COMP0035 2023/24 Coursework 2

1. Requirements

Explanation of the choice of techniques

From the 9 techniques of elicitation in BABOK. The techniques that will be used in this project are shown as follows. (BABOK, 2015)

Interface Analysis: The interactions between app users, the REST API and the web app will be identified. This technique is used because it can help define the scope of the project of interest.

Observation or ethnography: The information of app users will be evaluated from the persona created in coursework 1.

Prototyping: Interface requirements can be captured and evaluated using user stories throughout the life cycle of the app development. They can be visualised using wireframes.

Analysing documentation: Requirements can be easily captured by looking at similar existing web apps.

In documenting the requirements, as this project is suitable for Agile method, **user stories** will be used so that I can have a better understanding of needs of app users.

In prioritising the requirements, the technique that will be used is **MoSCoW**. The reason for choosing this technique is that it categorises the requirements into four groups: “Must have”, “Should have”, “Could have” and “Won’t have for now”. It is easy to assign the requirements to these categories without any ambiguity.

Prioritised requirements

The documented and prioritised set of requirements is shown in the following table.

Number	User story	Acceptance Criteria	Functional/non-functional	Category
US01	As a web user, I want the app to support different browsers so that I can access the app using any browsers.	1. Support the 5 most popular browsers. 2. Comply with cookie policy.	Non-functional	Must have
US02	As a web user, I want to see the charts clearly on a desktop so that I can get insights into the	1. The layout is clean and simple. 2. The graphs are readable and clear	Non-functional	Must have

	data.	in landscape orientation on a standard 1920x1080 pixel screen.		
US03	As a web app user, I want to be able to specify the chart which I currently want to see so that I can search for and look at only the relevant data.	1. The chart of interest can be chosen from the sidebar. 2. The sidebar includes all suitable charts available.	Non-functional	Must have
US04	As a web app user, I want to be able to interact with the charts by selecting the item and time range of interest so that I can obtain further information without needing to open more charts.	1. Charts have filtering tools embedded which allow the user to display only specific information.	Non-functional	Should have
US05	As the store manager, I want to update the dataset so that the data is up to date.	1. Add new data. 2. Remove data from the existing dataset.	Functional	Should have
US06	As a data scientist, I want to post my comments of the dataset so that I can share it with other web users.	1. Support account management, including login, registering, and resetting passwords. 3. All comments should be recorded and trackable	Functional	Could have
US07	As a data scientist, I want to have download functionality so that I can use the data and charts for further analysis.	1. Download the data in xlsx or csv format. 2. Download the chart in png format. 3. For the maximum size of data and chart, the download process	Functional	Could have

		finishes within 30 minutes.		
US08	As a foreign web user, I want to have translation functionality so that I can understand the outcome of the app easier.	1. Connected with a translation API. 2. Return the translation results within 10 seconds.	Functional	Won’t have for now

Table 1. Documented and prioritised requirements.

2. Design

Interface design

Wireframes that represent the interface for the web app are shown as follows:

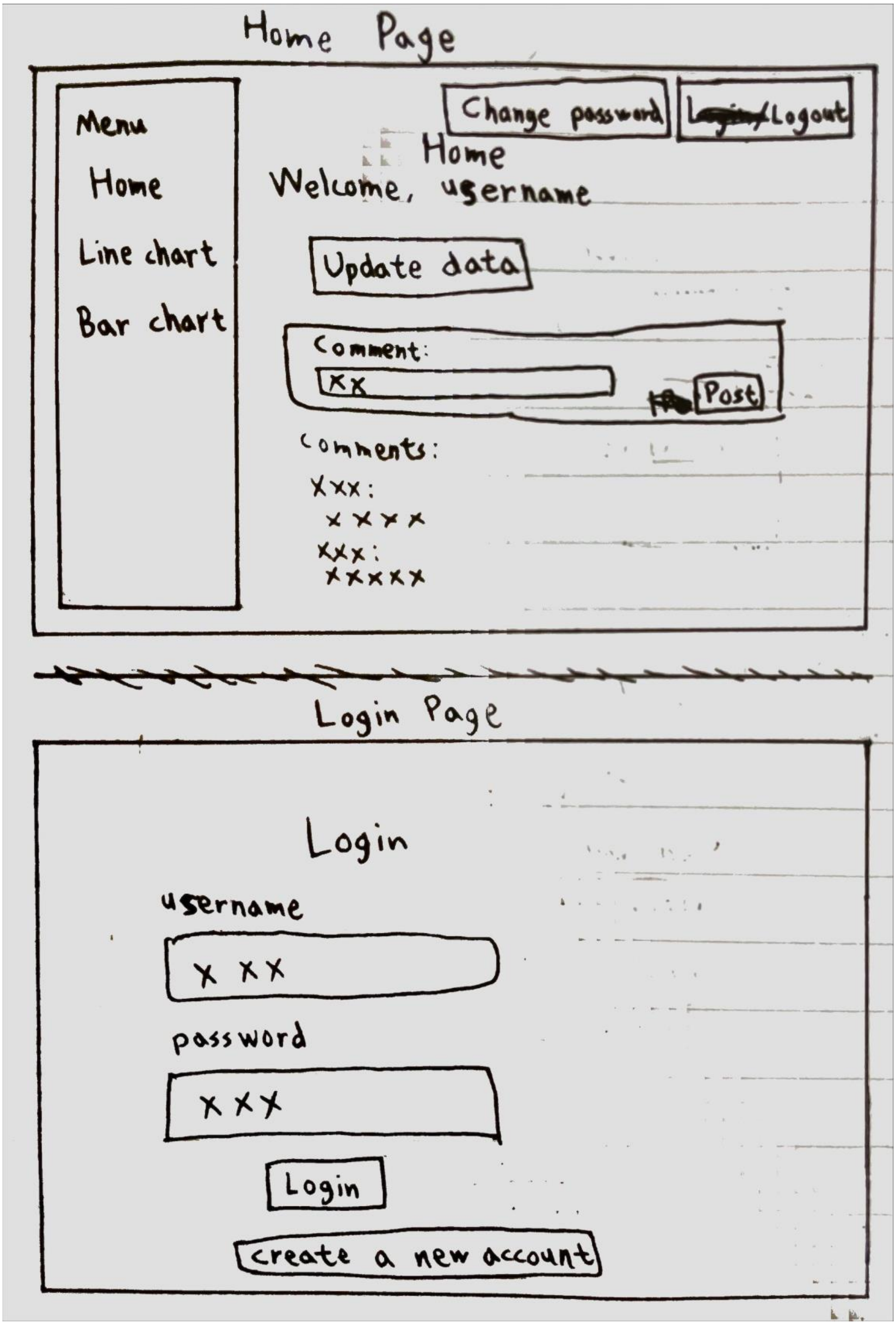


Figure 1. Handwritten wireframes for home page and login page.

