Intelligent Mobile Scheduling & Tracking Application

Software Engineering - Mental Health Application Proposal

Group 21

Authors

`Harry Liddall` - hxl868@student.bham.ac.uk
`Jonas Schäfer` - jxs1270@student.bham.ac.uk
`Marina Tihova` - mtt862@student.bham.ac.uk
`Phan Minh Cuong` - cxp895@student.bham.ac.uk
`Jinming Zhang` - jxz784@student.bham.ac.uk
`Zhen Su` - zxs777@student.bham.ac.uk
`Kunyong Li` - kxl706@student.bham.ac.uk

Computer Science BSc
Year 2, Autumn Term

Lecturer

`Prof. Rami Bahsoon`

College of Engineering & Physical Sciences

School of Computer Science

University of Birmingham

Birmingham, December 13, 2019

Contents

1 Proposed System

1.1 Introduction

The goal of this project is to design an **Android app** that reduces the stress an average university student experiences and assist them in managing their schedule. As reported by a Uni Health study in the UK, 82% of students in higher education report symptoms of stress or anxiety.

According to an article by the guardian, a large number of students experience poor work-life balance, which is "a huge contributing factor to mental health issues and stress". This means that many students are unable to stay focused, be sufficiently productive or avoid procrastination.
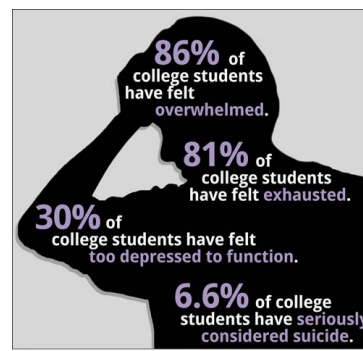

Figure 1: Mental Health (2011)

This imbalance is known to lead to mental health issues such as:

Sleep Disorder: As students either fail to work efficiently to complete coursework in time, or are unable to find a healthy work-life balance due to social or professional pressure, sleep is often neglected.

Anxiety: As students struggle to micro-manage the large number of tasks they are expected to handle on a daily basis, they lose their feeling of self-control. This can lead to a variety of mental health issues including anxiety but can lead as far as **panic attacks, depression** or **suicidal thoughts**.

The Mental Health Tracking app tries to battle these issues, which, if not addressed properly early on, can permanently damage individuals in their ability to live a healthy life and achieve future endeavours.

Our app aims to allow University of Birmingham's students to spend more time on the important tasks in their university life. It allows them to easily organise their day by automating a large part of their schedule minimal user input. Using tracked data of the user, the application additionally proposes assistance and advice if it detects potential for improvements to their lifestyle and rewards the user for reaching their goals.

This gives valuable time back to the user and gives them a feeling of being in control. The scope of this app is addressed below along with the assumption made about other APIs the app wishes to interact with.

1.2 Scope

All functionalities of the mental health app are accessible when the user logs into the app after they have created an account. As such, information about the user made available to the app includes:

- Name

- Email

- Phone number

- Preferences (recorded by a questionnaire within the sign-up process)

If given access to the user's university or canvas account, it will be able to access, extract and use the information below:

- University schedule (Tom Moses University Schedule API)

- Assignment deadlines and related events (Canvas API)

The user can also choose to give their preferences based on their classes found in the calendar, and the assignments found in canvas. For assignments with deadlines, the user can tell the system how long they think they will need to complete this tasl. Otherwise the system will allow a set default value for time allocated each week to the task.
The user will be able to use the app without needing credentials for either of their university account or their canvas account. However, doing so will limit the immediate functionality of the provided Intelligent Behaviour Analysis AI (IBA) for scheduling.

The app will collect extra information on the preferences of the user within the questionnaire. IBA will make recommendations based on these which the user can choose to edit and/or remove. The questionnaire will ask them about:

- When and how long does the user want to sleep

- Procrastination habits

- Time spent on grocery shopping

- Time spent on social activities

- Time to get ready in the morning

- Exercising goals

- Specific routines

The AI will take in all the above timing information and attempt to shuffle them within the calendar in an efficient manner, the result of which will be the base schedule every new user will be greeted with. As the user continues to interact with the app, they will be actively recommended modifications based on new inputs. The user can selectively choose which changes to include in their schedule.

Every day when the user first logs in, the app will ask them simple questions, such as for how long have they slept. The AI will then recommend a longer sleep time if the user recorded low sleep the previous day.

All collected inputs are stored locally and on the cloud servers to help sync across all user devices. As such, all transfer of information must be done securely and efficiently. The AI will plan for a month at a time. This accounts for uncertainty within students' actual schedules during the weeks and to compress user info synced to the cloud servers.

As information is sent to a shared cloud with other accounts, potential issues to concurrency and scalability should be considered, as all users will upload their data to the servers. Therefore, the servers will handle all accounts' synchronisation requests in a reliable and efficient manner.

Assumptions for the app

- The user of the app must create an account with a valid email to be able to receive periodic notification, and verify its account.

- It is assumed that data transfer is scalable proportional to users once the system is set in place as new features may be introduced, increasing data transfer size.

- It is assumed that the user will provide their own information, either manually setting their own calendar, or giving the app their preferences and letting the system set the calendar for them.

- It is assumed that IBA will always produce an efficient calendar catered to the user's needs and preferences depending on the amount of information the user enters.

2 Requirements Analysis

2.1 Functional Requirements

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 across the number of devices the user is using.
  - The app will upload all user data to server

    * The app will upload questionnaire answers and preferences.
    * The app will upload user-collected information.
    * The app will upload API and login credential information.

  - The app will fetch user data from server
  - The app will fetch data from connected APIs

    * The app will download new changes from the user's **Bham** calendar if given account credentials. (see: 6. Account Management)
    * The app will download new changes from the user's **Canvas** account if given account credentials. (see: 6. Account Management)

  - The app will resolve changes between devices to prevent conflicts with 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.
    * Upon loggin in, the user will choose to save credentials for automatic login next time they start the app.
    * For security reasons, user credentials are stored in the cloud (see: 1. Cloud-based) as well as locally.
    * New password resets will be uploaded to the cloud, which is compared against locally saved password.
    * The unchanged local password will fail from now on when being compared to new credentials on the 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 (email and password), and then a confirmation email will be sent to their email address in order to confirm the user-email.
    * The user must confirm the email to verify registration.

* The user can provide the app with their Bham credentials and if done so:
  · The app will extract the information regarding schedule from their Bham account, and insert it into the calendar (see: 3. Calendar)
* The user can provide the app with their Canvas credentials and if done so:
  · The app will extract information regarding deadlines and will add it to the calendar and todos (see: 3. Calendar, 4.Todo)
* The user can provide their personal information through a questionnaire as part of the sign-up process. The user can choose to skip any of the given questions.
  · The questions from the questionnaire will be used to tailor the Behaviour Analysis AI output to the user needs. (see: 7. AI)
* 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 has forgotten the password, they can opt to reset password by receiving a change password hyperlink 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 all required 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 chosen event.

- The calendar will be modified by fetching calendar information from Bham and Canvas 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 user-added events and modify them.

- The app will attempt to modify the calendar (the user can reject the changes) based on user-collected information from both the trackers (see: 5. Tracker) and personal information given from questionnaire (see: 6. Account Management) using the integrated AI (see: 7. AI).

  – The calendar will receive recommended output from AI based on given information to schedule the calendar with daily tasks (such as exercise, study etc.).
  – The user can choose to accept none to all of the recommendation the AI makes.
  – The AI will offer only 2 weeks worth of event recommendations because it needs to account for changes in the already existing schedule which can lead 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 the 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

- The tracker will collect information from health trackers either built-in on device from applications, or external fitness trackers, and user input within the app if given.

  – The app will request permission to access existing tracked information on device and extract health information.

  – The user can input tracking data themselves to the app.

  – The app will display data for different time intervals about user health.

  – The app will ask the user to give tracking data on their daily activities, such as sleep lengths.

  – The user can choose to edit tracking data.

  – The information collected through the tracker will be passed onto the AI (see: 7. AI) to create recommendations to improve the user's health.

  – The tracker will send notifications to user about their health progress.

6. Account Management

- The user will be able to review their account information, to either add, edit, remove information.

  – The user can edit the questionnaire information that they provided when signing up.

  – The user can edit their credentials, password and emails to change them.

  – The user can see all the devices that are accessing the account.

  – The user can log out of the app, which transfer to log-in screen.

    ∗ The user can log out of all devices.

- The user will be able to review and edit preferences regarding other functionalities of the app

  – The user can choose to turn on or off the AI (see: 7.Behaviour Analysis AI)

  – The user can choose to delete AI stored information about user.

- The user can choose to sync cloud data manually.
- The user can choose to set sync period, or disable sync. it'll be set to a default value otherwise.

- The app will be able to make use of the information given by the user. The user must be able to set how much information the app can use.

7. **Intelligent Behaviour Analysis AI (IBA)**

- IBA will create scheduling models based on collected information from other functionalities of the app that the user can access

    - IBA collects information based on changes from the Calendar (see: 3. Calendar), User-given personal information (see: 6. Account Management), and from Tracker (see: 5. Tracker)
    - IBA will attempt to sort the information coming in to produce an optimised schedule for a period of 2 weeks.
    - IBA will recommend modifications to its own schedule model for the week based on actual user activity throughout the day collected from trackers (see: 5.Tracker).
        * i.e. recommends more sleep hours for the following day if User reports light sleep for the previous day.
    - IBA will change its recommendation models based on how much of the models it recommends are rejected or accepted in attempt to study user preferences.

- IBA can be switched on or off by the user in Account Management (see: 6. Account Management)

- IBA's saved preferences can be deleted by the user in Profile Management (see: 6. Account Management)

2.2 Non Functional Requirements

1. Synchronisation performance:

- The app shall upload user data to the server within 3 seconds

- The app shall fetch user data from server within 3 seconds (based on >1MB/s download speed of user)

2. Security

- The app will implement hybrid cryptography for secure data transfer between cloud server and local app process used by user.

  - The app must ensure data communications with external sources are either local or also secure. Unless necessary, all communications are primarily between local app and cloud server.
    * Tracker applications and devices will be accessed without using any interaction between local app and cloud server.
    * Tracker devices that rely on Bluetooth technology must also encrypt all transfer of communication between the device and the app.
    * Fetching calendar/deadlines information from Bham API and Canvas API rely on the security of the respective systems.
    * Credentials must be encrypted before transferring to API to avoid access-points into app or respective accounts of Bham and Canvas.

3. Reliability

- The app should be designed to account for possible errors and failures in the components and attempt to address it without hurting user experience or minimising it.

  - The AI should be designed and stress tested in mind to handle any and all changes in the app components.
  - The app should be designed to sync information from cloud coming from older versions of the app and updating the local app accordingly.
  - Conversely, components within the local app must be designed so that, once newer versions of the app are introduced with changes to its components, it must be able to interpret information from older versions of app without error. The app will only modify information based on changes to component between versions.

- The cloud server should be designed with redundancy in mind, syncing with back-up server(s) after every set uploads from all users to account for possible failure from either servers.

4. Scalability

- The app should be constructed that an increase of users does not adversely affect the user experience of the individual user.

  - Be designed that additional servers can be added or removed seamlessly and proportionally handle requests from users as user base increases or decreases.

10

– Load balancers control server traffic to prevent server overload. Cloud server maintains control over sync request from local app, and can issue earlier sync requests to reduce expected spikes in data due to set upload time, or divert sync request to additional identical servers.

5. Efficiency

- Actions the users may take must take a minimal amount of processing time of 1 second within the app.

  – Communications to external API i.e. Canvas or Bham must not add on significant waiting time than based on the users internet connection speed.
  – All background tasks should be done in the background and thus can take more time, but must finish as soon as possible
    * Server syncs should compare information between local app and cloud server to upload the smallest possible data.
    * Behavioural Analysis AI should minimise time complexity given increase in information coming in to efficiently create recommendation models.
      · The BAI should compare incoming information with current one that it is actively using, and only pass new information to update recommendation models, rather than recreating them each time.

6. Maintainability

- The app should be able to operate with minimal human oversight

  – The BAI should be able to run autonomously without direct user interaction and run effectively with any given amount of information collected.
  – The cloud servers should be able to operate either indecently from other servers in case of required server maintenance and updates.
  – The cloud server should be set up to accept sync requests from older versions of the app as it only handles incoming packages of user information.

7. Accessibility and Usability

- All of the app should be able to be used effectively with minimal instructions, and is intuitive to the user. The app should also be accessible by people with disabilities.

  – The design of the UI must take into account of colour blindness so that people with them are not confused.
  – The questions asked by the questionnaire should be short, understandable, concise.
  – General word descriptors of the app functionalities should be short, understandable, concise.
  – Aim to display the app in mostly visuals instead of words.

