College of Science and Technology
School of Science and Technology

SOFT30121: Advanced Analysis and Design

GROUP ANALYSIS AND DESIGN REPORT

By

Group F

N0623403: Joe Zalewski

N0578685: Jack Powell

N0623631: Paul Havelin

N0645503: Alex Wells

N0626631: Sam Crane

SDSA Stroke Assessment Application

Page count: 18
(page count excludes title page, contents page and references)

Contents

Introduction

We have been given the task of the taking the Stroke Drivers Screening Assessment (SDSA) which is a series of paper-based tests designed to test if the patient has been severely effect by a stroke and if they are still capable of driving safely. The application would help address multiple issues associated with a physical copy including the lack of availability, number of parts and cost of manufacturing. These problems made large scale distribution of the assessment difficult at best however if the tests could successfully be translated into an application it would make the SDSA a more viable solution. In a previous document we outlined the functional and non-functional requirements then justified why they were necessary for the final product. This document will focus on our group's design and implementation of the SDSA application using a variety of UML diagrams to plan the elements of the program such as classes, user interaction and data flow as well as other techniques such as a refined version of the Lean UX to discuss features and create paper prototypes to show the layout of the application screens. After the programming the SDSA application we will individually critique our development process and final product to see if we met the requirements. We will also provide a acceptance test document to see if the program is up to standard and a short user guide on how to operate the application.

Rapid Lean UX

Personas

The first step to this process will be to look at the different personas that will be using the application and consider what they might need from the application, then from these needs and the requirements we can create a list of features that will be included in the application to meet the needs of the project to make our solution suitable.

Clinician	Patient
• Using the app to test patients • Needs to have unique account • Have all test resources present for ease-of-use – All SDSA tests and instructions • Store all test results for reviewing • Keep results separate based on patient id • Simple to use and explain • Fast performance on a variety of devices	• Need to be able to complete tests to best of their ability • Application should not impede their ability • May need accessibility options • Needs to be in correct language and relevant road signs

Features

Login System

The login in system will need to allow the clinician to enter their email and password for authentication which if correct, will let them access the application. They should then be able to set up the application for the patient to take the tests, this will involve them entering the patient details and select language/region options, once these have been completed the clinician needs a way to start the tests.

SDSA Tests

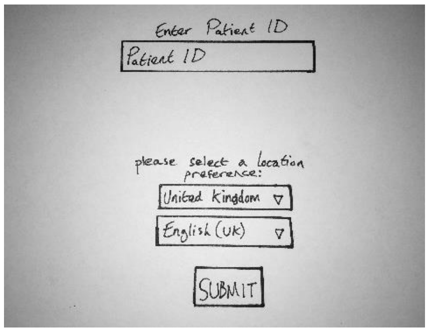
All the SDSA tests should be present in the application along with the necessary instructions which must be displayed for the clinician to read out before each assessment. All tests should be displayed clearly with an intuitive layout and simple interaction to allow the patient to take the assessment without being impeded by poor design or complex interactions. The tests should have their respective time limits enforced and the necessary results recorded to input into the final algorithm to calculate if the patient has passed or failed.

Results

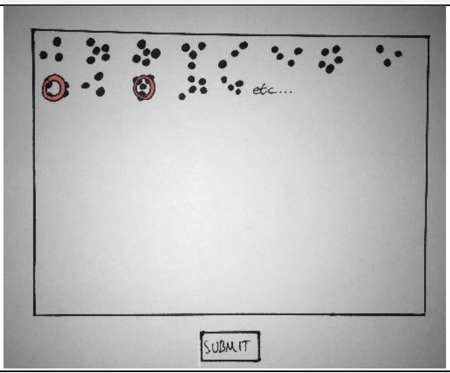
Once all the tests have been completed by the patient the application will then need to calculate the result using the SDSA algorithm which can then be viewed by the clinician; this result should also be saved to the database along with the patient details.

Paper Prototype

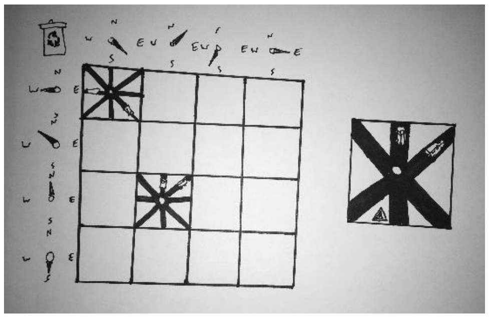
Patient Details



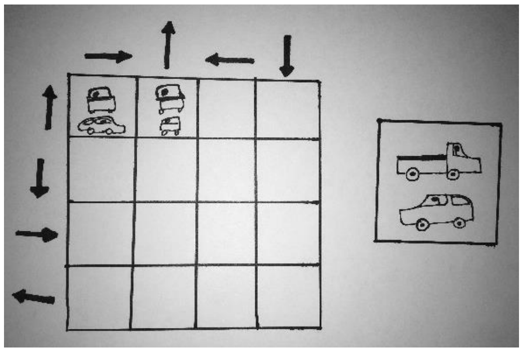
Dot Cancellation



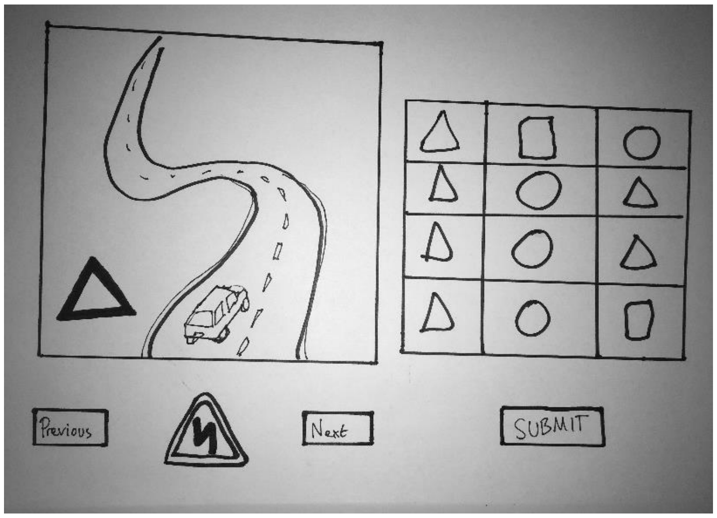
Square Matrices Compass



Square Matrices Directions



Road sign Recognition



Security

Application security is a critical aspect to consider during the development process, especially when handling user's personal data. The most common method of storage for applications is via a database, various implementations of data storage were considered, and the option chosen by the team was to use a cloud storage database. The database chosen was Firebase an online storage system that is relatively simple to access and implement. There are various issues surrounding both local databases and cloud databases. Local databases are generally more secure in the sense that they can be easily monitored by the application team, they are however more susceptible to physical attacks, i.e. damaging hardware and corrupting data. Cloud storage on the other hand due to its online nature one of the most significant concerns when using it is the hacking of data and gaining unwarranted access. Due to the nature and scale of this project opting for cloud-based storage seemed a better option, however, if scaled a manual system may want to be considered due to the nature of data being stored.

