University of Sunderland

Name: ROTIMI, Ayodeji Moses

Module Code:

CETM50 - Technology Management For

Organizations.

STUDENT ID : 229783515

ORCHID TECH INNOVATIONS REPORT

Preamble

This report covers the investigations, findings and recommendations of an inquisitive inquiry into the data analytics and engineering strategies of Orchid Tech Innovations Ltd, a small to medium business based in a single office in the United Kingdom. Their expertise spans across digital marketing, analytics, customer relationship management which they offer to businesses and high profile individuals.

Orchid Tech Innovation Ltd customer's data is fragmented across various off-the-shelf systems and because of this, requires manual collation to get a full overview of customer information. The business is multifaceted and customer data is used to cater to different business functions, notably finance and HR. These business functions work independently and in silos and as such, requires manual intervention in order to bring them together for comparison during data sharing between the different business functions.

In the wake of a highly successful financial year, marked by a huge increase in their customer base, Orchid Tech Innovations Ltd., is seeking to explore ways to capitalise on their growth. Due to the prevalence of social media marketing, they are looking to enhance their service offerings by focusing on integrating more social media and digital engagement strategies. This strategic move is driven by the evolving landscape of digital marketing and the increasing importance of social media platforms in customer engagement. They are therefore rapidly expanding to the foreign markets which means more customers and invariably more data. Their first area of expansion is East Asia and they intend to open an office space in Tokyo, the capital of Japan and the most populous city in the world with approximately 14 million residents. This decision was driven by market research, enabling Tokyo to cater to the local market but also to act as a Hub within the East Asian market.

The new Tokyo office will have its own customers – particular to the region, will be staffed by local talent and staff will regularly communicate back and forth with the UK head office where core business operations will continue to be carried out. In addition, key personnel, executives and C-suite employees may travel and work in Tokyo for a period using data from the UK head office.

Given that Orchid Tech Ltd does not currently have a single cohesive record representing all of their customer data, they are looking to unify these ahead of further data investigation and exploitation, and to pool these data together into a central database. Prior to reaching the conclusions mentioned in this report, Careful analysis on the current technologies and operational infrastructure being employed was taken into account.

This report therefore highlights big data issues that could hamper the company expansion. It discusses at length better choices of database technology, reduction of operational cost and load balancing techniques to eliminate server overloads while maintaining an ACID(Atomicity, Consistency, Isolation and Durability) database.
It forecasts issues that may spring up in the future if the status quo is maintained and the Tokyo office expansion is not discontinued. It suggests legal and ethical concerns – foreign policy, secure communications – that Orchid Tech Innovation Ltd should be aware of as they look to expand their horizons and own foreign offices.

The initial portion of this report meticulously outlines the current technology stack of Orchid Tech Innovation Ltd. As we progress through subsequent sections, the focus shifts towards the company's expansion goal, the challenges this expansion would bring and possible resolutions to these challenges.

Evaluation Of Current Technologies
Orchid Tech Innovations Ltd. currently employs a file system as a database. It uses CSV, XML, JSON, and TXT files for data storage. Whilst this form of data storage is very simplistic and compact, it poses a threat to data retrieval especially as the company looks to expand. Here are some of the drawbacks associated with the current file system storage being employed by Orchid Tech Innovations Ltd:

- Data Redundancy: Storing data in CSV, XML, JSON, and TXT file systems frequently results in data redundancy, especially as such information is stored across multiple files. This situation can lead to inconsistencies and an escalation in storage needs.

- Limited Query Capabilities: Read and writes to a file system database could be burdensome because of the lack of a language of instruction(SQL). Retrieving specific information from a file system can be cumbersome, especially when complex queries are required. File systems lack the query capabilities provided by advanced databases.

- Data Integrity Challenges: It is quite challenging to enforce data integrity in a file system due to the absence of built-in mechanisms to enforce relationships and constraints between data entities.

- Concurrency Issues: Handling concurrent access to data by multiple users or applications can lead to issues like data corruption or loss, as file systems databases do not possess built-in mechanisms for managing concurrent access. Advanced databases usually enforce transactions across write cycles to abate data corruption.

