

## 1. Requirements

### 1.1 Explanation of the choice of techniques

- **Elicit the requirements**

*From Business Analysis Body of Knowledge (BABOK), there are several techniques used in this project to elicit requirements which is displayed below with explanations.*

- 1. Brainstorming**

*In order to produce a wide range of ideas and perspectives about the objectives and goals of the project, I need to gather a diverse group of stakeholders, including potential users and subject matter experts, for the purpose of brainstorming to elicit requirements for my project. The objective of this creative approach is to list and rank the features and functionalities that my data visualization software needs. Brainstorming facilitates the process of developing a thorough and well-rounded list of needs for my project by promoting creative thinking. By using this method, the final result will be certain to meet customer requirements and project goals.*

- 2. Research on Similar Web Apps**

*It's important to look for related web apps for a number of reasons. It gives me information on popular features and user-friendly design elements, so I can figure out what makes the application function well. Furthermore, studying competing applications can point out areas that need work, giving me the opportunity to set my app apart with special features or an improved user experience. In order to keep my app current and competitive, this research also keeps me up to date on industry trends and technology developments. When developing my app, I can make better judgements if I have a better insight of the user base and the state of the market.*

- **Document the requirements**

- 1. User stories**

*In Agile software development, user stories are a tool that help describe software features from the viewpoint of an end user. The format they usually take is this: "As a [role], I want [goal] so that [benefit]." This framework facilitates planning and prioritizing development activities while maintaining attention on user demands. Acceptance criteria are frequently included with user stories to provide a precise specification of the story's completion point. They guarantee that the product complies with user requirements and assist iterative development.*

- 2. Functional and non-functional requirements natural language specification**

*In natural language specifications, the terms "functional and non-functional requirements" relate to a way of describing software needs. Functional requirements outline the behaviours and functions that the system must do. Non-functional requirements cover things like user interface aesthetics, efficiency, and dependability in detail about how the system accomplishes specific tasks. This method makes itself understandable to all stakeholders involved, even those without technical skills, by using simple, straightforward language. It*

*guarantees comprehensibility and clarity, which promotes improved communication and comprehension of the goals and proper usage of the product.*

- **Prioritise the requirements**

- 1. MoSCoW prioritization**

*MoSCoW prioritisation is a technique used in software development and management to help stakeholders decide on how important it is to fulfil each requirement. The characters in the acronym stand for "Must have, Should have, Could have, and Won't have." This method aids with managing and clarifying expectations, guaranteeing the delivery of essential criteria while allowing for flexibility in allowing for other less important aspects.*

- **Context diagram**

*A context diagram is a straightforward visual aid that illustrates the limits and scope of the system as well as its external entities and their interactions with it. This high-level perspective doesn't go into specifics on the inner workings of the system, preferring to comprehend the environment in which it functions and how it interacts with external factors. This tool helps to understand system needs and boundaries by providing a visual representation of the system within a larger environment.*

## **1.2 Prioritised requirements (app2 only)**

### **Using MoSCow prioritization techniques by following steps**

- 1. Identify key stakeholders**

*Identifying key stakeholders should base on my dataset and product. It's about Initial teacher training performance profiles in England. So the key stakeholders will be teachers in England, government education official, job seeker, students who are in education institution, education policy analyst, teacher training provider and customers and etc.*

- 2. List**

*State each user story by "functional" or "non-functional" for the current project. Make a comprehensive list in the order of "Must have", "Should have", "Could have" and "Won't have" to make the list direct and understandable.*

- 3. Start prioritising**

*Approach each feature or initiative on the list methodically, asking the following questions. Confirm each one in "should have", "must have", "could have" and "won't have".*

ID	User Story	Type	MoSCoW
US1	As a policy maker, I want to view trends in teacher qualifications (QTS) and employment status over time, so that I can make informed decisions about education policy.	Functional	Must have
US2	As a diversity officer, I want to evaluate the ethnic diversity of teachers, so that inclusive recruitment practices can be ensured.	Functional	Must have
US3	As a school administrator, I want to access to statistics regarding the age distribution of teachers, so that I can make decisions about future hiring and training needs	Functional	Must have
US4	As a policy analyst, I want to see how teachers are distributed by gender and age group, so that I can understand demographic trends in the teaching workforce.	Functional	Must have
US5	As a customer, I want a user-friendly navigation, so that I can access to wanted data quickly.	Non-Functional	Must have
US6	As a teacher training provider, I want to find the relationship between course levels and employment results, so that training programs can be improved.	Functional	Must have
US7	As a governmental auditor, I want to compare the percentage of teachers with disabilities, so that I can evaluate the level of inclusion in the labour force.	Functional	Should have
US8	As an educational researcher, I want to investigate the gender distribution of teachers across various course levels, so that I can take gender equity studies.	Functional	Should have
US9	As a university admissions officer, I want to evaluate the success rate of our graduates in finding teaching positions, so that I can determine whether anything needs to be altered.	Functional	Should have
US10	As a teacher, I want to view historical trends in employment status, so that I can make a better career planning.	Functional	Should have
US11	As a user, I want high data accuracy and up-to-date information, so that I can make reliable analysis.	Non-Functional	Should have
US12	As a customer, I want interactive data visualizations, so that I can make more engaging data exploration.	Non-Functional	Could have
US13	As a user of the app on different devices, I want the app to support data interoperability, so that it is easily usable between laptops, phones and desktop computers.	Non-functional	Could have

