SINGAPORE INSTITUTE OF TECHNOLOGY

INF2003 – Database Systems

Project Title: Happy Hub

HAPPY
HUB

Group 13	
Student ID	**Name**
2203235	Timothy Tan Zhe Xu
2201541	Lim Xiang Wei, Alex
2200869	Tan Kai Yang
2200507	Roy Teong Ying Jun
2201190	Zaw Wana

1. Introduction

This report will highlight our application's project work aimed at developing a Happiness Index application as part of this Database Systems module. Measuring and comprehending the well-being and happiness of individuals and groups has become increasingly crucial in today's fast-paced and connected world. This proposal aims to address the need for an accurate tool to measure and track happiness levels, aligning with our group's dedication to fostering positive social change and improving people's lives.

The Happiness Index Application will include features such as quizzes, data visualizations, articles on happiness index to inform users on how to improve factors affecting happiness like quality of life, work-life balance etc. The ultimate aim of the application is to use real-world datasets to improve the happiness level of users by analyzing datasets on what makes people happy, to then suggest improvements/changes to the user's lifestyle and mentality.

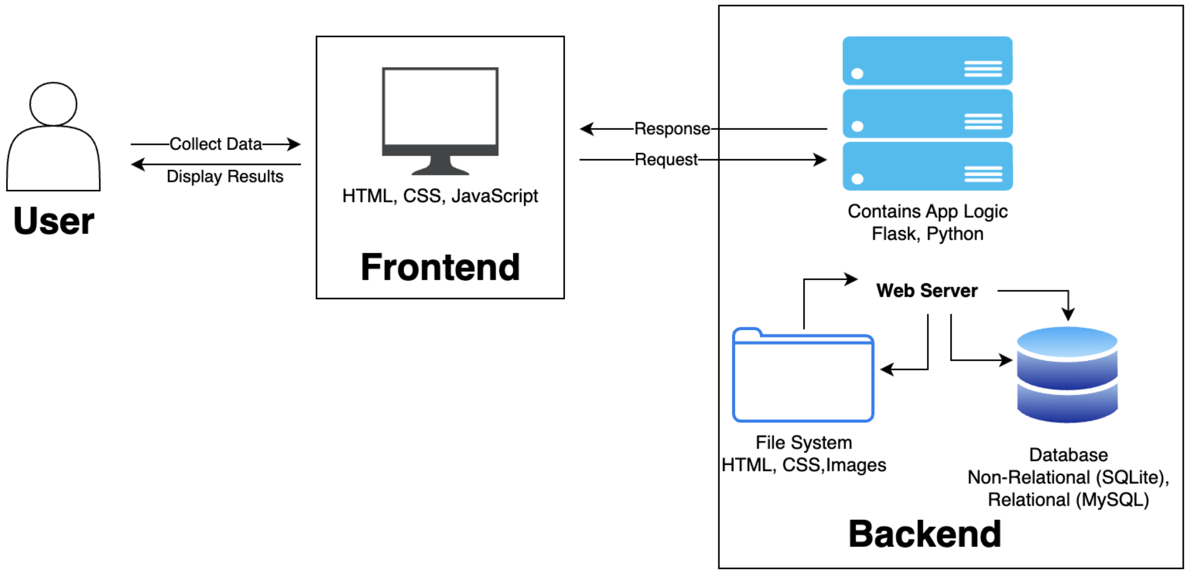
Project Objectives

- Design and develop a database application: The primary objective is to create a database application that utilizes both a relational database and a non-relational (NoSQL) database.
- Explore relational and NoSQL databases: Explore and use the latest technologies and principles related to relational databases and NoSQL databases.
- Apply theories and principles in a real-world scenario: Apply the concepts and principles learned in the module to solve a real-world problem through the development of the database application.
- Identify an application: Select an application for the project that involves both types of databases, either by repeating functionalities in both databases or using each database for different functions.
- Identify and use real-world datasets: Identify and use real-world datasets

Team Contributions and Responsibilities

- Gathering of datasets: Team Research
- Non-relational database: Timothy
- Relational database: Kai Yang
- Integration of Databases: Zaw
- App Logic: Zaw & Alex
- Front-End (GUI) : Zaw & Roy

2. System Architecture Design & Requirements



Client-side (React JS)

The client-side of the application is developed using React JS, a popular JavaScript library for building user interfaces. React JS provides a declarative and component-based approach to developing the front-end, offering a rich set of tools and features to enhance the user experience. In this case, because we are developing an application that is user-centric. React's component-based architecture allows for the creation of reusable and self-contained UI components. This modular approach promotes a user-centric development mindset by enabling developers to break down the user interface into smaller, manageable pieces. Each component can be focused on a specific user interaction or visual element, making it easier to understand, test, and maintain.

