**CS3354 Software Engineering Final Project Deliverable 2**

**TicketSlackers** by the TicketTeam

**TicketTeam**

Brian Nguyen

Muhammad Quadri

Adam Lipscomb

Akash Prakash

Dylan Hua

Ray Raymos

Thang Vu

Github Repo Link

<https://github.com/RayBrandServices/3340-TicketTeam>

1. **Delegation of Tasks**

**Project Deliverable 1**

1.1 - 1.3. Ray - Github Repo, add TA, add team members

1.4. Brian - Make commit

1.5. Akash - Make another commit including a pdf/txt/doc file named “project\_scope”.

1.6. Akash - Include website in deliverable

2. Thang - Delegation of tasks

3. Brian - Which software process model is employed in your project and why. (Ch 2)

4a. Thang - Functional req

4b. Akash - Non-functional req

5. Adam - Use case diagram

6. Muhammad - Sequence diagram

7. Dylan, Ray - Class diagram

8. Brian - Architectural design

Extra: Brian - Organizing deliverable report

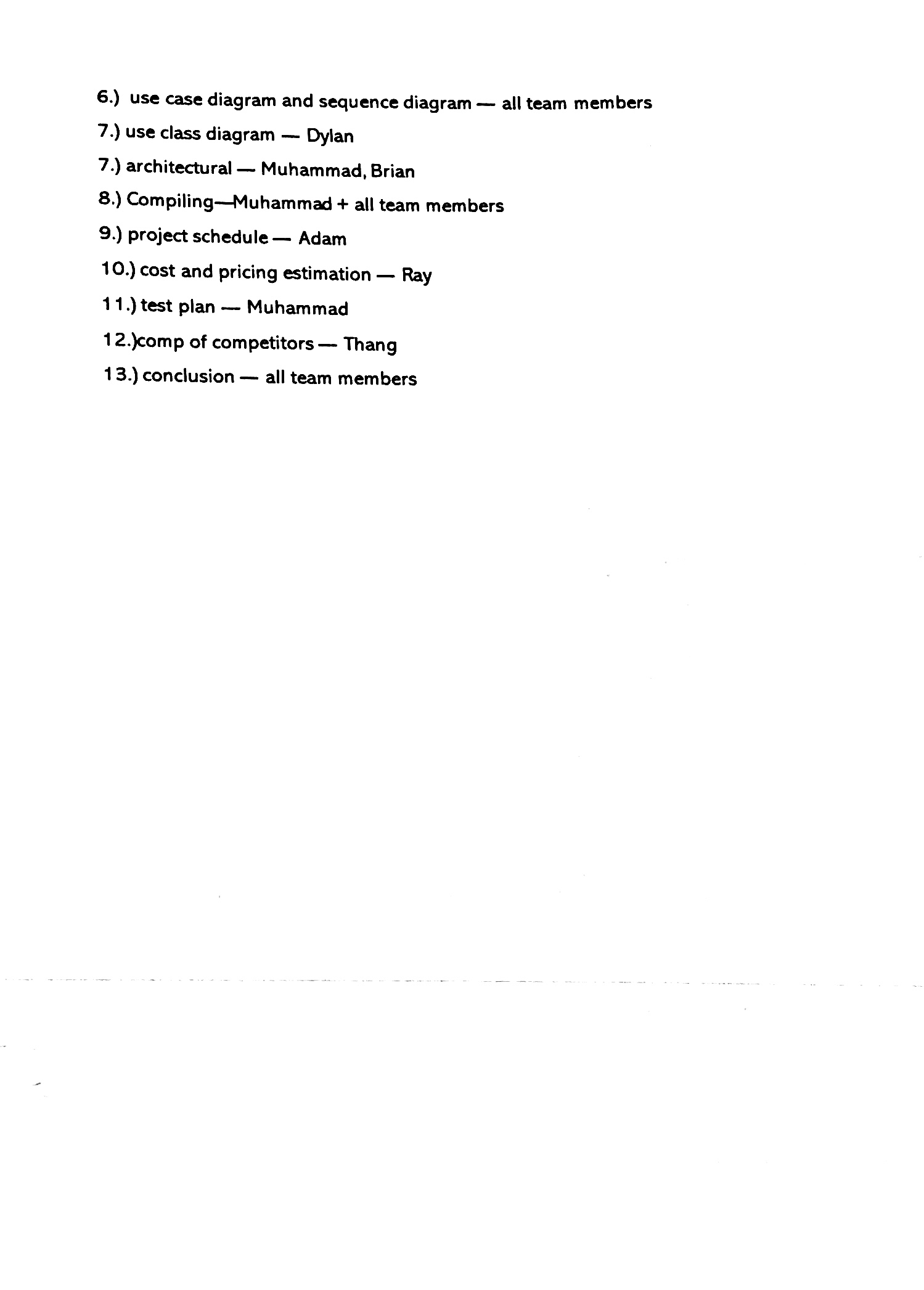
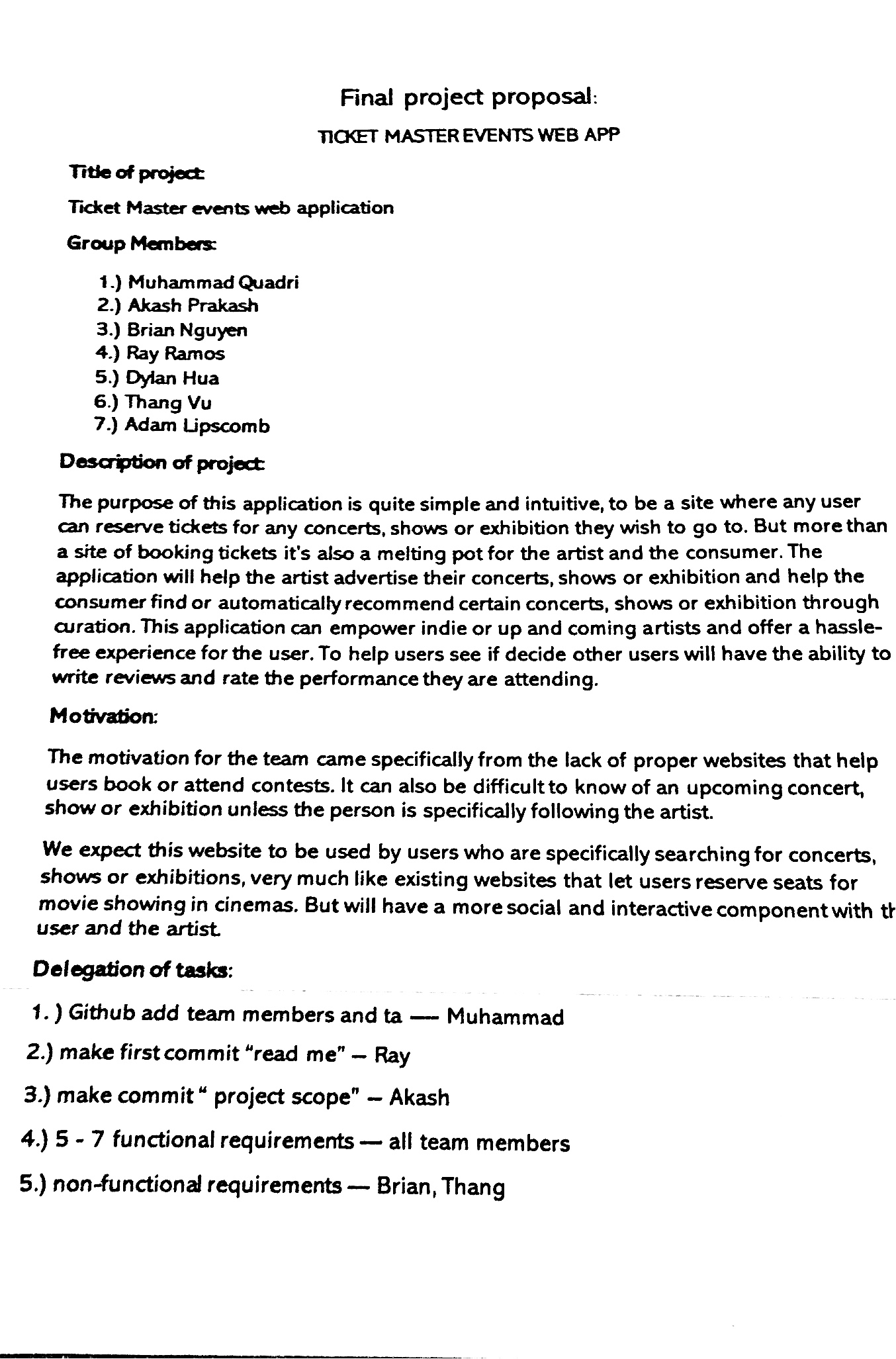
**Project Deliverable 2**

