kubernetes knowledge.

Many of the academic papers from IEEE and ACM highlights the need for internal Function-as-a-Service (FaaS) platforms which work better with CI/CD, security, and organizational workflows.

This project takes advantage of the usability of Spring Boot and ReactJS, merging the best of serverless concepts and secure internal infrastructure tooling.

**1.3 Existing Process and Limitations**

Currently, most company rely on external cloud platforms to deploy server-less functions or write custom scripts in production systems. This has the following limitations:

* Vendor Lock-In: Functions tied to AWS/GCP cannot be migrated easily.
* Security Risk: Sensitive logic to public clouds may violate compliance or uploading internal code.
* Lack of Flexibility: Developers must adjust to command-line interface workflows and packaging that is specific to each provider.
* No Standard Internal Tools: Many teams lack a centralized platform or re-invent function deployment.

This project will fill the gap by offering an internal FaaS engine that is secure, customizable, and modular — deployable on servers or cloud VMs.

**1.4 Justification for Selected Methodology**

The selected approach divides the solution into three distinct phases: backend, middleware, and frontend. This configuration facilitates concurrent development and testing. The architecture promotes loose coupling and a clear separation of concerns, thereby enhancing maintainability.

* Spring Boot is employed for both backend and middleware due to its widespread adoption in enterprises, its simplicity in creating REST APIs, and its robust support for security and encryption libraries.
* MySQL offers a reliable way to organize relational data, especially when it comes to user and function information.
* ReactJS is chosen because of its component-oriented architecture and responsiveness.

The CI/CD stack comprises GitHub Actions, SonarQube, and Maven, which collectively ensure quality control.

This methodology fosters iterative development, automate the CI/CD processes, and accommodates the future enhancements such as container orchestration with Docker and Kubernetes.

**1.5 Dissertation Methodology**

The approach for dissertation methodology is agile development via modular stages:

Stage 1: Backend Development (Spring Boot)

* Implement REST APIs for function creation, execution, and logging
* Integrate RabbitMQ for asynchronous execution
* Integrate with MySQL for handling persistent data
* Implement file storage and calling logic

Stage 2: Middleware Services (Spring Boot)

* Implement user authentication via JWT
* Implement services for AES-based file encryption/decryption
* Implement simulated virus/malware scanning on upload

Stage 3: Frontend Development (ReactJS)

* Login UI, function upload, dashboard UI
* Axios integration with backend and middleware
* Live feedback on deployment and execution outcomes

Every stage consists of:

* Weekly milestones
* Postman/API testing
* Unit testing
* Documentation and logging of progress

**1.6 Benefits derivable from the work**

* **Developer Productivity**: Micro-utilities functions can be deployed by developers without creating VMs or cloud thus increasing developer productivity.
* **Security**: Malware scans and encryption make the functions and files secure in accordance with IT standards.
* **Reusable Infrastructure**: Multiple projects and teams can be supported by a single build.
* Connecting with Jenkins, Docker, or Kubernetes is possible
* Beneficial for organizations aiming to automate and orchestrate internal functions.
* Minimizes the time required for development and testing teams by automating deployment pipelines.
* Improvement in file security and an increase in logging transparency.
* Reduces dependence on cloud vendors for straightforward Function as a Service (FaaS) scenarios

**1.7 Other Supporting Details**

* Allows reverting functions to previous versions
* Ensures safe file management through middleware
* Can be expanded to work with Docker, Kubernetes, or Jenkins for use in production

**2. Detailed Plan of Work** (for 16 weeks)

|  |  |  |  |
| --- | --- | --- | --- |
| **Serial Number of Task** | **Tasks/subtasks** | **Planned duration in weeks** | **Specific Deliverable in terms of the project** |
| **1** | **Finalize scope, tools, and architecture** | **Week 1** | **Architecture diagram, tool justification, tech stack** |
| **2** | **Set up Spring Boot backend with MySQL** | **Week 2** | **REST base project, DB connection, Postman test** |
| **3** | **Define entities, services, repositories, DTOs** | **Week 3** | **Complete data model for users, functions, logs** |
| **4** | **Create endpoints for function registration** | **Week 4** | **File upload API, save metadata, error handling** |
| **5** | **Execution simulation, Postman testing** | **Week 5** | **mock runtime execution, API testing using Postman** |
| **6** | **Integrate RabbitMQ: Configure, test queue setup** | **Week 6** | **Spring Boot AMQP dependency, RabbitMQ broker, test send/receive** |
| **7** | **Refactor invocation: Send function invocation to RabbitMQ queue** | **Week 7** | |  | | --- | |  |  |  | | --- | | **Producer logic (on "Invoke"), create basic consumer/worker** | |
| **8** | **Mid-Term Report + Review** | **Week 8** | **Mid-Sem report: highlight asynchronous flow and messaging, demo working queue** |
| **9** | **Setup middleware Spring Boot app on 8081** | **Week 9** | **JWT-based login, password encryption, integrate RabbitMQ if needed** |
| **10** | **Add file encryption and decryption** | **Week 10** | **AES encryption before storage** |
| **11** | **Add malware scan simulation module (if possible)** | **Week 11** | **Middleware complete, tested APIs** |
| **12** | **Setup React frontend, login/register** | **Week 12** | **React Router, Vite + Tailwind CSS** |
| **13** | **Implement upload dashboard UI** | **Week 13** | **Function upload page, status view, show "queued/executed" status** |
| **14** | **Add invocation UI, logs, result view** | **Week 14** | **Axios to backend, show async output, job status feedback** |
| **15** | **Full integration and UI testing** | **Week 15** | **System tested across all layers with RabbitMQ in workflow** |
| **16** | |  | | --- | |  |  |  | | --- | | **Final report writing, screenshots, viva prep** | | **Week 16** | |  | | --- | |  |  |  | | --- | | **Documentation, code packaging, PPT (includes queue diagrams & async workflow demo)** | |

**3. Any other relevant information in support of your work.**

* The project is in line with topics such as Middleware Technologies, DevOps, Cloud Computing, and Network Security.
* GitHub will serve as the version control system.
* It can be containerized and scaled in the future using Docker/Kubernetes.