**CS3354 Software Engineering**

**Final Project Deliverable II**

**Trek Travel App**

Alejandro Amaya Giron,

Churong Zhang,

Xizhen Yang,

Kaitian Li,

Jisoo Kim,

Kent G,

Frank Cazarez

Table of Contents

[**Task Delegation**](#_6jjf9hhp97ia)**…………………………………………………………………………………………... 3**

[**Project Deliverable I Content**](#_zhzmjbe0q5as)**…………………………………………………………………………… 4**

[Introduction](#_z2kqoow1drv1)………………………………………………………………………………………….. 4

[Motivation](#_lbzakahjd7fn)…………………………………………………………………………………………... 4

[Final Project Draft Feedback](#_tnnwh0ns6xws)………………………………………………………………………... 4

[Team GitHub Repository](#_yayotdape9v8)……………………………………………………………………………. 6

[Task Delegation](#_9v3cibhipy6q)……………………………………………………………………………………... 6

[Software Process Model](#_uf68v5wd0e8r)…………………………………………………………………………….. 7

[Software Requirements](#_c6lp7z6olzbp)…………………………………………………………………………….... 7

[Functional Requirements](#_4602n6guk0u) ………………………………………………………………………... 7

[Non-functional Requirements](#_xvwx6atvd7pc)……………………………………………………………………. 8

[Use case diagram](#_1h8eppilycw)…………………………………………………………………………………….. 9

[Sequence Diagram](#_fbfvhlspn1en)……………………………………………………………………………………. 10

[Class diagram](#_vyh1v1irliya)………………………………………………………………………………………... 12

[Architectural Design](#_a8lscb9yiaa0)…………………………………………………………………………………. 15

[**Project Scheduling**](#_dp8omld0p8c9)**………………………………………………………………………………………. 16**

[**Cost Estimation**](#_qh4zcck30nge)**…………………………………………………………………………………………. 16**

[Function Point (FP) Method](#_53lgvjrn4aqz)…………………………………………………………………………. 16

[Hardware Cost](#_3jw03ghaosxs)……………………………………………………………………………………….. 20

[Software Cost](#_d693kmm9m6iu)………………………………………………………………………………………... 20

[Personnel Cost](#_8etc4v8urgdh)………………………………………………………………………………………. 22

[**Test Plan**](#_6imdm5t681pf)**………………………………………………………………………………………………….. 22**

[**Comparison**](#_ktuex3veromr)**…………………………………………………………………………………………….... 22**

[**Conclusion**](#_jfu8wl388gj2)**……………………………………………………………………………………………….. 23**

[Challenges](#_2xshlkolvqk9)……………………………………………………………………………………….. 23

[Achievements](#_xjhjz3bbyuqs)……………………………………………………………………………………. 23

[**Reference**](#_2q6sx1ivkllg)**……………………………………………………………………………………………….... 24**

## **Task Delegation**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Group Member | Task 1 | Task 2 | Task 3 | Task 4 | Task 5 |
| Alejandro Amaya | Proposal – Writing Contents | Deliverable 1 – Use Case Diagram | Deliverable 2 – Report Conclusion | Deliverable 2 -  PowerPoint Slides |  |
| Frank Cazarez | Deliverable 1 – Class Diagram | Deliverable 2 – Software Test Plan | Deliverable 2 - Report | Deliverable 2 -  PowerPoint Slides |  |
| Kent Grossling | Deliverable 1 – Software Processing Model |  |  |  |  |
| Jisoo Kim | Proposal – Template and Format | Deliverable 1 – Report | Deliverable 2 – Comparison with Similar Software Design | Deliverable 2 -  PowerPoint Slides | Deliverable 2-  Report |
| Kaitian Li | Deliverable 1 – Sequence Diagram | Deliverable 2 – Project Scheduling | Deliverable 2 – PowerPoint Slides | Deliverable 2-  Junit Test Code | Deliverable 2-  Hardware Cost |
| Xizhen Yang | Proposal – Assign Tasks | Deliverable 1 – Architecture Design | Deliverable 2 – Delegation of Tasks | Deliverable 2 – Cost, Effort, and Pricing Estimation | Deliverable 2 – UI Design |
| Churong Zhang | Github Setup | Deliverable 1 – Software Requirements | Deliverable 2 – Estimated | Deliverable 2 – PowerPoint Slides | Deliverable 2-  Junit Test Code |

