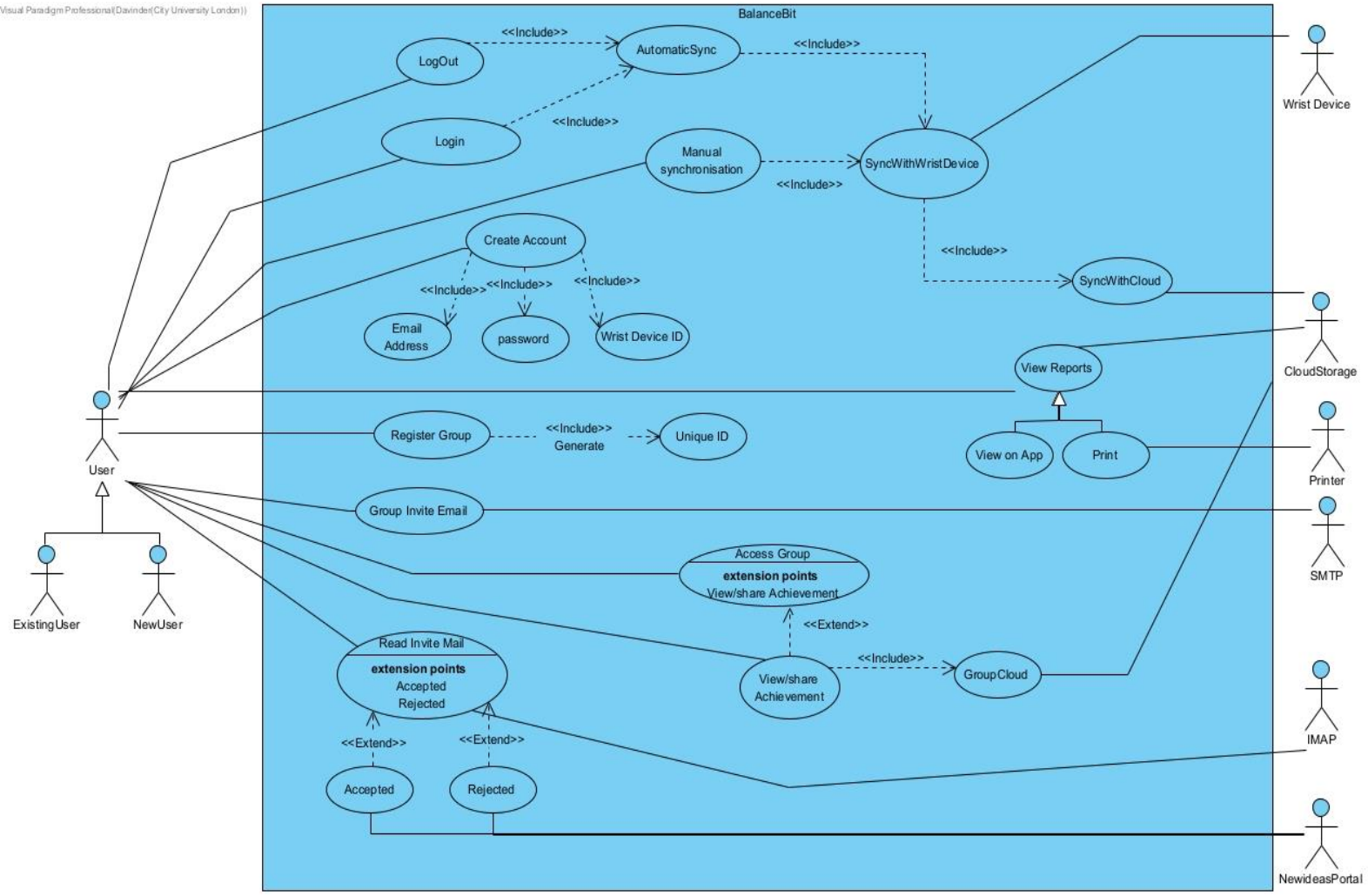
Contents

Assignment 1 – Analysis Question 1: User requirements

Requirement ID: 01	**Requirement Type: FR**	**Event/Use Case 2**
Description: The device must be able to detect steps and maintain a steps counter for the current day.		
Rationale: This will help record steps for the day and analyse user movement to help with work/life balance.		
Source: Balance Bit scenario		
Fit Criteria: External devices can be worn along to measure step count. Balance bit step count must be in specific range (~100 + or - steps) for it to be considered accurate measuring device. Motion sensors can be used to measure movement and step count. Each day's step count must be separated should start from 0 at beginning of day.		
Customer Satisfaction: 5	**Customer Dissatisfaction: 5**	
Priority: Essential	**Conflicts:** None	
Supporting Material: Initial Statement	Volere Source: Atlantic Systems Guild	
History: New requirement		

Requirement ID: 01	**Requirement Type: NFR** (Reliability)	**Event/Use Case 2**
Description: The Balance Bit software and wristwatch must be reliable.		
Rationale: The data is independent to each individual and must be recorded consistently for it to be effectively used in analyses (Determining users work/life balance). The system and product must be functional throughout day without failures (eg: randomly stops recording step count)		
Source: Balance Bit scenario initial statement.		
Fit Criteria: Stress/bug testing is essential for all software and electronic products. It helps to find flaws and limits within software/product. By testing (Against existing devices), reliability can be measured by determining if wrist device functions correctly throughout the day.		
Customer Satisfaction: 5	**Customer Dissatisfaction: 5**	
Priority: Essential	**Conflicts:** None	
Supporting Material: Essential requirement for products	Volere Source: Atlantic Systems Guild	
History: New requirement		

Question 2: Use case Diagram



IMAP helps to read internet messages.

Question 3: Use case Specifications

ID: 1.0	**Use case:** SyncWithWristDevice

Brief description:

Upon a successful login/logout an automatic synchronisation with wrist device is performed which obtains data from device and stores it within desktop application (Ready to be uploaded to cloud storage). The synchronisation can be triggered manually by the user when he/she is logged in.

