# Software Projects Milestone 5

## System Requirements & Technical Specification

## **Group Number:** 20

## **Concept Name:** Garden Assistant App

**System Requirements & Technical Specification**

This summarises and brings together all the previous sections into a specification that should be fully expanded in the appendix with the following points, in providing what is known as the System Requirements Specification (SRS). This collects various information that you have previously agreed and worked on, such as the UML diagrams, prototypes, technical architecture etc.

**Contents:**

1. **Purpose**

To assist gardeners in planning and maintaining small to medium sized gardens, by improving their capabilities in maintenance management, tracking the growth process of plants, and through this help people to have a closer connection to nature and plants.

1. **Scope**
   1. **Functional requirements**:

**End Users:**

* Ability to select plants and add them to canvas/tables
* Ability to set reminders to the calendar
* Ability to set the orientation of a garden’s floor plan
* Ability to view real-time VR if a camera is detected
* Ability to upload an image of a garden’s floor if no camera is detected
* Ability to view weather alerts on the calendar
* Ability to view plant information on the database page
* Ability to keep track of plant growth progress through the calendar, in such a way that shows the key dates for each plant
* Ability to keep track of plant growth progress through coloured tags on tables
* Ability to view articles and post comments on the forum. This will not be part of our MVP, however will be included in our roadmap as we will not be able to implement this in the time scale of this project, hence not part of the backlog.

**Administrators:**

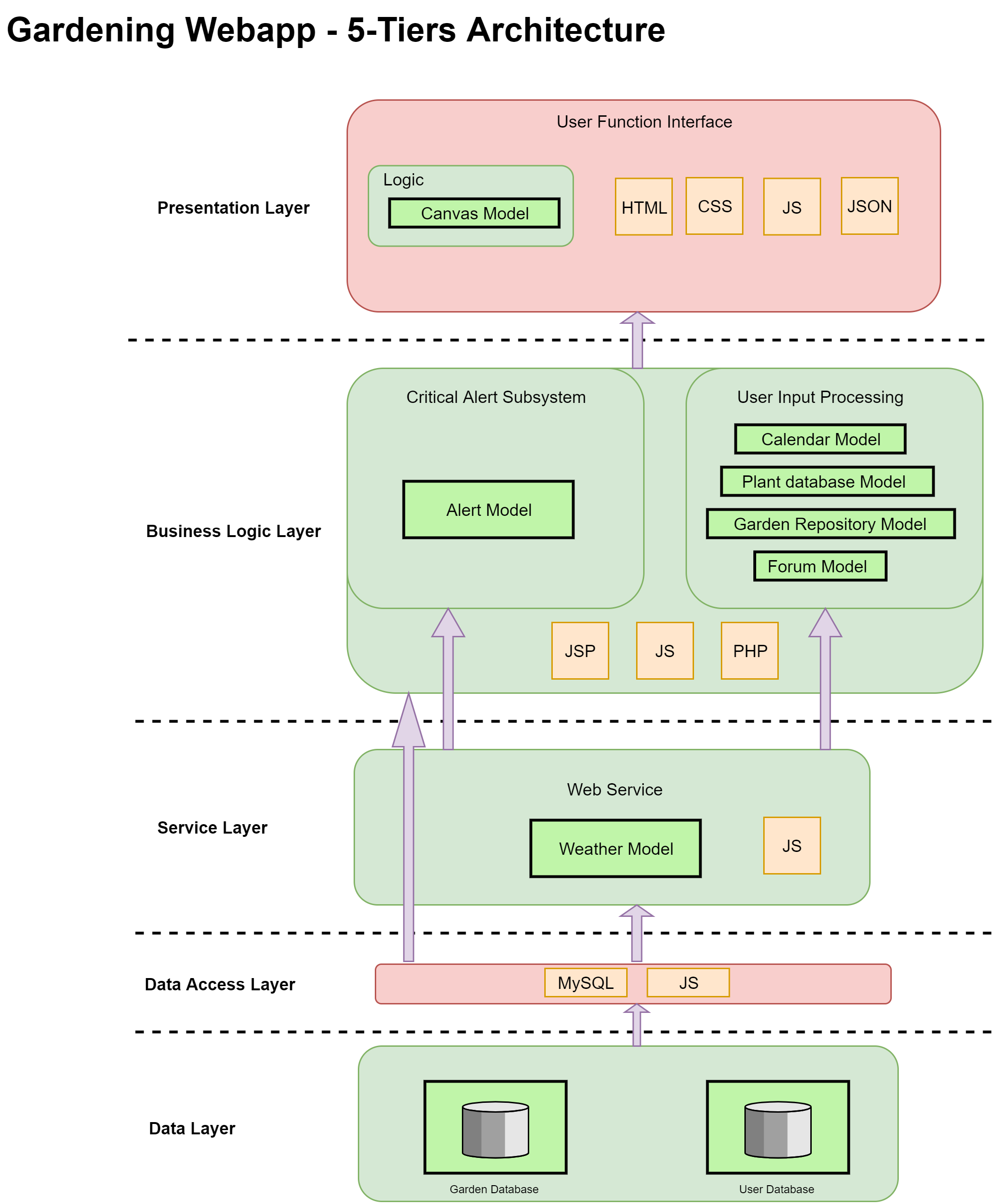
* Ability to delete users’ accounts
* Ability to update plants information in the database
* Ability to reply to users’ posts in the forum
* Ability to delete posts in the forum
* Ability to send system alerts to users for any critical information

