SOFTWARE DESIGN

Team 5

Mark Friel
Baibing Ji
Conchur McCullough

Introduction

This report will outline the main architecture designs for an internet of things solution to be implemented in the computer science building. The solution will use data collected from sensors to improve utilisation of the building by students and will improve the operation of the building by providing building managers with more information about the facilities of the building and offering an analytsis to control the HVAC system to improve the energy efficiency of the building.

Design thinking techniques set out by the Hasso-Plattner institute of design were employed to drive forward the ideation and solution for problems that the main stakeholders of the computer science building face.

The design architecture model that will be used to design the system will be the 4+1 model set out by Kruchten in his paper Architectural Blueprints. This model introduces multiple views, these views are meant to reflect the multiple stakeholders concerned with the design of the software system: end-users, system developer, project managers. The views are designed with an architecture centric scenario driven design. The four views that will be outlined in the document are:

- Scenario
- Logical
- Process
- Physical
- Development

The purpose of having the different views is to avoid the problem where one diagram tries to capture the entire design of the software. Having different views allows different architecture styles and notion to be used for the best fit for the view being designed.

User Scenarios

The elements in the "4+1" model developed by Philippe Kruchten in [reference] work together seamlessly by the use of a small set of scenarios, these are general use cases for which the key functionality of the system is described. This view is actually redundant when taken with the other four views, as those views give a lower level insight of the system. However Scenarios serve two purposes:

1. Act as a driver to discover architectural elements during architecture design.
2. System can be validated against the scenarios after architecture design is complete.

The small set of scenarios below will be used to then develop the other 4 views in the architectural model that. The scenarios that we have developed were part of a Design thinking model. The process of creating these scenarios will be outlined below.

About Design Thinking

Design Thinking revolves around a deep interest in developing an understanding of the people for whom the product is being designed for. It is an iterative approach in which we seek to understand the user, challenger assumptions and redefine problems in an attempt to identify solutions that might not be instantly clear. The design thinking model that will was used to generate scenarios for this product was proposed by the Hasso-Plattner institute of Design at Stanford. The five stages of design thinking set out by the school are: Empathize, Define, Ideate, Prototype and Test. This document will detail the first three stages in this model.

To empathize with the users of the system, firstly they had to be identified. The primary stakeholders in the computer science building were divided into two categories: Students and Building managers. Building managers acts as an umbrella term for those who work, maintain and are concerned with the running of the building, on and offsite. Once the stakeholders or personas, as they will now be called, had been identified. It was then a case of trying to understand the problems that occur and the way that they do things. To get into the frame of mind of a student studying in the building was not a challenge given that the writers of this report are students themselves studying in the computer science building. To change our frame of mind to that of building manager was more of a challenge however studying in the building for the past three years has offered a good opportunity to observe the inefficiency's in running the building.

Once the personas had been identified the next step was to identify the problems and annoyances that will be faced in the day to day use of the computer science building. An actionable problem statement needed to be written this gives us a point-of-view into the problem that we are trying to address. To define problems we split the process by the personas. In a session of an hour each member put themselves into the perspective of the relevant persona and then tried to identify problems that the persona would have in the day to day use. Ideas were noted down onto sticky notes and put on a board, after it was felt that no more problems were going to be identified, then began the process of narrowing down the problems that we felt would have the biggest impact on the personas. By creating a more narrowed problem statement we felt that the resulting solution would be of higher quality and have a bigger impact on the personas.

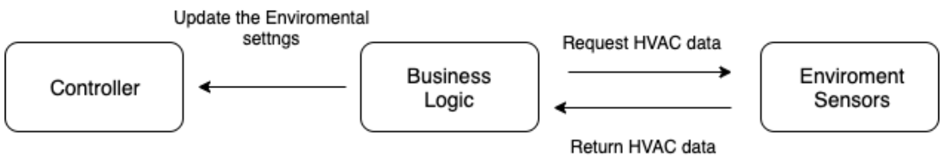
Once the problem statement had been decided upon the process of ideation began. This process combined the knowledge of the people we are designing the solution for, gained from the empathize stage, and the experience of those people using the building. Similar to the identifying problems, another session of an hour was held to discover solutions to the problem that we had identified. Again the proposed solutions were noted on sticky note and put on a board. After the ideation session it was a propose of selecting those solutions that were the most feasible and had the greatest impact. In the following scenarios some ideas we consider aspirational. These ideas will not be within the scope of the prototype or test phase of this five-part model. However they are included to show how far the solution could be taken.

This was a first exposure to Design thinking for some parts of the team. The idea of putting yourself in the mindset of the persona who will be using the system received a consensus of approval. It was felt that another step should be prescribed in the design thinking model and that would be feedback on both the problem statement and proposed solution. This would act as a checkpoint rather than prototype a solution to then find out that the problem identified has little real bearing on the daily use.

The time limited sessions of problem identification and solution ideation was felt to be an important part that we included in the design thinking process. This method was used to encourage people to vocalize their ideas quickly and concisely without being too concerned with having an extensively thought problem or solution. As the next part of the process was to narrow down options and then there is when the ideas would be fleshed out fully into a problem-statements and proposed solution.

Scenarios

The following scenarios will describe the key functionality of the business. Each scenario will follow the same template, Fist describing the persona, the goal of the persona and then how the process of the system to fulfil this goal.