US14	As a student considering a teaching career, I want an interactive feature where I can input my qualifications to predict my employability, so that I can make career plan to be a teacher.	Functional	Won't have
------	--	------------	------------

When we finish the prioritised requirements by MoSCoW, we can go through the next part.

## 2. Design

### 2.1 Interface design (app2 only)

There are three primary sets of pages we need to think about based on our goals:

1. Home page
2. Data analysis pages
3. User-related pages
4. Report and feedback pages

Let's go through each set of pages in turn.

#### Home page

The home page is where users first in when they open the app.

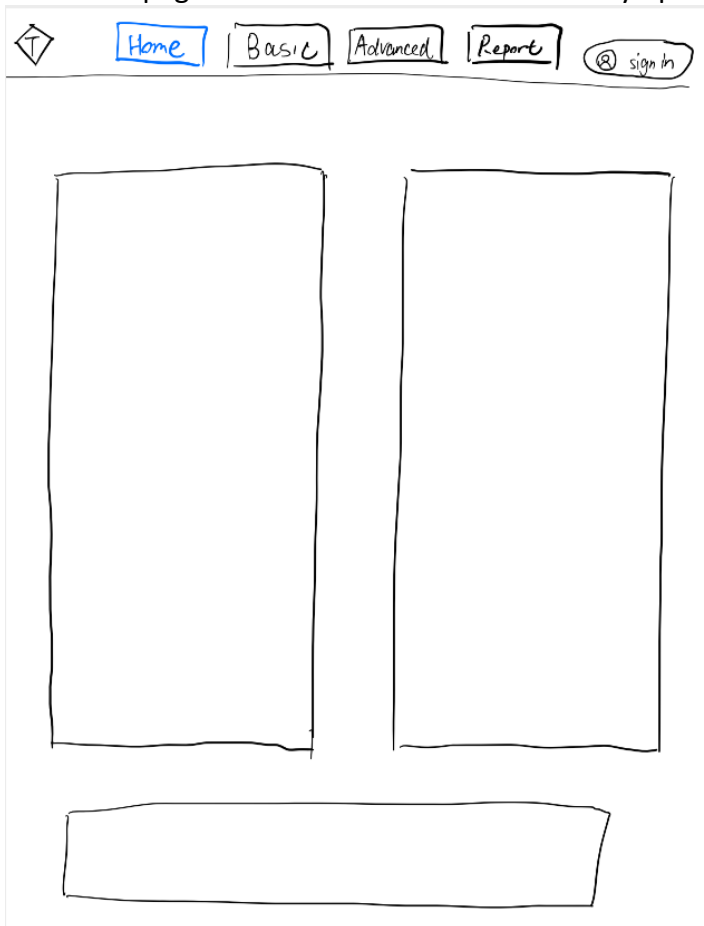


Figure 1. Home page

#### Data analysis pages

The data analysis pages display the main content of the app2, which include basic data analysis page and advanced data analysis page.

