ACS: Design Document

By: Harsh Bali

4 Design

4.1 Introduction

The focus of this chapter is to present the design and design choices for this project. The chapter starts with a discussion on the software architecture appropriate for this project and the technologies that can potentially be used to realize this architecture. Then the next sections present the design activities that were carried out over the course of the Scrum, in a sprint-by-sprint format.

4.2 Software Architecture

The core of this project is simulation; thus, the software solution's architecture must accommodate the nature of the project. There are various technologies available that allows to build a software solution that has simulation at its core.

4.2.1 Determining the architecture

It is discovered in Background chapter, that the service that would be the primary focus of this project would be "booking and triaging" service and the simulation models would be developed using higher-level programming languages. A very minimum software solution would be a single script that takes the input from the decision maker, runs the simulation, and then generates the output, as shown in *Figure 8*.

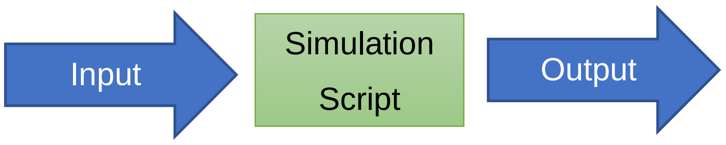


Figure 8 Single Script Architecture

However, using single script architecture would not allow the software solution to meet the few of the sprint goals and these are:

- **Storing the input and output for future experiments:** Inability to store the input and output datasets, means future experiments cannot be conducted using this dataset.
- **Creating interface for non-technical users to interact with the simulation:** The decision maker would not be able to interact with software through interface and would require to directly interact with the script. If decision makers are a team that consists of non-technical members, then it would become harder for them to use the software solution.

In-order to accomplish these requirements an interface and a database could be used. User-interface would allow non-technical decision makers to interact with the simulation whereas a database would provide the ability to store the input and output data.

The Background chapter also states that in the future, the system would include simulations of other hospital's services. Using same script to simulate these services would make it harder to maintain and use the script. Also, many of these services would need to be simulated in different manner. Therefore, a type of software architecture is required, that allows to meet the requirements but also offer flexibility in which the future services would be simulated using different simulation technologies to identify risks.

A software architecture that provides this functionality is called Service Oriented Architecture (SOA) and one way realize this architecture is through web services. Implementing SOA through web-services, would allow the system to comprise of frontend that allows user's interaction with system, a backend that consists of various simulations of different hospital's services, and a database for storage purposes. The *Figure 9* gives a high-level software architecture of the project.

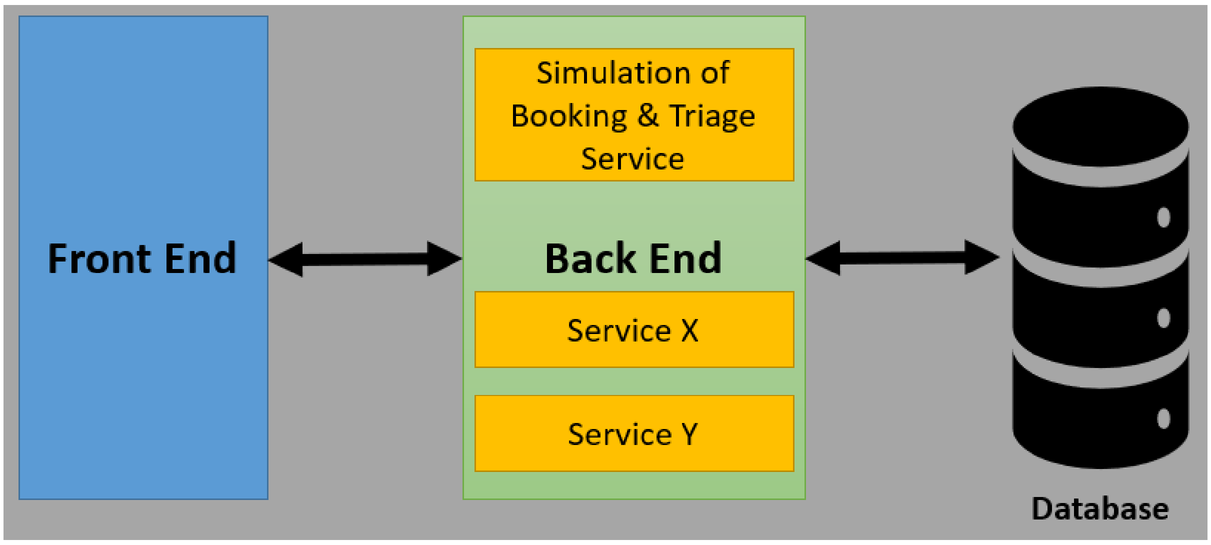


Figure 9 SOA through Web Services

4.2.2 Determining technologies

For the creation of the frontend, technologies like HTML, CSS, and basic JavaScript could be used to provide an interface to the decision makers to interact with the application. However, as per *Pros and Cons of ReactJS - javatpoint, 2022,* using these technologies results in developing static web pages that can cause performance issues as the DOM gets statically updated. Moreover, using these technologies involves using a lot of boiler plate code again and again which could increase the development time of the frontend. On the other hand, using React JS ability to reuse components eliminates the need for coding boilerplate HTML code, thus, saving time. Moreover, the virtual DOM introduced by React JS provides better performance than static HTML pages. Thus, React JS would be used to develop the front-end for the system.

