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, 80% of students in higher education report symptoms of stress or anxiety. According to an article by the guardian [cite the article], 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. 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 task. 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

- he 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.

5

* 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.Upload performance:

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

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

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.
    - 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
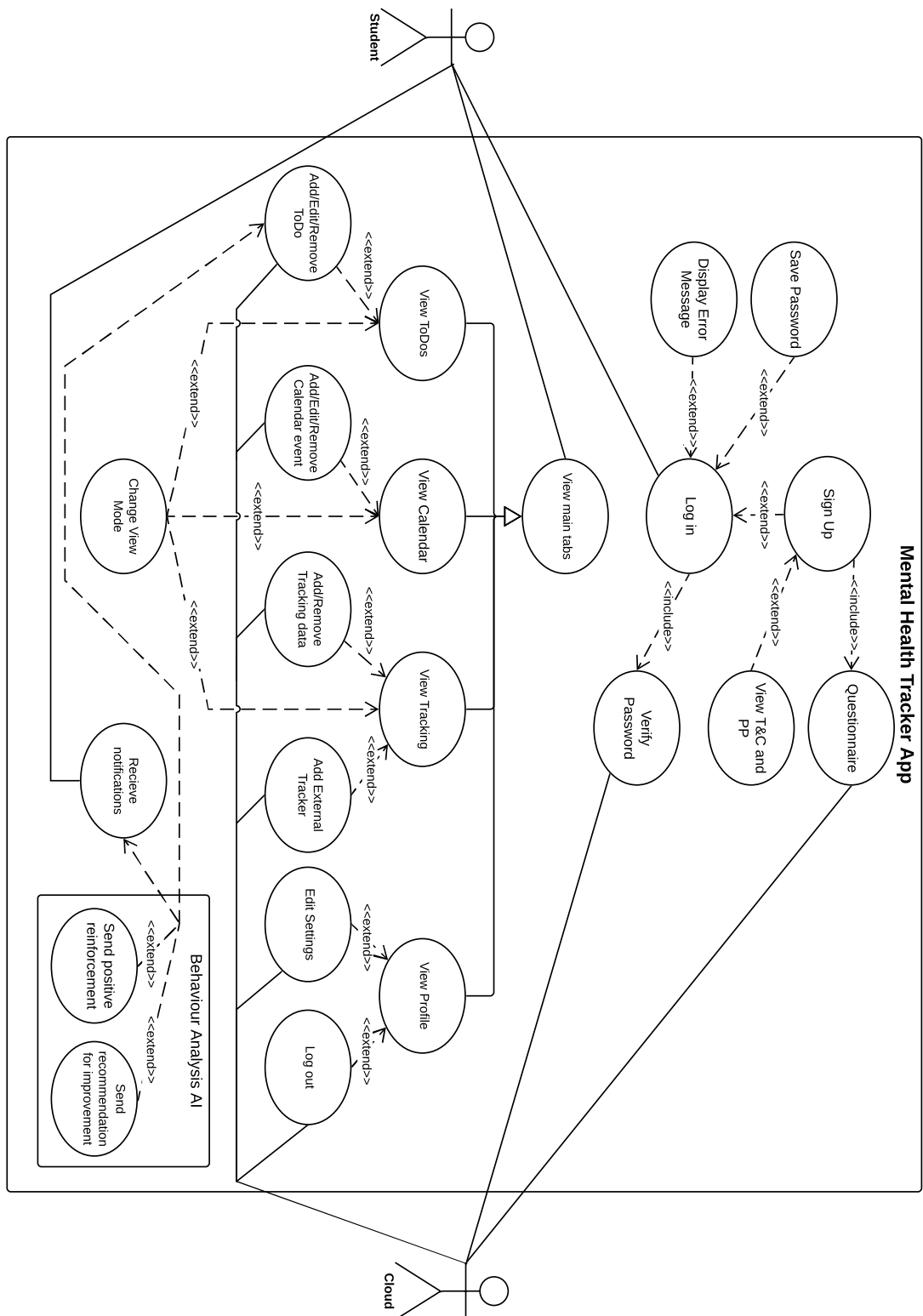
3.2 Non-trivial Use Case: Add a tracker