Github

As a group, it was decided that we would collaborate via an online repository using GitHub. After setting it up, the group understood that it must be as organised and professional as possible in order to simulate a real-life scenario – as if we were a team building this application for a business. The directory structure is as follows:

- doc/ which contains all documents relating to the development of the application and which will be included in the final report, for example, UML diagrams and also smaller write-ups like this one
- code/ holds the code itself – this is a self-contained folder, and the project is nicknamed the SDSAStrokeApp, where all project files are held in order to make the app
- info/ contains assignment specification along with individual submission information – this is kept separate to avoid possible confusion with doc/
- resc/ stands for 'resources' and holds collected sprites for the game which are also used by the app when building images

The issues tab has been fully utilised to be a to-do tracker and keep all team members assigned to different parts of the project, which are closed when it is completed. It also is excellent for a centralised discussion on specific topics rather than using Facebook Messenger – our choice of contact within the group.

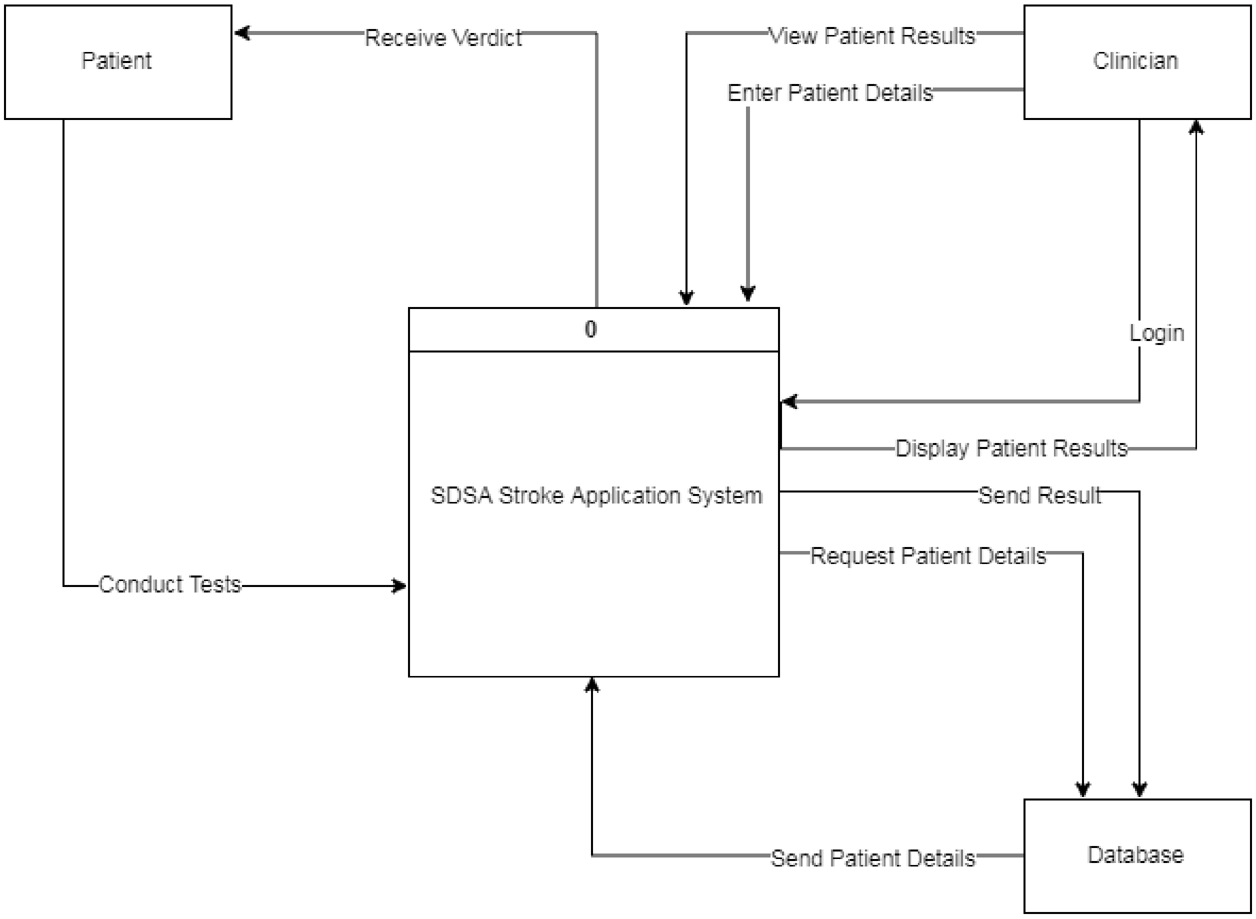
The heavy use of the repository comes from a software development module completed in previous years where heavy use of GitHub was encouraged, and it was seen how useful having a principal location for all project files was.

Data Flow Diagrams

Dataflow diagrams depict the flow of information and data in a system or for its individual processes. Constructing this diagram allows for a more user-friendly interpretation of information and interaction. Various levels of DFD's exist which can focus more specifically on a system as a whole or on individual processes.