Through the process of comparing technologies, most of the backend technologies available can meet the current backend needs of the system, however, what about the future needs? It was discovered before that the project starts with simulation of BT service in the Emergency department, but it is not limited to this service. Thus, in the future, the project may include many simulation models of services offered by the hospital's departments. Therefore, choosing a backend technology that offers the simulation modelling for many types of simulations and provides the ability to analyze datasets would be ideal for this project. Python would be used as a backend for this project as it offers many libraries to construct simulation models and offers libraries that helps with analyzing datasets.

A simple Python script would not act as communication medium between frontend and the database. Therefore, Flask web framework of Python would be used, in-order to establish the communication medium between the frontend and the database.

HB

The service being simulated is BT service and it is being simulated as a discrete event simulation. A Python library that allows to construct discrete event simulations is called SimPy. Thus, this library would be used to build simulation of BT service for this project.

There are many types of database types that can be used to store input and output datasets for this project. However, MySQL database would be used in this project to benefit from scalability and flexibility offered by this relational database. Therefore, using MySQL would allow the database to scale up when the simulation of the new service is added.

The *Figure 10* shows which technologies would be used to realize SOA in this project.

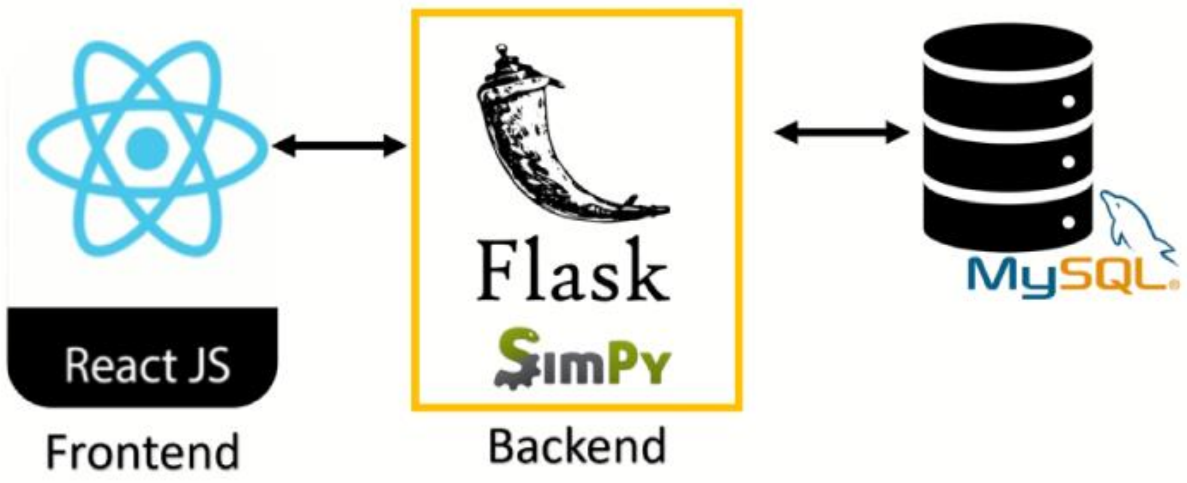


Figure 10 Technologies used to realize SOA

4.3 Sprint 1

In sprint one, in-order to create simulation model for the “As-is” BT service a conceptual modelling activity was carried out and thus a simple conceptual model for the service was developed.

At first, literature (When to go to A&E, 2022) was used to create graphical representation of how patient traverses through the A&E department (*Figure 11)*. From this input and output were determined (*Table 4)*. After that, system’s components were identified from the graphical representation, and it is determined whether to include or exclude the component in conceptual model (*Table 5)*. Then how to represent these components and assumptions about components was determined (*Table 6)*. Lastly, the representation and assumptions of components were used to revise the graphical representation and then this representation becomes the high-level conceptual model of the system (*Figure 12)*.

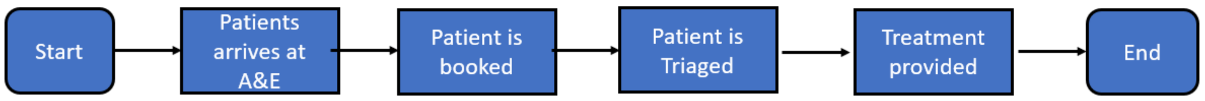


Figure 11 Graphical Representation

Table 4 "As-is" Simulation: Inputs & Outputs

Input	**Output**
Receptionist number No.	Time spent in booking queue
Triage nurse number No.	Time spent while getting booked
Patient interval time estimation	Time spent in triaging queue
Estimation of booking time	Time spent while getting triaged
Estimation of triage time	Total time spent waiting

Table 5 Components

Components		**Include/Exclude**	**Justification**
Entities	Patients	Include	Flow through the service process
	Receptionist	Exclude	Do not flow through the service process.
	Triage Nurse	Exclude	Do not flow through the service process.
Activities	Booking	Include	Related to speed of booking & triaging service.
	Triaging	Include	Related to speed of booking & triaging service.
	Treatment, Transfer & Discharge	Exclude	Not related to speed of booking & triaging service.
Queues	Booking queue	Include	Required for booking waiting time.
	Triaging queue	Include	Required for triaging waiting time.
Resources	Receptionist	Include	If the receptionist is busy, then time spent in booking queue increases.
	Triage Nurse	Include	If the nurse is busy triaging, then time spent in triaging queue increases.