Primary actors:

User (Manual sync when logged in system)

Secondary actors:

Wrist Device

Preconditions:

1. User must be logged in/log out within the desktop application and have wrist device linked with application.
2. User must request a sync after automatic sync is completed to trigger it again

Flow of events:

1. Successful login to the system.
2. System Connects to wrist Device and checks wrist device ID.
3. System checks for number of days the data is accumulated.
4. IF (WristDeviceID is same)
   4.1 Triggers sync with wrist

   ELSE

   Reconnect wrist device with same ID
5. Do (day 1 to day n)
   5.1 Do (0 to n number of activities)

   5.1.1 For each day the activities are individually stored for that specific day including the times and type of day.
6. Desktop holds data from wrist device
7. SyncWithCloud is triggered as syncWithWristdevice is complete.
8. If(Actor Logout) - triggers an automatic(mandatory) synchronisation.

Postconditions:

1. The data from wrist device is synced with desktop application.

Alternative flow:

WristDeviceNotConnected.

WristDeviceIDNotFound

Alternative flow: WristDeviceNotConnected
ID: 1.1
Brief description: Desktop Application cannot wirelessly connect with device. Causing synchronisation failure.
Primary actors: none
Secondary actors: none
Preconditions: Device unable to connect with desktop application for synchronisation.
Alternative flow: The flow starts after step 2 of the main flow. 1. Triggers if Device in main flow is unable to connect with desktop application. 2. Prompts the user to reconnect their wrist device. 3. Unable to synchronize with desktop application.
Postconditions: None.

Alternative flow: WristDeviceIDNotFound
ID: 1.2
Brief description: The registered wrist device ID does not match the one recorded in the Desktop application
Primary actors: none
Secondary actors: Wrist Device
Preconditions: Desktop unable to connect due to different IDs
Alternative flow: The flow starts if step 2 is false of the main flow. 1. Prompts the user to connect with same device that was registered with. 2. Unable to synchronize with desktop application.
Postconditions: None.

ID: 2.0	**Use case:** SyncWithCloud

Brief description:

After synchronisation with the wrist device is complete, the data is held in the desktop application ready to be uploaded to cloud. SyncWithCloud ensures the data is uploaded and deleted from desktop after completion.

