First Draft (Words count is 582, which is 82 words exceeded)

**Technical Architecture (MVP)**

The architecture is based on the individual design of components in which data will be modified then pass through layers into a database. The components comprise the software architecture include:

* **Canvas Model:** Provides the user interface for visually planning, consists of sub-components:
  + **Plant Objects**: Holds 3D Objects
  + **Thumbnail Viewer**: Stores 2D thumbnail images
  + **Coordinates Table**: Stores coordinates of an object and its ID number as a long string
  + **Toolbox**: Initializes/modifies the canvas layout
* **Calendar Model:** To modify reminders, display alerts and events.
* **Plant Database Model:** To pass data from the database to the user interface
* **Garden Repository Model**: To represent the garden layout
* **Weather Model**: Weather updates
* **Alert Model:** Provides the notification/alert abilities

The system uses 5-tier layered architecture to incorporate the above models, see *Appendix A, Figure 1.* Also, the design is based on the following non-technical facts:

* The team size and time resources. A layered architecture is more comfortable to implement, and each model can be built separately
* Sockets can be simulated, and each layer can be tested on its own
* No functional change will be made
* The system is not time-critical

The system has an additional IoT feature will be implemented in future releases, see *Appendix, Figure 2.*

**System Requirements & Technical Specification**

Purpose:

To assist gardeners in planning and maintaining gardens, by improving their capabilities in maintenance, tracking, and encourages users to have a closer connection to nature.

Functional requirements:

The abilities of end-user:

* Drag and drop objects from lists onto canvas/grids
* Set reminders
* Set orientation
* View plants on garden floor in real-time with a camera
* Upload an image of the garden floor
* Receive alerts on the calendar and garden page
* View plant information
* View the critical dates of each plant

The abilities of the development team:

* Delete user’s accounts
* Update plant description
* Add new models
* Send alerts to users

Non-functional requirements

* The system shall work on desktop, IOS and Android systems, or touch-screen devices, with a web browser has JS and WEBGL implemented
* The number of models placed on canvas shall not exceed 100 at once
* The calendar function shall not slow down when data size increases
* The system shall provide larger icons/texts for elderly people
* The system shall work with mouse and keyboard, or touch screen only

System Overview

The system is designed based on the individual component in which the input data from users passes through several layers into a centre database, see *Appendix A, Figure 3* for the data path.

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. Technology:

Presentation Layer:

* HTML
* CSS
* JavaScript

Business Logic Layer:

* JavaScript
* Node.js

Service Layer:

* JavaScript
* JSON

Data Access Layer:

* MySQL
* Node.js
* JS

1. APIs:

* Yahoo Weather API

1. Libraries:

* Three.js
* Blippar.js
* Passport.js

1. Network Protocols:

* TCP/IP
* HTTP
* HTTPS
* FTP

1. Database:

* MySQL

The design of the database is based on the following facts:

* + - Third-party databases need to be imported have a similar structure to MySQL
    - No real-time analysis in the system
    - The system needs multi-row transactions
    - The system needs an explicit schema for the alert function to work
    - Data size will not grow huge
    - Sensitive data will be passed from end-user, MySQL is a safer solution
    - Time limit

**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].

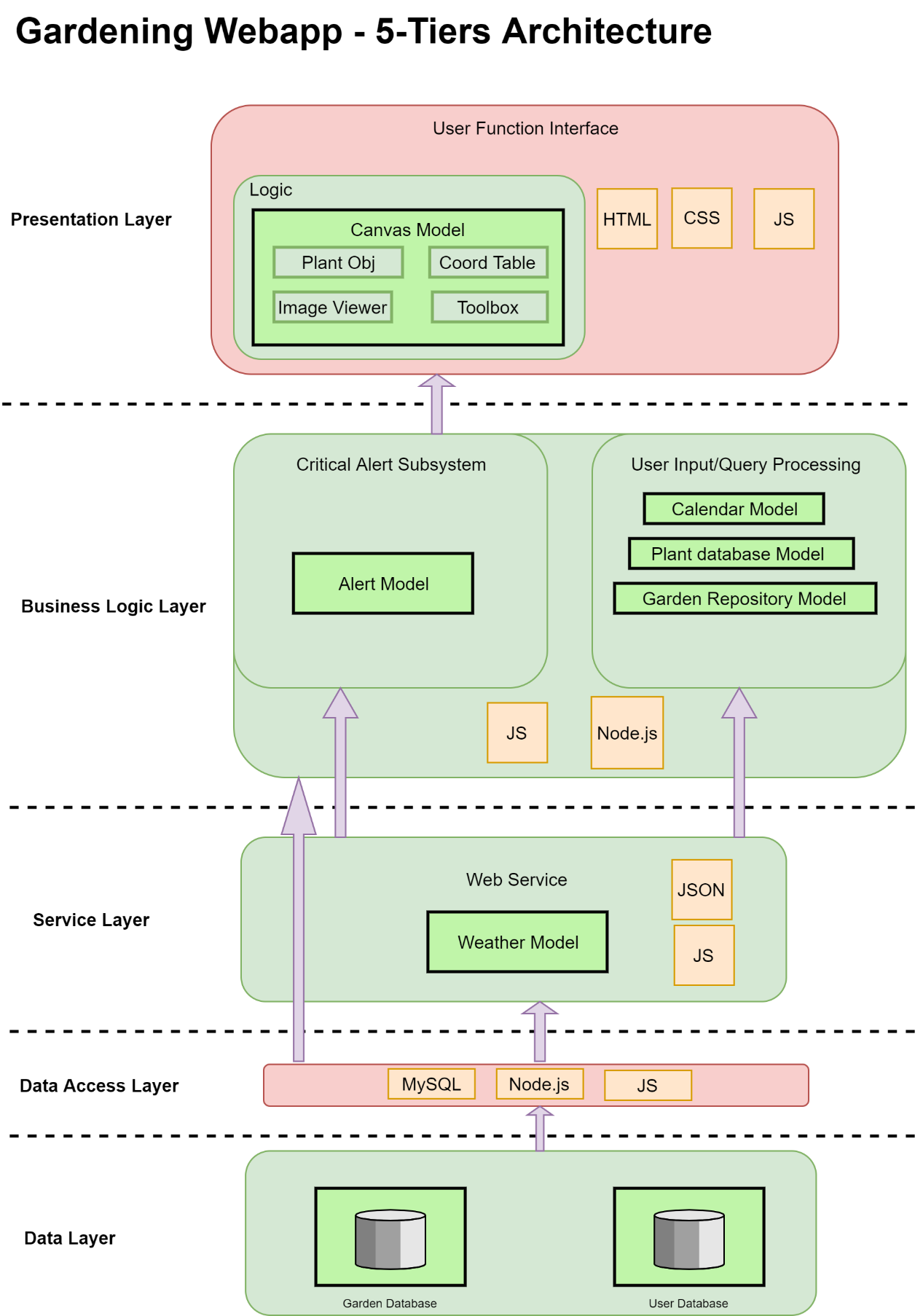
Passport.js. (n.d.). *Passport.js*. [online] Available at: http://www.passportjs.org/ [Accessed 5 Dec. 2019].

