Edinburgh Napier UNIVERSITY

Edinburgh Garden Partners

SET11106

Information Systems Coursework Part 1:

Proposal for New Information System for the "Garden Partners" Initiative

Matriculation Number: 40091970

Main body word count: 3485

Edinburgh Napier UNIVERSITY

Contents Page Number

Executive Summary

The report following is a proposal for the design of a new information system for the "Garden Partners" initiative. An overview of the client's current system is presented, showing that the client currently relies on manual data entry and paperwork handling. The problems associated with this are broken down and it is shown that the inefficiency and lack of data security of the current system are significant and a barrier to the growth of the business.

The requirements for the new system are then prioritised and presented along with the high-level use case diagram for the overall system and the actor descriptions and developed scenarios for a subset of the system that will facilitate visits. The actors using the proposed system have been declared to be the clients and volunteers on the customer side and managers and administrators on the client side.

Detailed use case descriptions and activity diagrams are also presented which show in-depth examples of:

- Clients requesting visits and the system arranging/confirming them
- Clients and volunteers providing feedback for visits
- Managers and administrators accessed stored feedback and shared visit history

The report will then conclude with an argument for the adoption of the RAD software development lifecycle methodology. The client is lacking in technical expertise and has requested that a prototype of subsystem be developed quickly, and the RAD methodology can ensure client needs are heard while delivering and continually refining and initial functioning prototype.

1. Introduction

This report will detail a proposed information system for the “Garden Partners” initiative. The current client system and its associated problems will be analysed and presented, along with prioritised requirements for the proposed system. The report will also present a high-level use case diagram for the new system, along with actor descriptions, scenarios, detailed use case descriptions and activity diagrams for an initial subset of the system that the client has requested, along with additional functionality. The report will conclude with a discussion of the most suitable software development lifecycle approach for the development of the new system.

2. Current System Overview

After analysing the feedback from the client, it can be concluded that the current client system is based entirely on manual documentation handling and basic data entry.

The current processes and standards have been summarised below.

- The Administrator manually records all details in a locally stored spreadsheet and a physical folder with paper forms, and communicates with clients and volunteers over the phone.
- Volunteers sign up with the administrator who records details about them via paper forms. The volunteer details include name, address, contact number, gardening skills, and details on travel capabilities/needs.
- The client details that are recorded on paper forms include name and address, contact number, and preferences (this includes what gardening skills they are looking for).
- Additionally, volunteers have to either hold a current Health and Safety certification, or undergo one before they are allowed to carry out visits, the administrator keeps track of and updates the status of this on their spreadsheet.
- Volunteers also have to undergo a background check before they are allowed to begin carrying out visits. They apply for this via a form they fill out and hand to the administrator. This is carried out by an external organisation and the completion times vary. The administrator keeps track of their status – awaiting approval/Passed/Rejected – and updates the spreadsheet manually. Administrator informs volunteers of status update.
- Currently, the administrator also arranges keeps track of all details of each visit. These details include visit date and time, volunteer details and gardening skills utilised, client details, and notes of any feedback from client and volunteer, including any photos. See figure 1.

2.1 Scenario for Current Handling for Visit Facilitation

- A client calls in to request a visit
- Administrator manually retrieves client file to check preferences and details
- Administrator searches folder with paper files of volunteers to find match
- Administrator calls closest suitable volunteer to request availability
- If volunteer unavailable, repeat above process to find next closest, suitable volunteer
- In case of no match found, administrator informs client and attempts to rearrange requested visit time
- In case of identifying suitable volunteer and visit time, the administrator informs the client and records upcoming visit details on spreadsheet
- After the visit, the administrator sends a request for feedback to client and volunteer
- If feedback is received, updates feedback section of spreadsheet
- Location and filename of any received photos are also recorded on spreadsheet, and confirmation of permission to use for marketing purposes

Figure 1 Scenario for current handling of "Client requests visit"

2.2 Problem Definition

From the interviews we can identify a number of problems with the current system. The decision to highlight specific issues and problems has been taken under consideration of past examples of problem definition on similar systems (Supriatna, 2018; Fatimah, Supriatna & Kurniawati, 2018; Beynon-Davies & Holmes, 2002). I have broken down the problems into the following sections:

- **Manual data handling** - Much of the administrator's time is spent recording and updating details and information manually. Details are not updated in real time and, as a result, will often be inaccurate. (Supriatna, 2018)
    - *Errors/Security* - Overall greater risk of user error or typo/mistake generation as all is done manually on a spreadsheet. This also represents a security risk as if any error occurs in relation to recording background check status can lead to severe consequences for organisation and clients.

    - *GDPR* - Also potential GDPR risk as the data stored on clients and volunteers is not accessible to them, and organisation would struggle to fulfil Subject Access Requests if necessary, due to nature of current data storage. (Mondaq Business Briefing, 2016)
- **Communication** - All communication between volunteers and clients is carried out over the phone by administrator, no electronic communication, one line of communication total.
- **Visit facilitation** - Administrator arranges and facilitates each individual visit. Visibility of client and volunteer status is limited by manual processes and administrator has to call to reconfirm every time.
    - Administrator has to spend a lot of time arranging individual visits as no way to automate location/skill matches.
- **Spreadsheet** - A locally stored spreadsheet is both vulnerable from a security standpoint and susceptible to local system file corruption or failure.
    - Locally stored information is also inaccessible to the rest of the organisation which would limit **scalability**, MI reporting, and business decision making.
- **Task management** - Administrator cannot carry out more than one task at a time – can't update records and arrange visits simultaneously.

The problems above create weaknesses that include accuracy, response time and ease of access to information. This can be solved directly by the development of a computer-based information system (Fatimah et al., 2018). Key problems that have been highlighted in past studies include inefficient communication, the lack of ability to record/link data, and a lack of electronic communication – these have all been highlighted as a "key problem by users" (Beynon-Davies & Holmes, 2002, p.587).

3. Overall Proposal

In line with the problems identified with the current system and the requirements gathered and listed below, this report will propose that a new information system is developed and implemented for the client.

The functional and non-functional requirements gathered from the client have been listed below. The format for the presentation and prioritisation of the requirements has been modelled after the mnemonic MoSCoW—Must have, Should have, Could have, Won't have (Supriatna, 2018).

3.1 Functional and Non-Functional Requirements

Functional Requirements		
#	**Description**	**Priority (MoSCoW)**
#FR001	Allow volunteers and clients to sign up, log in and provide personal information – name/address/skill types/skill type preferences	Must(M)
#FR002	Allow volunteers to supply travel capabilities, Health and Safety Certification status and background check status – uploads of supporting documentation	Must(M)
#FR003	Allow volunteers to apply for Health and Safety certification and Background check (safeguarding) if needed	Must(M)
#FR004	Record certification and safeguarding status, update volunteer access appropriately and allow administrator to update safeguarding/certification status	Must(M)
#FR005	Allow clients to request visits/volunteers to accept visits	Must(M)
#FR006	Arrange visits - match up clients to suitable volunteers – filtered by travel distance and skill type preferences	Must(M)
#FR007	Record visit information – date/time/location, client and volunteer details, skill types used	Must(M)

