**System Requirements and Design**



Attendance System

Preface

This document is a combination of two documents which are normally separated; Software Requirements Specifications and Software Design Documents. This merger has been decided due to the relatively small size of the project, and will provide both the development team and stakeholders with a better overview.

This document is also available online: //TODO: insert address for webpage

Current document version and its historical changes can be viewed in *Table 1: Document version control.*

Table : Document version control

|  |  |  |
| --- | --- | --- |
| **Document version** | **Dated** | **Whats new?** |
| 1.0 | 17.01.2017 | Document created |
| 1.1 | 24.01.2017 | Added flowcharts and figures, minor structural changes |
| 1.2 | 31.01.2017 | Added database model figures, added database and UML chapters |

Nomenclature

DAL – Data Access Layer

GPIO – General-Purpose Input/Output

GUI – Graphical User Interface

NUC – Next Unit of Computing

PC – Personal Computer

RFID – Radio Frequency Identification

RS232 – Recommended Standard number 232

SDD – Software Design Documents

SRS – Software Requirements Specifications

SQL – Structured Query Language

Table Of Contents

Preface 2

Nomenclature 3

Table Of Contents 4

1 Introduction 6

1.1 Purpose 6

1.2 Scope 6

1.3 Similar Existing Attendance Systems 6

2 System Overview 8

2.1 Module Breakdown 8

2.1.1 Computer 8

2.1.2 Web Application w/ Database 9

2.1.3 Reading Terminal 9

2.1.4 Arduino Microcontroller 9

2.1.5 RFID Reader 9

2.1.6 RFID Tag 9

2.2 System Flow 10

3 Technical Requirements 11

3.1 Functional Requirements 11

3.1.1 System 11

3.1.2 Client 11

3.1.3 User 11

3.1.4 Attendee 11

3.1.5 Host 12

3.1.6 Appointment 12

3.1.7 Course 12

3.1.8 Reading Terminal 12

3.2 Non-Functional Requirements 12

3.3 Graphical User Interface Specification 13

3.4 Data Communication Between Modules 15

4 System Architecture 16

4.1 Three-Tier Multi-layer Structure 16

4.1.1 Presentation Layer 17

4.1.2 Business Layer 17

4.1.3 Data Access Layer 17

5 Database 19

5.1 Logical and Physical Data Model 19

5.2 Database Description 20

5.2.1 APPOINTMENT Table 20

5.2.2 HOST Table 21

5.2.3 USER Table 21

5.2.4 ATTENDEE\_STATUS table 21

5.2.5 ATTENDEE Table 22

5.2.6 COURSE Table 22

5.2.7 RFIDREADER Table 22

5.2.8 RFIDREADER\_APPOINTMENT Table 22

5.2.9 ADDRESS Table 22

6 UML 23

6.1 Use Case Diagram 23

6.2 Use Case Documents 24

6.2.1 Register 24

6.2.2 Log in 25

6.2.3 Become Attendee 26

6.2.4 Manage Profile 26

6.2.5 Create Appointment 27

6.2.6 Manage Appointment 28

6.2.7 Create Course 29

6.2.8 Manage Course 30

6.2.9 Attend Appointment 31

6.3 Sequence Diagrams 32

6.3.1 User Registration 33

6.3.2 Host Registration 34

6.3.3 User Login 35

6.3.4 Host Login 36

6.3.5 User Update Profile 37

6.3.6 Host Update Profile 38

6.4 39

6.5 39

6.6 Class Diagram and Structure Maps 39

# Introduction

Some text

## Purpose

The purpose of this document is to present a detailed description of the CheckPoint Attendance System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do and the constraints under which it must operate. This document is intended for both the stakeholders and the developers of the system.

## Scope

The Checkpoint Attendance System will be a web based attendance system capable of registering and verifying attendance to a wide range of event types. This could include students attending obligatory course events or workplaces holding compulsory meetings for employees. Often, such systems are implemented with attendances being manually recorded on paper. This approach can be both error prone and time consuming, especially when collating data and calculating individual levels of attendance for larger events.

In general, the system will allow anyone hosting an appointment or event to efficiently track attendance, removing the cost-ineffective and time-consuming task of manually tracking and managing attendance. In addition, it will allow users to easily discover and sign up for local and national events thought their web based user panel.

An extension to this is the ability to use CheckPoint as a ticket system, allowing users to pay a host of an event to be included on the attendance list and then using their unique RFID tag identifier to verify their payment and gain access to the event.

A second extension to this is the usage of CheckPoint as a safety measure in the industry, by allowing employees to register their attendance at the fire assembly point in the case of a fire or accident. This would allow the shift team leader to keep track of all the employees and make sure everyone is accounted for.

## Similar Existing Attendance Systems

There are a few similar solutions already on the market, one prominent system being “myAT”, found at <www.myattendancetracker.com>, which is a web application specifically targeting attendance tracking in schools. It features an online system that allows a teacher to create classes and add students. A teacher can request attendance reports and perform more school-specific tasks like creating grading scales and evaluation assignments.

The main drawback with “myAT” is that it requires you to log the attendance manually, marking students either present, absent or late while being logged in to the web page. If the teacher does not have access to a computer connected to the internet, or is unable to be present at the beginning of a class, the system does not perform its intended task. The CheckPoint Attendance System automatically registers the attendance using unique RFID tags, eliminating the need for manual recording. All attendance is checked and verified post-appointment, assuring all attendance is recorded even if the host is absent.

Another similar attendance system is “uAttend”, found at <www.uattend.com>. It is also a web based attendance software marketed towards small businesses, more specifically for handling working hours and breaks. The manager is able to create work schedules and add employees, which works similarly to creating repeating appointments and adding the employees as attendees. It utilizes biometric registration of attendance through fingerprint scanning, and allows employees to quickly and easily log in and out.

The requirement for physical attendance registering is more in line with the main idea behind the CheckPoint Attendance System, but the use of biometric parameters for registering attendance is controversial in terms of privacy and laws regarding its use may be subject to change. Therefore, the use of RFID tags is a safer and sustainable solution.

# System Overview

An overview of the general system structure is illustrated in *Figure 2‑1: General system structure.* This maps the different modules of the system and shows how they are connected. The “Host Package” refers to the modules that is delivered to a host managing an appointment. A more detailed breakdown of each of these modules can be found in *2.1 Module Breakdown.*

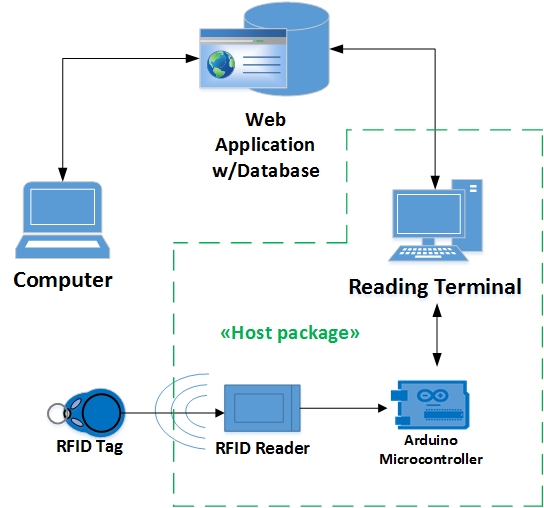


Figure 2‑1: General system structure

## Module Breakdown

This section gives a more detailed breakdown of the modules found in *Figure 2‑1: General system structure,* explaining each modules purpose and technical specifications. An overview of the data flow between the different modules can be found in *3.4 Data Communication Between Modules*.

### Computer

The computer module represents the entry point for the CheckPoint Attendance System, and is the main tool for employing the applications functionality. An active internet connection together with a web browser is required to use the application. The computer can be of any type; desktop, laptop or mac.

### Web Application w/ Database

The web application represent the main software module and the part of the system which interacts directly with the user. It hosts the user interface in the form of web forms and contains the applications business logic and DAL. For a more detailed explanation of the applications architecture, see *4. System Architecture.*

The web application and the associated SQL database resides on an external Microsoft Azure server. This assures a stable uptime, good backup security and a dynamic storage solution to meet an increasing user base.

