**1.0: PROJECT INTRODUCTION**

**1.1: Purpose**

The purpose of this document is to learn and understand all aspects to creating our service. The main objective is to give many detailed descriptions and diagrams of an artificial intelligent chatbot with unique interface. Furthermore, this document will cover theoretical practical application of it including a feasibility study. Lastly, it’ll cover how exactly we will solve a solution, aswell as analysing the threats, opportunity and potential of it all.

**1.2: Scope**

This document is meant for use by the developers and will be the basis for developing the final delivered system. The requirement changes to the document can be made due to changes in the client needs, change in technology, new updates in the chat extensions etc.

**1.3: Project Overview**

The application is a service in the form of an AI discord bot with an improved UX. The AI bot should provide a unique user interface, aswell as provide other services such as:

* Answering Messages
* Creating Content
* Hiring people in different time zones
* Improve how users experience AI with blockchain, design and gamification.

**2.0: SYSTEMS ANALYSIS AND FEASABILITY STUDIES**

**2.1.0: Systems Analysis**

*Descriptions in natural language of functions, services, and operational constraints.*

* This bot should provide a correct answer instantaneously when asked a question.
* This bot should recognise what the different time zones are and should contact them accordingly.
* This bot should be bug free.
* This bot should realise the difference between (morally) right and wrong.
* This bot should continuously learn the right things about the right project.
* This bot should always be active and running so it can learn.
* This bot should have a good UX.
* This bot should have its own "personality" (from a data set).
* This bot should be able to differentiate between users and treat every account separately.

*Are they Valid (VA), Consistent (CON), Complete (COM), Realistic (RE)???*

**2.1.1: Conditional Analysis**

System analysis will be performed to determine of it is feasible to design based on DARK CITY idea, fit user requirements and eliminate weaknesses of the present system.

**2.1.2: Existing System**

There are many existing chat-bots that already exist for discord. Here are some disadvantages on what we need to look out for. These include:

* A lot of AI chat-bots and A.I. copies already exist, meaning there will be a competitive market.
* We will have to rely on other AIs. Changes in them could cause setbacks.
* VERY low funding.

**2.2.3: Proposed System**

This project has been aimed at developing an AI Discord bot that works with crypto and Web 3.0. This can be accessed and downloaded off the internet, more specifically, our DARK CITY website. It provides the primary function of offering a multifunctional discord bot using AI with an improved UX. It’ other functions include. Regarding the AI side of it, it uses clustering and contextual analysis in order to interpret the information given and earn with time.

A chain of components are needed to implement this idea:

* Discord to chat-bot API
* PyMongo to chat-bot API
* Chat-bot API to A.I. algorithm

If all of this is implemented, this would create a perfect system for the chat-bot to do the following:

* Receive requests
* Process requests
* Creating content
* Hiring people in different time zones

The bot’s advantages include:

* Good UX, not cold looking like the other ones
* Very personalised to users
* A.I. will learn about you as a person

Our reasoning for trying to get into this market (solutions) is as follows:

* Most existing bots really only focus on sales corporate, therefore discord bots only really focus on social media management.
* Many discord bots lack personality.
* Many discord bots do not focus on Web 3.0 and NFTs.
* Many 3D and AR chat-bots are mainly used for tours.
* Building templates are VERY scalable.
* No NFT project is really doing anything with AI bots.

**2.2.3.1 Functional Requirements**

*This is what the system should do. These may include the reaction to specific scenarios & data specifications.*

1) In the case of a negative sentence (e.g. burn all pakis), the bot should realise that "paki" is a bad or offensive word and ignore the request. It should then:

- Give a warning to the user if first offence

- If second offence, warn them again and add 1 to “warning counter”

- If 3rd offence, block their requests for 2 weeks

2) In the case of a new user, the bot should aim to learn key fundamentals about the new user.

This may include:

- Name, Age

- Where they reside, their time zone

- Their hobbies and interests (to tailor (job?) opportunities to them)

3) In the case of a job request for 2 people or setting up meetings, the bot should look at the data stored about each user and calculate when to send a request. E.g. If the meeting is at 7AM EMT, time zone calculations should be done to make sure the meeting happens at the same time for all.

4) In the case of "being told" there has been a project update, it should react accordingly. E.g. for DARK CITY the bot should constantly reference more about the new update (which could be a new AI) rather than focusing on the existing website.

**2.2.3.2 Non - Functional Requirements**

*These are global statements on the system. They are not directly concerned with specific services to users.*

* System should be secure.
* Private data should not be accessed by anyone unauthorised.
* UX should be clean.
* Bot should be VERY EASY to set up (assuming the avg user is a dumbass).
* Cost of the software should be made under £200 (everything included).

