A white paper with black writing

Description automatically generated

**AI POWERED CHATBOT**

The AI powered chatbot will be used to increase efficiency of the customer service side of the company. It will decrease waiting times for customers, and relief the human representatives of more work. However, for it to be effective to its goal and prove useful to the users, it must be built accurately. When I say accurately, I mean the goals must be constantly reviewed and up to date, so the model that is created for the AI chatbot is built upon the correct data that is relevant. Aswell as creating an accurate product, it must also abide by the many strict standards of laws in the UK when it comes to handling sensitive information (such as the Data Protection Act 2018 or the General Data Protection Regulation). This is especially essential due to the potential sensitive information that customers may omit to the AI chatbot, and any guidance or advice the AI may provide must be meticulous.

I have chosen the widely popular ‘CRISP-DM’ (Cross-Industry Standard Process for Data Mining) methodology. This is due to it being data driven and is aligned with the business’s needs.

The original CRISP method does not contain Continuous Delivery and Integration (CD/CI), since this is found in more DevOps methodologies. However, it is possible to integrate CD/CI practices to include robust and efficient deployment of versions of the AI chatbot. This can be done be enhancing the phases (not adding new ones) with automation, live feedback/ monitoring and pipelines. For example, when creating the model in the ‘modelling’ phase, it will be automatically trained with the dataset. This will dramatically improve the production time and response to improvement suggestions.

Though security Integration is not explicitly addressed in the standard framework. It can be easily established in the existing phases. It will largely consist of securing the data that is being collected and used to test models. This will be achieved through encrypting the data, and rigorously testing attacks on the system (such as penetration testing or simulating a Denial-of-service attack).

Overall, this lifecycle with a few easy implementations would be an incredibly effective method on developing an AI chatbot.

A diagram of a diagram

Description automatically generated with medium confidence

**RPA (Robotic Process Automation) system**

The RPA system will be used for simple, repetitive tasks (in this case such as invoice processing or payroll management) that could be achieved by a human. However, this system will have the ability to save hours of manual labour, reduce manual errors and derive conclusions from analysing large datasets. The tasks are simple, yet undeniably important for the company to function and for the customers to continue their daily lives. For this is handling their finance. Therefore, we need a development lifecycle that can produce a version of the system in a timely manner, with the opportunity to adapt and scale up the system and act in accordance with all legal requirements. Security of the system must be one of the top priorities when developing, due to the sensitivity of the data it will be handling.

I chose a DevSecOps diagram to represent the development of the RPA (Robotic Process Automation) system for Multiple is due to security vulnerabilities being identified early, rigorous testing and as well as a fast delivery procedure to release new versions as fluently as possible.

Due to this system automating basic functions for a financial service company, it is essential that updates of the system are processed as smoothly and efficiently as possible to avoid hindering current operations. This enables Continuous Delivery by using tools such as Gitlab or Azure DevOps. This is done in the later stage of the development lifecycle (the Deploy phase). By now it would have been thoroughly tested and theoretically should not cause any negative affects on live procedures.

The automated security tools like the SAST (static Application Security Testing) and Code + Infrastructure scanning found in the Build phase, or the DAST (dynamic application security testing) found in the test phase allow consistent security standards across the project.

This methodology also allows firsthand feedback. For it can have a version of the system working live, handling actual customers and actual data in the company. This can be monitored and reviewed, and suggestions can be delivered back to the development team to implement in the next version. Constant feedback loops like this can also mitigate security vulnerabilities and Incidents (like the system going down).

A close-up of a paper

Description automatically generated

**Mobile Banking App**

For the Mobile Banking app, I have chosen to use an Agile method lifecycle. This is due to its high levels in iteration, flexibility, collaboration and frequent delivery of high values.

The most notable characteristic of the Agile method is its small increments (also known as sprints), this breaks down the overall goal into day to day aims. This allows the team to deliver functional parts of the product sooner. This makes medium and large sized projects (such as a banking app) much more manageable and allows insight to the final product to all stakeholders. It also can allow much more frequent updates to a live version that has been previously deployed. This is essential for a mobile app. The reason for the app is to increase accessibility to customers, therefore updates for improvements and bugs must be swift and smooth.

The Agile method does not make Continuous Delivery and Iteration mandatory; it can easily be practiced. During the sprints, pipelines can be put in place to automate testing of new code and prepare them to be deployed. This is most beneficia since in an Agile method, there are numerous codes commits in every iteration. Additionally, the Agile method depends deeply on customer feedback for direction of improvements or adjusting their objectives.

For Security implementation, this begins at the concept phase. Here you can outline risk assessments and steps to take to comply to any laws (such as the GDPR). During the execution phase (where the software is developed), checks such as the Static app security testing and dynamic app security testing (SAST and DAST) can be implemented. Once it released you can continue live monitoring of traffic. This will allow you to identify anomalies and possible security threats. Security audits and penetration testing will also decrease risk to the system.

A diagram of a diagram

Description automatically generated

**Zero Trust Diagram**

The three applications will have three levels of security, the AI chatbot being the lowest and most accessible, while the RPA system will be the strictest to access. This is due to the data the applications have access to, and the functionality they have on said data. By applying the principles ‘assume breach’, ‘use least privilege access’ and ‘verify explicitly’, this architecture ensures robust security.

The users are split into 3 primary sections. Mobile customers, non-mobile customers and non-customers, and thirdly Employees. The Mobile customers will be able to access the mobile app and can have increased security through passwordless verification (such as fingerprint and face authentication). All users would have access to the chatbot, employees, customers and non-customers. The devices they use is irrelevant therefore the authentication required for access to this application is the least critical. The third type of user (Employee) would be the only user to be able to access the RPA system. This would be through levels of authentication. Such as multi-factor authentication, their identity being granted though the IAM, and additionally device authentication. It is standard practice for companies to have authorised devices that employees must have to access higher graded applications.

The Policy Enforcement point (PEP) and the Policy Decision Point (PDP) are standard sections of any Zero Trust Diagram (ZTD). The PDP evaluates the access requests, while the PEP applies the access controls. The PEP grants the user level of access to the applications.

The data that is inserted in the Mobile Banking App should be encrypted for increased security. This is in case of a data breach to the database, however if the data is encrypted, it would be no use to the perpetrator(s).

The diagram also has a ‘Security control zone’ where in the network, zero trust principles are withheld and enacted upon the rest of the network, for example the Role-base access control (RBAC) will be crucial for the RPA. It will allow the system access to specific datasets, not the entire database. This is a key principle in the Zero Trust philosophy. To minimise access as must as possible at all times.