#FR008	Allow for cancelling of visits from client/volunteer and option to rearrange visit date/time	Must(M)
#FR009	Allow for and record client and volunteer feedback including photo upload	Must(M)
#FR010	Allow creation of client-side accounts and for administrator and manager to log into system	Must(M)
#FR011	Allow administrator and manager to access all stored information on clients and volunteers, as well as visits and feedback	Must(M)
FR0012	Allow administrator and manager to download saved information	Must(M)
#FR013	Allow for updating of client and volunteer skillset	Must(M)
#FR014	Send notifications to volunteers with 3-month notice alert of certification expiry	Should(S)
#FR015	Trusted Friends and Family – option for clients to provide details of external individuals who will receive notifications of their upcoming visits and access to visit feedback/photos	Should(S)
#FR016	Display (preferably on a map) current volunteer and visit location information	Should(S)
#FR017	Alert when clients consistently give bad feedback – to indicate communication needed	Should(S)
#FR018	Alert when volunteer is likely in need of assistance or mentorship and record mentor/mentee status	Should(S)
#FR019	Allow generation of quarterly MI report for Manager	Could(C)
#FR020	Fully automate safeguarding/certification status updates	Could(C)

Table 1.1 Functional requirements

Non-Functional Requirements		
#	**Description**	**Priority (MoSCoW)**
#NR001	Ensure compliance with GDPR	Must(M)

#NR002	Hold records for 7 years	Must(M)
#NR003	Be accessible to user base which is likely to be retirees/disabled	Must(M)
#NR004	Ensure system is scalable - plans to increase from 6 volunteers and 12 clients to 50000 volunteers and 200000 within 5 years	Must(M)

Table 1.2 Non-functional requirements

3.2 High-Level Use Case Diagram

This section details the high-level use case diagram for the proposed system.

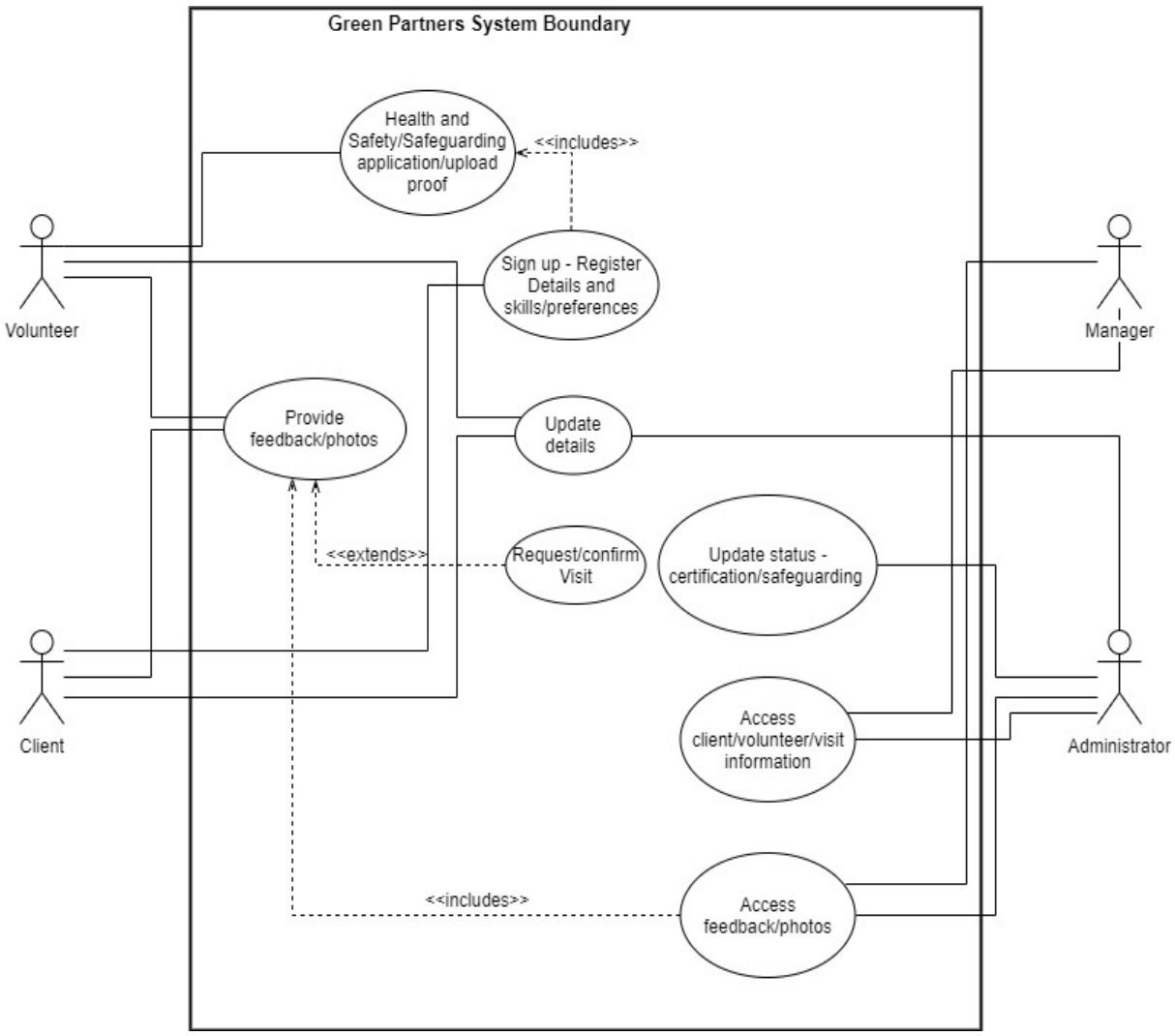

Figure 2 High-level use case for proposed system

3.3 Essential/Must-Have functionality

Client and Volunteer interaction/functionality

The essential functionality this system will provide has been listed below with reference to Table 1.1 and Figure 2. Clients and volunteers will be able to:

- Sign up and create accounts and provide and update personal information - #FR001
    - This will include selecting which gardening skills they can provide/require, examples provided by the client so far include:
        - Grass cutting
        - Weeding
        - Planting
        - Hedge cutting
        - Tree surgery
- Supply travel capabilities and (if applicable) provide proof of Health and Safety certification and safeguarding status- #FR002
- Apply for Health and Safety certification and Background (safeguarding) check - #FR003
- Request a visit - #FR005
- Accept a visit arrangement - #FR005
- Cancel/rearrange an arranged visit - #FR008
- Provide feedback on past visits and/or upload photos - #FR009
- Update their saved details and preferences/skillset - #FR013

Manager and Administrator interaction/functionality

- Allow for the creation of client-side accounts - #FR010
- Allow for Manager and Administrator accounts to log in and access stored details - #FR011 –relating to:
    - Client/Volunteer information
    - Past visits
    - Feedback – can be used to inform best practice as requested
    - Client and volunteer visit history
- Allow for the downloading of stored information and feedback - #FR012

- Allow for the updating of client and volunteer details – #FR013
- Allow for the administrator to update the safeguarding and certification status of volunteers - #FR004

System functionality

- Store all provided client and volunteer details and information - #FR001
- Facilitate visits and store full visit details - #FR006
- Store provided feedback details and photo uploads - #FR009
- Allow for the administrator and manager to download stored information - #FR0012
- Display shared client and volunteer visit history, including data about volume and total visits - #FR011
- Limit/restrict or grant access to system functionality based on the safeguarding and certification status of the volunteers as defined by the client - #FR004
- Adhere to non-functional requirements listed above – #NR001-004

A system that fulfils the above will meet the essential needs of the client as expressed in the recorded interviews.

3.4 Additional Functionality

This section will detail additional functionality/interactivity that the client has expressed a desire for, but may not be 100% essential in the short-term, see table 1.2.

