Nicholas Jurgens

TIM 158

5/9/17

TIM 158 Midterm

**Problem 1: Planning**

1. **Define the Problem**
   1. Create a plan and time-schedule for completing the final exam. Use an appropriate table to track how well you execute your schedule and make notes on any obstacles and problems you encounter.
2. **Create a Plan**
   1. **Step 1:** Clearly state the intent of the midterm.
   2. **Step 2:** Determine the design/development sub-tasks and activities.
   3. **Step 3:** Create a design/development activity matrix purpose to understand the dependencies between the sub-task.
   4. **Step 4:** Create a schedule of tasks using a GANTT chart.
   5. **Step 5:** Identify the “critical path” for the project using PERT chart.
   6. **Step 6:** Create a table and update it with obstacles and problems that occur throughout the project.
3. **Execute**
   1. **Step 1:** The intent of the final is to gain a better understanding of material. In order to do this I need to be able to complete all nine problems in a timely manner.
   2. **Step 2:**
      1. **A:** Reading
      2. **B:** Planning
      3. **C:** Dissecting Information Technology and Product for Uber
      4. **D:** Software Requirements for Uber
      5. **E:** Client-Server Software/Hardware Architecture Design for Uber
      6. **F:** Database Design for Uber
      7. **G:** Network Architecture for Uber
      8. **H:** IT Integration and Virtualization for Uber
      9. **I:** Excel Solver
      10. **J:** Execution of Your Plan
   3. **Step 3:** Activity Matrix

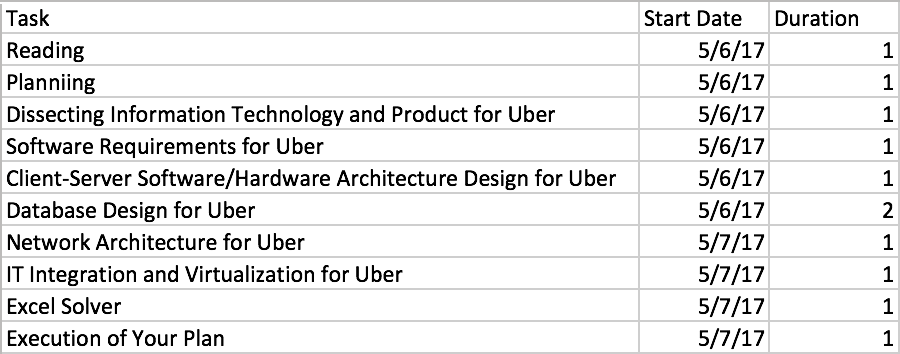
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** |
| **A** | **A** |  |  |  |  |  |  |  |  |  |
| **B** |  | **B** |  |  |  |  |  |  |  |  |
| **C** | **X** | **X** | **C** |  |  |  |  |  |  |  |
| **D** | **X** | **X** | **X** | **D** |  |  |  |  |  |  |
| **E** | **X** | **X** | **X** | **X** | **E** |  |  |  |  |  |
| **F** | **X** | **X** | **X** | **X** | **X** | **F** |  |  |  |  |
| **G** | **X** | **X** | **X** | **X** | **X** | **X** | **G** |  |  |  |
| **H** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **H** |  |  |
| **I** |  | **X** |  |  |  |  |  |  | **I** |  |
| **J** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **J** |

Notation:

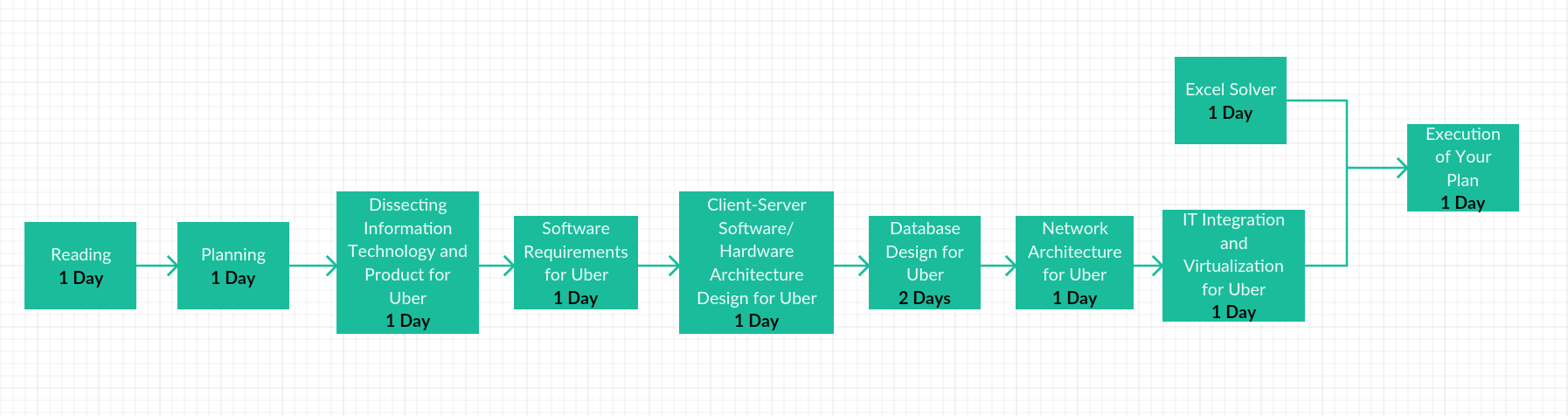
X = “depends on”

BxA = subtasks B depends on subtask A

* 1. **Step 4:** GANTT Schedule



* 1. **Step 5:** PERT Chart



* 1. **Step 6:**

|  |  |
| --- | --- |
| **Problems that occurred** | **Solutions** |
| 5/7/17 I visited my girlfriend’s father for his 50th birthday and it took more time away from the midterm than I thought. | Add more time for problem 7. |
| 5/7/17 Couldn’t finish problem 8 due to pushing back problem 7. | Push back problem 7. |
| 5/7/17 Finished later than I Thought I would. | Dedicate the next day to looking over my work. |

1. **Draw Conclusions**
   1. My plan was to have everything done by the 7th. However, I ran into problems and had to push some of the work back. This is due to underestimating how long each problem would take. I also didn’t take into account how long I would be working for my job. Next time I make a plan I will make sure to leave more room for error. This will help reduce any risk in surprises.

**Problem 2: Dissecting Information Technology and Product for Uber**

1. **Define the Problem**
   1. Describe the company: its vision, mission, products and services, customers, and financial performance.
   2. Based on above source and other suitable internet research, locate as much information as you can on the IT hardware and software infrastructure for Uber. Organize this information in the form of a FAST diagram.
2. **Create a Plan**
   1. **Step 1:** Describe the Company
      1. **Step 1.1:** Define the company’s vision.
      2. **Step 1.2:** Define the company’s mission.
      3. **Step 1.3:** List out the products and services of the company.
      4. **Step 1.4:** Describe the company’s customers
      5. **Step 1.5:** Analyze the financial performance of the company.
   2. **Step 2:** Perform research on Uber and create a FAST diagam.
      1. **Step 2.1:** First understand how the specific product works.
      2. **Step 2.2:** Make a list of the important subsystems and components that are relevant to the FAST diagram.
      3. **Step 2.3:** Make a list of the main or primary function and the key sub-functions of the product
      4. **Step 2.4:** Write down the main or primary function for the system
      5. **Step 2.5:** Organize the FAST diagram with the “WHYs” to the right and the “HOWs” to the left
      6. **Step 2.6:** Creating a FAST diagram for a complex problem is a “Trial and Error” process.
3. **Execute**
   1. **Step 1:** Company Description
      1. **Step 1.1:** Uber’s vision
         1. Uber’s vision for the future is smarter transportation with fewer cars and greater access; transportation that’s safer, cheaper, and more reliable; transportation that creates more job opportunities and higher incomes for drivers.
      2. **Step 1.2:** Uber’s Mission
         1. Uber’s mission statement is “make transportation as reliable as running water, everywhere, for everyone.”.
      3. **Step 1.3:** Products and services
         1. Transportation service
         2. Uber mobile application to track drivers
         3. Information provided for each driver
         4. Rating system for rides
      4. **Step 1.4:** Company’s Customers
         1. People who don’t own a car
         2. People who do not want to drive themselves to a party or function
         3. People who like to travel in style and want to be treated as a VIP
         4. People who want a cost-efficient cab at their drrstep
      5. **Step 1.5:** Financial Performance
         1. In 2016, Uber recorded a revenue of $6.5 billion on $20 billion bookings.
         2. Rapid growth came at a cost, Uber lost $2.8 billion in 2016.
         3. Was losing $1 billion a year in China. Had to sell that part of the business.
   2. **Step 2:**
      1. **Step 2.1:** How it works
         1. The process for Uber is:
            1. Request a ride from Uber
            2. Uber then asks where you are.
            3. Uber finds a nearby driver and provides you with an eta of the driver to pick you up.
            4. After the ride is over, Uber asks you to rate the ride.
            5. The cost of the ride is then automatically charged to your credit card.
         2. The technologies that Uber uses to make this work include the following:
            1. Geolocation

Identifying a device’s location: The Uber app for iOS uses the CoreLocation framework to locate a user’s device. It provides classes and protocols to configure and schedule location delivery and send location events to the server. IT also allows Uber to define geographic regions and monitor a device’s movements as it crosses defined boundaries.

Providing driving directions: Uber uses MapKit to display point-to-point directions on a map within the app. Registering the app as a routing app the makes directions available to the Maps app and all other mapping software on a user’s device. Android routes and directions are provided by the Google Maps Android API.

Integrating with mapping software: Uber implemented Google Maps for both iPhone and Android versions of their app. They buy mapping technology companies to solve their logistics issues.

* + - * 1. Push Notifications and SMS

Receive a notification when the driver accepts your request.

Receive a notification when the driver is less than a minute away.

Notifications will also be sent if the ride has been cancelled.

* + - * 1. Payment Integration

Uber uses a cashless system. Payments are accepted via debit or credit card, or a promo code.