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.
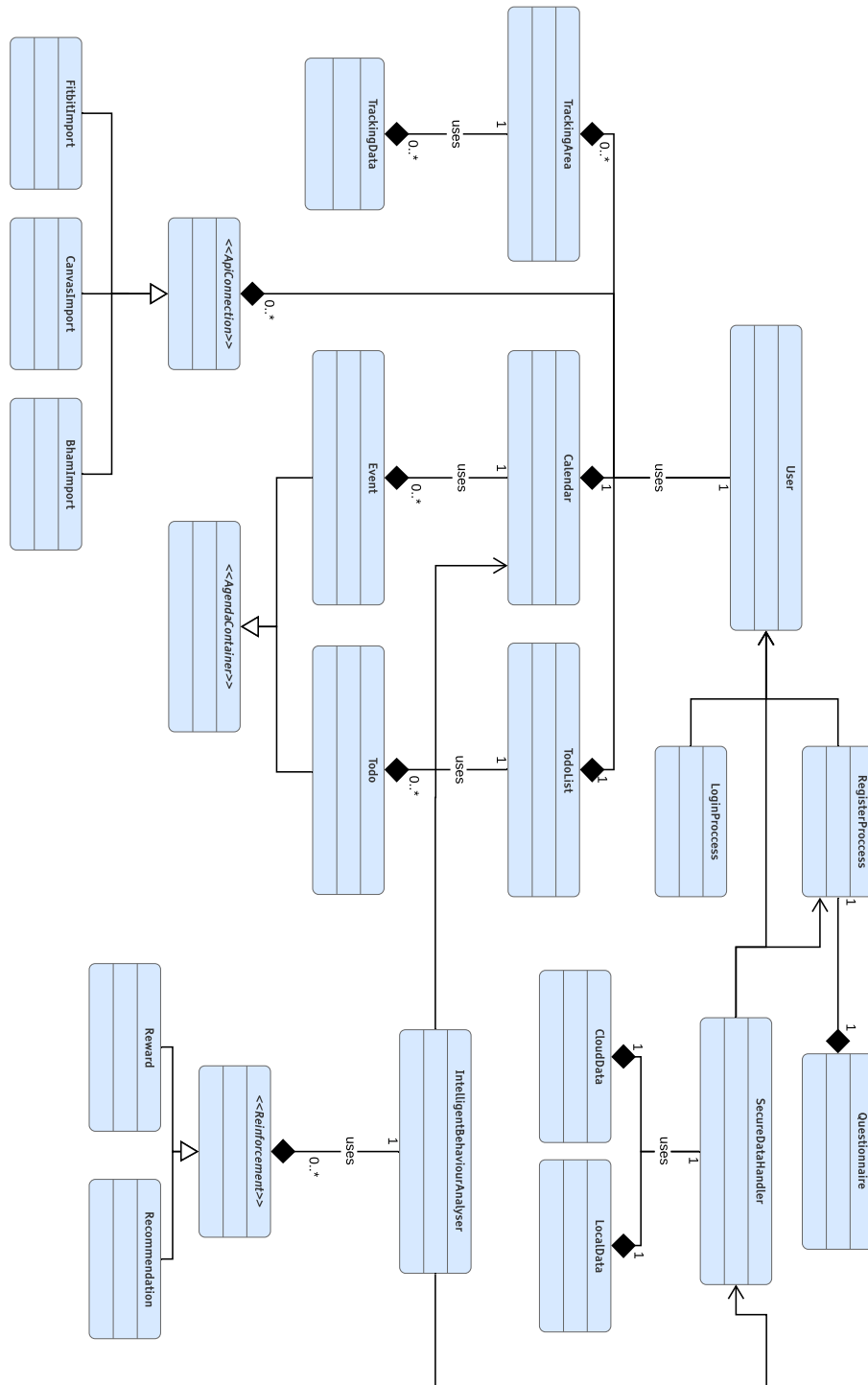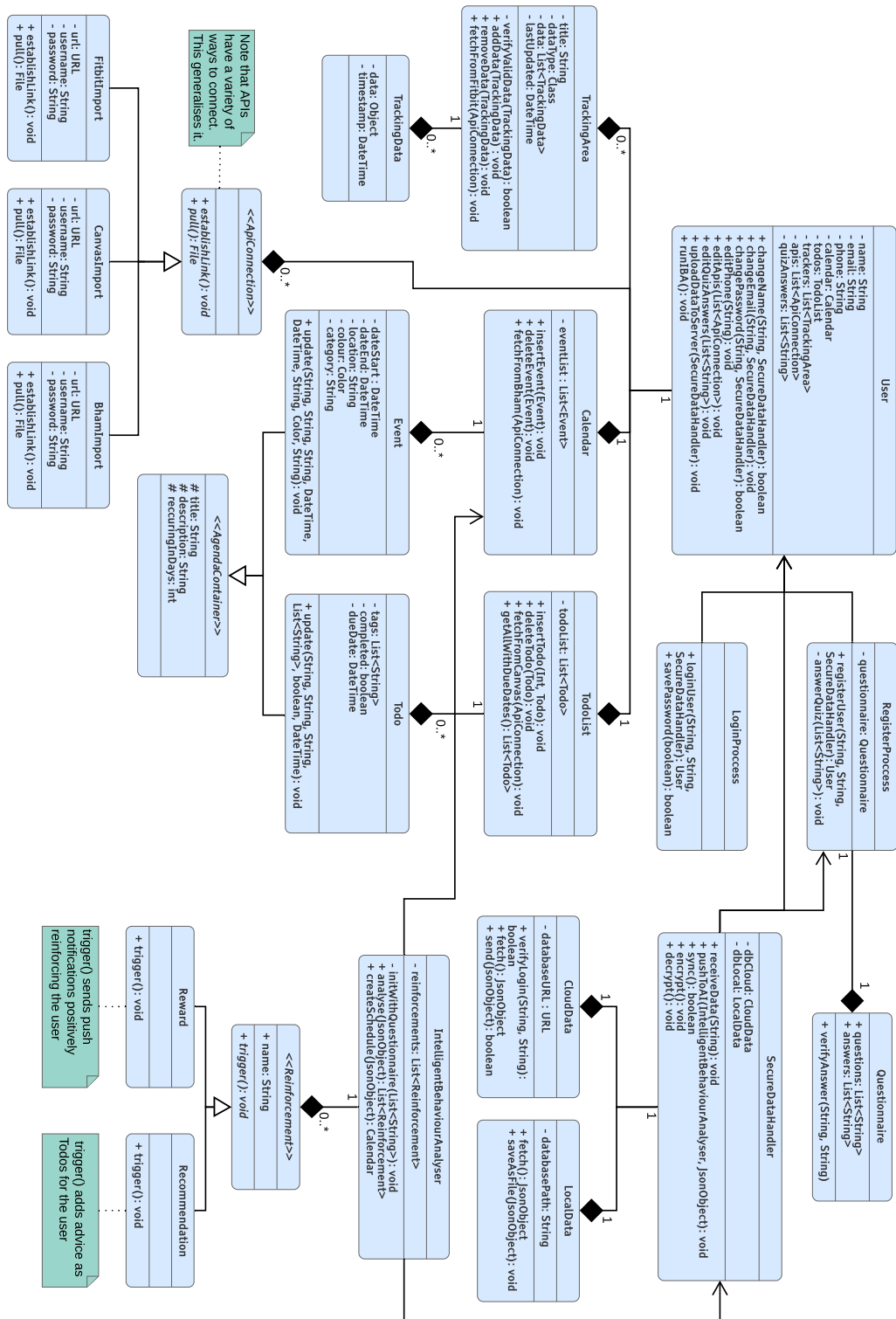