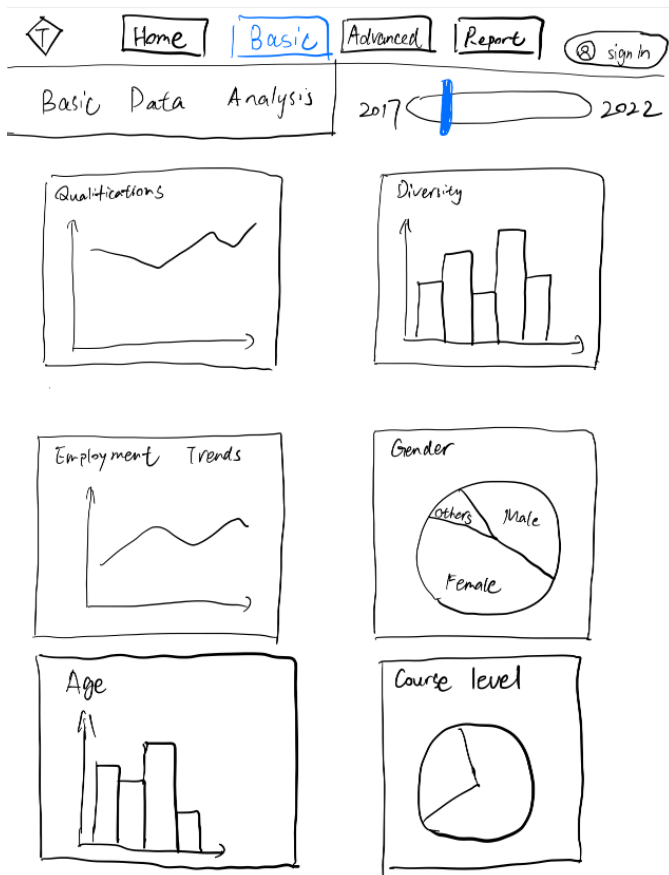


Figure 2. Basic data analysis page

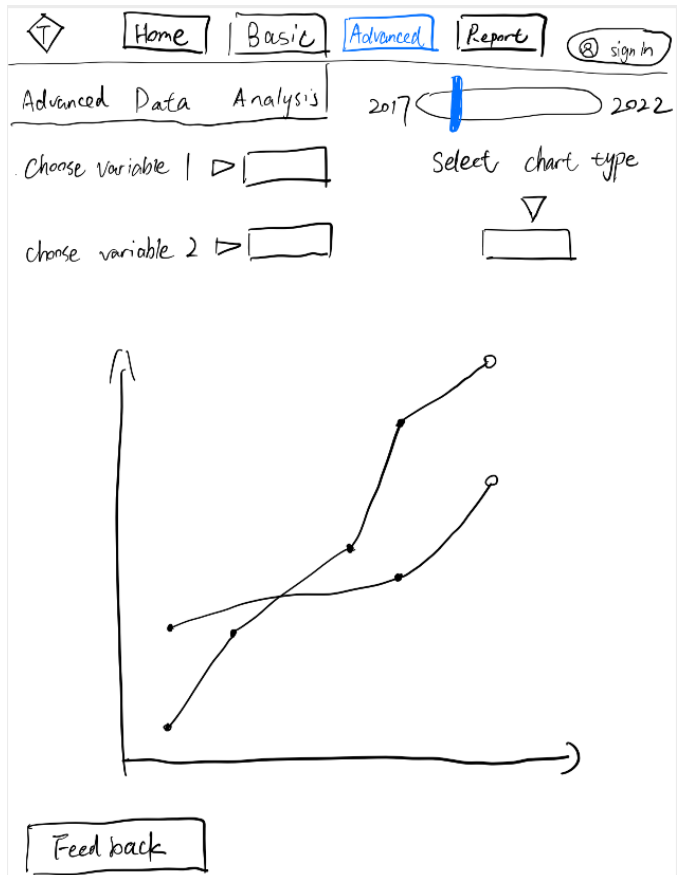


Figure 3. Advanced data analysis page

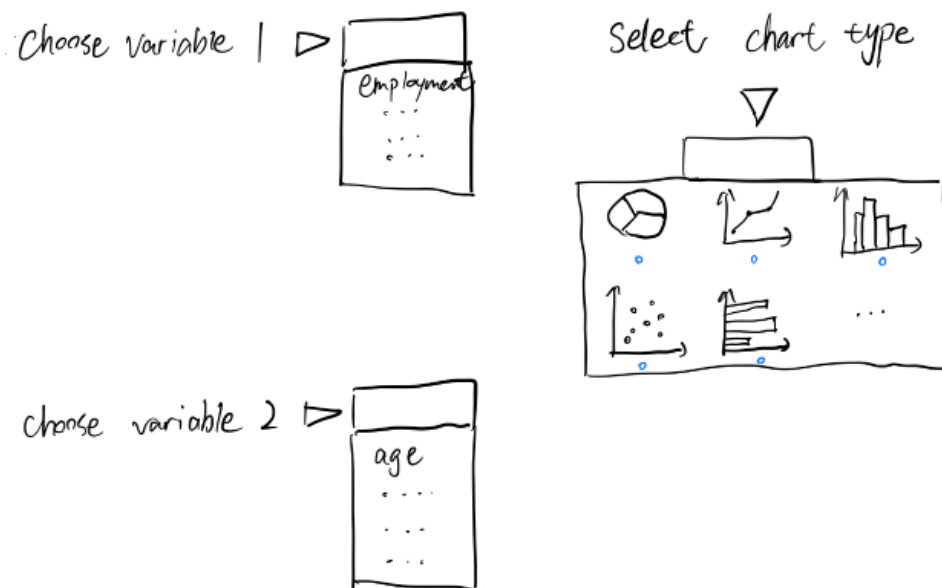






Figure 4. Details of advanced data analysis page

## User-related pages

User-related pages involve login page, register page, change password page and user profile page.