1. Project scheduling (MS Project) - Thang, Akash, Muhammad, Dylan
2. Project Deliverable 1 - Muhammad, Akash, Dylan, Brian, Ray, Thang, Adam
3. Cost, Effort and Pricing Estimation - Adam
   1. Function point - Adam
   2. Estimated cost of hardware products - Brian
   3. Estimated cost of software products - Brian
   4. Estimated cost of personnel - Dylan
4. Test Plan - Ray, Brian
5. Powerpoint - Muhammad, Akash, Dylan, Brian, Ray, Thang, Adam
6. Comparison -Muhammad
7. Conclusion - Akash
8. References - Akash, Dylan, Ray
9. UI Mockup - Ray, Brian

**2. Project Deliverable 1 content**

**1. Draft Description**

We did not receive any additionally critiques on our final project proposal. However, to address the general critiques that were made to all groups we have made sure to account for the things that were mentioned to have in the final deliverable. As stated in the announcement we will have a citations, unique features, and a comparison to a similar product in our final deliverable. Since our final project proposal was approved by the professor with no additional comments we have gone ahead and proceeded with completing the following deliverable.

****

**2. Delegation of Tasks**

1.1 - 1.3. Ray - Github Repo, add TA, add team members

1.4. Brian - Make commit

1.5. Akash - Make another commit including a pdf/txt/doc file named “project\_scope”.

1.6. Akash - Include website in deliverable

2. Thang - Delegation of tasks

3. Brian - Which software process model is employed in your project and why. (Ch 2)

4a. Thang - Functional req

4b. Akash - Non-functional req

5. Adam - Use case diagram

6. Muhammad - Sequence diagram

7. Dylan Ray - Class diagram

8. Brian - Architectural design

Extra: Brian - Organizing deliverable report

**3. Software Process Model**

For this project, we are leveraging the incremental process model. The incremental process model combines both aspects of linear and parallel process models. Each linear sequence produces an incremental deliverable of the software. At each iteration of this process model, we are able to quickly communicate, plan, model, construct, and deploy a part of the software in order to facilitate an efficient and effective development over a period of time. Since we communicate often with the customer after finishing an iteration, if the customer’s requirements change, we can easily adapt our development process to meet the customer’s requirements. It is easier to identify and manage risks because risks are handled at each iteration of the process. It is also easier to debug and test because each the changes to the overall software made during the iteration is relatively small. Ultimately, this process model is more flexible, which makes changes to project scope and requirements less costly.

**4. Software Requirements**

**Functional Requirements**

1. Homepage:
   * Display Event Search and ‘Money Shot’ option
   * Allow the user to make event search request or ‘Money Shot’ Request
2. Event search request: when requested, the app must display events that align with the user’s search
   * Event search criteria: Zip code or city, date range, specific artist or venue or specific event
     + Any combination of search criteria can be taken as search input. However, there must be at least one search criteria provided by the user.
   * If zip code or city entered, pull the 10 (max) recommended events in the near proximity
   * If additional date range is specified (all, this weekend, or a specific date range), pull the 10 (max) recommended events located near the proximity of the zip code or city entered that match the date ranges
   * If additional information is given such as an artist, a venue, or a specific event, pull 10 (max) recommended events located near the proximity of the zip code or city entered that match the date range and the additional information
   * When more than 10 events are available under the search criteria, show the option ‘view more’ for the user.
     + If ‘view more option’ is requested from the user, pull 10 (max) more recommended events under the same search criteria. These events must be displayed with the previous 10 recommended events.
3. Each user may subscribe to some of the search criteria.
   * Event search criteria: specific artist
     + Any combination of search criteria can be taken as search input. However, there must be at least one search criteria provided by the user.
     + In this search criteria, a price range (min$, max$) for events can also be provided by the user. If the user opts in for this option, apply the price filter to the events in addition to the search criteria.
     + A price range filter must be accompanied by any combination of the search criteria
     + Database keeps track of phone number and their artist subscriptions
       - Whenever a notification gets sent out, allow the user to opt out of the subscription by replying “STOP”
         * STOP triggers unsubscribe method.

Unsubscribe unregisters the user’s phone number with the artist in the database. This would mean the user will no longer get text notifications regarding the events tied to that artist subscription

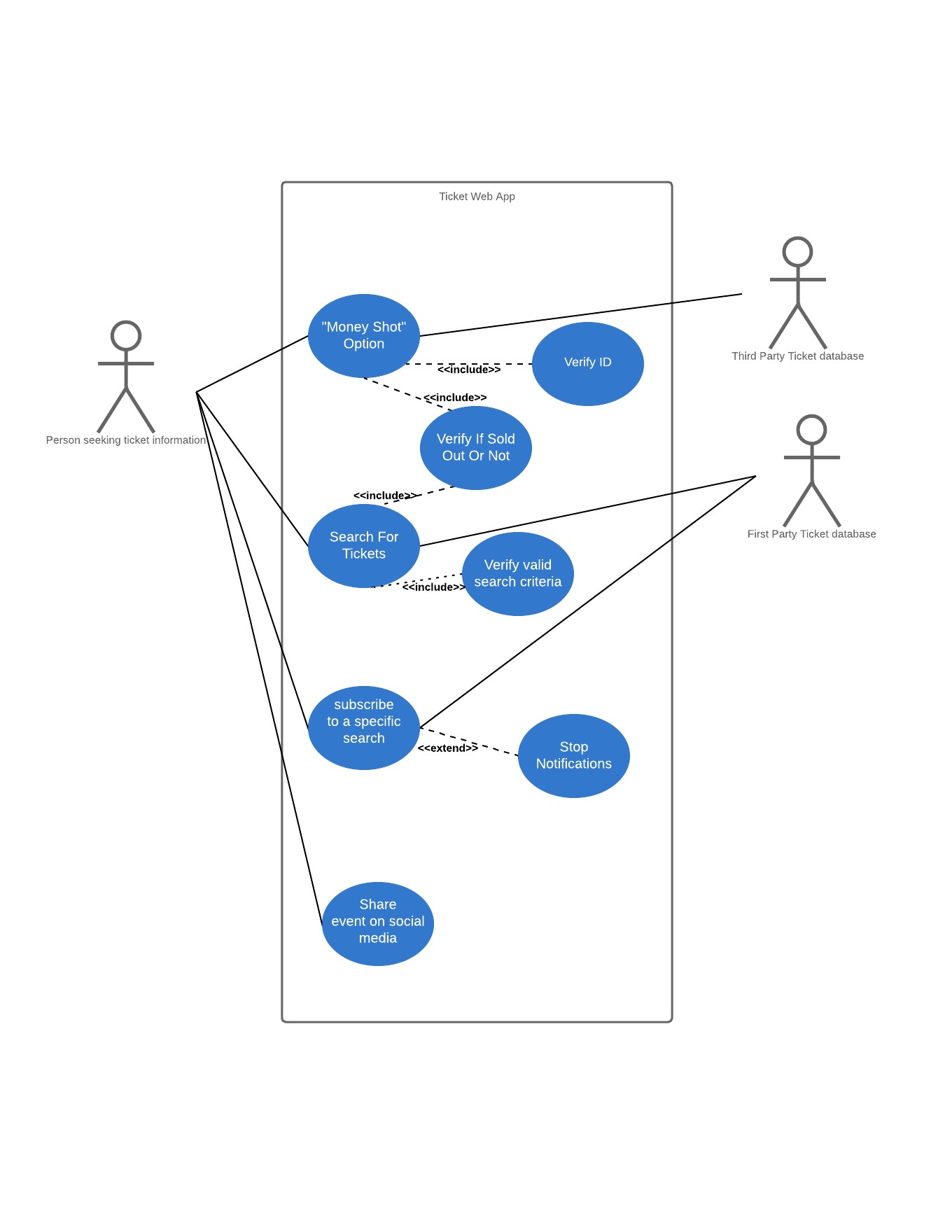
* + If new events get added that matches the specific artist, send a text notification to the user about the new event stating “A new event featuring {artist} just got added” along with the link to the event.
  + If multiple events are added and match that artist, send a text notification to the user stating “New events you might be interested in just got added.”
    - Send a link along with the text that allows the user to view the events (10 max)
      * If there are more than 10 events, allow the user to have ‘view more’ option. Show 10 (max) at each ‘view more’ option.
    - Also allow the user to unsubscribe to the artist by replying “STOP”