HB

	Doctors	Exclude	Patient only interacts with doctors in **treatment activity** and the treatment activity is excluded.

Table 6 Representation & Assumptions

Components		**Representation**	**Assumptions**
Entities	Patients	A patient would be represented as an entity. 1 entity = 1 patient. A Patient would have an ID.	Once patient enters the A&E, they can only leave when they are seen by doctor.
Activities	Booking	Booking & Triaging activities would be represented as the processes in the system that the patient goes through.	1 time unit in simulation would represent 1 minute.
	Triaging		
Queues	Booking queue	Patient waits in this queue to go through the booking process	The queues have unlimited capacity.
	Triaging queue	Patient waits in the queue to go through the triaging process	
Resources	Receptionist	For the booking activity, receptionist is allocated. Patient would only complete the booking activity if they have been seen by receptionist.	The receptionist & triage nurse do not take breaks.
	Triage Nurse	For the triaging activity, receptionist is allocated. Patient would only complete the triaging activity if they have been seen by triage nurse.	

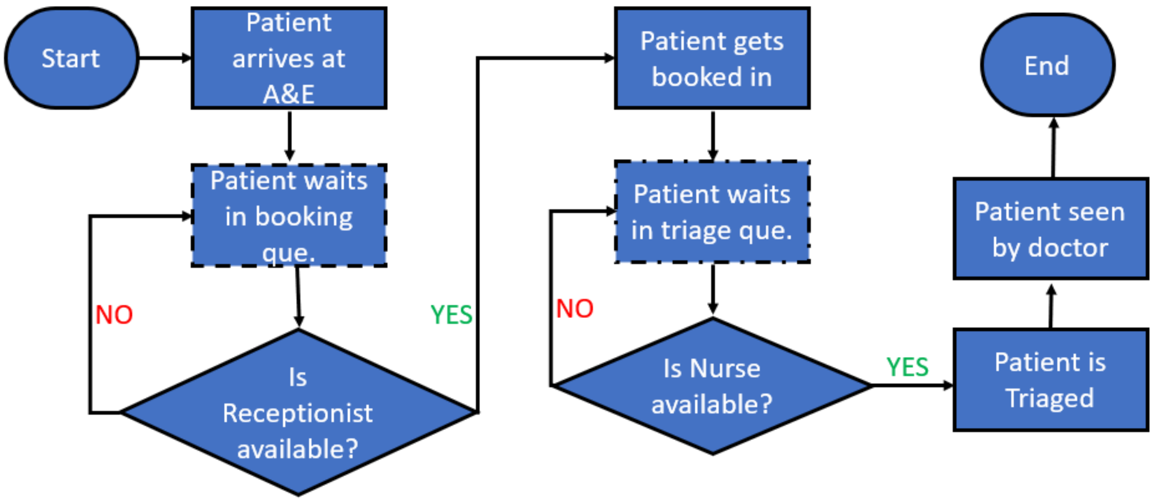


Figure 12 "As-is" Conceptual model

4.4 Sprint 2

In sprint two, conceptual model for "digitalized" BT service was created and this section documents the conceptual modelling process.

At the time of sprint two, no literature was available that describes how patients would traverse through BT service that uses IoT devices. Therefore, in-order to create conceptual model for digitalized service, the conceptual model in sprint one was used. During the creation of the conceptual model for the "digitalized" BT service, it was discovered that conceptual model for "digitalized" service shares most of its features with the conceptual model of "As-is" service (*Figure 13)*. Thus, it was determined that rather than building two different conceptual models, the conceptual model from sprint 1 could be used and components that represents digitalization could be built on top of it.

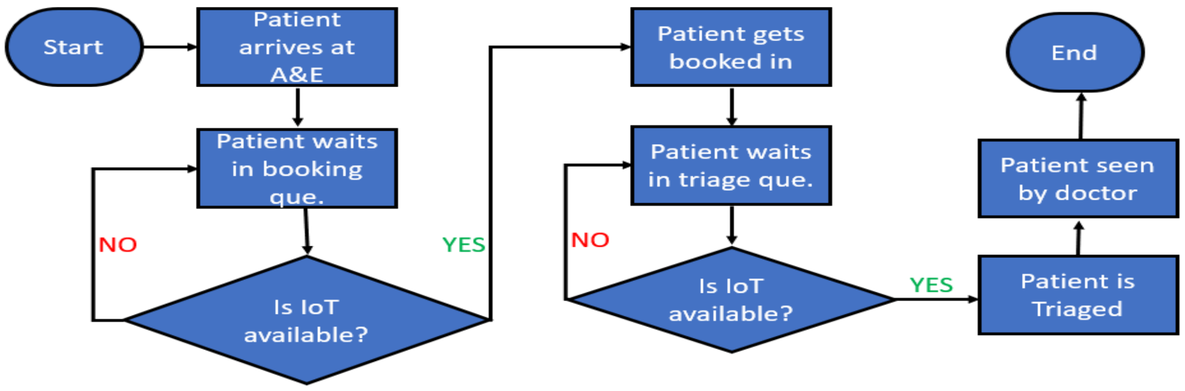


Figure 13 "Digitalized" 1st Conceptual Model

It was identified in the Background Chapter, that in digitalization version of the BT service, IoT devices could replace receptionists and triage nurses. However, in-order to compare the services, the number of IoT booking device would be **equivalent** to number of receptionists and number of IoT triaging device would be **equivalent** to number of nurses. Therefore, no input is needed for number of IoT devices.