Primary actors: User

None

Secondary actors:

Cloud

Preconditions:

1. After syncwristdevice successfully operated and Desktop Application must contain data ready to be uploaded to cloud (previous day upload failed or couldn't connect to cloud).

Flow of events:

1. Successfully connected to cloud storage
2. IF cloud storage not full AND connected to internet (
3. The system checks data for accumulated days and selects n days to be upload to cloud.
4. Do (day 1 to day n)
   4.1 Do (0 to n Number of activities)
   3.1.1 For each day the activities are individually stored for that specific day including the times and type of day.)

   ELSE
   PROMPT MESSAGE FOR NOT ENOUGH STORAGE OR NO CONNECTION
5. After successful sync, the data held in system is deleted and space is freed for new data.
6. Actor Logout triggers an automatic(mandatory) synchronisation that also syncs with cloud.

Postconditions:

1. The data from Desktop application is successfully transferred to cloud storage.
2. Data from desktop application is deleted after being sync with cloud is complete

Alternative flow:

NoInternetAccess (unable to connect to cloud storage).

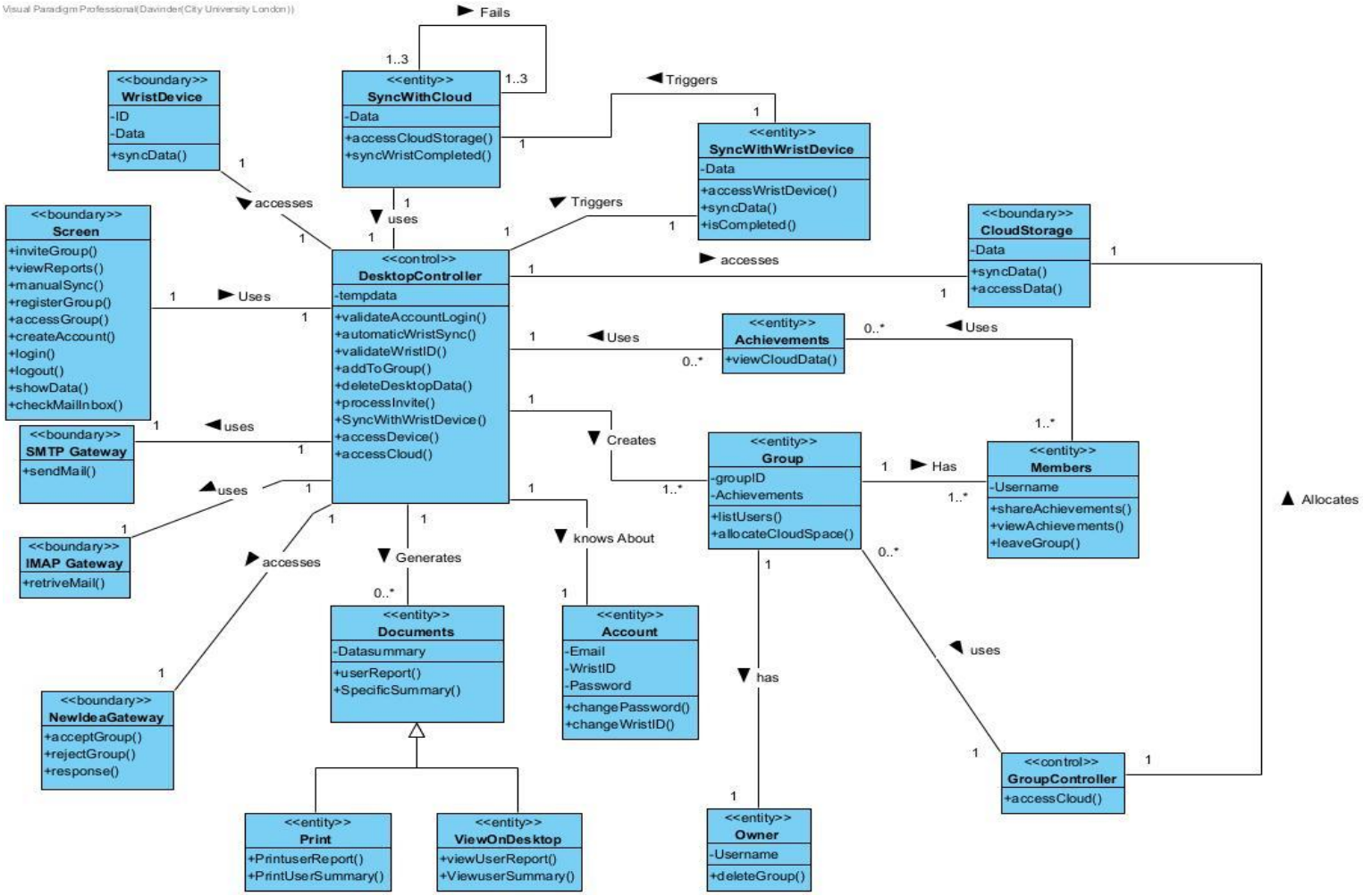
Not sufficient Cloud Space

Alternative flow: NoInternetAccess
ID: 2.1
Brief description: Desktop Application is unable to connect to internet, being unable to upload to cloud.
Primary actors: none
Secondary actors: none
Preconditions: Desktop does not connect to internet.
Alternative flow: **Happens if the first step or second step in main flow fails** 3.2 Prompts the user to reconnect their internet. 1.2 Unable to synchronize with Cloud storage. 1.3 System should not delete data. It should store it for when connection to cloud is available.
Postconditions: None.

Alternative flow: Not sufficient Cloud Space
ID: 2.2
Brief description: Existing data size is larger than capacity available on cloud storage.
Primary actors: none
Secondary actors: cloud storage
Preconditions: Not enough space for data transfer (Syncwithcloud)
Alternative flow: 1. Prompts user about insufficient space 2. Unable to synchronise until sufficient space made available
Postconditions: None.

Question 4: Analysis class diagram

4.a



4.b)

Nouns are highlighted in Red

Verbs are highlighted in Blue