Uber partners with Braintree to accept card payments.

Uber also uses PayPal’s Card.io service for credit card scanning on iOS.

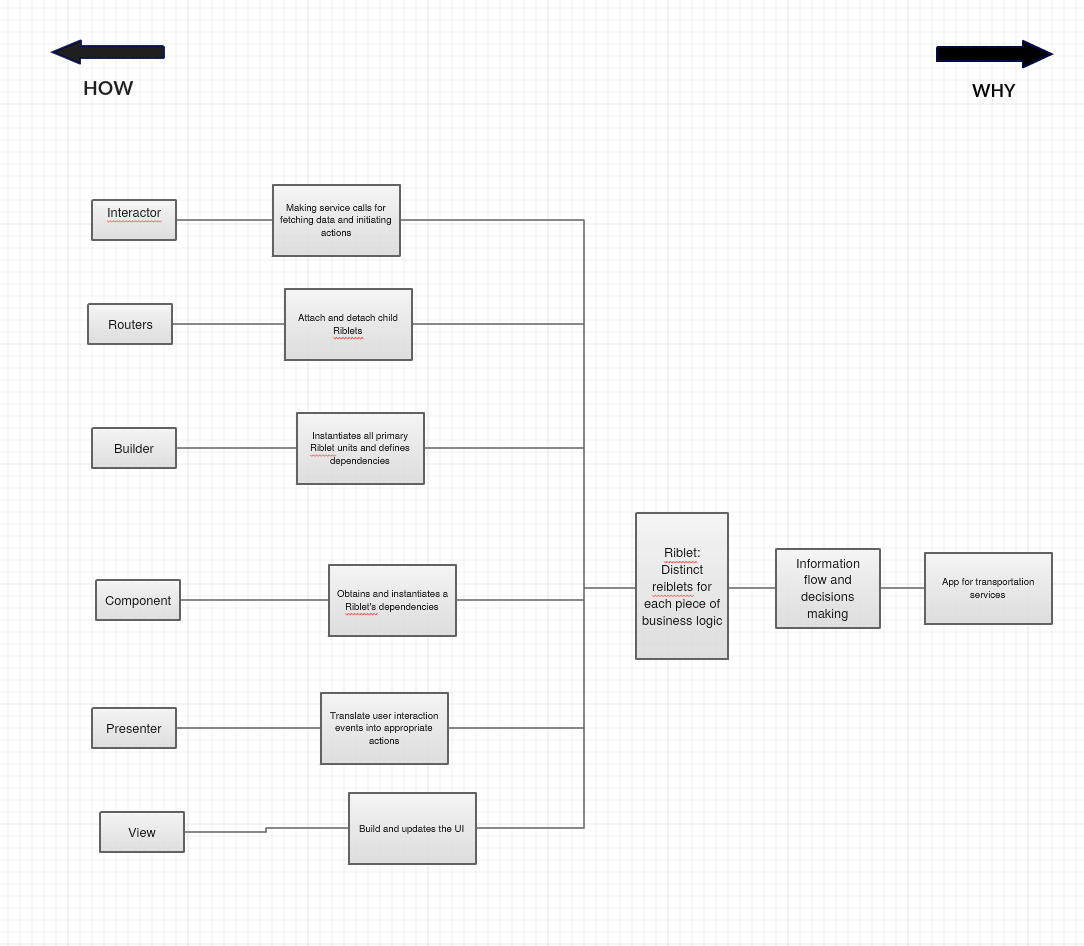
* + 1. **Step 2.2:** System/Subsystems
       1. System -> Uber
       2. Subsystems
          1. Builder
          2. Component
          3. Routers
          4. Interactors
          5. View
          6. Presenter
    2. **Step 2.3:** Primary/Sub-functions

|  |  |
| --- | --- |
| Sub-System | Sub-functions |
| Builder | The builder instantiates all primary Riblet units and define dependencies. |
| Component | Obtains and instantiates a Riblet’s dependencies. This includes services, data streams, and everything else that isn’t a primary Riblet unit. |
| Routers | Routers form the application tree by attaching and detaching child Riblets. These decisions are passed by the Interactor. Routers also drive the Interactor lifecycle by activating and deactivating them at certain state switches. RouterContain two pieces of business logic:   1. Helper Methods for attaching and detaching Routers. 2. State-switching logic for determining states between multiple children |
| Interactors | Interactors perform business logic. This includes, for example:   * Making service calls to initiate actions, like requesting a ride * Making service calls to fetch data * Determining what state to transition to next. For instance, if the root Interactor notices that a user’s authentication token is missing, it sends a request to its Router to switch to the “Welcome” state |
| View | Views build and update the UI, including instantiating and laying out UI components, handling user interaction, filling UI components with data, and animations. |
| Presenter | Presenters manage communication between Interactors and Views. From Interactors to Views, the Presenter translates business models into objects that the View can display. |

* + 1. **Step 2.4:** Primary function
       1. The primary function of Uber is to provide reliable everyday rides.
    2. **Step 2.5:** HOWs and WHYs

|  |  |
| --- | --- |
| HOW | WHY |
| Builder | Instantiates all primary Riblet units |
| Component | Obtains and instantiates a Riblet’s dependencies |
| Routers | Attach and detach child Riblets |
| Interactors | Makes service calls |
| View | Build and updates the UI |
| Presenter | Translate user interaction events. |

* + 1. **Step 2.6:** FAST Diagram

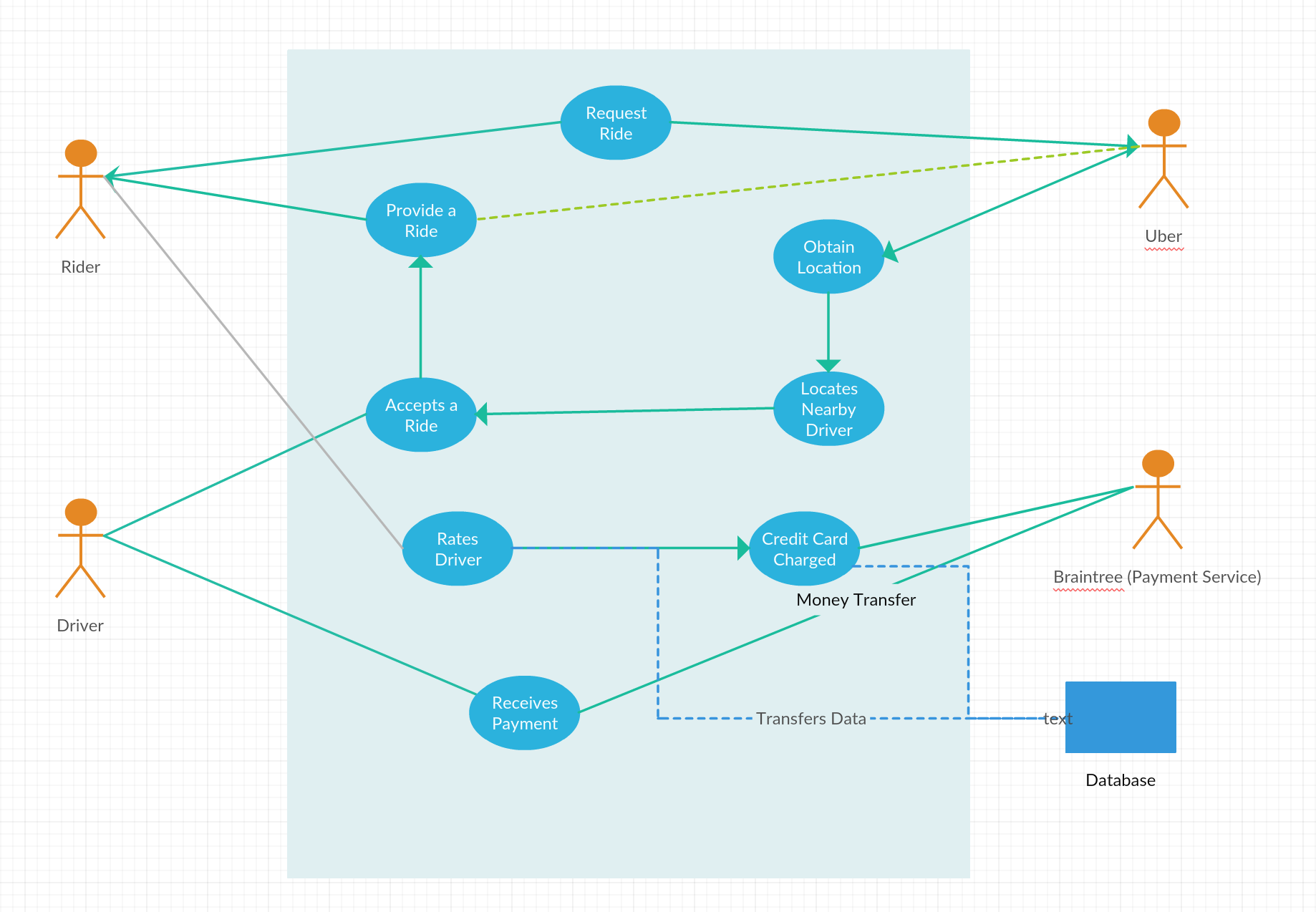
****

* The above diagram is a FAST diagram dissecting Uber’s technology stack and process for how their app works.

1. **Draw Conclusions:** After completing research on Uber I was able to answer all of the needed questions for the first problem. I created a fast diagram dissecting their process for their functioning app using Riblet.