3 Use Case

3.1 Comprehensive Use Case Diagram
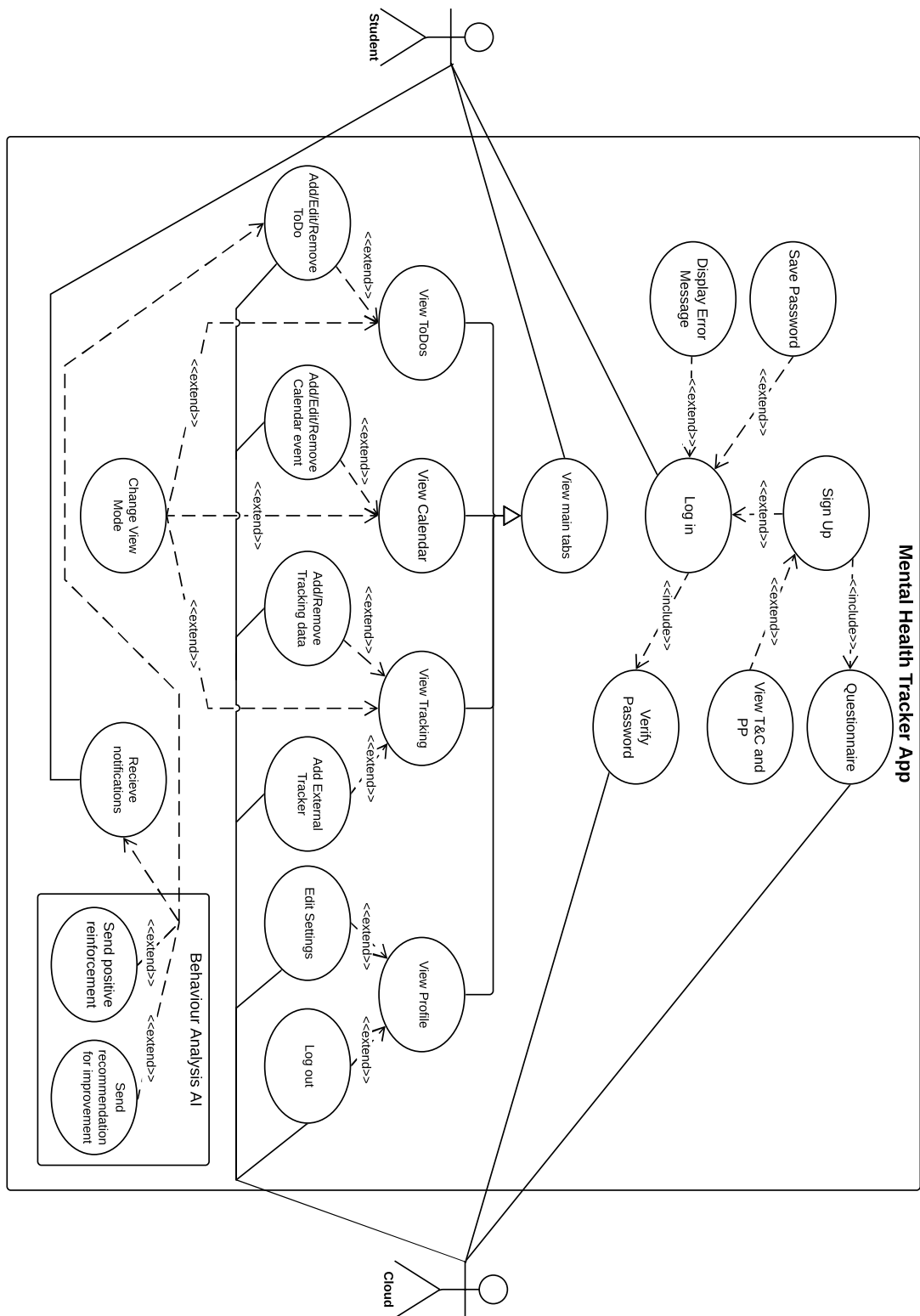
3.2 Non-trivial Use Case: Add a tracker

Precondition

1. User has created an account.

2. User has logged into their account.

3. User has selected tracker tab.

4. (User selects add a tracker option from the tracker page.)

Flow of Events

1. User selects Add a tracker option from the Tracking page.

2. List of trackers that it is possible to import data from is shown to user.

   (a) If the user selects an item from the list, they are redirected to an authentication page.

   (b) If the user selects cancel, then the user is asked to confirm their decision, after which they are returned to the tracker page or prompted to select a tracker from the list.

3. Once on the authentication page the user will be asked to give permission to allow the app to access their data from the selected tracking service.

   (a) If the user accepts, they are shown a "gathering information" page

   (b) If the user rejects the authentication they are returned to the list of trackers.

4. Request is sent to the service the user selected

5. (a) If the request is declined/fails user is shown a box which tells them the data import failed, and to try again later, then they are returned to the select tracker screen.

   (b) If request is accepted, then a token is returned which is used to get data using the chosen services api.

6. Request made to selected service.

   (a) If errors occur with the requests they are tried again x times, if continued errors the process will be cancelled and the user will be returned to the select a tracker screen and shown a message telling them an error occurred.

   (b) If no errors occur, then the system continues.

7. The data collected is put into apps databases etc to be viewable by the user in the app.

   (a) The data will be handled differently depending on the app it was collected from.

8. User data stored on the cloud gets updated.

9. Once the process is complete the user is returned to the select tracker page and told that the import was a success.

Post Conditions

1. System updates the user's information, using the new data imported from the added tracker, if a tracker was added.

2. The system returns to an idle state and waits for the next input.

Actors

The user is the main actor in this use case as they will initiate the case by selecting the tracking tab in the app, and the tracking application that the data will be collected from. Another actor will be the cloud as the user data will be updated if the user makes any changes.

Scenarios

1. User A selects "Add a tracker" option from the tracker page and is prompted to select the tracker that they would like to import data from. They select Fitbit from the list of options. The user is redirected to the Fitbit OAuth 2.0 authorization page where the user selects "allow all" to give the app permission to collect all of their data from Fitbit. HTTP request is made to access the user's data. User data is pulled from the server and imported to the in-app tracker. Data is added to local databases for use by other parts of the app etc. Data uploaded to cloud database. user is shown a message confirming their data has been imported and is returned to the selection screen.

2. User B selects "add a tracker" option from the tracker page in is prompted to select the tracker that they would like to import data from. They select Fitbit from the list of options. The user is redirected to the Fitbit OAuth 2.0 authorization page where the user selects "deny all" User is sent back to the tracker page. System waits for user input.

4 Activity Diagram

User | Local System | Server | AI

Visit Login / Registration page

Enter the username & password

Receive Confirmation E-mail

Click On Confirmation Link

Write Questionnaires

Add Local Data

Saved Questionnaires

[Not-Registered]

[Login Invalid]

[Registered]

Enter the Email & password &some details

Save Uer's Username & Password

[Save User]

[Not Save User]

[Login Valid]

Logged in

View App Main Page

View Calender

View ToDos

View Tracking

View Profile

Add Event

Delete Event

Add Event

Delete Event

Delete Tracker

Delete Tracker

Change Login info

Add Phone

Show Devices

Display Devices

Store Phone Info

Renew info

View Devices

update user info

Delete/Add Local Data

Sync Cloud Data

Analyze User Data

View Notifications

Display Notifications

Receive AI info and Send Info to Local

Send Notifications

Click Quit App

Receive Requirement

Save Local Data

Sync Local Data

5 Class Analysis

5.1 Verb-noun Analysis

Text: noun

<u>Text:</u> verb

1. Cloud-based

- The **local app** will ensure the **user** will be able to <u>**retrieve their accounts**</u> and the contents from the **cloud database** onto a **new device**

    – he app must <u>**upload user data**</u> to server to <u>**sync**</u> user account information across the number of devices the user is using

    – The app will <u>**upload user data**</u> to server within set time period

        * The app will <u>**upload new changes in the user's Bham calendar**</u> if given
        * The app will <u>**upload new changes in the user's Canvas calendar**</u> if given
        * The app will <u>**upload user-input personal information**</u>
        * The app will <u>**upload user-collected information**</u>
        * The app will <u>**upload up-to-date calendar configuration**</u>

    – The app will <u>**fetch user data**</u> from server within set time period

    – The app will **resolve changes** between **devices** to <u>**prevent conflicts in information**</u>

    – In the case of "unreachable to **cloud server**", it will <u>**attempt to upload once re-established connection**</u>.

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 <u>**create an account**</u>.

        * The **user** will log in with their **credentials**, email and password.
        * Once logged in, the user can choose to <u>**save credentials for automatic login**</u> next time they start the app.
        * For security reasons, user credentials are <u>**stored in the cloud**</u> as well as locally.
        * **New password resets** will be <u>**uploaded to the cloud**</u>, which is <u>**compared against locally saved password**</u>.
        * 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 <u>**provide the app with their credentials as email and password**</u>, of which a **confirmation email** will be sent to their account to affirm the existence of email account.
        * The user can <u>**provide the app with their Bham credentials**</u>, of which they will be asked to login to <u>**confirm credentials**</u>.

· 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 questions as to not turn the user away from the app by requesting this data to function.

· However, it is encouraged that the user provide the app which as much information as they are willing to.

· This allows 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 **fetching 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 calendar** 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:

- **Create recommendation based on tracking AI** (trained to recommend based on positive changes of the user?)

- **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. **Intelligent Behaviour Analyser AI (IBA)**

   - IBA will **create scheduling models** based on collected information from other functionalities of the app that the user can access

     - **IBA 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)
     - IBA will attempt to **sort the information** coming in to **produce an optimized schedule** for a period of 2 weeks.
     - IBA 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 the user report short sleep the previous day.
     - IBA will change its recommendation models based on how much of the models it recommend the user is rejected or accepted to study user preferences.

   - IBA 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	Reason
local app	no	To general
user	User	Main class of the system
retrieve accounts	no	To general
cloud database	CloudData	Class that interfaces with the cloud database.
new device	no	To general
upload user data	UploadToCloud	An action the system will execute, therefore it should be a method of the cloudDB class
sync	sync	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 Calendar, method
upload new changes in the users Canvas calendar	Pull	Will be used to pull data from the canvas calendar, method
upload user-input personal information	SecureDataHandler	Clas to handle data between cloud and local databases
upload user-collected information	no	Covered by sync
upload up-to-date calendar configuration.	no	Covered by sync
fetch user data	fetch	Used to get user data from the cloud on to a new device
devices	no	To general
resolve changes	no	To general
prevent conflicts in information	no	To general
attempt upload once re-established connection	no	
Registration	RegisterProcess	Handels the registration process, own class
Login	LoginProcess	Class that will be responsible for handling the login process
log-in screen	no	Part of the LoginProcess

create an account.	registerUser	Function executed by the system therefore it will be an method
user	no	duplicate
credentials	no	Tp general
save credentials for automatic login	savePassword	An action performed when logging in, therefore it will be a method
New password resets	no	Covered by other methods
uploaded to the cloud	no	Already covered
compared against locally saved password	no	Covered by other methods
provide the app with credentials as email and password	regiserUser loginUser	Action performed when performing login or registration
Confirm email	no	Covered by loginuser and registerUser
provide the app with their Bham credentials	no	Covered by other methods
extract the information regarding schedule	pull	Action performed when pulling data from my bham, therefore it will be a method
provide the app with their Canvas credentials	no	Covered by other methods
confirm credentials	no	Too general
extract information regarding deadlines	pull	Action performed when accessing data from canvas, therefore it will be a method
calendar	Calendar	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	Covered in other methods
questionnaire	Questionnaire	Component of the system, will be its own class
sign-up process	no	Already covered

add personal information	no	Covered 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	Covered by add and delete event
event	event	Component of the calendar class, will be its own class
add an event	addNewEvent	Action performed by the calendar 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	recurring	Attribute of the event class
delete an event	deleteEvent	Action performed by Calendar class, therefore its a method.
duplicate the event	dup	Attribute of the event class
create todo	insertTodo	Action performed by the todo list class, therefore it is a method
edit todo	updateTodo	Action performed by the todo list class, therefore it is a method
delete todo	deleteTodo	Action performed by the todo list class, therefore it is a method
Tracker	TrackingArea	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	ApiConnection	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	Rewared	Type of notification, therefore it will be a class

Add tracking data manually	TrackingData	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 the verification code.	no	Part of addPhone
Show devices	show devices	Action performed by the system therefore its a method
Log out devices	manageDevices	Action performed by the system therefore its a method
Intelligent Behaviour Analyser	IntelligentBehaviourAnalyser	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	Analyse	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 IBA, so it is an attribute of the class

Responsibility Driven Analysis

User	
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.	Calendar ToDoList Tracker

Calendar	
Responsibilities	**Collaborators**
Used to store a list of events that can be displayed to the user.	Event SecureDataHandler

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

ToDoList	
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.	Todo Calendar

Todo	
Responsibilities	**Collaborators**
Represent a task in the todo list, status can be changed to show completion.	