Server-side (Flask Py)

The server-side of the application is developed using Flask, a lightweight and flexible Python web framework. Flask is well-suited for building user-centric applications as it allows developers to create efficient and scalable server-side logic. Some of the key points that made us choose Flask for our server-side development are:

1. Routing and Request Handling:
   Flask provides a simple and intuitive routing system that enables developers to define endpoints and handle user requests. This allows for the implementation of user-centric functionalities, such as handling form submissions, processing user inputs, and managing user sessions.

2. Database Integration:
   Flask seamlessly integrates with various databases through libraries like SQLAlchemy, allowing developers to interact with data and store user-specific information. This enables the development of personalized user experience by storing and retrieving user data efficiently.

Gathering Datasets

In our database systems project, we explored two different approaches to obtain datasets for analysis: Kaggle and user surveys. Kaggle, a popular online platform for data science, provided us with a vast repository of diverse and well-curated datasets. We leveraged the power of Kaggle's community and its competitions to access high-quality data that aligned with our project goals. By exploring Kaggle's extensive collection, we were able to find relevant datasets on our topic.
Additionally, we recognized the value of gathering data directly from users through surveys. Conducting surveys allowed us to collect specific information tailored to our project's requirements. By designing well-crafted questionnaires and distributing them to a target audience, we obtained valuable insights that couldn't be found in existing datasets. This approach allowed us to capture user preferences, opinions, and behaviors, giving us a more comprehensive understanding of happiness for users.

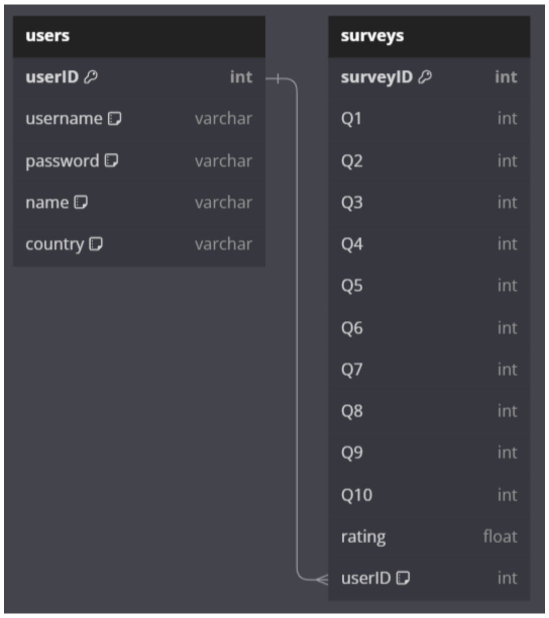
Kaggle Datasets:
Happiness Indicators | Happiness Type | HappyDB | Happiness Report 2023 | World Happiness | Happiness Index

3. Implementation details of relational database and non-relational database

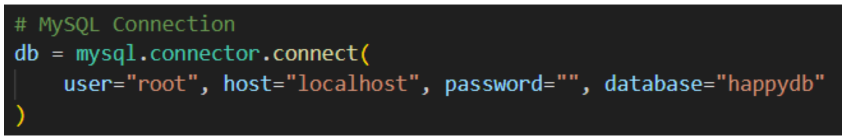
Relational database implementation (**MySQL**):

We implemented MySQL for our relational database largely because it is a database system that our group is familiar with. MySQL is an open-source solution, which means it is cost-effective and can be easily accessed by developers without licensing constraints when considering this project to be of a real-world scenario. In addition, MySQL has been in use for many years and is backed by Oracle Corporation. Its widespread adoption in various industries and applications demonstrates its reliability and suitability for diverse projects.

Entity Relationship (ER) Diagram



Integration with Server Application



```
# MySQL Connection
db = mysql.connector.connect(
    user="root", host="localhost", password="", database="happydb"
)
```

CRUD Functions

Create User : Creating new entries in SQL user table
Login User: Retrieve and read from SQL user table
Update Profile: Update account times in SQL table
Delete User: Delete user data row in SQL table

Advanced design/functions

Session Management:
The primary purpose of session management is to control user access to the database and prevent unauthorized actions. It also facilitates tracking user activities, optimizing resource usage, and providing a seamless experience for concurrent users.