### Reading Terminal

The reading terminal represents the module that is deployed at the appointment location and contains the software responsible for validating and logging the attendance to the database. The host managing the appointment rents the reading terminal, and it is part of the “Host Package”.

A standard terminal consists of a windows based NUC mini-pc, and has no external monitor attached. It requires an active internet to connect and validate with the database, but can also store attendance locally on file if the internet connection fails.

### Arduino Microcontroller

An Arduino microcontroller is attached to the reading terminal and is responsible for interpreting the RFID tag id registered by the RFID reader. It then transfers this id to the reading terminal. The microcontroller is also responsible for displaying the validation status to the attendee after the RFID tag has been scanned.

The microcontroller is an Arduino Uno Rev 3 and is part of the “Host Package”.

### RFID Reader

The RFID reader registers the RFID tag and transmits it to the Arduino microcontroller

The reader is of type MFRC522.

### RFID Tag

The tag is issued to each newly registered user and is used to uniquely identify the attendee at an appointment. The attendee swipes his tag on the RFID reader to register his attendance.

The tag is a passive type with a frequency of 13.56MHz.

## System Flow

The flow chart seen in *Figure 2‑2: Detailed System Flow* describes the system flow in the CheckPoint Attendance System.



Figure 2‑2: Detailed System Flow

# Technical Requirements

The system’s requirements can be divided into two main groups: functional and non-functional.

## Functional Requirements

These requirements are essential to the functionality of the software and will transform into use cases, which are handled in *6.1 Use Case Diagram.*

### System

The system is defined as the application software in combination with the database. The following represents the functional requirements for the system:

* The system must be a web-based application.
* The system must support being accessed by multiple simultaneous users without disruption.
* The system must be able to run on Google Chrome and Safari web browsers.

### Client

A client is defined as an unregistered person and does not yet have an active account in the CheckPoint system. The following represents the functional requirements for a client:

* The client must be able to register and create a new account either as a user or a host.

### User

A user is defined as someone who has registered with the system, but has not applied to attend an appointment, nor been added to an appointment by a host. A user receives an RFID tag, which functions as the user’s unique identifier, after registering. The following represents the functional requirements for a user:

* Each user of the system must be uniquely identified by his or her RFID tag number.
* All data entry must be validated to avoid duplicate identification fields.
* In the event that data entry validation is unsuccessful the user must be provided with the necessary feedback about this through the webpage.
* A user must log in to the system with a username and password.
* A user must be able to log out of the system.
* A user must be able to view all non-private appointments
* A user must be able to apply to attend any non-private appointments
* A user must be able to change their login details.
* A user is required to submit a username, first name, last name and email.
* A user must have the ability to edit their details whenever necessary.

### Attendee

An attendee is defined as a user who has applied to attend an appointment and been accepted by the host of that appointment. Once the appointment(s) are finished, the attendee goes back to being a user. The following represents the functional requirements for an attendee:

* An attendee must be able to log their attendance at an appointment with his or her RFID tag.

### Host

A host is defined as someone who has registered with the system as a host, and is someone who hosts/creates appointments for attendees to attend. A host will receive a reading terminal to use at his/her appointment(s) when registering. The following represents the functional requirements for a host:

* A host must be able to create/read/update/delete a new appointment.
* A host must be able to add users to and remove attendees from appointments.
* A host must log in to the system through the website with a username and a password.
* A host is required to submit a username, first name, last name and email.
* A host must be able to log out of the system.
* A host must not have the ability to edit the attendance logs.
* A host must be able to request/ order an RFID reader.
* A host must be able to generate a customizable attendance report.

### Appointment

An appointment is defined as a single event or happening that attendees can attend. The following represents the functional requirements for an appointment:

* An appointment must be created with a name, location, date, starting time, ending time, description and the host’s username.
* An appointment must be possible to set as either public or private.
* An appointment must be able to be set as obligatory or not for an individual attendee.

### Course

A course is defined as a collection of appointments, and allows appointments to be grouped together. The following represents the functional requirements for a course:

* A course must be able to add and remove appointments.

### Reading Terminal

The reading terminal is defined as a device responsible for registering an attendee’s attendance, by reading the RFID tag and registering the attendance in the database. The following represents the functional requirements for a reading terminal:

* The reading terminal must be able to store registrations locally to a text file as backup.
* The reading terminal must be able to connect to the internet and communicate with the server.
* The reading terminal must display the validation status on a screen to the attendee.

## Non-Functional Requirements

These requirements are related to performance and convenience, and are equally subject to testing. They can at a later date become functional requirements.

* The reading terminal should be responsive enough to process and register attendance within 4 seconds.
* The system should have an uptime of 22 hours per day.
* The GUI should not require the user to have any prior technical experience in order to navigate the system.
* The database storage should be easily scalable to meet future demands of a dynamic user base.

## Graphical User Interface Specification

The GUI will be a webpage accessed through a browser and is the primary point for users and hosts to interact with the CheckPoint System. All available functions should be self-explanatory and made simple to navigate. Included is a set of first drafts for the GUI, found in *Figure 3‑1: Draft of home page*, *Figure 3‑2: Draft of login* pageand *Figure 3‑3: Draft of user panel*