- Scalability Limitations: As data volumes grow, file systems may face scalability limitations. Managing multiple files can become inefficient, leading to

performance bottlenecks. This is very evident in Orchid Tech Innovations Ltd., as they look to expand their operations.

- Security Concerns: File systems usually have limited security features relative to modern database management systems. Access controls are more challenging to implement which could result in security vulnerabilities. This is a major concern to Orchid Tech as privacy concerns over customer data could cause heavy losses in lawsuits and company reputation.

- Lack of Data Independence: Changes in the structure of data may require significant modifications to applications that use the file system. This lack of data independence can result in very high maintenance costs. This is a cause for concern forOrchid Tech as the schema of customer data from both regions of operation are dissimilar.

- No Transaction Support: Transactions are a core defining feature of databases. File systems generally lack transactional support. This subverts the ACIDity of file system databases because there is no built-in way to ensure that a series of operations either succeed or fail as a single, atomic unit. CSV, JSON, XML and all file system storage are very prone to data incoherence.

- Limited Data Retrieval Efficiency: Retrieving specific pieces of data from a file system may require parsing through large files, resulting in inefficient data retrieval. This contrasts with databases optimised for efficient data retrieval based on indexes.

- Challenges in Data Backup and Recovery: Managing backups and recovering data in the event of a failure can be more complex in file systems compared to database management systems, which often have sophisticated backup and recovery mechanisms.

- Verbose: XML, JSON, CSV and other file system databases can be verbose, leading to larger file sizes. The hierarchical structure of Orchid Tech Ltd’s application makes it more complex for simple data storage needs. File system storage lacks a predefined structure, making it challenging to organise and retrieve data in a structured manner. TXT files are not suitable for complex data relationships and may require additional parsing for structured data.

While file systems are appropriate for certain types of data storage and retrieval requirements, they are often less capable than current database management systems in terms of managing complicated data relationships, preserving data integrity, and providing advanced querying capabilities.

For simple tabular data, CSV may suffice, while XML and JSON are more suitable for complex and hierarchical data structures. TXT files are suitable for basic text storage needs. In many real-world scenarios, a combination of these formats may be used based on the nature of the data and the application's requirements.

As a result, for applications with more sophisticated data requirements, a relational database or another advanced data management solution may be more appropriate However, when considering using any of these formats as a database for Orchid Tech Innovation Ltd., it's important to assess factors such as data integrity, security, and scalability. In our case, our application has very sophisticated data requirements which suggests that a dedicated database management system (DBMS) would best suit our needs.

Issues Related to Big Data

The data of Orchid Tech Innovation Ltd., characterised by the large volume and variety, presents several challenges and issues. Addressing these challenges is crucial for them to effectively leverage big data for valuable insights and decision-making.

Here are some key issues related to big data:

- Data Security and Privacy: It is paramount to protect customer's sensitive information from unauthorised access and ensure compliance with privacy regulations across both the UK and Tokyo offices. Data breaches can result in severe reputational and legal consequences therefore robust security measures, encryption, access controls should be implemented which must comply with data protection laws across both regions.

- Scalability: As Orchid Tech Ltd. looks to expand, one key requirement is the use of systems that can scale to handle growing volumes of data. Inadequate scalability can lead to performance bottlenecks and system failures. This can be solved by utilising scalable infrastructure, cloud services, and distributed databases and servers.

- Data Integration: Integrating data from diverse sources with different formats and structures could pose a big problem especially as the nature of customer information in East Asia and Europe could vary vastly. Inconsistent or incompatible data can hinder accurate analysis leading to incorrect data insights. Implementing data integration tools and standardising data formats is a first step to prevent the event of data read or write failures in the future.