**Problem 3: Software Requirements for Uber**

1. **Define the Problem**
   1. From Problem 2, identify the key business problem that Uber is proposing to solve using IT, and the identify the important business processes to solve this problem.
   2. Define and list a basic set of requirements for the software to automate these business processes. Then define the specific applications necessary to automate these business processes.
   3. Develop use cases for these business processes. Use these use-cases to check the list of software requirements in (b).
2. **Create a Plan**
   1. **Step 1:** Identify the key business problem that Uber is proposing to solve with IT, and then identify the important business processes to solve this problem.
   2. **Step 2:** Define and list a basic set of requirements for the software to automate these business processes.
   3. **Step 3:** Develop use cases for these business processes.
   4. **Step 4:** Use the use-cases to check the list of software requirements found in step 1.2.
3. **Execute**
   1. **Step 1:** Key business problem
      1. The business problem that Uber is proposing to solve with IT, is to make transportation safe, reliable and accessible for everyone, everywhere.
         1. Business process to solve this problem
            1. Request a ride
            2. Obtain location
            3. Provide a driver
            4. Rating System
            5. Payment System
   2. **Step 2:** Requirements for each business process.
      1. Request a ride
         1. Uber Application
      2. Obtain location
         1. CoreLocation framework
         2. Google’s Location APIs
      3. Provide a driver
         1. Apple Push Notifications Service
         2. Google Cloud Messaging
         3. Twilio telecommunications
      4. Rating System
         1. Database/Storage
      5. Payment System
         1. Via credit/debit card. Braintree allows the mobile payment
         2. PayPal’s Card.io service
   3. **Step 3:** Use Case

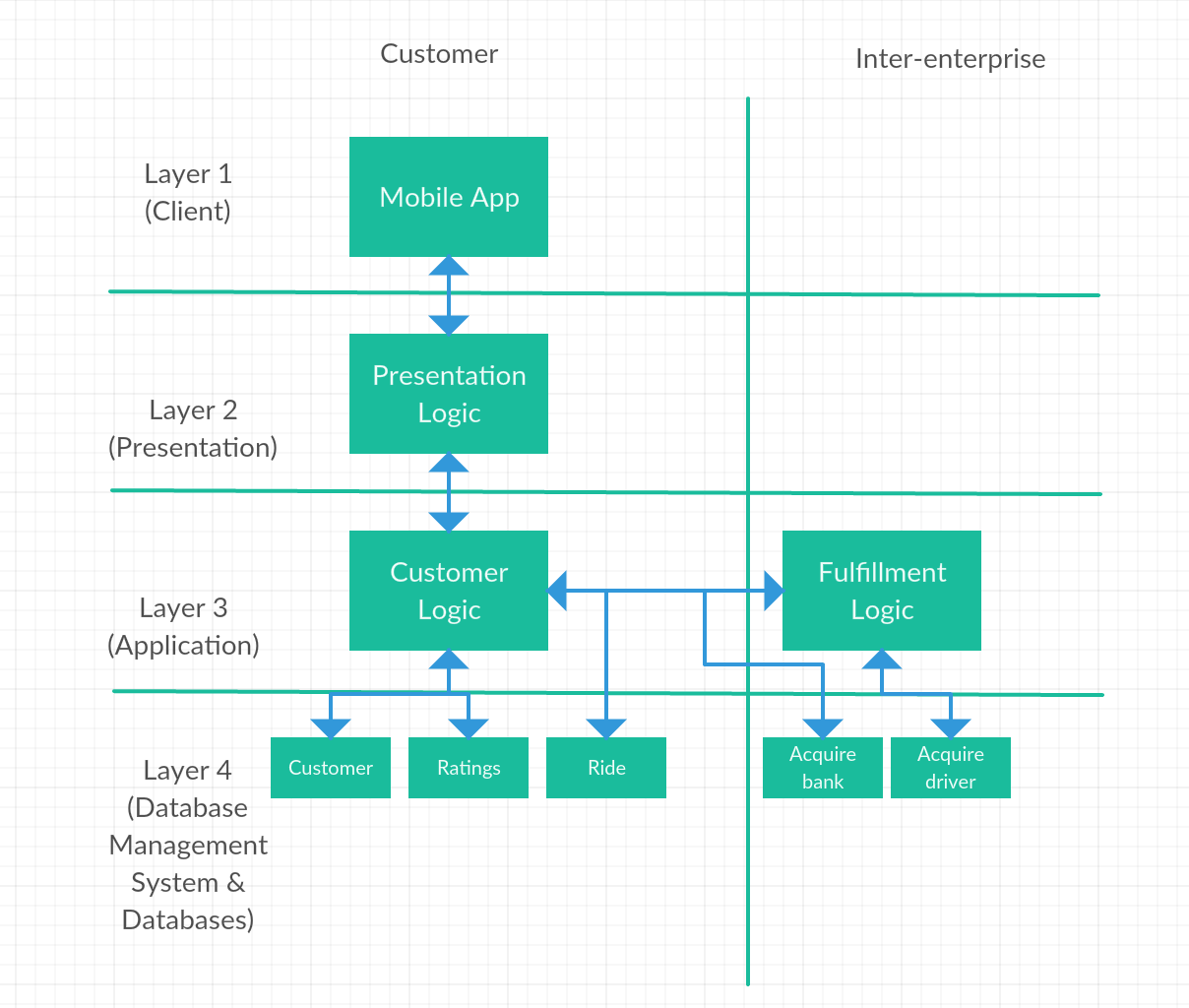
****

* The above use case diagram shows the business processes for the Uber application. It shows how the rider, driver, Braintree, and Uber are all connected and which parts they play.
  1. **Step 4:** Check the Software Requirements in step 2.
     1. From looking at the use case and the requirements we can see that it is accurate.
        1. To request a ride you need to use the Uber application.
        2. For Uber to obtain your location it must use the CoreLocation Framework and Google’s Location API.
        3. To provide a ride Uber uses push notifications and google cloud messaging to send updates to the rider on where there driver is and if they accepted the ride.
        4. The rating system allows riders to have a reliable dring experience and have information on their driver. This information is stored in Uber’s databases.
        5. Uber pays their drivers and allows riders to submit payments by usisng credit/debit card through Braintree. PayPal’s card.io service is also acceptable.

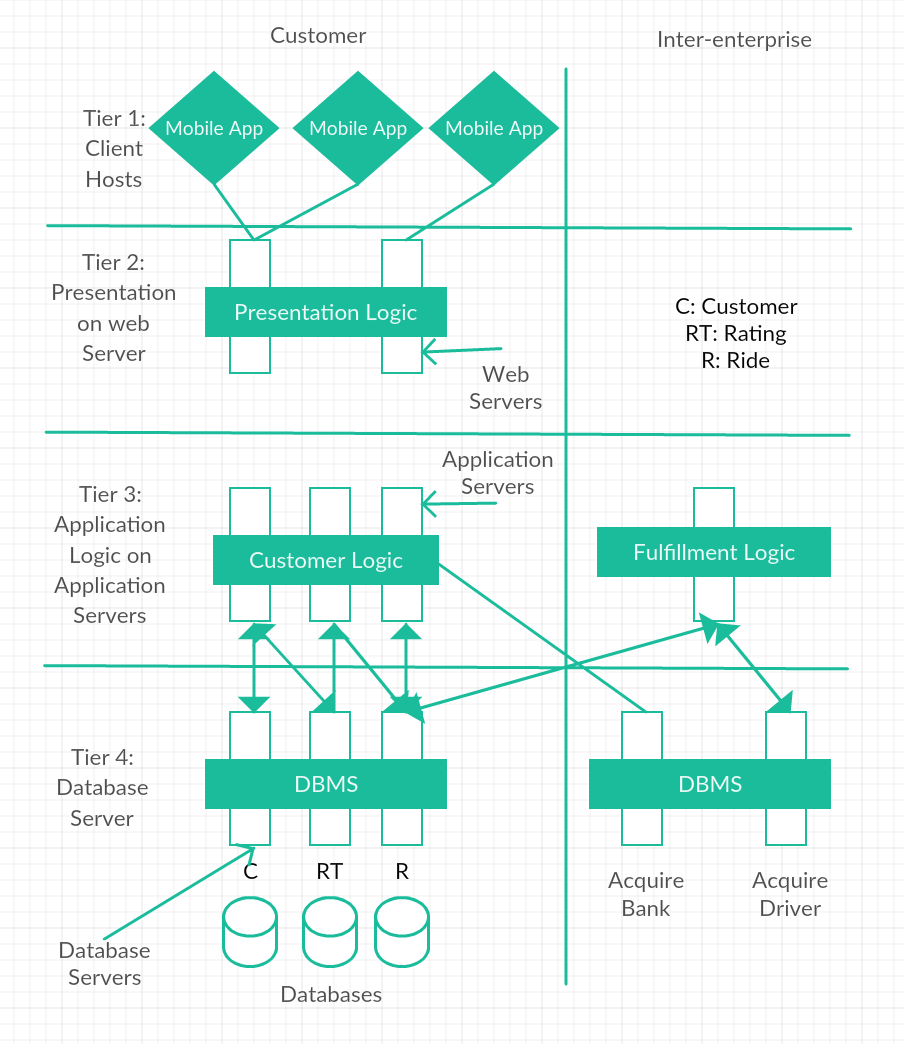
1. **Draw Conclusions**
   1. I found that Uber’s main business problem that they are trying to address is to provide safe and reliable transportation. I was able to create a list of business process to be used to solve this problem through IT.