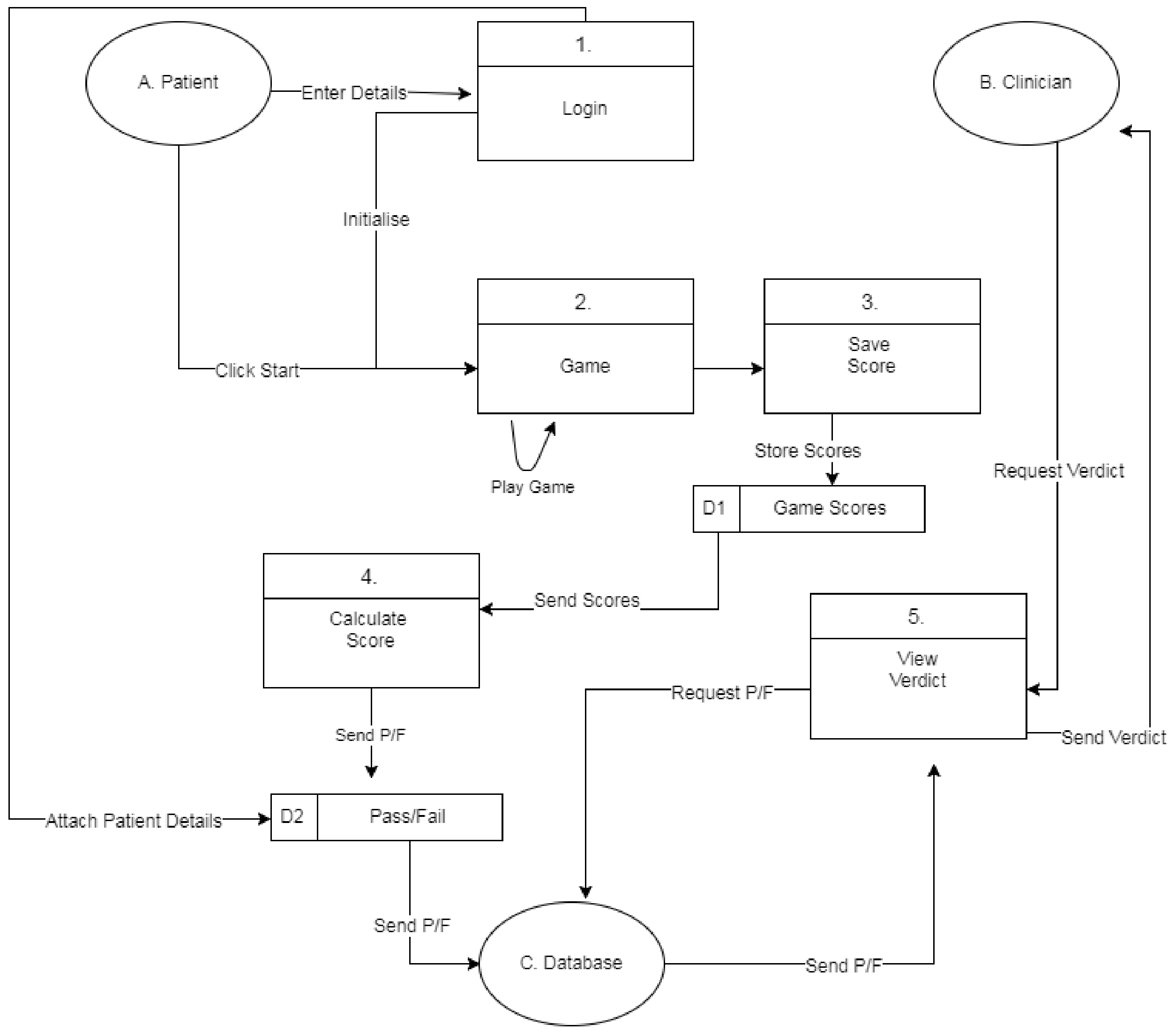
Context Level DFD

A context level DFD is a top-level diagram which contains only one process node and generally depicts the major outcomes of the system. Other levels of the diagram can progress in order to reach a narrower scope of aspects and interactions. E.g. in our system a context level may be as simple as patient/clinician/database -> app and then more specific diagrams can be deduced from this basis.



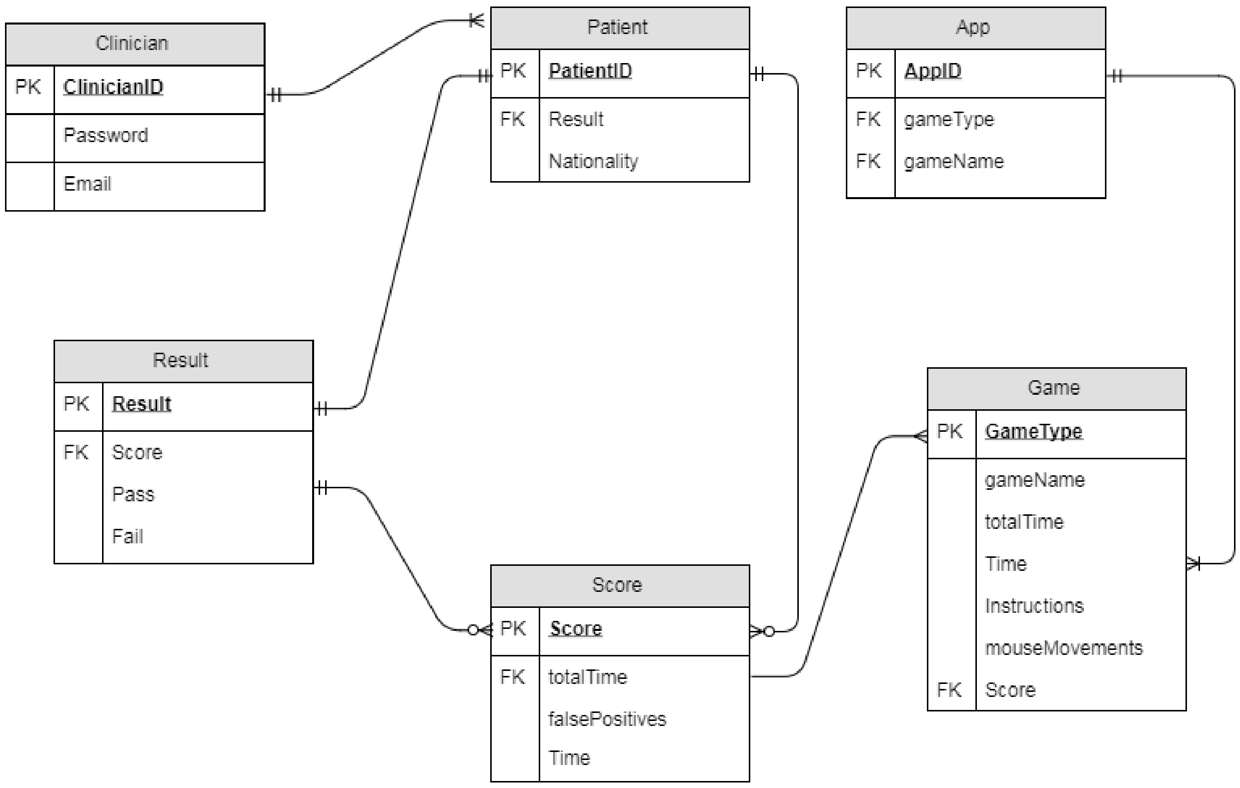
Level One DFD

After conducting dataflow analysis the group decided that only context and level 1 diagrams were required to depict the system accurately, further levelling could be conducted however was not deemed necessary. The level one DFD depicts the sub-systems and processes that operate in the system, external agents and the flows between them. Each process node must have at least one dataflow to an external agent which as shown our diagram does. The DFD constructed consisted of 3 external entities, 5 processes, 2 data stores and the directional flows around the diagram.