- Data Storage: Inefficient storage practices can result in high costs and reduced performance. Therefore, managing and storing large volumes of data cost-effectively must be a given. For data with high volumes such as that of Orchid

Ltd, a DataBase Management System(DBMS) would be a go-to solution. An Relational DataBase Management System(SQL) is best suitable to store customer bio data because of their similar schema and possible needs for complex queries to fetch data. However, recurring changing data like account information alongside transactions should use Non-Relational DataBase Management System(NoSQL) because they are easier to scale horizontally across a variety of nodes. This will ease data fetches from key employees working from Tokyo but have needs for UK customer data.

- Cost Management: Migrating to a DBMS in the cloud often comes with its associated costs. Managing the costs associated with acquiring, storing, and processing such a large volume of data becomes a headache. Due to the likely future expansion of Orchid Ltd, purchasing an on-demand instance alongside a provisioned input/output storage will suit this scenario. Uncontrolled costs can lead to budget overruns and financial strain, therefore measures to implement cost monitoring tools and optimise infrastructure usage should be put in place to checkmate budget overruns.

- Regulatory Compliance: Adhering to various data protection and privacy regulations in the new region of expansion without breaching the laws of the current regions of operation is one consideration that could easily be overlooked. Non-compliance can result in legal penalties and damage to the organisation's reputation.

- Ethical Considerations: Addressing ethical concerns related to data use, bias, and safeguarding user's privacy. Unethical practices can lead to public backlash and legal consequences.

Addressing these issues requires a combination of technological solutions, organisational strategies, and a commitment to data governance and ethics. Regular monitoring, adaptation to evolving technologies, and a proactive approach to problem-solving are essential for successful big data implementation.

Critique of potential solutions

There are several solutions to improve data management, accessibility, and efficiency at Orchid Tech Innovations Ltd.

One tactful approach would be to migrate the database. Consider migrating the data to a relational database management system (RDBMS) and a NoSQL database. Depending on the nature of the data and specific use cases from the Tokyo office, the choice of database could be ascertained. A SQL database provides structured storage with the ability to define relationships which is very ideal for the customer data within the UK region.

Another savvy approach would be to standardise the data storage format to a single, more versatile format that suits the majority of the use cases. Consider setting up a data warehouse that integrates and consolidates data from different formats. Data warehouses facilitate centralised data storage, provide a common schema, and support efficient querying and reporting. This could include implementing data integration tools that can seamlessly handle different data formats. These tools can automate the extraction, transformation, and loading (ETL) processes, ensuring data consistency and reliability across the organisation. The customer data from the Tokyo office being a diverse set of data types and formats from that of the UK office, Orchid Tech Innovation Ltd should consider implementing a data lake. Data lakes can store raw, unstructured, or structured data, providing a unified repository for diverse data sources.

It will also be ideal to adopt NoSQL Databases for JSON and XML. For data stored in JSON and XML formats, consider using NoSQL databases, such as MongoDB for JSON or XML databases like MarkLogic. These databases are designed to handle semi-structured and unstructured data efficiently.

Orchid Tech can and should also explore cloud-based storage solutions that offer scalable and cost-effective storage. Cloud platforms like Amazon Web Services(AWS) and Google Cloud Platform(GCP) often provide managed database services that support a variety of data formats and offer seamless scalability.

It goes without saying that upskilling existing employees should be a top priority. Orchid Ltd should provide training programmes for employees on best practices for handling different data formats. Ensure that new and old employees are properly trained on storage conventions, data models, and integration processes to ensure consistency in data management practices.

By adopting these potential solutions, Orchid Tech can streamline its data management processes, enhance data accessibility, and lay the foundation for more advanced analytics and insights.

Conclusion

Utilising a Database Management System (DBMS) offers numerous advantages that will significantly enhance the efficiency, reliability, and scalability of data management in Orchid Tech Innovation’s Ltd.

First and foremost, a DBMS will provide a structured and organised framework for storing and retrieving data, ensuring data integrity and minimising redundancies. With features such as ACID compliance, a DBMS guarantees that transactions are executed reliably, maintaining the consistency of your data even in the event of system failures or errors. This level of reliability is particularly crucial in Orchid Ltd’s business applications where accuracy and consistency are paramount.