It was also identified in the Background Chapter, that IoT devices would be equipped with security control that mitigates this risk but causes a **delay**. As, delay varies device to device the delay would be entered by the decision maker (*Table 7)*.

Moreover, as the goal for the simulation experiments is compare the performance of digitalized BT service with the "as-is" service, thus, additional outputs would also be required, and these are listed in *Table 7.*

Table 7 Additional Inputs & Outputs

Additional Inputs	**Additional Outputs**
Delay	Time spent in digitalized booking queue
	Time spent while getting booked digitally
	Time spent in digital triage queue
	Time spent while getting triaged digitally

After that, additional inputs and outputs were used to identify additional components of the digitalized service (*Table 8)*. Then how to represent these components and assumptions about components was determined (*Table 9)*. Lastly, a process flow diagram created in sprint 1 was updated to represent the conceptual model of both services (*Figure 14)*.

Table 8 Additional Components

Additional Components		**Justification**
Activity	Delay	Related to speed of "digitalised" booking & triaging service.
Resources	IoT Receptionist No.	If the IoT receptionist is busy, then time spent in "digitalised" booking queue increases.
	IoT Triage Nurse No.	If the nurse is busy triaging, then time spent in "digitalised" triaging queue increases.

HB

Table 9 Additional Components' Representations & Assumptions

Additional Components		**Representation**	**Assumptions**
Activity	Delay	Delay would be represented as process.	Delay occurs every time a patient uses IoT devices.
Resources	IoT Receptionist No.	Equivalent to receptionist no.	IoT devices do not break. The connectivity is always available.
	IoT Triage Nurse No.	Equivalent to triage nurse no.	

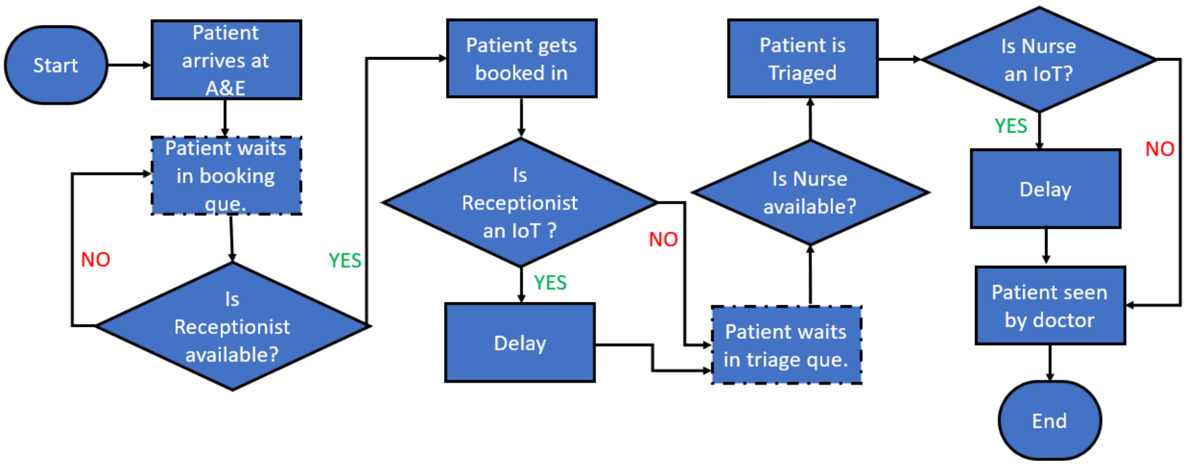


Figure 14 Conceptual Model that represents "as-is" & "digitalized" BT service

4.5 Sprint 3

4.5.1 Use Case Diagram

At first, the Use Case diagram for the system was created to uncover the interactions of the decision maker (user) with the system *(Figure 15).*