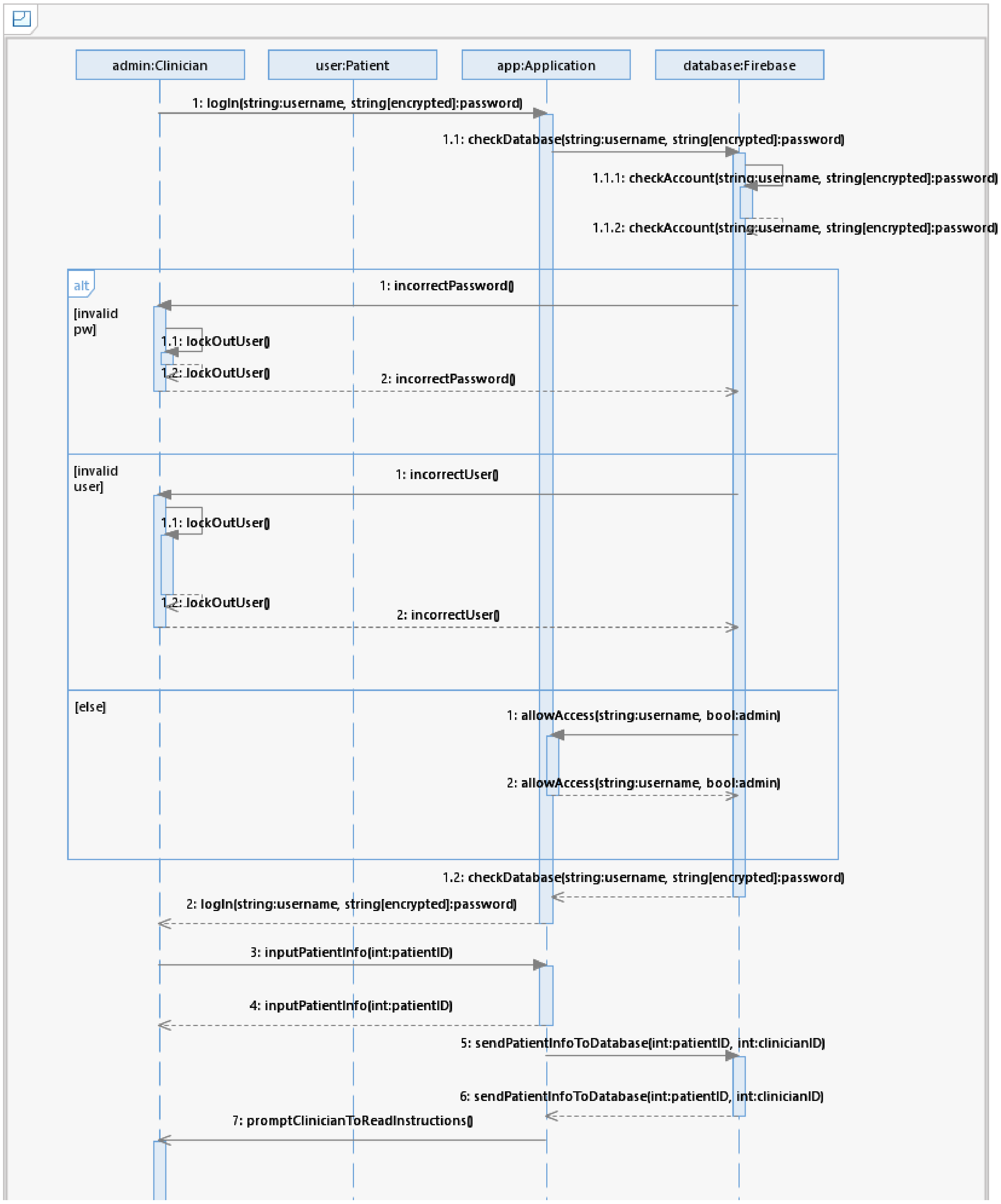


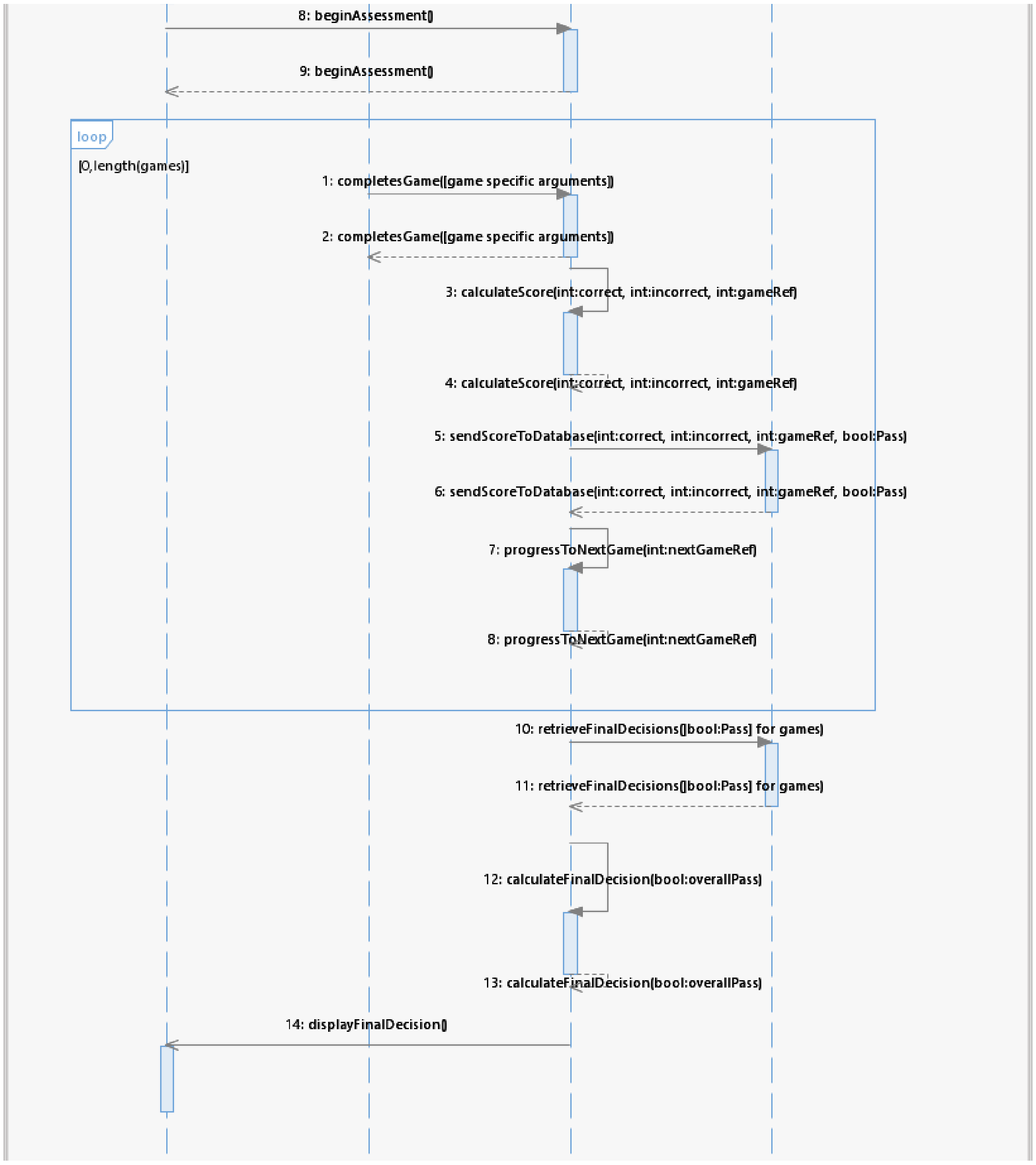
Entity Relationship Diagrams

An entity relationship diagram is a modelling technique used on systems producing a graphical illustration. This includes the entities of a system and the relationships between them; this is a useful technique as it provides a more user-friendly view of how parts of a system can interact with one another. An entity is a unit of data that can be classified; these entities contain attributes which are its properties and relationships which define how they interact with one another.
There are various types of relationships, e.g. one too many which defines one entity can interact with many others. However, the many others can only interact with one instance of that entity. The ERD for the project divided the diagram into 6 primary entities, these were physical and virtual instances that hold sources of data and require interaction between them in order to transport that data.