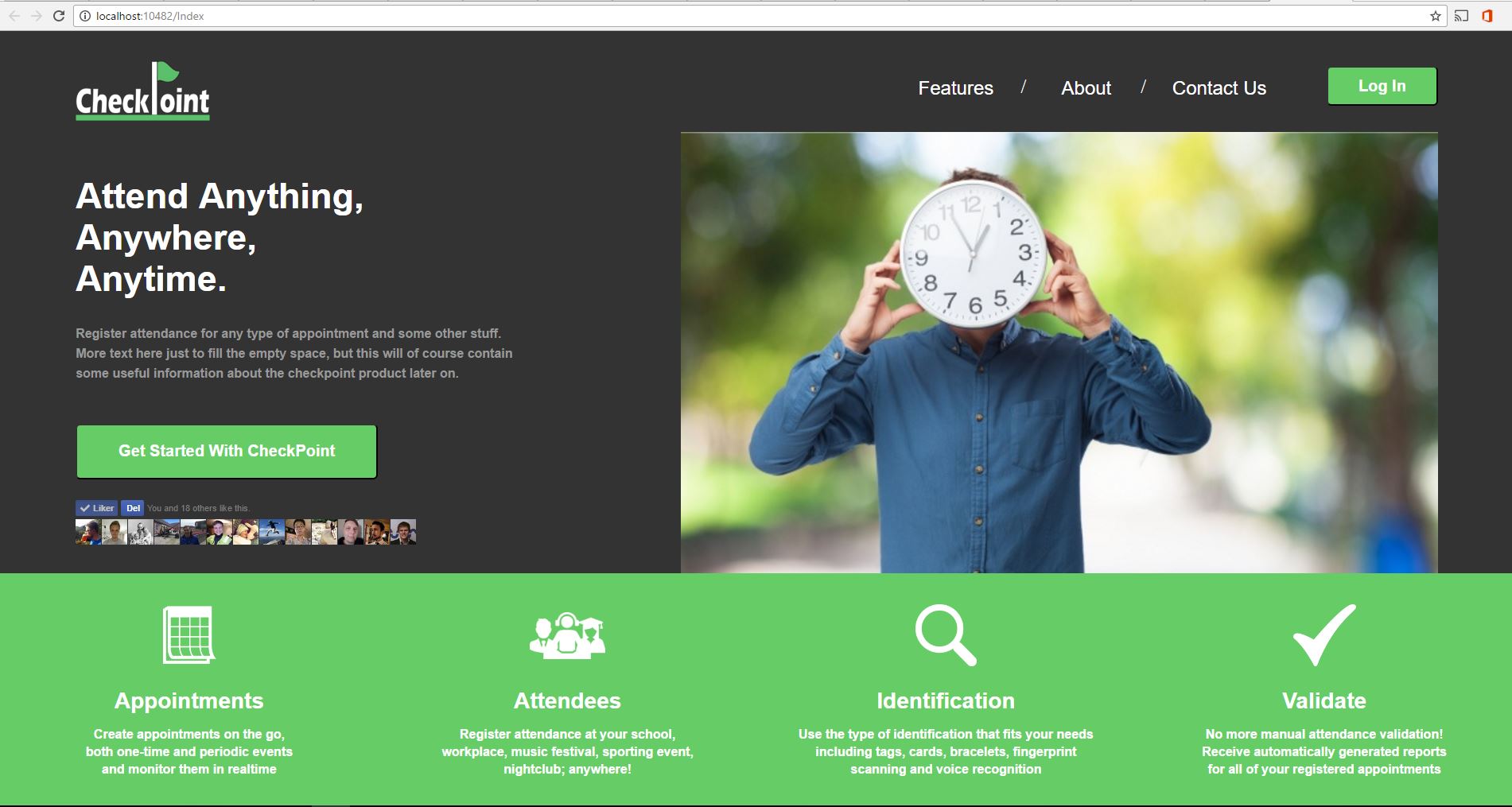


Figure 3‑1: Draft of home page

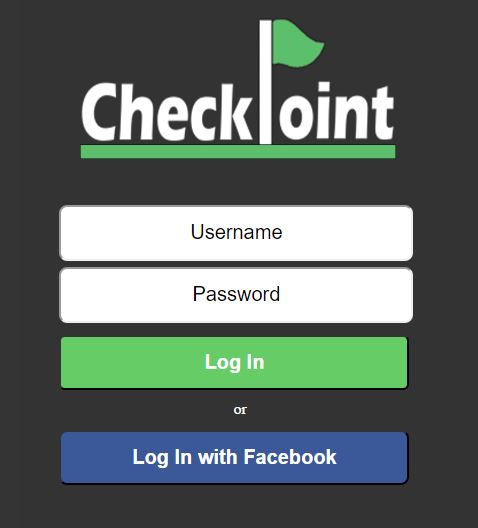


Figure 3‑2: Draft of login page



Figure 3‑3: Draft of user panel

## Data Communication Between Modules

The data communication flow between the different modules in the system is illustrated in *Figure 3‑4: Communication between modules.*

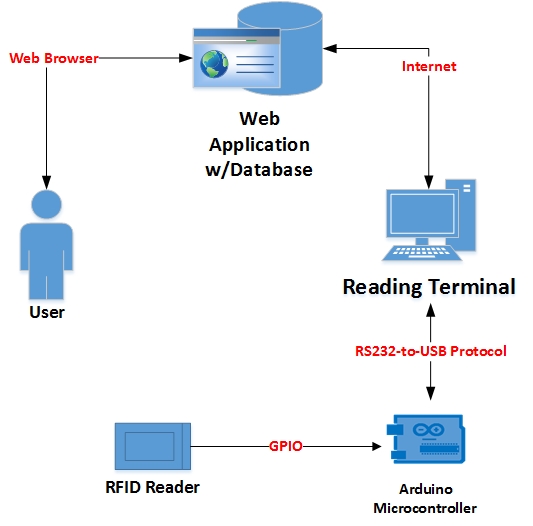


Figure 3‑4: Communication between modules

The main communication entry point for the user will be with the web application through a web browser. The reading terminal will communicate with the database server through an internet connection, allowing it to insert and extract the necessary data. The Arduino microcontroller communicates with the reading terminal though a RS232-to-USB cable and a custom transfer protocol made specifically for this application. Finally, the RFID reader communicates with the Arduino microcontroller through the microcontrollers GPIO pins and associated protocol.

# System Architecture

The following chapter will give a brief explanation of the general system architecture of the CheckPoint Attendance System.

## Three-Tier Multi-layer Structure

The CheckPoint system will adhere to a 3-tier structure. This involves different parts of the application running as separate entities either physically or logically in different locations. In the case of the CheckPoint system, this will mean having a separate database server, web application server and browser. The primary aim of this structure is to provide flexibility and easier maintenance since each layer may be modified without it being necessary to make changes throughout the entire application.

Moreover, a layered structure provides an easier pathway to scalability and performance since it is possible to dedicate extra resources only where they are needed instead of globally. If more servers were required to handle increasing web requests this could be achieved without adding extra servers to the database or web-server application layers. *Figure 4‑1: 3-Tier System Diagram* below shows the basic layout of a 3-tier system.



Figure ‑: 3-Tier System Diagram

### Presentation Layer

The top tier is often referred to as the presentation layer. This layer is the point of interaction between the system and the client and includes the UI of the application. The main role of the presentation layer is to provide a convenient user-friendly way to use the application and access the functionality that the system affords. The presentation layer of the Checkpoint system will primarily consist of an ASP.NET Webform application. This will be delivered to a browser such as Google Chrome or Mozilla Firefox running on a client’s individual desktop PC or laptop. Additionally, the Checkpoint system will includes an RFID reading terminal comprising a NUC, Arduino-UNO and an RFID scanner. This part of the system will be a point of interaction at the location of all appointments. The terminals will facilitate scanning the attendees RFID tags and will provide feedback on status of the scanning procedure.

### Business Layer

The business layer is responsible for the main logic of the application. Data handling, calculations, processing requests and co-ordination of the application are all tasks performed by the business layer. It is effectively the brain of the system translating the user requests into actions and defining what operations can and cannot be performed on the data stored in the system. The CheckPoint business layer will be a C# .NET program capable of interacting with the ASP.NET Webform presentation layer and the database in the data access layer.

### Data Access Layer

The Data Access Layer describes the part of the application that communicates with the persistent storage location, usually a database server. It contains the services capable of collecting and transferring data between the storage facility and delivering them to the business layer. All interaction with the database should take place in this layer such that the other layers are unaware of the type of database or storage system being used. SQL Management Server will be the data access layer of the CheckPoint system. *Figure 4‑2: CheckPoint Architecture Overview* beneath shows a basic graphical overview of the CheckPoint system architecture.



Figure ‑: CheckPoint Architecture Overview

# Database

The database in the CheckPoint Attendance System is a Microsoft Azure SQL database and is an intricate part of the functionality of the software. This section explains the database structure and its tables and attributes. The database was modelled using the ERwin database-modelling tool.

## Logical and Physical Data Model

The database can be represented in two states; the abstract logical state and the physical “as implemented” state. These can both be seen respectively *in Figure 5‑1: Database logical data model* and *Figure 5‑2: Database physical data model*. A detailed explanation of the tables and attributes can be found in *5.2 Database Description*.

