Verb-noun analysis
Bold = noun
Bold = verb

1. Cloud-based

- The **local app** will ensure the **user** will be able to **retrieve their accounts** and the contents from the **cloud database** onto a **new device**.
    - The app must **upload** **user data** to server to **sync** user account information accross the number of devices the user is using.

    - The app will **upload user data** to server within set time period

        - The app will **upload new changes in the users Bham calendar** if given
        - The app will **upload new changes in the users Canvas calendar** if given
        - The app will **upload user-input personal information**.
        - the app will **upload user-collected information**.
        - The app will **upload up-to-date caldendar configuration**.
    - The app will **fetch user data** from server within set time period

    - The app will **resolve changes** between **devices** to **prevent conflicts in information**

    - In case of unreachable to **cloud server**, will **attempt upload once re-established connection**.

2. **Registration** and **Login**

- The user will not be able to **access** the main functionalities of the **app** until they have either logged in or signed up for an account to log in.
    - The app will check if there is an account registered and log in, otherwise, the user will be directed to a **log-in screen** and has the option to **create an account.**

        - The **user** will log in with their **credentials**, email and password.
        - Once logged in, the user can choose to **save credientials for automatic login** next time they start the app.
        - For security reaons, user credentials are **stored in the cloud** as well as locally.
        - **New password resets** will be **uploaded to the cloud**, which is **compared against locally saved password**.

        - Unchanged local password will fail once compared to new credentials recorded in cloud server.
    - When signing up for an account, the app will direct them to a new screen.

        - The user must **provide the app with their credentials as email and password**, of which a **confirmation email** will be sent to their account to affirm the existence of email account.
        - The user can **provide the app with their Bham credentials**, of which they will be asked to login to **confirm credentials**.
            - The app will **extract the information regarding schedule** from their Bham account, and **insert it into the calendar**
        - The user can **provide the app with their Canvas credentials**,of which they will be asked to login to **confirm credentials**.
            - The app will **extract information regarding deadlines** and add to the **calendar** and **todo**
        - The user can **provide their personal information** through the **questionnaire** as part of the **sign-up process**. The user can choose to answer any number of the questions as to not turn the user away from the app through too much input given.
            - However, it is encouraged that the user provide the app which as much information as they are willing to.
            - Because this allow the app to personalize the main component.
        - The user will be able to **add or remove personal information** in their **account management** after signup (see: 6. Account Management)
    - If the user has an account but forgot the password, they can opt to **reset password** by sending it to the email they used to register their account with.

3. Calendar

- The user will be able to **modify the calendar** upon signing into their account.
    - The user can **add an event** given information on **name, date and length**, with length 0 being a deadline event.
    - The user can provide additional information on the **event**
        - **location of event**
        - the pattern of the event of if it is a **recurring event**.
    - The user can **delete an event**, **duplicate the event**, or **edit any sub-information belonging to the event**.
        - the user will be able to specify if they are modifying for all subsequent events of the same name, or just the instance of the choosen event.
- The app will to **modify the calendar** by f**etching calendar information from Bham** and **Canvas calendar**, if given access by user.
    - The app will have access to modification rights as if it was another user.

    - The app will not consult user on adding events taken from the external sources, but the user is able to treat them as an user-added event.
- The app will attempt to **modify the caldendar** based on user-collected information from both the tracker and personal information given from questionnaire
    - The calendar will **receive recommended output from AI** based on given information to schedule the calendar with **daily tasks**.
    - The user can choose to accept none to all of the recommendation the AI makes, to commit to their weekly schedule.
    - The AI will manage 2 weeks worth of recommendation of daily task, to account for changes in actual schedule which leads to uncertainty.

4. ToDo

- The todo acts as a daily representation of the calendar, and thus **syncs information between itself and the Calendar**.
- It will display it in a phone-friendly format to allow user to better see **task requirements for the day**, in a focused manner.
    - It will add **User events**, **Calendar events**, and **AI events** and **display** them for user based on current date.
- The user can **create/edit/delete** a todo in the same manner as they can an event in a calendar (see: 3.Calendar)
    - The new todo changes will be synced to calendar as an event.