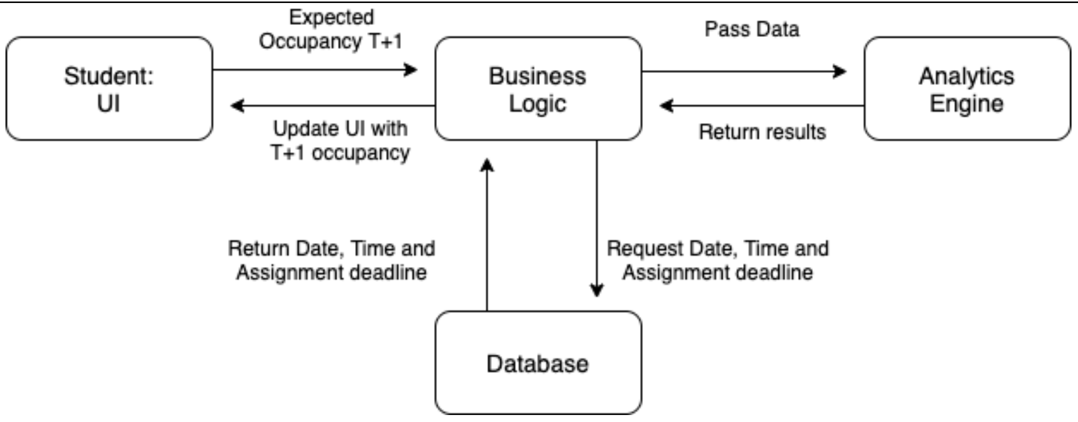
Persona: Building Managers
Problem: Currently only the electrical consumption of the building is recorded, offering no breakdown of where the most energy is consumed. For a building whose use will vary frequently during the course of a day, driven by the timetabling of the university, there is opportunity for energy wasted if the building was to operate the same way over the course of a day. **Solution:** Our solution will install sensors that will be able to track the Heat, temperature, air conditioning of the rooms in the building and then will be able to respond to changes in the environment in order to save on energy. This could be turning the lights off in an unoccupied room.
1. Requests the current : Temperature, Occupancy of the rooms in the building 1. Updates temperature, Air conditioning in the rooms, may switch lights off if room unoccupied
 Update the Enviromental settngs Controller Business Logic Request HVAC data Return HVAC data Enviroment Sensors

Persona: Student

Problem: A significant number of students who use the computer science building live outside of Belfast, to then make the journey to the computer science building can be a decision that needs to be thought out. As the undertaking of making the journey and finding that it is near capacity is not ideal.

Solution: The student will be able to check the expected occupancy one hour in advance of the current time. This will allow them to make an informed decision whether to make the journey in or not.

1. The student changes the occupancy view of the indoor map to one hour ahead of time.
2. Business logic fetches the date, time, assignment deadlines and passes these parameters to the analytics engine.
3. Analytics engine inputs these parameters to the model and passes the expected occupancy to the business logic.
4. The business logic then updates the indoor map to display the expected occupancy in one hour from the current time.


Student: UI
Expected Occupancy T+1
Update UI with T+1 occupancy
Business Logic
Pass Data
Return results
Analytics Engine
Return Date, Time and Assignment deadline
Request Date, Time and Assignment deadline
Database

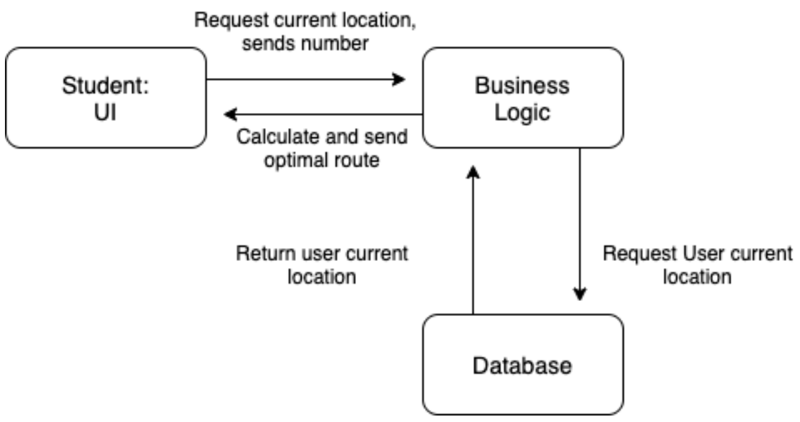
Persona: Student, Building Managers, and Visitors

Problem: The computer science building hosts the students that will go onto define and create the technology of tomorrow as well as lecturers who are experts in their field who are driving that change today. It is no surprise that the building then faces a number of visitors from companies to visiting faculty from another university. Navigating the building can be difficult for those who have no prior experience with it.

Solution: We propose a wayfinding system that will be able to receive the users current location and the room they are trying to navigate to. On the basis of these two parameters an optimal route to their desired room will be calculated.

1. User navigates to the wayfinding tab and inputs the room number they are looking directions to.
2. User interface request the user current location in building and sends room number
3. Business logic queries the database for the user’s current location in the building

4. Business logic calculates the optimal route to the room and passes to the user interface to display route.

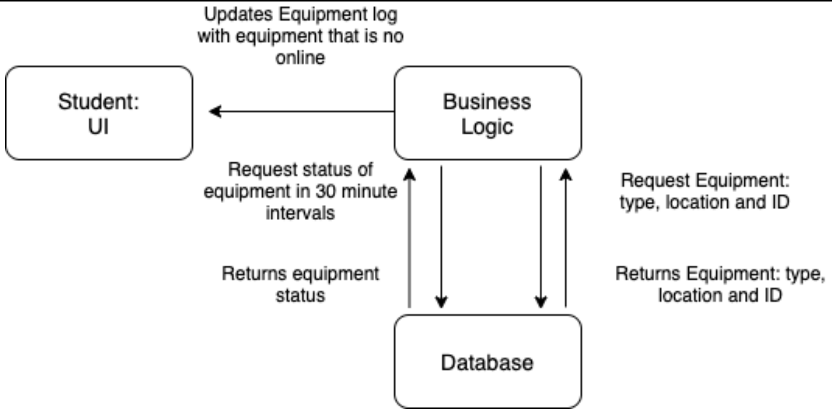

Request current location, sends number
Student: UI
Business Logic
Calculate and send optimal route
Return user current location
Request User current location
Database

Persona: Building Managers

Problem: There is a plethora of equipment contained in the building, ranging from expensive 3D printers to water dispensers. To perform a check on each piece of equipment is a timely task, one that is boring and very easily automated.