AgendaContainer	
Responsibilities	**Collaborators**
Superclass of the even and todo class	Event Todo

LoginProcess	
Responsibilities	**Collaborators**
Responsible for the login process, makes sure the user has entered valid credentials, and remembers if they want to stay logged in	User SecureDataHandler

RegisterProccess	
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

Questionnaire	
Responsibilities	**Collaborators**
Questionnaire class is used to gather information from the user.	RegisterProcess

IntelligentBehaviourAnalyser	
Responsibilities	**Collaborators**
The IntelligentBehaviourAnalyser class will use data gathered from the user and tracker to produce todos for the user	SecureDataHandler Reward Recommendation

Reinforcement	
Responsibilities	**Collaborators**
Superclass of reward and recommendation	Reward Recommendation

Recommendation	
Responsibilities	**Collaborators**
The recommendation class will be responsible for generating new todo entries for the todoList, based on the data the IntelligentBehaviourAnalyser processes.	TodoList

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

SecureDataHandler	
Responsibilities	**Collaborators**
Responsibility of the Datahandler class is to handle 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 IntelligentBehaviourAnalyser when new data is available.	CloudData LocalData IntelligentBehaviourAnalyser

CloudData	
Responsibilities	**Collaborators**
Interact with the cloud server	

LocalData	
Responsibilities	**Collaborators**
Interact with the local database	

TrackingArea	
Responsibilities	**Collaborators**
Used to represent tracked data.	TrackingData

TrackingData	
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.	

BhamImport	
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.	

CanvasImport	
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 Calendar and store the data on the local and cloud database.	

5.2 First-Cut Class Diagram

5.3 Detailed Class Diagram

FitbitImport
- url: URL
- username: String
- password: String
+ establishLink(): void
+ pull(): File

CanvasImport
- url: URL
- username: String
- password: String
+ establishLink(): void
+ pull(): File

BhamImport
- url: URL
- username: String
- password: String
+ establishLink(): void
+ pull(): File

<<ApiConnection>>
+ establishLink(): void
+ pull(): File

Note that APIs have a variety of ways to connect. This generalises it.

TrackingData
- data: Object
- timestamp: DateTime

TrackingArea
- title: String
- dataType: Class
- data: List<TrackingData>
- lastUpdated: DateTime
+ verifyValidData(TrackingData): boolean
+ addData(TrackingData): void
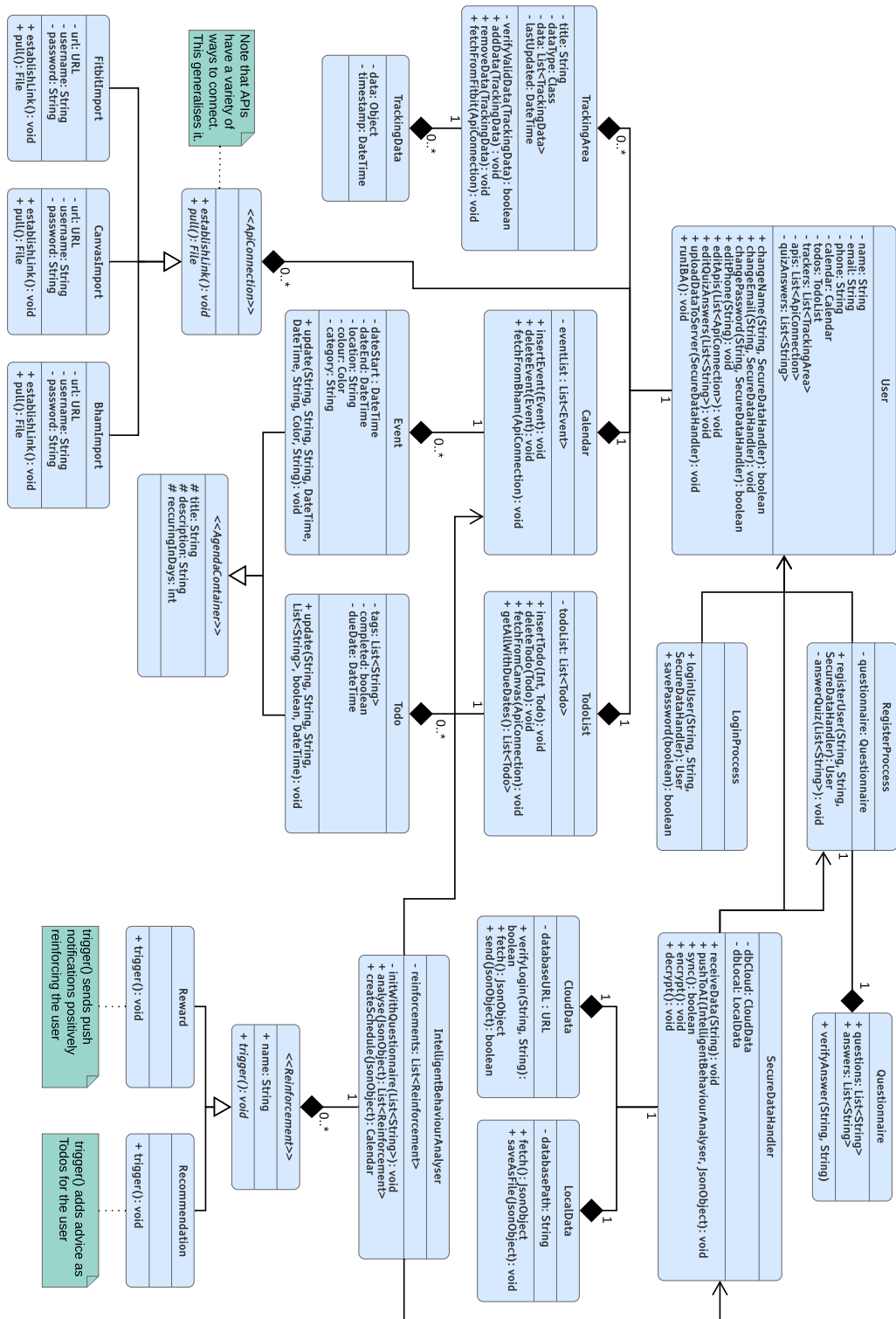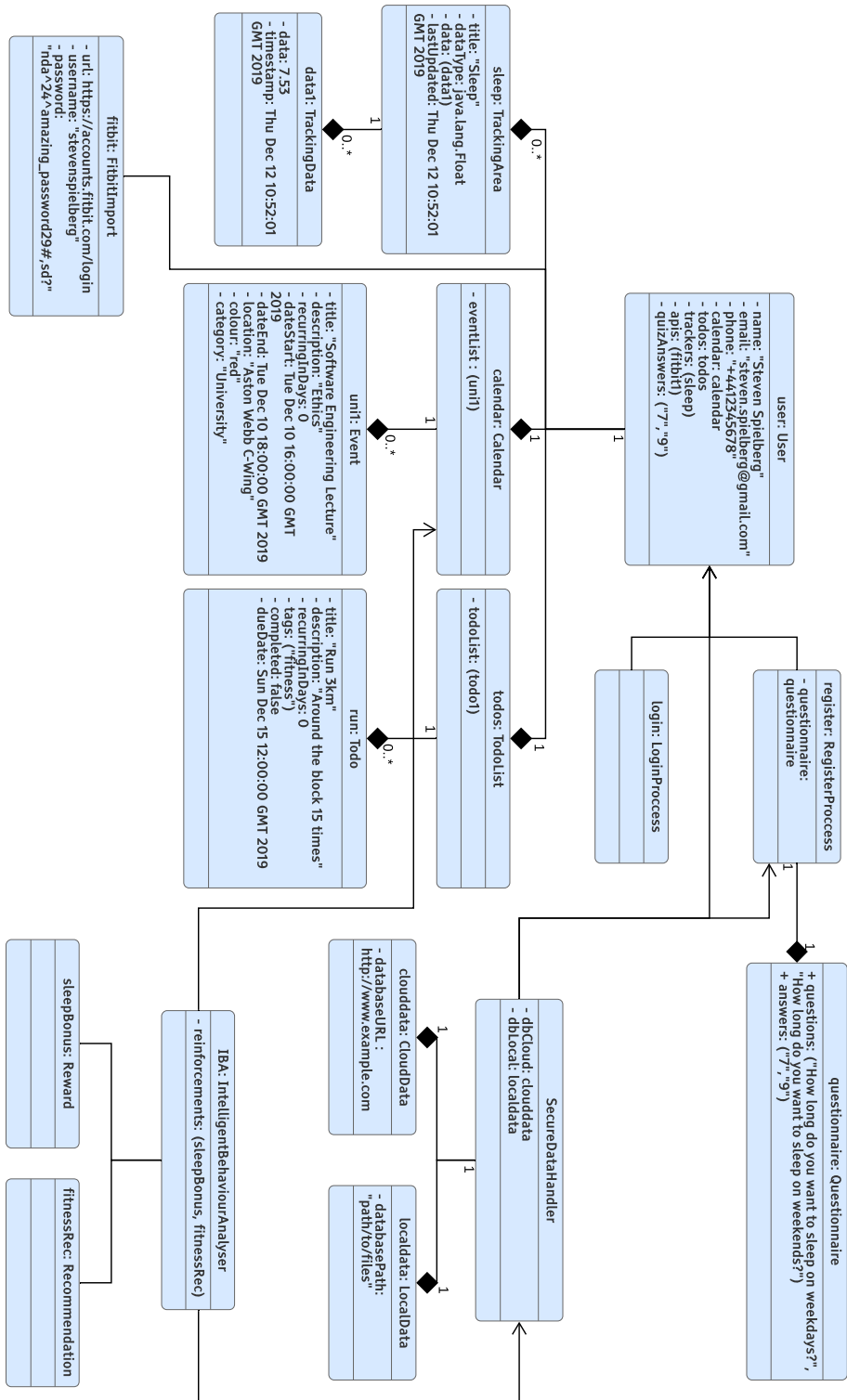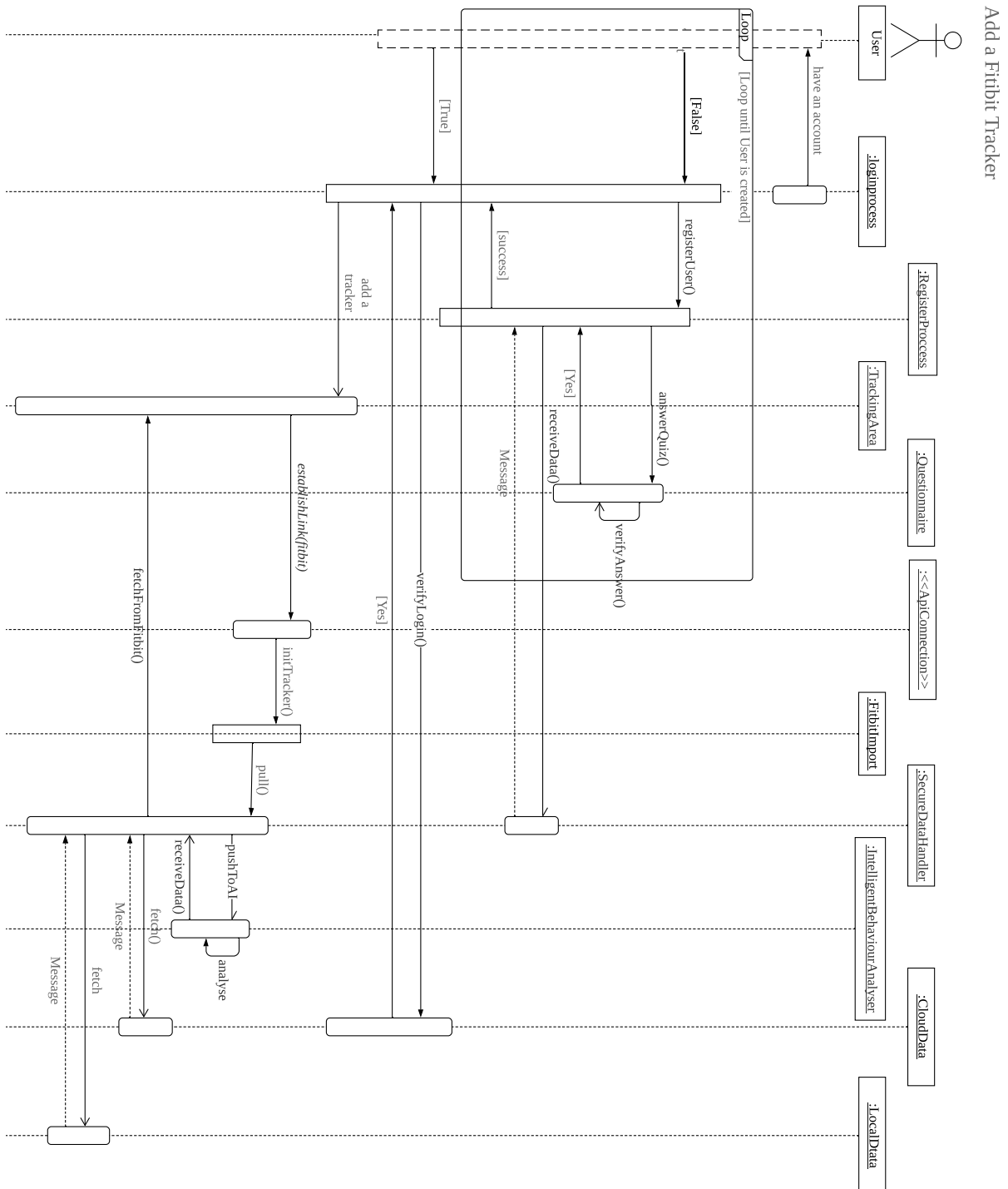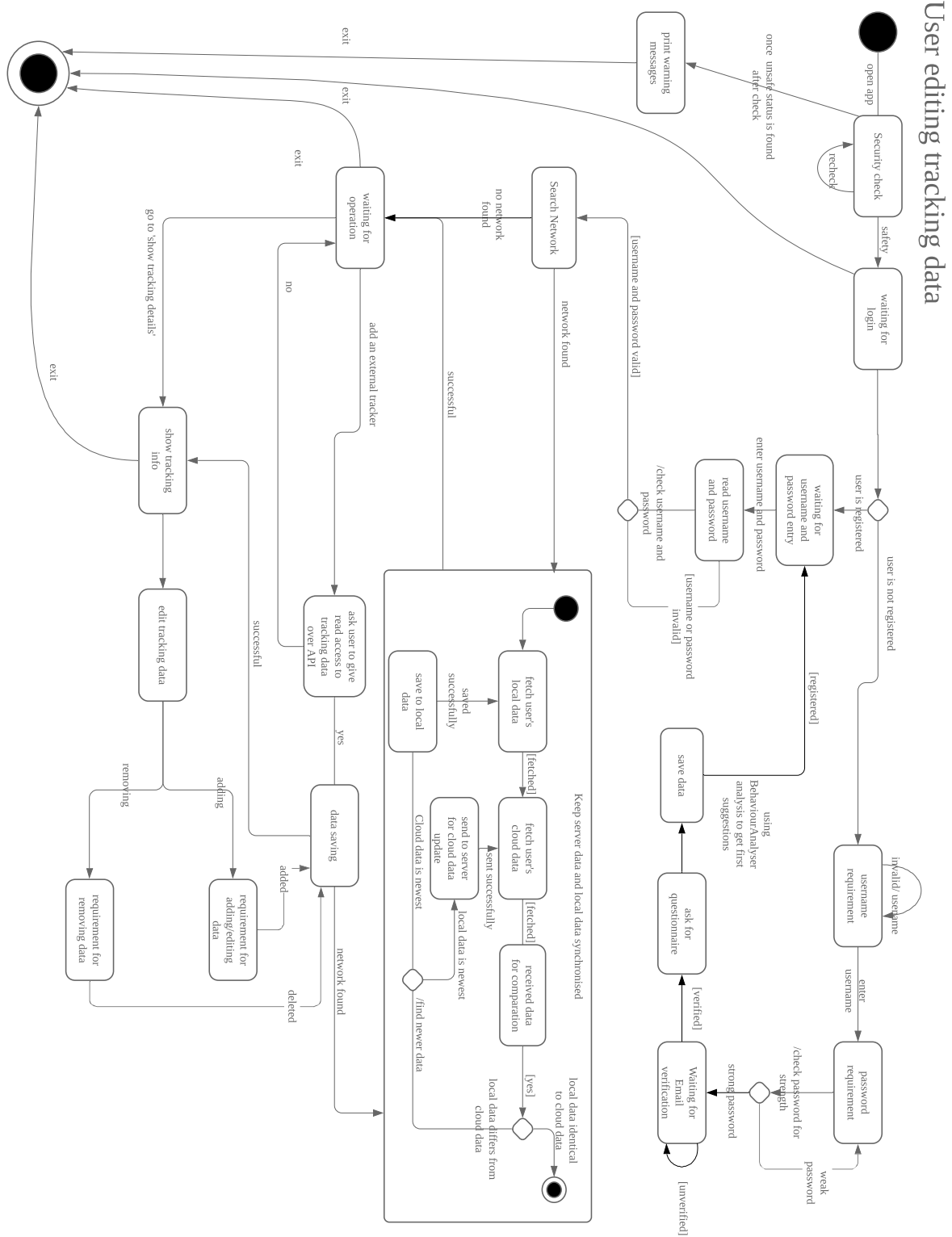+ removeData(TrackingData): void
+ fetchFromFitbit(ApiConnection): void

