

# Curriculum Vitae

Mohamed Bana

Contact	Web
+44-20-7097-4635	<a href="https://bana.io">https://bana.io</a>
+212-624-846-935	<a href="https://github.com/banaio">https://github.com/banaio</a>
<a href="mailto:mohamed@bana.io">mohamed@bana.io</a>	<a href="https://linkedin.com/in/mbana">https://linkedin.com/in/mbana</a>

I am a well-rounded Software Engineer, with over 10 years of Software Development experience. I understand what it takes to release a product having worked on the back-end, front-end, testing and finally the deployment aspect of several products.

Please read my [Cover Letter](#).

## Work Experience

**Senior Software Engineer (Remote), ViTRiFi Limited, 6th Floor 33 Holborn, London, England, EC1N 2HT, UK - 01/11/2021–03/03/2022 - Contractor**

Delivery of a log aggregation pipeline on Amazon EKS for real time log analysis:

- Go module/library/binary development.
- API design and implementation.
- Architectural component definition and build.
- Deployment of code to AWS platform via CI/CD pipelines.

Cortex/FirstLight:

- Build up the functionality of the Cortex Portal.
- Work on the interface of the Cortex Portal
- Work on the interface of Cortex Core.
- UI design and implementation.

**Tech:** Golang, Kubernetes, Docker, Docker Compose, AWS, Amazon EKS, Grafana, Loki, Prometheus, Shell scripting/Bash, Visual Studio Code Remote - Containers, gRPC, Protocol Buffers, Kafka, Redpanda Kafka, GitLab.

**Golang Engineer (Remote), Paymentsense Limited, The Brunel Building, 2 Canalside Walk, London, W2 1DG, London, UK - 05/07/2021–31/10/2021 - Permanent**

Worked as a Golang Engineer on Connect-E (<https://docs.connect.paymentsense.cloud/ConnectE/GettingStarted>).

**Tech:** Golang with modules, Docker, Docker Compose, TypeScript, GCP, Google Cloud Datastore, Google Cloud Big Query, Google Cloud Pub / Sub.

**Remote Cloud Software Engineer / Golang, IBM United Kingdom Limited, Hursley Park, Winchester, UK - 14/09/2020–30/04/2021 - Contractor**

I worked on IBM Cloud as Software Engineer on the IKS Cluster (IBM Cloud™ Kubernetes Service) team as Cloud Software Engineer / Golang Engineer. IKS is effectively something like AWS, GCP or Azure, see [IBM Cloud](#).

- I'm a member of the IBM Cloud™ Kubernetes Service team responsible for delivering IBM's Kubernetes managed container service. As a certified K8s provider, IBM Cloud Kubernetes Service provides intelligent scheduling, self-healing, horizontal scaling, service discovery and load balancing, automated rollouts and rollbacks, along with secret and configuration management and a fully managed image registry with integrated vulnerability scanning.
- Working in an agile way and operating with a continuous delivery model.
- Team/Squad consisted of around nine (9) people and we managed the complete life cycle of deliveries.

**Tech:** Go, Golang, Shell Scripting, Bash, Docker, Docker Compose, Kubernetes, RedHat, [Travis CI](#), [LogDNA](#), [go.uber.org/zap](https://go.uber.org/zap), [etcd](#) or [What is etcd?](#) | [IBM](#).

## Full Stack Go and Vue.js Developer, **Open Banking Limited**, St Katharine's & Wapping, London, UK - 08/05/2018–01/01/2020 - Contractor

Working as a full stack developer at Open Banking on a tool that will validate a bank's implementation of the OpenBanking API spec, see:

- <https://bitbucket.org/openbankingteam/conformance-suite>
- <https://hub.docker.com/u/openbanking/>

**Tech:** Go, Golang, Node.js, **Vue.js**, **Vuex**, Jest, Docker, Docker Compose, Kubernetes, **OpenID Connect**, **JSON Web Token (JWT)**, **Kompose**, **CircleCI**, **Swagger**, **WebSocket**, **Bitbucket Pipelines**, **OpenAPI 3.0**, <https://bitbucket.org/openbankingteam/conformance-suite>, <https://hub.docker.com/u/openbanking/>.