# Login

by   

\_\_\_\_\_ or \_\_\_\_\_


Email

\_\_\_\_\_

Password:

\_\_\_\_\_

Figure 5. Login page



# Sign Up

\_\_\_\_\_

Email :

\_\_\_\_\_

User Name :

\_\_\_\_\_


Password :

\_\_\_\_\_

Confirm password:

\_\_\_\_\_

Figure 6. Register page



# Change Password

\_\_\_\_\_

Email :

\_\_\_\_\_

New password :

\_\_\_\_\_


Confirm new password:

\_\_\_\_\_


Email confirm letter:

\_\_\_\_\_


Figure 7. Change password page



# User profile



## Basic Information

Photo : 

Name : >

Date of birth : (optional) >

Gender : (optional) >

## Contact Information

email : \_\_\_\_\_

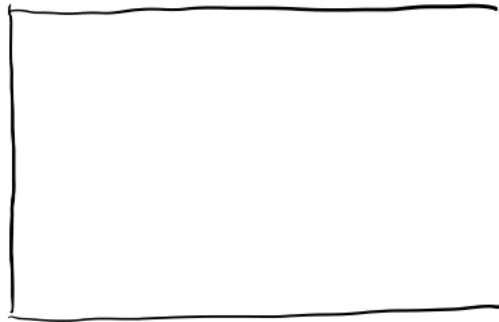
Phone number : (optional) \_\_\_\_\_

Address : (optional) \_\_\_\_\_

Figure 8. User profile page

## Report and feedback pages

Send us Feedback:



Pictures (optional):

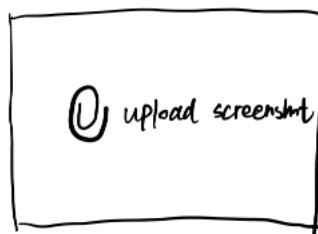


Figure 9. Report and feedback page

### Flow of application

The flow of the application, as designed in the wireframe usually follow the next steps

- **Sign up and login in**  
*Users need to register an account and log in the app to function it.*
- **Start at the main screen**  
*The home screen is where users start, with a simple top navigation bar for easy access.*
- **Data visualisation**  
*Find basic data on Basic data analysis page and choose wanted variables and chart type to find their relationship on Advanced data analysis page. The application dynamically shows a graph or chart beneath the dropdown menus once the variables have been selected.*
- **Interaction with data**  
*The displayed data can be interacted with by users. This could include dragging the cursor over graph parts to view more information, modifying the variables for various analysis, or tweaking the filters to focus the data.*
- **Feedbacks**  
*If users find problems or errors, they can state the problems with the help of screenshots on Report page. Then developers will revise the wrong part as soon as possible.*



- **User profile**

*Users can modify their information on User profile page. They can also log out if they finish their work and quit the app.*

Map out the user's path through the app using flowcharts.