5. Tracker

First time

- **Request permission to access existing tracked information** on device (e.g. Health apps)

System can

- **import tracking data** from all permitted tracking applications
- **display progress** for different time intervals (day, week, month, year)
- perform **tracking analysis** to recommend user options of improvement (see: Tracker process)
- **send reward notification** when achieving goals

User can

- **Add tracking data manually** (e.g. [+] slept x hours)
- **Remove manually entered tracking data**

Tracker process:

1. **Create recommendation based on tracking AI** (trained to recommend based on positive changes of the user?)
2. **Send notification** to user (e.g. slept sufficiently / caught up on sleep tonight)

6. Account Management

Change account login details (email/password)

1. **Enter current and new** email/password.
2. UI to change credentials
    - fields: old password, new password, confirm new password
    - confirm and cancel box

Add Phone

1. **Enter phone number**
2. **Verification code sent via sms**
3. **Enter verification code.**

Show devices

- Show the user all the devices that are logged into the account
- allow user to **log out** any of these devices so these devices must re-enter login details and get kicked out of current session

7. Behaviour Analysis AI

- The AI will **create scheduling models** based on collected information from other functionalities of the app that the user can access
    - The **AI collects information** based on changes from the Calendar (see: 3. Caldendar), User-given personal information (see: 6. Account Management), and from Tracker (see: 5. Tracker)
    - The AI will attempt to **sort the information** coming in to **produce an optimized schedule** for a period of 2 weeks.
    - The AI will **recommend modification** to it own schedule model for the week based on actual user activity throughout the day collected from tracker (see: 5.Tracker).
        - i.e recommends more sleep hours next day if User report light sleep previous day.
    - The AI will change its recommendation models based on how much of the models it recommend the user is rejected or accepted to study user preferences.
- The AI must be able to be switched on or off by user, as well as its saved profile on user preferences be deletable.

Word/Phrase	Name of class/attribute if used	Reson
local app	no	To general
user	User	Main class of the system
retrieve accounts	no	To general
cloud database	CloudDB	Class that interfaces with the cloud database.
new device	verifyNewDevice	An action executed t
upload user data	UploadToCloud	An action the system will execute, therefore it should be a method of the cloudDB class
sync	syncWithLocal	Action that will Sync the local and cloud databases, therefore it will be a method
upload new changes in the users Bham calendar	Pull	Will be used to pull data from the bham callender, method
upload new changes in the users Canvas calendar	Pull	Will be used to pull data form the canvas callender, method
upload user-input personal information	uploadToCloud	Action executed by the app, it will be a method
upload user-collected information	no	Covered by upload to cloud
upload up-to-date calendar configuration.	no	Covered by upload to cloud
fetch user data	downloadFromCloud	Used to get user data from the cloud on to a new device
devices	devices	Used to store the id of devices that have logged on to the account
resolve changes	no	To general
prevent conflicts in	no	To general

information		
attempt upload once re-established connection	no	
Registration	register	Handels the registration process, own class
Login	no	Covered by register
log-in screen	Userinterface	Part of the user interface class

create an account.	no	duplicate
user	no	duplicate
credentials	no	Tp general
save credentials for automatic login	saveCredientials	An action performed when loging in, therefore it will be a method
New password resets	DBupdatePass	Action performed when updating a password, so it will be a part of the class that handels this.
uploaded to the cloud	no	To general
compared against locally saved password	DBcontainsPass	Action to compare
provide the app with their credentials as email and password	getPassword getUsername	Action performed when performing login or registration
Confirm email	confirmEmail	An action that will be performed when making changes to the account, so it will be a method
provide the app with their Bham credentials	no	Covered by other methods
extract the information regarding schedule	pull	Action performed when pulling data from mybham, therefore it will be a method

provide the app with their Canvas credentials	no	Covered by other methods
confirm credentials	no	To general
extract information regarding deadlines	pull	Action performed when accessing data from canvas, therefore it will be a method
calendar	Calender	Component of the system, will be its own class
todo	ToDoList	Component of the system, will be its own class
provide their personal information	no	Coverd in other methods
questionnaire	Questionnaire	Component of the system, will be its own class
sign-up process	SignUp	Component of the system, will be its own class

add personal information	no	Coverd by other methods
remove personal information	removeData	Function the user will need to be able to perform therefore its a method
account management	No	To general
reset password	resetPassword	Function of the user class therefore its a method
Calendar	no	repeat
modify the calendar	no	Coverd by add and delete event
event	event	Component of the callender class, will be its own class
add an event	addNewEvent	An action that will be performed by the callender class, therefore it is a method

name	name	Attribute of the event class
date	date	Attribute of the event class
length	duration	Attribute of the event class
location of event	location	Attribute of the event class
recurring event	reccuring	Attribute of the event class
delete an event	deleteEvent	Action performed by callender class, therefore its a method.
duplicate the event	dup	Attribute of the event class
createtodo	insertNewTask	Action performed by the todo list class, therefore it is a method
edittodo	updateTask	Action performed by the todo list class, therefore it is a method
deletetodo	deleteTask	Action performed by the todo list class, therefore it is a method
Tracker	progressTracker	Component of the system, therefore it is a class
Request permission to access existing tracked information	addTracker	Action performed when adding a tracker, therefore its a method
import tracking data	dataImport	Component of the system, therefore its a class.
display progress	showTrackedData	Action, so it will be a method
tracking analysis	processTrackers	Action performed utilising the behavioral analyst ai, therefore its a method
send reward notification	showNotif	Action performed by the system therefore its a method