Sequence Diagrams

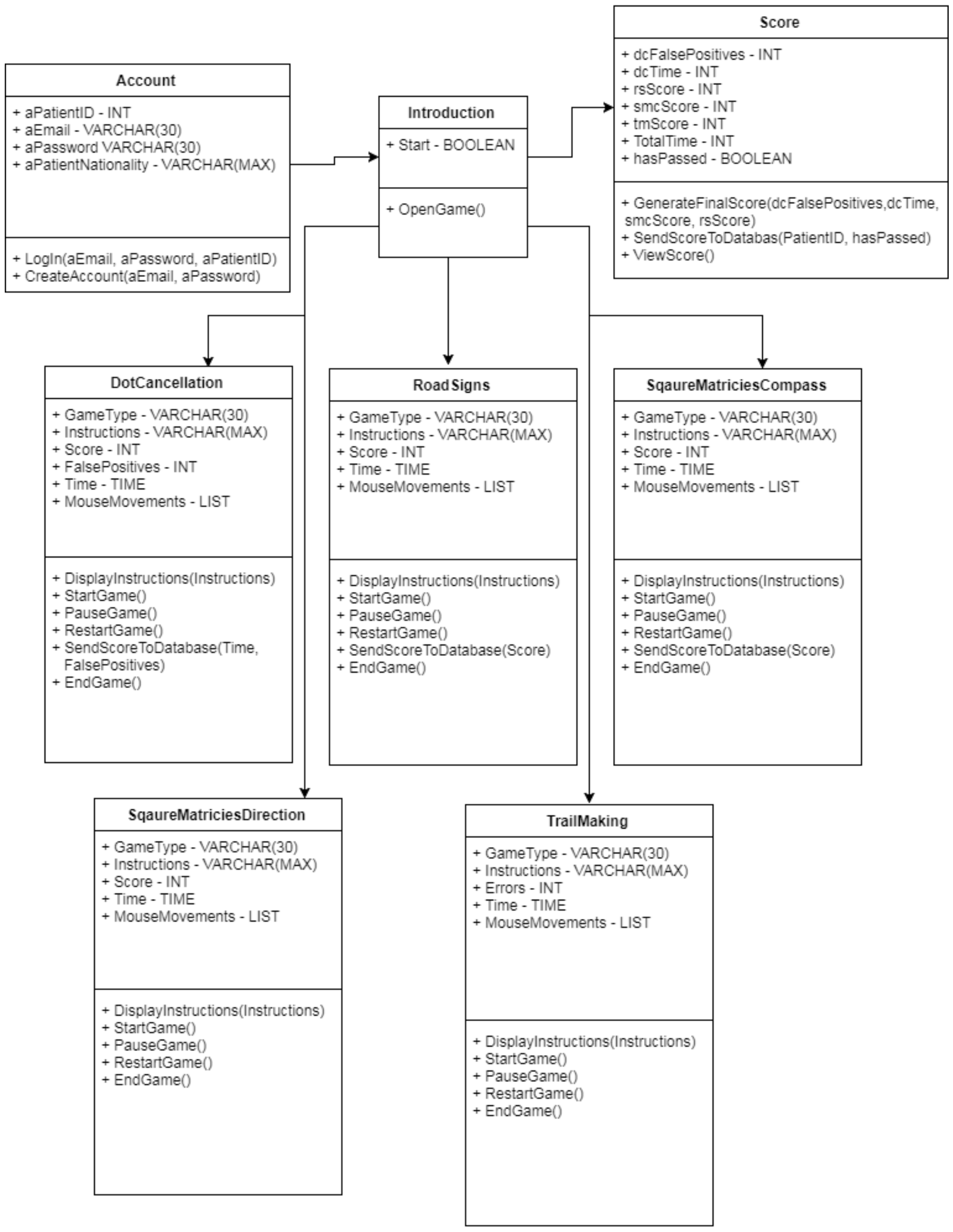

admin:Clinician
user:Patient
app:Application
database:Firebase
1: logIn(string:username, string[encrypted]:password)
1.1: checkDatabase(string:username, string[encrypted]:password)
1.1.1: checkAccount(string:username, string[encrypted]:password)
1.1.2: checkAccount(string:username, string[encrypted]:password)
alt
[invalid pw]
1: incorrectPassword()
1.1: lockOutUser()
1.2: lockOutUser()
2: incorrectPassword()
[invalid user]
1: incorrectUser()
1.1: lockOutUser()
1.2: lockOutUser()
2: incorrectUser()
[else]
1: allowAccess(string:username, bool:admin)
2: allowAccess(string:username, bool:admin)
1.2: checkDatabase(string:username, string[encrypted]:password)
2: logIn(string:username, string[encrypted]:password)
3: inputPatientInfo(int:patientID)
4: inputPatientInfo(int:patientID)
5: sendPatientInfoToDatabase(int:patientID, int:clinicianID)
6: sendPatientInfoToDatabase(int:patientID, int:clinicianID)
7: promptClinicianToReadInstructions()