To further butress, a DBMS facilitates powerful querying and reporting capabilities, the ability to create complex queries and perform aggregations will empower Orchid Tech Ltd to make informed decisions based on real-time and historical data trends. DBMS solutions, such as relational databases, offer robust security features, allowing for fine-grained access control and encryption, ensuring that sensitive information is protected from unauthorised access. Data security is a stringent need for Orchid Tech Innovations Ltd given that the data of customers in the UK is distinct from the data of customers in Tokyo. With seamless scalability options, a DBMS can effortlessly adapt to the growing volume of data.

In summary, the integration of a DBMS to Orchid Tech Innovations Ltd not only enhances data management practices but also lays the groundwork for scalable, secure, and efficient information and database systems.

References

1. Database Guides.
   Available Online. **What does ACID mean in Database Systems?**
   [Accessed 13/12/2023]

2. Perfect Learning Blog
   Available Online. RDBMS vs File System: Which is Better for Data Management? (perfectelearning.com)
   [Accessed 13/12/2023]

3. Cloud Storage on AWS. Reliable, scalable, and secure storage for your data
   Available Online. Cloud Storage on AWS (amazon.com)
   [Accessed 13/12/2023]

4. Encyclopedia Britannica, Geography and Travel.
   Available Online Tokyo | Japan, Population, Map, History, & Facts | Britannica
   [Accessed 13/12/2023]