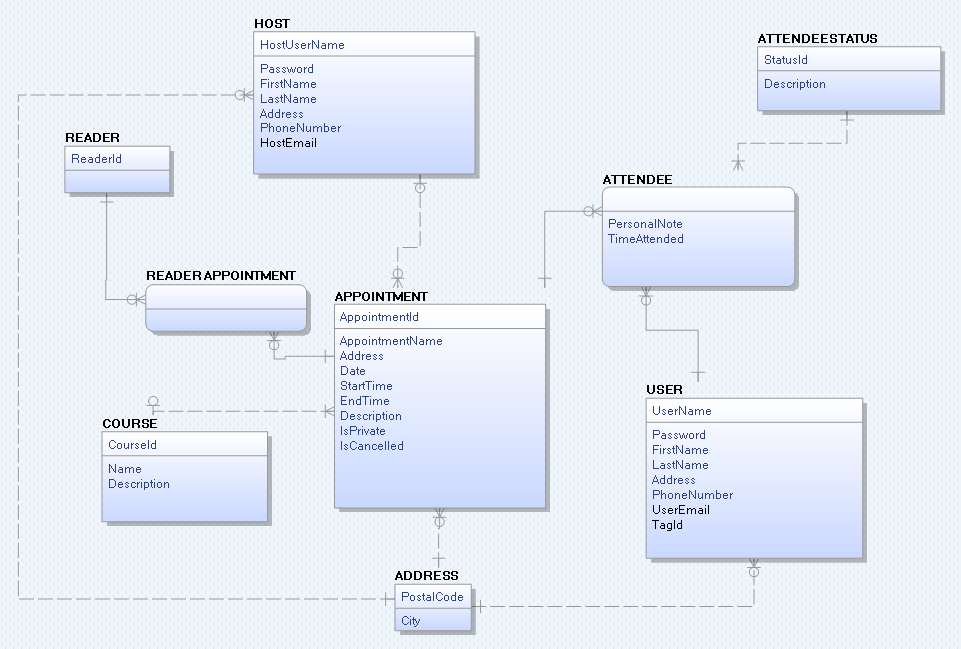


Figure ‑: Database logical data model

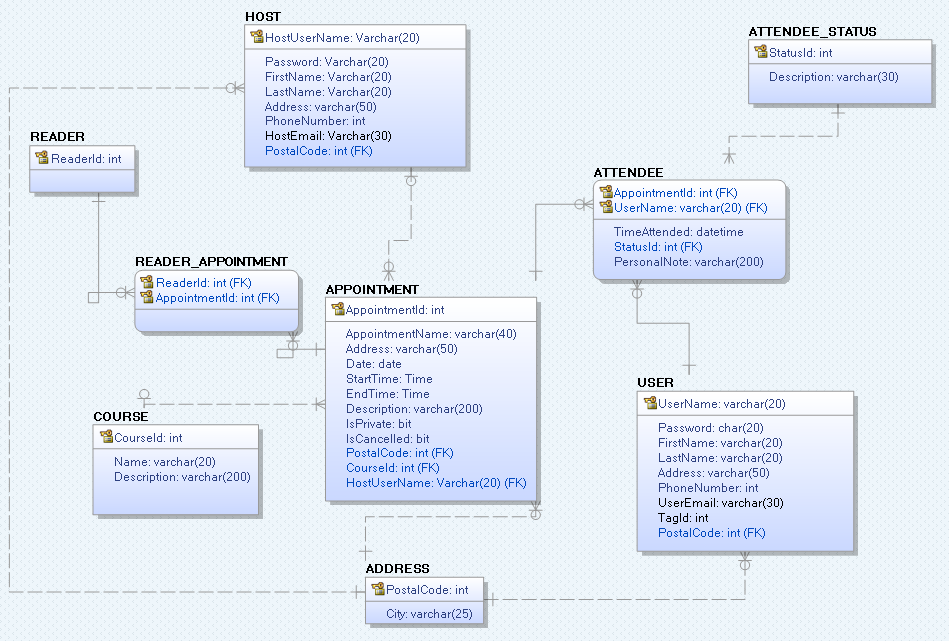


Figure 5‑2: Database physical data model

## Database Description

This gives a more detailed breakdown of the database structure by explaining the tables and attributes, as well as their relationships, keys and constraints found in the physical data model illustrated in *Figure 5‑2: Database physical data model.* Primary keys are marked with an underscore and foreign keys are marked with an asterisk, as per standard notation.

### APPOINTMENT Table

The appointment table handles all data related to an appointment..

* **AppointmentId** identifies an appointment and is automatically incremented for each newly created appointment.
* **AppointmentName** allows the host to give the appointment a name.
* **Address** allows the host to add a location for the appointment.
* **Date** holds the appointment date.
* **StartTime** holds the time when the appointment starts.
* **EndTime** holds the time when the appointment ends.
* **Description** allows the host to add additional information about his appointment.
* **IsPrivate** is used to check whether the host has made the appointment public or private.
* **IsCancelled** is used to check whether the appointment is active or has been completed/cancelled.
* **CourseId\*** references the COURSE table.
* **HostUserName\*** references the HOST table.
* **PostalCode\*** references the ADDRESS table.

### HOST Table

This table holds the data associated with a host.

* **HostUserName** identifies a host and is also used by the host to log into his/her account.
* **Password** holds the host’s password for accessing his/her account.
* **FirstName** holds the host’s first name.
* **LastName** holds the host’s last name.
* **Address** holds the host’s address.
* **PhoneNumber** holds the host’s phone number.
* **HostEMail** holds the host’s email address.
* **PostalCode\*** references the ADDRESS table.

### USER Table

This table holds the data associated with a user.

* **UserName** identifies a user and is also used to by the user to log into his/her account.
* **Password** holds the user’s password for accessing his/her account.
* **FirstName** holds the user’s first name.
* **LastName** holds the user’s last name.
* **Address** holds the user’s address. This is used to determine where to send the RFID tag when a new user is registered.
* **PhoneNumber** holds the user’s phone number.
* **UserEMail** holds the user’s email address.
* **TagId** holds the id of the RFID tag associated with the user.
* **PostalCode\*** references the ADDRESS table.

### ATTENDEE\_STATUS table

This table holds a user/attendee’s current status and keeps track his/her current status.

* **StatusId** dictates the user/attendee’s current status with integer values which are mapped in the following order:
  + **0** : User has requested to attend an appointment and is awaiting approval by the host of the appointment.
  + **1**: The user’s request has been approved by the host, and the user is now an attendee. **The appointment has been marked by the host as NOT obligatory.**
  + **2:** The user’s request has been approved by the host, and the user is now an attendee. **The appointment has been marked by the host as obligatory.**
  + **3**: The user has attended the **NON-OBLIGATORY** appointment.
  + **4**: The user has attended the **OBLIGATORY** appointment.
* **Description** holds information regarding the different states.

### ATTENDEE Table

This junction table holds the data associated with an attendee.

* **AppointmentId\*** references the APPOINTMENT table.
* **UserName\*** references the USER table.
* **TimeAttended** holds the information on when the attendee registered his attendance at the appointment.
* **PersonalNote** allows the attendee to append notes to the appointments he/she is attending.
* **StatusId\*** references the ATTENDEE\_STATUS table.

### COURSE Table

This table allows the host to establish courses associated with appointments, which allows for grouping of appointments.

* **CourseId** identifies the specific course.
* **Name** allows the host to give the course a name.
* **Description** allows the host to add a description to the course.

### RFIDREADER Table

This table holds reference to all the reading terminals.

* **ReaderId** identifies a specific reading terminal.

### RFIDREADER\_APPOINTMENT Table

This junction table allows more than one reader to be associated with a single appointment and keeps track of which reading terminal is deployed to which appointment.

* **ReaderId\*** references the RFIDREADER table.
* **AppointmentId\*** references the APPOINTMENT table.

### ADDRESS Table

This table facilitates normalization since a city can be associated with several postal codes.

* **PostalCode** holds all the postal codes registered in Norway.
* **City** holds the city associated with the specific postal code.

# UML

This section contains the UML documentation used in the design and implementation of the software. It provides both an overview of the general functionality of the software in the use case diagram, but also provides a greater level of detail in the use case documents and class diagrams. Because of the agile nature of the development model through the use of SCRUM, these documents are dynamic and susceptible to change due to changing functionality requirements.

## Use Case Diagram

The use case diagram represents an overview of the main functionality of the CheckPoint Attendance System found in *3.1 Functional Requirements* and can be viewed in *Figure 6‑1: Main use case diagram for CheckPoint Attendance System*.

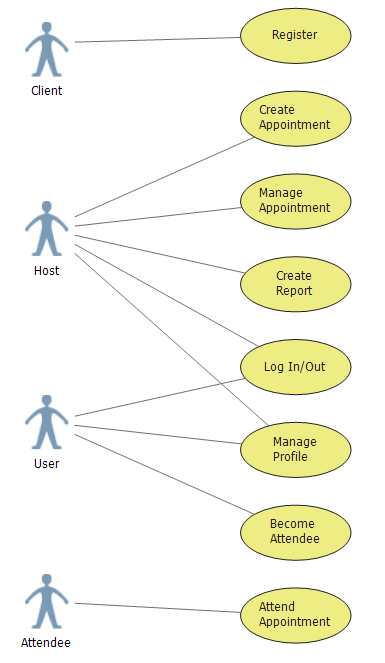


Figure 6‑1: Main use case diagram for CheckPoint Attendance System

## Use Case Documents

The use case documents provide a more detailed explanation of the different use cases found in *6.1 Use Case Diagram*. The use case documents follow the template shown in REF.

