**Nail Salon Management System**

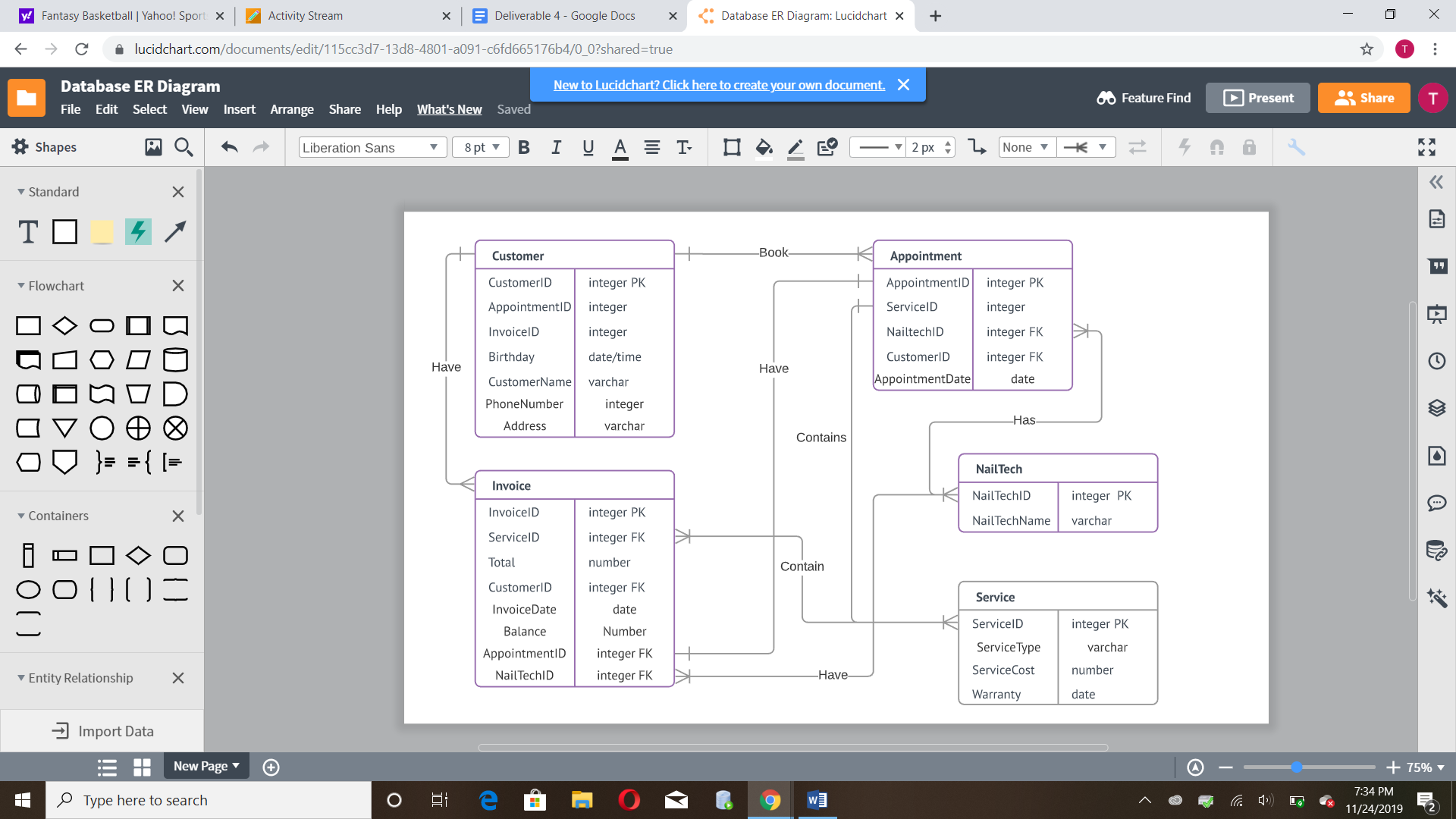
**Deliverable 4 Report: Design Phase**

Derek Luong, Thomas Zhang, Spencer Wilkins, Asad Khan, Hamza Mir, Salman Khan

IS 436

Structured System Analysis and Design (04.8331)

**ER Diagram**



**Entities and Processes:**

**Customer** - Customer entity has an appointment id and an invoice id corresponding to their invoice and appointment. They have personal information such as name, phone number, address and birthdate.