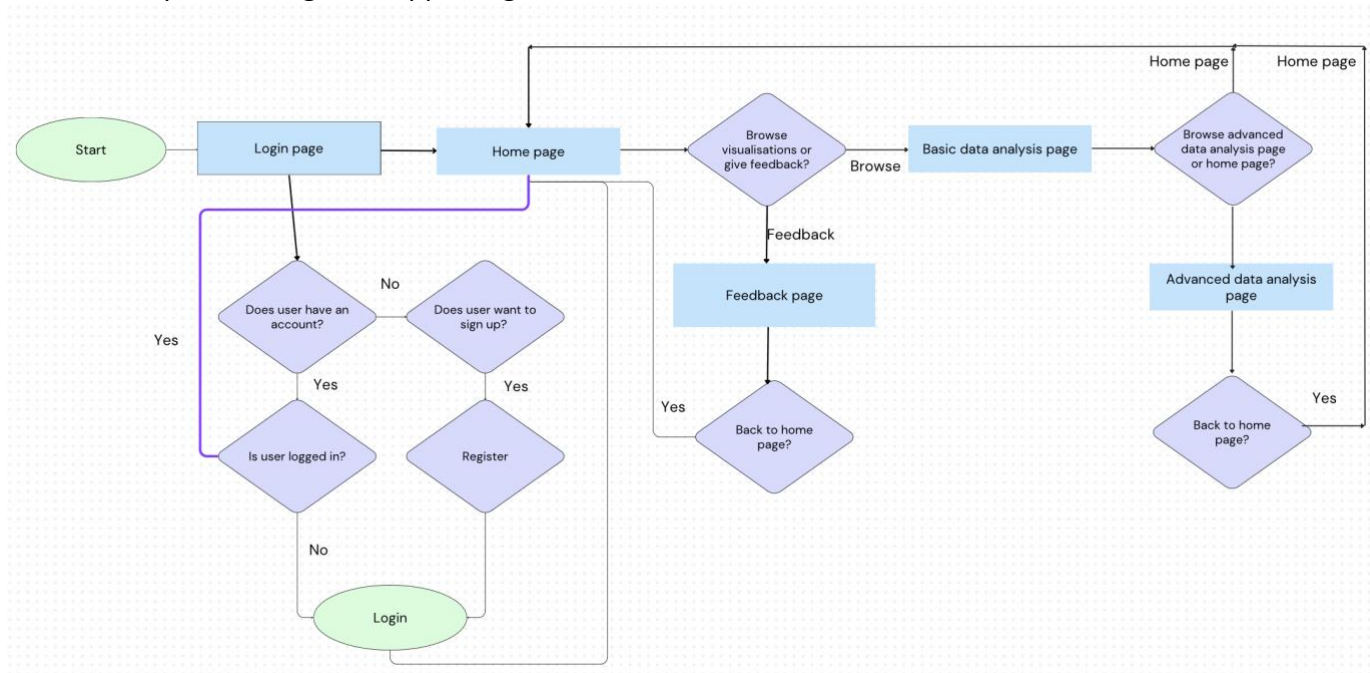


Figure 10. Flow of application

## 2.2 Application design (app2 only)

### Application design for a data visualisation app

#### 1. Decide on Multi-page app

*A Dash app that only has one page with all of the content loaded is called a single-page Dash app. Without visiting various URLs, users can interact with various components and visualisations. It works well for applications that are engaging, focused, and offer a smooth user experience.*

*In comparison, a multi-page Dash programme has several pages, each with an own URL. It uses the standard web navigation method, in which users navigate between pages by clicking links or buttons. This arrangement works better for intricate applications that need separate pages for organisation and clarity and have different information types.*

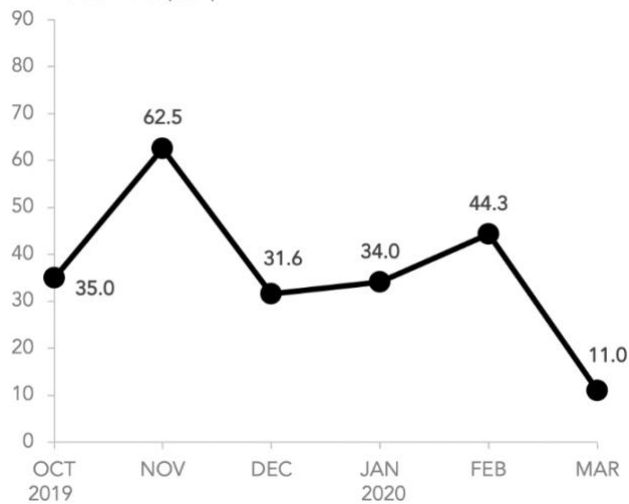
*A multi-page Dash app would be the most suitable for my project. The wireframes that I designed indicate that separate pages are required for various forms of data analysis, user interactions and problem report. This aligns well with the multi-page structure, where each page can be dedicated to specific functionalities like basic data analysis, advanced data analysis, and user-related operations (like login, registration). When managing diverse forms of data analysis and user interactions, a multi-page application facilitates a more systematic and organised approach. Because of my application's complexity and range of requirements, this architecture works well.*

#### 2. Choose chart types

*Based on the user stories and the wireframes, there are some suitable charts can be applied within the dashboard.*

- **Trends in Teacher Qualifications and Employment Status:**

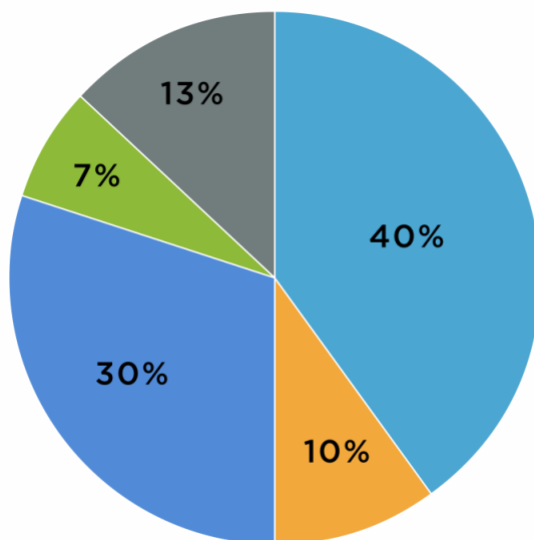
*A line chart or a multi-series line chart would be good to show trends over time.*