### Register

**Scope:** CheckPoint Attendance System; Web application module

**Goal:** Register a new user or host and create an associated account.

**Brief:** A client accesses the CheckPoint Attendance System web page through a browser and wants to become a user or host by creating an account. He enters his account information and becomes a user or host. He is then redirected to his personal homepage.

**Primary actor:** Client

**Stakeholders and Interests:**

Client - wants to become either a registered user or host and gain access to the system’s functionality.

**Preconditions:** CheckPoint Attendance System webpage is loaded and showing in browser.

**Success Guarantees:** The client successfully registers and creates a user or host account, and is redirected to his homepage.

**Main success scenario:**

1. The client clicks the button for registration on the homepage and is redirected to the registration page.
2. The system prompts the client for new account information:

Username

Password

First name

Last name

Email

Phone number

Address

Postcode

1. Client enters new account information
2. System validates new account information
3. The new user/host is created and populated with the account information.
4. The user/host is added to the database.
5. The new user/host is logged in.
6. The new user/host is redirected to his homepage.

**Extensions:**

4.a Username already exists

1. System displays an appropriate error message.
2. The use case continues at step 2.

4.b Client details are in the wrong format

1. System displays an appropriate error message.
2. The use case continues at step 2.

6.a Unable to connect to the database

1. System displays an appropriate error message,
2. The use case continuous at step 2.

**Frequency of occurrence:** Everytime a client attempts to register a new account.

**Miscellaneous:**

### Log in

**Scope:** CheckPoint Attendance System; Web application module

**Goal:** The user or host logs in to his personal homepage.

**Brief:** A user or host accesses the login web page through a browser, enters his username and password and is redirected to his personal home page.

**Primary actor:** User, Host

**Stakeholders and Interests:**

User – Wants to gain access to his homepage in order to view his current appointments, look for new appointments, apply to appointments and change his personal details.

Host – Wants to gain access to his homepage in order to view attendance for the appointments he is hosting, make new appointments, accept applied users as attendees and change his personal details.

**Preconditions:** Login webpage is loaded in a browser.

**Success Guarantees:** The client successfully logs in and is redirected to his/her personal homepage.

**Main success scenario:**

1. The client clicks the button for logging in on the homepage and is redirected to the log in page.
2. The system prompts the client for account information:

Username

Password

1. Client enters account information
2. System validates account information
3. The user/host is logged in.
4. The user/host is redirected to his homepage.

**Extensions:**

4.a Account does not exist

1. System displays an appropriate error message.
2. The use case continues at step 2.

4.b Account information in the wrong format

1. System displays an appropriate error message.
2. The use case continues at step 2.

**Frequency of occurrence:** Everytime a user or host attempts to log in.

**Miscellaneous:**

### Become Attendee

**Scope:** CheckPoint Attendance System; Web application module

**Goal:** The user is allowed to attend an appointment.

**Brief:** A user searches for appointments on his homepage and finds an appointment he would like to attend. He marks the appointment and requests to be an attendee for it. The host receives the request and decides if he wants to allow the user to attend his appointment. If he accepts the request, the user is now an attendee for that appointment and is allowed to attend.

**Primary actor:** User

**Stakeholders and Interests:**

User – Wants to attend one or several appointments.

Host – Wants to have attendees for his appointment(s).

**Preconditions:** User is logged in and on the “Find appointments” user tab on his homepage.

**Success Guarantees:** The user gets his request for attendance approved by the host of the appointment and becomes and attendee.

**Main success scenario:**

1. User clicks the “Search for appointments” button.
2. System displays available appointments.
3. User clicks on the appointment he wishes to request attendance for.
4. Appointments details are shown in a panel below the gridview.
5. User clicks the button for requesting attendance.
6. Request is registered and user is awaiting approval by host.
7. Host receives the request on his “My Appointments” tab and approves
8. User receives confirmation of approval on his “My Appointments” tab and becomes an attendee for the appointment.

**Extensions:**

7.a. The host does not approve the request

1. User status remains “applying to attend”
2. Appointment ends.
3. Appointment gets archived in user’s “My Appointments” user tab

**Frequency of occurrence:** Everytime a user requests to attend an appointment.

**Miscellaneous:**

### Manage Profile

**Scope:** CheckPoint Attendance System; Web application module

**Goal:** The user or host wants to change the personal information related to his account.

**Brief:** A user or host wants to edit his personal details and accesses his

**Primary actor:** User, Host

**Stakeholders and Interests:**

User – Wants to edit/update the personal details linked to his account.

Host – Wants to edit/update the personal details linked to his account.

**Preconditions:** User/host is logged in and on the “Personal Details” tab.

**Success Guarantees:** The new user details are saved and displayed on the “Personal Details” tab.

**Main success scenario:**

1. The system prompts the user/host to change account information, with the current information displayed in the respective field:

First name

Last name

Password

Email

Phone number

Address

Postal code

1. User/host changes the desired fields.
2. User/host clicks button to save changes.
3. System validates new account information
4. User/host is redirected to his personal homepage.

**Extensions:**

2.a. User/host leaves “Personal Details” tab before saving changes

1. The old account information is kept and no changes are made
2. User/host redirected to other page.

4.a. New account information is in incorrect format

1. System displays an appropriate error message
2. The field(s) with incorrect format is highlighted.
3. The use case continues at step 1.

**Frequency of occurrence:** Everytime a user or host opens the “Personal Details” tab.

**Miscellaneous:**

### Create Appointment

**Scope:** CheckPoint Attendance System; Web application module

**Goal:** Create a new appointment as a host.

**Brief:** A host fills in the required details for the appointment and creates it. He can choose to have it publicly available to all users or private.

**Primary actor:** Host

**Stakeholders and Interests:**

Host – Wants to create a new appointment.

**Preconditions:** Host is logged in and on the “Create Appointment” tab.

**Success Guarantees:** A new appointment is created.

**Main success scenario:**

1. The system prompts the host to enter new appointment information:

Appointment name

Address

Date

Start time

End time

Description

Postal Code

Course Id

Availability

1. Host enters new appointment information
2. Host clicks button to create appointment.
3. System validates new appointment information
4. New appointment is created and saved.
5. Host is redirected to “My Appointments” tab.

**Extensions:**

2.a. Host leaves “Create Appointment” tab before creating appointment

1. The fields are cleared.
2. User/host redirected to other page.

4.b. Course Id does not exist.

1. System enquires host if he wants to create a new course with the provided course id.
   1. Host wishes to create a new course
      1. A new course is created with the provided Course Id field value and the appointment is added to the course.
   2. Host does not wish to create a new course
      1. Course Id field remains empty.
2. Use case continues at step 3.

**Frequency of occurrence:** Everytime a host loads the “Create Appointment” tab.

**Miscellaneous:**

### Manage Appointment

**Scope:** CheckPoint Attendance System; Web application module

**Goal:** Update/edit an appointment’s information.

**Brief:** A host chooses one of his created appointments and updates/edits its information. The changes are saved and the host and the appointment’s attendees can see the new information on the “My Appointments” tab.

**Primary actor:** Host

**Stakeholders and Interests:**

Host – Wants to edit/update the personal details linked to his account.

Attendee – Wants to know what information has been changed about the original appointment.

**Preconditions:** Host is logged in and on the “My Appointments” tab.

**Success Guarantees:** The new appointment information is saved and displayed to the host and all it’s attendees.

**Main success scenario:**

1. Host clicks on the appointment he wants to update/edit from the gridview.
2. Appointment information is shown below the gridview.
3. Host clicks the “Edit appointment” button.
4. The system prompts the host to change appointment information, with the current information displayed in the respective fields:

Appointment name

Address

Date

Start time

End time

Description

Postal Code

Course Id

Availability

1. Host updates the desired fields.
2. Host clicks button to save changes.
3. System validates new account information
4. Appointments’ information is saved to the database.
5. Host is redirected to his personal homepage.