HB

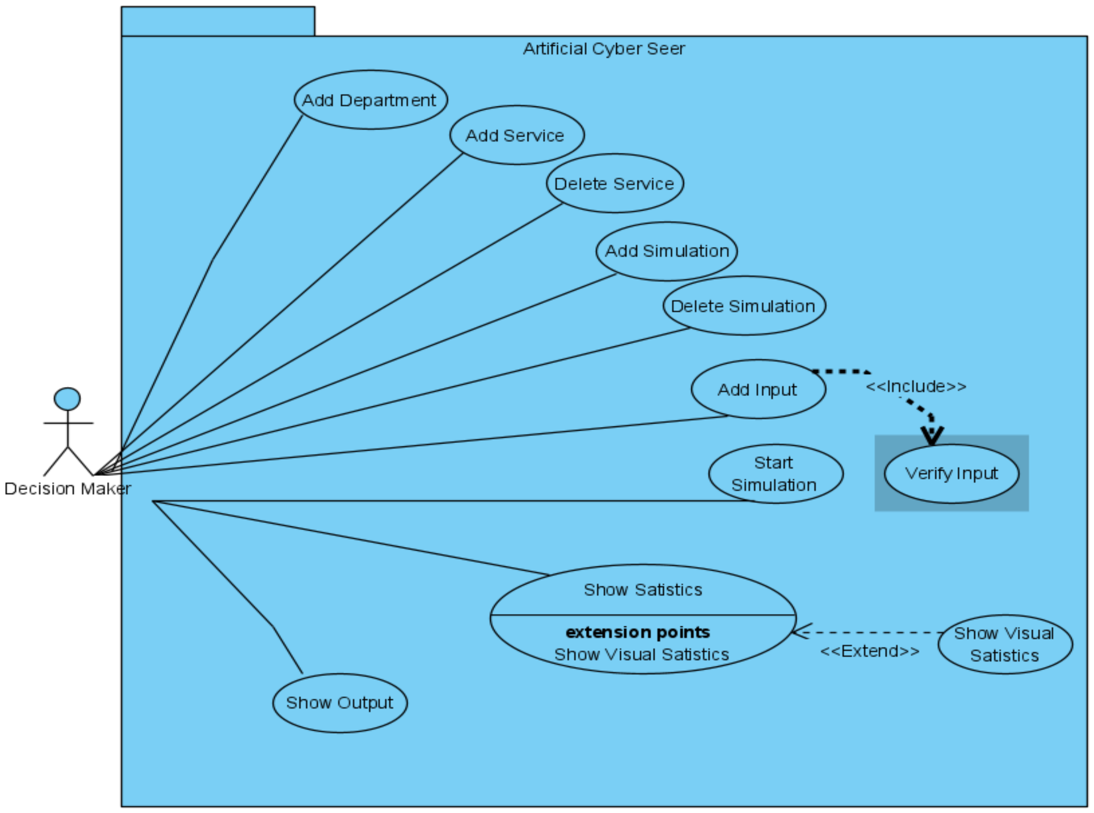


Figure 15 Use Case Diagram of Artificial Cyber Seer

4.5.2 ERD diagram

Using Use case diagram ERD diagram was created to discover the entities of the system, the attributes of an entity and the relationships between the entities (*Figure 16).*

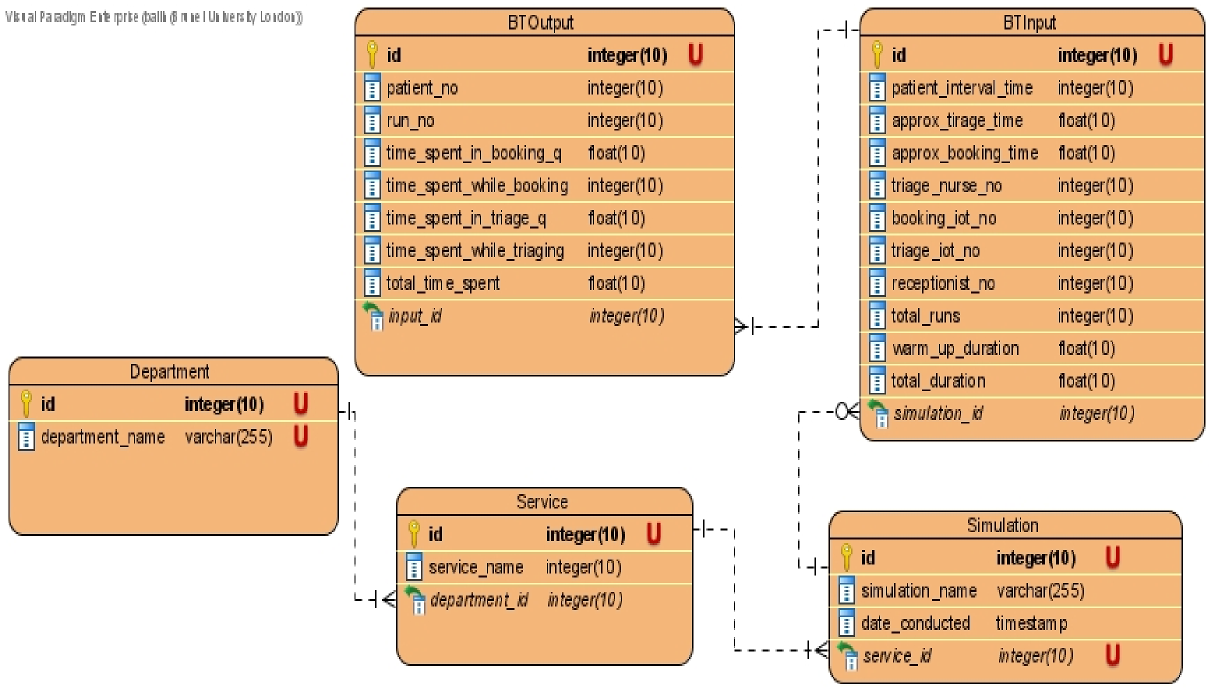

Visual Paradigm Enterprise (Brunel University London)
BTOutput
id integer(10) U
patient_no integer(10)
run_no integer(10)
time_spent_in_booking_q float(10)
time_spent_while_booking integer(10)
time_spent_in_triage_q float(10)
time_spent_while_triaging integer(10)
total_time_spent float(10)
input_id integer(10)
BTInput
id integer(10) U
patient_interval_time integer(10)
approx_tirage_time float(10)
approx_booking_time float(10)
triage_nurse_no integer(10)
booking_iot_no integer(10)
triage_iot_no integer(10)
receptionist_no integer(10)
total_runs integer(10)
warm_up_duration float(10)
total_duration float(10)
simulation_id integer(10)
Department
id integer(10) U
department_name varchar(255) U
Service
id integer(10) U
service_name integer(10)
department_id integer(10)
Simulation
id integer(10) U
simulation_name varchar(255)
date_conducted timestamp
service_id integer(10) U

Figure 16 ERD of ACS

HB

4.5.3 Class diagram

The Service oriented architecture would be used for this project, and it would be implemented through web services. Due to this, the architecture of the system would be **divided into frontend, backend, and a database.** Therefore, to understand the separation in structure of the system caused by these three divisions a class diagram that implements MVC design pattern was created (*Figure 17)*.