- Allow for the display of current volunteer/client location on a user-friendly map for the assistance of the Manager's reports and business decision making. – #FR016
- Alert manager/administrator when a client is consistently giving negative feedback - #FR017
- Send notifications to volunteers when their Health and Safety certification is due to expire in 3-months – #FR014
- Provide clients a "Trusted friends and family" option so that nominated external individuals can monitor a client's activity- #FR015

- Alert manager/administrator when a volunteer is likely in need of assistance or mentorship and record mentor/mentee status - #FR018
- Allow for the generation of the Manager's quarterly MI report – #FR019
- Automate the safeguarding/certification status updates (as the safeguarding is completed by a third party this will prove too complex for the initial system model) - #FR020

4. Actor Descriptions

This section will provide detailed actor descriptions for the relevant roles involved in the proposed system. The format of the following descriptions has been taken from sources on UML formatting (Britton & Doake 2005; Bennett, McRobb & Farmer, 2010).

The Administrator: uses the system to access and update client and volunteer details, including updating safeguarding and certification status and volunteer and client access and privileges. As the safeguarding/certification applications will be done via a third party, the Administrator will retain responsibility for verifying certificates once received and updating volunteer status. They can also access and download stored visit and feedback details. They may also establish and update mentor/mentee relationships, and use the system to communicate directly with volunteers and clients regarding any queries. The system interfaces presented to this actor will facilitate the above functions and not present report generating or data analysis options.

The Administrator can be the current administrator or the manager choosing to perform the above administrative duties. This role can also be played by suitable future hires as the “Garden Partners” initiative expands.

The Manager: uses the system to access any and all stored information around clients, volunteers, visits, and feedback. They will use the system to display stored information in ways that will help facilitate business decision making and the generation of reports – examples may include: geographically visualising the current volunteer base to target areas of opportunity; accessing and downloading feedback information to inform best practice or for marketing purposes; and accessing data around volunteer and client numbers for report generation. The system interfaces presented to this actor will facilitate the above functions and not allow for the updating of client/volunteer personal details or statuses.

The Manager can be the current manager carrying out the above duties or the current administrator if they have been tasked with report/Management Information generation.

The Client: uses the system to sign up and register their personal information – address, contact information, name, DOB, special requirements, and the type of gardening skills they will require. They can also request visits, cancel or rearrange visits, and update their stored information. They are also allowed to provide feedback and/or upload photos for visits they have been a part of.

The Client will be the current and future base of clients using the “Garden Partners” initiative.

The Volunteer: uses the system the sign up and register their personal information – address, contact information, name, DOB, travel capability/requirements, and the type of gardening skills they can offer. They will be unable to take part in visits until a safeguarding check and Health and Safety Certification have been completed. Volunteers can upload proof of a valid Health and Safety Certification themselves but must submit a Safeguarding check via the system. They can confirm requested visits and provide feedback and/or photos for visits they have been a part of.

The Volunteer will be the current and future volunteer base using the “Garden Partners” initiative.

4.1 Scenarios

This section will detail relevant typical scenarios that relate to the initial subset of the system requested by the client as well as additional functionality. These scenarios will inform the following Use Case Descriptions and Activity Diagrams (Britton & Doake, 2005; Bennett et al., 2010).

- Client Rachel accesses company website
- She selects the sign up option
- She is presented with a sign up form which asks for personal details and gardening skill needs
- She fills in and submits all required information
- The system generates an account for her

Figure 3.1 Successful “sign up” for client

- Volunteer Karen accesses the company website
- She selects the log in option
- She is presented with the log in screen which asks for email address and password
- Karen submits the correct log in information
- The system verifies the details and presents her with the welcome screen

Figure 3.2 Successful "log in" for volunteer

- Client John logs in and requests a visit
- He selects that the visit take place next Wednesday between 10am-1pm
- The system searches for a suitable volunteer that matches John's skill set needs and availability and is within traveling distance
- The system finds suitable volunteer, Jane, and sends a visit request
- Jane confirms they can attend
- The system alerts the client and asks for final visit confirmation
- Client confirms
- The system generates the visit details and sends them to both John and Jane

Figure 3.3 *Successful scenario for the use case "Client requests visit" and "Match volunteer/client and arrange Visit"*

- Client Bobby logs in and requests a visit
- He selects that the visit take place tomorrow between 12pm and 5pm
- The system searches for a suitable volunteer that matches Bobby's skill set needs and availability and is within travelling distance
- The system fails to find a suitable volunteer and asks Bobby if they would like to attempt to rearrange the visit
- Bobby chooses not to rearrange and does not receive a visit

Figure 3.4 Scenario for the "Client requests visit" use case where the system fails to confirm a suitable volunteer

- Client John logs in and accesses his stored visit details
- The system prompts John with the option to provide feedback for his last visit

- John enters in text feedback for the visit and submits it
- The system asks John if he would also like to provide any photos
- John uploads a photo relating to the visit
- The system thanks John for providing feedback and returns to previous window

Figure 3.5 *Successful scenario for the use case "Provide feedback"*

- Administrator Tara logs in and accesses the stored visit details
- The system presents options to access stored client/volunteer or specific visit details
- Tara selects a visit that took place last Wednesday at 11am between client John and volunteer Jane
- The system presents visit details and allows Tara to access the stored feedback information, as John has provided some
- Tara accesses the stored feedback information and selects the option to download the feedback and photo that John has provided

Figure 3.6 *Successful scenario for the use case "Access feedback/photos"*

- Manager Jessica logs in and accesses the stored visit details
- The system presents options stored client/volunteer or specific visit details
- Jessica selects a visit that took place last Wednesday at 11am between client John and volunteer Jane
- The system presents visit details and allows Tara to access the stored feedback information, as John has provided some
- Jessica decides to access the shared visit history between John and Jane instead
- The system presents a list of the past visits John and Jane have shared
- Upon seeing that there is only one so far, Jessica decides to exit

Figure 3.7 Successful scenario for "Access visit history"

5. Use Case Descriptions

This section will provide detailed Use Case decryptions that related to the above scenarios (Britton & Doake, 2005; Bennett et al., 2010).

5.1 Figure/diagram relationships

The following figures represent detailed use case descriptions, the relationship between scenario, use case description, and activity diagram are as follows:

- **Figure 4.1** – Use case description – “Sign Up”, see Scenario **Figure 3.1** and Activity Diagram **Figure 5.1**
- **Figure 4.2** – Use case description – “Log In”, see Scenario **Figure 3.2** and Activity Diagram **Figure 5.2**
- **Figure 4.3** – Use case description – “Visit facilitation” see Scenario **Figures 3.3 and 3.4** and **Activity Diagram figure 5.3**
- **Figure 4.4** – Use case description – “Client/Volunteer provides feedback” see Scenario **Figure 3.5** and Activity Diagram **Figure 5.4**
- **Figure 4.5** – Use case description – “Manager/Administrator accesses feedback/details” see Scenario **Figures 3.6 & 3.7** and Activity Diagram **Figures 5.5 & 5.6**

Use case: Client/Volunteer signs up with system

Preconditions: None

Actors: Client/Volunteer, System

Goal: To successfully create an account for the client/volunteer

Overview:

A client/volunteer accesses the company website which provides an interface with the system. They are prompted to fill in the required details to generate an account. Once the required details are submitted, the system generates an account for them.