*Figure 11. An example of line chart*

- **Evaluating Ethnic Diversity of Teachers:**

*Pie charts can effectively represent the distribution of ethnic groups.*



*Figure 12. An example of pie chart*

- **Statistics on Age Distribution of Teachers:**

*Bar charts would be suitable to show age distribution.*

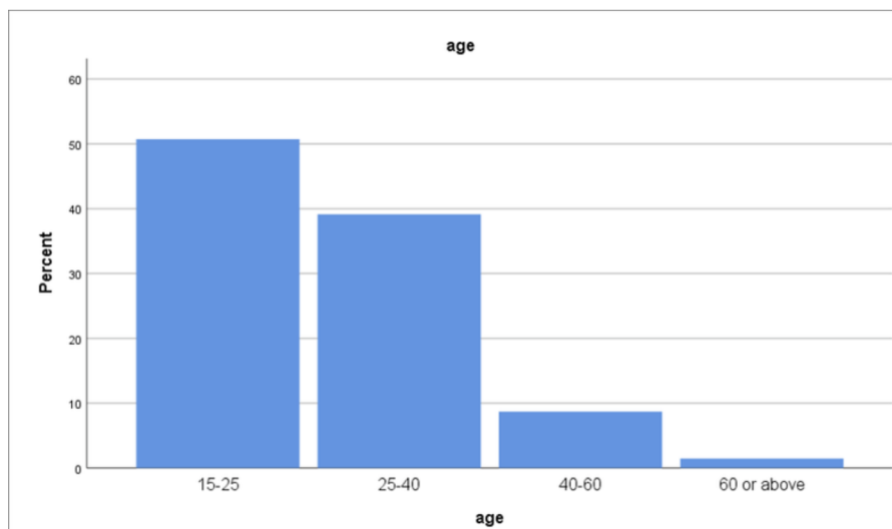


Figure 13. An example of bar chart

- **Relationship Between Course Levels and Employment Results:**  
Scatter plots might be used to analyse relationships between these two variables.

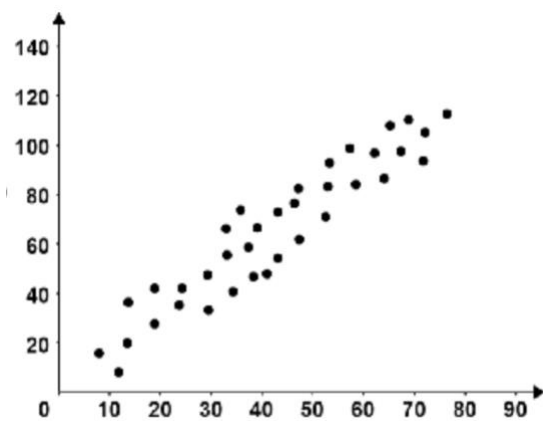


Figure 14. An example of scatter plot

- **Comparison of Teachers with Disabilities:**  
A comparative bar chart may be used to find these percentages.

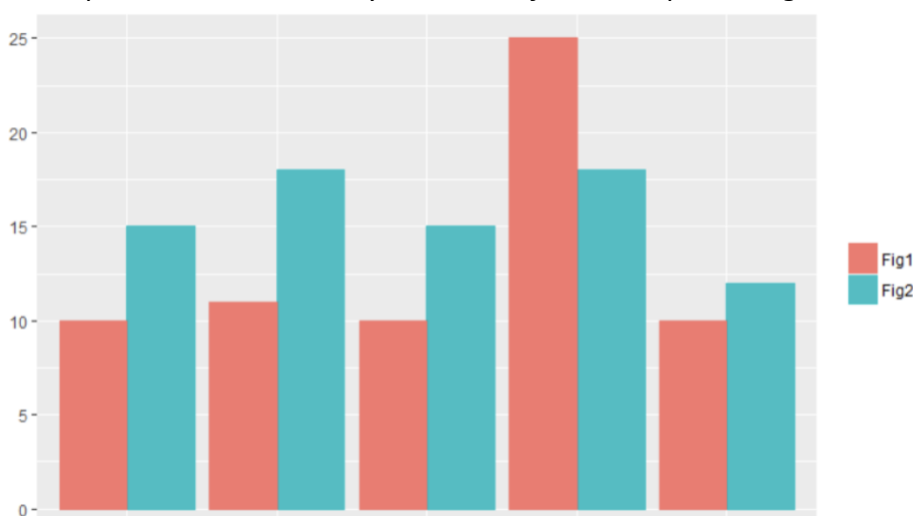


Figure 15. An example of comparative bar chart

With each sort of chart, information will be effectively communicated to my target audience as each chart type aligns with the nature of the data and the specific insights from my user stories.

### 3. Identify functions

*By considering the user stories, my overall dashboard approach and different charts I would like to create, a table is constructed below which decides elements presented on each page.*

Feature/Function	Description	Chart type	Data attributes
Homepage	A summary of data insights that provide access to the main features.	N/A	Summary metrics from all categories
Trends analysis	Time-based trends are shown in charts that can be filtered by year or other relevant time periods.	Line chart etc.	Time period, Trends data
Demographic analysis	The distribution of demographics, such as age or ethnicity, can be seen visually	Bar chart etc.	Age groups, Ethnic data
Qualification analysis	Graphics that show the relationship between employment outcomes and qualifications.	Scatter plot etc.	Qualification level, Employment level
Login	For user to log in the app.	N/A	User credentials, account details
Sign up account	For user to register a new account to log in.	N/A	User credentials, account details
Password change	For user to change their passwords if they forget the original one.	N/A	User credentials, account details
Feedback	A feedback submission form with an option to view responses in a table or list format.	N/A	User feedback content, Response status
User profile	For user to change their own information.	N/A	User information detail

*These elements should be designed for easy navigation, clear presentation of data, and efficient user interaction.*

### 4. Identify model classes

*To identify model classes, we can consider the structure of the dataset along with the features with the help of table.*