**Problem 4: Client-Server Software/Hardware Architecture Design for Uber**

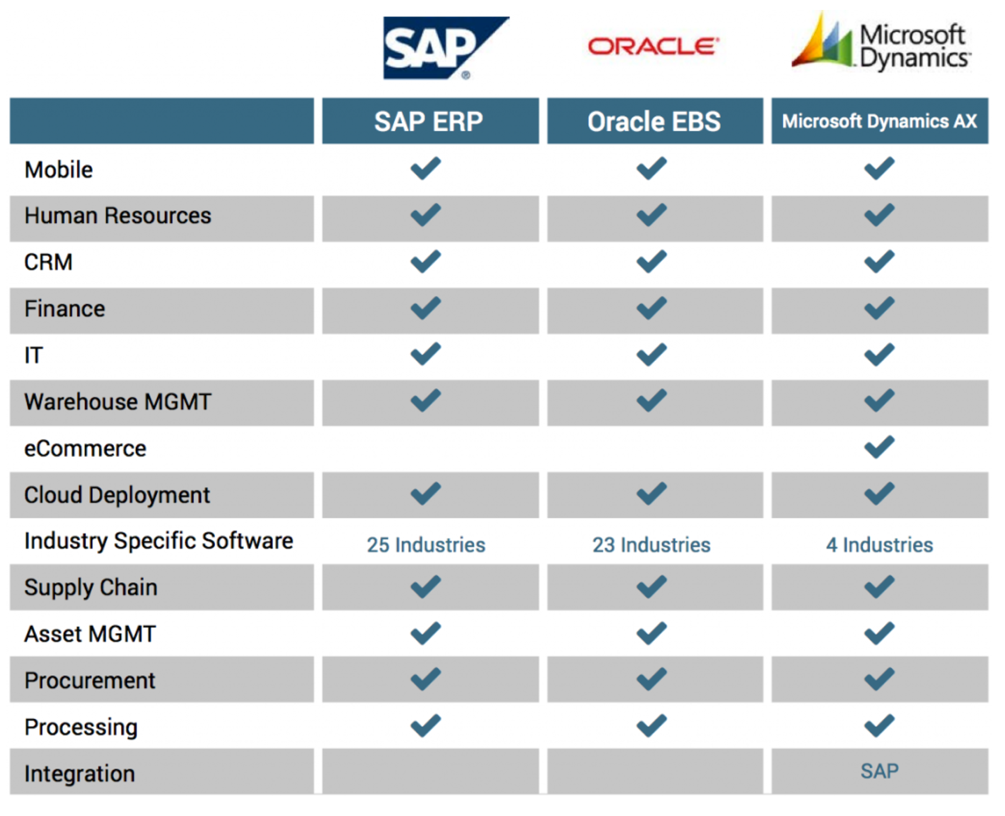
1. **Define the Problem**
   1. Design the client server software and hardware architecture to solve Uber’s key business problem.
   2. Specify software and hardware vendor options (alternatives) for different tiers (layers) in your Uber’s architecture.
   3. Create a set of criteria to address the selection of the software and hardware options (alternatives), and then use these selection criteria to choose the appropriate software and hardware vendor for each tier (layer) of your design.
2. **Create a Plan**
   1. **Step 1:** Define the software architecture
   2. **Step 2:** Determine the number of server tiers to host the software architecture in step 1.
   3. **Step 3:** Specify software and hardware vendor alternatives for the different layers in the IT design.
   4. **Step 4:** Create a set of selection criteria to select between the software and hardware options.
   5. **Step 5:** Use the criteria you created and choose the appropriate software and hardware vendor for each tier of your design.
3. **Execute**
   1. **Step 1:** Software Architecture
      1. Layer 1: Enable the GUI(Uber application) for the client on the client’s mobile device.
      2. Layer 2: Present the necessary information: Surge rates, available rides, driver information, etc..)
         1. Create the GUI for the client
         2. Enables visualization, data entry, communication with the application & output of results
      3. Layer 3: Application Logic
         1. Customer Logic
            1. Customer information
            2. Interacting with driver selection
            3. Interacting with payment process
            4. Interacting with rating system
         2. Fulfillment Logic
            1. Interaction with Braintree/Paypal
            2. Interaction with Bank
            3. Interaction with Driver
      4. Layer 4: Manipulate and Store Data
         1. Customer
         2. Drivers
         3. Ratings
         4. Payment Info

****

* The above figure is the software architecture for Uber. As we can see we have customer rating and ride for customer logic. For fulfillment logic we have acquire bank and acquire driver. Acquire bank is also connected to customer logic because the customer does interact with the payment process. I also connected ride to fulfillment logic because Uber needs to find the driver a ride and make sure that they received one.
  1. **Step 2:**

****

* The above figure is the hardware architecture diagram for the 4-tier client server architecture that we did above. Tier 4 shows the 3 different databases.
  1. **Step 3:** Software and Hardware Vendor Alternatives
     1. Layer 1: Client
        1. Customer decides which device to use to access the app.
           1. Android
           2. iPhone
     2. Layer 2: Presentation
        1. Appery.io
        2. Mobile Roadie
        3. The AppBuilder
        4. Good Barber
     3. Layer 3: Application
        1. Microsoft
        2. Oracle
        3. SAP
     4. Layer 4: Database Management System
        1. Oracle RDBMS
        2. IBM DB2
        3. Microsoft SQL Server
        4. SAP Sybase ASE
  2. **Step 4:** Selection Criteria
     1. Layer 2: Presentation
        1. Ease of Use
        2. Customization
        3. Platform Support
        4. Accessibility
     2. Layer 3: Application
        1. For application software vendor we want a product that can cover any and all departments within the organization.
     3. Layer 4: Database Management System
        1. We want a software that allows our programmers to have a systematic way of creating, retrieving, updating, and managing data.
  3. **Step 5:** Best Vendor
     1. Layer 2: Presentation
        1. Using the criteria chosen in the previous step, the best vendor choice would be Appery.io. This is because the app builder can be used to create apps for multiple platforms (Android, iOS, and Windows Phone). Also the builder runs in the cloud making it easy to start. Most importantly collaboration is simple, allowing you to share the mobile project with development teams and business users in real time.
     2. Layer 3: Application
        1. Below is an image that compares the three software vendors.

****

* + - 1. From the figure above we can see that Microsoft has the most options and appears to be the best choice.
    1. Layer 4: Database Management System
       1. The database management vendor to choose for this situation is Microsoft SQL Server. This is because our application layer is also using Microsoft. Microsoft SQL Server also provides enterprise-class data management and integrated business intelligence tools. SQL Server provides impressive performance for mission-critical applications, using in-memory technologies, faster insights from any data to any user in familiar tools.

1. **Draw Conclusions**
   1. After looking over previous homework’s and through lecture notes I was able to create a framework for this problem. The most difficult part for me was taking the time to perform adequate research for Uber and their software/hardware vendors.

**Problem 5: Database Design for Uber**

1. **Define the Problem**
   1. Based on a clearly defined process specify the key databases (tables) that you need for this application, and provide the appropriate entity (or category) name for each database. Show the relationships between the databases using an entity relationship diagram.
   2. For each database, state its attributes (fields) and its primary key. Show/explain how the information in the different databases are related (connected) to each other for a typical use-case.
   3. Identify a list of potential DBMS vendors and storage vendors for Uber’s IT architecture.
2. **Create a Plan**
   1. **Step 1:** Identify the Databases that contain the information you need.
   2. **Step 2:** Create an entity relationship diagram.
   3. **Step 3:** Specifying the DB Servers for the application and the vendors for these servers.
      1. How many DB servers?
      2. Functionality of each server?
      3. Vendor
3. **Execute**
   1. **Step 1:** Databases
      1. Customer Info

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Primary Key | First Name | Password | E-mail | Payment Information |
| Customer ID 1 |  |  |  |  |
| Customer ID 2 |  |  |  |  |

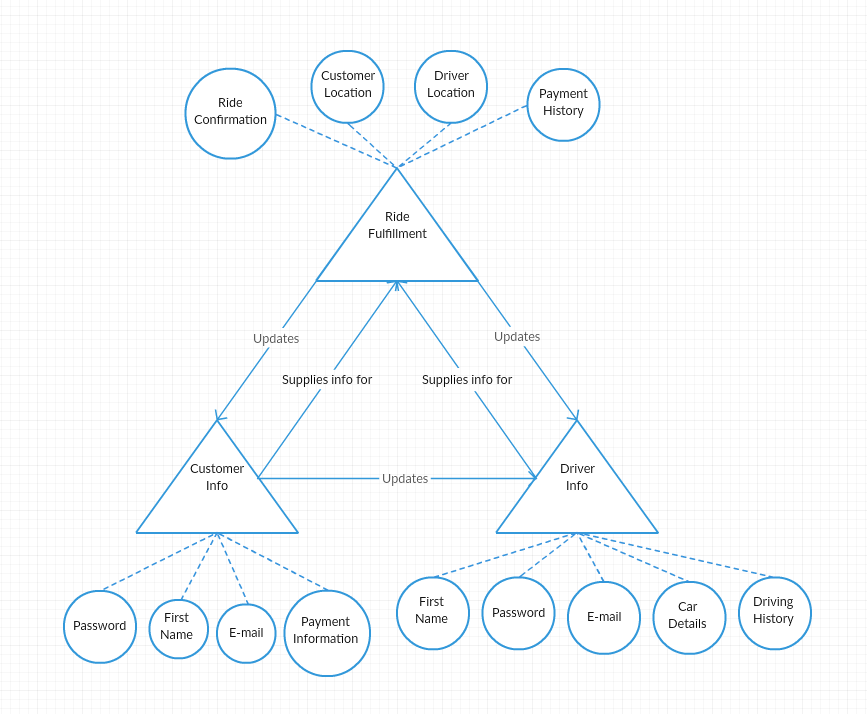
* + 1. Driver Info

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Primary Key | First Name | Password | E-mail | Car Details | Driving History |
| Driver ID 1 |  |  |  |  |  |
| Driver ID 2 |  |  |  |  |  |

* + 1. Ride Fulfillment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Primary Key | Ride Confirmation | Customer Location | Driver Location | Payment History |
| Ride #1 |  |  |  |  |
| Ride #2 |  |  |  |  |

* 1. **Step 2:** Entity Relationship Diagram



* This entity relationship diagram shows the relationship between the three database servers: ride fulfillment, customer info, and driver info. Each database has balls showing the information that it holds.
  1. **Step 3:** DBMS vendor options
     1. How many DB servers?
        1. We have three database servers
           1. Customer Info
           2. Driver Info
           3. Ride Fulfillment
     2. Functionality of each server?
        1. Customer Info
           1. Stores the information of the driver. This includes their account information to access the application and their payment information. Their payment information is stored in order to pay the driver through PayPal or Braintree.
        2. Driver Info
           1. Stores the drivers information. This includes their rating and reviews from previous rides. Their account info to access the application. Their car details and driving credentials are also stored for the company to verify they are a valid driver.
        3. Rider Fulfillment
           1. This stores data about customer locations and driver locations in order to connect the driver and rider. Confirmation that a ride has been accepted and completed is also stored in the database. Payment history is also stored in order to make sure rides have been paid for.
     3. Vendor
        1. Vendor Options
           1. Oracle Database
           2. Microsoft SQL Server
           3. IBM DB2
           4. SAP Sybase ASE
           5. PostgreSQL
           6. MySQL
        2. Vendor Choice
           1. MySQL
           2. Prior to MySQL Uber used Postgres. They ran into the following limitations:

Inefficient architecture for writes

Inefficient data replication

Issues with table corruption

Poor replica MVCC support

Difficulty upgrading to newer releases

* + - * 1. MySQL supports multiple different replication modes.
        2. Statement-based replication replicates logical SQL statements

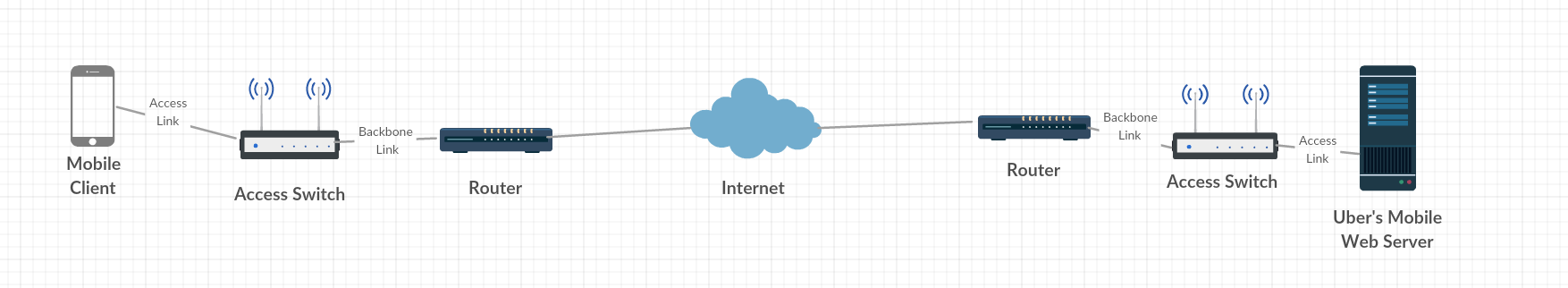
1. **Draw Conclusions**
   1. I was able to complete this problem by using my knowledge on Uber and what I have compiled from earlier steps. I found that Uber should have 3 databases: customer info, driver info, and ride fulfillment. This seemed to be the most efficient combination for the company. In the relationship diagram you can see how I connected all of the databases and the information that they contain.

**Problem 6: Network Architecture for Uber**

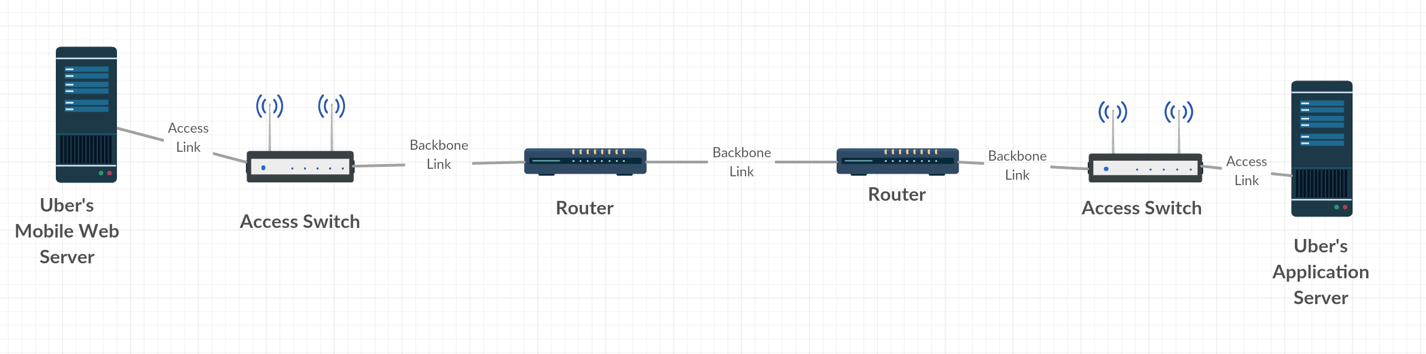
1. **Define the Problem**
   1. Based on a clearly defined process develop the appropriate network architecture for Uber’s data center.
   2. Show the network topology to transmit application commands and data back and forth between Uber and the end-user for a typical use-case.
   3. Clearly explain the information/data flow on the internet across the four primary layers (application, network, etc.) for a typical use-case.
2. **Create a Plan**
   1. **Step 1:** Define the network devices that are needed.
   2. **Step 2:** Map the requirements to the devices.
   3. **Step 3:** Based on the layering of the IT architecture, define a set of subnets.
   4. **Step 4:** For each subnet, determine the network topology to connect the servers in the subnet using switches (& routers).
   5. **Step 5:** Connect between the subnets using routers & switches to achieve the n-tier architecture.
   6. **Step 6:** Add load balancers for each layer of the architecture.
   7. **Step 7:** Add firewalls for each layer of the architecture.
3. **Execute**
   1. **Step 1:** Network Devices
      1. Routers & switches
      2. Load balancers
      3. Firewalls
   2. **Step 2:** Requirements for Devices

|  |  |  |  |
| --- | --- | --- | --- |
| Requirements | Routers & switches | Load balancers | Firewalls |
| Scalability | X | X |  |
| Reliability |  | X |  |
| Security |  |  | X |
| Agility | X | X |  |

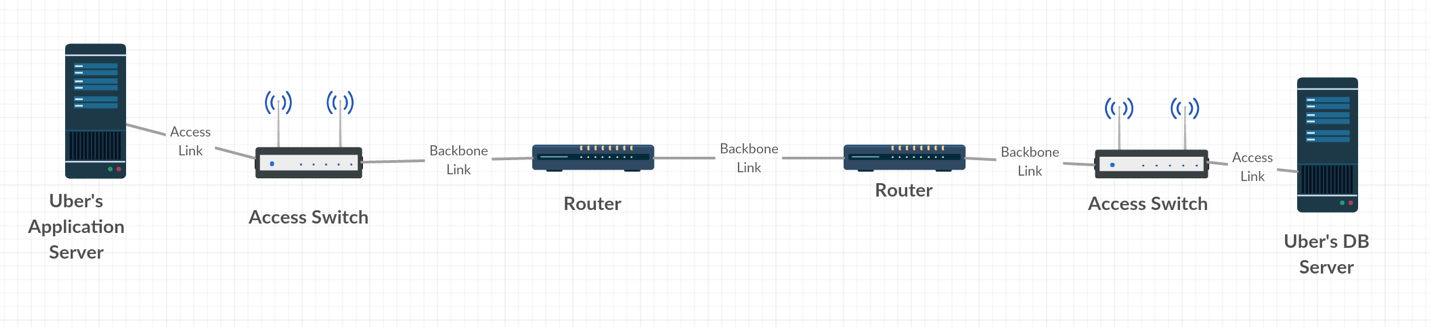
* 1. **Step 3:** Subnets
     1. Web Server Layer Web Server Subnet
     2. Application Server Layer Application Server Subnet
     3. DB Server Layer DB Server Subnet
  2. **Step 4:** Network Topology
     1. Web Server Topology

****

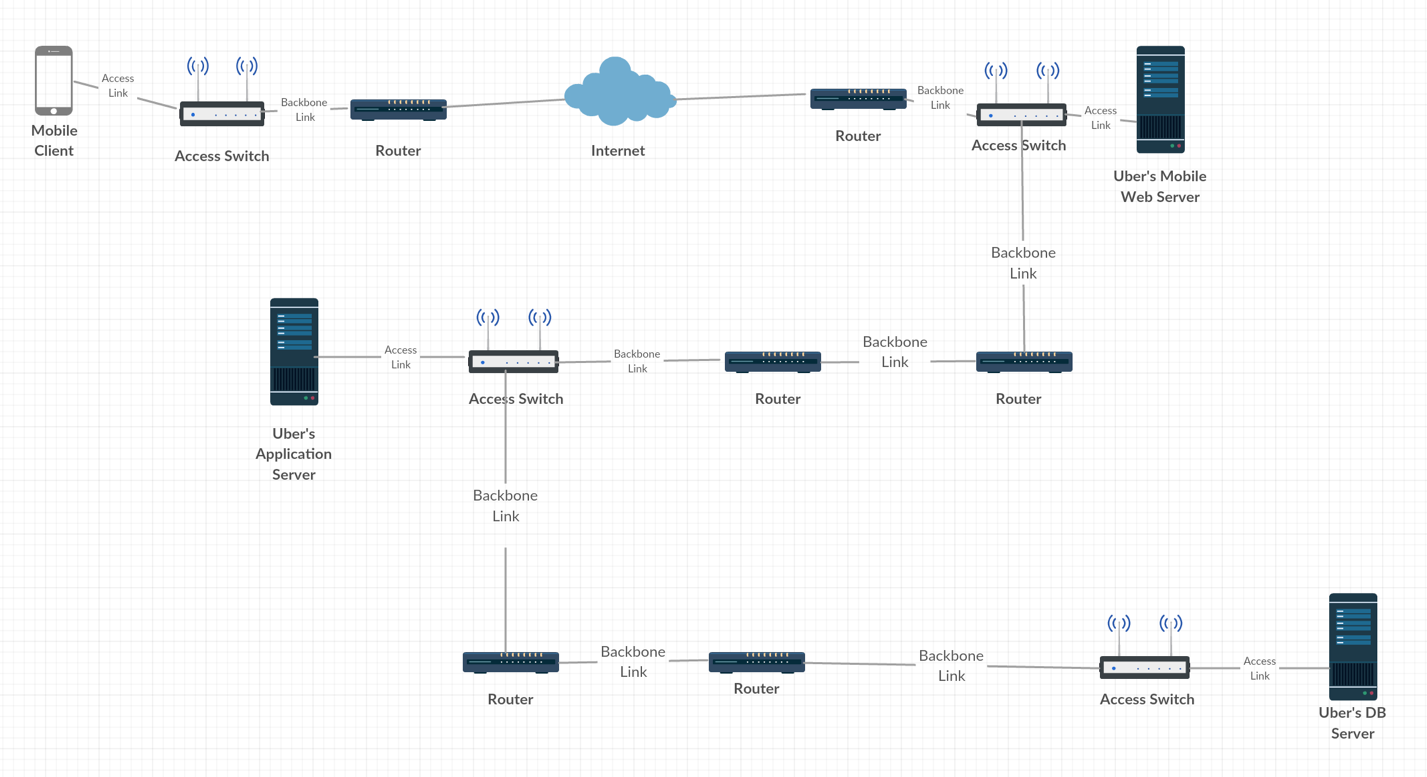
* + - 1. The above figure shows the connection between the Mobile client and Uber’s web servers. The client user must go through the internet in order to access Uber’s web servers.
    1. Application Server Topology