Security:
Password hashing is a fundamental technique used to enhance the security of user passwords stored in databases or systems. Instead of storing plain text passwords, which could be easily compromised if a data breach occurs, hashing algorithms convert passwords into fixed-length, irreversible strings of characters. Employing strong, salted hashing techniques is crucial for safeguarding user credentials and maintaining the confidentiality of sensitive information in modern digital environments.

Non-relational database implementation (**MongoDB**):

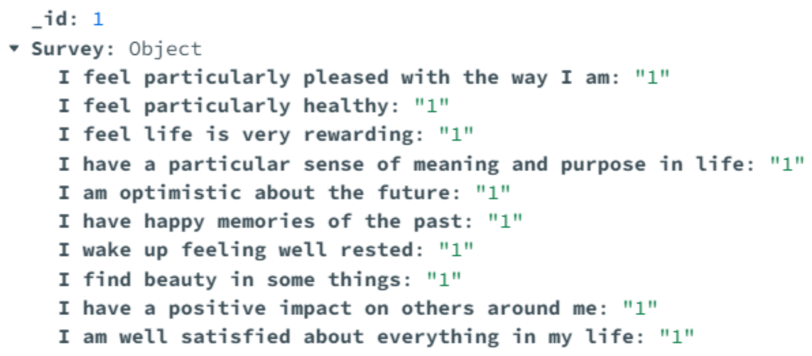
Implementation & Design of NoSQL

Our NoSQL implementation is independent from our SQL implementation. We decided to use NoSQL as we want to address certain limitations of traditional SQL databases, particularly in terms of scalability, flexibility.
For our NoSQL design we first started by identifying the data model. We used the document model to store our survey response. Our team made use of MongoDB, which was what we learnt during our lectures and lab

Create, Read, Update, Delete (CRUD)

Allow Anonymous users to submitting survey response (Create)
Allow Anonymous users to retrieve survey responses which will retrieve their survey responses, happiness index score and retrieve recommendations. (Read)

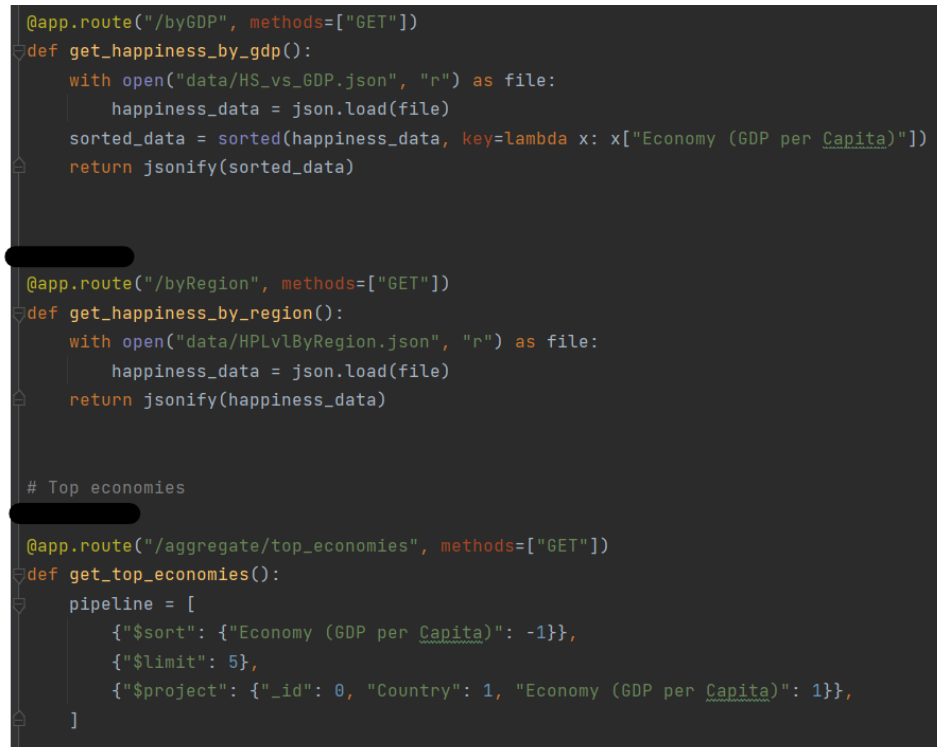
As shown in the image below, the data is saved and retrieve from the NoSQL



```
 _id: 1
▾ Survey: Object
   I feel particularly pleased with the way I am: "1"
   I feel particularly healthy: "1"
   I feel life is very rewarding: "1"
   I have a particular sense of meaning and purpose in life: "1"
   I am optimistic about the future: "1"
   I have happy memories of the past: "1"
   I wake up feeling well rested: "1"
   I find beauty in some things: "1"
   I have a positive impact on others around me: "1"
   I am well satisfied about everything in my life: "1"
```