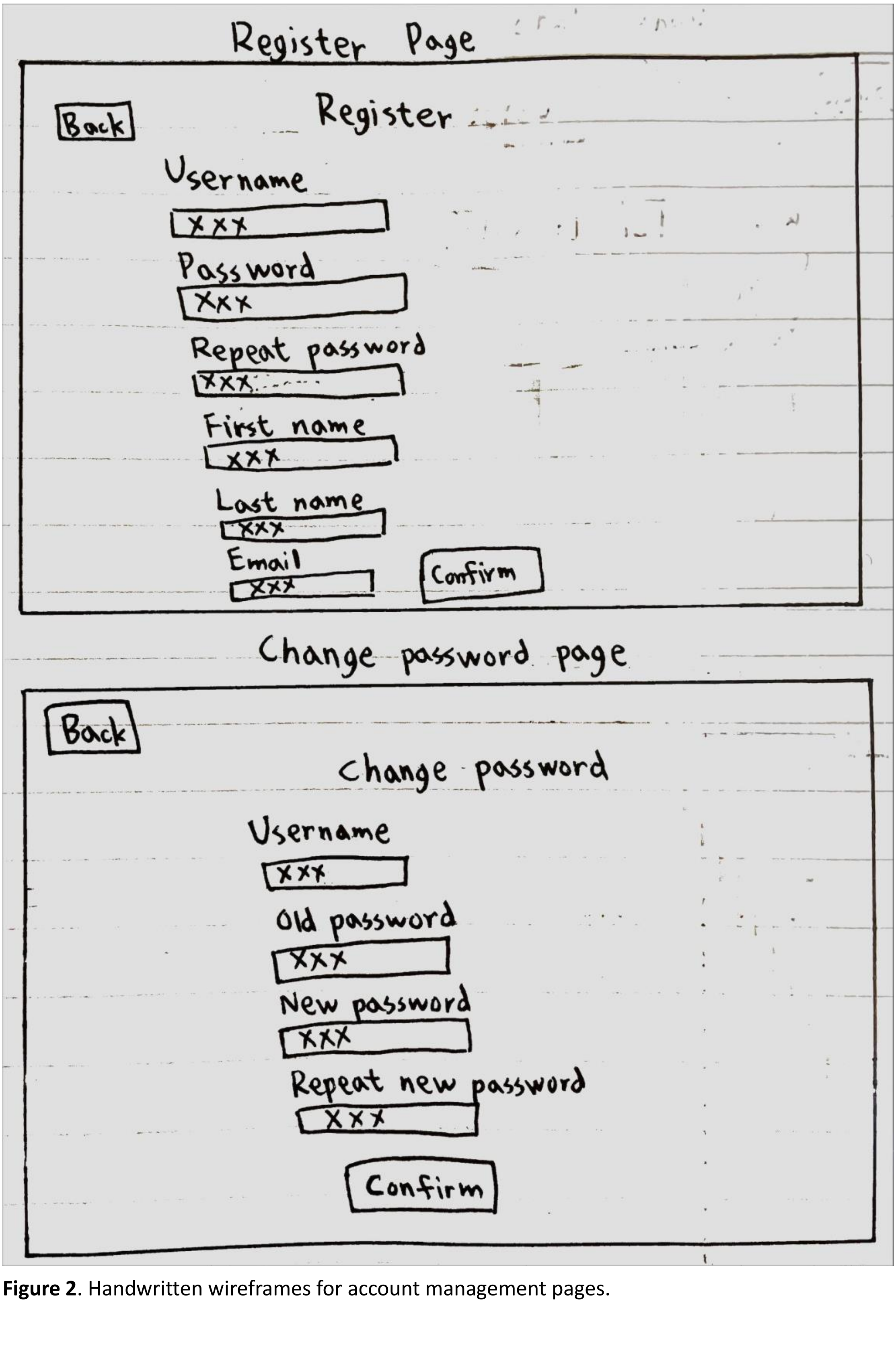


Figure 2. Handwritten wireframes for account management pages.