****

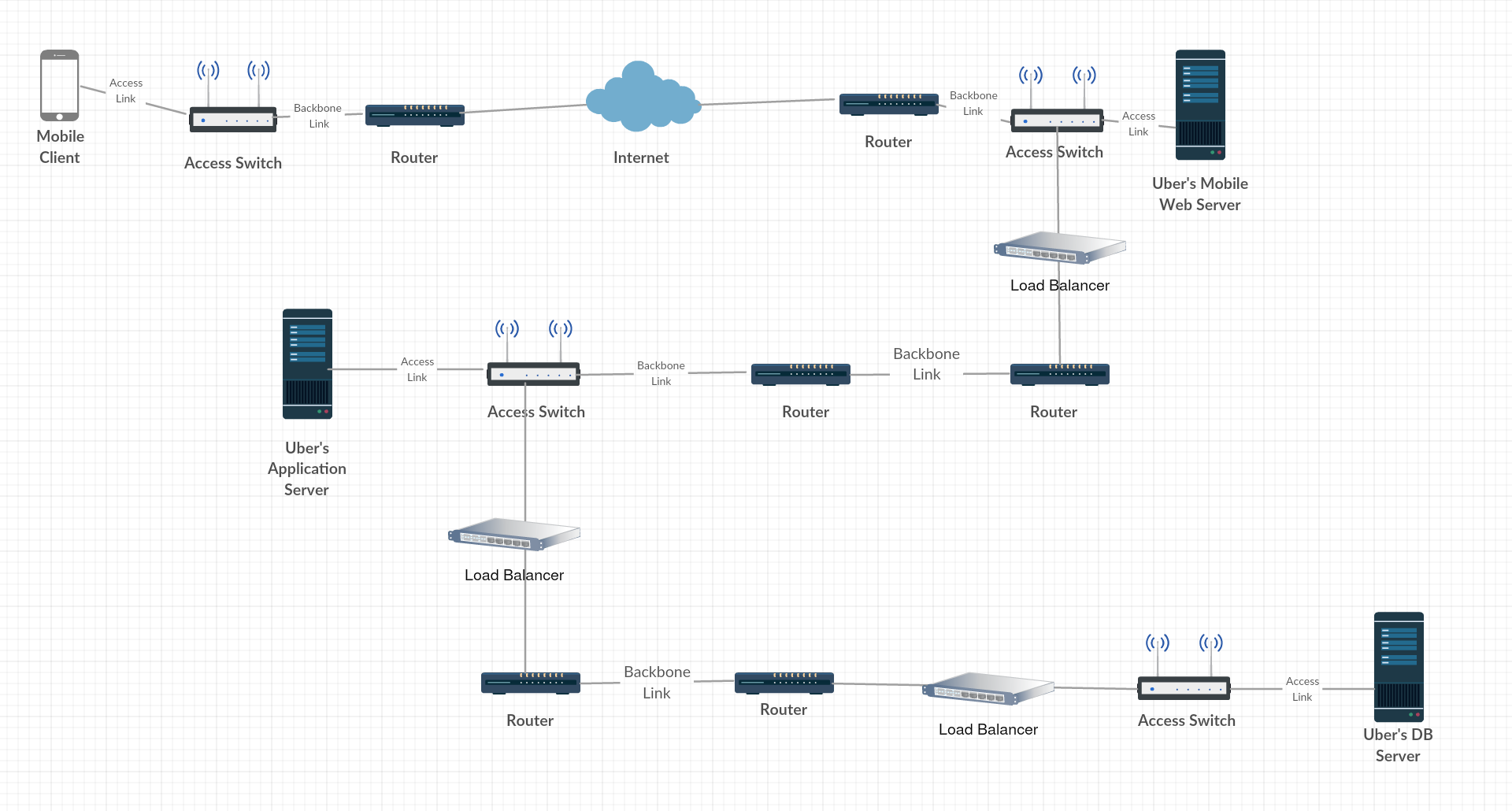
* + - 1. Uber’s web server is connected to it’s applications servers through routers and access switches.
    1. DB Server Topology

****

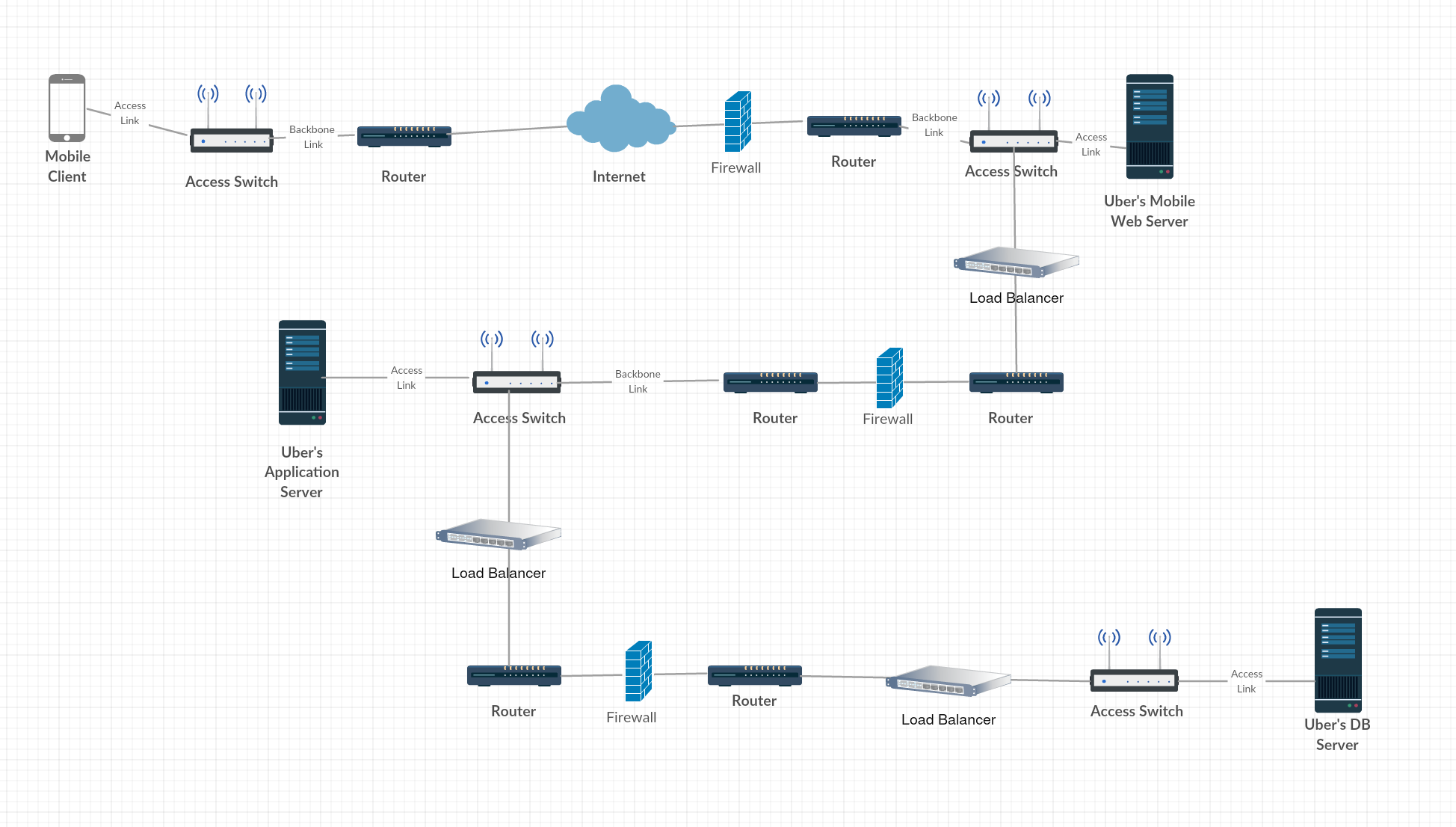
* + - 1. Amazon’s application server and DB server are also separated by routers and switches.
  1. **Step 5:** Connect all of the subnets

****

* + 1. The above figure shows all of the servers connected to form the network.
  1. **Step 6:** Add load balancers

****

1. The above figure shows the load balancers added in which helps Uber deal with surges in bandwidth. This can be extremely helpful late at night when a lot of people leaving bars are requesting rides.
   1. **Step 7:** Add firewalls