‘Administrators’ has been included here because the system needs the ability to add users, delete users, add plants etc. However, it is important to note that this is not the Admin on the end-user’ side, it is the back-end of the system (our team).

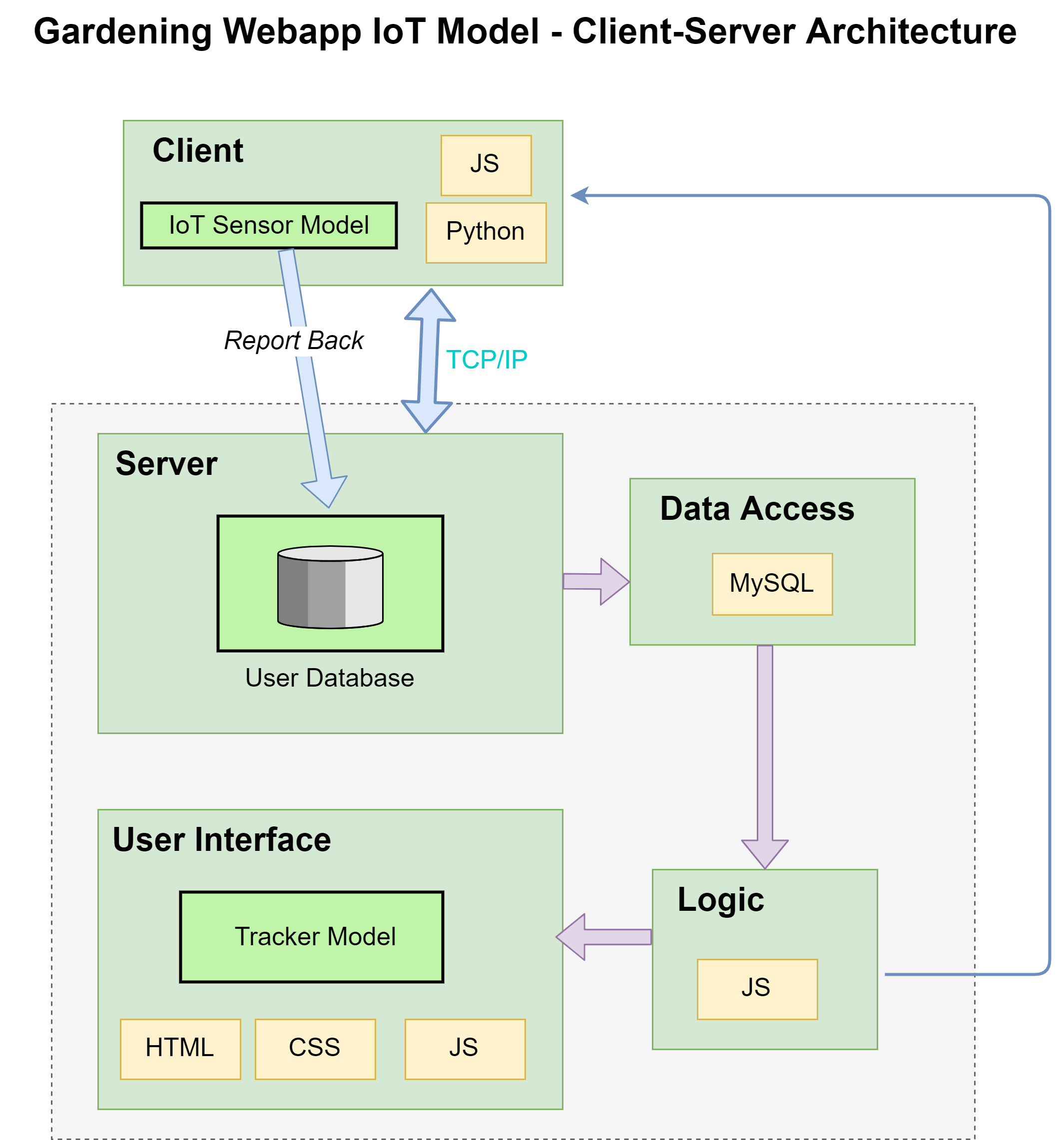
* 1. **Non-functional requirements**
* The system shall work on a desktop system, IOS/Android tablet system, with a web browser that has JS and WEBGL enabled
* The system shall work on any touch screen device, with a web browser that has JS and WEBGL enabled
* The number of plant models placed on a single canvas shall not exceed 100 at a time
* The service shall be online 99.9% of the time
* The Webapp shall provide larger icons/texts for elderly people
* The system shall work with mouse and keyboard, or, touch screen only
* The system shall have a fixed Schema
* The system shall work without a camera, or otherwise, if a camera is detected, the system shall ask users’ permission to make use of it