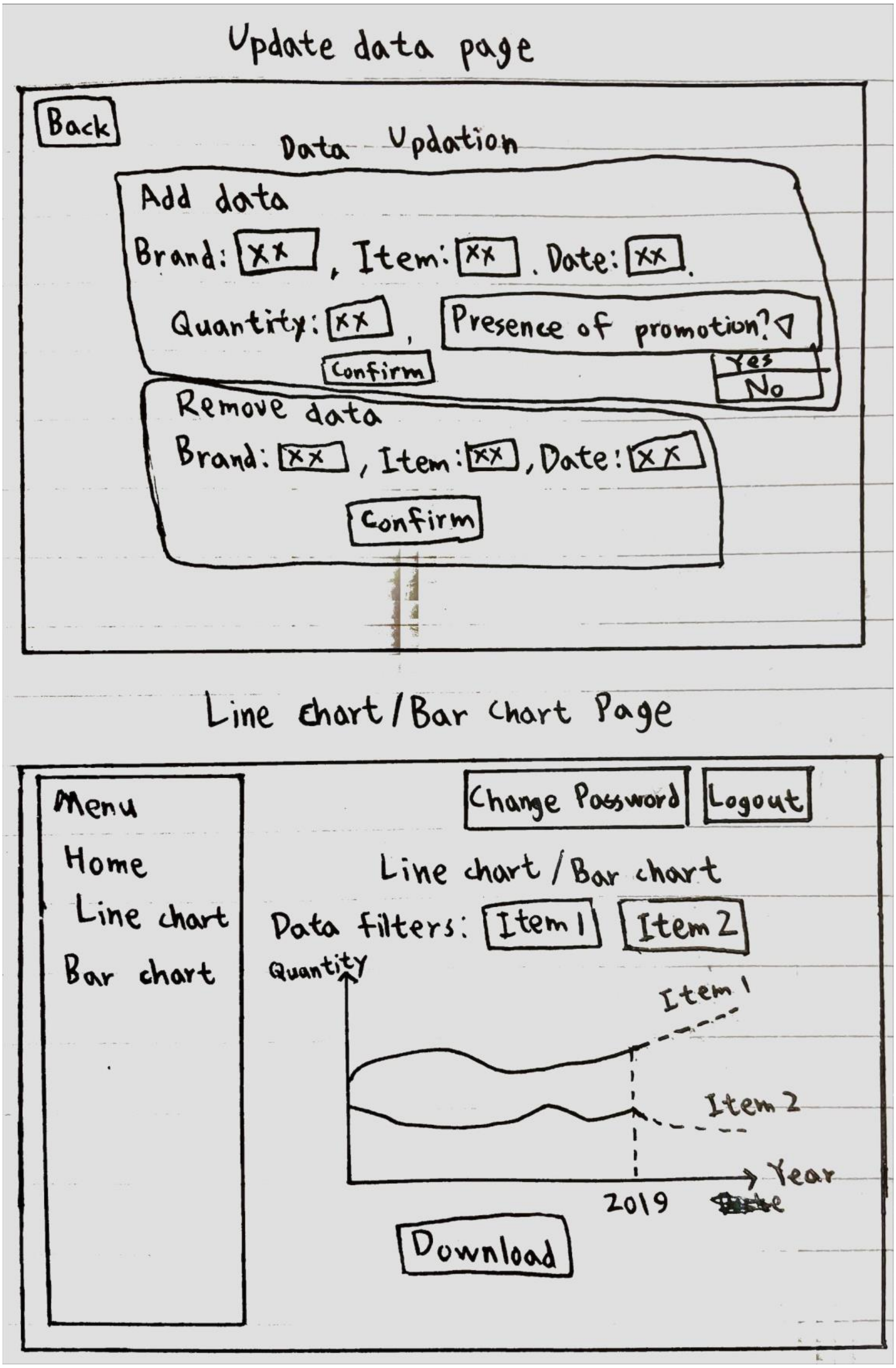


Figure 3. Handwritten wireframes for data updating page and visualisation pages.

Application design and related database design

The classes that will be used are shown in the following table.