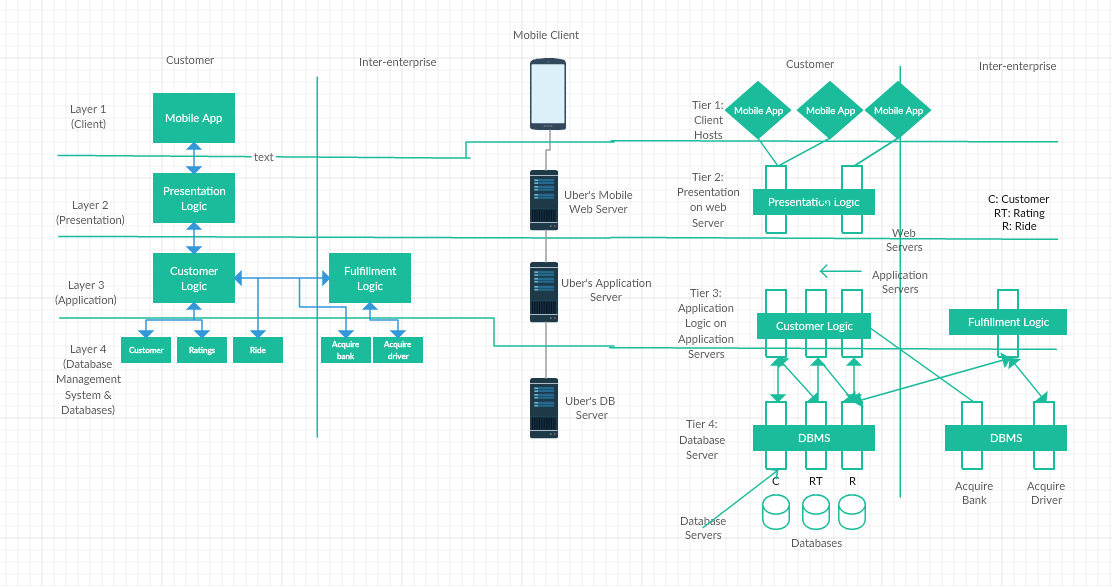
****

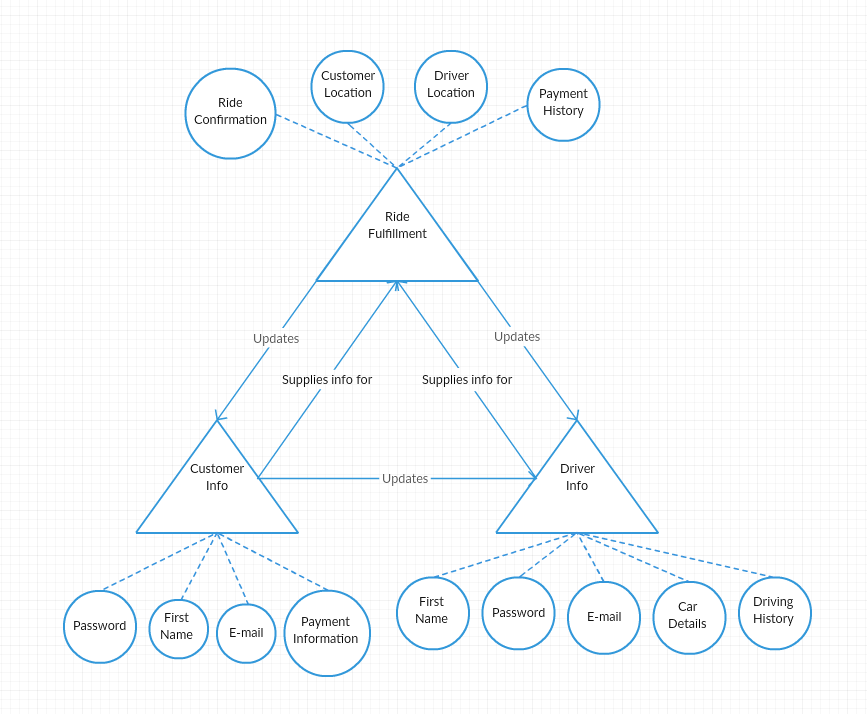
* + 1. The above figure shows the firewalls in the network. They are there to block any unauthorized access while permitting outward communication. This allows the customers to have a safely pay Uber for their rides.

1. **Draw Conclusions**
   1. I was able to create the topology for Uber’s servers based off of the previous information that I’ve gathered in the earlier problems. I found that they had a direct route to their servers.

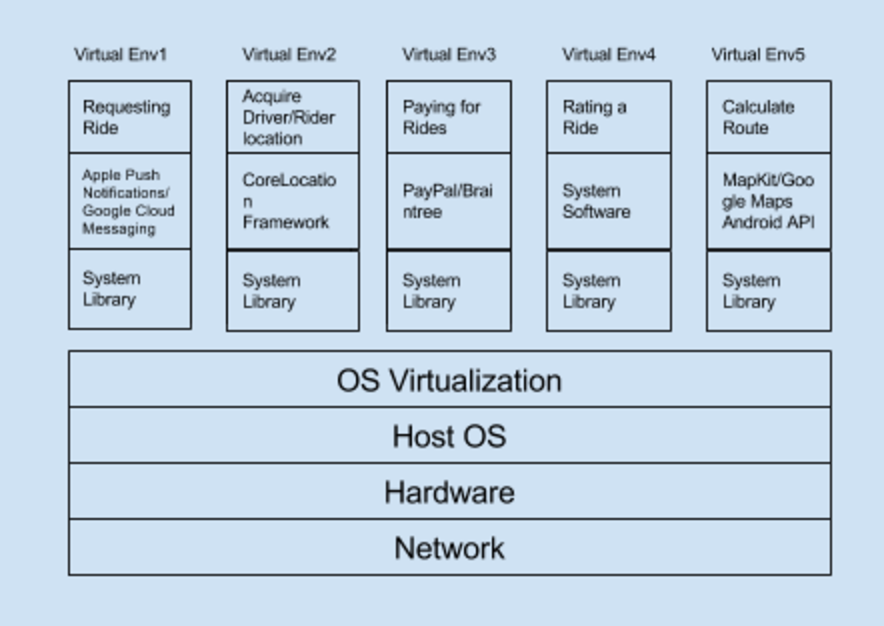
**Problem 7: IT Integration and Virtualization for Uber**

1. **Define the Problem**
   1. Based on a clearly defined process, design the end-to-end integrated IT architecture for Uber: client-server architecture, database, storage and network architecture.
   2. Develop and implement a process for virtualizing Uber’s data center.
2. **Create a Plan**
   1. **Step 1:** Design the end-to-end integrated IT architecture for Uber.
   2. **Step 2:** OS Virtualization
3. **Execute**
   1. **Step 1:** End-to-End Integrated IT architecture

****



* The above diagram is the end to end IT integration of all the architectures we design in the previous steps. We have the software and hardware architecture side by side leading to the databases servers.
  1. **Step 2:** OS Virtualization

****

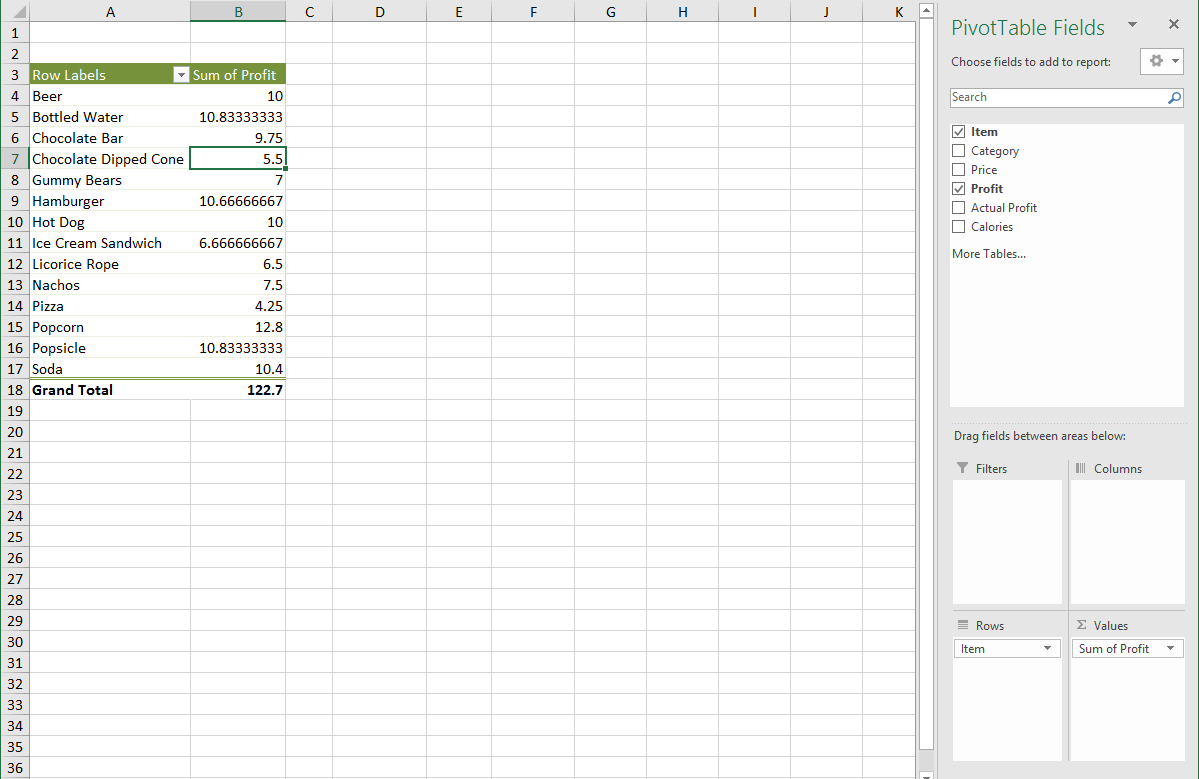
* The above diagram shows the OS Virtualization for Uber. There are 5 different applications and the different system software used for each one are shown below.

1. **Draw Conclusions**
   1. I had a hard time figuring out how to design the end to end it architecture. I just combined the knowledge that I have gained from the previous problems to create an overall it architecture. I also decided to do an OS virtualization for Uber because they are more software oriented.