Solution: We propose that sensor will be installed for the equipment in the computer science building. These will be checked in set intervals. If a piece of equipment is not working to its full capacity, then an alert will be sent to the building managers.

1. Business logic will request the status: Online, offline, error, of equipment every 30 minutes
2. Database returns the status of equipment in the computer science building
3. Business logic requests the: Equipment type, location and ID of equipment whose status in not Online
4. Updates equipment log in the user interface

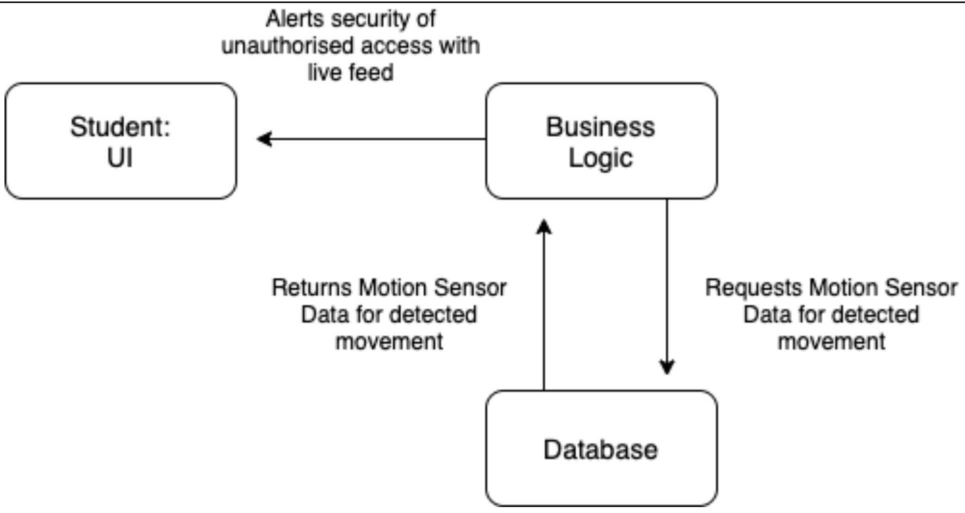

Updates Equipment log with equipment that is no online
Student: UI
Business Logic
Request status of equipment in 30 minute intervals
Request Equipment: type, location and ID
Returns equipment status
Returns Equipment: type, location and ID
Database

Persona: Building Managers

Problem: The value of equipment in the computer science building is a significant amount. It offers a tempting target for thieves as the university does not have the resources to have around the clock security for the building.

Solution : The motion sensors that have already been proposed to monitor occupancy could also serve as a security feature. If there is motion detected in the building outside the hours of operation, then an alert will be sent to the building Managers along with a live feed of where the motion was detected. This will add a layer of security as well as a quick response to a break in.

1. Business logic request motion sensor data in intervals of 5 minutes
2. If motion is detected in the building outside of operation hours alert sent to user interface
3. A live feed of the building from the CCTV cameras is displayed

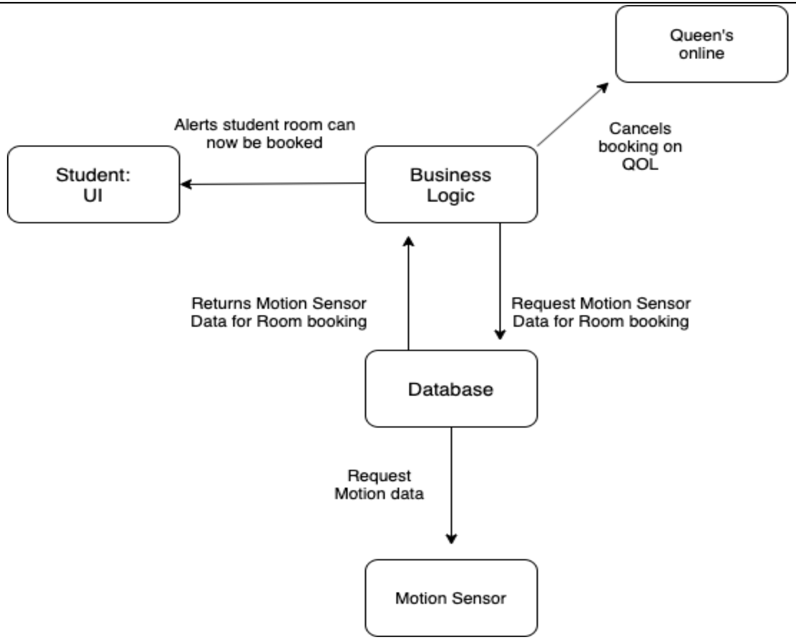

Alerts security of unauthorised access with live feed
Student: UI
Business Logic
Returns Motion Sensor Data for detected movement
Requests Motion Sensor Data for detected movement
Database

Persona: Student

Problem: The booking of a room in the computer science building and then that room going unused for the period of its booking is a problem that is frequently encountered. With the nature of software projects being quite a collaborative effort, rooms are in demand.

Solution: Our solution will leverage motion sensors in order to track the utilization of a room. If no motion is detected within the first 10 minutes of a room being booked. The booking for the room will be cancelled. If a student finds that all rooms are booked they can leave a request for a room at a specific time. When a room booking is cancelled then an alert will be sent to the app notifying them of this event

1. The user will have requested to book a room from the Queen's online website
2. Business logic requests motion sensor data from the database at 3-minute interval
3. Database sends motion data that room is unoccupied
4. Business logic cancels the room booking

5. Sends alert to student web application that a room is free to book.
 Queen's online Alerts student room can now be booked Cancels booking on QOL Student: UI Business Logic Returns Motion Sensor Data for Room booking Request Motion Sensor Data for Room booking Database Request Motion data Motion Sensor

Logical view

The logical view is designed to show a high level of how the system is broken down into its key abstractions. It decomposes the system into logical parts that interact in order to fulfil the system requirements. In this part, we provide two types of view, layer view and class view both come with brief explanation to help understand each type.