## Senior Engineer, **90POE - Ninety Percent of Everything Limited**, Marble Arch, London, UK - 01/02/2018–20/04/2018 - Contractor

Worked at startup specialising in software that runs on ship on two projects that were heavily Go-based.

**platform-document-storage-service:** Document storage and retrieval to be used by others services, so it's a core service. The core of the service was written in Go and exposed using **gRPC** and http using **go gorilla/mux**. Both write and retrieve supported arbitrarily large files which was achieved using **gRPC** unidirectional streams. The underlying store was MongoDB's **GridsFS** which I interfaced with using the Go driver **mgo**.

**auditing:** Currently working on this. The service is structured very similar to the preceding in that the underlying service is exposed using **gRPC** but the top-level interface is done using **GraphQL**. I wrote the GraphQL server in go using **graphql-go**. The underlying store is in Postgres and the library I used to interact with it is **GORM**.

**Tech:** Go, Golang, **gRPC**, **go gorilla/mux**, Protocol Buffers, **GORM**, MongoDB, **MongoDB GridsFS**, **mgo**, **GraphQL**, **graphql-go**, Docker, Docker Compose, Kubernetes, NodeJS, Jest, **Concourse CI**, Postgres.

## Full Stack Developer, **Root Capital LLP**, London Bridge, London, UK - 09/10/2017–24/12/2017 - Contractor

Worked as a full stack Node.js developer on the **Minds for Life** application, mainly on the forum.

### Frontend:

- **react**, **react-redux**, **react-boilerplate**.
- Single Page Application (SPA) targeting mobile platforms.
- ES56 using most of the latest ES56 features; **async**, **await**, **classes** etc.
- Serverless and hosted on **Amazon S3** as static assets, with **Amazon CloudFront** as the CDN.

### Backend:

- NodeJS server written in ES6, like the frontend.
- **Koa**.
- MySQL as the datastore, using the **Knex.js** library.
- Packaged as a set of docker containers.

### CI, Devops and Infrastructure:

- Services were packaged as containers. Used **docker** and **docker-compose** to start them.
- Builds managed by **Semaphore CI** and **Wercker**.

**Tech:** JavaScript, ES6, Node.js, **react**, **react-redux**, **react-boilerplate**, **Webpack**, **Koa**, **Knex.js**, **Sequel Pro**, **Docker**, **Docker Compose**, **Kubernetes**, **Semaphore CI**, **Wercker**, **Amazon S3**, **Amazon CloudFront**

## Full Stack Node Developer, **Lloyds Banking Group PLC**, London Bridge, London, UK - 20/03/2017–22/05/2017 - Contractor

### NodeJS - JavaScript:

- **Loopback** for server-side of the code.
- ES5/6-based code base.

### Monitoring/Devops/Misc:

- **Splunk** and sending logs via (<https://en.wikipedia.org/wiki/Syslog>) a **LogDrain** service available on **Bluemix**.
- **Gerrit** for managing code. CI:
- **Jenkins**: Configuring, managing and installing.
- **Jenkins 2**: Same as previous plus writing pipeline scripts.

**Tech:** JavaScript, Node.js, ES5/6-based code base, **LoopBack** for server-side of the code, **SysLog**, **Splunk**, **Jenkins**, **Jenkins 2**, **Gerrit Code Review**, **IBM Bluemix**

## Senior Front-End Engineer, **Synthace Ltd.**, King's Cross, London, UK - 13/04/2016–04/11/2016 - Permanent

Did a fair amount of architectural UI work:

- JWT-based authentication: Implemented most, if not all, of the authentication related UI features. Polymer didn't have an authentication module as it's fairly new requiring me to re-implement this feature.
- API interactions: I introduced Swagger JS and did the conversion from plain XHR to Promises, and ensured API was in-sync with the state of the authentication.
- Updates via the Web Socket for notifications and async task updates: STOMP Over WebSocket.
- Bootstrapped the testing using Web Component Tester.
- Deciding on the build, test and hosting strategy, e.g., hosting our own CDN using Azure.
- Performance: 1) pushed to have HTTP/2 enabled, and prototyped, on our custom server written in Go, 2) implemented lazy-loading of our Web Components which are included using HTML Imports, 3) Significantly improved UI build scripts; went from a somewhat un-deterministic build to one that almost always runs.
- Introduced ES6 to the code-base, and moving to defining Polymer elements using ES6 classes.
- Misc: libraries/utils to ease UI development.

