TORONTO, XX XXXX AVE
(416) XXX-XXXX
MXXXXXXXXXXXX@GMAIL.COM

MATTHEW KUZMINSKI

A dedicated and detail-oriented software developer with over 8 years of experience in the implementation of large scale applications. Tried and tested in all phases of **SDLC** including requirement gathering, analysis, project scoping, design, coding, testing, deployment and release management. Strongly skilled with **Java** and the **Spring** framework and proficient with **GCP** and building solution on **Kubernetes**. Experienced in leadership with excellent interpersonal and motivational abilities to promote collaborative relationships and high-functioning teams. A problem solver with an aptitude for troubleshooting and the ability to quickly learn new skills and actively adopt new technologies and roles/responsibilities.

EXPERIENCE

SENIOR CLOUD APPLICATION DEVELOPER
TRANSLUCENT COMPUTING
2018 **–** PRESENT

During my time at Transluent Computing I worked on many projects and on different teams. The work involved heavy investment in multiple roles with **NoOps**-like responsibility.. The development flow was **TBD** and **GitOps,** and the development process involved strict **TDD**. The **AGILE/SCRUM** methodology was used with Atlassian **Jira** and both **Sprints** and **Kanban** were used at different times for different projects/purposes. Prod support and a **Firefighter** role was a heavy responsibility especially for the for **GoToLoans/WippyPay** project.

Throughout all projects **Git** was used as a source control system with both **Bitbucket**, **Github** and **Google Cloud Source Repository** as remotes for different projects/initiatives. And **IAC** and **CICD** technology I worked with included **Maven**, **Nexus**, **Jenkins**, **SonarQube, GCR**, **Helm**, **Chartmuseum**, **Monocular**, **Spinnaker**, **Terraform** and **Atlantis**.

To facilitate development for throughout projects at **docker** and **docker-compose** was used extensively, **minikube** and **kind** were used for local Kubernetes clusters**,** and tools as **kubectl**, **helm, tekcli** (proprietary software and wrapper for helm templating and deployment)/**tekapp-cli** (proprietary software to make kubernetes services and endpoints for local routing), **telepresense, nginx-certbot**. For ocal prototyping and testing **Postman** and **newman**, **SoapUI** and **Curl**. And **visualvm**, **eclipse mat** and **Chrome, Angular and Redux DevTools** were used. Other tools and technologies included **swagger2-postman2**, **jhipster**, **editorconfig, prettier, eslint, devcontainers, bash, angular-cli** and **gulp**.

TekStack

TekStack is a platform and a set of core libraries used throughout the most projects I worked on at Transcluent Computing. The library was design as a maven parent project and provided ready APIs and services, and configurations such security.

My role was a developer involved a lot of refactoring and expanding the TekStack libraries especially for GoToLoans/WippyPay. Some of the noteworthy things I worked on were:

1. Overhaul of the database export APIs and Excel document creation services.
2. Expand file management API adding configuration and beans that give support **MinIO** integration
3. Refactor of the application runner to work gracefully with **Liquibase** during application termination and respect **Kubernetes** kill signals to work gracefully with pod rescheduling.

The language I used for TekStack was specifically **Java**, however different languages and technologies compromise the software stack.

GoToLoans/WippyPay

GoToLoans (now **WippyPay**) is loan creation and management system for car maintenance and repairs, under the **TekStack.**

My role throughout this project was at first a senior developer then later an effective lead developer and release manager. I was part of the original development team and worked on the application from its inception, being the biggest and longest contributor to the project. During this project I was heavily involved and working with the BAs and PMs to groom CRs, and create, breakdown and schedule tasks.

GoToLoans/Wippy is implemented with the **twelve-factor app methodology** and broken down into many microservices hosted under **Kubernetes**. The APIs were designed as Rest/HATEOAS respecting **maturity level 3** of the **Richardson Maturity Model**. The consumers were multiple **SPA** webapps being different portals for different user roles and purposes, and an Android app that acted as a kiosk preinstalled on tablets.

The service was hosted under **GCP**, made uses of various **GKE** and various Google services .

I primarily worked on the backend of GoToLoans and specifically the loan managenet and creation. The backend provided amulti-actor workflow for the construction, scheduling and financing of loans. The system was broken down into proprietary subsystems and integrations that authenticate an end user and allow them to: qualify for, construct and subsidize a loan. There were different kinds of end-users in the workflow to approve and finance these loan, or be commissioned through the loan. Calculators/algorithms and were written to determine loan payment breakdown and schedule based on defined conditions and dynamic behavior/reconstruction. My work on this system strickly involved backend exposed over a **REST**/**HATEOAS** for different mobile and web conssumers. The implementation was decomposed into microservices and integrated and orchestrated by **Kubernetes**.