Book (To invoice)- Customers are able to book appointments after creating their accounts. They are able to book multiple appointments.

Have (to appointment) - Customers have an invoice that contains all the information about their appointment

**Invoice** - Invoice entity contains information for each appointment that the customer is booking. It contains the services, appointment, nail tech worker, customer name, date of appointment, and total balance.

Contain (to service) - invoice contains a service indicating what service that customer booked for their appointment. Invoice can have multiple services.

Have (to appointment) - invoice contains an appointment that the customer booked. Every invoice has to have only one appointment

Have (to nailtech) - invoice lists the nail tech employees that will be handing the services of the customer that date.

**Service** - Service entity identifies the service that the customer is booking during the appointment. It is identified by the service id, service type and cost.

Contain (to invoice) - services are contained in an invoice depending on what the customer booked.

Contains (to appointment) - services are contained in an appointment, similar details as invoice.

**Appointment** - The appointment entity represents the customer’s appointment that is being booked, identified by its appointment id and date. The appointment has a service id, nail tech id and customer id that represents the customer that is getting the appointment, the nail tech worker and the service being provided.

Book (to customer) - Every appointment has one customer that books that appointment

Have (to invoice) - Every appointment has one invoice that lists the appointment details

Contains (to service) - Appointment can contain one or more services that are specified by the customer

Has (to nail tech) - Appointment has a nail tech employee working during the appointment. Can have multiple employees working.

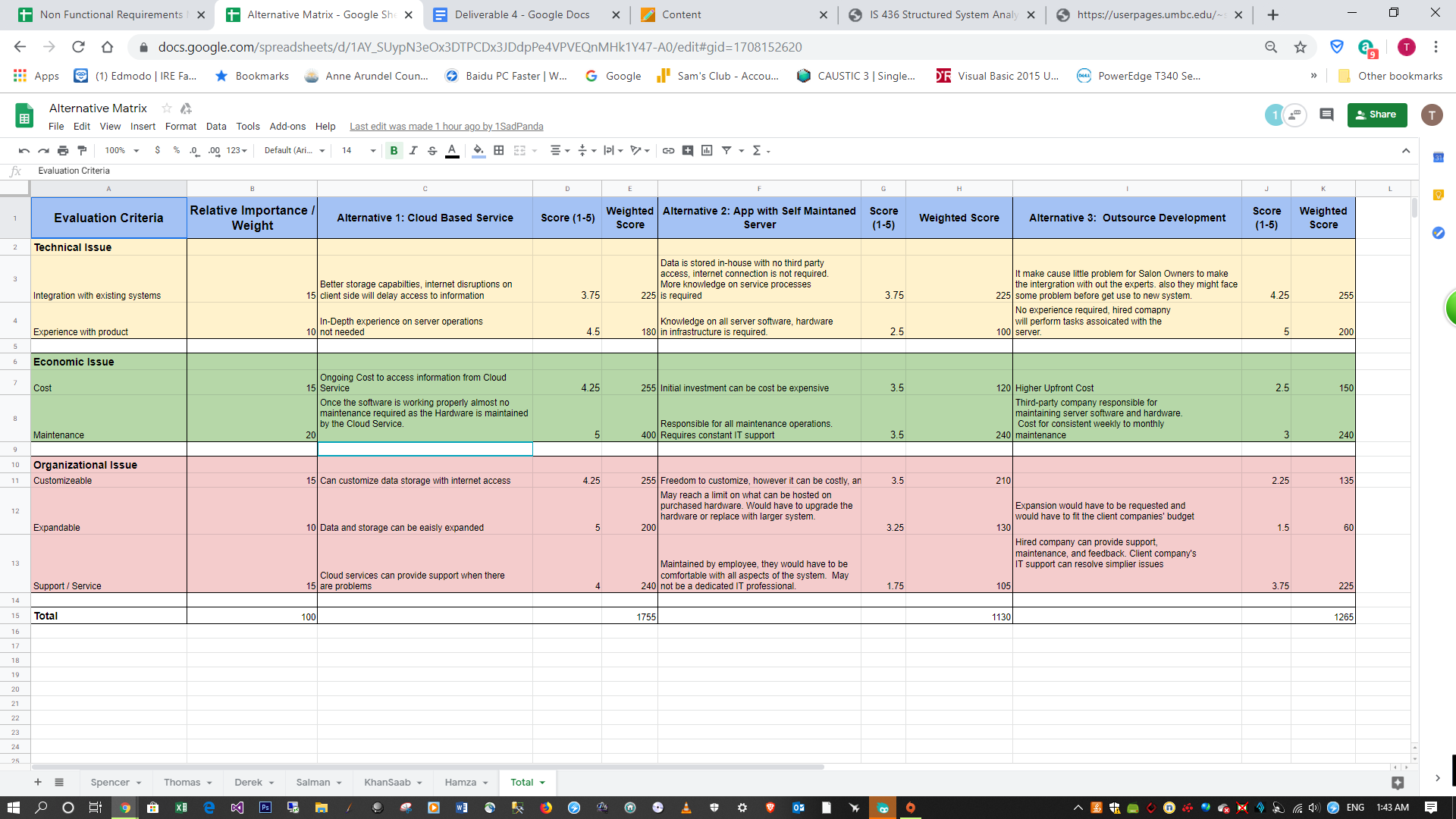
**NailTech** - The Nailtech entity represents the nail tech employee that is working during the appointment. They are identified by the nail tech id and the name of the employee.

