MXXXXXXXXXXX@GMAIL.COM
(416) XXX-XXXX
TORONTO, ONTARIO

MATTHEW KUZMINSKI

A dedicated and detail-oriented software developer with approximately 9 years of working with 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 under **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.

EDUCATION

Toronto Metropolitan University (formerly Ryerson University)

Bachelor of Science in Computer Science

CERTIFICATION

* Google Cloud Professional Cloud Developer — https://google.accredible.com/0e8f8746-60ae-4aef-8c5c-fce3a3b668ca
* Google Cloud Professional Cloud DevOps Engineer — In progress
* Certified Kubernetes Administrator — In progress

EXPERIENCE

SENIOR SOFTWARE ENGINEER
TRANSLUCENT COMPUTING
2018 — PRESENT

In my tenure as a software developer at Translucent Computing, I have embraced a multifaceted role spanning the full spectrum of the application lifecycle. My journey has involved ideating robust software architectures, executing complex development tasks, championing **test-driven development**, and mastering **DevOps** practices. In action my day to day work was captured with both **TBD** and **GitOps** development flows**,** and under **Agile** management frameworks with both **Sprint** and **Kanban** used for different projects/purposes.

For application development I worked with: **Java**, **Spring Boot**, **Spring Data JPA**, **Spring Data Rest**, **Spring Web**, **Spring Security OAuth**, **Hibernate, Liquibase**, **Lombok**, **Thymeleaf**, **Swagger**, **Spring Boot Test**, **JUnit[4,5]**, **Mokito, Powermock**, **H2, Testcontainers**, **Terraform**, **Python**, **Node**, **Maven**, **Typescript, SQL**, **Bash**, **YAML**, **XML/SOAP**, **HTML**, **JS**, **Markdown**, **LogQL**, **Elasticsearch Queries**, **Postman Tests**, **NestJS** and **NX**.

Resources I worked with included: **Kubernetes**, **Elasticsearch**, **Opensearch**, **Hazelcast, MinIO, MYSQL**, **SQLite**, **RabbitMQ**, **Apache Kafka**, **Dgraph, Keycloak**, **Redis**, **Sentry**, **NGINX/nginx-ingress** and **Apache Airflow**.

DevOps and SRE technologies included: **Jenkins** (**Jenkinsfiles**), **Ansible**, **Docker/Docker Compose (Dockerfiles)**, **Makefiles**, **Nexus**, **SonarQube**, **Helm (Helm Charts)**, **Chartmuseum**, **Monocular**, **Spinnaker**, **Terraform**, **Atlantis, Kubernetes Manifests**, **Grafana**, **Loki** and **Prometheus**.

And cloud services were predominantly scoped to **GCP** and included: **Kubernetes Engine**, **Compute Engine**, **Container Registry**, **Cloud Build, Cloud Profiler**, **Cloud Trace, Cloud Storage**, **Heathcare** and **BigQuery**

To facilitate development for throughout projects I used **docker** and **docker-compose** extensively, **minikube** and **kind** for local **Kubernetes** clusters**,** and tools as **kubectl**, **helm** and **telepresense**. For local prototyping and testing I used **Postman** and **Newman**, **SoapUI** and **Curl**. And for debugging I used **visualvm**, **eclipse mat** and **[Chrome**, **Angular, Redux**] **DevTools**. Other tools and technologies I used throughout my work at TC included **jhipster**, **editorconfig, prettier, eslint** and **devcontainers**.

TEKStack

TEKStack is a platform and a set of core libraries used throughout the most projects I worked on at Translucent 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. Noteworthy work included:

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

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/WippyPay is a system 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 backend provided a multi-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 loans, or be commissioned through the loan. Calculators/algorithms and were written to determine loan payment breakdown and schedule based on defined pre-conditions and dynamic behavior with support for reconstruction.

I primarily worked on the backend by designing and implementing systems and APIs to facilitate and manage the loan lifecycles. The system architecture I worked with was decomposed into **microservices**, integrated through and managed by **Kubernetes**.

The system heavily leveraged third-party services with some notable integrations including

* CBB: valuate a vehicle and use the valuation as parameter for the loan quote. 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**.

* Vincario: Determine vehicle VIN failover. The integration was over REST/HTTP.

* 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 the scheduling breakdown math LoanPro 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. Both 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 Vision API: A parser for vehicle ownership, VIN and license plate data. The integration was over internal HTTP.