Cross-reference:

#FR001

Typical course of events:

Actor Action	System Response
1. Client/Volunteer accesses system site	2. System site provides welcome page and necessary options
3. Client/Volunteer selects “Sign up” option	4. System presents sign up screen showing sign up form
5. Client/Volunteer enters all required information and submits form	6. System confirms required information has been submitted
	7. System generates account for client/volunteer

Alternative courses:

Steps 5-7 Client/Volunteer fails to fill in all required fields, system then reloads page with an alert error and re-prompts for required information

Steps 5-7 Client/Volunteer choses to exit after failed attempt

Figure 4.1 – Use case description – “Sign Up”

Use case: Client/Volunteer logs into system

Preconditions: Client/Volunteer has successfully signed up with system

Actors: Client/Volunteer, System

Goal: To successfully log in and access account on system

Overview:

A client/volunteer accesses the company website which provides an interface with the system. They select the log in option and are prompted for their log in details. They submit the correct details and are given access to their system account.

Cross-reference:

#FR001

Typical course of events:

Actor Action	System Response
1. Client/Volunteer accesses system site	2. System site provides welcome page and necessary options
3. Client/Volunteer selects “Log In” option	4. System presents log in screen with required fields – email/username and password
5. Client/Volunteer enters correct log in information	6. System verifies log in information
	7. System presents account welcome page and relevant system access

Alternative courses:

Steps 5-7 Client/Volunteer fails to enter correct information – system will then display error alert and allow a set number of new log in attempts

Steps 5-7 Client/Volunteer choses to exit instead of attempting again

Figure 4.2 – Use case description – “Log In”

Use case: Client requests visit and system matches with volunteer

Preconditions: Client has already signed up and registered their details/preferences and is logged in

Actors: Client, System, Volunteer

Goal: To arrange a visit between client/volunteer

Overview:

A client requests a visit from the system. The system will match the client with a suitable volunteer and send out a request to confirm visit attendance. Once confirmed, the visit will be arranged, volunteer and client will be notified of details and details will be saved by the system.

Cross-reference:

#FR005, #FR006, #FR007

Typical Course of Events:

Actor Action	System Response
1. Client logs in and requests visit	2. System matches client to volunteer
	3. System sends request/s to suitable volunteer/s
4. Suitable volunteer confirms	5. Once accepted by volunteer, system alerts client and asks for final confirmation
6. Client confirms	7. System compiles and saves visit details and sends confirmation to client and volunteer

Alternative courses:

Steps 2-3 Suitable volunteer may not be found – system prompts client to ask if they would like to change requested visit date or time and attempt to re-match

Steps 4-5 Suitable volunteer does not confirm – system repeats steps 2-3 for suitable volunteers if other options available to system, if not,

	prompts client to ask if they would like to change requested visit date or time and attempt to re-match

Figure 4.3 – Use case description – "Visit facilitation"

Use Case: Client/volunteer provides feedback/photo/s

Preconditions: Visit has been arranged and completed and client/volunteer has logged in

Actors: Client/Volunteer, System

Goal: For client/volunteer to successfully provide feedback/photo/s

Overview:

Client/Volunteer logs in to provide feedback and/or upload photo/s about a specific visit in the past.

The system stores feedback details and photo/s provided, if any.

Cross Reference:

#FR009

Typical Course of Events:

Actor Action	System Response
1. Client/Volunteer selects feedback option for specific visit	2. System provides feedback input window
3. Client/Volunteer enters feedback	4. System stores feedback and provides photo upload option
5. Client/Volunteer uploads selected photo/s	6. System saves photo/s and closes upload window

Alternative Courses:

Steps 5-6 Client/Volunteer chooses not to upload photo/s. System then closes window

Figure 4.4 – Use case description – "Client/Volunteer provides feedback"

Use Case: Manager/administrator accesses visit details and feedback

Preconditions: Visit has been arranged and completed. Client/Volunteer has provided feedback/photo/s, administrator/manager has already logged into system

Actors: Manager, Administrator, System

Goal: For feedback and visit details to be accessed and/or retrieved by Manager or Administrator

Cross-Reference:

#FR010, #FR011, #FR012

Overview:

Manager/Administrator logs in and accesses visit details. Administrator/Manager chooses to access visit feedback and stored photo/s.

Manager/Administrator downloads feedback information.

Typical Course of Events:

Actor Action	System Response
1. Administrator/Manager selects access to saved system information	2. System provides information options – client/volunteer/visit
3. Administrator/Manager selects specific visit	4. System provides visit details and option to access feedback/photo/s if applicable, and the specific client/volunteer visit history – shared or individual
5. Administrator/Manager accesses feedback option	6. System provides feedback and photo details and option to download information in relevant format
7. Administrator/Manager downloads information	8. System window returns to previous window with information access options

9. Administrator/Manager selects option to view volunteer/client shared visit history	10. System provides list of past visits between selected client and volunteer if available, and presents same options as above for each

Alternative Courses:

Steps 4-8	No feedback has been provided for selected visit/selected visit has not taken place yet. No option to access feedback information.
Steps 3-7	Administrator/manager chooses to access shared or individual visit history instead of feedback option
Steps 7-8	Administrator/Manager decides not to download feedback information and exits feedback window.
Steps 10	May be no shared visit history between selected client and volunteer

Figure 4.5 – Use case description – "Manager/Administrator accesses feedback/details"

6. Activity Diagrams

This section will present detailed activity diagrams for the above use case descriptions (Britton & Doake, 2005; Bennett et al., 2010). All activity diagrams have been presented with the same preconditions as their respective use case descriptions – for example, “Sign Up” and “Log In” are assumed to have been completed successfully.

Notably, the activity diagram for the use case description “Manager/administrator accesses visit details and feedback” has been split into two parts to more clearly represent the alternative courses of action available in this use case description.