User
- name: String
- email: String
- phone: String
- calendar: Calendar
- todos: TodoList
- trackers: List<TrackingArea>
- apis: List<ApiConnection>
- quizAnswers: List<String>
+ changeName(String, SecureDataHandler): boolean
+ changeEmail(String, SecureDataHandler): void
+ changePassword(String, SecureDataHandler): boolean
+ editPhone(String, SecureDataHandler): void
+ editApis(List<ApiConnection>): void
+ editQuizAnswers(List<String>): void
+ uploadDataToServer(SecureDataHandler): void
+ runIBA(): void

Calendar
- eventlist : List<Event>
+ insertEvent(Event): void
+ deleteEvent(Event): void
+ fetchFromBham(ApiConnection): void

Event
- dateStart : DateTime
- dateEnd : DateTime
- location: String
- colour: Color
- category: String
+ update(String, String, DateTime, DateTime, String, Color, String): void

<<AgendaContainer>>
title: String
description: String
recurringInDays: int

Todo
- tags: List<String>
- completed: boolean
- dueDate: DateTime
+ update(String, String, String, List<String>, boolean, DateTime): void

TodoList
- todolist: List<Todo>
+ insertTodo(1Int, Todo): void
+ deleteTodo(Todo): void
+ fetchFromCanvas(ApiConnection): void
+ getAllWithDueDate(): List<Todo>

LoginProcess
+ loginUser(String, String, SecureDataHandler): User
+ savePassword(boolean): boolean

RegisterProccess
- questionnaire: Questionnaire
+ registerUser(String, String, SecureDataHandler): User
- answerQuiz(List<String>): void

Questionnaire
+ questions: List<String>
+ answers: List<String>
+ verifyAnswer(String, String)

CloudData
- databaseURL : URL
+ verifyLogin(String, String): boolean
+ fetch(): JsonObject
+ send(JsonObject): boolean

LocalData
- databasePath: String
+ fetch(): JsonObject
+ saveAsFile(JsonObject): void

SecureDataHandler
- dbCloud: CloudData
- dbLocal: LocalData
+ receiveData(String): void
+ pushToAI(IntelligentBehaviourAnalyser, JsonObject): void
+ sync(): boolean
+ encrypt(): void
+ decrypt(): void

IntelligentBehaviourAnalyser
- reinforcements: List<Reinforcement>
- initWithQuestionnaire(List<String>): void
+ analyse(JsonObject): List<Reinforcement>
+ createSchedule(JsonObject): Calendar

<<Reinforcement>>
+ name: String
+ trigger(): void

Reward
+ trigger(): void

trigger() sends push notifications positively reinforcing the user

Recommendation
+ trigger(): void

trigger() adds advice as Todos for the user

30

6 Object Diagram

sleep: TrackingArea
- title: "Sleep"
- dataType: java.lang.Float
- data: (data1)
- lastUpdated: Thu Dec 12 10:52:01 GMT 2019

data1: TrackingData
- data: 7.53
- timestamp: Thu Dec 12 10:52:01 GMT 2019

fitbit: FitbitImport
- uri: https://accounts.fitbit.com/login
- username: "stevenspielberg"
- password: "nda^24^amazing_password29#,sd?"

user: User
- name: "Steven Spielberg"
- email: "steven.spielberg@gmail.com"
- phone: "+441234456778"
- calendar: calendar
- todos: todos
- trackers: (sleep)
- apis: (fitbit)
- quizAnswers: ("7","9")

calendar: Calendar
- eventList: (unit1)

unit1: Event
- title: "Software Engineering Lecture"
- description: "Ethics"
- recurringInDays: 0
- dateStart: Tue Dec 10 16:00:00 GMT 2019
- dateEnd: Tue Dec 10 18:00:00 GMT 2019
- location: "Aston Webb C-Wing"
- colour: "red"
- category: "University"

todos: TodoList
- todoList: (todo1)

run: Todo
- title: "Run 3km"
- description: "Around the block 15 times"
- recurringInDays: 0
- tags: ("fitness")
- completed: false
- dueDate: Sun Dec 15 12:00:00 GMT 2019

login: LoginProccess

register: RegisterProccess
- questionnaire: questionnaire

questionnaire: Questionnaire
+ questions: ("How long do you want to sleep on weekdays?", "How long do you want to sleep on weekends?")
+ answers: ("7", "9")

SecureDataHandler
- dbCloud: clouddata
- dbLocal: localdata

clouddata: CloudData
- databaseURL : http://www.example.com

localdata: LocalData
- databasePath: "path/to/files"

IBA: IntelligentBehaviourAnalyser
- reinforcements: (sleepBonus, fitnessRec)

sleepBonus: Reward

fitnessRec: Recommendation

7 Sequence Diagram

8 State Diagram

User editing tracking data

open app

Security check
recheck
safety

print warning messages

once unsafe status is found after check

exit

exit

exit

exit

exit

go to 'show tracking details'

waiting for login

user is registered

user is not registered

waiting for username and password entry

read username and password

enter username and password

/check username and password

[username or password invalid]

[username and password valid]

[registered]

using BehaviourAnalyser analysis to get first suggestions

save data

ask for questionnaire

[verified]

Waiting for Email verification

[unverified]

username requirement

invalid/ username

enter username

password requirement

/check password for strength

strong password

weak password

Search Network

no network found

network found

waiting for operation

add an external tracker

ask user to give read access to tracking data over API

no

yes

show tracking info

edit tracking data

removing

adding

requirement for removing data

requirement for adding/editing data

deleted

added

data saving

successful

successful

network found

Keep server data and local data synchronised

fetch user's local data

save to local data

saved successfully

[fetched]

fetch user's cloud data

[fetched]

received data for comparison

send to server for cloud data update

sent successfully

local data is newest

Cloud data is newest

/find newer data

local data differs from cloud data

[yes]

local data identical to cloud data

33

9 System Architecture

9.1 Component Diagrams

9.1.1 Candidate 1: 3-Tier Architecture

9.1.2 Candidate 2: Client-Server Architecture Candidate

9.2 Deployment Diagram

9.3 Architecture Justification

9.3.1 ATAM evaluation: Utility Tree

(A,B)
A = importance of attribute to system.
B = dificulty to acheive given attribute.

Utility

Performance
- responsivness —(H,L)— any user action should be executed in < 1 second
- response time
  - (L,M) user data should be uploaded to the server in < 3 seconds
  - (L,M) user data should be fetched from the server in < 3 seconds
- BAI —(H,H)— additional data imported to the app should be analysed by the ai in the least time possible

Modifiability
- updating —(H,L)— the app data from older versions of should be compatible with newer versions
- BAI —(M,M)— additional data should be able to be imported and analysed by the ai without having to reanalyse all data.

Availability
- Uptime —(M,L)— cloud server should be available 99.99% of the time.
- hardware faliure —(H,M)— local database faliures detected and fixed in < 1 min

Security
- Data confidentiality
  - (H,M) use hybrid cryptography for secure data transfer/storage
  - (H,L) customer database authorization works 100% of the time the cloud server is online.
- data integrity —(H,L)— ensure updates made by user are updated on both local and cloud DB

36

9.3.2 Justification

We will use the Two-tier **Client-Server architecture**. This is because our app is only using the server for the database synchronisation which means there is no need for the added complications and labor costs involved in implementing a Three tier architecture. While this can generalize and implement further features for the system, our software would not have significant benefit as the server is only used as a remote backup.

The Client-Sercer Model allows for faster communication to the Server because the app directly make requests to the server rather than having to go through an additional layer which add complexity to the software. This additional *feature* would slow down development where the app should be focused on quick deployment in order to provide its service to the users in need. Additionally the Server-Client architecture is easier to secure due to the lower complexity. Data is sent directly to and from the server so there are less points where data can leak, although an advantage of the three tier system would be that you are able to cache data to still access it if the servers are down. This is mostly irrelevant to the user as he has the local data storage which is usually the most up-to-date data.