Class	Description	Attributes
Teacher class	Represent the teachers	'age_group', 'gender', 'ethnicity', 'qualification_level', 'disability_group'
Course class	Represent the teacher training course	'course_id', 'course_level'
Time class	Represent the time of academic year	'year', 'time_identifier'
Employment class	Contains data related to employment outcomes	'employment_rate', 'employment_status'

User class	Represent user information	'user_id', 'user_name', 'password', 'email', 'photo', 'user gender'
Feedback class	Store feedbacks from users	'feedback_id', 'user_id', 'feedback_content', 'feedback_time'

*I can use these classes to organise the backend of my application by capturing the main entities and their attributes.*

## 5. Model the relationships between classes and functions

*To model the relationships between classes and functions by creating a UML diagram which includes classes, attributes, methods and relationships.*

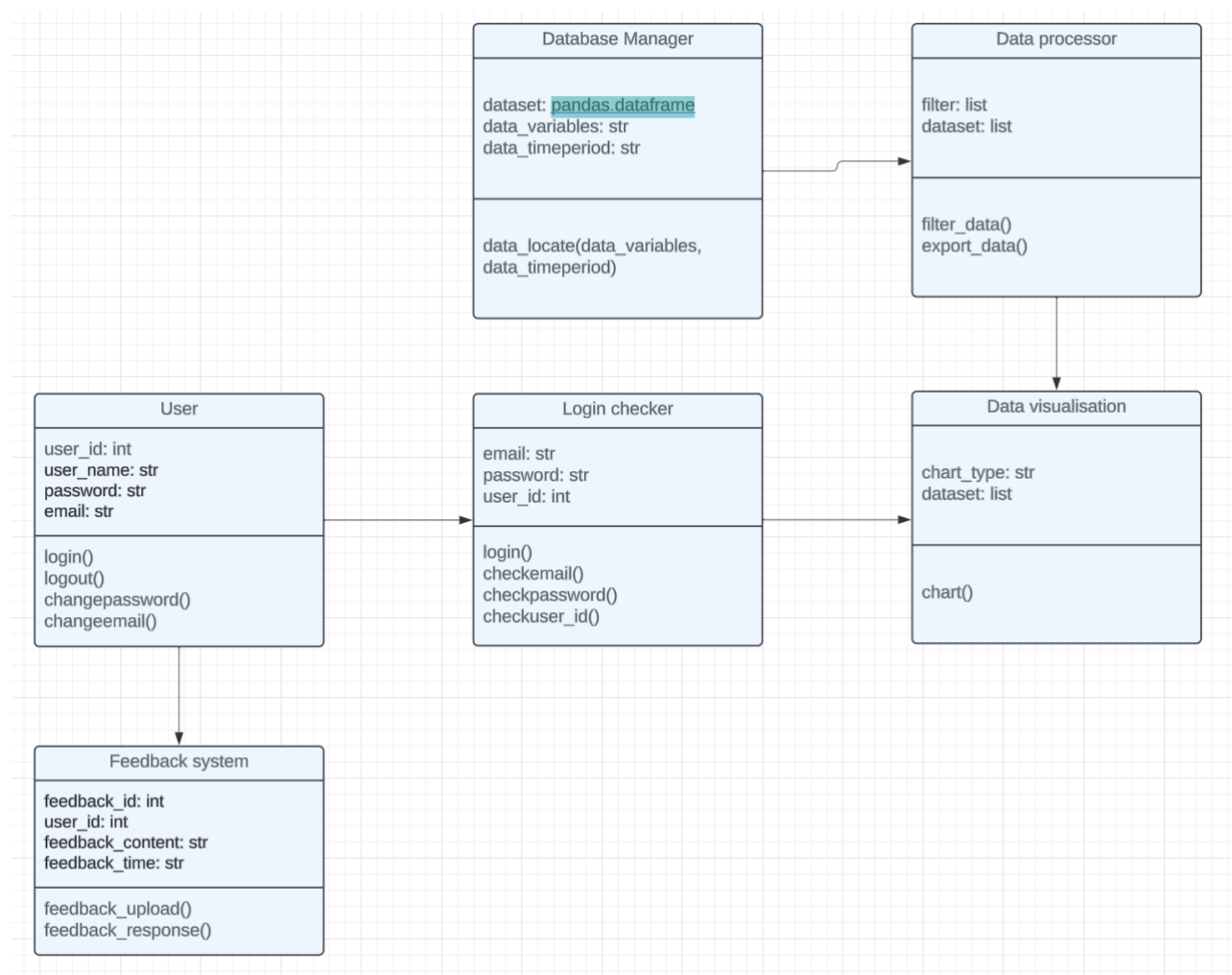


Figure 16. UML diagram

- **User**  
*The user block includes User class and its attributes with related methods*
- **Login checker**  
*The login checker block includes the same attributes with user block and check the accuracy of users' email and passwords.*
- **Database Manager**

The database manager block contains data from Teacher class, Course class, Time class and Employment class with corresponding attributes summarised in data\_variables.