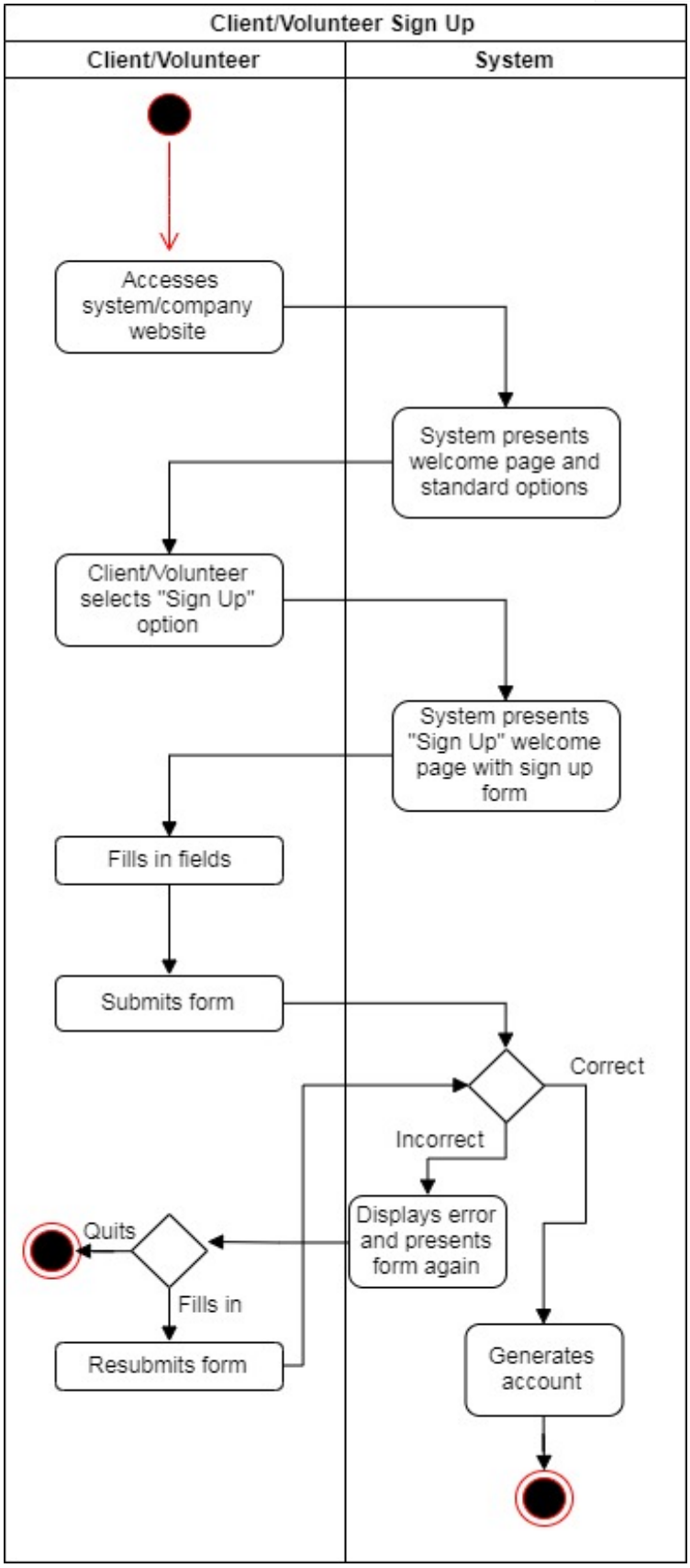

Figure 5.1 Activity Diagram for "Client/Volunteer signs up with system"

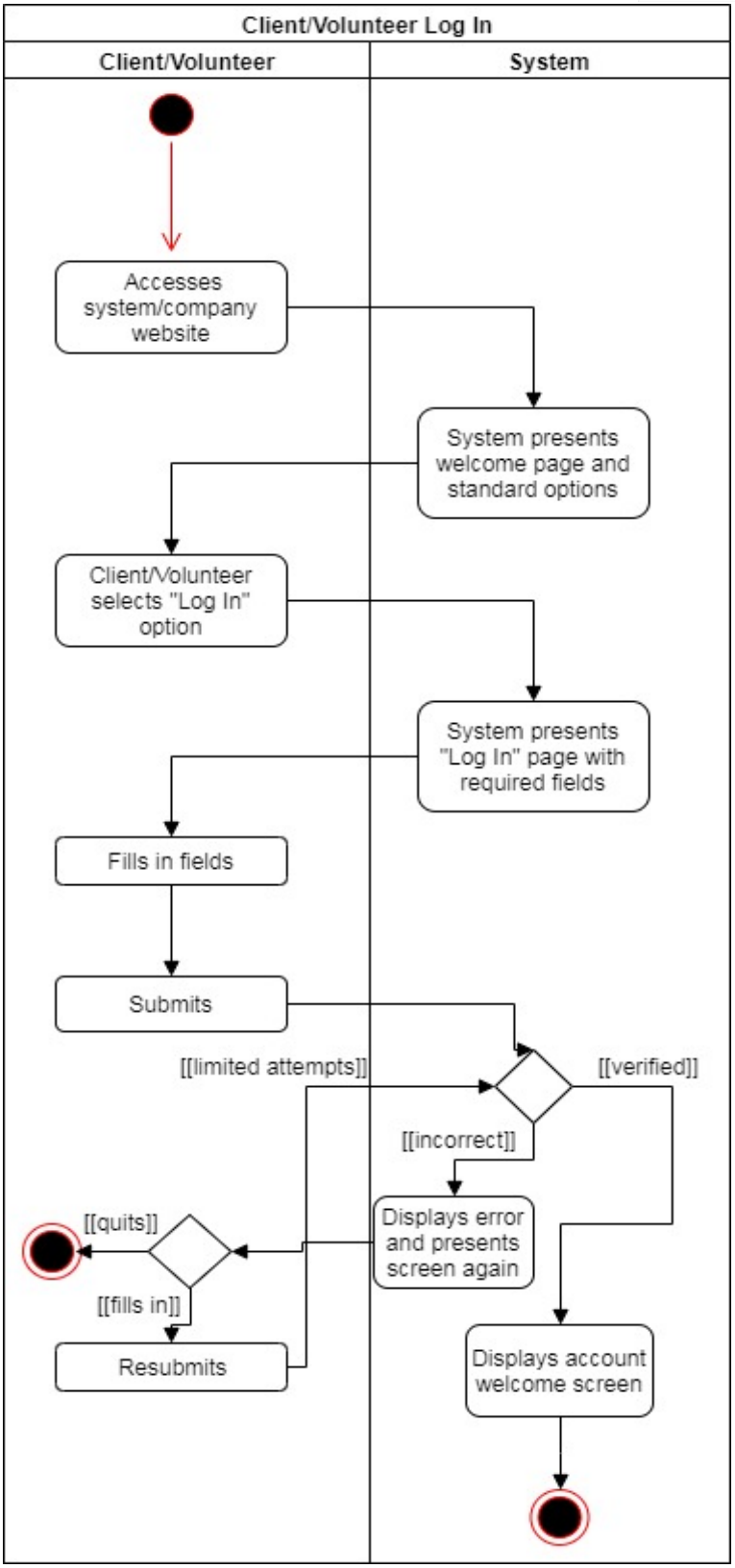

Figure 5.2 Activity Diagram for "Client/Volunteer logs into system"