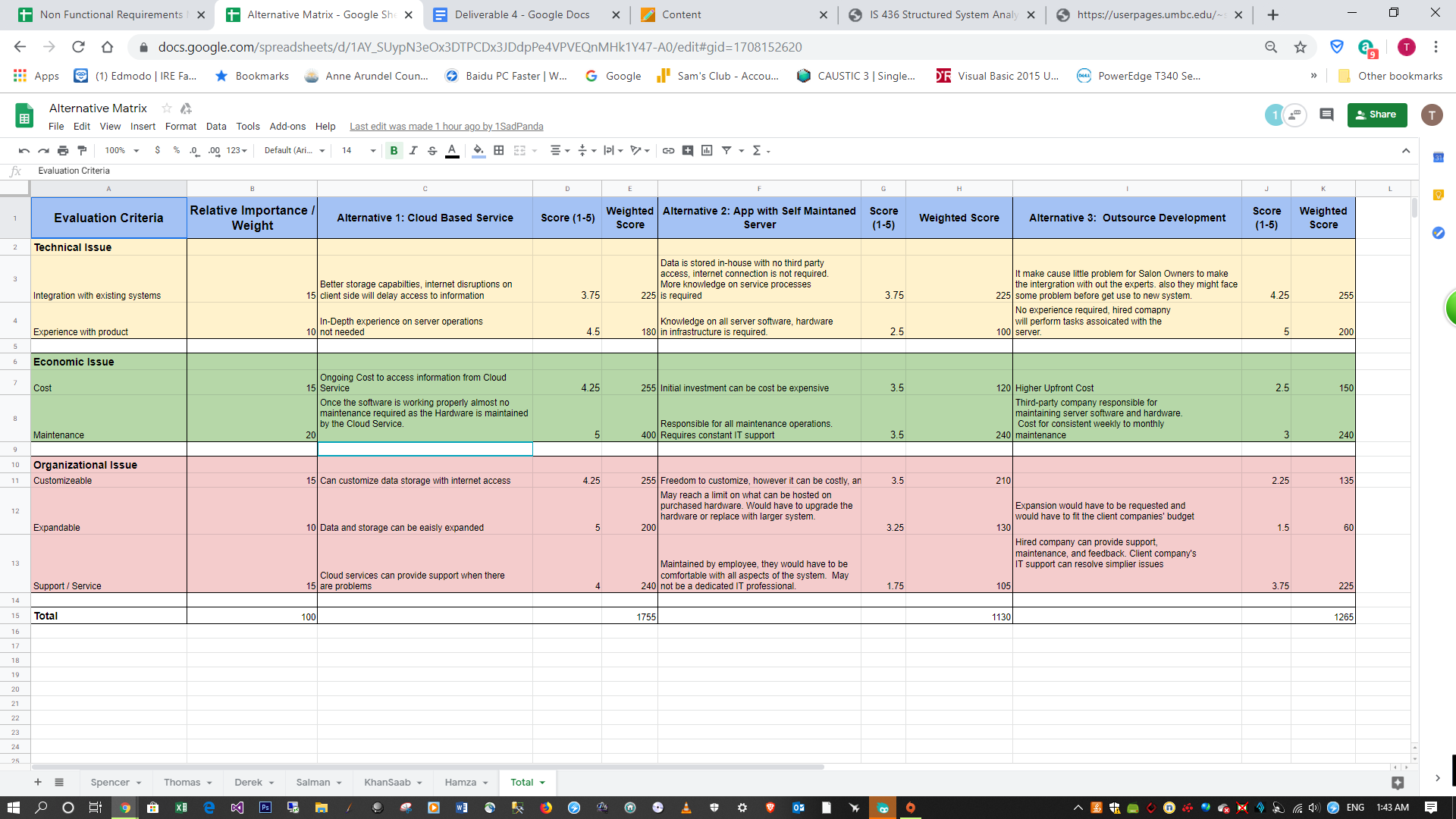
Have (to invoice) - A nail tech employee is put on invoice to specify who is working during the appointment date/time.