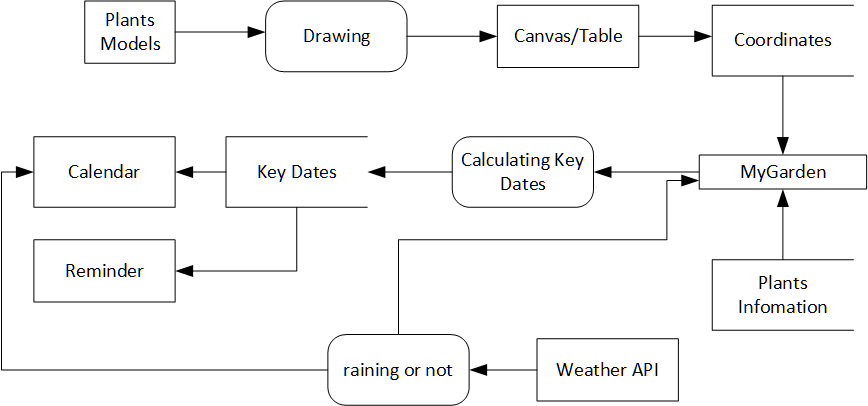
1. **System Overview**: Re-present the technical architecture for the product, including the data passing between functional components. Specify the technologies that will be used to realise the functions. Justify your choice of technologies with reasoned arguments for rejecting or retaining alternative technologies.

* **Technical Architecture for MVP:**



* **Reasons for 5-Tier Layered Architecture:**
  + Considering the deadline and time resources we have left, a layered architecture is much easier to implement
  + Easy to maintain, sockets for each layer can be simulated, and each layer can be tested on its own
  + The size of our team is five members, and each member can work individually on a separate task, to deliver the MVP on time
  + Functional requirements are decided, won’t make many changes in the near future
  + The nature of the system is not time-critical; it won’t be expansive if the server needs to be taken down for several hours, to upgrade the whole system.
* **Technical Architecture for IoT Additional Feature:**



* **Reasons for Client-Server Architecture:**
  + Considering the purpose of the feature is monitoring real-time conditions of a garden, the sensor will listen to ports on the server, and report back if a request is made.
  + Several clients will connect to a centre server
  + Client-side is expandable and easy to set up
* **Data Flow between Functional Components**
* **Technologies will be Used and Justification of Technical Choices**
  1. **Hardware:**

The system is based on the university’s Igor server that has already been deployed

* 1. **Software:**

The system design is based on major web browsers that have WEBGL and JS components.