HB

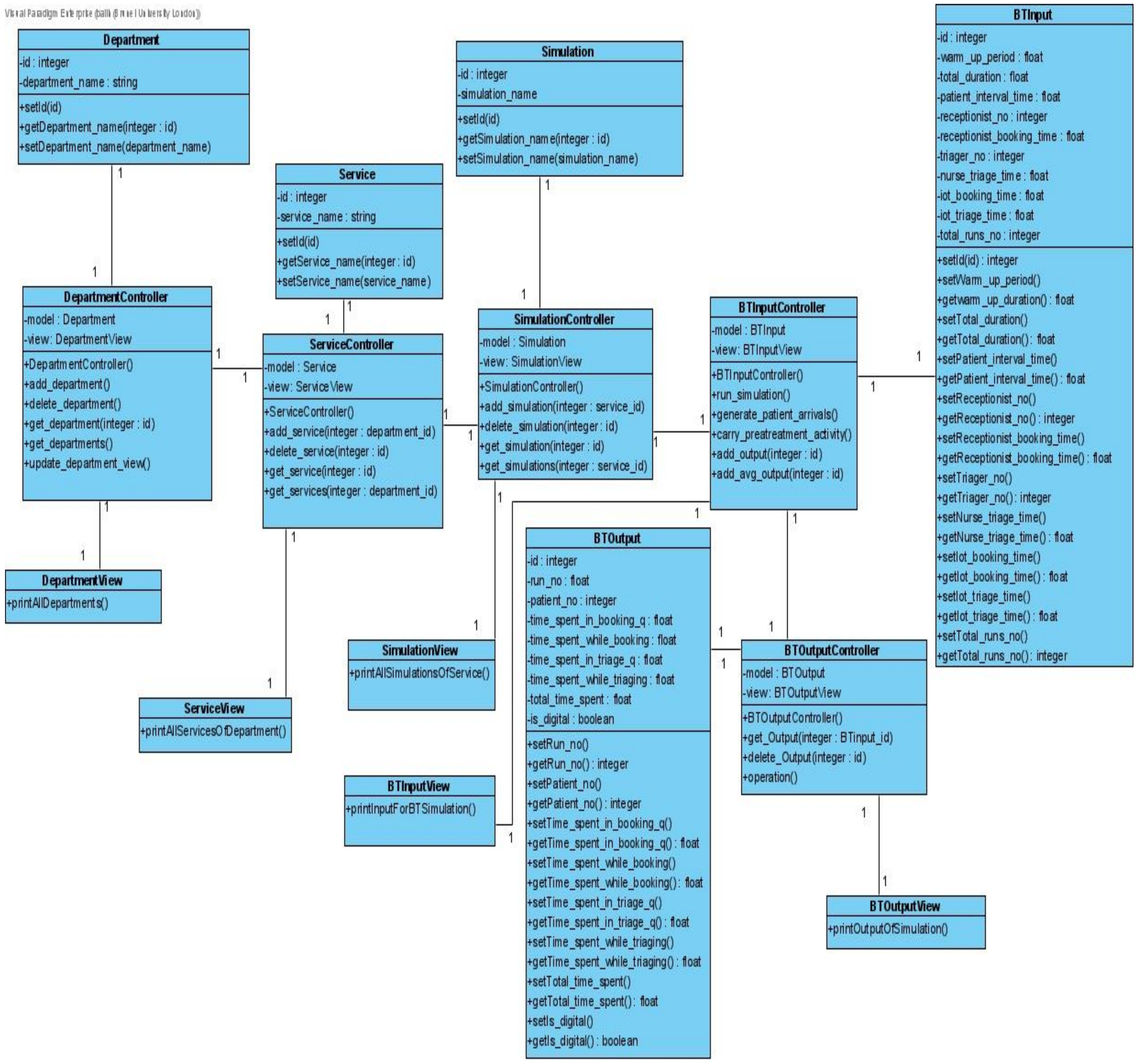

Visual Paradigm Enterprise (balli (Brunel University London))
Department
-id : integer
-department_name : string
+setId(id)
+getDepartment_name(integer : id)
+setDepartment_name(department_name)
Simulation
-id : integer
-simulation_name
+setId(id)
+getSimulation_name(integer : id)
+setSimulation_name(simulation_name)
BTInput
-id : integer
-warm_up_period : float
-total_duration : float
-patient_interval_time : float
-receptionist_no : integer
-receptionist_booking_time : float
-triager_no : integer
-nurse_triage_time : float
-iot_booking_time : float
-iot_triage_time : float
-total_runs_no : integer
+setId(id) : integer
+setWarm_up_period()
+getwarm_up_duration() : float
+setTotal_duration()
+getTotal_duration() : float
+setPatient_interval_time()
+getPatient_interval_time() : float
+setReceptionist_no()
+getReceptionist_no() : integer
+setReceptionist_booking_time()
+getReceptionist_booking_time() : float
+setTriager_no()
+getTriager_no() : integer
+setNurse_triage_time()
+getNurse_triage_time() : float
+setIot_booking_time()
+getIot_booking_time() : float
+setIot_triage_time()
+getIot_triage_time() : float
+setTotal_runs_no()
+getTotal_runs_no() : integer
Service
-id : integer
-service_name : string
+setId(id)
+getService_name(integer : id)
+setService_name(service_name)
DepartmentController
-model : Department
-view: DepartmentView
+DepartmentController()
+add_department()
+delete_department()
+get_department(integer : id)
+get_departments()
+update_department_view()
ServiceController
-model : Service
-view: ServiceView
+ServiceController()
+add_service(integer : department_id)
+delete_service(integer : id)
+get_service(integer : id)
+get_services(integer : department_id)
SimulationController
-model : Simulation
-view: SimulationView
+SimulationController()
+add_simulation(integer : service_id)
+delete_simulation(integer : id)
+get_simulation(integer : id)
+get_simulations(integer : service_id)
BTInputController
-model : BTInput
-view: BTInputView
+BTInputController()
+run_simulation()
+generate_patient_arrivals()
+carry_preatreatment_activity()
+add_output(integer : id)
+add_avg_output(integer : id)
DepartmentView
+printAllDepartments()
BTOutput
-id : integer
-run_no : float
-patient_no : integer
-time_spent_in_booking_q : float
-time_spent_while_booking : float
-time_spent_in_triage_q : float
-time_spent_while_triaging : float
-total_time_spent : float