8: beginAssessment()
9: beginAssessment()
loop
[0,length(games)]
1: completesGame([game specific arguments])
2: completesGame([game specific arguments])
3: calculateScore(int:correct, int:incorrect, int:gameRef)
4: calculateScore(int:correct, int:incorrect, int:gameRef)
5: sendScoreToDatabase(int:correct, int:incorrect, int:gameRef, bool:Pass)
6: sendScoreToDatabase(int:correct, int:incorrect, int:gameRef, bool:Pass)
7: progressToNextGame(int:nextGameRef)
8: progressToNextGame(int:nextGameRef)
10: retrieveFinalDecisions([bool:Pass] for games)
11: retrieveFinalDecisions([bool:Pass] for games)
12: calculateFinalDecision(bool:overallPass)
13: calculateFinalDecision(bool:overallPass)
14: displayFinalDecision()

It was essential to use a real sequence diagram simply since the lifelines in the diagram represent what will be happening simultaneously when an assessment is being completed – this was necessary to see how actors and classes alike will be interlinking and cooperating to perform the assessment.

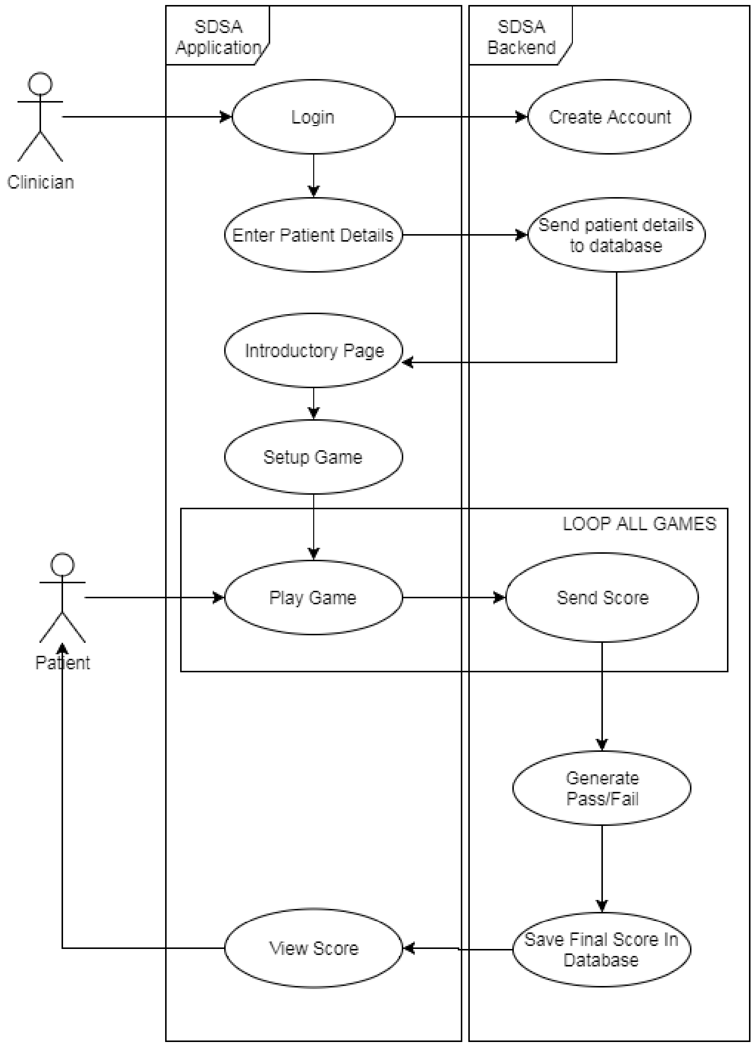
Class Diagrams

A system consists of various classes to split each area of the system. Class diagrams are used to visually represent the classes to be used and their attributes, methods and relationship with other classes within the system. Class Diagrams are displayed in UML (Unified Modelling Language) format, visually represented relationships and source code dependencies within classes (Rouse, 2018). The SDSA application contains eight class, one for each of the five cognitive tasks to retrieve information to calculate the final score, an account class for log in both clinician and patient information, a score class and an introductory class which asks as a menu to set up the tests.