- **Data processor**

The data processor block includes data filter to meet the needs of users and process the right data, then it exports data to the next block Data visualisation.

- **Data visualisation**

The data visualisation block shows the right chart and supply interactive action to users.

- **Feedback system**

The feedback system block includes Feedback class and attributes which offers users to report problems and get response.

The benefit of using UML is that it offers a standardised method for visualising system architecture. UML provides a straightforward approach to organising and recording a system's requirements and architecture. It helps in understanding a system's dynamic behaviour as well as its static structure, which is important for implementation, design, and maintenance.

## 2.3 Database design

### Entity Relationship Diagram (ERD)

An Entity Relationship Diagram (ERD) is a graphic representation that shows the relationships between entities. In order to ensure a well-organized and normalised database, an ERD is used to graphically map out a database's structure before constructing the database. For effective data retrieval and integrity, it aids in the identification of redundant data, architectural planning, and relationship and constraint establishment.

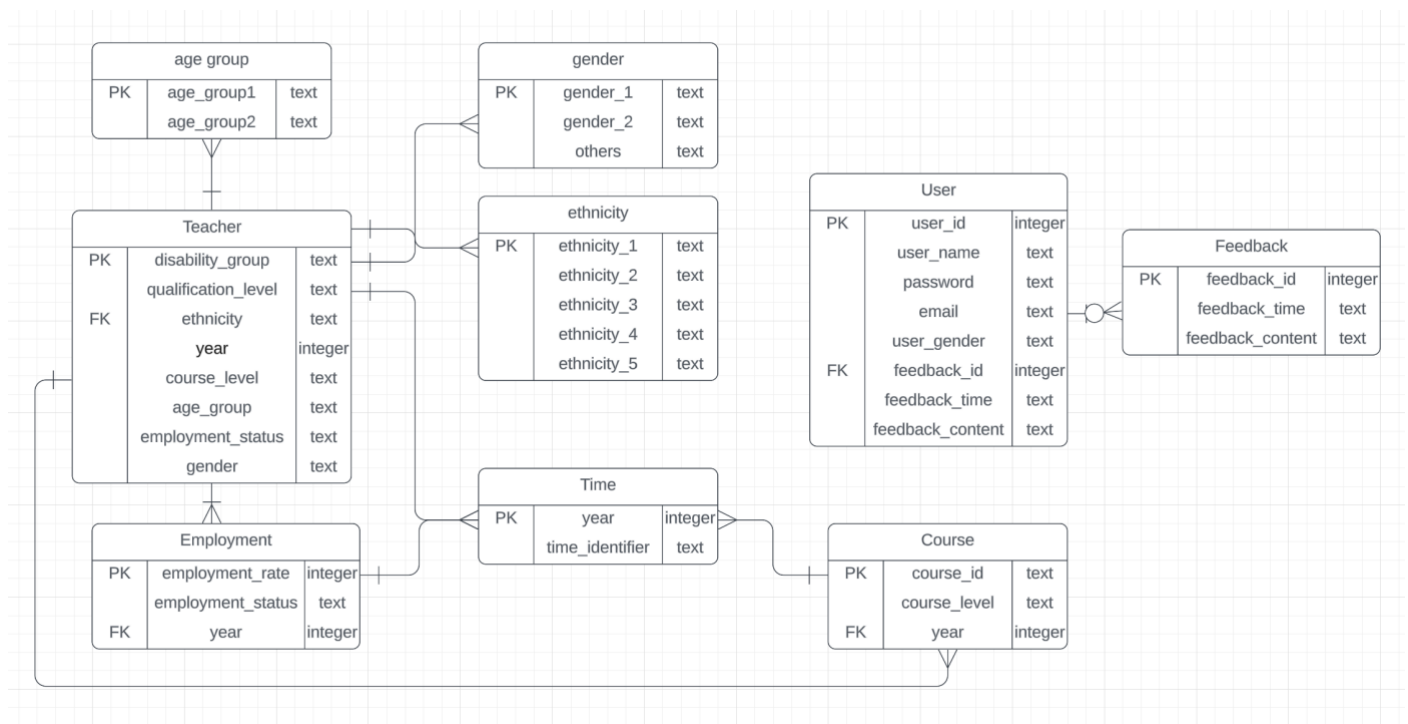


Figure 17. ERD diagram

For my ERD, each table has a unique primary key to identify its records. There are clear relationships between tables, like Teacher linking to Age Group, Employment, Time, Ethnicity, Course and Gender through foreign keys. If I delete a record from a lookup table like Gender, it won't affect the Teacher records

*directly, preserving data integrity. For normalization part, I chose to normalized Age group, Gender, Ethnicity, Employment, Course and Time to reduce redundancy and improve data integrity. I established a single source of truth for each of these attributes by making reference tables for them. This implies that I just need to make the modification once to update an attribute, such as adding a new gender category or changing the age range. Additionally, by lowering the quantity of duplicate data, it can enhance efficiency and simplify queries.*