Class name	Attributes	Methods
Account	user_id: int username: str password: str first_name: str last_name: str email: str	verify_password(str): bool reset_password(str): str

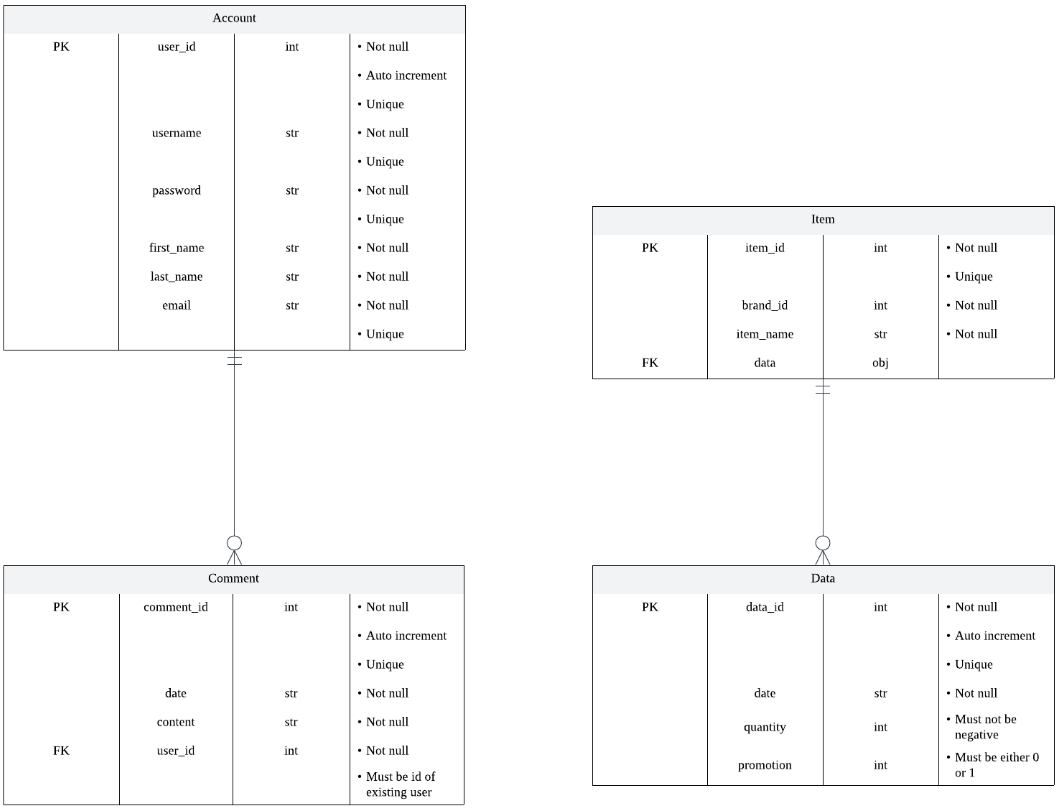
Comment	comment_id: int
	date: datetime
	content: str
	user_id: int
		post_comment(str)
		delete_comment()

Item	item_id: int
	brand_id: int
	item_name: str
	data: obj
		add_item()
		delete_item()

Data	data_id: int date: datetime quantity: int promotion: bool	

Table 2. Application design.

The logical design of the database is shown in the entity relationship diagram (ERD) below.



Account

PK	user_id	int	• Not null • Auto increment • Unique
	username	str	• Not null • Unique
	password	str	• Not null • Unique
	first_name	str	• Not null
	last_name	str	• Not null
	email	str	• Not null • Unique

Item

PK	item_id	int	• Not null • Unique
	brand_id	int	• Not null
	item_name	str	• Not null
FK	data	obj	

Comment

PK	comment_id	int	• Not null • Auto increment • Unique
	date	str	• Not null
	content	str	• Not null
FK	user_id	int	• Not null • Must be id of existing user

Data

PK	data_id	int	• Not null • Auto increment • Unique
	date	str	• Not null
	quantity	int	• Must not be negative
	promotion	int	• Must be either 0 or 1

Figure 4. Entity relationship diagram (ERD) of the database.

Since users have to log in to post comments, information about the comments including users who post these comments needs to be saved in the database. In the prepared dataset, there are different items of different brands, so the information about items themselves will be included in the "Item" table. For each item, there are two time series of data: quantity of the item and whether it is in the presence of promotion. They are both included in the "Data" table.