There are four layers in our system, presentation layer focus on user interface, supported by HTML language, which used to generate static pages, JavaScript help to deal with dynamic contents and receive update from lower layer. Control layer is placed in same level as presentation layer, contains APIs to control different devices, in other words, it can be seen as another presentation layer because it present lower layer's command in different ways. Then Business logic layer, this layer contains different business logics and communication infrastructures, business logic layer get metadata from data collection layer or database layer, process them, and send to presentation layer or controller. Data collection layer integrates required APIs of sensor and database queries to help this layer collect data and write data to database. Finally it is database layer support send and receive operations acts as a storage unit in whole system.

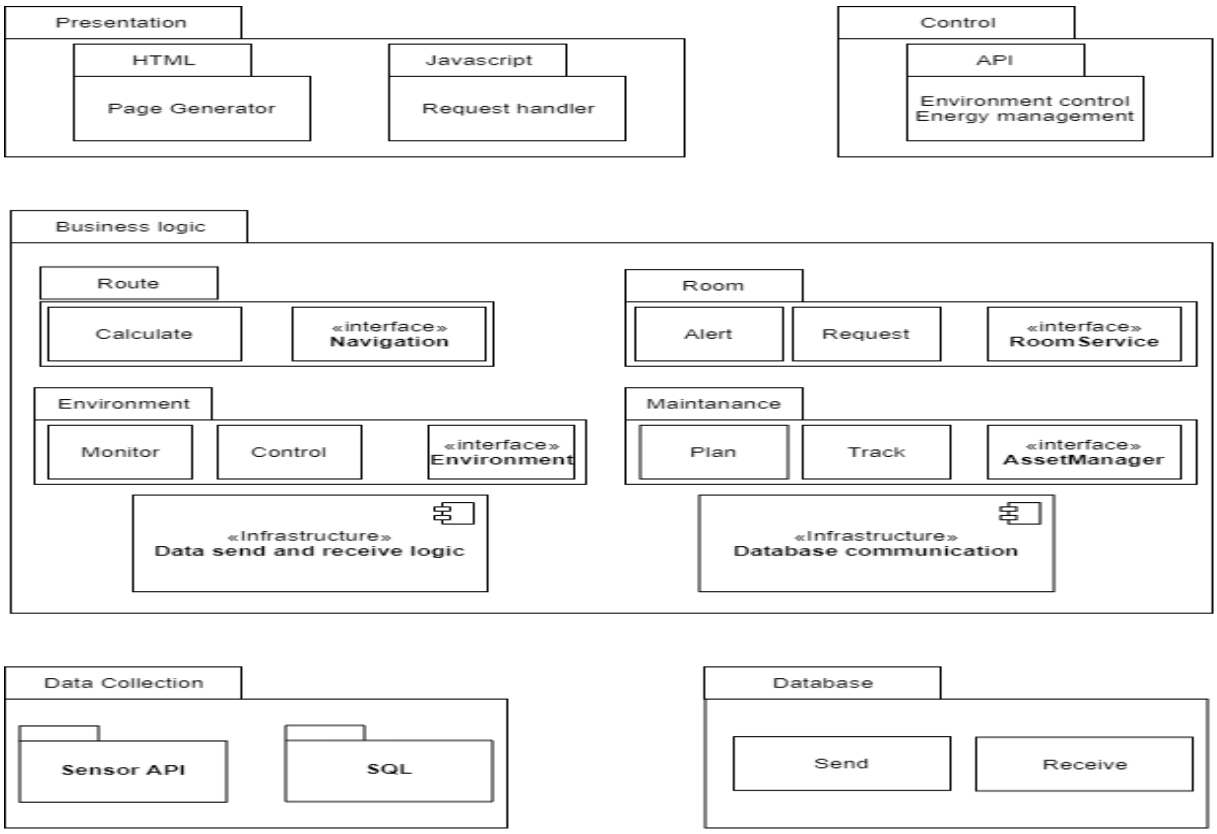
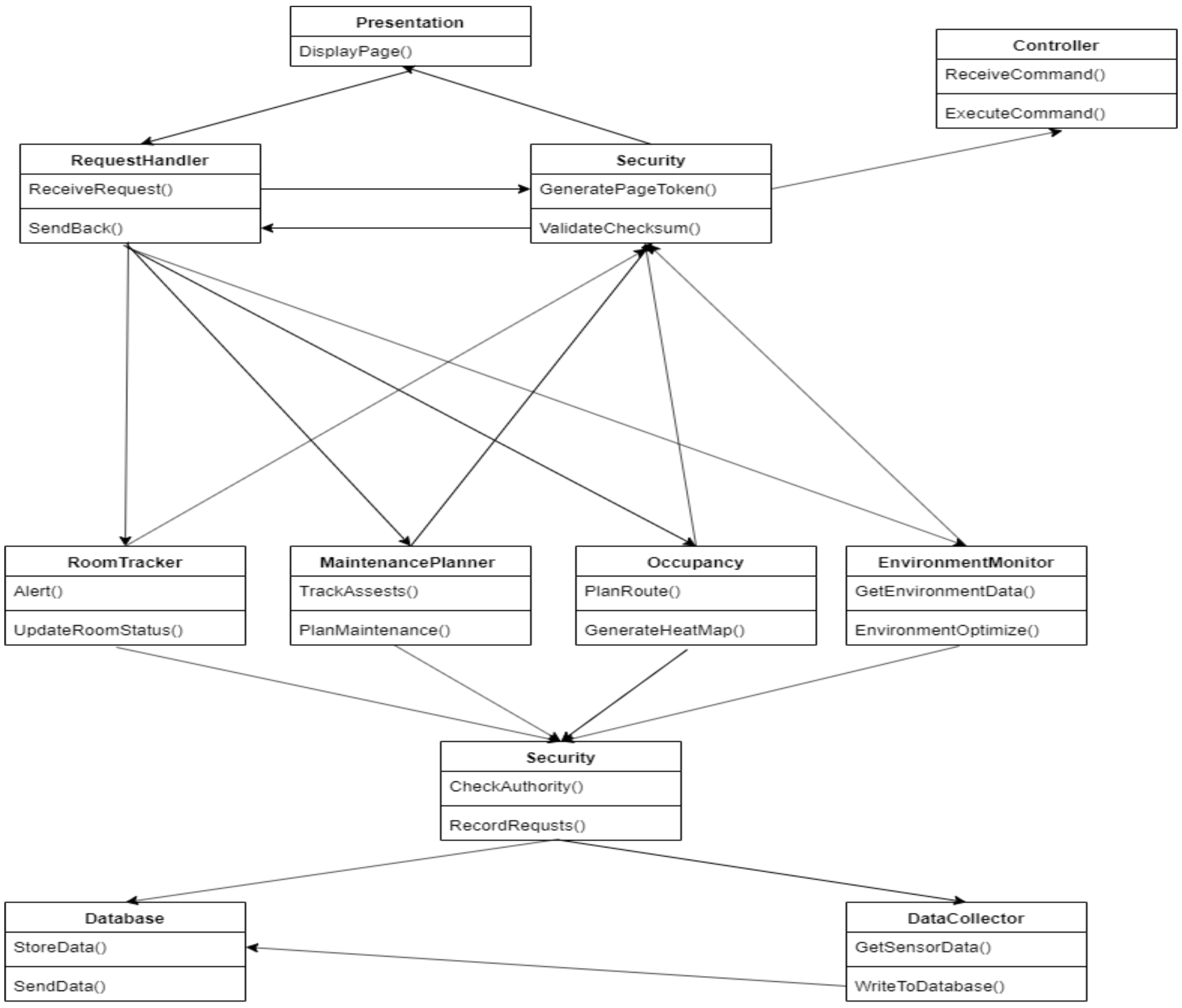
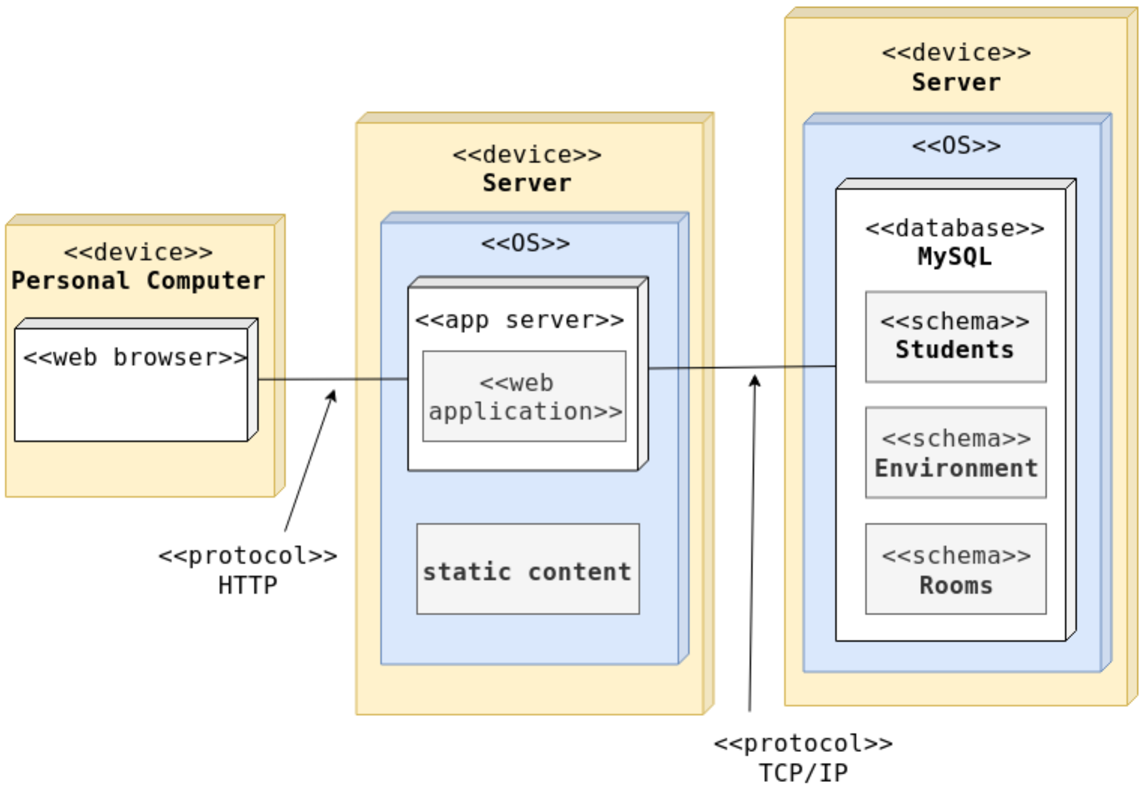


Figure: Logical view of different layers

Figure below describes logical view by different classes. Presentation and controller's class are similar, receive data and execute them. Request handler contain classes to receive requests from presentation layer and send them to other functionality classes. Higher level security class responsible for generate page token for presentation layer to prevent CSRF attack and validate checksum of received data to ensure integrity of system. Then several functionalities class deal with different requirements. Lower level security class check authority to activities access sensors and read database, record each request at same time. Database class send and receive data from and to different classes and Data collector class collects data from sensor and writes them to database.