-is_digital : boolean
+setRun_no()
+getRun_no() : integer
+setPatient_no()
+getPatient_no() : integer
+setTime_spent_in_booking_q()
+getTime_spent_in_booking_q() : float
+setTime_spent_while_booking()
+getTime_spent_while_booking() : float
+setTime_spent_in_triage_q()
+getTime_spent_in_triage_q() : float
+setTime_spent_while_triaging()
+getTime_spent_while_triaging() : float
+setTotal_time_spent()
+getTotal_time_spent() : float
+setIs_digital()
+getIs_digital() : boolean
SimulationView
+printAllSimulationsOfService()
BTOutputController
-model : BTOutput
-view: BTOutputView
+BTOutputController()
+get_Output(integer : BTinput_id)
+delete_Output(integer : id)
+operation()
ServiceView
+printAllServicesOfDepartment()
BTInputView
+printInputForBTSimulation()
BTOutputView
+printOutputOfSimulation()
1

Figure 17 Class diagram of ACS

HB

4.5.4 REST API design

The frontend would need to communicate with the backend and for this communication REST API would be used. The design for the API is documented in the *Table 10.*

Table 10 REST API Design

Category	**Endpoint**	**HTTP method**	**What should be returned?**
Department	/departments	GET	RETURNS all departments
	/department	POST	ADDS a department
	/<id>/department	GET	RETURNS a department by id
	/department/delete/<id>	DELETE	DELETE a department by ID
Service	/department /<department_id>/services	GET	RETURNS all services of a department
	/department /<department_id>/service	POST	ADDS a service to a department
	/department/service/delete/<service_id>	DELETE	DELETE a service
Simulation	/service /<service_id>/simulations	GET	RETURNS all simulations of a service
	/service/<service_id>/simulation	POST	ADDS a simulation for a service
	/service/simulation/delete/<simulation_id>/	DELETE	DELETE a simulation
	/simulation/<simulation_id>/inputs	GET	RETURNS all inputs of a simulation

HB

BTInput	/simulation/<simulation_id>/input	POST	ADDS an input to a simulation.
			Also Triggers simulation’s start
	/simulation/input/delete/<input_id>	DELETE	DELETE an input
BTOutput	/input/<input_id>/outputs	GET	RETURNS the output associated with an input
			This is output of the simulation
	/input//output/<id>	DELETE	DELETE an Output

4.6 Sprint 4

In-order, to design the user interface (UI) low fidelity wireframes of the frontend were developed in the beginning Sprint 4. The design activity of developing the use case allowed to discover the states of the user interface and these are:

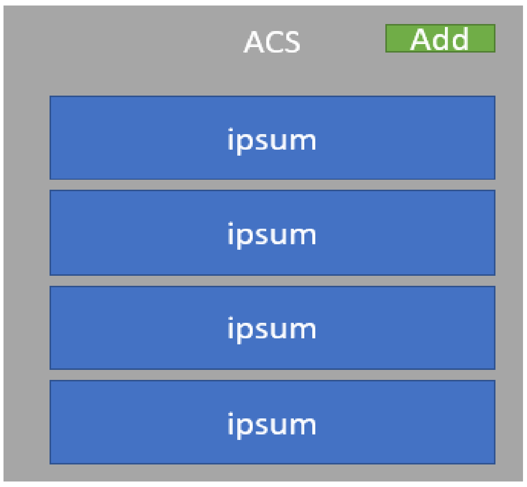
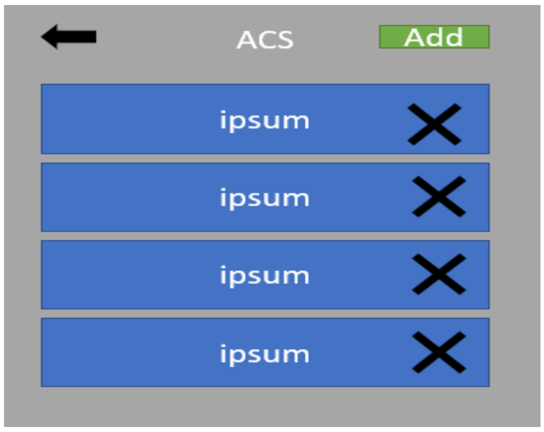
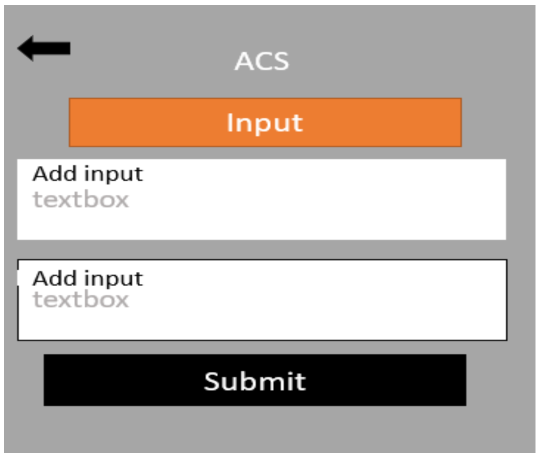
1. Show Department
2. Show Service
3. Show Simulation
4. Add Input
5. Show Output
6. Show Statistics
7. Visualize Statistics

The wireframe created are documented within the *Table 11.*

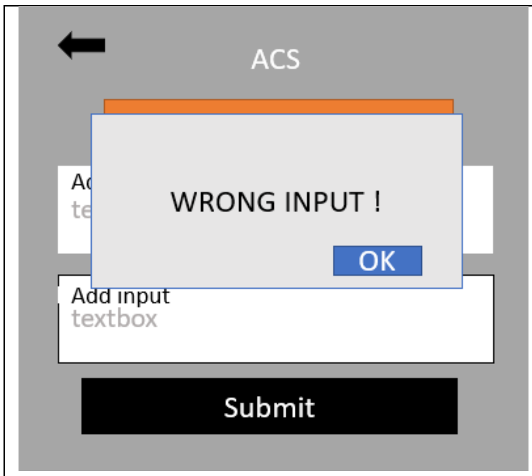
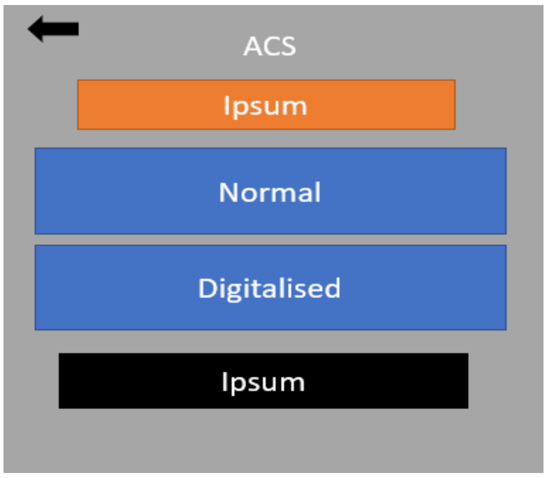
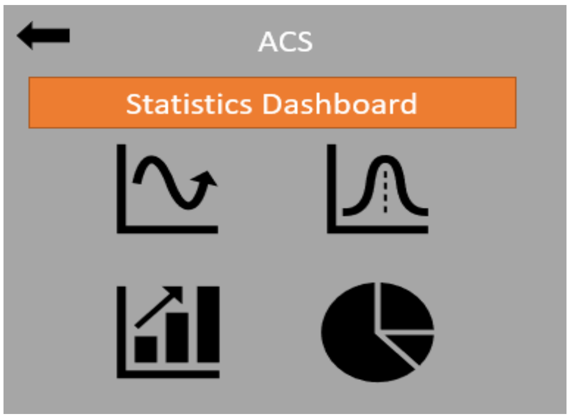
Table 11 Wireframes

Wireframe	Description

HB

	This wireframe is for the department. The blue list is used to represent list of departments that exist in the system. The add button allows the user to add the department.
	This is the wireframe for service and simulation state. Service and simulation states have same wireframe because their functionalities are similar. However, if it is service interface by clicking on the service name, the user navigates to simulation interface. On the other hand, if it is simulation interface by click on simulation name, the user navigates simulation input interface.
 ACS Input Add input textbox Add input textbox Submit	This wireframe shows the interface for the simulation input. It shows that the user would be asked to enter the input required for the simulation.

HB

 ACS Ad te WRONG INPUT ! OK Add input textbox Submit	This wireframe shows the interface for the simulation when incorrect input is submitted. The user is alerted and by clicking on “ok” button the user should be able to re-enter the data.
	Similarly, to the service and simulation interface, the simulation output and statistics interface shares the common functionality of presenting the data to the user.
	This wireframe shows the interface for the visual statistics. It shows the statistics presented through graphs.

4.7 Design Summary

Design activities plays crucial role in software development project this is because it becomes the foundation for the project. The chapter starts by discovering that the Service Oriented architecture is appropriate for this project. This further helps with determining the technologies used to realize the service-oriented architecture (SOA) of this system. Then the chapter moves onto documenting the design activities conducted throughout the Scrum. These activities

HB

involve carrying out conceptual modelling, creating UML diagrams, creating Rest API design and wireframes.

HB