We deploy Docker images to our Kubernetes cluster running in Azure using:

- Docker: Fairly comfortable using this.
- Kubernetes: I've done deployments of dev branches, so I understand the deployment model, navigating the Kubernetes dashboard and crude command line interactions, e.g., port-forwarding of the service the pod is running from the cluster to the local machine.

I'm finishing up on adding support for our language Antha to Monaco Editor, the editor that powers Visual Studio Code.

Since this is a startup I have done a fair amount of work and I have been given a fair share of responsibility, more so than any of my prior roles.

**Tech:** JavaScript, ES6, **Polymer Project**, **Web Components**, TypeScript, VSCode, Azure, Docker, Kubernetes, Go, **Polymer Web Component Tester**, Git, Swagger.

## Developer, **IG Index Ltd.** Cannon Bridge, London, UK - 09-2014–18/02/2016 - Permanent

### JavaScript Developer

- *Price & Indicator Alerts:* My main responsibility was handling the UI aspect—setting, configuring, triggering—of the various Alerts we support, from the basic Price Alert to Technical Indicators such as RSI and Moving Average. **Tech New:** Modern UI powered by ES2015 (Babel), Ember.js, Handlebars and Less. The UI was then composed of isolated and reusable Ember components. Runs on a NodeJS server, managed with npm and bower, and version controlled using Git. Integration and unit tests written in Mocha then run using Karma. **Tech Old:** Vanilla JavaScript using an in-house framework when changing the previous UI.
- *Charts:* Assisted in the conversion of the Adobe Flex Real Time Charts to an SVG-based version. **Tech:** d3 and tested like above.
- *Deeplinking:* A hashed action link, like typical deeplinks, that we send to our Clients which then launches the mobile IG app, or directs them to the app store for the device with the IG app pre-selected. Upon login the action is carried out automatically. I did the bulk of the work with the team lead overseeing it. **Tech:** Java 8, Spring and acceptance tested using Cucumber. Redirecting and launching of the IG app was done using vanilla JavaScript.

## Software Developer, **ITRS (International Trading Room Software) Group Ltd.** Moorgate, London, UK - 02-2010–09-2014 - Permanent

**JavaScript (UI) Developer, 01/2014–09/2014** UI for the next generation of the product which is built around, loosely speaking, a real-time distributed analytics store. The aim is for the old system to stream data to the new system so we can provide all the great visualizations available in the HTML ecosystem, which was not achievable in a reasonable amount of time in Swing.

The code is entirely modularized using RequireJS so we can test each viewmodel without creating a view, we then test the entire UI (end-to-end tests) using WebDriverJS.

Some Java/C coding required to write NodeJS bindings to interact with the store. We evaluated several frameworks, Angular, Batman and KnockoutJS, by writing prototype applications that connected to our backend for a period of roughly 4-6 months before we chose to settle on Knockout.js.

**Tech:** JavaScript, NodeJS, Node-WebKit, Durandal, KnockoutJS, RequireJS, Git, Jasmine, Protractor (WebDriverJS), Jenkins, Bower, HTML5, CSS, LoopBack.io.

**Java (UI) Developer - 07/2012–12/2013** Worked with two developers and one QA member on a new Swing UI, ACLite, that uses our new streaming-based API to access Geneos data, for more info. see <https://resources.itrsgroup.com/OpenAccess>. We access data from a fault-tolerant Cluster that is built on a set of distributed nodes. The system we coded against is somewhat similar to the Amazon distributed key/value store, DynamoDB, except with support for streaming, so I am familiar with dealing with distributed systems.