10   User Interface
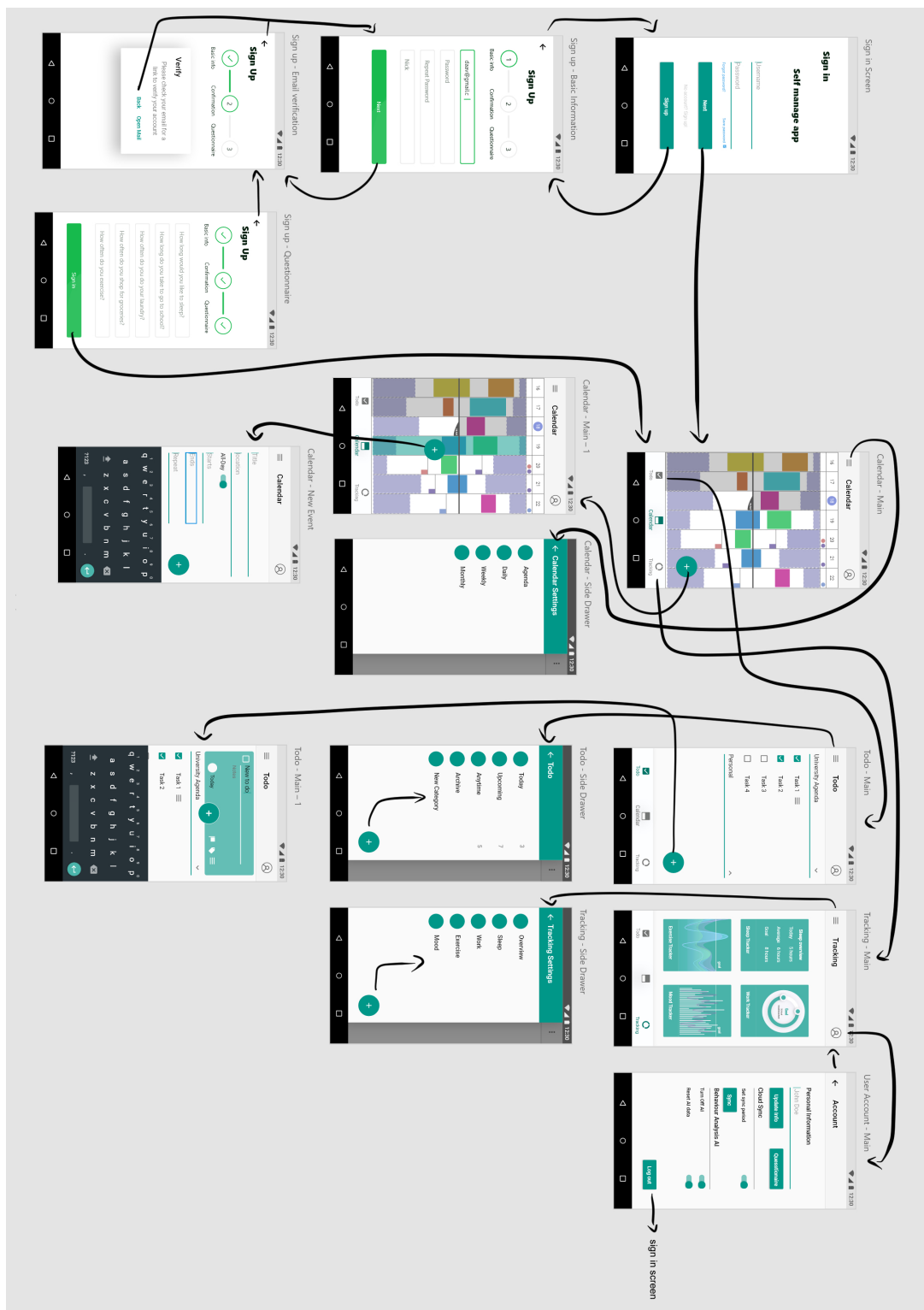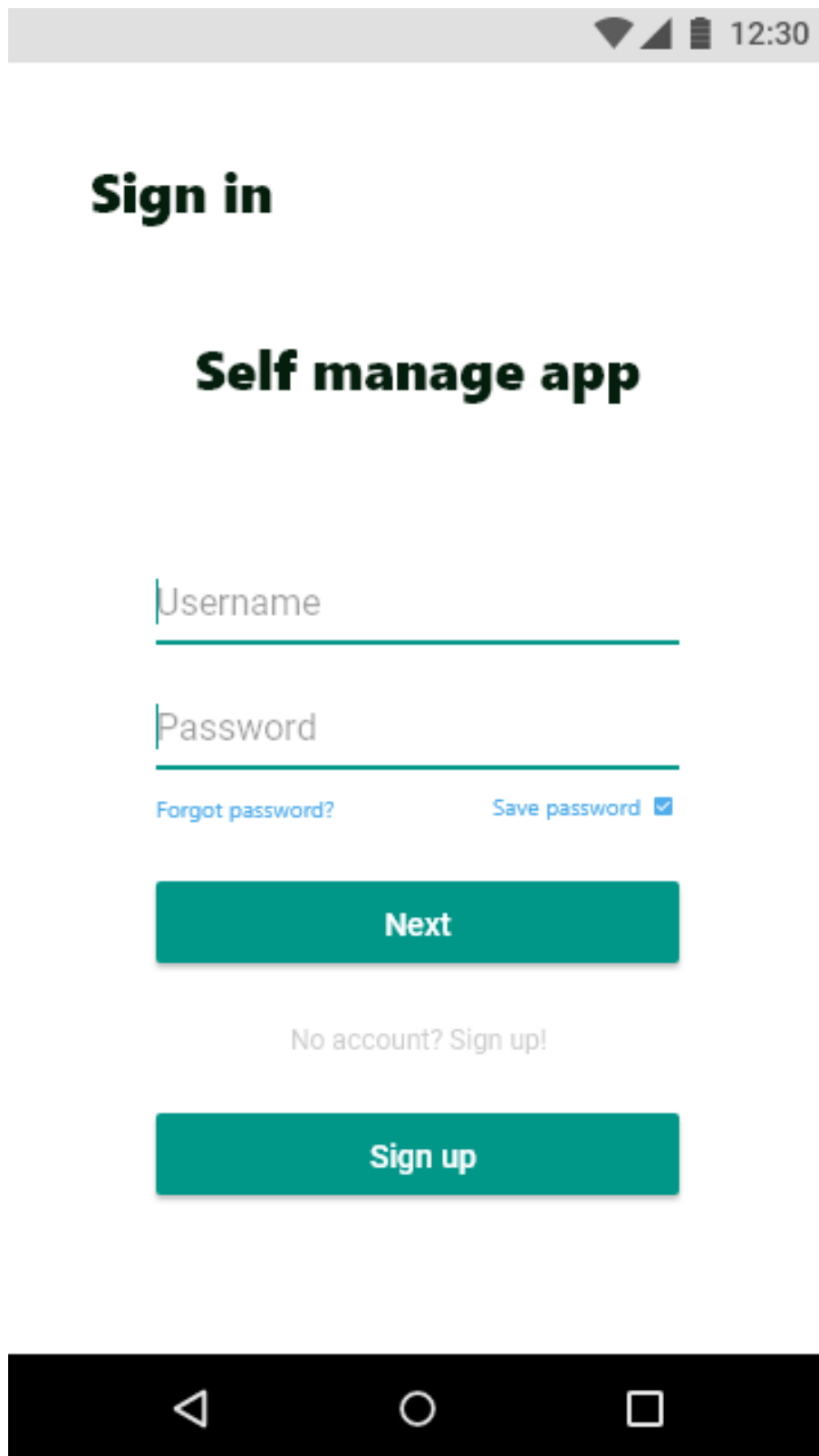
10.1   Overview

Figure 2: Overview of User Interfaces and UI action consequences

10.2 Login/Registration

Figure 3: Sign in

Figure 4: Beginning of Registration process
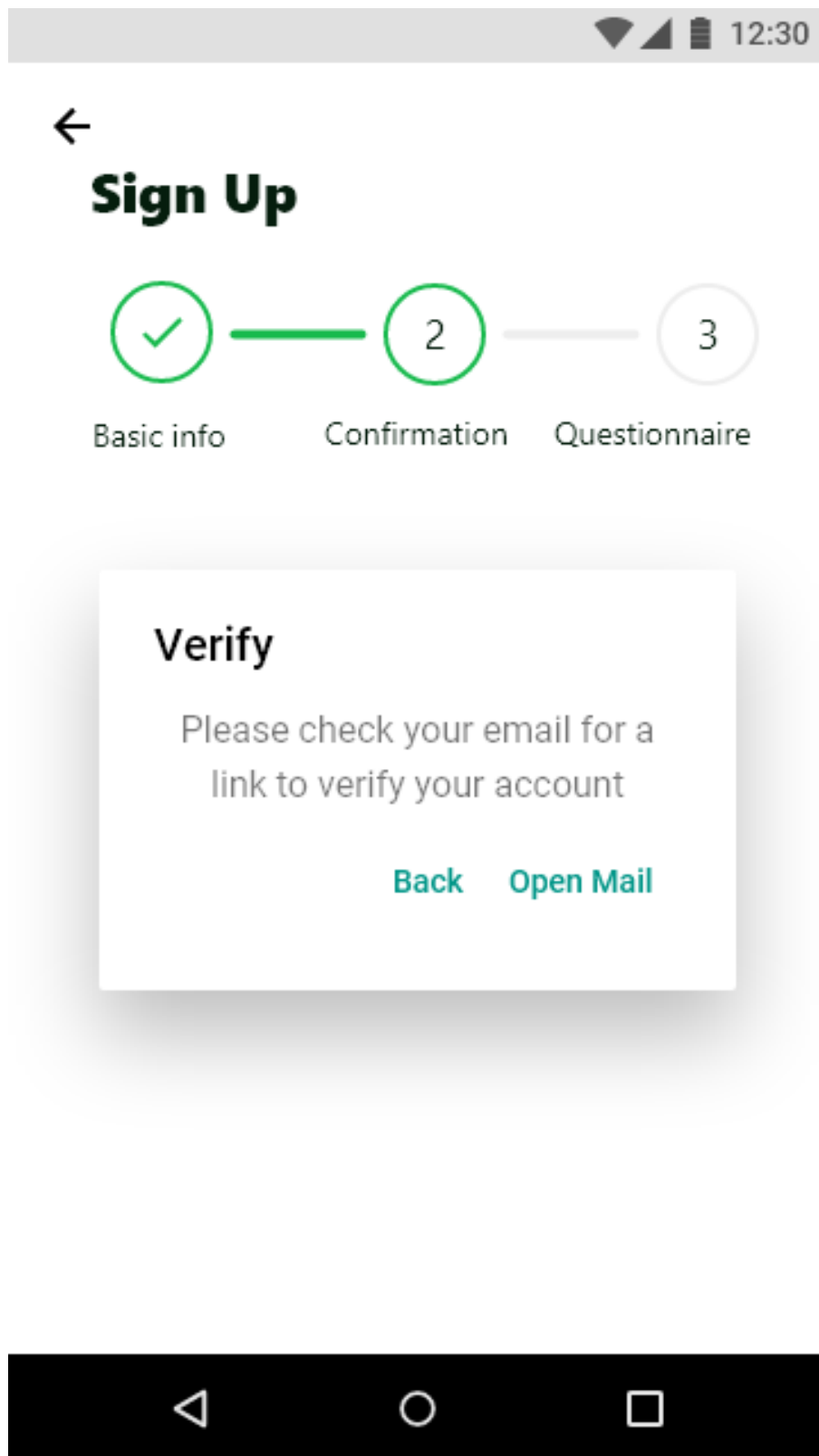
Figure 5: Verification request