1. At each display of event, show availability (sold out or not), prices (general admission, VIP, VIP+, etc.), and official ticket vendor link.
   * If ticket is sold out from the official ticket vendor, offer the user to look up the event in ‘Money Shot’
     + If the user opts in for ‘Money Shot’, directly take the user to the ‘Money Shot’ feature, and provide the event ID the user picked as input for the ‘Money Shot’ feature
2. ‘Money Shot’ request: when requested, the application will also allow users to search up a specific event by ID or name and display the prices of the tickets from different sources.
   * Display ticket prices (general admission, VIP, VIP+, etc.) for retail and third parties along with the link
3. Share Event via. Facebook, Reddit, Twitter, SMS/iMessage, Discord, Snapchat, Email, Copy by Link, WhatsApp.
   * Forwards the event to a friend

**Non-Functional Requirements**

1. Product Requirement: TicketTeam shall be available to the user at all times.
2. Organizational Requirements: Users will provide their phone numbers when they wish to subscribe to receive notifications for events pertaining a specific artist.
3. External Requirements: The subscriptions will not be shared.
4. Efficiency: Results should be delivered under one tenth of a second.
5. Dependability: The TicketTeam application can recommend events to the user at all times.
6. Security: Phone numbers are encrypted in the database
7. Usability: Color-blindness option.
8. Performance: Search requests must be made in one millisecond otherwise, timeout.
9. Space: Very minimal space usage because we are only storing 3 attributes per user
10. Environmental: We will be using highly efficient servers. Our web platform is lite therefore it will not be highly demanding on the user’s device (low-power).
11. Operational: We allow users to review our application by sending reviews to our customer satisfaction email account.
12. Development: We will have system engineers that will be working on the product everyday until the launch.
13. Regulatory: Making sure all of our third party ticket vendors are still trustworthy every week
14. Ethical: Providing accurate ticket prices.
15. Legislative: TicketTeam will be protected by copyright.
16. Accounting requirements: Our app is free to use, but we would run ads to generate revenue
17. Safety/Security: All ticket recommendations are from trusted vendors.