Appendix

```
#Importing the necessary libraries
import csv
import json
from xml.dom import minidom
from pony import orm
from decimal import Decimal

#Dictionary to hold all unified customer data
all_entries = {}

#Parsing the XML file
document = minidom.parse("user_data_23_4.xml")
elements = document.getElementsByTagName('user')

#Parsing the JSON file
json_file = open('user_data_23_4.json', 'r')
json_data = json.load(json_file)

#Parsing the CSV file
csv_file = open('user_data_23_4.csv')
csv_reader = csv.reader(csv_file)
next(csv_reader)

for element in elements:
    first_name = element.attributes.get('firstName').value.strip()
    last_name = element.attributes.get('lastName').value.strip()
    age = element.attributes.get('age').value.strip()
    sex = element.attributes.get('sex').value.strip()
    retired = element.attributes.get('retired').value.strip()
    number_of_dependants = element.attributes.get('dependants').value.strip()
    marital_status = element.attributes.get('marital_status').value.strip()
    company = element.attributes.get('company').value.strip()
    salary = f'{int(element.attributes.get("salary").value.strip()) + 2100}' if company ==
'Williams-Wheeler' else element.attributes.get('salary').value.strip()
    pension = f'{round(1.15 * int(element.attributes.get("pension").value.strip()))}' if
element.attributes.get('pension').value.strip() == '22896' else
element.attributes.get('pension').value.strip()
    address_postcode = element.attributes.get('address_postcode').value.strip()
    commute_distance = element.attributes.get('commute_distance').value.strip()

    duplicate_checker = first_name + last_name + age
```

```
    if all_entries.get(duplicate_checker) is None:
        all_entries[duplicate_checker] = {
            'first_name': first_name,
            'last_name': last_name,
            'age': age,
            'sex': sex,
            'retired': retired,
            'number_of_dependants': number_of_dependants,
            'marital_status': marital_status,
            'salary': salary,
            'pension': pension,
            'company': company,
            'address_postcode': address_postcode,
            'commute_distance': commute_distance,
        }

for csv_data_entry in csv_reader:
    cross_first_name = csv_data_entry[0].strip()
    cross_last_name = csv_data_entry[1].strip()
    cross_age = csv_data_entry[2].strip()
    cross_sex = csv_data_entry[3].strip()
    vehicle_make = csv_data_entry[4].strip()
    vehicle_model = csv_data_entry[5].strip()
    vehicle_year = csv_data_entry[6].strip()
    vehicle_type = csv_data_entry[7].strip()

    cross_duplicate_checker = cross_first_name + cross_last_name + cross_age

    additional_info = {
    'vehicle_make': vehicle_make,
    'vehicle_model': vehicle_model,
    'vehicle_year': vehicle_year,
    'vehicle_type': vehicle_type,
    }
    if all_entries.get(cross_duplicate_checker) is not None and cross_sex ==
all_entries.get(cross_duplicate_checker).get('sex'):
        all_entries.get(cross_duplicate_checker).update(additional_info)
    else:
        all_entries[cross_duplicate_checker] = {
            'first_name': cross_first_name,
            'last_name': cross_last_name,
            'age': cross_age,
            'sex': cross_sex,
        }
```

```
        all_entries.get(cross_duplicate_checker).update(additional_info)

for json_data_entry in json_data:
    comparison_first_name = json_data_entry.get('firstName').strip()
    comparison_last_name = json_data_entry.get('lastName').strip()
    comparison_age = str(json_data_entry.get('age'))
    comparison_address_postcode = json_data_entry.get('address_postcode').strip()
    iban = json_data_entry.get('iban').strip()
    credit_card_number = json_data_entry.get('credit_card_number').strip()
    credit_card_security_code = '762' if f'{comparison_first_name}
{comparison_last_name}' == 'Valerie Ellis' else
json_data_entry.get('credit_card_security_code').strip()
    credit_card_start_date = json_data_entry.get('credit_card_start_date').strip()
    credit_card_end_date = json_data_entry.get('credit_card_end_date').strip()
    address_main = json_data_entry.get("address_main").strip()
    address_city = json_data_entry.get("address_city").strip()
    debt_profile = json_data_entry.get("debt")

    comparison_duplicate_checker = comparison_first_name +
comparison_last_name + comparison_age

    supporting_info = {
        'iban': iban,
        'credit_card_number': credit_card_number,
        'credit_card_security_code': credit_card_security_code,
        'credit_card_start_date': credit_card_start_date,
        'credit_card_end_date': credit_card_end_date,
        'address': f'{address_main}, {address_city}',
        'debt_profile': {
            'amount': debt_profile.get('amount').strip() if isinstance(debt_profile, dict) else
debt_profile or 'N/A',
            'debt_age': str(debt_profile.get('time_period_years')) if
isinstance(debt_profile, dict) else 'N/A'
        }
    }

    if all_entries.get(comparison_duplicate_checker) is not None and
comparison_address_postcode ==
all_entries.get(comparison_duplicate_checker).get('address_postcode'):