Figure 6: Questionnaire

10.3 Calendar
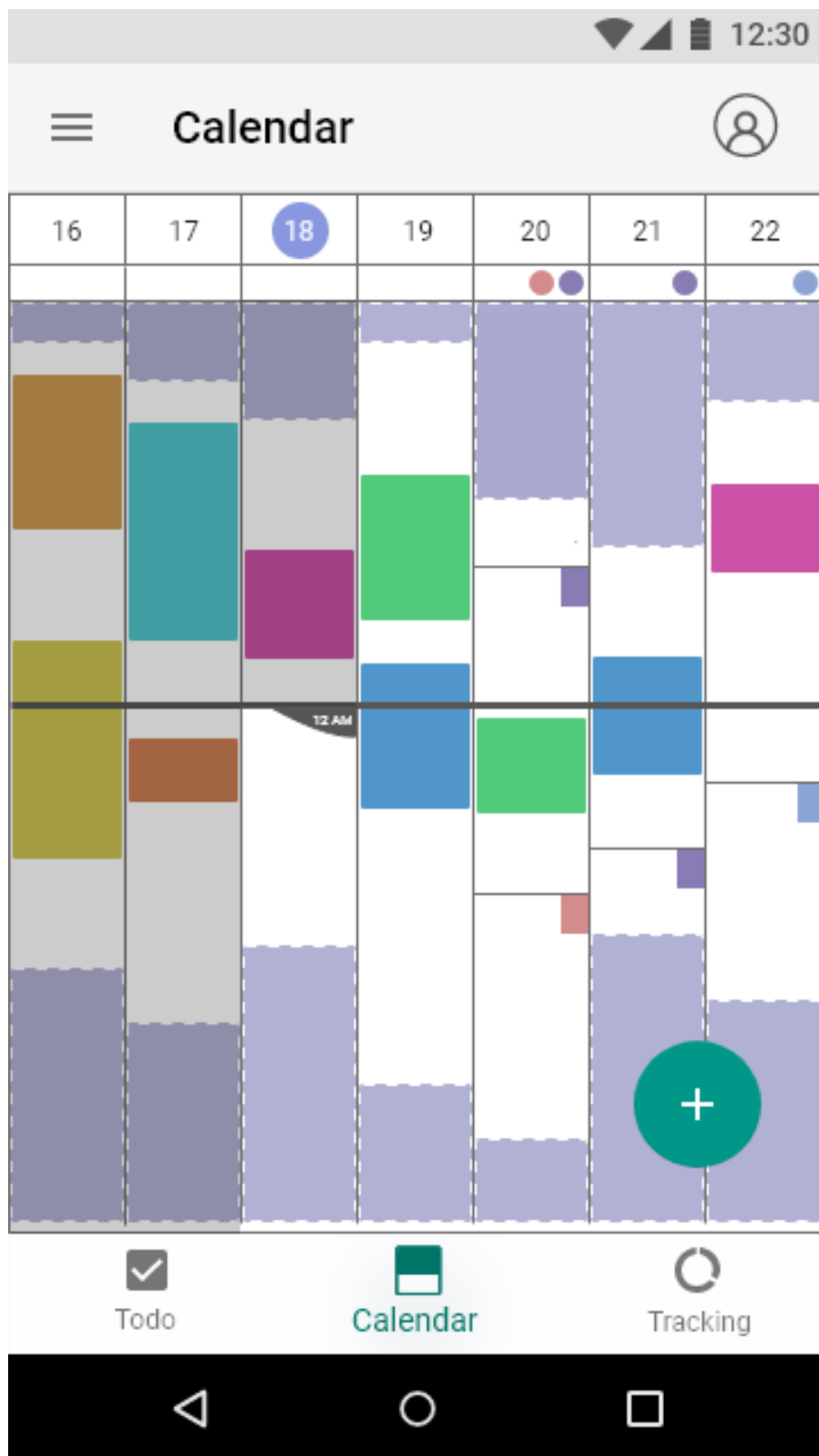
Figure 7: Weekly View mode of the Calendar tab (default page after login)
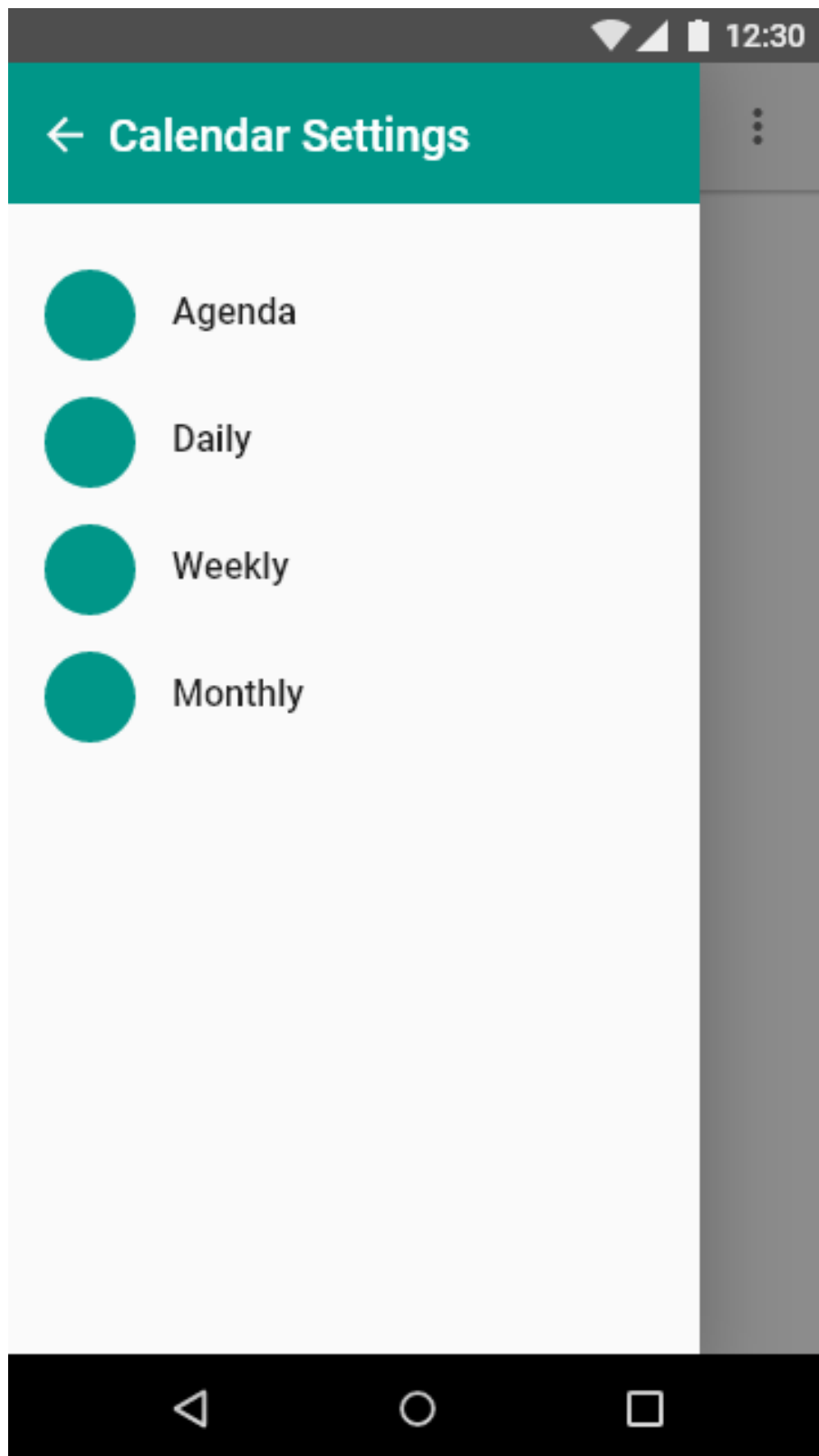
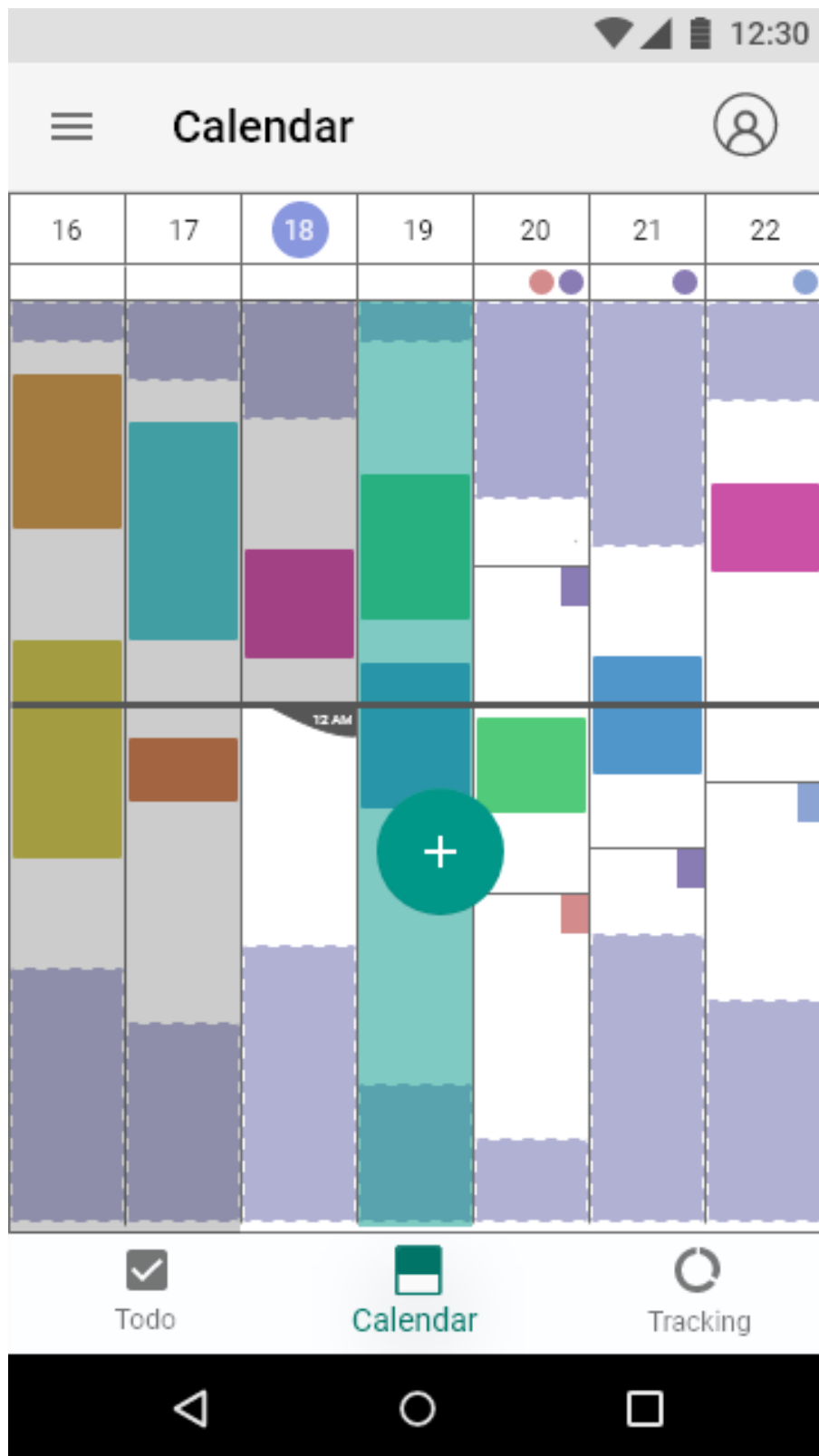Figure 8: View modes of the Calendar (top left button)
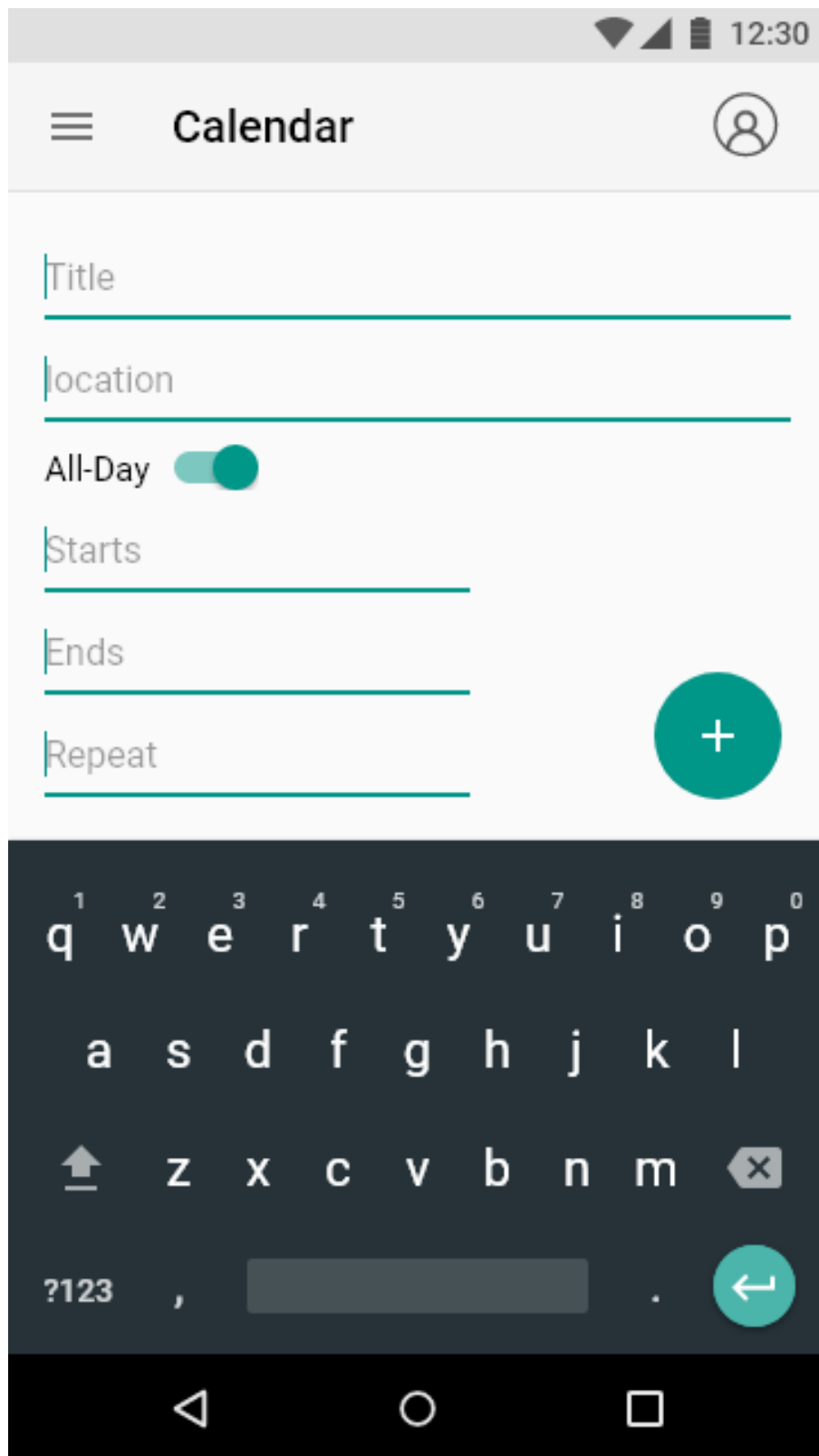
Figure 9: Adding an event using the draggable Plus-button