Presentation
DisplayPage()
Controller
ReceiveCommand()
ExecuteCommand()
RequestHandler
ReceiveRequest()
SendBack()
Security
GeneratePageToken()
ValidateChecksum()
RoomTracker
Alert()
UpdateRoomStatus()
MaintenancePlanner
TrackAssests()
PlanMaintenance()
Occupancy
PlanRoute()
GenerateHeatMap()
EnvironmentMonitor
GetEnvironmentData()
EnvironmentOptimize()
Security
CheckAuthority()
RecordRequsts()
Database
StoreData()
SendData()
DataCollector
GetSensorData()
WriteToDatabase()

Physical view



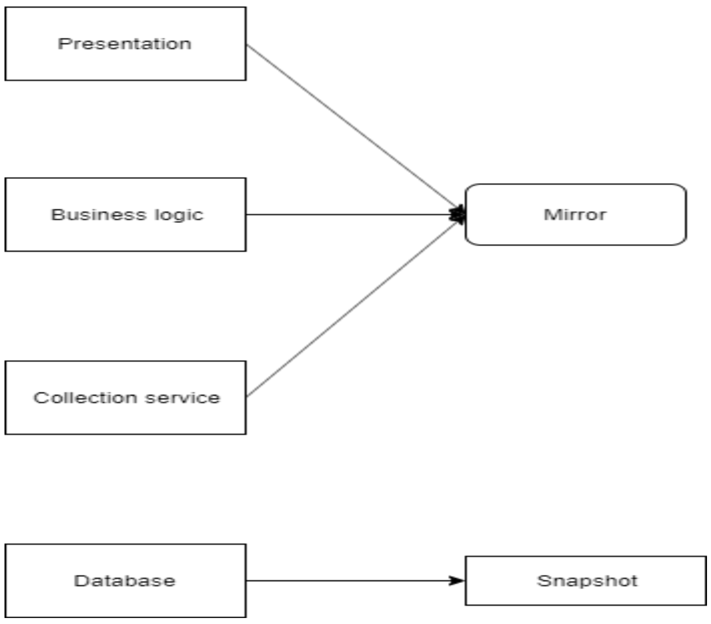
The system will be deployed as a web application hosted on an application server. The DBMS and database will be hosted on a database server. The application server communicates with the database server via TCP/IP. The user will use a web browser on their PC to connect to the application server via HTTP. The application server will serve static and dynamic content to the user, communicating with the database server when data is required for necessary calculations.

Process view

Our system will be design with a layered architecture, it will consist of four layers: presentation, Business Logic and Data collection layer. The presentation layer will display graphics, deliver content to user's and get requests from user's activities. The business logic layer receives requests from presentation layer, request data from database layer or data collection layer, process it, and supplies data to presentation layer. The data collection layer either collect on-demand data directly from sensors or regularly send data to database and the database layer will receive and store collected data, supplies data to business logic layer when needed. The use of this layered style has many advantages namely availability, scalability and security.

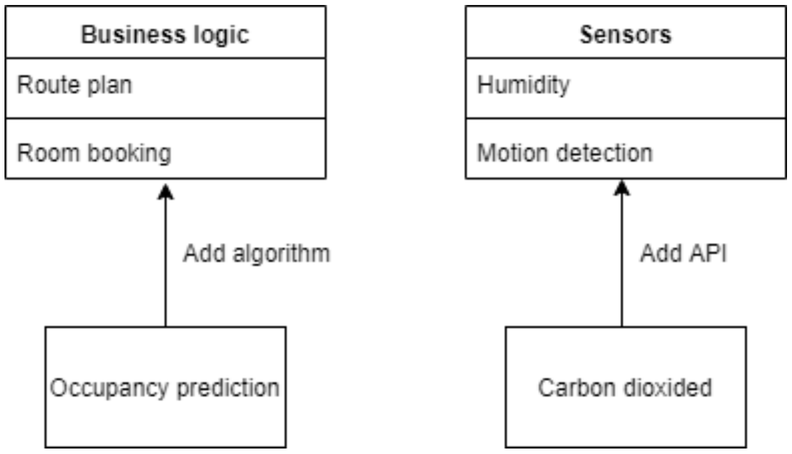
Availability

The layered architecture will achieve high availability. In having a separation of layers we could cache the presentation layer to prevent pages from displaying errors while there is a fault or failure in one of the other layers. A non-interaction page can still be displayed. If there are no changes to a specific part before the error, that part can be ignored, and the system can recover more efficiently. To back up the layered architecture is straightforward, we can set a mirror layer once the current layer is down, and a snapshot of the data storage and collection layer can also be set regularly. It is simplistic to monitor the running of each layer; it is fast and flexible for this architecture to recover from a fault.