**5. Use Case Diagram**

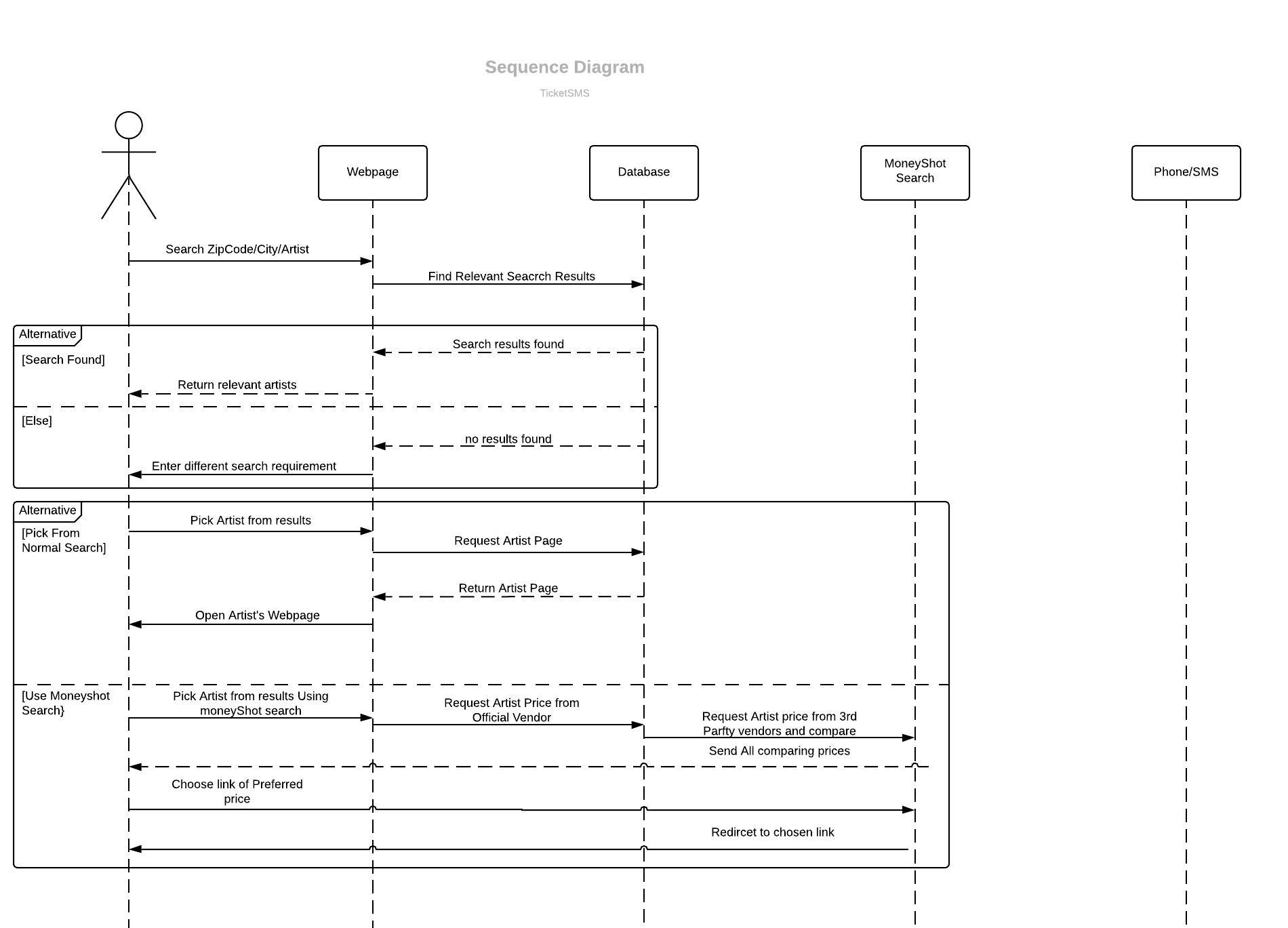
****

**6. Sequence Diagrams**

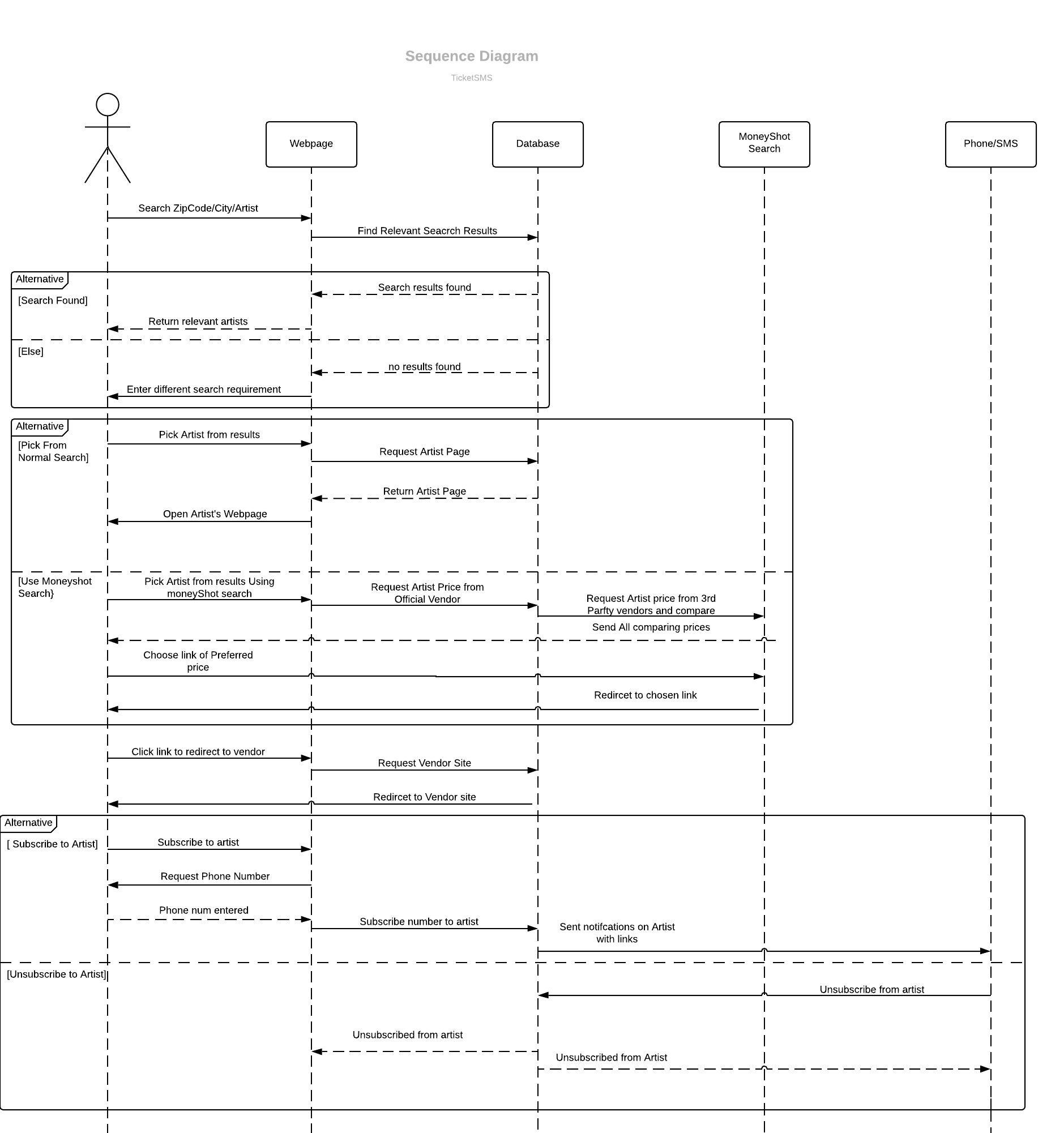
In this iteration of our program we have two specific use cases where we can classify our users in.

The first case is where our user is either a new user or a user who wishes to be a guest and not sign up for the account. In this case the user will still be able to use our website and the Money shot search engine, but will not be able to subscribe to specific artists.

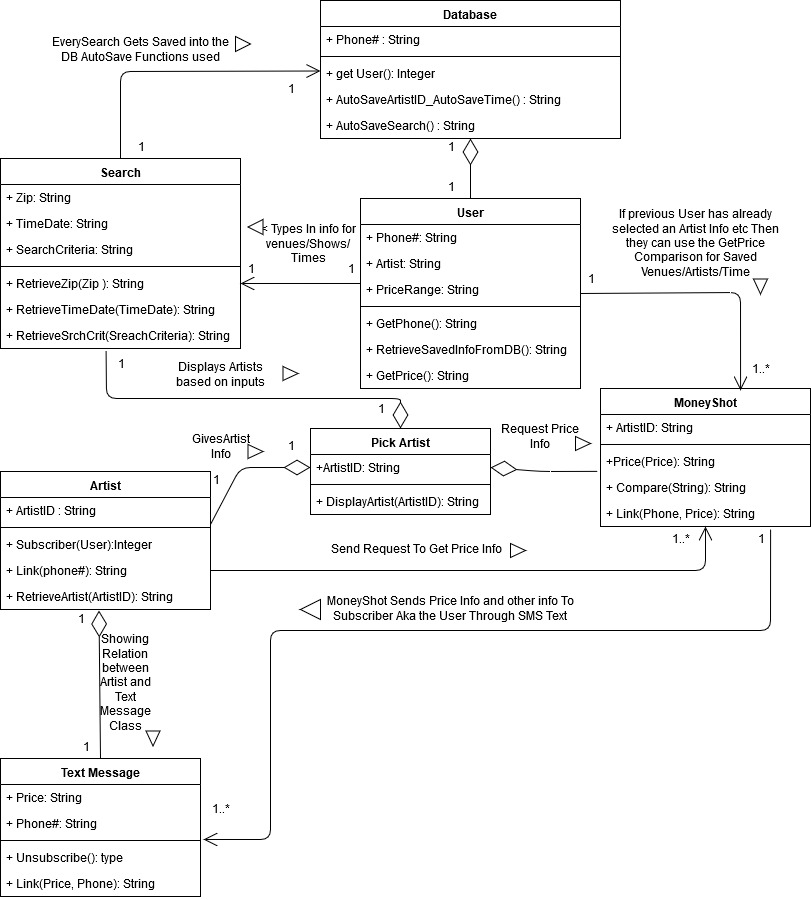
The second use case is where the user subscribes to their artist of choice and our service by just using phone number. They can access all the features our service artists including the Money Shot search engine. The user can also unsubscribe to any artist through the text they receive.



**Fig 6.1)** Use case diagram where user cannot access the subscribe option but still use our Money Shot search engine.

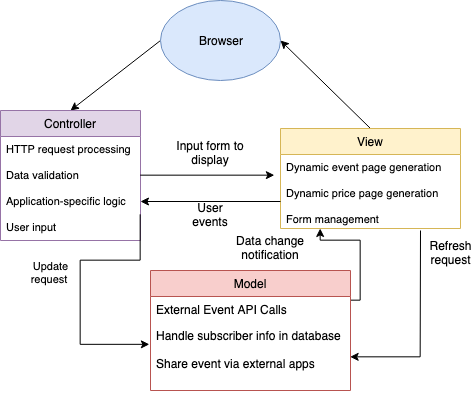
**  
Fig 6.2)** Use case diagram where user can access the subscribe option and use our Money Shot search engine. They can also subscribe from the system or artist through their phone.