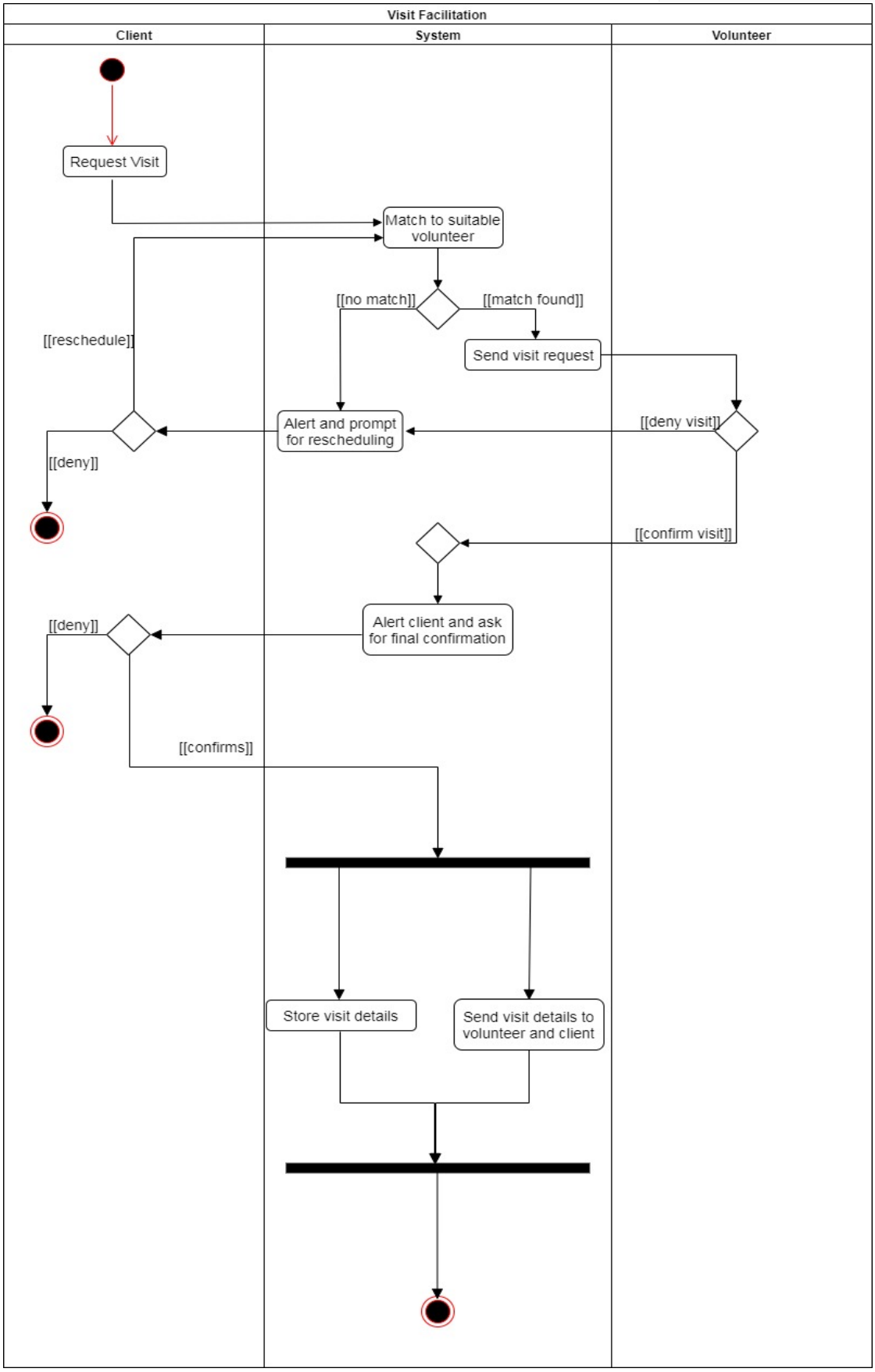

Figure 5.3 Activity diagram for "Client requests visit and system matches with volunteer"

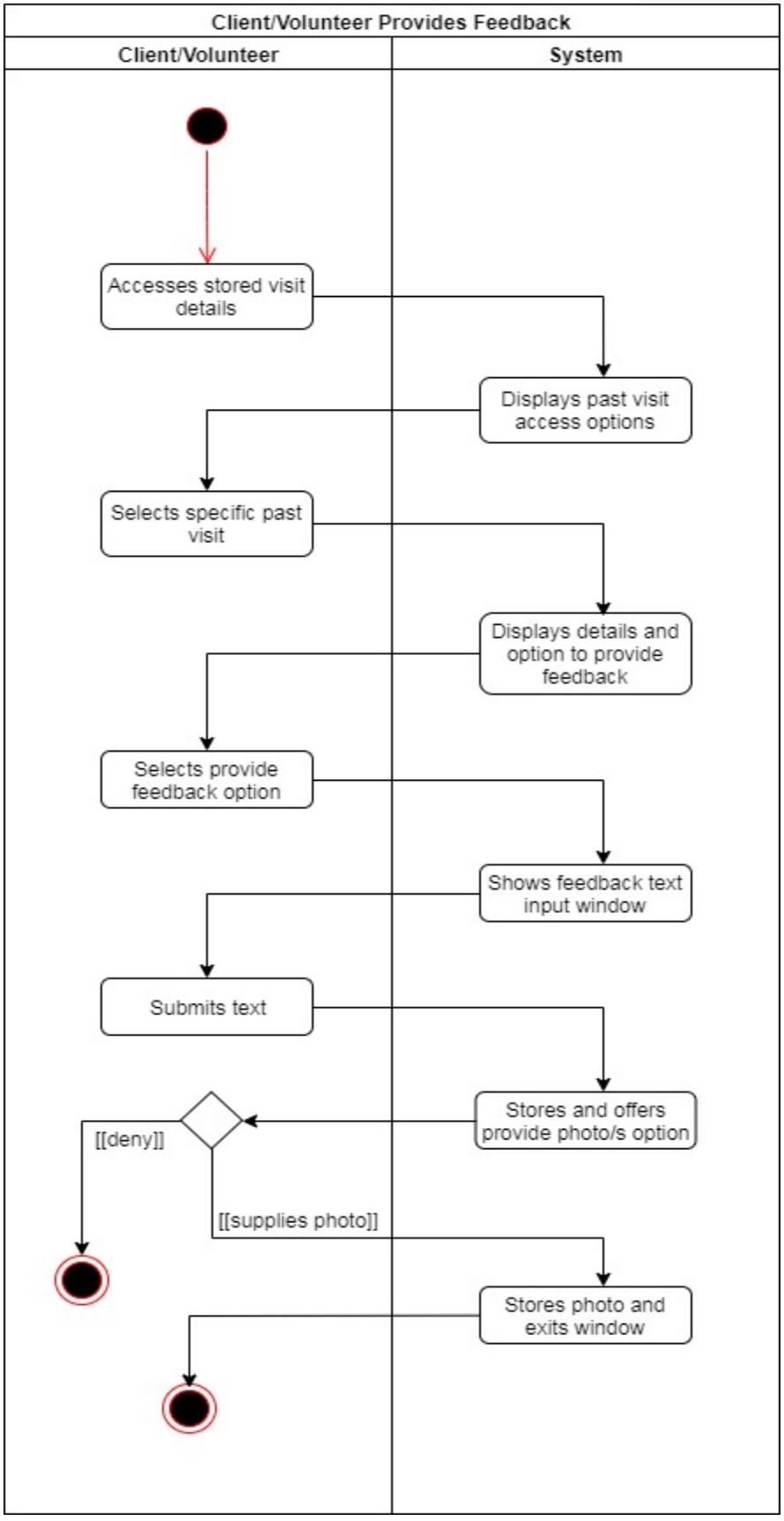

Figure 5.4 Activity diagram for "Client/volunteer provides feedback/photo/s"