Integrations developed throughout the system

* **CBB**: valuate a vehicle and use the valuation as parameter for the loan quote. The integration was over REST/HTTP.

* **Vincario**: Determine vehicle vin (backup system). The integration was over REST/HTTP

* **Carfax**: Determine vehicle vin and Coshare data with Carfax. There was a mutli-engine integration using HTTP and **SFTP**, **FTPS**

* **Inverite**: Validate customer banking information. The integration was over HTTP

* **Sinch**: Verify customer by SMS. The integration was over HTTP

* **PPSA/RSLA**: Find and register lien on vehicle. The integration was over a **SOAP** api.

* **Docusign**: Managed a loan contract. The integration was over HTTP and webhooks, although abstracted through provided Java Libraries.

* **LoanPro**: Scheduling and management of loans. Multiple engines were built around integrations with LoanPro. This was the most complex piece since we utilized many resources/features offered by LoanPro, and we had to reverse engineer the and implement our math work with and imply exactly what the scheduling breakdown math LoanPros was doing. Loanpro was tightly integrated through many components in our application from loan creation to financing. The integrations were over a proprietary HTTP API and **ElasticSearch**.

* **Sendgrid/Mailtrap**: Sendgrid added to GTL to offload email control to a managed service. The integration involved a massive system refactor to centralize all email all emails and develop a sub-system to have all system emails schedulable, mappable and integrated with sendgrid templates. An integration to send or capture emails for testing was also developed with Mailtrap. Boths integrations were implemented over an HTTP API..

* **BMO**: Finance and subsidize loan. Independent jobs were developed to send EFTs and poll for Returns. Integration was over HTTP

* **Google OCR**: This integration only went through a prototyping, but I developed a parser for vehicle ownership, vin and license plate data. The integration was over internal * HTTP.

Languages and DSLs I used for development within the GTL projects included: **Java**, **SQL**, **Bash, Dockerfile, Helm Charts, Kubernetes Manifiests, Jenkinsfile, YAML, XML/SOAP, HTML, JS, MarkDown, LogQL, Elasticsearch Queries and Postman Tests**

Libraried I used for development within the GTL projects included: **Spring Boot**, **Spring Data JPA, Spring Data Rest**, **Spring Web**, **Spring Security OAuth, Hibernate, Liquibase, Lombok,Thymeleaf, Swagger, Spring Boot Test, JUnit,, Mokito**, **Powermock** and **H2**

Resources I worked with included: **ElasticSearch**, **HazelCast**, **MinIO**, **MYSQL**, **RabbitMQ, Sentry** and **NGINX/nginx-ingress**

For DevOps and SRE I worked with **Grafana, Loki** and **Elasticsearch.**

Throughout my work on this project I also worked as a crucial member of the OPS team being a main member for a portion of time in parallel to working as an backend developer. Some of my part/responsibilities and initiatives within the OPS team were:
1. To introduce new **Spinnaker** pipelines for new microservice deployments
2. Add observability with by adding new application metrics and creating dashboards in **Grafana**
3. Setup **renovate** to keep software through the entire software up to date.

There was some work for the webapps but this was negligible with the most important work being debugging api/system misusage.

GoToLoans/WippyPay Loan ETL

Created and ETL to aggregate data from internal sources and LoanPro. This was used as a source for scheduled reporting. This was a separate project commissioned and developed that added on to the GoToLoans/WippyPay system.

My role was the system architect, sole developer and DevOps.

The application runtime was broken down into **short-lived jobs** which were implemented in and executed through **Docker** containers managed by **Kubernetes,** with **Airflow** as the scheduler and operator.

The language used was strictly **Java.** With libraries including **Sprigt Boot**, **Spring Batch**, **Hibernate**, **Lombok**, **Spring Boot Test**, **JUnit**, **Mokito, Powermock** and **H2**.

EMRchiver/Tekstack Healthpilot

Worked on an ETL and API for a webapp to observe and navigate patient health data derived from electronic medical records (**EMR**). Created multiple data pipelines to ingest, manipulate and reduced data from a **FHIR** stores for application development. This involved levels of normalization and sub-pipelines to resolve related data and make it accessible and navigability. Implemented data aggregation for searchability and tailored models with highly granular access and authorization. The data was aggregated to different databases optimized for efficient accessibility for application develpoment, searchability and time series collection, and exposed over different interfaces. Many proprietary libraries were developed for this project.