Figure 10: Adding event data interface

10.4 Todo
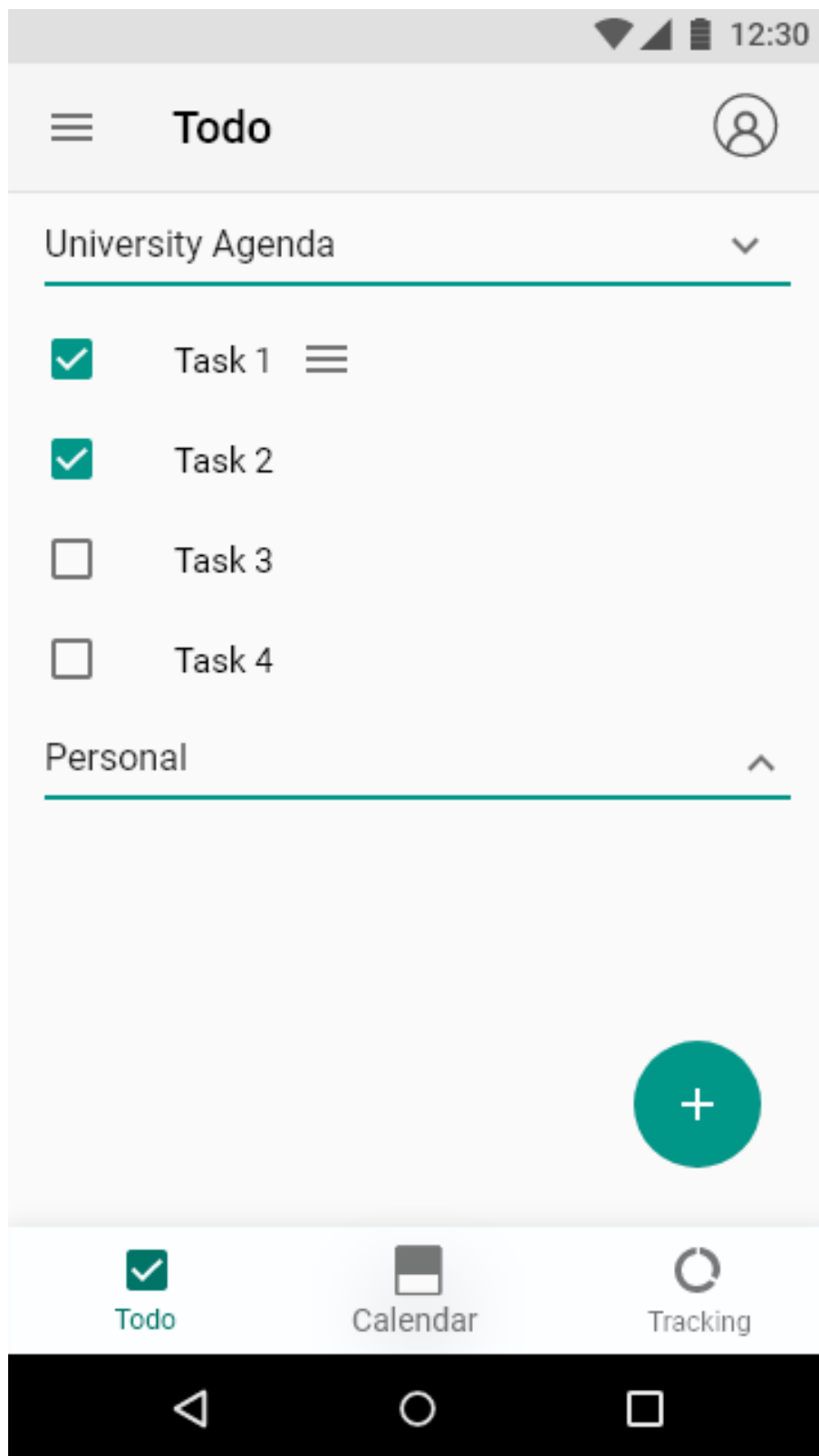
Figure 11: Default Todo tab View mode
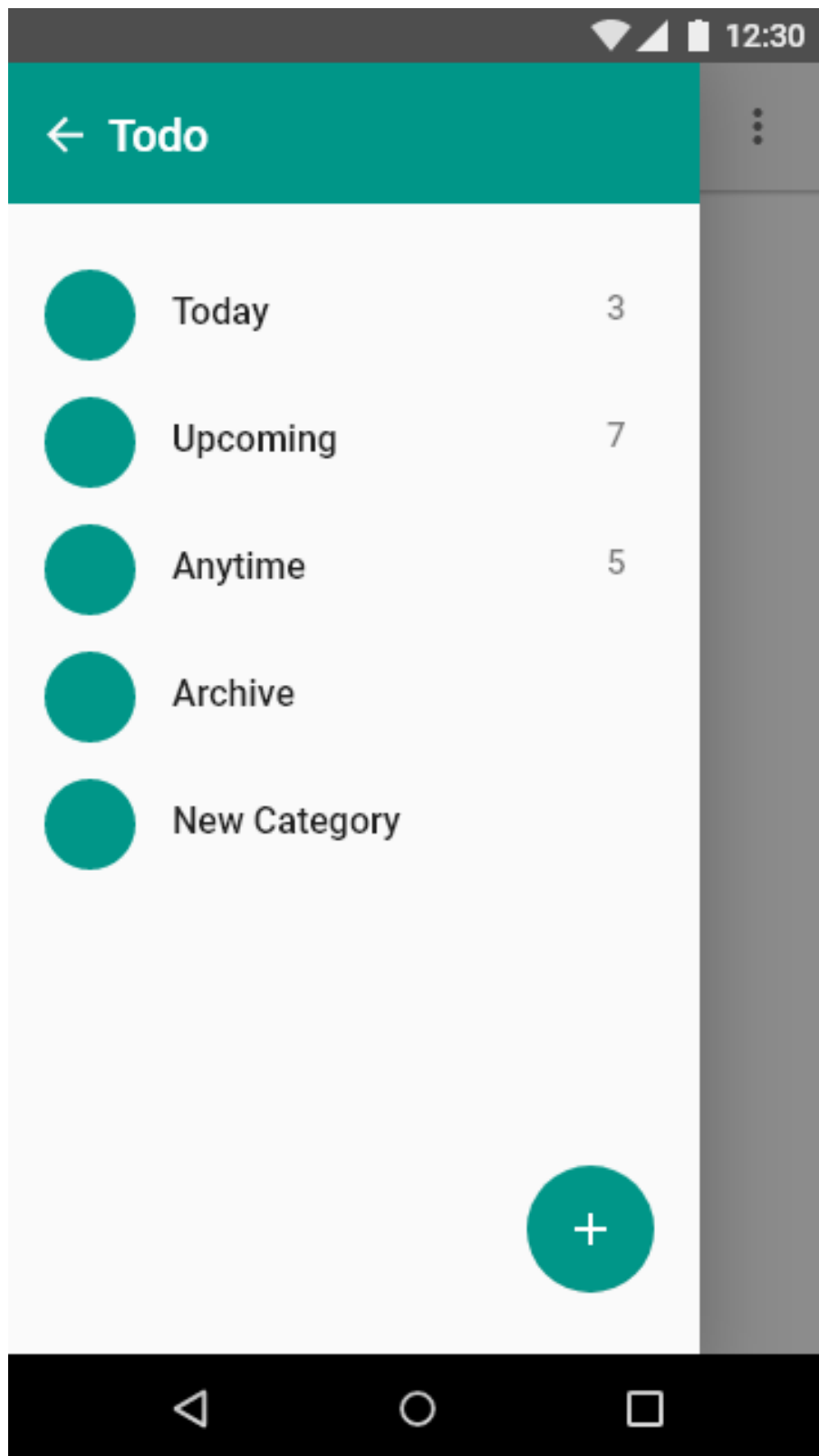
Figure 12: View modes of the Todo interface (top left button)
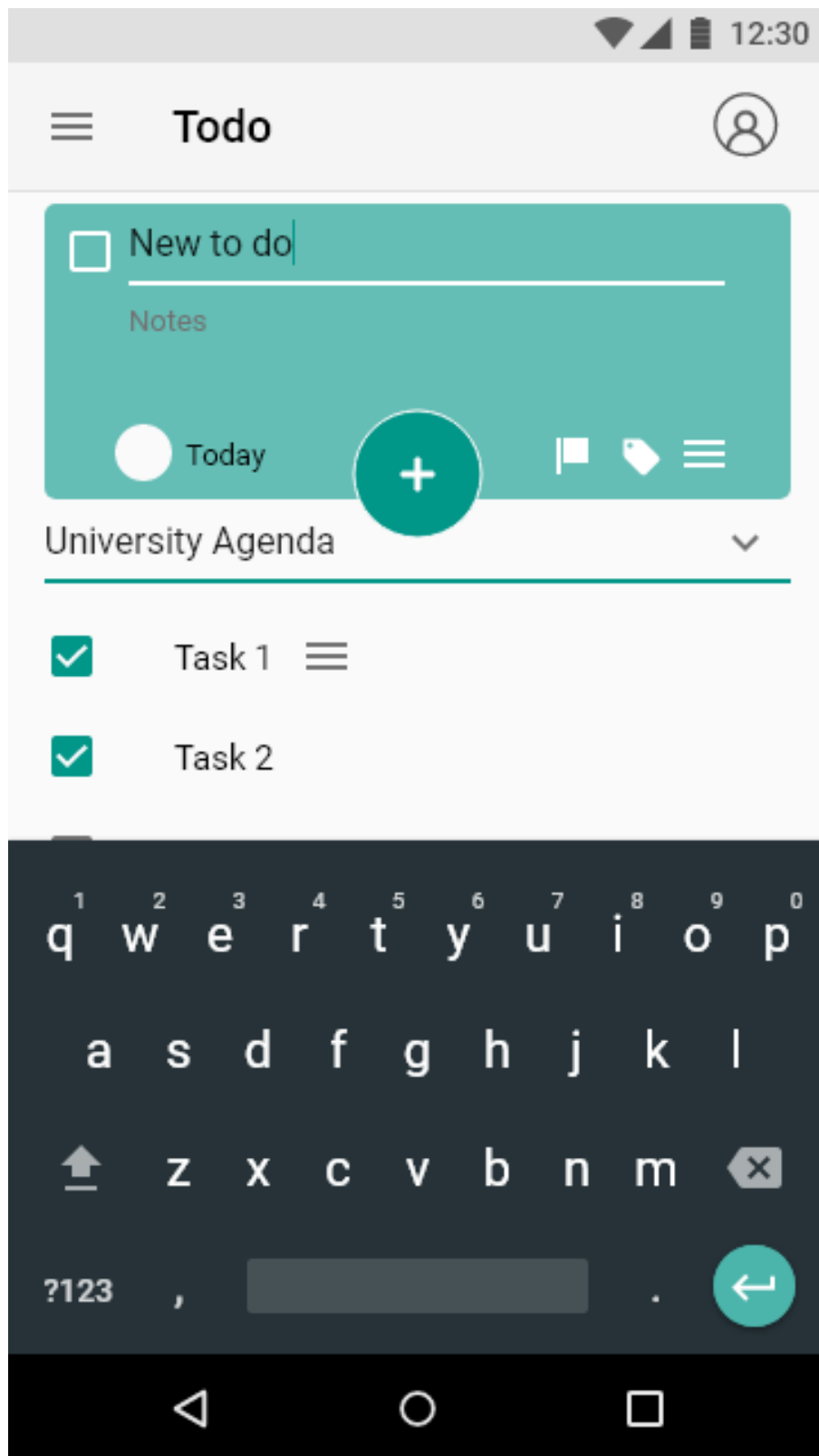
Figure 13: Adding a ToDo using the draggable Plus-button