**2.3: Feasibility Study**

**2.3.1: Economic Feasibility**

This application / service may have a great impact on its relating organisations. Expenses must be justified, although there isn’t much that can be financially injected into the project at this current stage. The proposed system can be developed within the budget, and this was obtained as most technologies used are free.

**2.3.2: Technical Feasibility**

This application / service will not have a high demand on technical resources. Being primarily composed of commercially small, efficient algorithms, there will not be high demands placed on the clients / customers of the service. Furthermore, it should be able to run efficiently as a discord add-on, meaning it will run smoothly and efficiently.

**2.3.3: Social Feasibility**

This application / service is very feasible on a social scale. The main extent of this is regarding training the user to use the service efficiently. The user will not be threatened or confused by the system, but is expected to embrace, mainly it due to its UX. Our project is by far the best alternative of discord chatbot, in the regards to A.I., UX, and more. There is no need for user training as everything will be self-explanatory and user friendly, which opposes the large learning curve in learning new technologies in today’s world.

**3.0: REQUIREMENTS SPECIFICATION**

**3.1: Hardware Configuration (For Development)**

* Processor: MIN 4th Gen Intel Core I3, 2.2 GHz +
* RAM: MIN 4GB
* Hard Disk: MIN 500GB
* Monitor: Generic PnP Monitor

**3.2: Software Configuration (For Development)**

* Operating System: Windows 8.1 +
* AI Algorithm Language: Python
* Database Language: MongoDB
* UX Languages: HTML, CSS, JavaScript, React
* Documentation: MS Office
* IDE: IntelliJ, VSCode

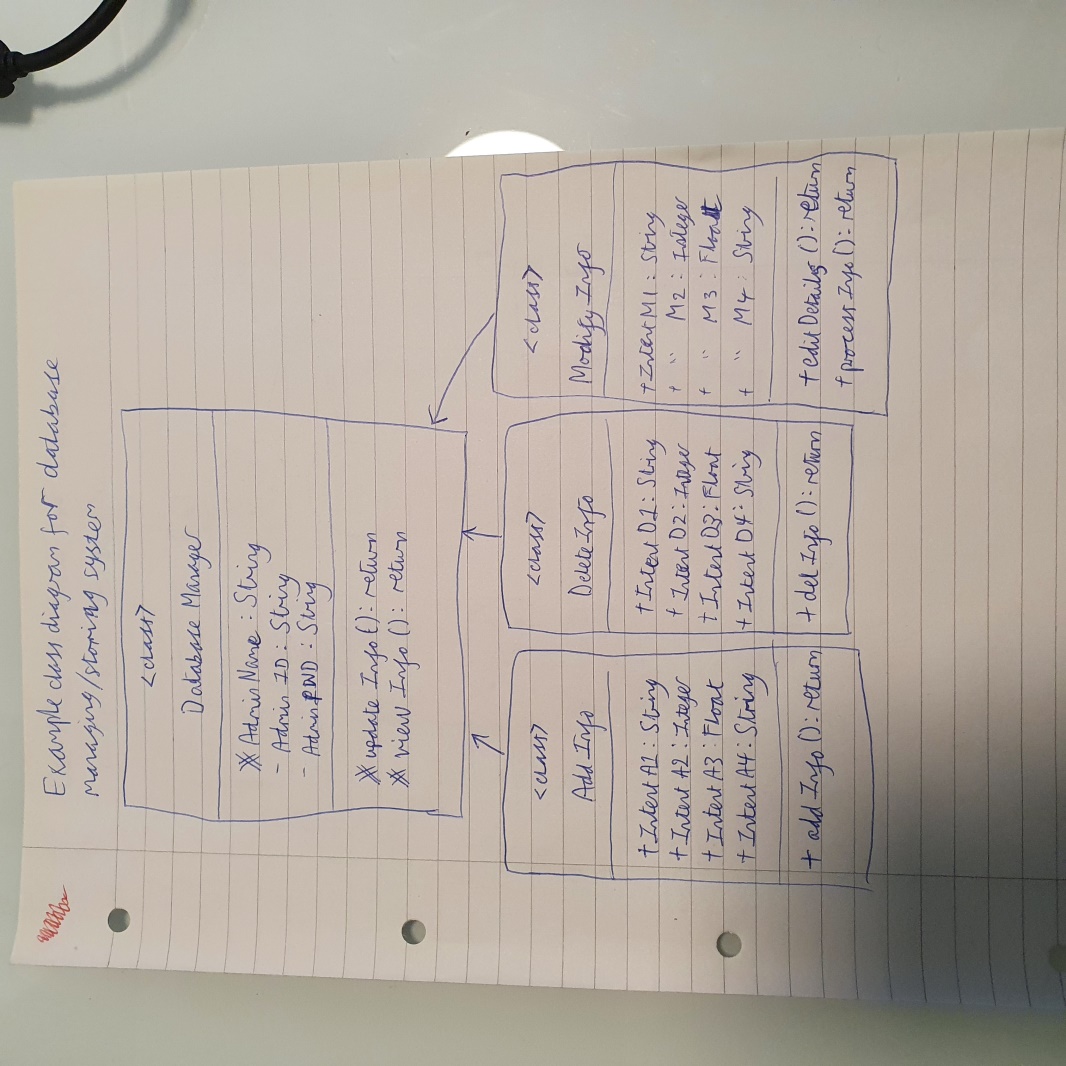
**3.3: Deployment System Configuration**