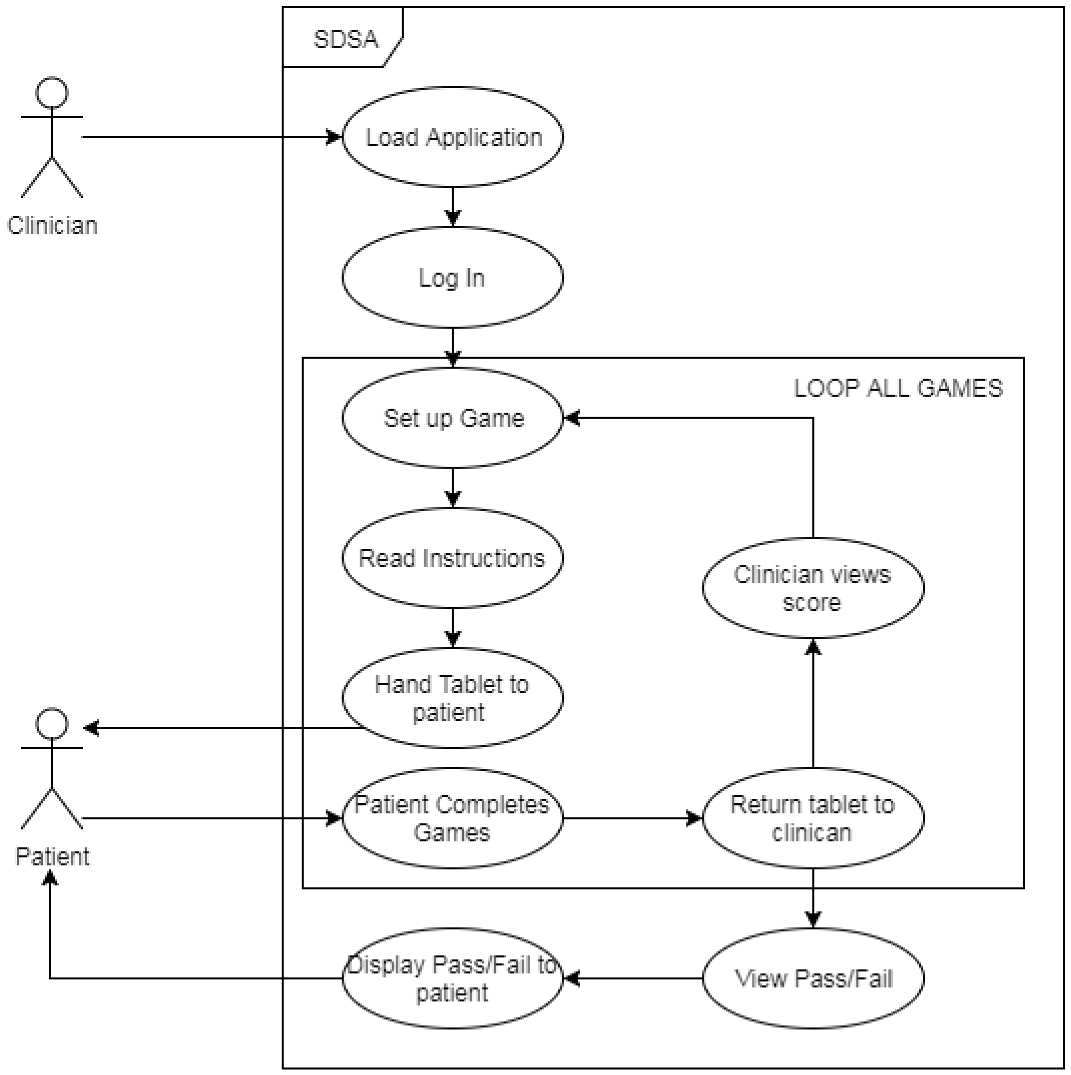


Use Cases

Use Cases are a vital aspect of the analysis and design of a system, as they identify, clarify and organise system requirements (Rouse, 2018). Designed to log how users will interact with the system to achieve the desired goal, such as calculating a Pass/Fail score to identify if a stroke patient is ready to drive again (Brandenburg, 2018). Within the SDSA application, two actors interact with the system, the clinician and the patient. Each actor pursues different roles; e.g., the clinician performs admin roles such as creating accounts, setting up each game, and sharing the final pass/fail score to the patient, whereas the patient has the lowest system access, however, spends the most time handling the application, in which they play through all five cognitive tests. Before another of the use cases can begin, the clinician must log into the system, if an account does not exist for the patient; the clinician must create an account first. The only use case the patient interacts with directly is the 'Play Game' use case, in which they complete the five cognitive task after the clinician sets up each game. A secondary use case diagram was created to display the system from the point of view of the clinician to show their user experience with the SDSA application and their interaction with the patient actor.



Secondary Use Cases



Text Use Cases

ID: P1	**Name:** Login
Actors: Clinician	**Preconditions:** Device is turned on, the application is loaded, and an account has been created.
Main Flow: When the user loads the application, a prompt will be displayed to enter the clinician email and password, these inputs are validated and checked through the database. A valid login will need to be accepted before the clinician can access the further elements of the system.	**Alternative Flow:** Clinician enters incorrect email, resulting in an error message appearing. Clinician doesn't have an account, in which they will have to create an account.
Post Conditions: Access to the main menu and the cognitive tests	

ID: P2	**Name:** Create Account
Actors: Clinician	**Preconditions:** Application is loaded, the clinician doesn't have an account for the SDSA system.
Main Flow: If the clinician doesn't have a login, they will be able to create one to use to access the SDSA system further, the Create Account method will require inputs such as email and password. These details will be saved in the database to be used for login validation.	**Alternative Flow:** Invalid details are entered. Therefore the account cannot be saved to the database and created.
Post Conditions: Clinician is able to log into the system	

ID: P3	**Name:** Enter Patient Details
Actors: Clinician	**Preconditions:** The clinician has successfully logged into the system with a valid email and password, which appears in the database
Main Flow: To ensure the SDSA is specific to each patient, the clinician will enter patient information into the system such as ID and locality, e.g. European Signs, this information will be entered into the database and used to link test scores to a specific patient.	**Alternative Flow:** Patient details are incorrectly entered therefore the system cannot proceed, e.g. entering a string datatype for Patient ID instead of an integer.
Post Conditions: patient information will be saved to the database.	

ID: P4	**Name:** Send patient details to Database
Actors: N/A	**Preconditions:** patient details have been entered
Main Flow: When button is clicked, patient details entered will be saved in the relevant table/fields to be retrieved later to link patient to the score.	**Alternative Flow:** Invalid format of details entered. Patient table cannot be found. Patient's details already in the database.
Post Conditions: Introductory page will display allowing the cognitive tests to begin.	

ID: P5	**Name:** Introductory Page
Actors: Clinician	**Preconditions:** Device is turned on, application is loaded, and an account has been created.
Main Flow: Once the patient details participating in the current SDSA have been entered, the system will display screen to enter patient details, select locality and begin testing.	**Alternative Flow:** Screen doesn't display due to errors in clinician/patient log in.
Post Conditions: Clinician will set up the cognitive tests for the user to participate in.	

ID: P6	**Name:** Setup Game
Actors: Clinician	**Preconditions:** Patient ID entered, locality selected, introductory page displayed
Main Flow: N/A	**Alternative Flow:** Errors setting up game, button link from introductory not working.
Post Conditions: Patient ready to do assessment	

ID: P7	**Name:** Play Game
Actors: Patient	**Preconditions:** The clinician is logged in, the current SDSA test session is patient specific, and the initial game has been set up by the clinician.
Main Flow: To begin the loop within the overall uses cases. On the application, the patient will view the instructions for each of the five cognitive tests, before completing them to the best of their ability by clicking the 'next' button at the end of each test to begin the subsequent test, until the last test is completed, allowing the next use case to begin.	**Alternative Flow:** The instructions do not display, possibly resulting in invalid results as the user may not fully understand the tests. Patient isn't able to access the next game, as they cannot finish a game due to ambiguity or other errors.
Post Conditions: The system can collect a score from each cognitive test completed by the patient to enable the final pass/fail score to be generated and stored later in the system.	