In this project I worked mainly as **DataOps** but I also designed and developed the api using both **Spring Boot** and **NestJs**. However I was also very invested in the **DevOps** provisioning and managing/configuring of infrastrcuture, resources (especially **Keycloak**) and **CICD** pipelines for all the different components of the application.

Managed **GCP** services I used and worked with for this project: **Compute Engine, Cloud Storage**, **Heathcare** and **BigQuery**

The resource stack I worked with for this project was compromised of: **Apache Kafka**, **DGraph**, **Opensearch**, **Keycloak**, **Redis**, **MySQL** and **Sentry**

Languages/frameworks used for the the libraries and applications compromising the project: **Terraform**, **Java** (**Spring Boot**, **Sprint Rest**, **JUnit5** and **Testcontainers)**, **Node** (**Typescript**, **NestJS**, **NX**) and **Liquibase**

And devops technology which I have personally added and managed in this project included: **Jenkins, GCP Cloud Build**, **Ansible and Docker/Docker Compose**

CBB Syndication

Work on cloud migration for a engine to aggregate vehile data from multiple sources. My worked involved full analysis, decomposition and proposal of a solution/architecture to make the existing software cloud ready. Refactored the software for the cloud; making it ready to be migrated to **Amazon EC2** and rewriting segments of the system to make use of cloud services. Environments were created and systems were added for development lifecycles.

I took on a lead role in this project and worked in cooperation with developers at CBB

Languages included **Bash**, **Python** and **Java.** And a **MySQL** instance was used.

With my work on this project I created a pipeline for development, moving the source code to **Bitbucket/Git**, setting up and moving dependencies to a **Nexus** repository, adding **Maven**, and adding **Jenkins** to build and deployment the application. I introduced a **Mailtrap** and **Amazon S3** integration to the code and optimized/refactored the **Python** code and **SQL** communication.

LEAD SOFTWARE DEVELOPER AND SYSTEM ADMINISTRATOR
DEBUT LOGIC INC. - READYPORTAL/TRIYOSOFT
2015 – 2018

Worked E2E on the Readyportal which is multi-tenant website and intranet platform. It allows for the construction of web portals via a WYSIWYG interface.

Starting as a front-end developer I progressed into full-stack development and later had the compounded responsibility of sole server-side developer and manager of the development team. I also adopted the Server Administrator role with the unforeseen passing of our at the time Server Administrator.

Frontend Web Development at Readyportal involved the technology stack of **HTML**, **CSS/SAAS**, **JS** with **Modular Design Patterns** and **Apache Velocity** as our templating engine.

There were over 15 client implementations I worked on, with the most notable ones being an intranet for ONS, web portals and cms for IEHP, Davey Tree, Fidelity and a workflow management system for Woodgreen. I was responsible for the full development lifecycle of 4. As a senior frontend, I created two standard template bases that served as the roots for all the preceding client implementation we did with Readyportal, and worked on the precursor of the Triyo product line.

The implementation of Triyo, a document collaboration system, within the Readyportal platform was the most notable of my front-end work as it was our first single page application. An SPA wasn't easy to achieve with the platform being designed as a static prerendered html webserver. A front-end engine and adapters were developed to strip, render and load partial content delivered as preprocessed pages parameterized with **Apache Velocity**, to the **AngularJS MVC.**

Serverside Development at Readyportal platform is primarily compromised of **Java/JEE**. The platform was developed with existing Java and using a proprietary DSL which compiled into Java. However was expanded with additional services written in **Spring** which provided a webdav and **Jackrabbit** integration, and an email and webhook interface written in **Golang**. The platform is very extensive and proprietary and provided a lot of technical debt, this required more comprehensive and meticulous testing during development. The product had around 50,000 users registered segregated into portals and intranets. The need to write adapters to the platform to expand existing client implementation was the business driver for the backend work I did. I worked onthe following projects/enhancements:

- * Upgrading and expanding exiting components from being static html to be controllable and rendered through the **Apache Velocity** engine. This allowed access customizable interaction with sever-side features in the expected security context and provided more fluidity and customization for individual clients/organizations within their portals.

- * The membership component was overhauled by me and this regressed into refactoring major components of the proprietary dao layer for query granularily. This was designed to be backwards compatible with the existing static user management and allowed for parameterization and customization of the user management interface per portal with the addition of expiry scheduling and user account locks.