**\*need to talk about cloud platforms here\***

**4.0 SYSTEM DESIGN**

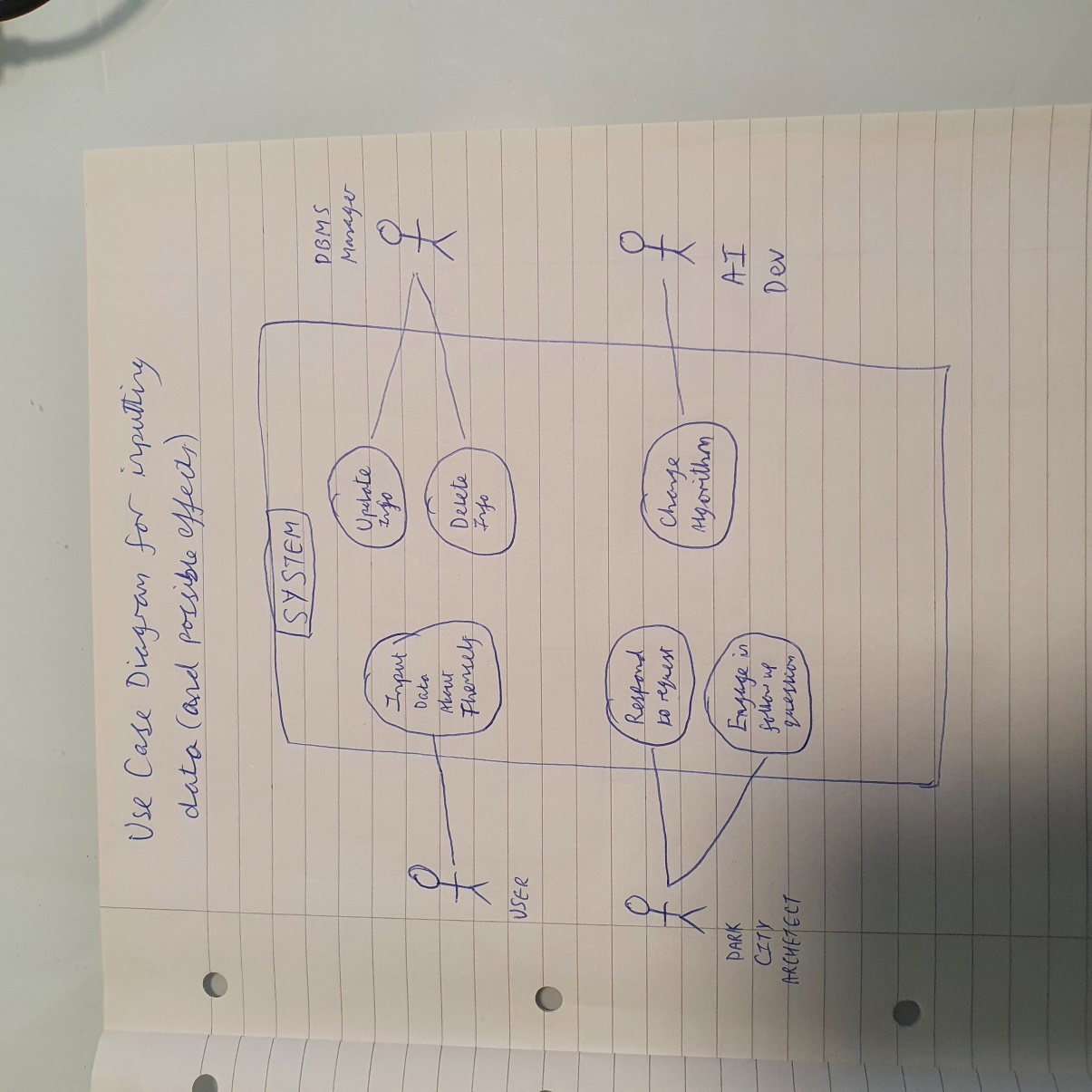
**4.1 System Architecture (needs to be done)**

**4.2.1 Class Diagram (Database)**

****

A class design shows a set of class interfaces and collaborations in the static view of the system. From the diagram above, we can identify at least 4 classes that may need to be made in the (potential) SQL database. The diagram also shows the relative links between the subclasses and super classes, as well as example data (with the data types) we may need to be conscious of.