Has (to appointment) - Similar to invoice details above

<https://www.lucidchart.com/invitations/accept/05954fb3-85fd-4545-9d56-34935d0d9263>

**Total Team Matrix**





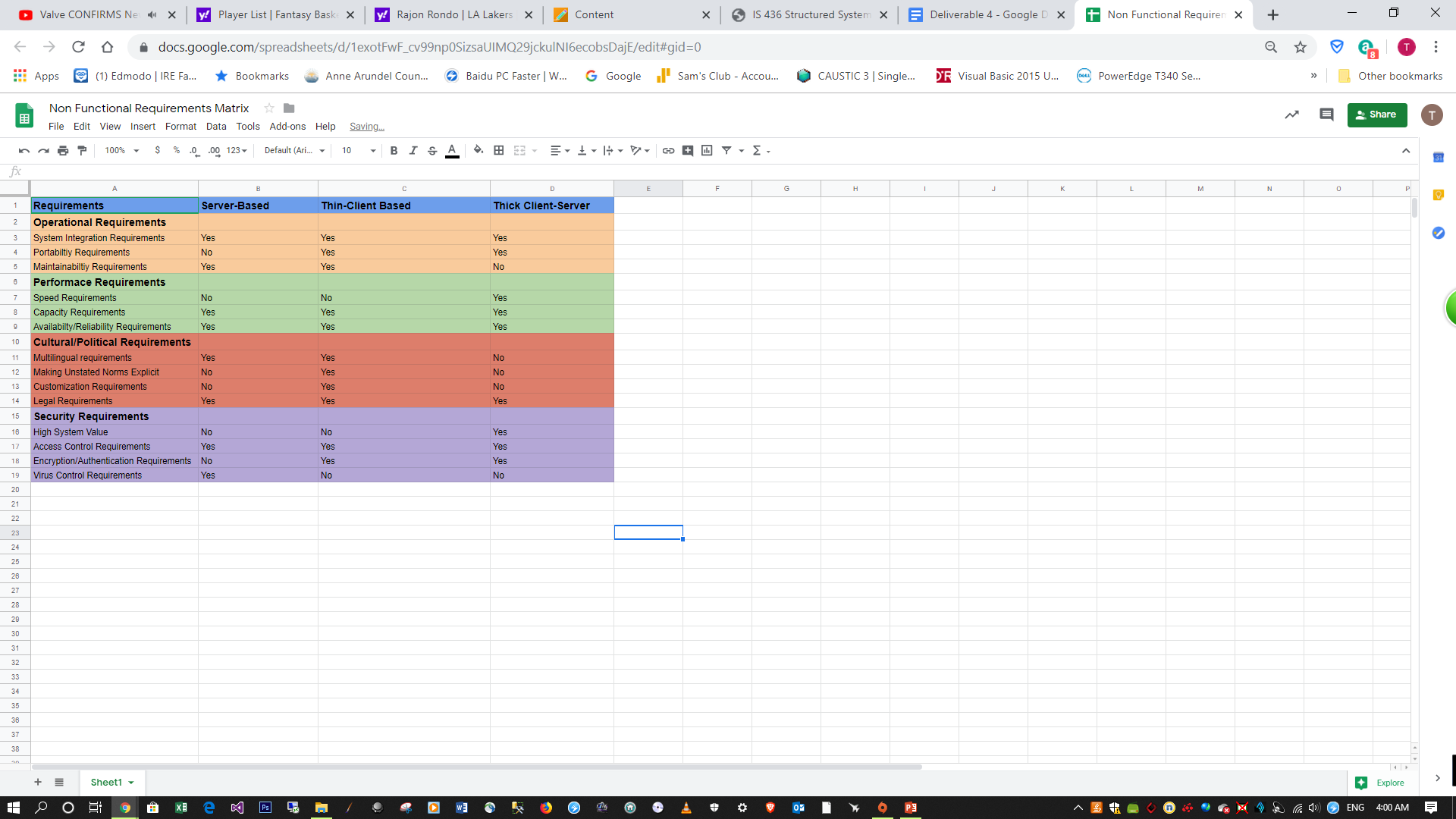
**Alternative Choice**

Alternative 1 was selected as the best choice due to its combination of easier maintainability, cheaper cost, easier expandability and customization and better storage capabilities. This choice best fits with our business that requires massive customer data storage and constant data flow. Although alternative 3 had comparable integration and maintainability requirements, requesting expansion and customizations as well as having a large