**Tech:** Java, Maven, Swing, Eclipse, Jenkins, Git, Vagrant.

**C++ Developer - 02/2010–06/2012** Spent one year with Run The Business (RTB) team, a team set-up to fix critical bugs that Customers encounter. A very challenging role which requires all-around product knowledge, good debugging skills and being able to liaise with our Support staff in dealing with the Customers. Prior to this I was one of three developers working in the Transactions and Latency Monitoring team (part of the backend team) doing core C++. We wrote and maintained the following plug-ins that are part of the Geneos suite:

1. **Geneos FIX plug-in:** Monitors FIX (protocol) messages.
2. **Feed Latency Monitoring Plug-In:** Monitors feeds and calculates latency of instruments and fields across the monitored feeds.
3. **Latency Monitoring - Message Tracker FIX adapter:** Tracks, generally, FIX messages across several checkpoints.
4. **Market Data Monitor:** See below. And bespoke plug-ins written for specific firms.

We also maintained several non-finance specific plug-ins. I ported another bespoke plug-in called Price Latency Monitor (provides latency figures for bonds) to MS VC++ when I worked on this team. Projects:

- I converted the Windows version of the entire product suite from Visual Studio 2005 to 2010.
- I wrote the Market Data Reliability plug-in. This plug-in connects to the Patsystem's Trading API (<www.patsystems.com>) to monitor commodity prices, using their C API, to determine if prices are 'stale'.
- I ported a significant part of our product to Solaris x86-64 (64-bit non-sparc architecture).

**Tech:** C++, STL, Boost, Visual Studio, Linux/Unix, GDB, DBX, Make, Configure, XML, XPath, CPPUnit, **Financial Information eXchange (FIX) protocol**, **Geneos FIX plug-in**, **Feed Latency Monitoring Plug-In**, **Latency Monitoring - Message Tracker FIX adapter**, **Market Data Monitor**, **Geneos PATS-STATUS Plug-In**, **Patsystem's Trading API**.

**Software Developer Intern, then Tester, Thomson Reuters. Canary Wharf, London, UK - 05-2009–11-2009**

**C# Developer** One of four developers working on a search and navigation interface to Global Product Search. Full life-cycle of development; requirements engineering, analysis and design to implementation, testing and deployment.

**Tech:** C#, Silverlight 3.0, MS SQL Server 2005, LINQ, Web Services (WCF), XML and Visual Studio 2008. I handled deployment using CruiseControl.NET.

**Tester** User Acceptance Testing of the latest release of Thomson Reuter's 3000 Extra, then called UTAH, now called Eikon. UTAH combines the data from Thomson and Reuters. My primary responsibilities were to validate the end product against pre-defined requirements/workflows. 1. Worked on Thomson Reuters project UTAH as part of a large team. 2. Tasks included testing, observing, documenting software bugs, issues and errors before final release of Utah. 3. Testing was done over multiple iterations.

## Education

- **2005-2008:** BSc Computer Science (2.1), City, University of London.
- **2008-2009:** MSc Software Systems Engineering (Attended), University College London, and Trading & Financial Market Structure module, London Business School.

## Additional Information

### Misc

- **Passport/Nationality:** I have a British Passport.
- **Drivers Licence:** Full UK Driving Licence.
- **Background/Criminal Check:**
  - **Basic Disclosure and Barring Service (DBS) check:** **Date of issue:** 31/07/2021 (13th July 2021), **DBS ID Number:** P0000GBYC03, **Certificate number:** 001023825352:  
Police Records of Convictions, Cautions, Reprimands and Warnings:  
**NONE RECORDED**
  - **Police Certificates - ACRO Criminal Records Office:** **Date of issue:** 08/03/2021 (8th March 2021).  
Summary of convictions and reprimands/warnings/cautions/impending prosecutions/under investigations held on UK police databases and disclosed in accordance with the ACRO stepdown model:  
**NO TRACE.**

### Languages

- **English and Swahili:** Native.
- **Arabic:** Intermediate. I own an apartment in Marrakech, Morocco. I have lived in Cairo, Egypt and have travelled several times to the UAE.