**7. Class Diagram**

****

**8. Architectural Design**

**Web App MVC Diagram**

****

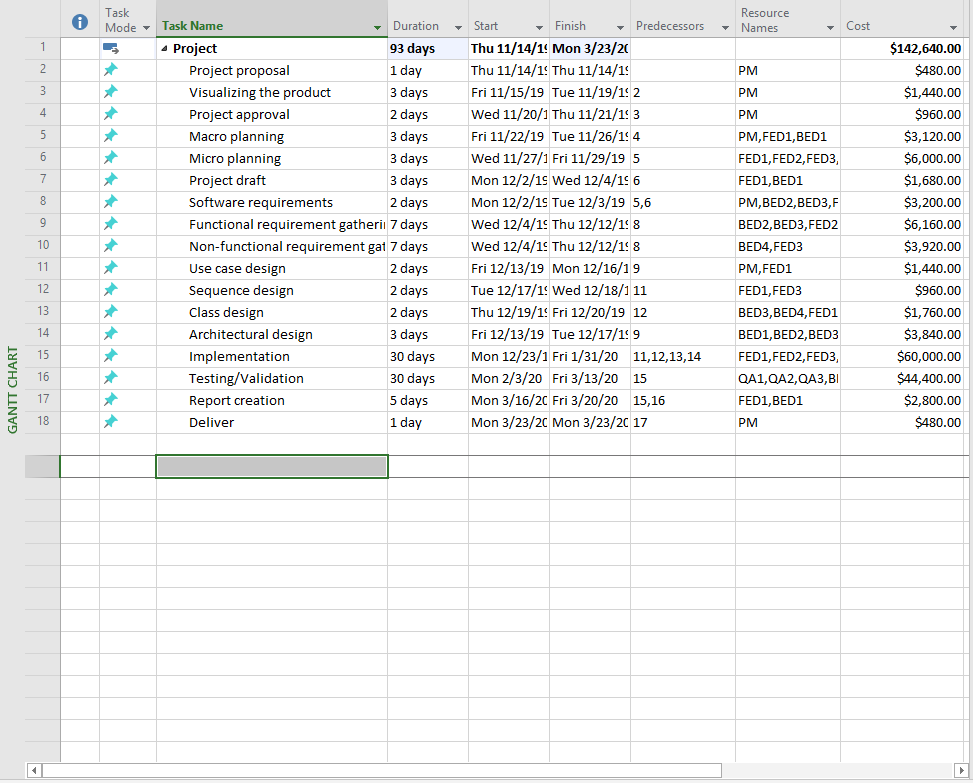
**Deliverable 2:**

**3. Project Scheduling, Cost, Effort and Pricing Estimation**

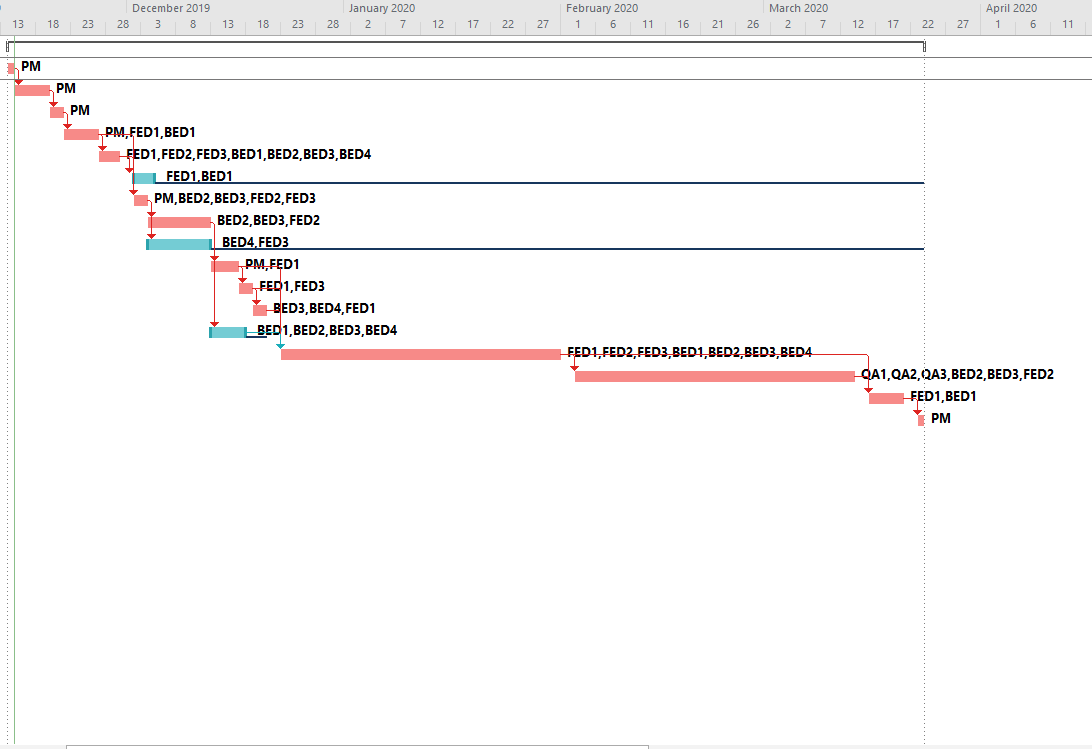
**3.1 Project Scheduling**

Our project is a small scale web based service for that reason we decided that the workforce we needed would have 1 project manager who would overlook the project from start to finish and uphold the vision of the project. Next we have a total of 7 engineers, 4 back end developers and 3 front end developers. Their job is to execute the vision and code the entire project. Backend developers will deal will handle the systems and logic of the project while the front end would focus on the UI and the Human interaction element with the software. Lastly we have a Quality Assurance team consisting of 3 QA testers. Their job is to test the overall system and help the developers polish the final product and iron out any unexpected or unwanted behavior.

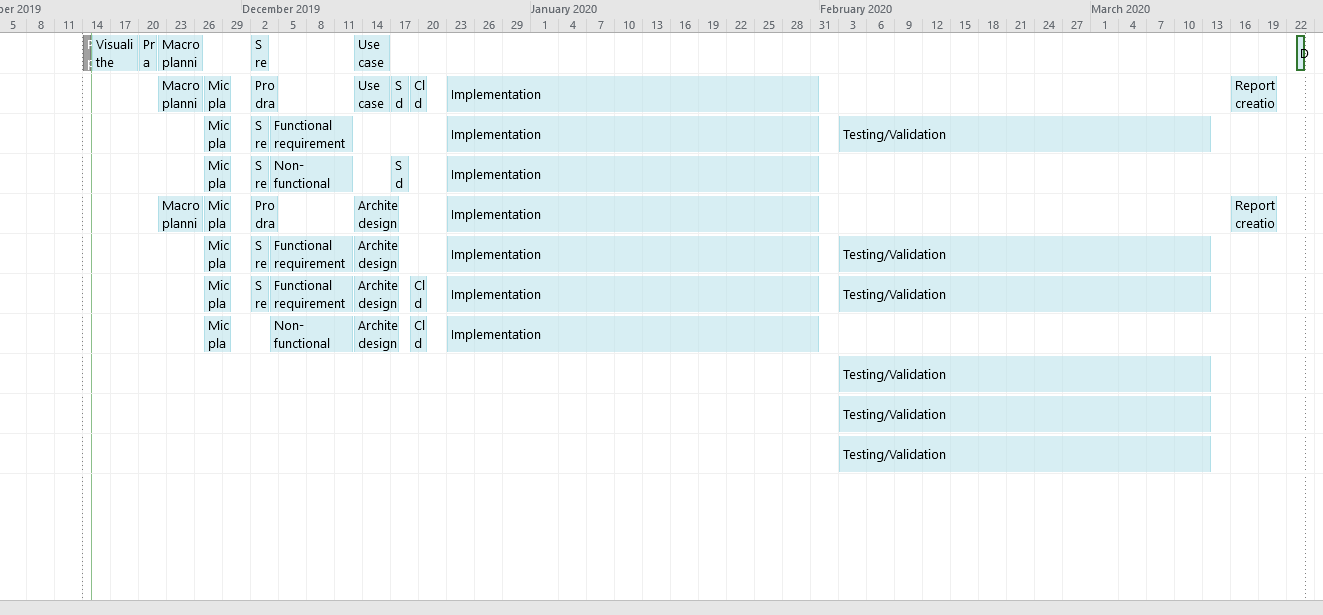
Below are detailed tables and reports that go in detail on what task each member has been assigned, their overall work hours and hourly pay. The overall cost of the project can also be found along with the cost report. To calculate the cost we assumed that we are paying each individual according to their national average salary. For a project manager the average salary is $88,600. For a back end developer the average salary is $113,524. For a Frontend developer the average salary is $86,178 and finally for a QA tester the average salary is $61,072. All of these were found on glassdoor with a sample size for each field being over one thousand. Finally a timeline is also available and the project would go for an overall 93 days.