4 Activity Diagram
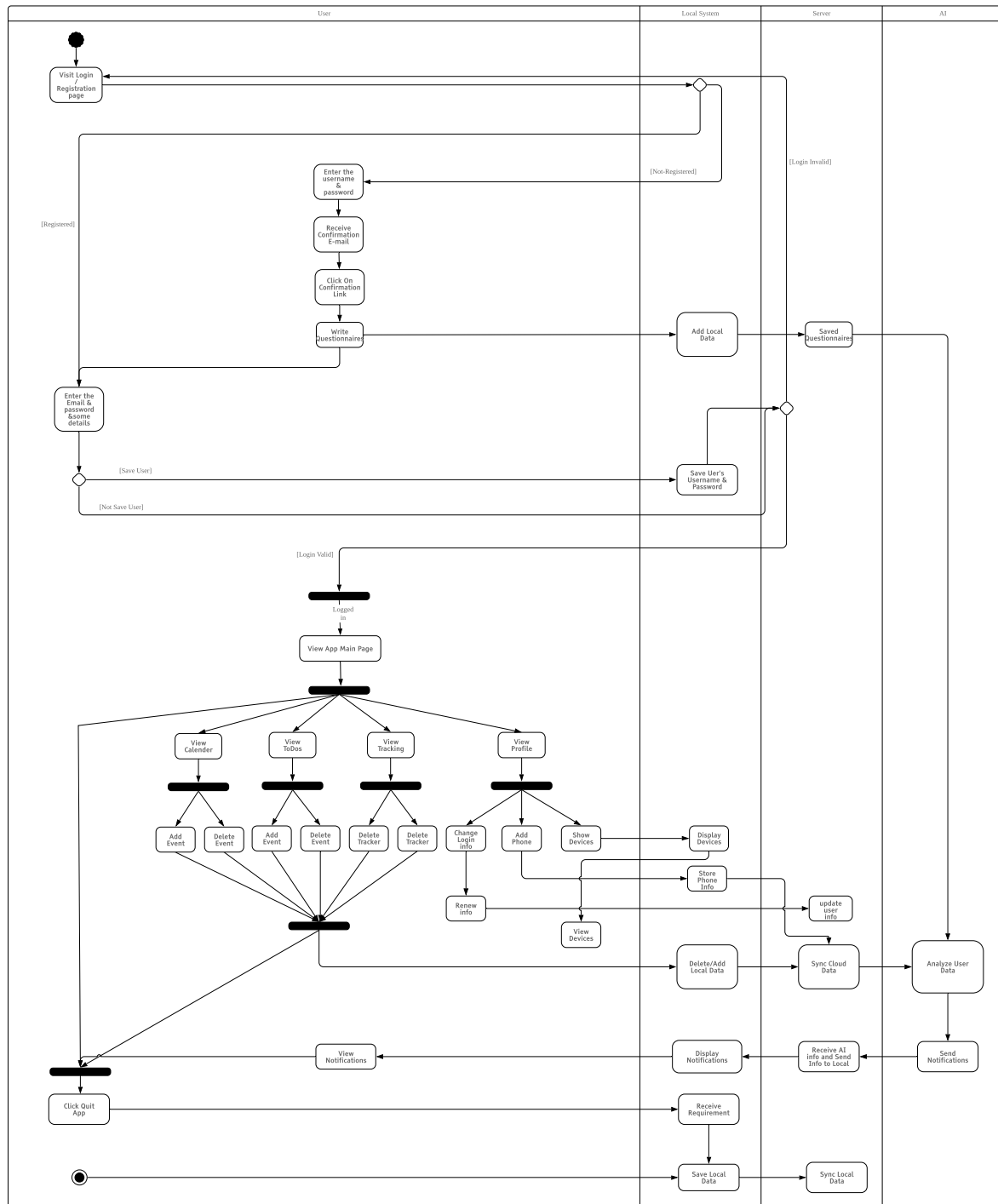