"The desktop application also allows users to create group(s) of friends/family to share own "achievements" with them, e.g., the daily count of steps, etc. Each user can register a new group: the user becomes the "owner" of the group, and the group is assigned a unique id. The owner of a group can invite new members to join the group using the functionality of the Desktop application to send emails via an external (SMTP) mail service. The invitations are either accepted or rejected by the respective invitees, who must access dedicated web-pages held on the web-portal of NewIdeas. Once the user accepts an invitation to join a group they are added to the group and may view the current data ("achievements") held in the cloud server on the members of the group(s)"

Noun phrases:

I will provide my judgement on the chosen noun phrases after duplicates removed. In analysis model we have 3 options: noun considered a class or noun considered an attribute or its ignored.

Desktop application -> Ignored, Generic, not part of problem domain. Desktop app can be referred to any app.

Users - class **users/account**, an essential abstraction from problem domain.

Group - Class **Group**, an essential abstraction from problem domain.

Friends/family -Ignored, not part of problem domain, part of users in group.

Share - ignored, part of **invite class**

Achievement – part of **documents**, an essential abstraction from problem domain.

(Count/steps) Documents – Class **Documents** for accessing user data, an essential abstraction from problem domain.

Owner – an attribute for 'group' class part of problem domain.

Members – Class **Members**, an essential abstraction from problem domain.

Functionality – doesn't relate to problem domain, ignored generic.

SMTP - Class **SMTP**, an essential abstraction from problem domain.

Invitation – Class **Invite**, an essential abstraction from problem domain.

Invitees – Attribute of invite class showing which users were invited, part problem domain.

Access – generic word, ignored.

Webpages – generic, ignored, not part of problem domain.

New ideas portal – class **new ideas portal**, an essential part of the problem domain.

Cloud server - Class **Cloud**, an essential abstraction from problem domain.

Verb phrases:

I will provide my judgement on the chosen verb phrases after duplicates removed. In analysis model we have 3 options: verb considered an operation or an association or its ignored.