Project Overview:

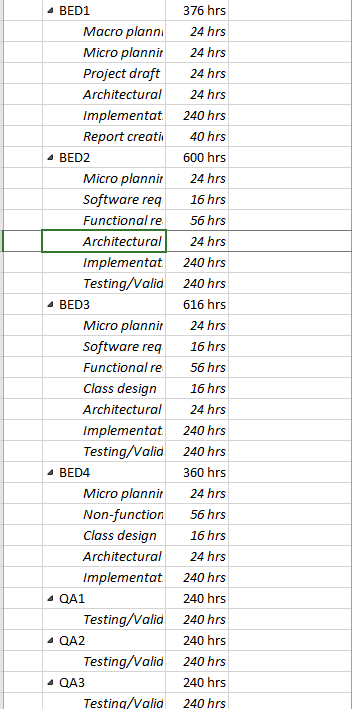
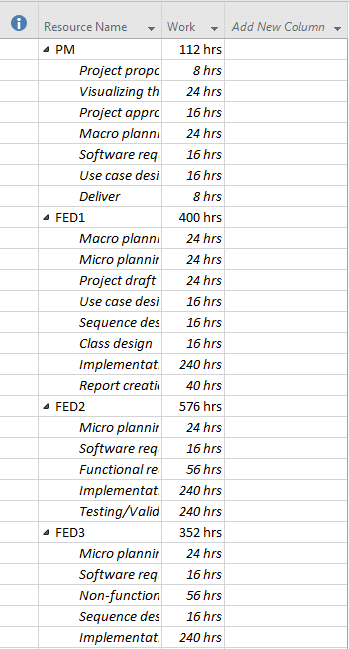
Gantt chart with critical path:

****

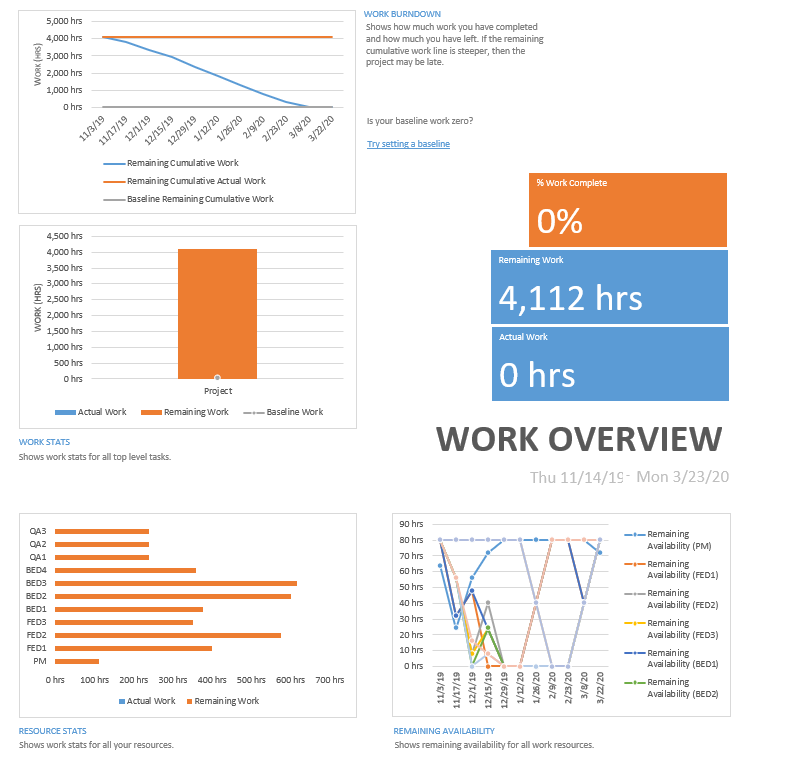
Team Planner:

****

Resource Table:

****

Work Report:

****

**3.2 Function Point Estimation**

We chose function point estimation as our choice of algorithmic estimation method to calculate the effort that is needed for the project. The function point estimation will give us the estimated effort that is required per person. We chose to use the function point estimation because it is simple, easier to read, and fits best with the overall simplicity of our project. Now that we have the effort required per person per week we can see based on the length of our project a rough estimate of what the cost of the entire project would be based on their salaries. The process of determining the effort required is fairly simple. First we make a table with the specified function categories and then attribute a count to each of those functions. Then from that we can compute the gross function point. After that we determine the processing complexity by assessing our answers to 14 different criterias. Then from the processing complexity and gross function point we can easily get all the other calculations done for our estimations. Additionally, we also used the lines of code estimations to estimate the total cost of the software of the project. This will give us a rough estimate of the total cost of the code that needs to be written. We can compare these two pricing estimates to get an even better understanding of what our budget looks like.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Function Category** | **Count** | **Simple Complexity** | **Average Complexity** | **Complex complexity** | **Count \* complexity** |
| **1** | **Number of user input** | **4** | **3** | **3** | **1** | **12** |
| **2** | **Number of user output** | **4** | **3** | **5** | **9** | **12** |
| **3** | **Number of user queries** | **10** | **10** | **2** | **10** | **100** |
| **4** | **Number of data files and relational tables** | **20** | **10** | **14** | **10** | **200** |
| **5** | **Number of interfaces** | **4** | **4** | **9** | **6** | **16** |
|  |  |  |  |  | **GFP** | **340** |

**Determining PC complexities**

1. = 3 average
2. = 5 essential
3. = 3 average
4. = 3 average
5. = 3 average
6. = 3 average
7. = 4 significant
8. = 4 significant
9. = 3 average
10. = 2 moderate
11. = 4 significant
12. = 3 average
13. = 3 average
14. = 3 average

PCA = 0.65 + 0.01(3 + 5 + 3 + 3 + 3 + 3 + 4 + 4 + 3 + 2 + 4 + 3 + 3 + 3) = 1.11

GFP = (4\*3) + (4\*3) + (10\*10)+(20\*10)+(4\*4)=340 fp

FP = GFP \* PCA = 340 \* 1.11 = 377.4 fp

E=FP/productivity = 377.4/60 = 6.29 about 7 person-weeks

D=E/team size = 7/11=0.63 about 1 week

Calculate estimated cost based on function point:

Estimated Cost based on effort = 7 \* 40 \* (36.25/per hr \* 24hr) = 243,600

From using lines of code estimation we can get:

|  |  |
| --- | --- |
| Function | Estimated LOC |
| User Interface | 1500 |
| TicketmasetMaster API functionality | 1000 |
| Error checking | 1000 |
| Design | 1500 |
| Database management | 1000 |
| Estimated lines of code | 6000 |

Assuming the average productivity of the workers is 250 LOC/pm

Pm = 7 person-week

Labor rate of Frontend and backend developers is: $30 per hour $1200 per week about $5 per line of code

Total Estimated Project Cost for code = $5 \* 6000 = $30,000

**3.3 Estimated Cost of Hardware products:**

Amazon EC2 a1.2xlarge Instance: $0.204 per Hour per instance [1]

Computers: $600 per person

**3.4 Estimated Cost of Software products**

Twilio Text Message API: $0.00075 per message [2]

TicketMaster API: $0 [3]

Free and Open Source software:

Visual Studio Code: $0

Apache Jenkins: $0

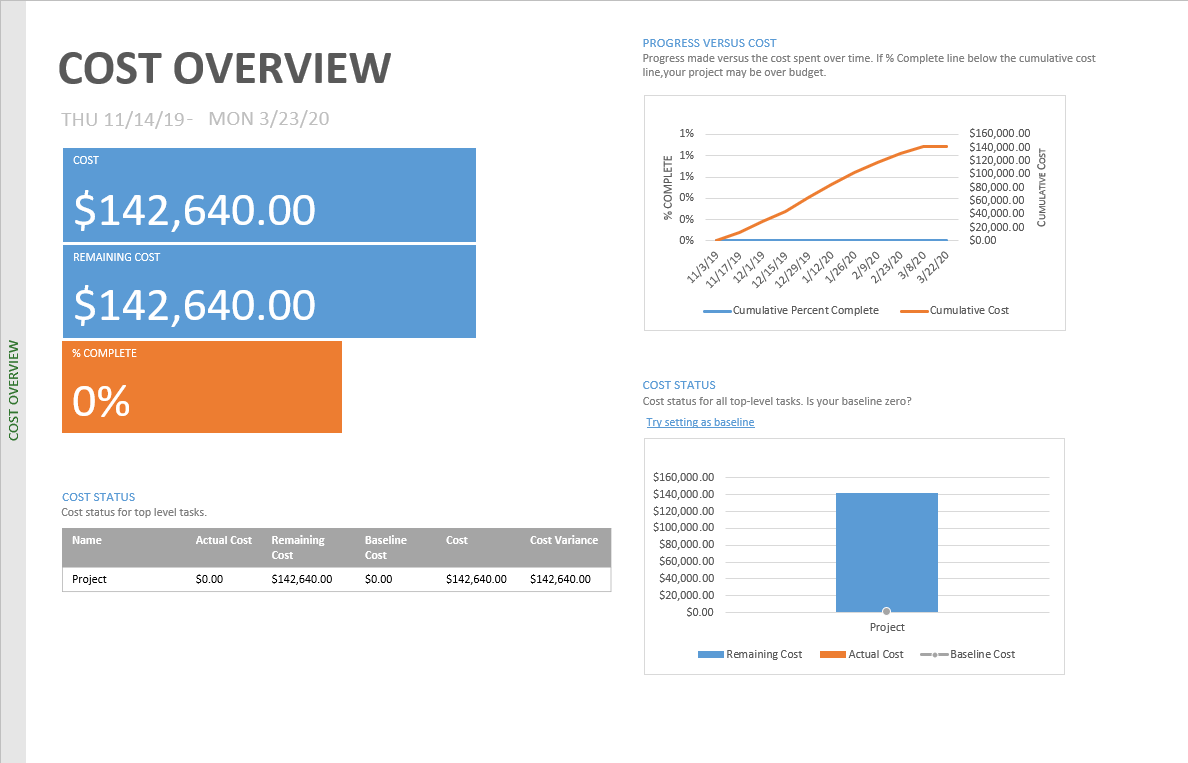
Eclipse: $0

^-- The max requests per day using the Discovery Api is 5000, but however the requests per second is 5 max.

We will use a table from the Discovery Api which has the prices table and the offers table. This will allow us to actually retrieve prices and if a match occurs for the querying we can use the offers table to see if there is a discount available or a better pricing for the venue comparison.

**3.5 Estimated Cost of personnel**

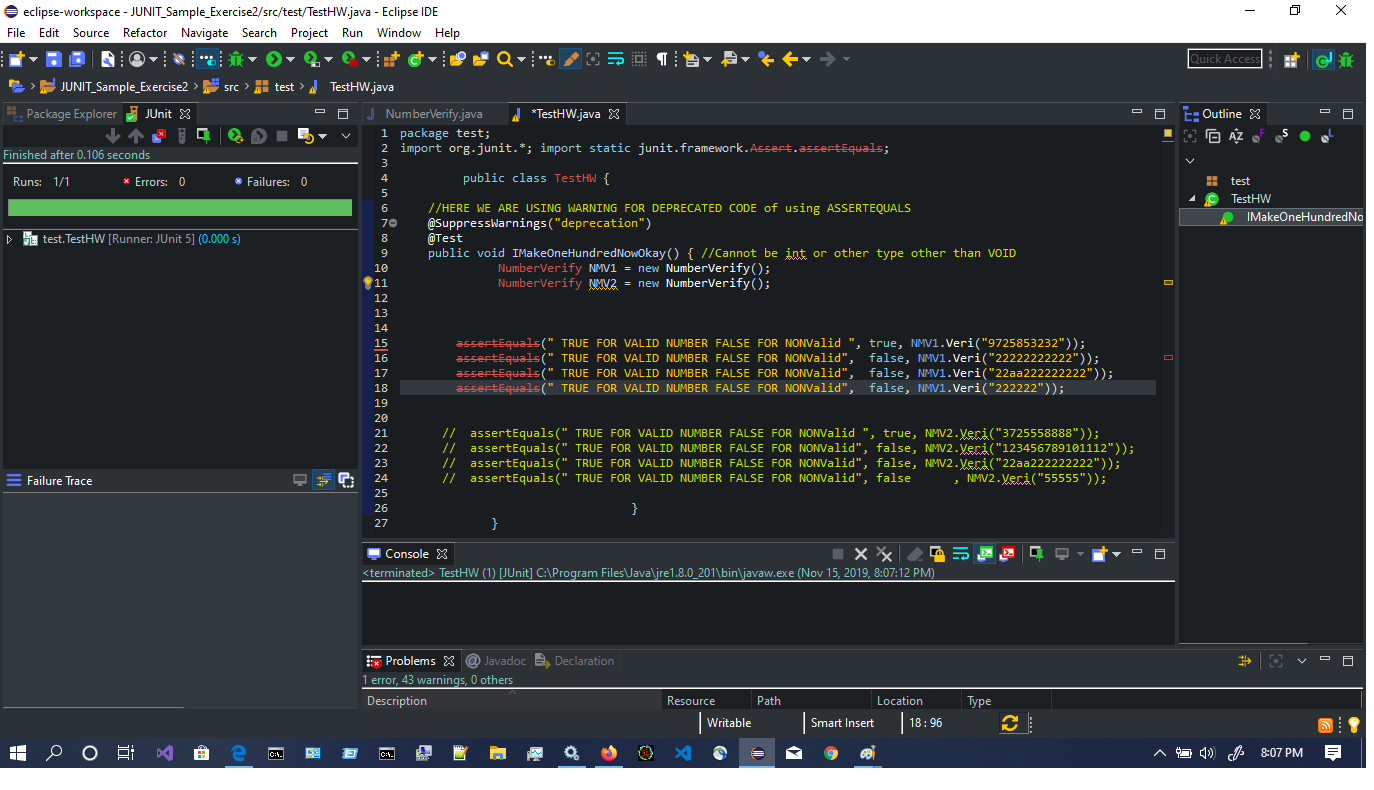
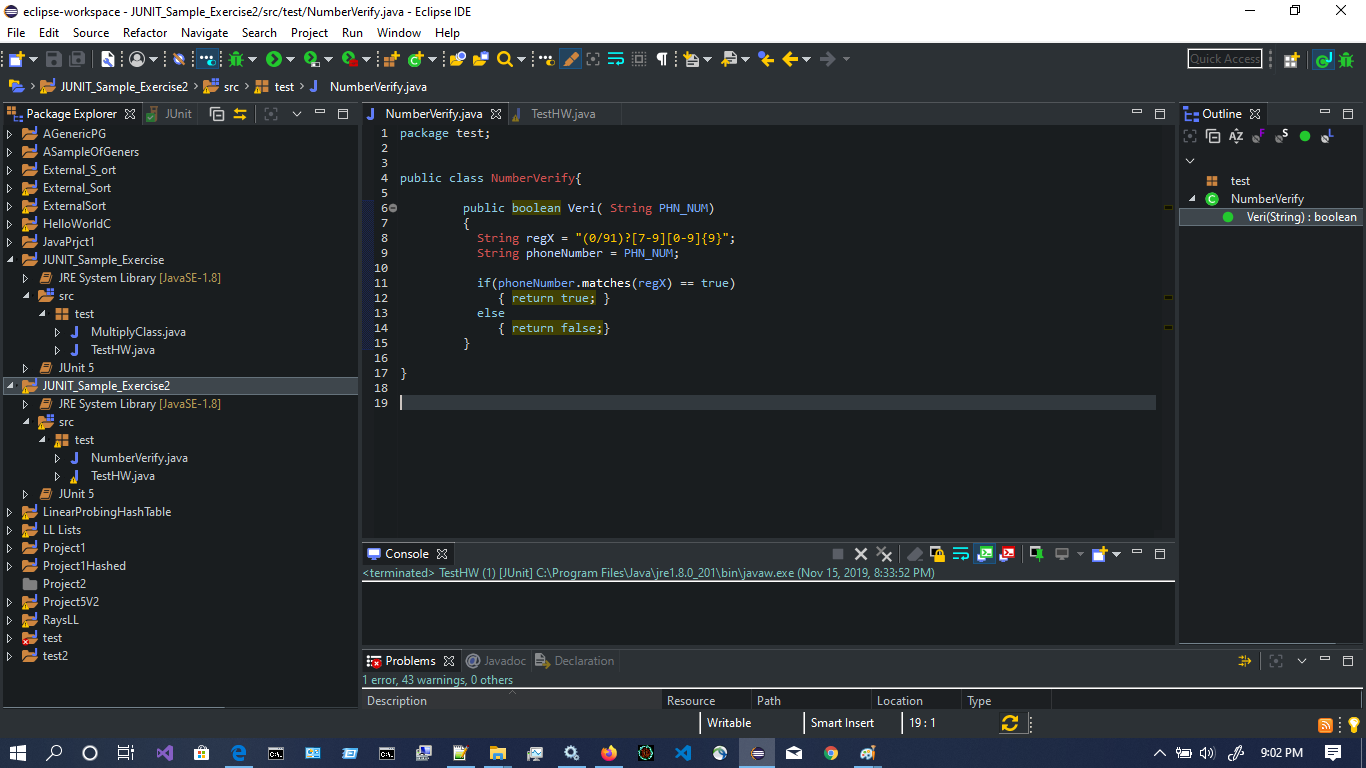
Cost Report