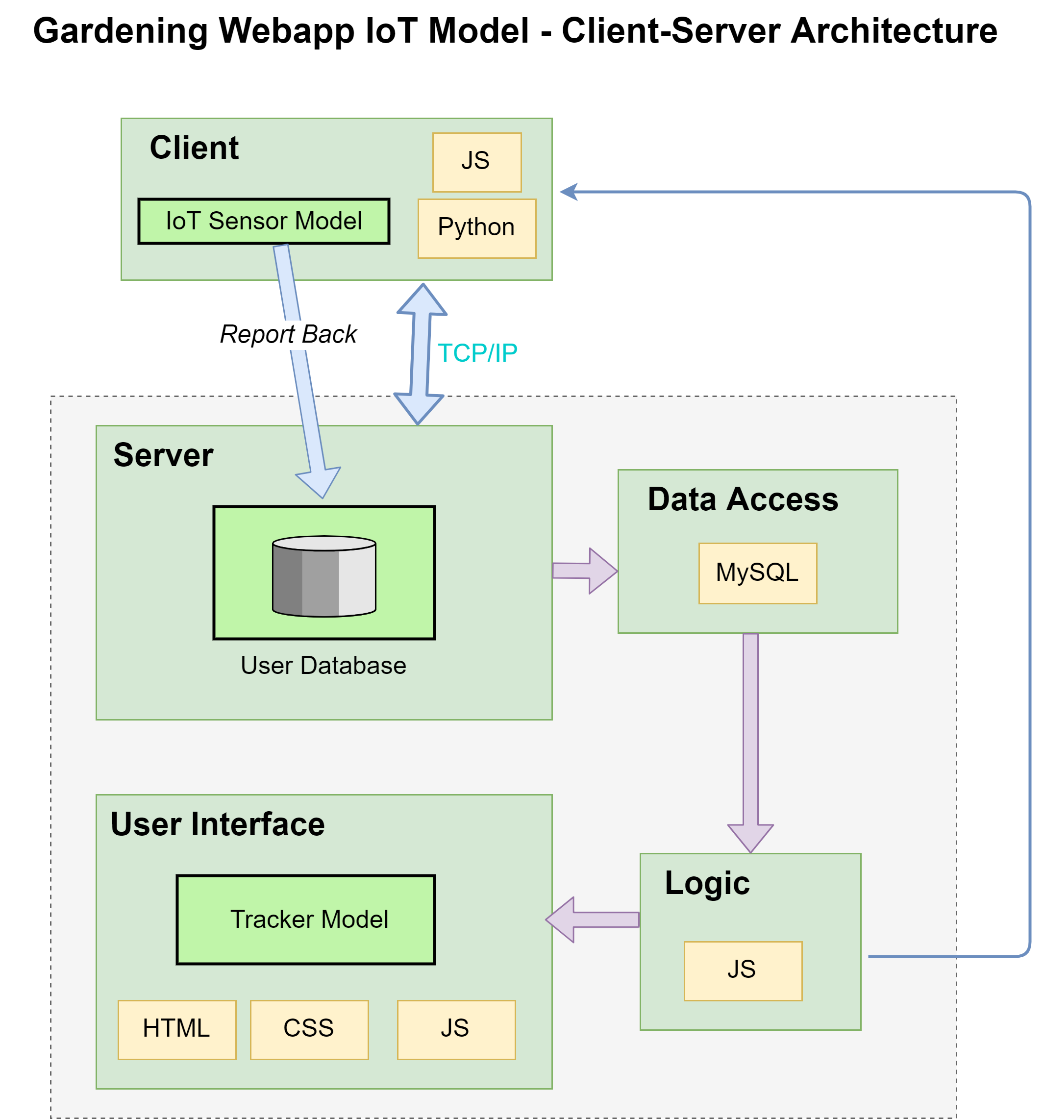
Threejs.org. (n.d.). *three.js – JavaScript 3D library*. [online] Available at: https://threejs.org/ [Accessed 5 Dec. 2019].

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

**Definitions**:

|  |  |
| --- | --- |
| **API** | application programming interface |
| **critical information** | any vital 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 |

**Appendix A**



* **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