* 1. **Languages:**

Presentation Layer:

* HTML
* CSS
* JavaScript
* JSON (for the calendar feature)

Business Logic Layer:

* PHP or JSP (but most likely PHP as JSP is another option)
* JavaScript

Service Layer:

* JavaScript

Data Access Layer:

* MySQL
* Node.js
  1. **APIs:**
     + Yahoo Weather API
  2. **Libraries:**
     + Three.js
     + Blippar.js
     + Passport.js
  3. **Network Protocols:**
     + TCP/IP
     + HTTP
     + HTTPS
     + FTP
  4. **Database:**
     + MySQL

**Reasons to choose MySQL over alternatives:**

* + - * The third-party databases on plants information that needs to be imported, have a similar structure to MySQL
      * There isn’t much real-time analysis in the system
      * The system needs multi-row transactions on plants information
      * The system needs a clear schema, such as the alert system needs to link to each plant
      * The data size of the system won’t grow very large
      * Security. There is some sensitive data passed from end-user, MySQL is a much safer solution compares to MongoDB.
      * The system needs to be built quickly with a time limit

1. **References**:

830-1993 IEEE Recommended Practice for Software Requirements Specifications. (n.d.). IEEE.

Blippar API. (n.d.). *Blippar*. [online] Available at: https://developer.blippar.com/portal/ar-api/home/ [Accessed 5 Dec. 2019].

Passport.js. (n.d.). *Passport.js*. [online] Available at: http://www.passportjs.org/ [Accessed 5 Dec. 2019].

Threejs.org. (n.d.). *three.js – JavaScript 3D library*. [online] Available at: https://threejs.org/ [Accessed 5 Dec. 2019].

Yahoo. (n.d.). *Yahoo Weather API*. [online] Available at: https://weather-ydn-yql.media.yahoo.com/forecastrss [Accessed 5 Dec. 2019].

1. **Definitions**:

|  |  |
| --- | --- |
| **API** | application programming interface |
| **critical information** | any important information such as server down |
| **CSS** | cascading style sheets |
| **end-users** | a person who ultimately uses or is intended to use a product ultimately |
| **FTP** | file transfer protocol |
| **functional components** | a function that perform certain functionalities |
| **functional requirements** | calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish |
| **growth progress** | a plant’s growth stage |
| **HTML** | hypertext mark-up language |
| **HTTP** | hypertext transfer protocol |
| **https** | hypertext transfer protocol secure |
| **Igor** | the department uses three servers; a back-end file-server (moya), which is used to hold the deployed web-content, and two front-end web-servers (computingws1 and Igor), which serve the various types of content to the world wide web. |
| **ios** | a mobile operating system created and developed by apple inc. |
| **IoT** | internet of things |
| **js** | JavaScript |
| **json** | JavaScript object notation |
| **jsp** | java server pages |
| **key dates** | dates are critical to each plant during the stage of growth |
| **layer** | the components are organised in horizontal layers |
| **MVP** | minimum viable product |
| **MySQL** | open-source relational database management system |
| **non-functional requirements** | a requirement that specifies criteria that can be used to judge the operation of a system |
| **PHP** | a server-side scripting language |
| **schema** | the organisation of data as a blueprint of how the database is constructed |
| **sockets** | one endpoint of a two-way communication link between two programs running on the network |
| **tcp/ip** | internet protocol suite |
| **VR** | [virtual reality](https://en.wikipedia.org/wiki/Virtual_reality) |
| **web browsers** | include but not limited to [google chrome](https://en.wikipedia.org/wiki/Google_Chrome), [Mozilla Firefox](https://en.wikipedia.org/wiki/Mozilla_Firefox), [internet explorer](https://en.wikipedia.org/wiki/Internet_Explorer), [safari](https://en.wikipedia.org/wiki/Safari_(web_browser)), [Microsoft Edge](https://en.wikipedia.org/wiki/Microsoft_Edge), [Opera](https://en.wikipedia.org/wiki/Opera_(web_browser)), [UC browser](https://en.wikipedia.org/wiki/UC_Browser), [Yandex browser](https://en.wikipedia.org/wiki/Yandex_Browser) |
| **WebGL** | a JavaScript API for rendering interactive 2d and 3d graphics within any compatible web browser |