Advanced Functions

Incorporated Kaggle data into MongoDB allowing us to perform complex aggregation functions, enabling in-depth data manipulation and analysis. By combining survey responses with external data sources, we derived valuable insights into happiness trends, factors affecting happiness scores, and regional variations. Using MongoDB's aggregation pipeline, we crafted queries that facilitated seamless integration of diverse datasets, presenting users with comprehensive and actionable information as shown in the image below



```
@app.route("/byGDP", methods=["GET"])
def get_happiness_by_gdp():
    with open("data/HS_vs_GDP.json", "r") as file:
        happiness_data = json.load(file)
    sorted_data = sorted(happiness_data, key=lambda x: x["Economy (GDP per Capita)"])
    return jsonify(sorted_data)


[redacted]
@app.route("/byRegion", methods=["GET"])
def get_happiness_by_region():
    with open("data/HPLvlByRegion.json", "r") as file:
        happiness_data = json.load(file)
    return jsonify(happiness_data)


# Top economies
[redacted]
@app.route("/aggregate/top_economies", methods=["GET"])
def get_top_economies():
    pipeline = [
        {"$sort": {"Economy (GDP per Capita)": -1}},
        {"$limit": 5},
        {"$project": {"_id": 0, "Country": 1, "Economy (GDP per Capita)": 1}},
    ]
```

4. Discussions

Scalability

By using the horizontal scaling capabilities of NoSQL, the application efficiently distributed data across multiple nodes, allowing it to handle a growing number of users and survey responses. As the user base expanded, we could handle increasing data load without compromising performance. This scalability was crucial in ensuring a smooth user experience and provided the infrastructure to accommodate future growth

Reliability

By utilizing both SQL and NoSQL databases, we ensured data redundancy and robustness in case of failure. The SQL database maintained critical structured data, while the NoSQL database managed dynamic and unstructured survey responses. Additionally, we implemented regular data backups and monitoring to detect and address potential issues promptly. As a result, the application provided users with a dependable platform to assess their happiness levels, and a reliable website for users to use.

Security

Implementation of session as a exchange mechanism that will be used between the user and the web application to share and continuously exchange the session ID. This ensure that authentication is in place and unauthorized users cannot access unauthorized pages

Limitations and Areas for Improvement

Performance Optimization:
As the user base and survey responses continue to grow, there may be opportunities to optimize query performance further. Regular performance monitoring can help enhance response times and overall system efficiency.

Enhanced User Engagement:
Implementing interactive features and personalized recommendations based on user responses could further engage users and create a more immersive user experience.

Continuous Security Audits:
Conducting periodic security audits and assessments can identify potential vulnerabilities and further strengthen the application's security measures.

User Privacy Controls:
Offering users more control over their data and the option to manage their privacy preferences can enhance user trust and confidence in the platform.

5. Reflection and takeaways

This project taught us that technical skills do not matter if you do not put your skills towards a meaningful project. Anyone can make use of SQL and NoSQL, but not everyone can make an impact with these skills. That being said, our group made it our mission to put our skills towards the betterment of society by trying to improve the mental health of our users.

As the most important thing in the project was the algorithm to manage the happiness index, we designed it in a way that it only showed relevant information to improve upon what the user seemed to lack in terms of fulfillment. We only show tips and advice on how to improve on aspects of a user's life that he is lacking in. We also made it such that we can see the improvement of happiness over time for a user. This made it such that users will have a feeling of accomplishment and improvement. Our goal is to continue this project after the submission, and to hopefully one day improve at least one person's happiness. That will mean the world to us.

This project ignited our group's passion to create or work on projects that benefit people in one way or another, and we will remember this experience for our future endeavors.

Appendix & References

[1] Kaggle Datasets

https://www.kaggle.com

https://www.kaggle.com/datasets/loveall/human-happiness-indicators

https://www.kaggle.com/datasets/sourabhbaldwa/bonding

https://www.kaggle.com/datasets/ritresearch/happydb

https://www.kaggle.com/datasets/ajaypalsinghlo/world-happiness-report-2023

https://www.kaggle.com/datasets/unsdsn/world-happiness

https://www.kaggle.com/datasets/sougatapramanick/happiness-index-2018-2019

[2] GitHub Source Code

https://github.com/kaya0910/INF2003_13

[3] Database References

https://www.mongodb.com/docs/

https://legacy.reactjs.org/docs/getting-started.html

https://flask.palletsprojects.com/en/2.3.x/

https://dev.mysql.com/doc/

END OF REPORT