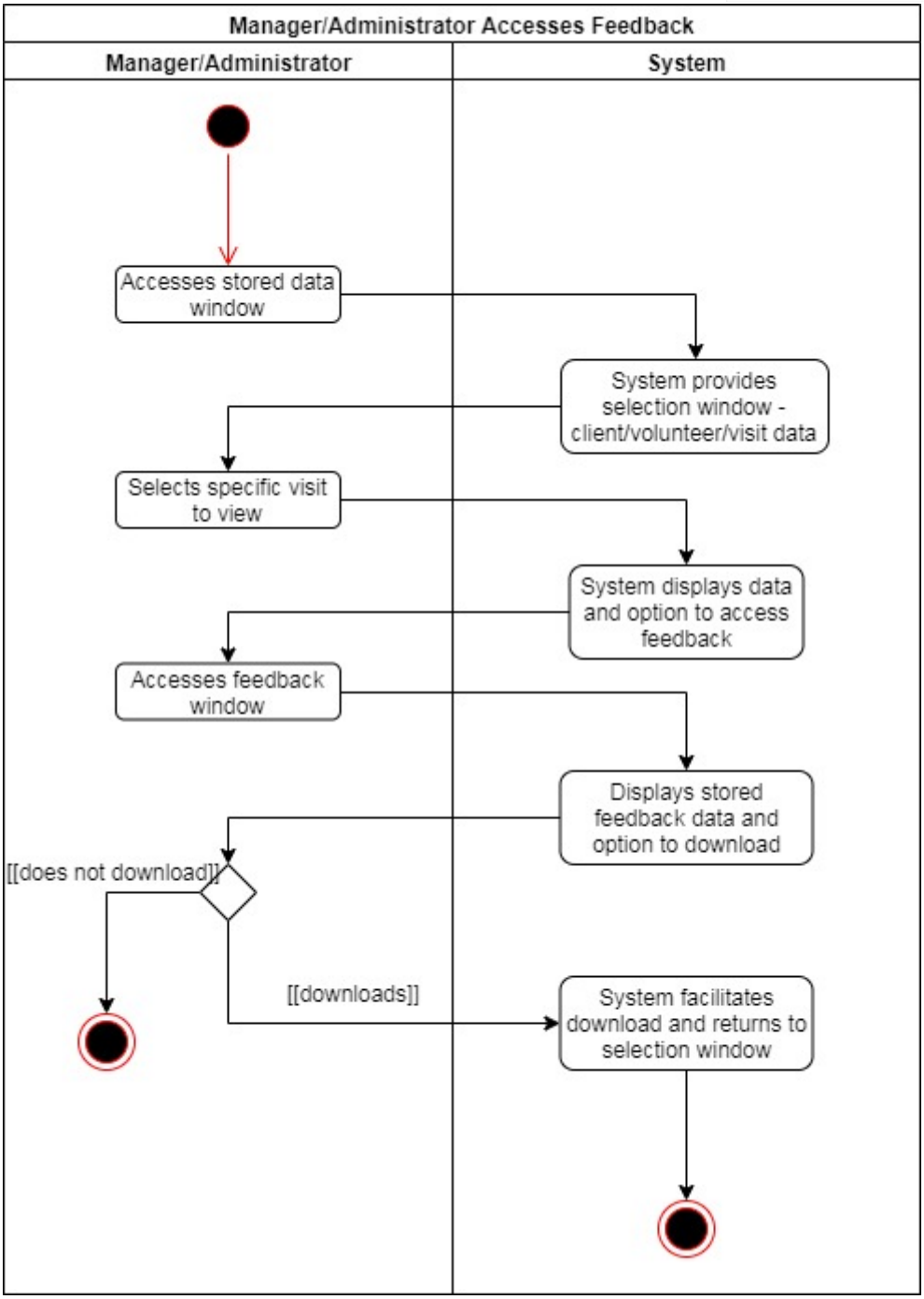

Figure 5.5 Activity diagram for "Manager/administrator accesses visit details and feedback" part 1 – feedback download

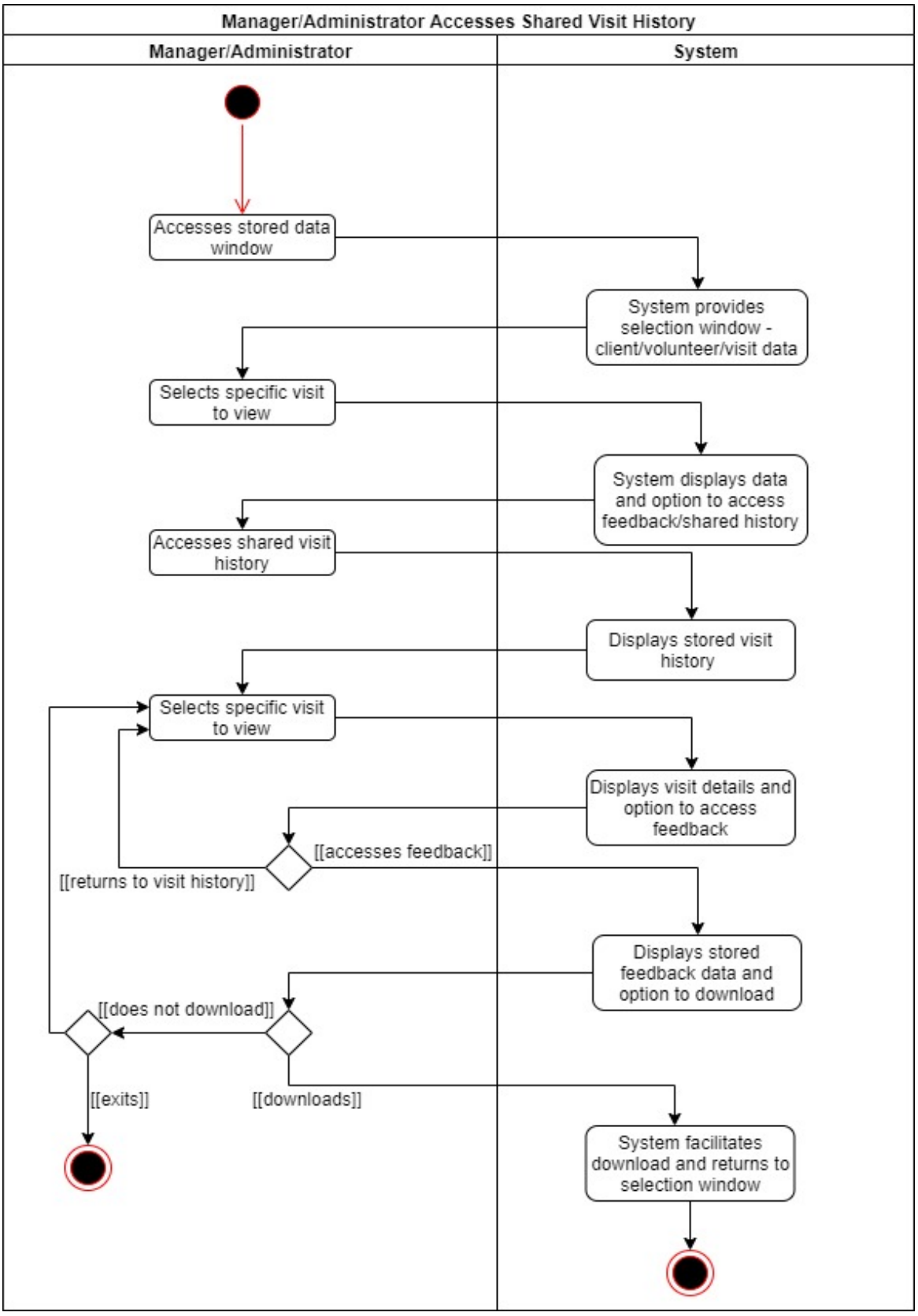

Figure 5.6 Activity diagram for "Manager/administrator accesses visit details and feedback" part 2 – shared history

7. SDLC Approach

Based on the nature of the requirements gathered and the current organisational infrastructure, this report will be recommending that the Software Development Lifecycle approach applied by this project is the Rapid Application Development methodology.

RAD is a “linear sequential software development model that focuses on a very short development cycle” (Fatimah et al., 2018, p.197). The client, Community Concern, has expressed that they would like a subset of the desired system to be prototyped as quickly as possible. RAD focuses on quickly and efficiently prototyping system subsets or “work models” in order to ensure user needs are being met and that user requirements shape the development project (Fatimah et al., 2018, p.197). An additional reason that RAD is suitable for the project is that the steps in the RAD method “are very clear and easy to follow” (Fatimah et al., 2018, p.196). As the two main stakeholders providing requirements and feedback for the system are not technical and have “limited prior exposure” to information systems and “no previous bespoke information system has been built for the organisation” so far, a RAD approach will help ensure the best collaboration and communication between the technical team and the users (Beynon-Davies & Holmes, 2002, p.584).