**Extensions:**

5.a. Host leaves “My Appointments” tab before saving changes

1. The old appointment information is kept and no changes are made
2. User/host redirected to other page.

7.a. New appointment information is in incorrect format

1. System displays an appropriate error message
2. The field(s) with incorrect format is highlighted.
3. The use case continues at step 1.

8.a. Unable to connect to the database

1. System displays appropriate error message.
2. Use case continues at step 4.

**Frequency of occurrence:** Everytime a host wants to edit an appointment.

**Miscellaneous:**

### Create Course

**Scope:** CheckPoint Attendance System; Web application module

**Goal:** Create a collection of appointments and have them in the same group.

**Brief:** A host wants to create a course which consists of several appointments on different dates and times. He creates a new course and adds the number of appointments he needs and sets each individual appointment’s information.

**Primary actor:** Host

**Stakeholders and Interests:**

Host – Wants to group his appointments to ensure that attendees are signed up for all of the appointments.

User – Wants to sign up for all the related appointments in a course, in order to make sure they

**Preconditions:** Host is logged in and on the page for creating courses.

**Success Guarantees:** The desired number of appointments are added to the course, and the course is saved to the database.

**Main success scenario:**

1. System prompts host for course information:

Course name

1. Host adds the desired number of appointments to the course.
2. Host enters new appointment information for each added appointment
3. System validates course and appointment information.
4. Host clicks the button for creating a new course.
5. New course and appointments are created and saved to the database.
6. Host is redirected to his personal homepage.

**Extensions:**

3.a. Host leaves the page for creating courses before saving changes

1. All information is cleared and added appointments are deleted.
2. User/host redirected to other page.

4.a. New course and/or appointment information is in incorrect format

1. System displays an appropriate error message
2. The field(s) with incorrect format is highlighted.
3. The use case continues at step 3.

6.a. Unable to connect to the database

1. System displays appropriate error message.
2. Use case continues at step 1.

**Frequency of occurrence:** Everytime a host wants create a course.

**Miscellaneous:**

### Manage Course

**Scope:** CheckPoint Attendance System; Web application module

**Goal:** Edit one or more appointments belonging to a course.

**Brief:** A host wants to edit/update one or more appointments group in a course. He chooses the specific appointments, updates their information and saves the changes. These changes now are visible to all attendees and users.

**Primary actor:** Host

**Stakeholders and Interests:**

Host – Wants to edit/update one or more appointments he administers in a course.

Attendee – Wants to know about changes being made to appointments he is signed up for.

**Preconditions:** Host is logged in and on the page for managing courses.

**Success Guarantees:** The desired number of appointments’ information is updated and the changes are saved. Users and attendees can view the changes that has been made.

**Main success scenario:**

1. Host chooses the appointment(s) he wishes to update.
2. The system prompts the host to change appointment information, with the current information displayed in the respective fields:

Appointment name

Address

Date

Start time

End time

Description

Postal Code

Course Id

Availability

1. Host enters new appointment information for each added appointment
2. System validates course and appointment information.
3. Host clicks the button for creating a new course.
4. New course and appointments are created and saved to the database.
5. Host is redirected to his personal homepage.

**Extensions:**

3.a. Host leaves the page for managing courses before saving changes

1. No information is updated/changed.
2. User/host redirected to other page.

4.a. Updated course and/or appointment information is in incorrect format

1. System displays an appropriate error message
2. The field(s) with incorrect format is highlighted.
3. The use case continues at step 3.

6.a. Unable to connect to the database

1. System displays appropriate error message.
2. Use case continues at step 1.

**Frequency of occurrence:** Everytime a host update/edit a course.

**Miscellaneous:**

### Attend Appointment

**Scope:** CheckPoint Attendance System; Reading Terminal module

**Goal:** Register and log an attendee’s attendance to the database.

**Brief:** An attendee swipes his RFID tag on reading terminal’s RFID reader and registers his attendance. The tag Id is transferred to the reading terminal and checked against the database. If the tag id exists in the database and is connected to the associated appointment, the attendance is logged with a timestamp.

**Primary actor:** Attendee

**Stakeholders and Interests:**

Attendee – Wants to register and log his attendance.

Host – Wants to gain an overview over who is attending his appointment.

**Preconditions:** User has received his RFID tag after registering. The user’s request to attend the appointment has been approved by the host, and the user is an attendee for the appointment.

**Success Guarantees:** The attendance has been registered and the database has been updated with the timestamp.

**Main success scenario:**

1. Reading terminal prompts attendee to swipe his RFID tag.
2. Attendee swipes his RFID tag on the reading terminal’s RFID reader.
3. Swipe is recognized by the reading terminal.
4. Attendee is found in the list of valid appointment attendees.
5. Attendee’s attendance is registered and logged to the database,
6. Reading terminal displays message to attendee confirming successful registration.

**Extensions:**

3.a. RFID reader fails to transfer tag id to the reading terminal

1. Reading terminal displays appropriate error message
2. Use case continues at step 1.

4.a. Attendee is not found in the list of valid appointment attendees.

1. Reading terminal displays appropriate error message.
2. Use case continues at step 1.

5.a Unable to connect to the database

1. Attendance information is serialized and stored locally on the reading terminal
2. Use case continues at step 6.

**Frequency of occurrence:** Everytime an attendee swipes his RFID tag to attend and appointment.

**Miscellaneous:**

### Create Report

**Scope:** CheckPoint Attendance System; Web application module

**Goal:** Create a report with information about courses, appointments and attendance.

**Brief:** A host wants an overview of the attendance records for his courses and appointments. He chooses the courses and appointments he wants to include in his report, adds the desired filtering and gets a pdf report file.

**Primary actor:** Host

**Stakeholders and Interests:**

Host – Wants a report of attendance information related to his courses and appointments.

**Preconditions:** Host is logged in and on create report page.

**Success Guarantees:** The report is created and made available for the host to download as a pdf file.

**Main success scenario:**

1. Host chooses the courses and appointments he wants to include in the report
2. The system prompts the host for report filename.
3. Host enters report filename.
4. System prompts host for filtering options.
5. Host enters filtering options.
6. Host clicks the button for creating a new report.
7. System validates report filename.
8. New report is created and made available for downloading.
9. Host downloads the report.

**Extensions:**

1.a. Host does not add any courses or appointments to include in the report

1. System displays an appropriate error message
2. Use case continues at step 1.

5.a. Host does not choose any filtering options.

1. Report is filtered with the default settings.
2. The use case continues at step 6.

7.a. Filename is blank

1. System displays appropriate error message.
2. Use case continues at step 3.

8.a. Host created report but does not download it before leaving page.