Add tracking data manually	trackerentry	Component of the tracker class, it will be its own class
Change account login details	changePassword	Action performed by the system therefore its a method
Enter current and new	no	Part of changePassword
Add Phone	addPhone	Action performed by the system, therefore it will be a method
Enter phone number	no	Part of add Phone
Verification code sent via sms	no	Part of addPhone
Enter verification code.	no	Part of addPhone
Show devices	showdevices	Action performed by the system therefore its a method
log out devices	manageDevices	Action performed by the system therefore its a method
Behaviour Analysis AI	Behaviour Analysis AI	Component of the system therefore it will be its own class
create scheduling models	createSchedule	Action performed by the system therefore its a method
collects information	PullData	Action performed by the system therefore its a method
sort the information	processData	Action performed by the system therefore its a method
produce an optimized schedule	no	repeat
recommend modification	no	To general
switched on or off	state	State of the ba ai, so it is an attribute of the class

<u>Responsibility Driven Analysis</u>

<u>User</u>	
Responsibilities	**Collaborators**
This class represents the user of the system, contains all of the users details, and allows them to execute any of the functions of the app.	Callender toDoList Tracker

<u>Calendar</u>	
Responsibilities	**Collaborators**
Used to store a list of events that can be displayed to the user.	Event dataHandler

<u>Event</u>	
Responsibilities	**Collaborators**
An event object stores all the information regarding one event on the user's calendar	

<u>toDoList</u>	
Responsibilities	**Collaborators**
The responsibility of this class is to provide the functionality of a todo list, allows the user to enter new tasks into the list.	Task calender

<u>Task</u>	
Responsibilities	**Collaborators**
Represent a task in the todo list, status can be changed to show completion.	

<u>AgendaContainer</u>	
Responsibilities	**Collaborators**
Superclass of the even and todo class	Event Todo

<u>loginProcess</u>	
Responsibilities	**Collaborators**
Responsible for the login process, makes sure the user has entered valid credentials, and remembers if they want to stay logged in	

<u>RegisterProccess</u>	
Responsibilities	**Collaborators**
Allows the user to create an account. Questionnaire is used to gather user information, which is verified using the verifier class and uploaded to the cloud.	User Verifier questionnaire

<u>Questionnaire</u>	
Responsibilities	**Collaborators**
Questionnaire class is used to gather information from the user.	dataHandler

<u>Behaviour Analysis AI</u>	
Responsibilities	**Collaborators**
The Behaviour Analysis AI class will use data gathered from the user and tracker to produce todos for the user	dataHandler Reward Recommendation

<u>Reinforcement</u>	
Responsibilities	**Collaborators**
Superclass of reward and recommendation	Reward Recommendation

<u>Recommendation</u>	
Responsibilities	**Collaborators**
The recommendation class will be responsible for generating new todo entries for the todoList, based on the data the BehaviourAnalysisAi processes.	TodList

<u>Reward</u>	
Responsibilities	**Collaborators**
Responsibility of the reward class is to send push notifications to the user when needed.	Reward Recommendation

<u>DataHandler</u>	
Responsibilities	**Collaborators**
Responsibility of the datahandler class is to handel data from the cloud and local databases, and interact with other parts of the system. Will make sure the cloud and local data are synced, whenever needed. It will also notify the BehaviourAnalysisAI when new data is available.	cloudData localData BehaviourAnalysisAI

<u>cloudData</u>	
Responsibilities	**Collaborators**
Interact with the cloud server	

<u>LocalData</u>	
Responsibilities	**Collaborators**
Interact with the local database	

<u>Tracker</u>	
Responsibilities	**Collaborators**
Used to represent tracked data.	apiConnection dataHandler

connectedApp	
Responsibilities	**Collaborators**
Responsibility of connected app class is to interact with a service that is connected to the app.	apiConnection dataHandler

userTracker	
Responsibilities	**Collaborators**
Used to represent a manually tracked type of data entered by the user.	

apiConnection	
Responsibilities	**Collaborators**
Used to interact with an api.	

fitbitImport	
Responsibilities	**Collaborators**
Responsibility of the fitbitImport class is to establish a connection to the fitbit api. It will access the users data stored in the fitbit databases, and store copies of it in the cloud database and the local database.	

<u>bhamImport</u>	
Responsibilities	**Collaborators**
Responsibility of the bhamImport class is to establish a connection to the users my.bham account. It will access the users timetable and pass the data to the calendar class to create events.	

<u>canvasImport</u>	
Responsibilities	**Collaborators**
Responsibility of the canvasImport class is to establish a connection to the users canvas account. It will access assignment deadlines and entries to the canvas callender and store the data on the local and cloud database.	