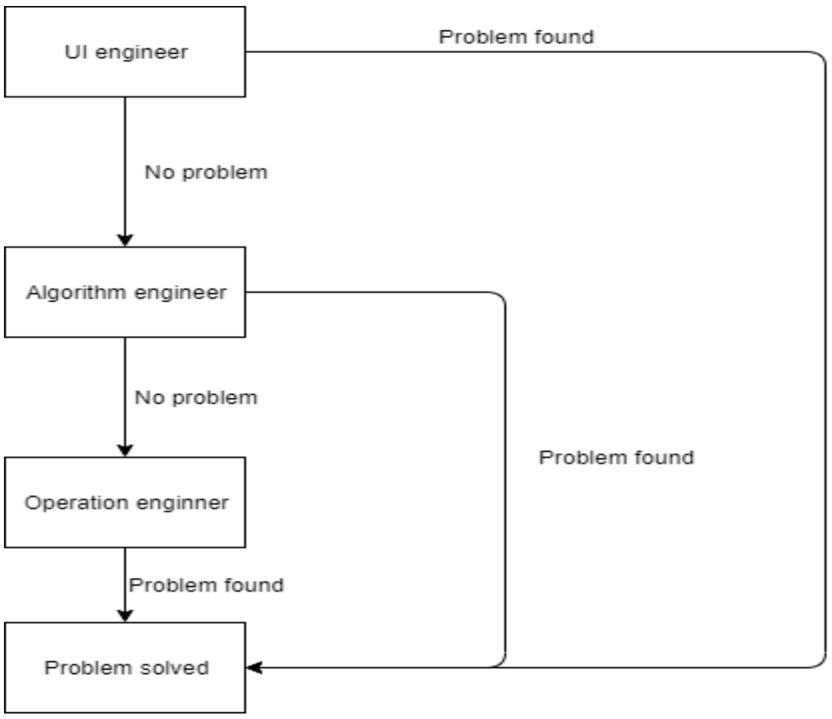
Scalability

The layered architecture for our system is a high cohesion and low coupling. User interface and low level service are independent in some aspects, so components can be reused easily. If we want to change the layout of the UI we don't need to change our business logic and if we want to add more functionality to sour system, whether it is software, machine learning algorithim, or hardware like advanced sensors by deploying layered architecture, some layers are replaceable, new processes such as APIs and algorithms can be easily integrated on our system, the user interface can also be updated if needed.



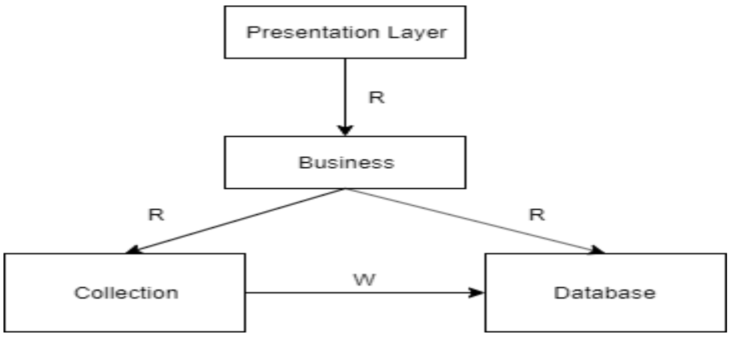
Maintainability

The different layers of the system are independent, so inner relationships and outer dependencies is clear to view, at the same time, encapsulated layers and class would be more efficient to distribute the maintenance of the work to different engineers or teams. Maintenance work can be performed in either layer without affecting others, in some extreme situations, some layers even replaceable without large amount of work. When a new feature wants to be added to our system it won’t require all the entire team to implement the feature.

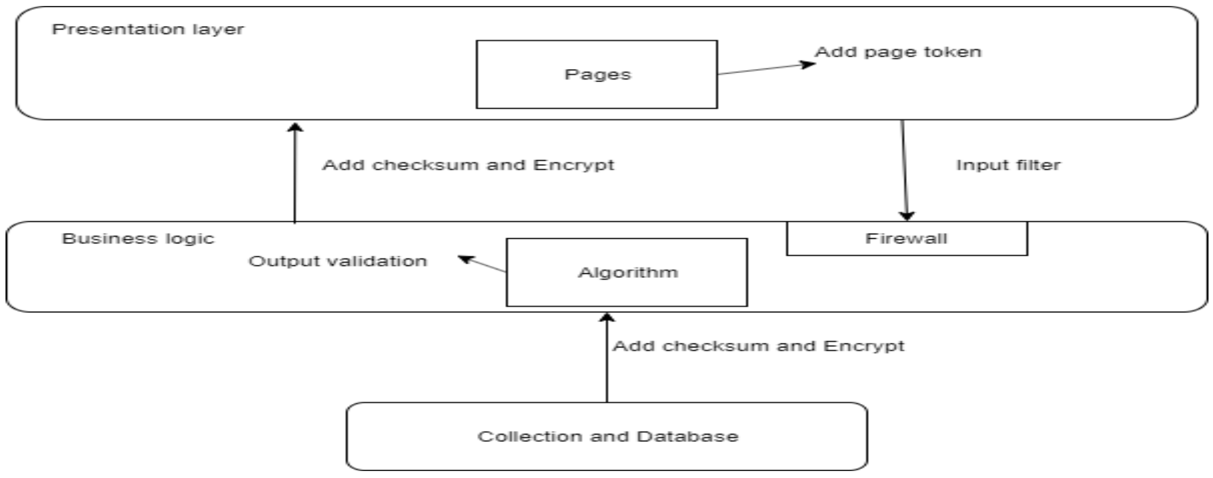


Security

Layer architecture allows for different authorization levels for different layers. Presentation only need authority to read data after business logic processing; Business layer need read authority of collection layer and database layer; Collection layer need to write to database system.