1. Created report is deleted.
2. Host is redirected to the new page.

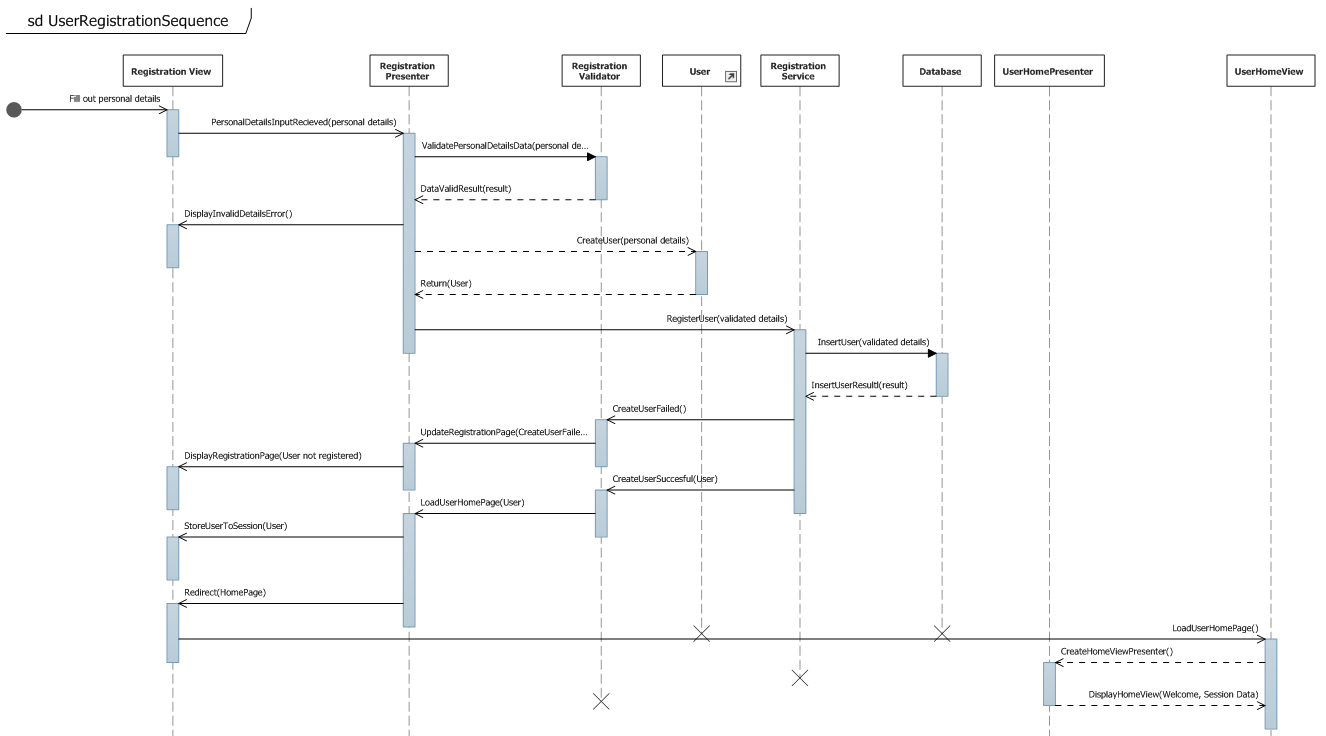
**Frequency of occurrence:** Everytime a host loads the page for creating a report.

**Miscellaneous:**

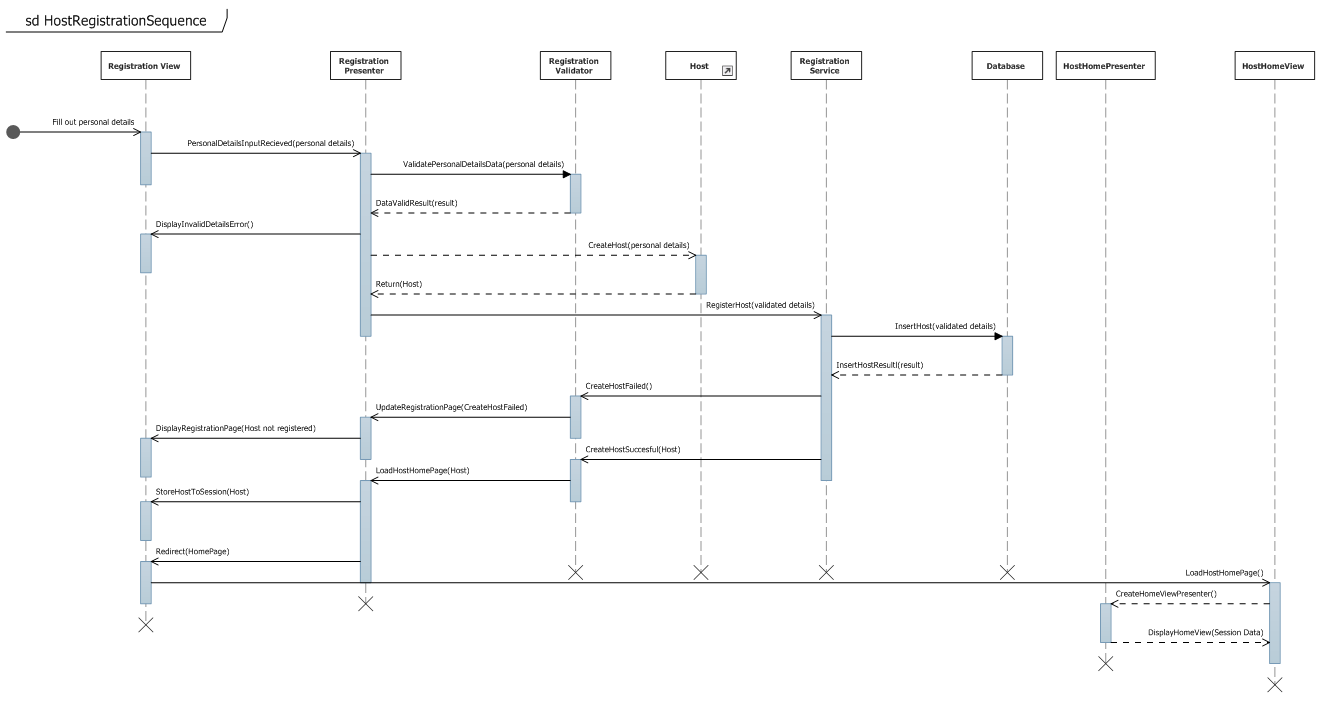
## Sequence Diagrams

The purpose of the sequence diagrams is to understand the interactions between objects in the use case specific parts of the software. This provides an informative at program flow and is a key part in the implementation.