10.5 Tracking
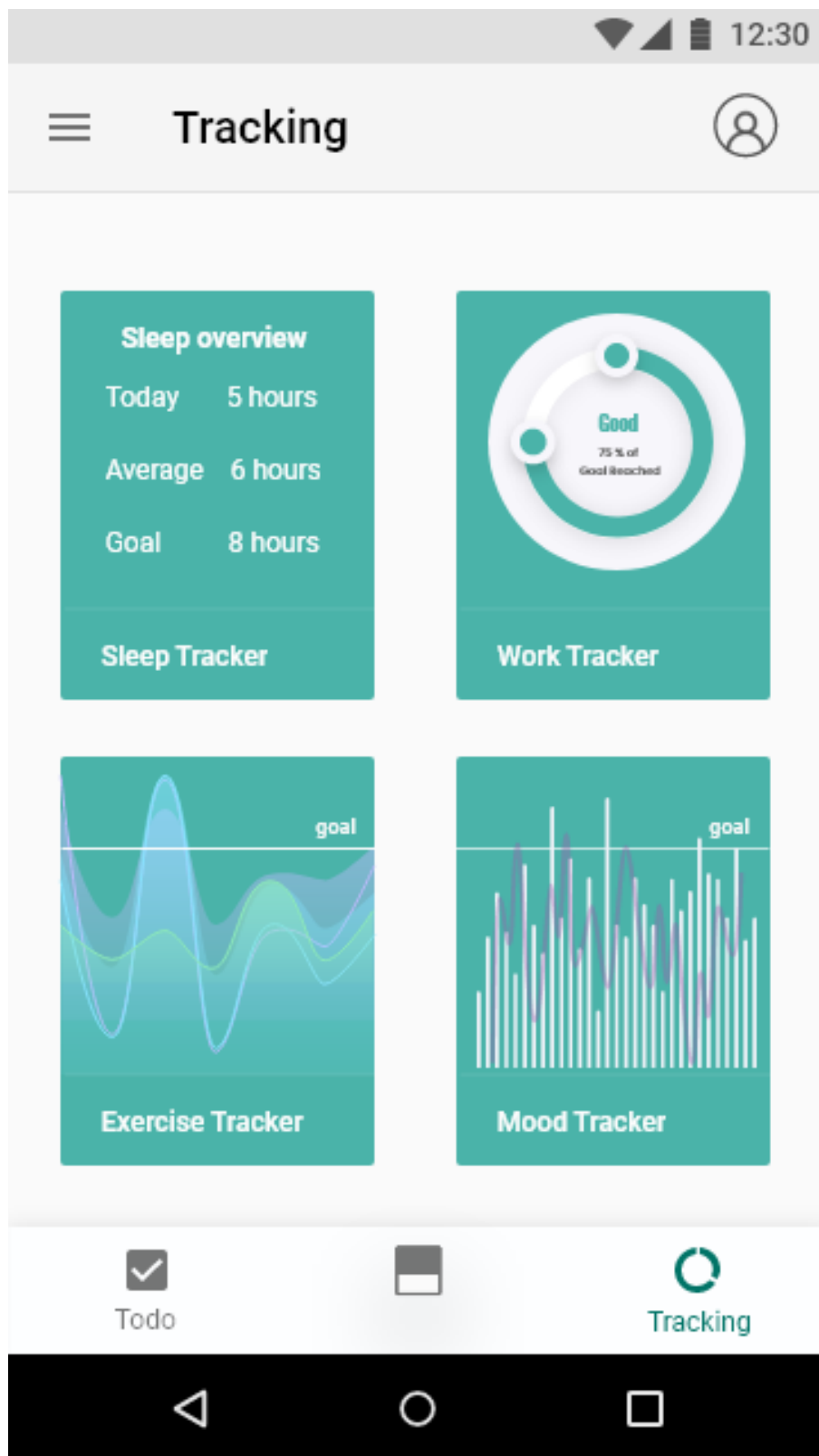
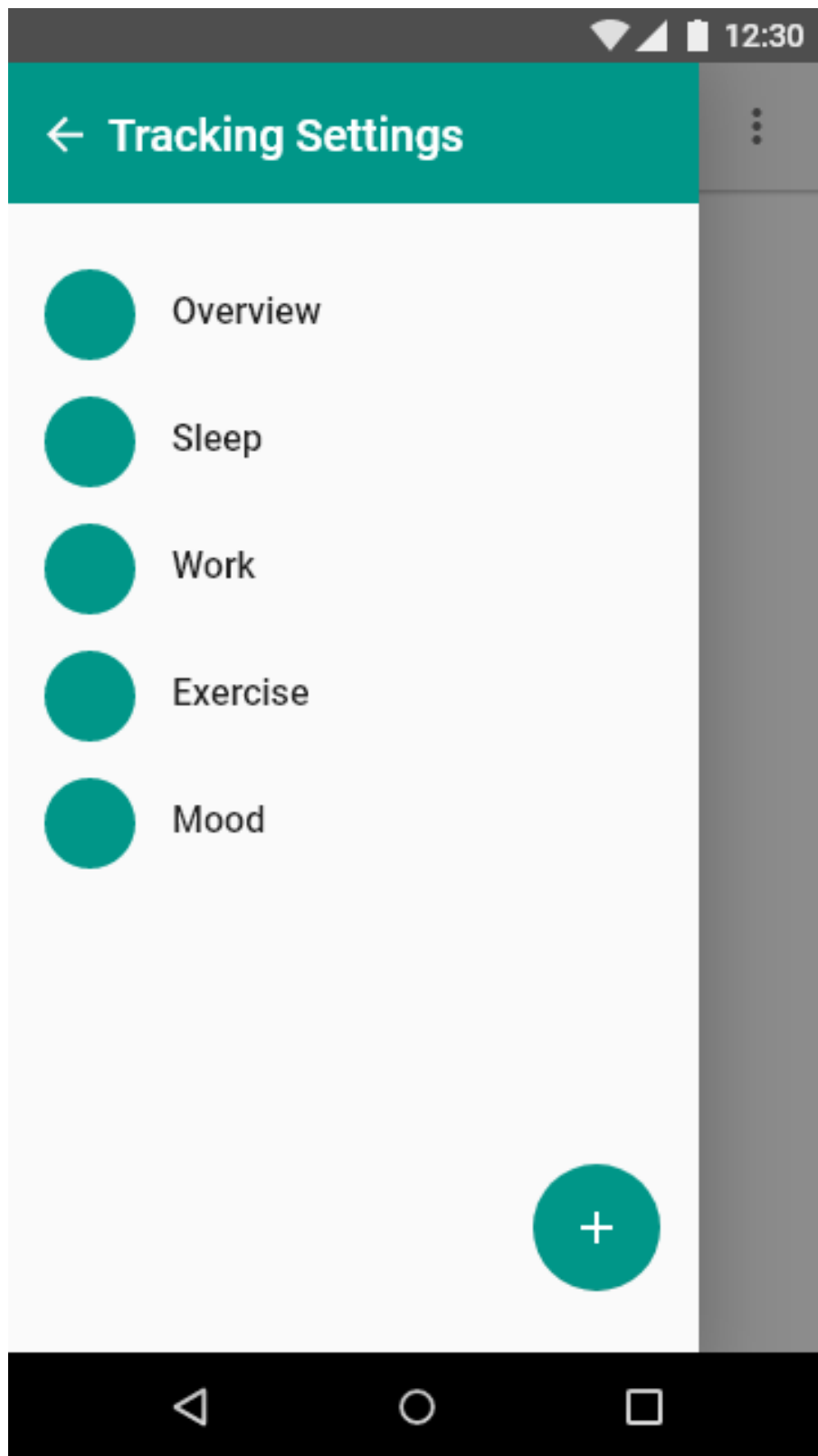Figure 14: Default Tracking tab Overview mode

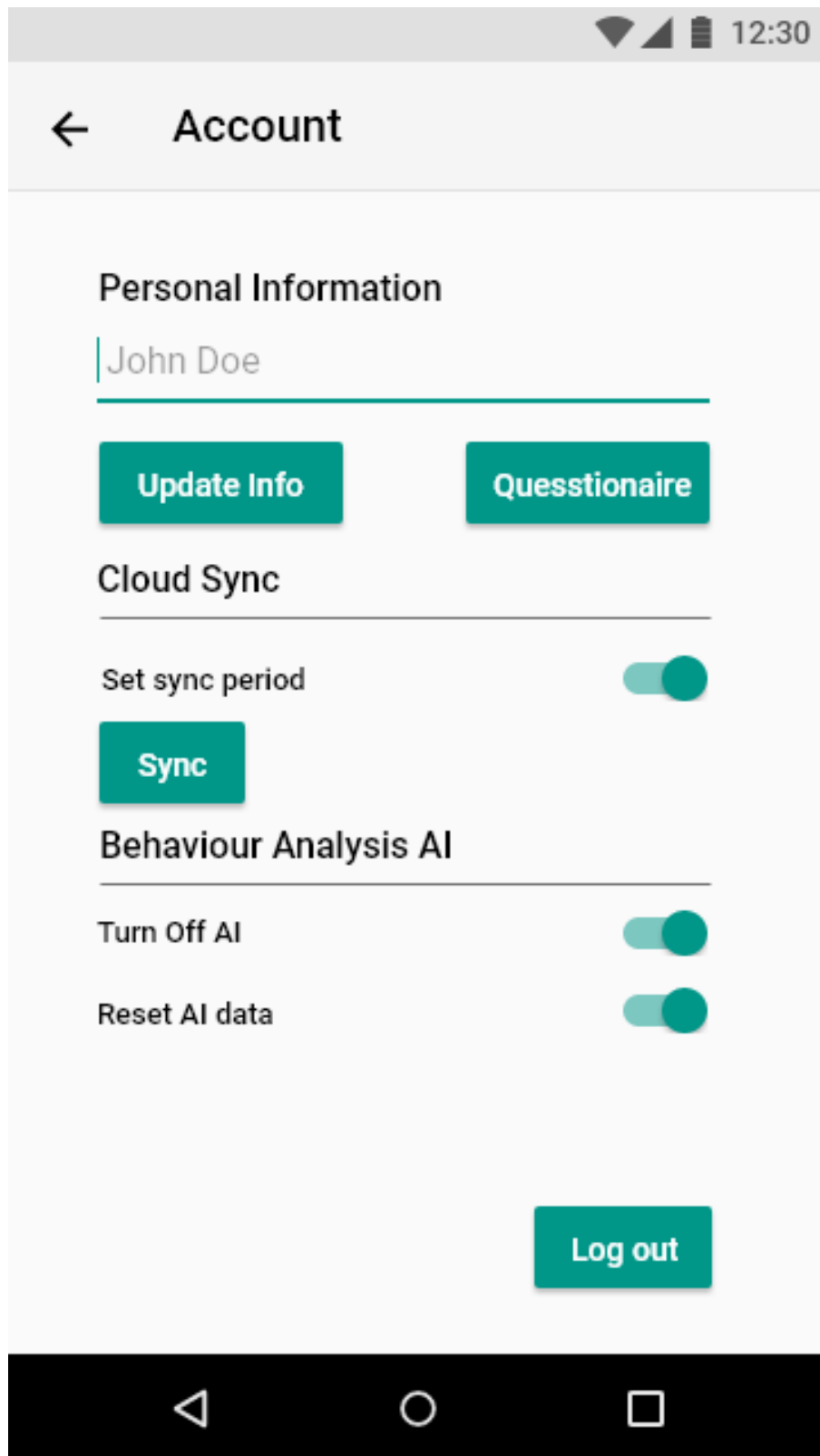Figure 15: View modes of the Tracking interface (top left button)

10.6 Profile

Figure 16: Settings/Profile tab (top right button)