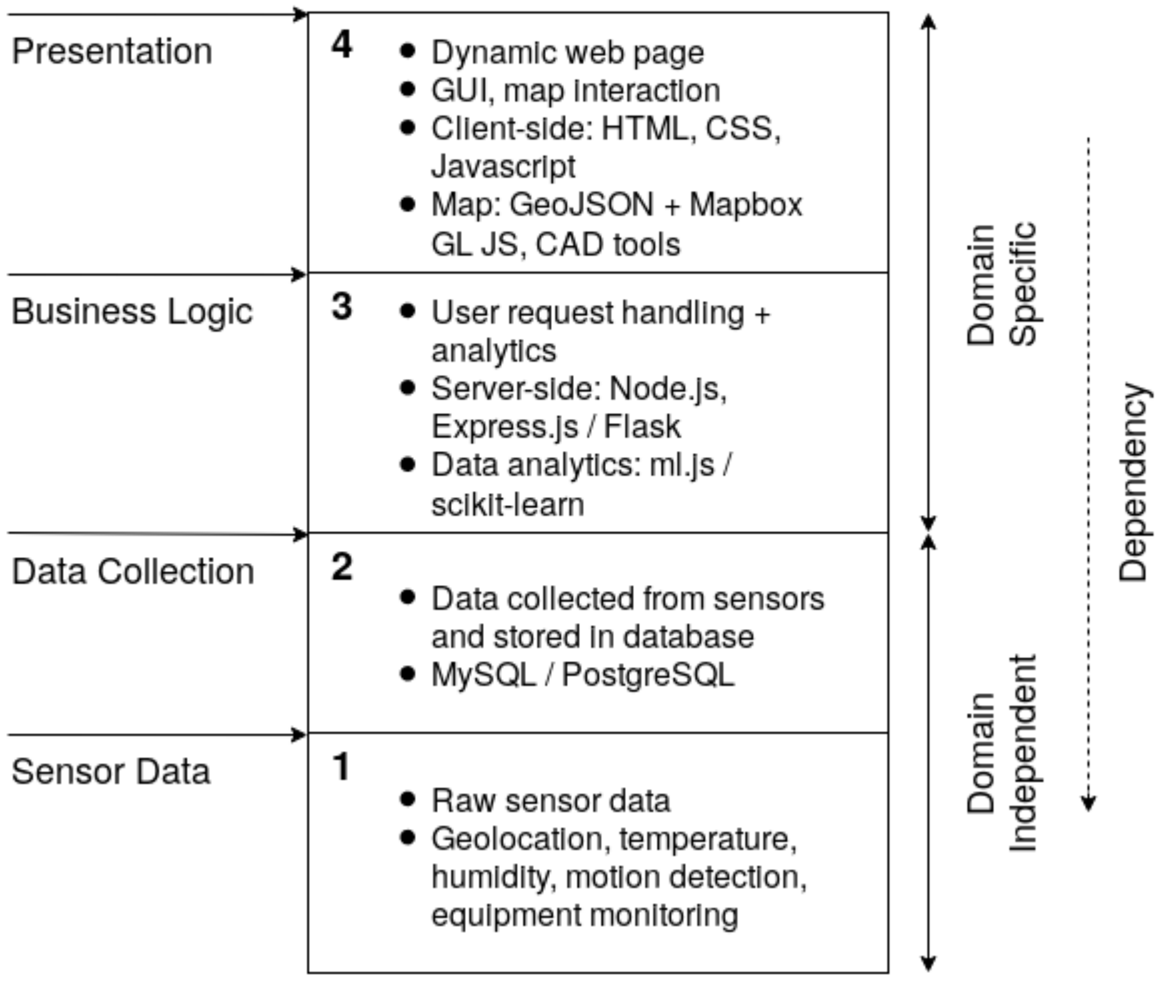


This matches the requirement of least privilege rule which is useful to prevent SQL injection. In layered architecture it is also quite easy to perform per-layer security procedures to enhance the security in each layer. For presentation layer, we can add page token to each page to avoid CSRF attack, for other layers we can add double-sides filter or firewall to each layer to ensure the source, destination, queries and contents are as expected to ensure integrity of whole system. Additional fields like checksum can be added to achieve non-reputation.



At the same time, after considering about the performance, our system doesn't need to care too much about performance because for our systems, layers just need to pass data instead of process, data process usually finished before sending, and there's not too much unnecessary work of our system except the collected data need to fill into database, most of system's working time is according to a 'need-given' pattern, only generate results when needed.

Development View


Presentation
4
Dynamic web page
GUI, map interaction
Client-side: HTML, CSS, Javascript
Map: GeoJSON + Mapbox GL JS, CAD tools
Business Logic
3
User request handling + analytics
Server-side: Node.js, Express.js / Flask
Data analytics: ml.js / scikit-learn
Data Collection
2
Data collected from sensors and stored in database
MySQL / PostgreSQL
Sensor Data
1
Raw sensor data
Geolocation, temperature, humidity, motion detection, equipment monitoring
Domain Specific
Domain Independent
Dependency

The map will be defined using geospatial data in GeoJSON format alongside Mapbox GL JS, a JavaScript library that uses WebGL to render interactive maps. CAD tools (such as AutoCAD) may be used for designing the map. The CAD files may be converted to the GeoJSON format using the ogr2ogr tool included in the GDAL library.

Express.js or Flask may be used as the web framework, for JavaScript and Python respectively, with ml.js or Scikit-learn being used for data analytics. ml.js is an interface to TensorFlow.js, a JavaScript library used for developing machine learning models, whereas Scikit-learn and Kera's are Python libraries which may be used for the same purpose.

We will use an agile-based approach to software development to allow for flexibility in the development process. We will need to be aware of which proposed features will be feasible to implement in our given time, alongside which features to prioritise so that we can ensure that a complete system with all the essential components can be ready in time. Regular meetings and communication will facilitate this.

Ideally, the workload should be split evenly so that each member contributes to a substantial portion of the system. We will use an online Kanban board (such as Trello) for managing the workload.
We will use git for version control, with all code being uploaded to a shared GitLab or GitHub repository. This would help us easily resolve work conflicts, as well as allow us to revert to older versions of the system if a mistake is made in development.

Conclusion

This report has outlined the methodology used to design a system to enhance the experience in using the computer science building. A design thinking model was employed to ideate the solution. Once the idea had been formulated the 4+1 view model was adopted to design how the software would be realised. A detailed analysis of each view has been given. This document will be used as a roadmap for the development of the system and a measure of validation.

Appendix 1

Table 1: Partition of work completed

Introduction	Mark Friel
Design Thinking	Mark Friel
Scenarios	Mark Friel
Logical View	Baibing Ji
Physical View	Conchur McCullough
Process View	Baibing Ji
Development View	Conchur McCullough