upfront cost makes it a worse option. Alternative 2 was the worst-rated option due to its difficult implementation and extensive knowledge required in IT server hardware and software. It would have high maintenance, customization, expansion and initial cost compared to alternative 1 and had a unique issue in that storage space would be required to house the servers. It does have the benefit of not requiring internet connections and not worrying about any third parties but the benefits do not outweigh the disadvantages.

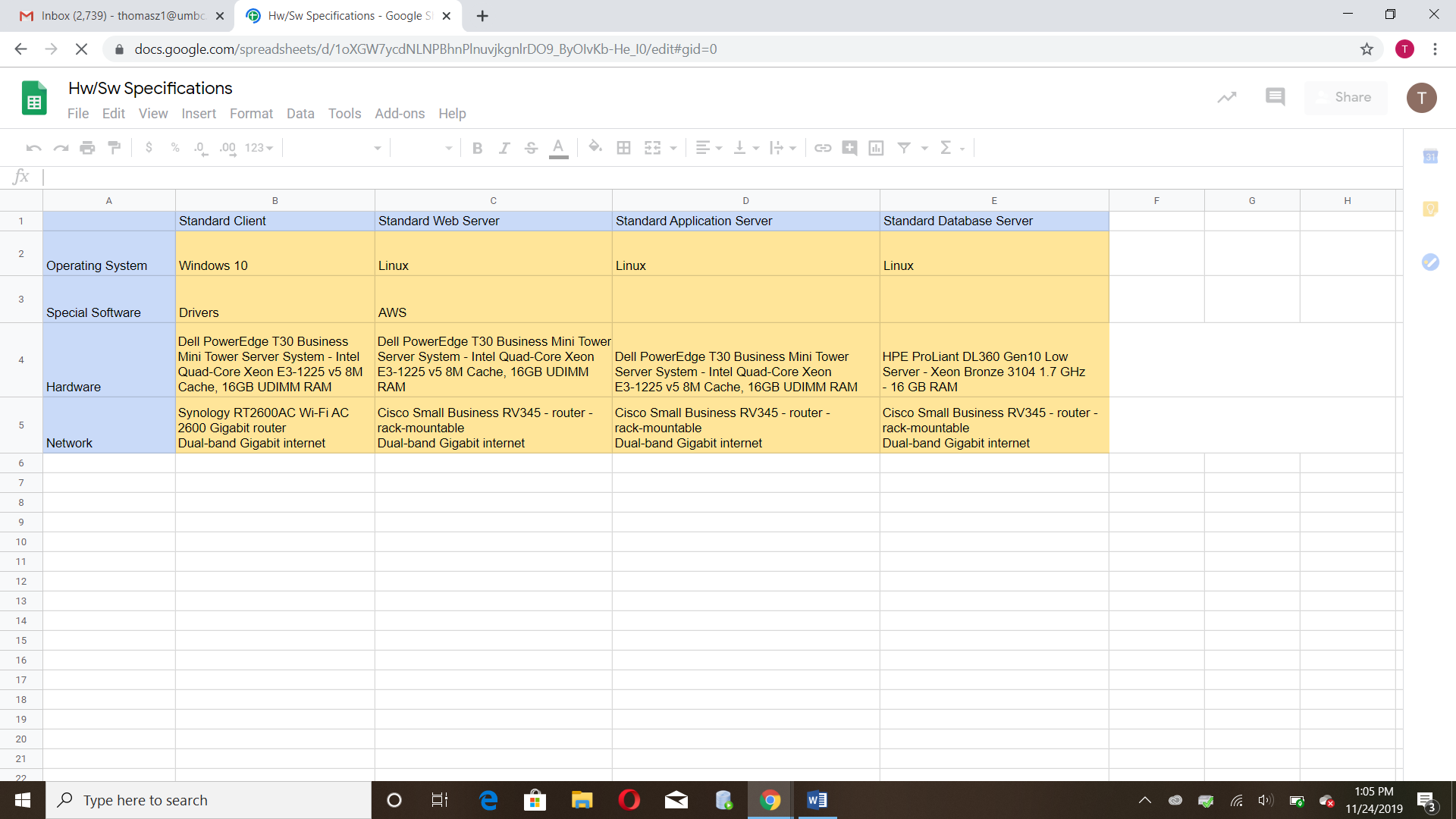
<https://docs.google.com/spreadsheets/d/1AY_SUypN3eOx3DTPCDx3JDdpPe4VPVEQnMHk1Y47-A0/edit?ts=5dd050d3#gid=0>