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. Some of my responsibilities and initiatives within the OPS team were:

* To introduce new **Spinnaker** pipelines for new microservice deployments
* Add observability with by adding new application metrics and creating dashboards in **Grafana**
* Setup **renovate** to keep software through the entire project up-to-date.

And additionally I did some frontend development where I worked with **Angular** and **NGRX**.

GoToLoans/WippyPay Loan ETL

I created an **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 and developer.

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.

TEKStack Healthpilot

I 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 development, searchability and time series collection, and exposed over different interfaces. Many proprietary libraries were developed for this project.

In this project I worked primarily as **DataOps** but I also designed and developed an API using both **Spring Boot** and **NestJs**.

I was also very invested in the **DevOps** provisioning and managing/configuring of infrastructure, resources (especially **Keycloak**) and **CICD** pipelines for all the different components of the application.

CBB Syndication

I worked on the cloud migration of data aggregation software for CBB. The software was designed to aggregate and store vehicle data from multiple external sources. It was imlpemented with using **bash**, **Java** and **Python** and my worked involved full analysis, decomposition and proposal of a solution/architecture to make the existing software cloud ready. I worked on refactoring 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

With my work on this project involved refactoring the existing code to make it portable and leverage cloud services where it can. To facilitate the migration 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 refactored the code by optimizing the **SQL** communication by introducing batching.

SOFTWARE ENGINEER AND SYSTEM ADMINISTRATOR

DEBUT LOGIC CANADA INC.

2015 — 2018

During my time at Debut Logic I worked on a single project called Readyportal, however the work involved many initiatives and responsibilities.

READYPORTAL and TRIYO

Readyportal 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 was primarily compromised of **Java/JEE** and build with **Apache Ant**. The platform was developed with both plain **Java** and a proprietary **DSL** which compiled into Java. However during my time the product expanded with additional services written in **Spring** and built with **Maven**, which provided a **WebDAV** interface and **Jackrabbit** integration. An email and webhook interface written in **Golang** were also added. 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 on the following projects/enhancements:

* Upgrading and expanding exiting components from being static html to be controllable and rendered through the **Apache Velocity** engine. This allowed 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 granularity. 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 client's 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** platform 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 be self-managed 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 onto 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 system installation and configuration, and application 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, Apache **Httpd** for reverse proxy to **GoLang** services, **Postfix** for mail relay to **GoLang** services and **VSFTPD** with a polling service. The Readyportal Platform itself had to be reconstructed which was deployed through **JVM**s with embed **Jetty** services distributed for balanced load throughout multiple server blades, **Jetty Hightide** for additional services that expand the platform, **OracleDB** on a dedicated blade with a **RAID5**, and **CVS** for development flow with **bash** scripts for the deployment pipeline.

SOFTWARE DEVELOPER
CYCLONE MANUFACTURING
2012 — 2013

At Cyclone manufacturing I was commissioned to develop software to automate the quoting procedures at Cyclone Manufacturing.

I 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 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 reduce toil of rudimentary tasks by the Quoting Department and piped it to a database to move the work away from pen and paper, and physical filing.

The languages I worked with during my time at Cyclone Manufacturing were **.Net**, **C#**, **CMD** 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 available for Slackware Linux.

*NIX PACKAGE

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 SCRIPT

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 Development 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

INTERESTS

LANGUAGES

I'm fluent in English and Polish, and I'm self-taught in Ukrainian and French on Duolingo. I practice on Duolingo daily where I hold a 1500-day streak
https://www.duolingo.com/profile/MattKuzmin

READING

I'm an avid reader especially of fiction and I track all of my reading at https://www.goodreads.com/user/show/71065261-matt-kuzminski

REFERENCES

Robert Golabek
Chief Architect & CEO at Translucent Computing Inc
+1 (416) XXX-XXXX

Stacee Ou Wai
VP of Product Management at Translucent Computing Inc
+1 (416) XXX-XXXX

Michael Lagowski
VP of Strategy & Product Development at TEKStack Health
+1 (416) XXX-XXXX

Rajiv Chatterjee
CEO and Co-Founder of Triyo
Previously CEO at Readportal
+1 (647) XXX-XXXX

Ugur Poyraz
Previously Product Manager at Readyportal
+1 (647) XXX-XXXX

Lukas Arent
Previously Director of Business at Readyportal and Triyo
Previously Program Manager at Cyclone Manufacturing
+1 (647) XXX-XXXX