Allows – ignored, generic word

Create –modelled as operation **RegisterGroup**() for account and controller class. Allows owner to register group

Can – ignored, generic word.

Becomes – ignored, generic word.

Assigned – ignored, generic word

Invite – Modelled as operation for Screen class, **invitegroup**() and controller class, allows owner to invite to group

Join – ignored, generic word

Using – ignored, generic word

Send – modelled as **sendMail**() for SMTP class

Email – Modelled as **checkmail**() in account class

Accepted – operation **accepted**() for newIdeaPortal

Rejected – operation **rejected**() for newIdeaPortal

Must – ignored, generic word

Dedicated – ignored, generic word

Held – ignored, generic word

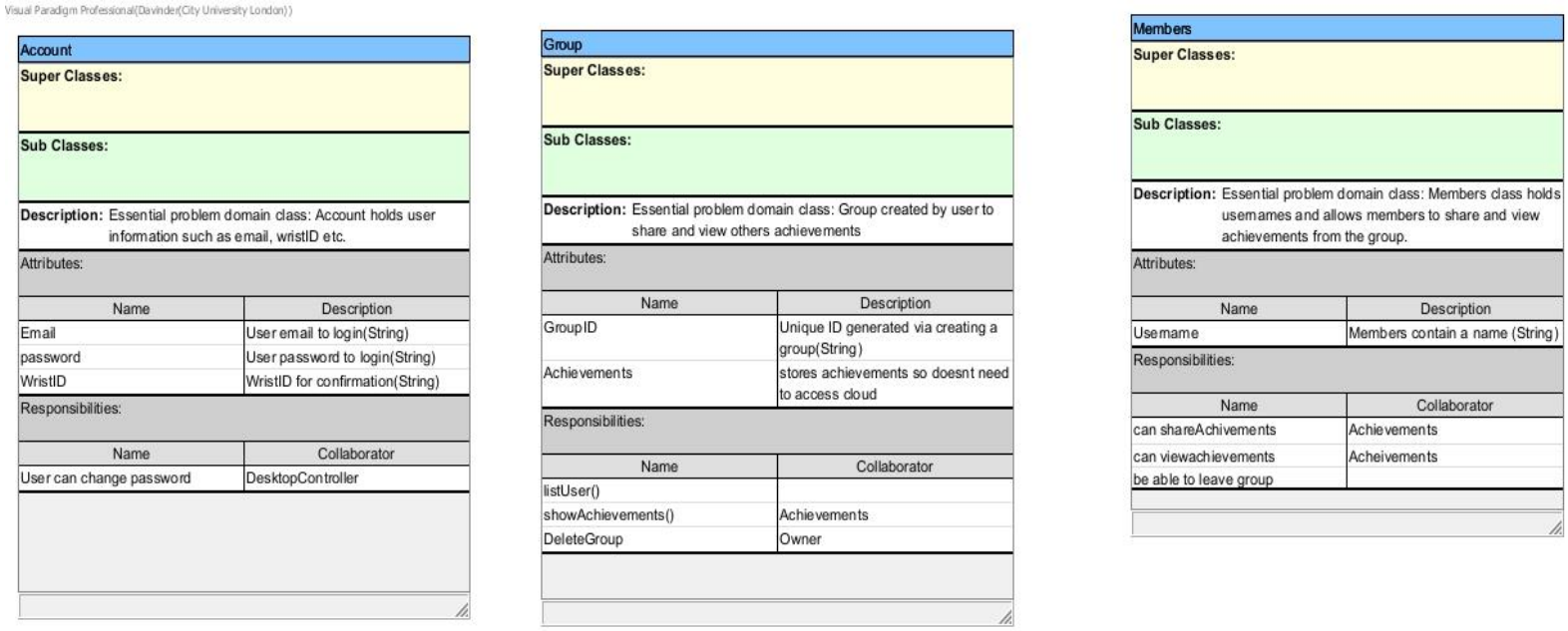
Are – ignored generic word

Added – modelled as **joinGroup**() for account class

May – ignored, generic word

CRC Diagram:

4.b.2



Question 5: Use case realisation (sequence diagram)