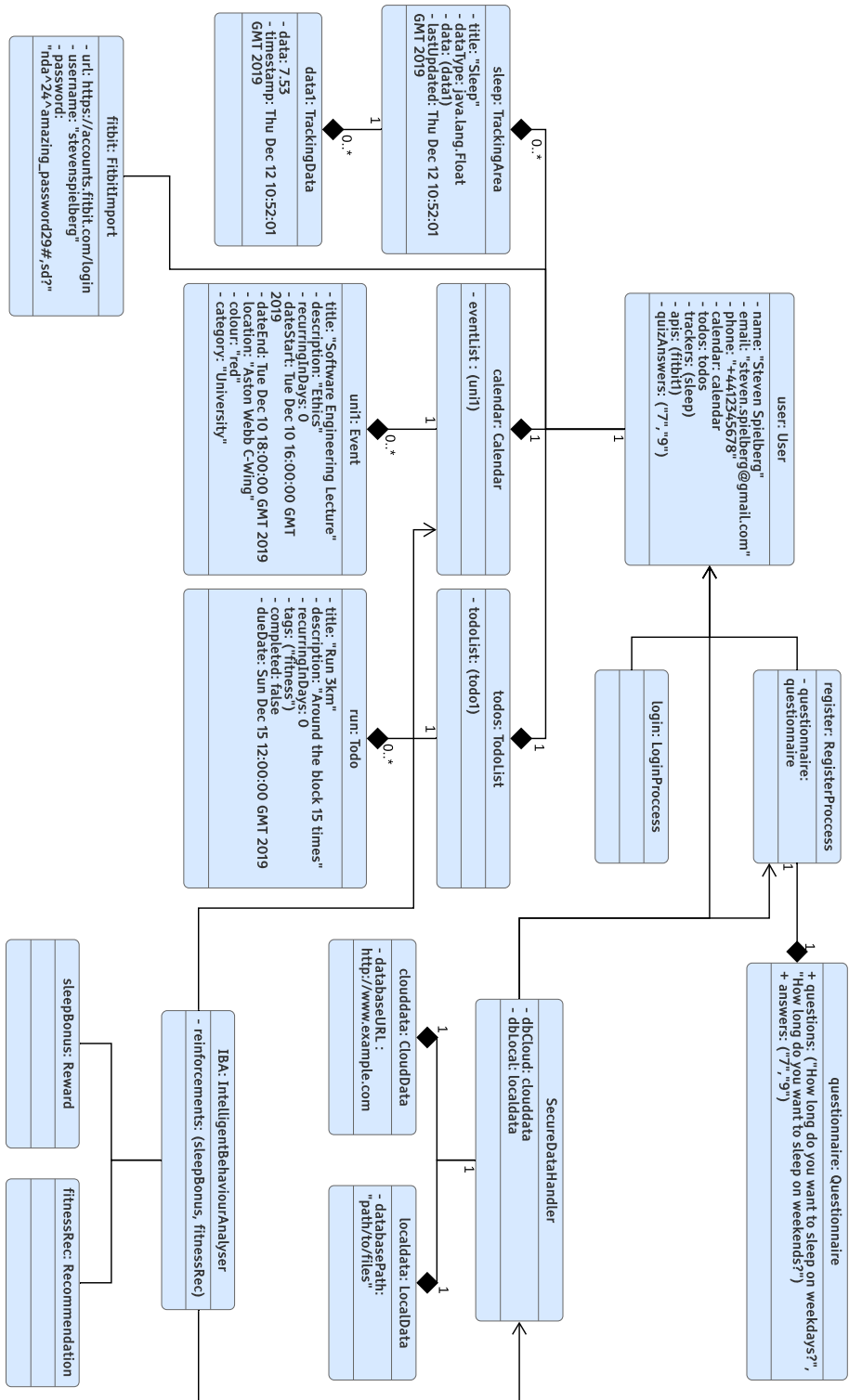
5 Class Analysis