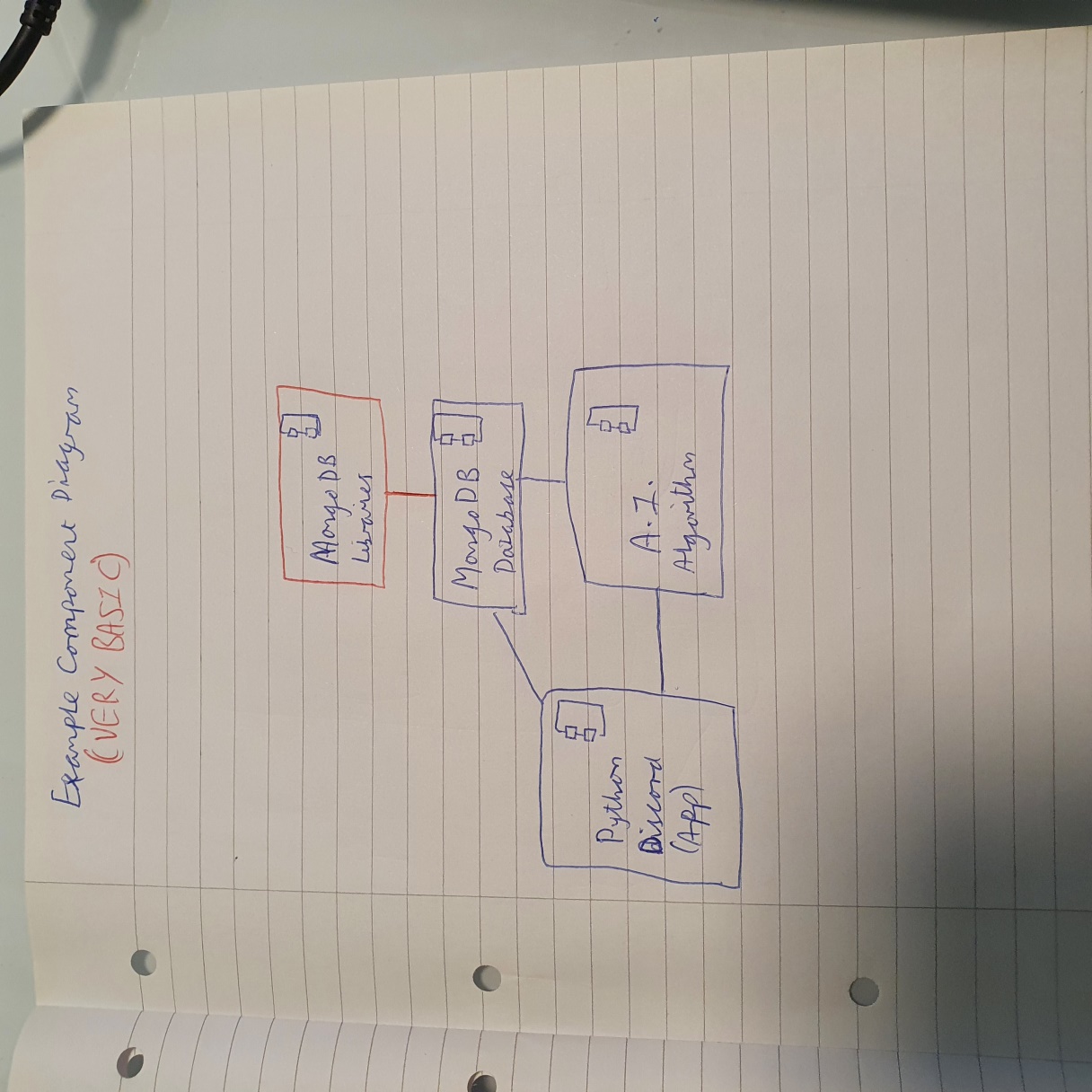
**4.3.1 Use-Case Diagram (Data Input)**

****

This case diagram shows the collection of cases and actors in their relationships. In this situation, the actors are USER“, “DARK CITY ARCHETECT”, “DBMS Manager” and “AI Dev”. This diagram shows each actor’s most common action, and what they are. From this, we can determine on what actions will influence each other. E.g. when the USER inputs data themselves:

* DARK CITY ARCHITECT will respond to the request AND engage in follow up questions.
* DBMS Manager will need to update the stored data OR delete it.
* In the long term, the AI Dev may need to change the algorithm to suit it better to user responses.

**4.4.1 (Component Diagram)**

****

A component diagram represents the organisations and dependencies among a group of components. At a very simple scale, the primary main communication between components will be between the MongoDB Database, the Python Discord App and the A.I. Algorithm. I have also showed individual relations. In our case, we have used some MongoDB Libraries, which only communicates with the MongoDB Database. We are using PyMongo add on, which is not written down in the diagram.