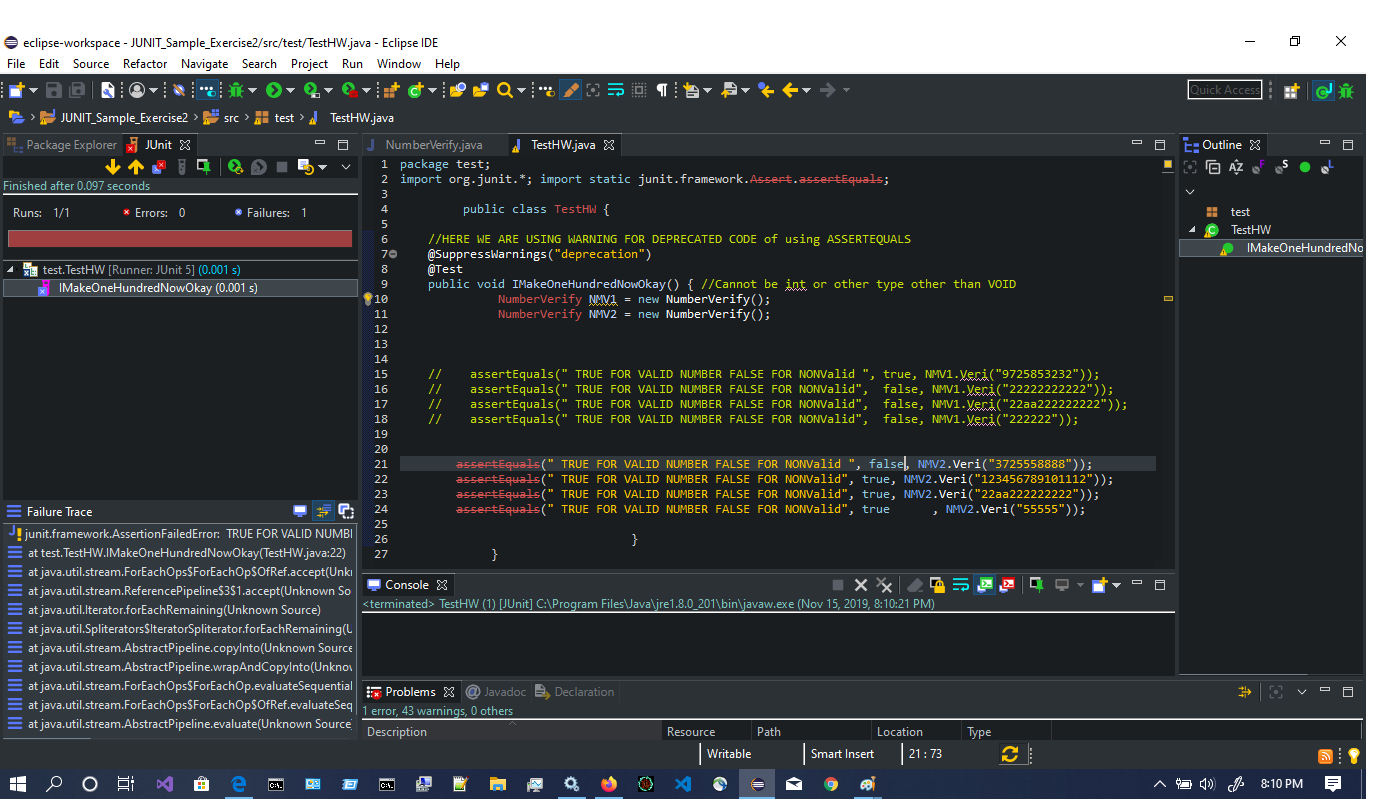
****

|  |  |
| --- | --- |
| **Job** | **Rate** |
| **Project Manager** | **$60/hr** |
| **Front End Developer 1-3** | **$20/hr** |
| **Back End Developer 1-4** | **$40/hr** |
| **QA1-3** | **$25/hr** |

[4,5,6,7]

**4. Test Plan**

****

****

For the test case we tested a string for a valid phone number. The reason we are showing this as a test is because it is an integral part for many of our systems, such as the subscribe function. The first case checks for valid phone numbers and it checks them all correctly so the overall J-unit test passes. However in the second part of the code there are invalid numbers that are marked as true but J-unit test catches the invalid numbers and fails the overall test. For us this validation is important as it will not accept a bad number so a customer can make a mistake and not have a faulty phone number in the system. Also it protects against spam bots that try to spit all kinds of illegal entries into the system to either to slow it down or damage it.

**5. Comparison of Work**

Interestingly, while we are using Ticketmasters API to power our website it doesn't naturally compete with ticketmaster instead it actually finds a comfortable space in between third party ticket vendors and first part vendors. The purpose of our website is not necessarily to become the one stop shop for selling tickets, rather its to inform the customer and keep them constantly updated on the best prices available at anytime. While acting as a middle man is not necessarily a novel concept, it still stands on its own due to the moneyshot search engine. An example of a system similar to ours is actually spotify, they act as a middle-man for artists to advertise their concerts and redirect customers to the official vendors site. However they do not let you subscribe to specific artists concerts and it is hard to navigate as the concert information is quite hidden in multiple menus.So we believe our system to be more evolutionary in nature rather than in iterative upgradeand it creates its own spacein this ecosystem. [3]

**6. Conclusion**

The process of developing our own web service came with many challenges, hurdles and important decisions. One of the key decisions and challenges was to decide on an API for the foundation of our service. At the end we decided to use the Ticketmaster API due to its open source nature and ease of integration. The next challenge that crept on us as we started getting further in the design process was keeping the user interface with the system simple and easy to understand despite the multi layered software architecture. However, we were able to efficiently communicate with each other, which enabled task distribution and cross examination to check for potential flaws and cohesiveness during software development. Along with this, we also gained a valuable experience during various stages of software designing and learned tools that helped us visualize the functions of our software. Finally, we were able to make an educated cost and development prediction which helped set a timeline and the resources needed to be allocated in order to develop TicketSlackers.

**7. References**

[1] D. J. Daly and D. J. Daly, “Economics 2: EC2,” Amazon, 1987. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/. [Accessed: 14-Nov-2019].

[2] “SMS Pricing for Text Messaging,” Twilio. [Online]. Available: https://www.twilio.com/sms/pricing/us. [Accessed: 14-Nov-2019].

[3] “Getting Started,” *The Ticketmaster Developer Portal*. [Online]. Available: https://developer.ticketmaster.com/products-and-docs/apis/getting-started/. [Accessed: 14-Nov-2019].

[4] “Salary: Project Manager,” *Glassdoor*. [Online]. Available: <https://www.glassdoor.com/Salaries/project-manager-salary-SRCH_KO0,15.htm>.

[5] “Salary: Backend Developer,” *Glassdoor*. [Online]. Available:   
<https://www.glassdoor.com/Salaries/backend-developer-salary-SRCH_KO0,17.htm>

[6] “Salary: Frontend Developer,” *Glassdoor*. [Online]. Available:   
<https://www.glassdoor.com/Salaries/front-end-developer-salary-SRCH_KO0,19.htm>

[7] “Salary: QA Tester,” *Glassdoor*. [Online]. Available:   
<https://www.glassdoor.com/Salaries/qa-tester-salary-SRCH_KO0,9.htm>