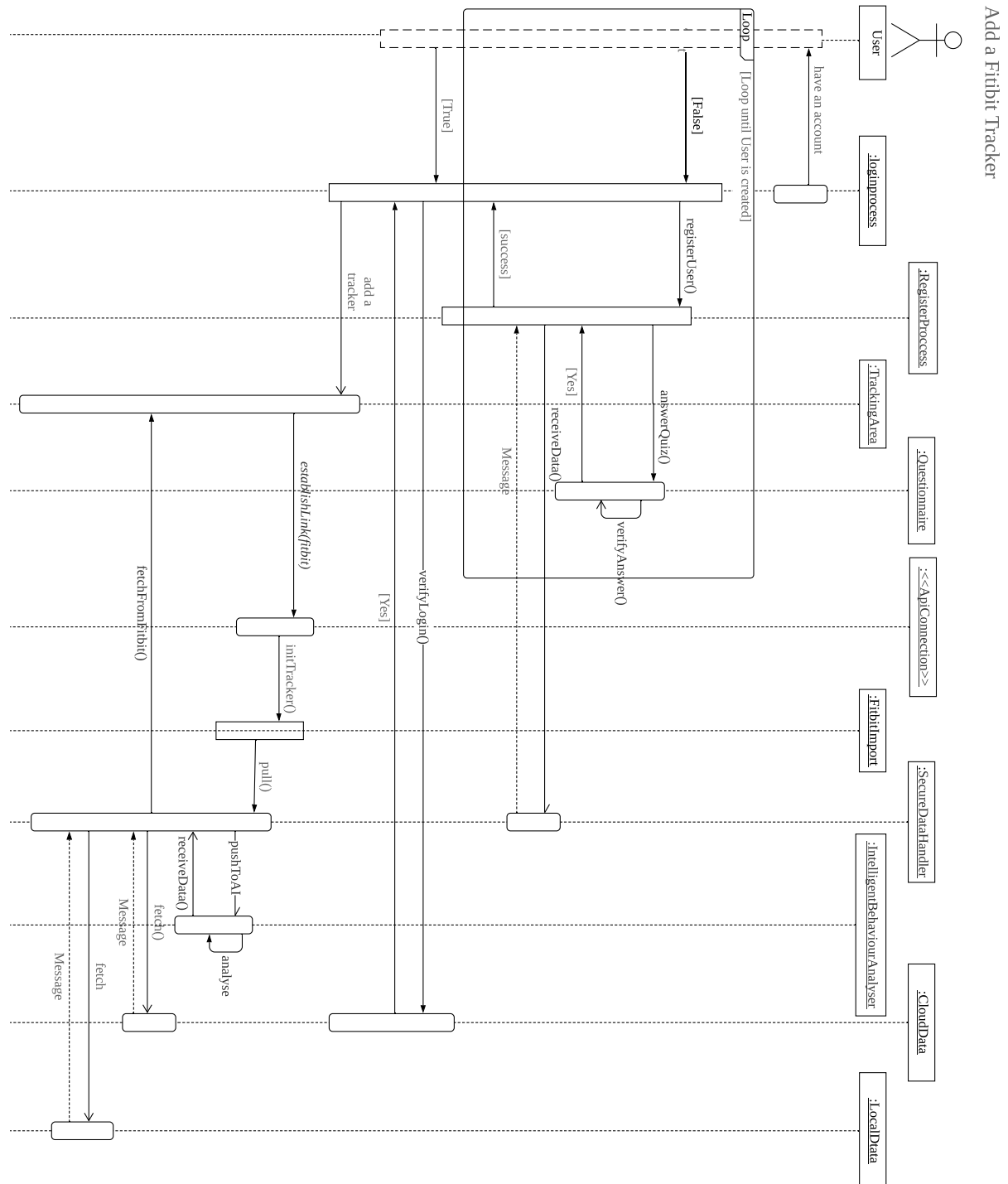## 5.1 Noun-verb Analysis

5.2 First-Cut Class Diagram

5.3 Detailed Class Diagram

6 Object Diagram
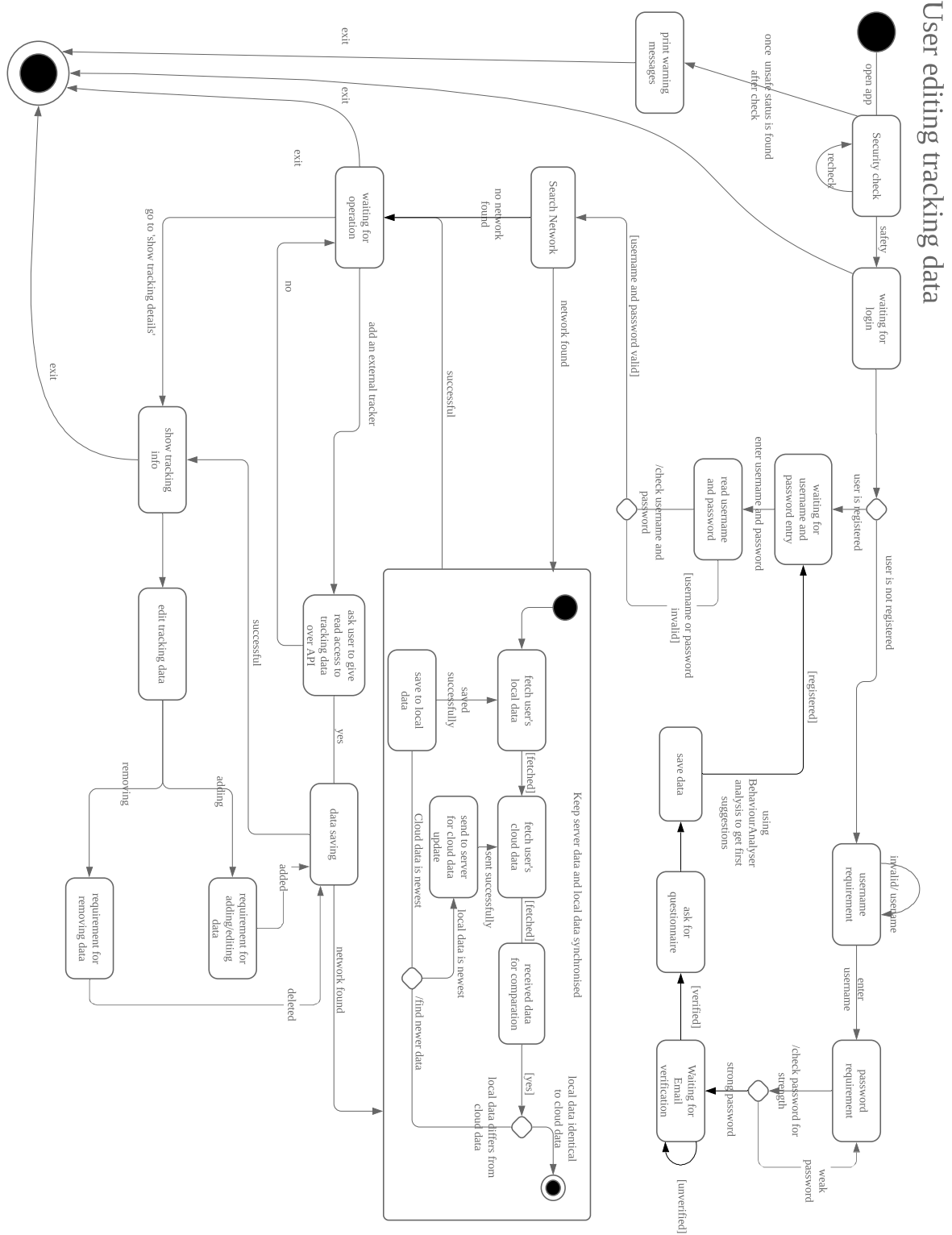
7 Sequence Diagram

8 State Diagram

User editing tracking data

9 System Architecture

9.1 Candidate Architectures

9.1.1 3-Tier Architecture Component Diagram

9.1.2 Client-Server Architecture Component Diagram

9.2   Deployment Diagram

9.3   Architecture Justification

9.3.1   ATAM evaluation: Utility Tree

Test2