        all_entries.get(comparison_duplicate_checker).update(supporting_info)
        if f'{comparison_first_name} {comparison_last_name}' == 'Charlie Short':
            all_entries[comparison_duplicate_checker]['age'] = '52'
    else:
        all_entries[comparison_duplicate_checker] = {
```

```
            'first_name': comparison_first_name,
            'last_name': comparison_last_name,
            'age': comparison_age,
            'address_postcode': comparison_address_postcode
        }
        all_entries.get(comparison_duplicate_checker).update(supporting_info)

#Initialising the database using ponyORM
db = orm.Database()

class Customer(db.Entity):
    '''
    This class is responsible for establishing an entity or table that consolidates all records pertaining to customers,
    serving as the authoritative and centralised source for all customer information within Orchid Tech Innovation Ltd.
    '''
    _table_ = 'ORCHID_TECH_CUSTOMERS'

    ID = orm.PrimaryKey(int, auto=True)
    FIRST_NAME = orm.Required(str)
    LAST_NAME = orm.Required(str)
    AGE = orm.Required(int)
    SEX = orm.Required(str)
    MARITAL_STATUS = orm.Required(str)
    RETIRED = orm.Required(bool)
    NUMBER_OF_DEPENDANTS = orm.Optional(int, nullable=True)
    ADDRESS = orm.Required(str)
    ADDRESS_POSTCODE = orm.Required(str)
    VEHICLE_MAKE = orm.Required(str)
    VEHICLE_MODEL = orm.Required(str)
    VEHICLE_YEAR = orm.Required(int)
    VEHICLE_TYPE = orm.Required(str)
    COMMUTE_DISTANCE = orm.Required(Decimal)
    COMPANY_NAME = orm.Optional(str, nullable=True)
    PENSION = orm.Required(int)
    SALARY = orm.Required(int)
    IBAN = orm.Required(str, unique=True)
    DEBT_AMOUNT = orm.Optional(int)
    DEBT_AGE = orm.Optional(int)
    CREDIT_CARD_NUMBER = orm.Required(str)
    CREDIT_CARD_START_DATE = orm.Required(str)
    CREDIT_CARD_END_DATE = orm.Required(str)
    CREDIT_CARD_SECURITY_CODE = orm.Required(str)
```

```
#Binding the database to a remote MySQL instance
db.bind(
    provider='mysql',
    host='europa.ashley.work',
    user='student_bi52nk',
    passwd='iE93F2@8EhM@1zhD&u9M@K',
    db='student_bi52nk'
)

db.generate_mapping(create_tables=True)

orm.set_sql_debug(True)

@orm.db_session
def commit_entry(callback):
    callback()

#Writing the unified entries into the database
for entry in all_entries.values():
    first_name = entry['first_name'].capitalize()
    last_name = entry['last_name'].capitalize()
    age = int(entry['age']) if entry['age'].isnumeric() else None
    sex = entry['sex'].upper()[0]
    retired = True if entry['retired'].capitalize() == 'True' else False
    number_of_dependants = int(entry['number_of_dependants']) if
entry['number_of_dependants'].isnumeric() else None
    marital_status = entry['marital_status'].capitalize()
    address = entry['address']
    post_code = entry['address_postcode']
    salary = int(entry['salary']) if entry['salary'].isnumeric() else None
    pension = int(entry['pension']) if entry['pension'].isnumeric() else None
    company = None if entry['company'] == 'N/A' else entry['company']
    commute_distance = Decimal(float(entry['commute_distance']))
    vehicle_make = entry['vehicle_make']
    vehicle_model = entry['vehicle_model']
    vehicle_year = int(entry['vehicle_year']) if entry['vehicle_year'].isnumeric() else
None
    vehicle_type = entry['vehicle_type']
    iban = entry['iban']
    debt = int(entry['debt_profile']['amount']) if entry['debt_profile']['amount'].isnumeric()
else None
    debt_age = int(entry['debt_profile']['debt_age']) if
entry['debt_profile']['debt_age'].isnumeric() else None
```

```
    credit_card_end_date = entry['credit_card_end_date']
    credit_card_number = entry['credit_card_number']
    credit_card_start_date = entry['credit_card_start_date']
    credit_card_security_code = entry['credit_card_security_code']

    try:
        commit_entry( lambda : Customer(
            FIRST_NAME=first_name,
            LAST_NAME=last_name,
            AGE=age,
            SEX=sex,
            RETIRED=retired,
            NUMBER_OF_DEPENDANTS=number_of_dependants,
            MARITAL_STATUS=marital_status,
            ADDRESS=address,
            ADDRESS_POSTCODE=post_code,
            VEHICLE_MAKE=vehicle_make,
            VEHICLE_MODEL=vehicle_model,
            VEHICLE_TYPE=vehicle_type,
            VEHICLE_YEAR=vehicle_year,
            SALARY=salary,
            PENSION=pension,
            COMMUTE_DISTANCE=commute_distance,
            COMPANY_NAME=company,
            IBAN=iban,
            DEBT_AMOUNT=debt,
            DEBT_AGE=debt_age,
            CREDIT_CARD_END_DATE=credit_card_end_date,
            CREDIT_CARD_NUMBER=credit_card_number,
            CREDIT_CARD_SECURITY_CODE=credit_card_security_code,
            CREDIT_CARD_START_DATE=credit_card_start_date,
        ))
    except orm.TransactionIntegrityError:
        print('Database Integrity Error, Duplicate entries detected for a supposed unique
row')
        break
    else:
        print('Something went wrong. Please ensure your database credentials are
correct.')
        break
```