ID: P8	**Name:** Calculate Score
Actors: N/A	**Preconditions:** Patient completes each cognitive game.
Main Flow: Using the various attributes of each game such as number of answer correct/incorrect and time, the system uses a specific algorithm method to calculate a percentage outcome of each game, these scores will be used to generate the final score later in the system.	**Alternative Flow:** Score cannot be calculated as user attempts to finish game before completing the current task.
Post Conditions: Each calculated score will be added to the database, and final pass/fail score generated for the patient.	

ID: P9	**Name:** Send Score
Actors: N/A	**Preconditions:** The score has been calculated/formatted to a percentage
Main Flow: The relevant scoring metric from each game will be added to the database, with a foreign key of the patient ID to ensure the scores are patient specific. To exit the loop of this process, the user must complete all five cognitive tests.	**Alternative Flow:** Unable to save to database, due to being offline. Score is incorrect format. Game ends before score can be taken.
Post Conditions: Generate the Pass/Fail score using the scores from each game in the database for the specific patient participating.	

ID: P10	**Name:** Generate Pass/Fail
Actors: N/A	**Preconditions:** All games have been completed relevant scores have been sent to database
Main Flow: Using the relevant scoring metrics from each game from the database as parameters, an algorithm will be used to identify if the patient has passed or failed the SDSA if the final percentage is above or below the boundary.	**Alternative Flow:** Not all metrics are taken correctly, e.g. error causes application to close. Any anomalies with individual test score parameters. Database can't be accessed to retrieve scores.
Post Conditions: Save final score to database and display the final outcome to the patient.	

ID: P11	**Name:** Save Final Score in Database
Actors: N/A	**Preconditions:** Pass/Fail for the patient has been generated.
Main Flow: The final score (pass/fail) generated in the previous use case will be saved to the database for that specific patient, this will be stored securely for later reference.	**Alternative Flow:** Table is not set up to save the final score. Final score isn't in the correct data format.
Post Conditions: Display the score to the patient.	

ID: P12	**Name:** View Score
Actors: Patient	**Preconditions:** Score generated and saved in the database.
Main Flow: The application will display to the patient if they have passed or failed the SDSA.	**Alternative Flow:** Screen doesn't display correct score System cannot retrieve score from database.
Post Conditions: Return score to user, return tablet to clinician.	

Secondary Use Cases – Clinician User Experience

ID: S1	**Name:** Load Application
Actors: Clinician	**Preconditions:** Tablet acquired by clinician
Main Flow: To begin the SDSA, the clinician will load the android application on the tablet	**Alternative Flow:** Tablet is unable to load the application due to compatibility or errors. Application is not downloaded on specific tablet Tablet doesn't have sufficient battery power to load.
Post Conditions: Clinician is able to log in allowing patients to complete the SDSA	

ID: S2	**Name:** Log In
Please Refer to Use Case ID: P1	

ID: S3	**Name:** Setup game
Please Refer to Use Case ID: P6	

ID: S4	**Name:** Read Instructions
Actors: Clinician, Patient	**Preconditions:** Game is Setup on the tablet and is ready to be played.
Main Flow: As each game has a different set of instructions, the clinician will read the instructions for each game to ensure the patent fully understands the cognitive test they are about to complete, to avoid any ambiguity issues that may result in invalid scores.	**Alternative Flow:** Screen doesn't display correct game instructions before the current test. Patient doesn't fully understand the instructions, clinician required to repeat instructions.
Post Conditions: Patient is ready to participate in the SDSA cognitive tests on the tablet	

ID: S5	**Name:** Hand Tablet to Patient
Actors: Clinician, Patient	**Preconditions:** Instructions for the current test have been read to the patient.
Main Flow: To enable the patient to interact with the system and participate in the cognitive tests, the clinician must physically hand the tablet containing the SDSA application to the patient.	**Alternative Flow:** n/a
Post Conditions: Patient completes cognitive tests.	

ID: S6	**Name:** Patient Completes Game
Actors: Patient	**Preconditions:** Patient is in possession of the tablet with the current game set up correctly.
Main Flow: The patient interacts with the tablet to complete each of the five tests by following the instructions stated earlier by the clinician prior to being handed the tablet.	**Alternative Flow:** patient doesn't click the finish button. Patient skips a section of the test Patient completes game incorrectly
Post Conditions: Patient returns the tablet to the clinician after each test	

ID: S7	**Name:** Return Tablet to clinician
Actors: Patient, Clinician	**Preconditions:** The current cognitive test of the SDSA has been completed by the patient.
Main Flow: The patient will show they have finished the current test and are ready for the next test or the end of the five tests, the Next or Finish button will be clicked and the tablet will be handed back to the clinician.	**Alternative Flow:** N/A
Post Conditions: Clinician views final score from tablet screen	

ID: S8	**Name:** Clinician views score
Actors: Clinician	**Preconditions:** Tablet has been returned to clinician from the patient.
Main Flow: Clinician can view the patient's score of each game that has been calculated by the system.	**Alternative Flow:** N/A
Post Conditions: Return to set up the next game and cycle through the loop again.	

ID: S9	**Name:** View Pass/View
Actors: Patient, Clinician	**Preconditions:** Loop has been exited after completing all five tests and handing tablet back to clinician.
Main Flow: The final score will be calculated for the clinician to view from the database on the application	**Alternative Flow:** System cannot retrieve score from database. Error calculating score. Retrieve wrong patients score. Score inaccurate.
Post Conditions: Share score will help patient.	

ID: S10	**Name:** Display Pass/Fail to patient
Actors: Clinician, Patient	**Preconditions:** Patient's Pass/Fail score retrieved from database
Main Flow: The clinician will share the calculated pass or fail score to the specific patient to share if they are able to resume driving again after their stroke.	**Alternative Flow:** System doesn't display score on screen.
Post Conditions: Close SDSA application	

References

Rouse, M. (2018). What is use case? - Definition from WhatIs.com. [online] SearchSoftwareQuality. Available at: http://searchsoftwarequality.techtarget.com/definition/use-case [Accessed 9 Feb. 2018].

Brandenburg, L. (2018). How to Write a Use Case. [online] Bridging-the-gap.com. Available at: http://www.bridging-the-gap.com/what-is-a-use-case/ [Accessed 9 Feb. 2018].

Rouse, M. (2018). What is class diagram? - Definition from WhatIs.com. [online] SearchMicroservices. Available at: http://searchmicroservices.techtarget.com/definition/class-diagram [Accessed 9 Feb. 2018].