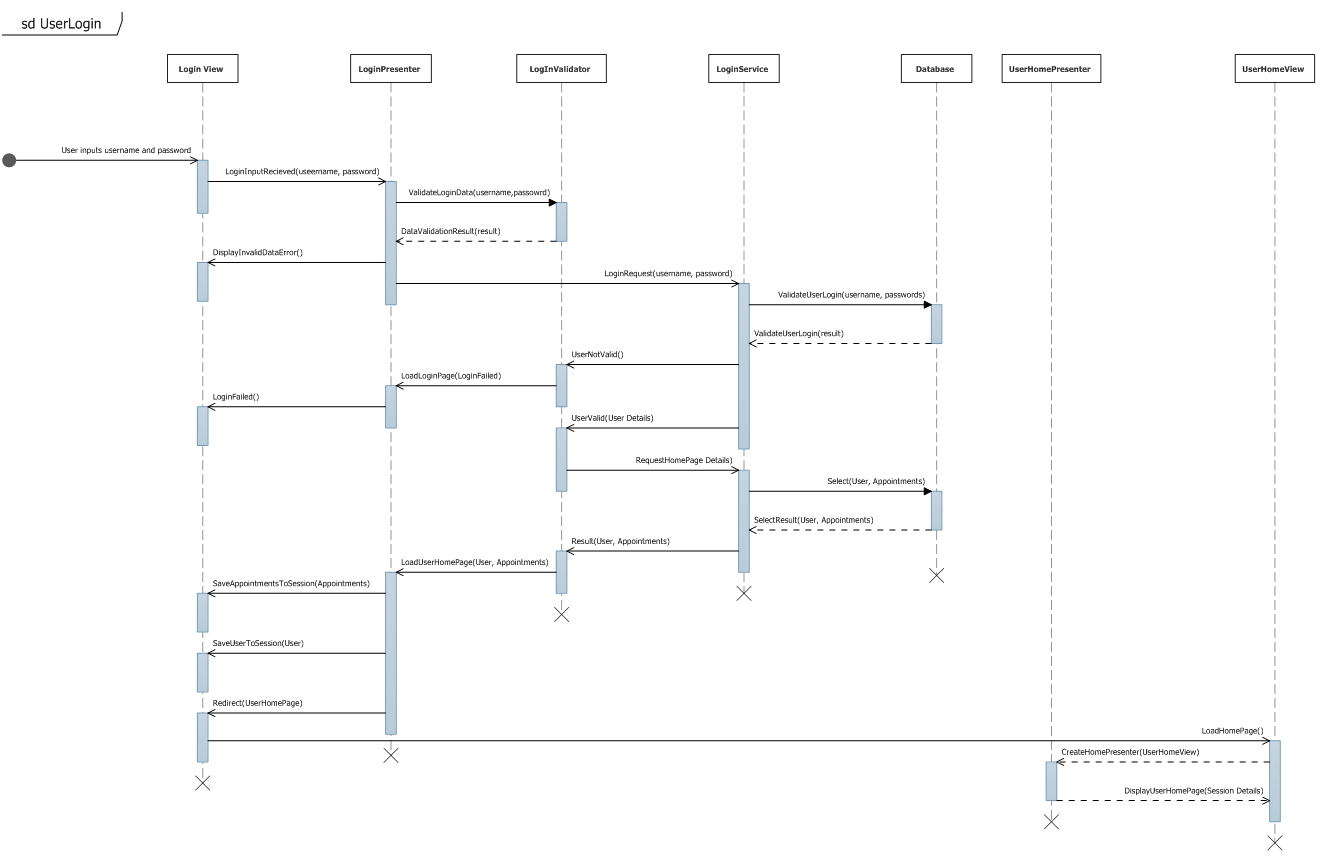
### User Registration



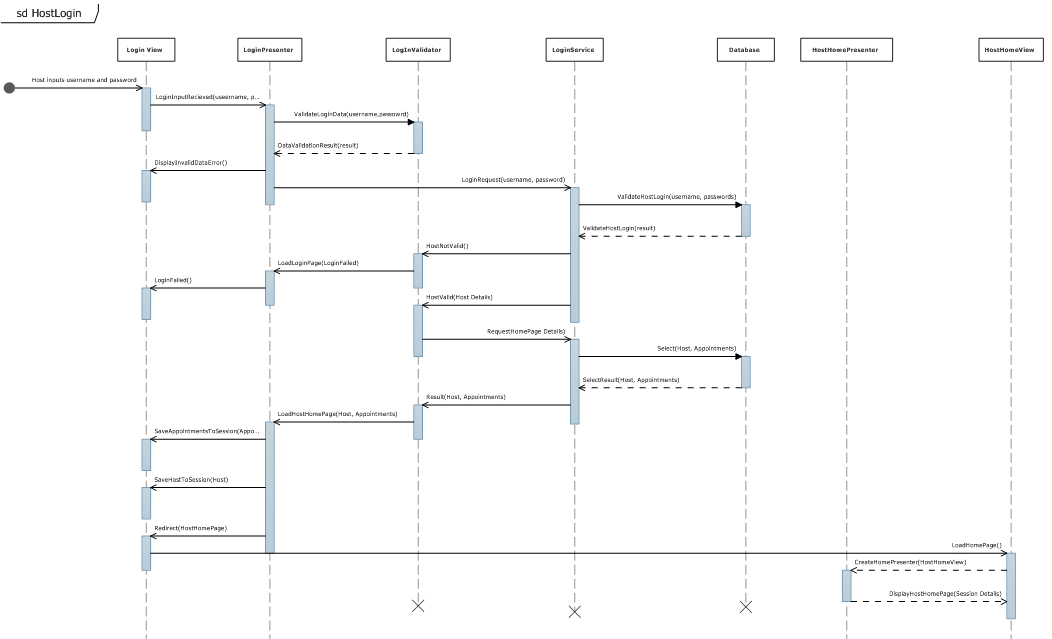
### Host Registration



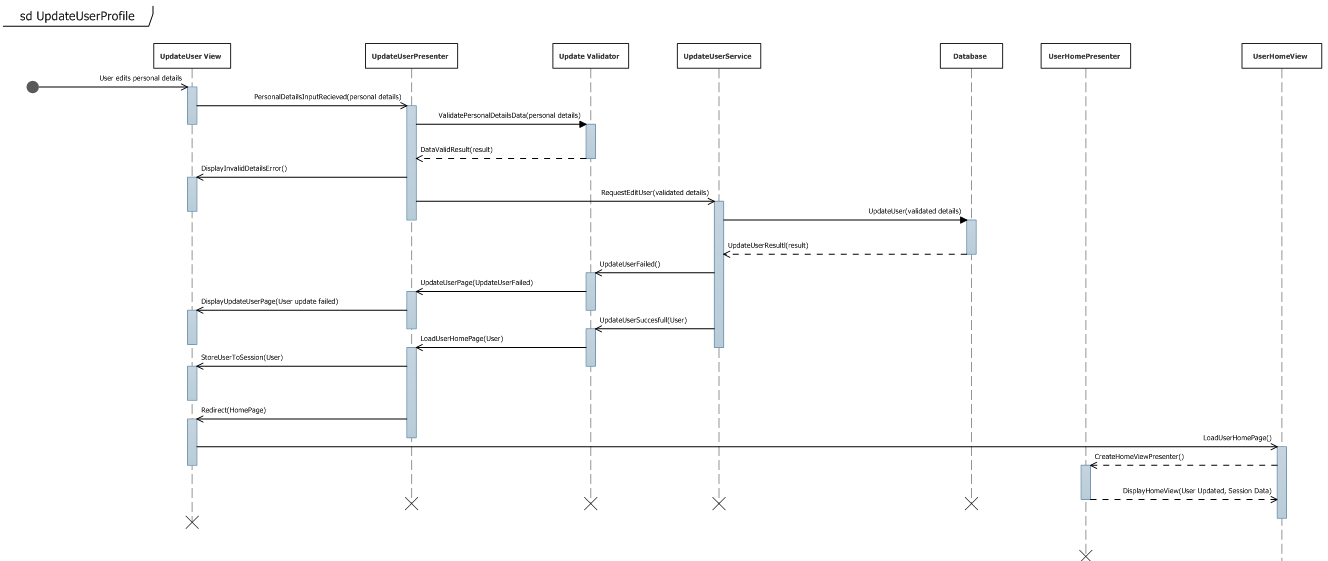
### User Login



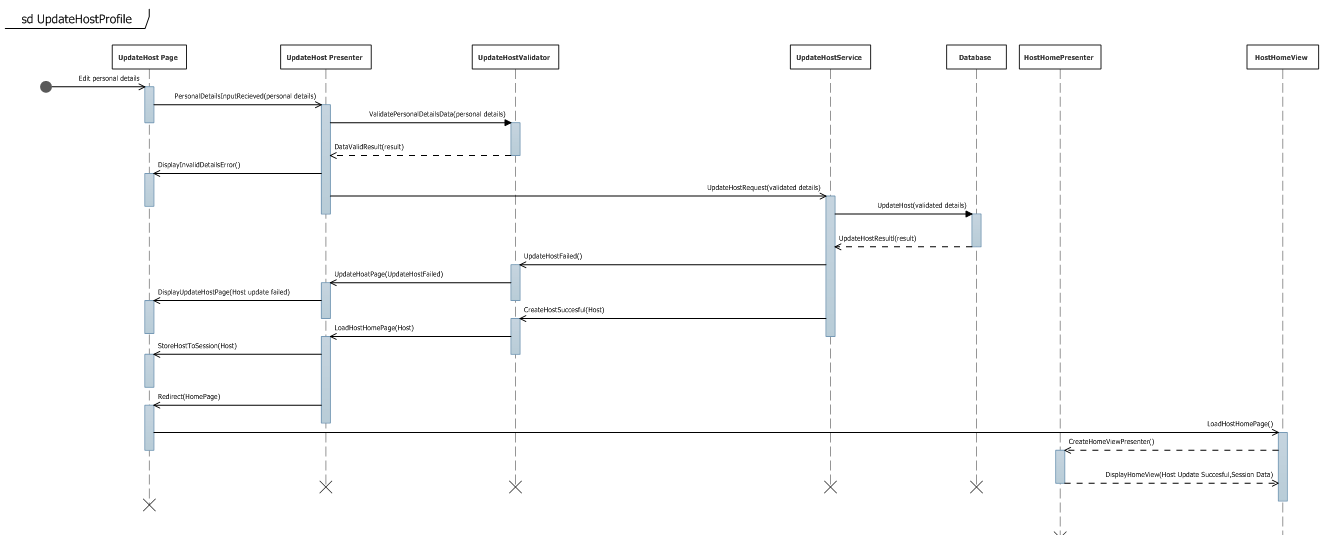
### Host Login



### User Update Profile



### Host Update Profile



## 

## 

## Class Diagram and Structure Maps