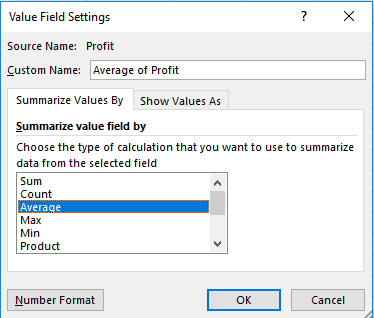
**Problem 8: Excel Solver**

1. **Define the Problem**
   1. Download the spread sheet listed under “Chapter 1” available at the following URL <http://www.wiley.com/WileyCDA/WileyTitle/productCd-111866146X.html>
   2. The “Concessions” tab in the spread sheet provides information on items such as the category it belongs to, price, profit, actual profit and calories. Read the section “Using PivotTables” in Chapter 1 of “Data Smart” and, and then answer the following questions:
      1. Use the PivotTable to find items that have a profit of 50%, 67%, and 83%?
      2. For each category of item (e.g. Beverages), what is the total (cumulative) “Actual Profit” of all items in that category?
   3. You must provide screenshots of the excel worksheet you used to answer the above questions.
2. **Create a Plan**
   1. **Step 0:** Download the spread sheet and read through the section “Using PivotTable” in Chapter 1 of “Data Smart”.
   2. **Step 1:** Find items that have a profit of 50%, 67%, and 83%.
   3. **Step 2:** Find the total cumulative “Actual Profit” of all items for each category .
3. **Execute**
   1. **Step 1:** Items with 50%, 67%, and 83% profit.

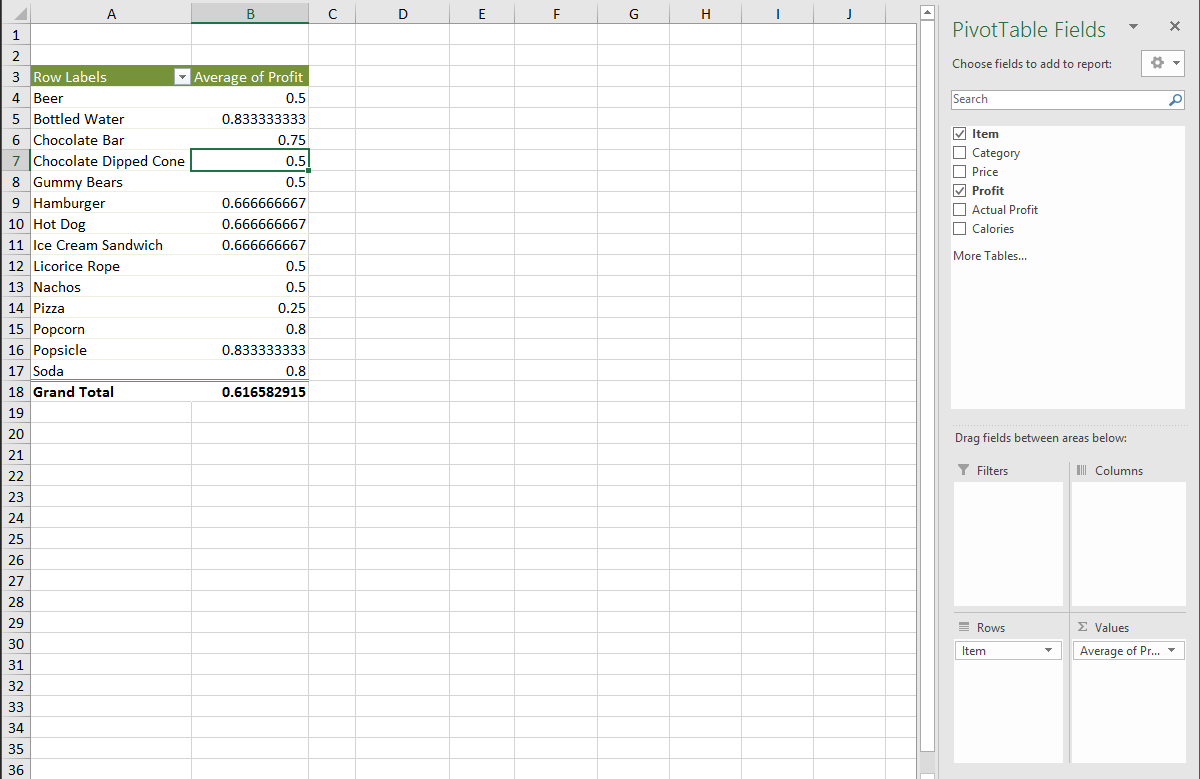
I first created the PivotTable by selecting the data from the Basketball Game Sales tab. This data was in the range A1:F200. We needed to find the profit for each item so I put the Item field in the Row box and the Profit field in the Values box. This results in the figure below.



I then needed to change the Profit from “Sum of Profit” to “Average of Profit” in order to find the percent values. The sum of profit was adding all of the percentage values. I changed this in the figure below



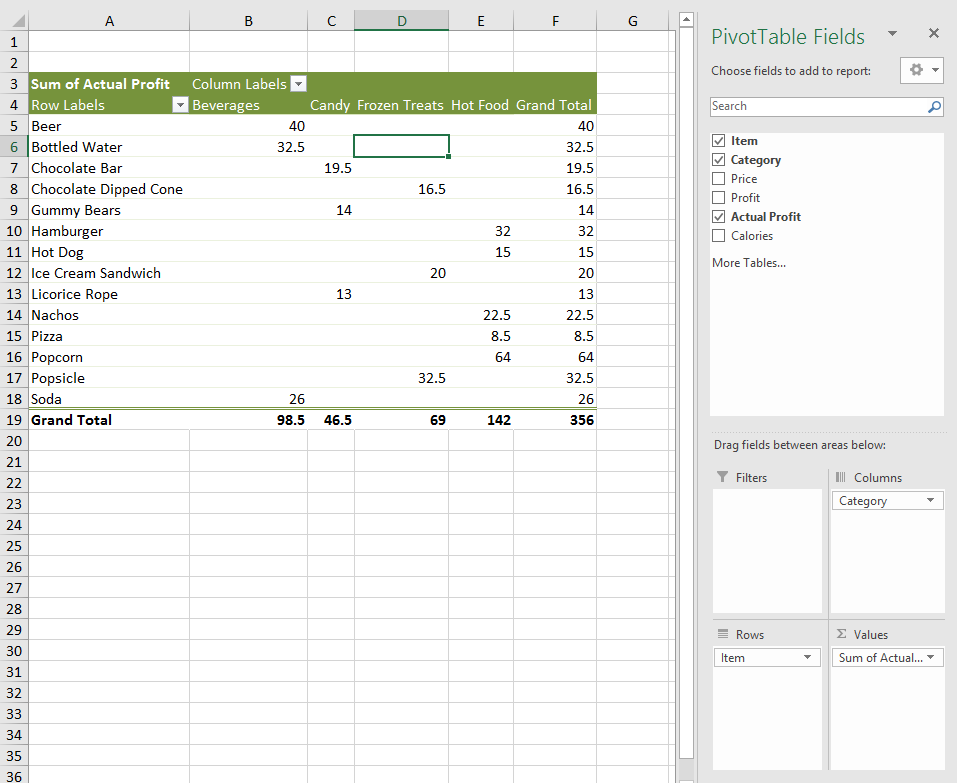
Below is the result after changing from “Sum of Profit” to “Average Profit”.



As we can see the we have the following results:

* + - 1. Items with 50% profit:
         1. Beer
         2. Chocolate Dipped Cone
         3. Gummy Bears
         4. Licorice Rope
         5. Nachos.
      2. Items with 67% profit:
         1. Hamburger
         2. Hot Dog
         3. Ice Cream Sandwich
      3. Items with 83% profit:
         1. Bottled Water
         2. Popsicle
  1. **Step 2:**

To find the sum of the Actual Profit for each category I used the following settings shown in the figure below.



This gives us the following results:

* + - 1. Beverages: $98.5
      2. Candy: $46.5
      3. Frozen Treats: $69
      4. Hot Food: $142

1. **Draw Conclusions**
   1. I found this problem to be fairly easy because the guide in Data Smart was straightforward. I also have previous excel knowledge from TIM 105 and TIM 125.

**Problem 9: Execution of Your Plan**

1. **Define the Problem**
   1. Using a table compare your plan from Problem 1 (column 1) with its execution (column 2). Indicate the reasons for the difference between the plan and its execution (column 3). Add at least one more column. What should column 4 contain?
2. **Create a Plan**
   1. **Step 1:** Create a table to compare the plan from problem 1 with its execution.
3. **Execute**
   1. **Step 1:** Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. **Problem** | **Plan** | **Execution** | **Reasons for Difference** | **What could you have done better?** |
| Reading | I planned to have this part done by 5/6/17 | I was able to finish the reading on time | No difference. | Nothing |
| 1 | I planned to have this part done by 5/6/17 | I was able to complete it in the planned time. | No difference. | Nothing |
| 2 | I planned to have this part done by 5/6/17 | I was able to complete it in the planned time. | No difference. | Nothing |
| 3 | I planned to have this part done by 5/6/17 | I was able to complete it in the planned time. | No difference. | Nothing |
| 4 | I planned to have this part done by 5/6/17 | I was able to complete it in the planned time. | No difference. | Nothing |
| 5 | I planned to have this part done by 5/7/17 | I was able to complete it in the planned time. | No difference. | Nothing |
| 6 | I planned to have this part done by 5/7/17 | I was able to complete it in the planned time. | No difference. | Nothing |
| 7 | I planned to have this part done by 5/7/17 | I wasn’t able to complete it in the planned time. | I visited my girlfriend’s father for his 50th birthday and it took more time away from the midterm than I thought. | I should have compensated for the fact that I was visiting family and planned accordingly. |
| 8 | I planned to have this part done by 5/7/17 | I wasn’t able to complete it in the planned time. | Had to push back this problem due to pushing back problem 7 | I should have allowed more time in case anything went wrong. |
| 9 | I planned to have this part done by 5/7/17 | I wasn’t able to complete it in the planned time. | Had to push back this problem due to pushing back problem 7 | I should have allowed more time in case anything went wrong. |

1. **Draw Conclusions**
   1. After doing this chart I realized that not all plans will be perfect. There will be hiccups in the plan and you have to be able to adapt accordingly. This is what determines whether the end product will be good.