**Requirements Matrix**

We chose to use thin-client based architecture due to our choice of using a cloud based server to store our company data. Thin-client based architecture fits better with out design since our clients and employees will be accessing information from a web server. Account creation and access will be using web browsers either on their phones or computers. Availability, maintainability, system integration and capacity will be easily managed since everything will be stored on cloud and changed in storage capacity can be purchased through AWS systems. A downside is a stable internet connection is required for a client to access the server. A server-based system would require physical storage space that the business have combined with needed knowledge on hardware/software components by employees would not make this a good option for our business. A thick-client server would not be convenient for customers as they would have to download software to access our interface. Strong security and virus protection would be needed.

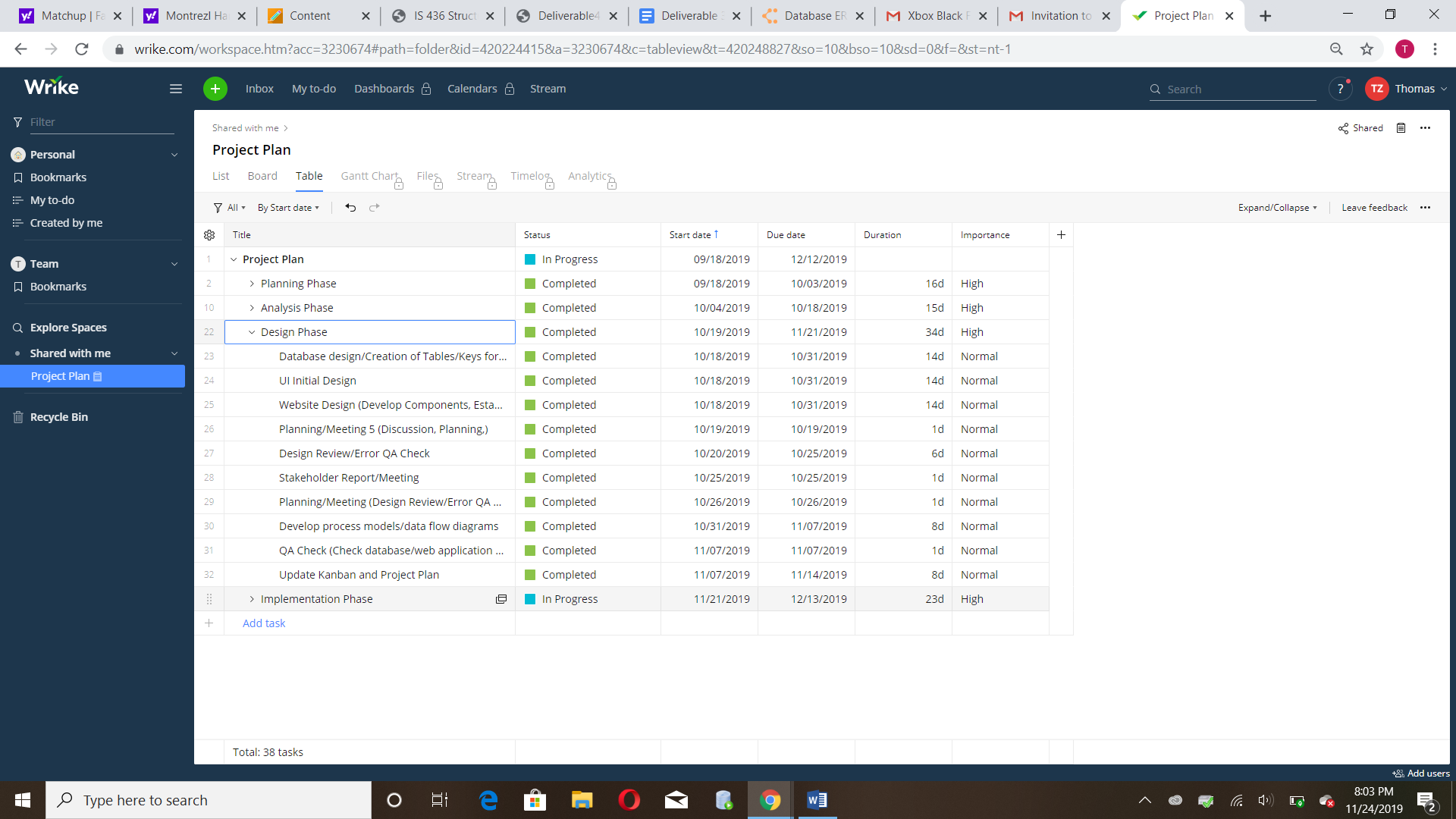


**Hardware/Software Specifications**

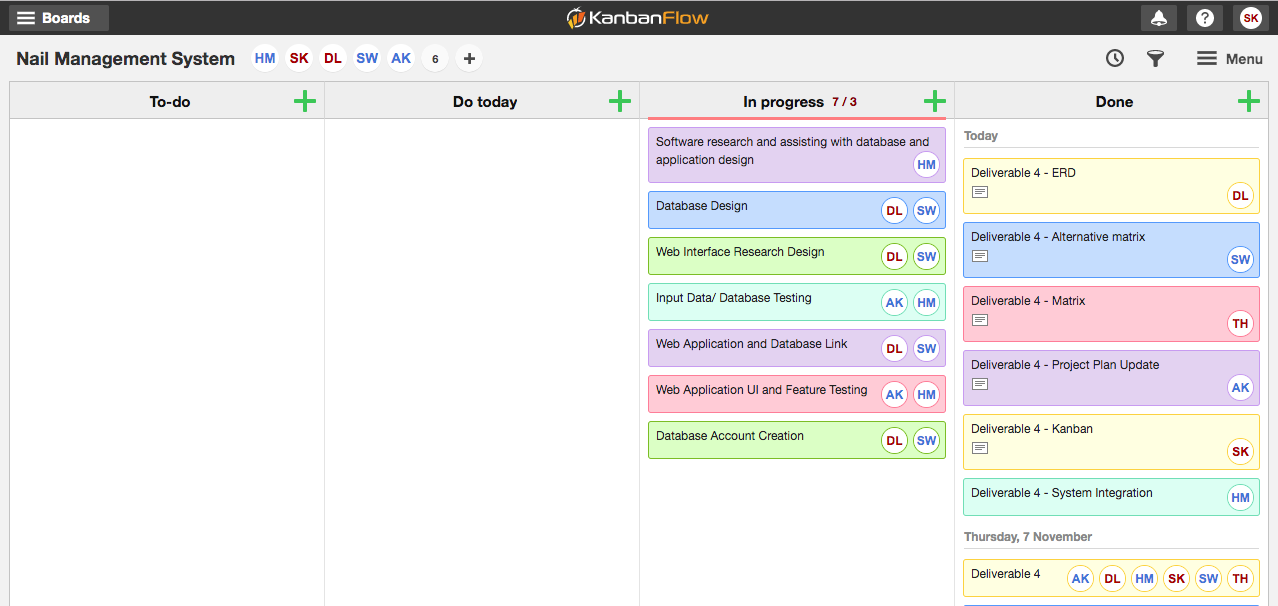


<https://docs.google.com/spreadsheets/d/1exotFwF_cv99np0SizsaUIMQ29jckulNI6ecobsDajE/edit#gid=0>

**Project Plan**



**Kanban**



**System Integration**

* Customer database information will be shared using third party cloud server with Paradise Pizza’s Restaurant.
* The account created can be used in both services.
* Using third party service will lessen security risks.