Additionally, in RAD, “teams are made up of both developers and users” who both can make design decisions, and RAD approaches will often budget for “activities which help to improve the cohesiveness of teams” (Beynon-Davies & Holmes, 2002, p.583). RAD projects “can run relatively faster because designers and users work together in the development stage” (Fatimah et al., 2018, p.196). Therefore, following a development approach that prioritises clear and easy to follow steps, that “can clarify every step taken”, and that puts the user requirements and feedback at the forefront of the project, will help ensure that the development of the new system is undertaken in the most appropriate way for the client and the users (Supriatna, 2018, p.2). This will also help ensure that the user needs and requirements can be utilised in the most efficient way, as RAD prioritises their engagement and facilitates their understanding.

RAD has been said to be especially suited to “small, relatively, low-key projects”, with underlying principles that “can be applied to any software project”

(Howard, 2002). Due to the current small userbase, this is another reason that a RAD approach will be suitable to the client. A RAD approach can ensure that a working subsystem is completed efficiently to help inform future business decisions and the plans for expansion shared by the Manager. Therefore, this approach will not only ensure client needs are met in the short term, but that a scalable, robust system will be built that can fulfil long-term business needs.

Figure 5 RAD model (Supriatna, 2018)

Project progress in RAD is carried out of 4 stages – Business Modelling, Data Modelling, Process Modelling, and Application Generation (see Figure 5). Using these steps, RAD continually refines the initial prototype and revaluates the client requirements to ensure accuracy of the deliverables (Beynon-Davies & Holmes, 2002; Theresharov, Krushkov, Stoyanov, & Popchev, 2017).

7.1 Project Team

RAD is “characterised by small development teams” usually between “four to eight persons” (Beynon-Davies & Holmes, p.583). The project team itself should be “made up of both developers and users” that will both be empowered to make design decisions (Beynon-Davies & Holmes, p.583). The users themselves can also potentially “act as mangers to projects” (Beynon-Davies & Holmes, p.583).

The first goal of RAD is to create a modular system or prototype that can be shared with the client to help continually shape the rest of the development process (Theresharov et al., 2017). The project team will “continually refine the prototype into something that is deliverable at the end of the project” (Beynon-Davies et al., 2002, p.583). All RAD team members “must be skilled in terms of communication”, as the

evolving dialog between the users and the developers is what determines the success of the RAD approach. To account for this, RAD also prioritises activities that continuously improve collaboration and communication (Beynon-Davies & Holmes, 2002).

The beginning of each phase of development should be “marked by an initiation meeting between developers and user representatives” (Beynon-Davies & Holmes, 2002, p.585). This is to enable the objectives of each scope to be clarified, negotiated, and classified into achievable tasks. Frequency of interaction between developers varies – it can be as often as every day or weekly, monthly or quarterly (Beynon-Davies & Holmes, 2002). The users should possess detailed “knowledge of the application area” and should be focused on the project work (Beynon-Davies & Holmes, 2002, p.583). Therefore, frequent and clear communication is necessary to enable the users to provide input that will drive the project forward in an efficient way. Developers should be “skilled in the use of rapid development tools” and should be completely focused on the development project (Beynon-Davies & Holmes, 2002, p.583).

7.2 Tools

RAD requires “good support from tools for rapid developmental change”. Software technology and the “programming language and the tools, servers, libraries etc. which allow us to create software applications” should be considered a primary tool for development-focused information systems projects (Theresharov et al., 2017, p.111). The tools required by a RAD approach will usually be some combination of fourth generation programming languages, graphical user interface builders, database management systems and computer-aided software engineering tools - examples include: OutSystems, Zoho Creator, mySQL and Oracle DBMS, and Software Ideas Modeller (Beynon-Davies & Holmes, 2002).

The utilisation of such tools can allow for “in-situ” changes to be made to prototypes at user-developer meetings, which is ideal for the collaborative nature of RAD (Beynon-Davies & Holmes, 2002, p.583).

Additionally, open-source project management and backlog prioritisation tools like Trello should be utilised to facilitate the organisation of tasks and deliverables.

7.3 Infrastructure

Organisational and team infrastructure is also a critical component of a successful RAD project. Proponents of RAD have the "tendency is to rule out the applicability of RAD for large-scale, infrastructure projects" (Beynon-Davies & Holmes, 2002, p.583). As such, the current infrastructure of the "Garden Partners" initiative is already quite suitable for a RAD approach, due to the small-scale nature of the initiative.

Changes to the current working arrangements would have to be made to include the primary users on the client side, the Manager and the Administrator, in the development team and the iterative cycle.

Most RAD projects are "on applications that are highly interactive, and that have a clearly defined user group" (Beynon-Davies & Holmes, 2002, p.583). The changes to the organisational infrastructure mentioned above will also have to encompass a change to the way the organisation interacts with and gets feedback from its users, as gathering requirements from the clients will have to be supplemented with regular feedback from targeted users of the prototype system. As such, it will be necessary to maintain an infrastructure that allows for improved, targeted engagement with the external user base for the system (Beynon-Davies & Holmes, 2002).

7.4 Conclusion

RAD ensures that the user needs are not only put at the forefront of the development process, but that the users themselves can help design and develop the product throughout the project lifecycle. This will allow for the delivery of a system that functions in the desired way, offering a robust and flexible interface and functionality. A RAD approach will also allow for the early delivery of a functioning prototype that can help inform business decisions earlier than other approaches. As such, plans for organisational and userbase expansion can be implemented earlier and more accurately, helping to ensure that the system scalability is developed in line with these changes as the project develops. The user-focused nature of a RAD approach can help guarantee that the end product is trusted and maintainable by the user and future internal technical teams, as the functionality of the system has been tailored by the user themselves. Therefore, using a RAD approach will help ensure that a

fully functioning, robust, scalable, trusted and maintainable system is produced for the client.

8. Overall Conclusion

It has been shown that the current business processes and standards utilised by the “Garden Partners” are extremely problematic and a barrier to the primary goals of the business. The requirements gathered and the system design proposed would eliminate the problems with the current business system and help facilitate the desired business expansion targets. The new system will also help security and automate most of the current business processes, and present a new, user-friendly interface for both the client and the customer base. If adhered to, the details in this proposal will ensure that an end product is produced that meets all required business needs.

Reference List

Britton, C., & Doake, J. (2005). *A student guide to object-oriented development*. Amsterdam ; London: Elsevier Butterworth-Heinemann.

Bennett, S., McRobb, S., & Farmer, R. (2010). *Object-oriented systems analysis and design using UML* (4th ed.). Maidenhead: McGraw-Hill Higher Education.

Supriatna, A. (2018). Designing library information system using rapid application development method. *IOP Conference Series: Materials Science and Engineering, 434*(1), 1-6.

Howard, A. (2002). Rapid Application Development: Rough and dirty or value-for-money engineering? *Communications of the ACM, 45*(10), 27-29.

Fatimah Dini Destiani Siti, Supriatna Asep Deddy, & Kurniawati Rina. (2018). Design of personnel information systems using rapid application development method. MATEC Web of Conferences, 197, 03016.

Beynon-Davies, P., & Holmes, S. (2002). Design breakdowns, scenarios and rapid application development. Information and Software Technology, 44(10), 579-592.

Cheresharov, Krushkov, Stoyanov, & Popchev. (2017). Modules for Rapid Application Development of Web-Based Information Systems (RADWIS). Cybernetics and Information Technologies, 17(3), 109-127.

The Subject Access Request That Led To A Security Breach, Or Why Having A System To Respond To Access Requests Is Essential. (2016). Mondaq Business Briefing, p. Mondaq Business Briefing, Nov 10, 2016.