- * Expanded the workflow component and created a sub-module that allowed external templating to inject server-side functionality through a configuration interface. This allowed for a huge level configurability and customization and was necessary for interfacing to external services for custom server to server integration per portal. This was specifically necessary for a business case that required data relay to an intermediate external api; our system authenticated to and pushed data to an api written under **TekStack** which intercepted, filtered and wrote the data with over a secure tunnel to a clients database.

- * Added **Email** and **Webhook** interfaces to certain portal components which was necessary for certain for the Triyo client integration. This was implemented through separate intermediate receiver/relay services that integrated with both the **Readyportal** platformand and the **JCR** database. The implementation utilized **Postfix** and **Apache HTTPD** acting as a reverse proxy and were written in **GoLang.**

System Administration was a major undertaking in 2016 as the company was left without a system administrator. I was responsible to bringing, recreating and migrating our entire product suite from Virginia to new servers in Toronto. This involved a lot of technical debt to overcome and many challenges due to the complexity of Readyportal. Readyportal was being moved from a managed cloud to self-managemed on physical hardware in a COLO and this provided a lot of additional work to compensate for the unmanaged infrastructure. I had to redefine the server breakdown/composition and networking on our physical hardware. A grid computing solution was implemented derived from what we had in the cloud but reduced to the hardware we had. I was responsible for both the physical installation aswell as the software system installation, setup and deployment. The software stack used in the implementation included **Bind** name resolution that played into the virtual hosting over Readyportal, **Qmail** for a list server, the Readyportal Platform itself which embed **Jetty** broken down into approximately 7 JVMS for balanced load (and distributed throughout multiple server blades), **Jetty Hightide** for additional services that expand the platofrm, **OracleDB** on a dedicated blade with a RAID5, and **CVS** for our development flow and **bash** scripts for our deployment pipeline.

SOFTWARE DEVELOPER
CYCLONE MANUFACTURING
2012 – 2013

Developed software to automate the quoting procedures at Cyclone Manufacturing.

Designed and wrote a management interface that produces quotes based on standard and custom business rules. The management interface replaced their current quoting department's need of Excel while providing all it's features that were being used. It was linked to a database supplanting their current quote data handling and integrated with existing software used at Cyclone.

I also wrote various scripts to increase productivity in rudimentary tasks being done by the quoting department and piped it to a database to move the work away from pen and paper, and physical filing.

The technology used here was mainly **.Net**, **C#** and **PowerShell**

PERSONAL PROJECTS

UNITY DESKTOP FOR SLACKWARE LINUX

Developed a collection of buildscripts to produce a package set for the **Unity desktop interface** on Slackware Linux: https://github.com/maciuszek/unity-slackbuild

The work involved a lot of debugging and patching of the Slackware package tree, with the goal being to integrate the software with a minimal deviation from the base system and minimal dependency stack. The package set was created with regard to Slackware's KISS philosophy avoiding complex dependencies and specifically systemd.

At the time and to my knowledge even now, this is the only package set of the Unity desktop ever made avaible for Slackware Linux.

*NIX PACKAGES

Maintain or previously have maintained packages for Slackware Linux https://github.com/maciuszek/slackbuilds and https://github.com/maciuszek/bashee_x86-slackbuild , Arch LInux https://github.com/maciuszek?tab=repositories&q=arch_&type=&language=&sort= and FreeBSD https://github.com/maciuszek/ports.

DISCORD BOTS

Forked and expanded an array of bots for discord https://github.com/maciuszek?tab=repositories&q=nsfg&type=&language=&sort=

TOOLS AND SCRIPTS

Throughout my career as a developer and user of *nix operating systems I have created an array of tools and scripts to help me along. Some I have made public are:

 * Scripts installation of Gentoo Linux https://github.com/maciuszek/setup_gentoo
 * Tools for CSGO Map Developement and Modification https://github.com/maciuszek/CMCST
 * Tools for Liquibase management https://github.com/maciuszek/liquibase-hash-checker-cli and https://github.com/maciuszek/liquibase-hash-checker-spring-demo
 * Converter for Arch Linux builds scripts to Slackware Linux build scripts https://github.com/maciuszek/pkgbuild2slackbuild

CERTIFICATION

Google Cloud Certified - Professional Cloud Developer

EDUCATION

Toronto Metropolitan University (formerly Ryerson University) – Bachelor of Science in Computer Science

REFERECES

Robert Golabek - CEO/Software Architect at Translucent Computing Inc - 416-XXX-XXXX

Rajiv Chatterjee - CTO of Triyosoft - 647-XXX-XXXX