## 

## **Project Deliverable I Content**

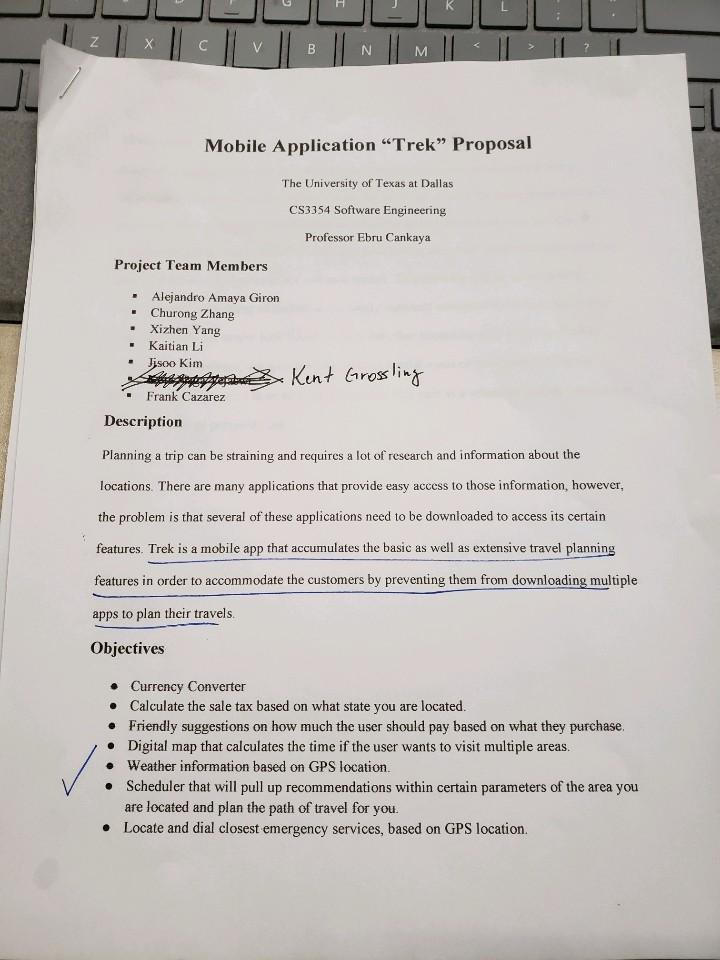
### **Introduction**

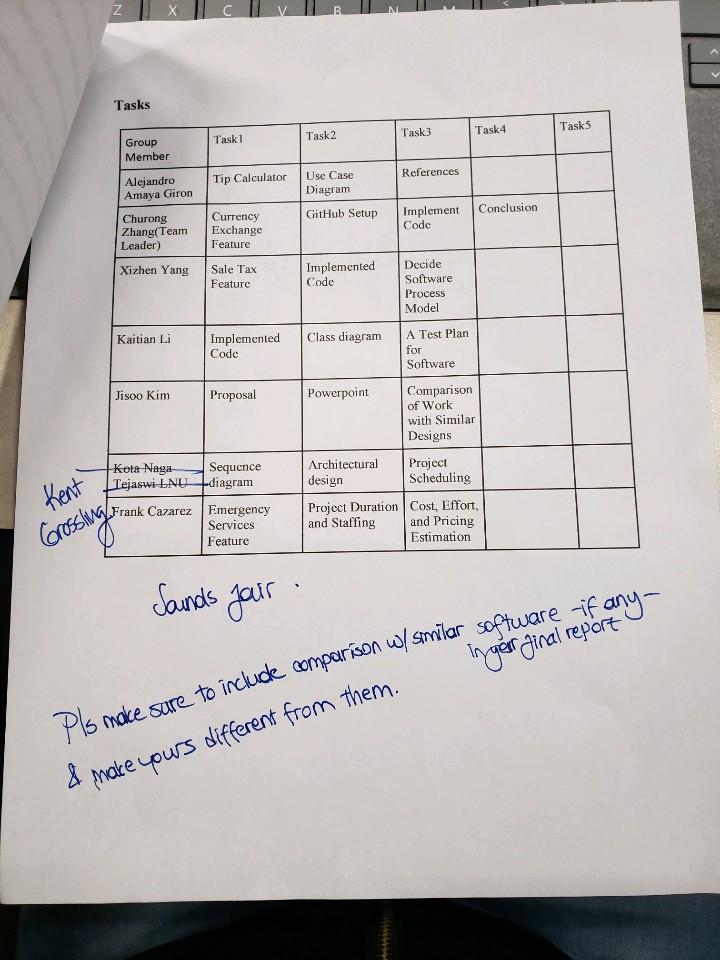
Planning a trip can be straining and requires a lot of research and information about the locations. There are many applications that provide easy access to that information. However, the problem is that several of these applications need to be downloaded to access its certain features. Trek is an app that accumulates the basic as well as extensive travel planning features in order to accommodate the customers by preventing them from downloading multiple apps to plan their travels.

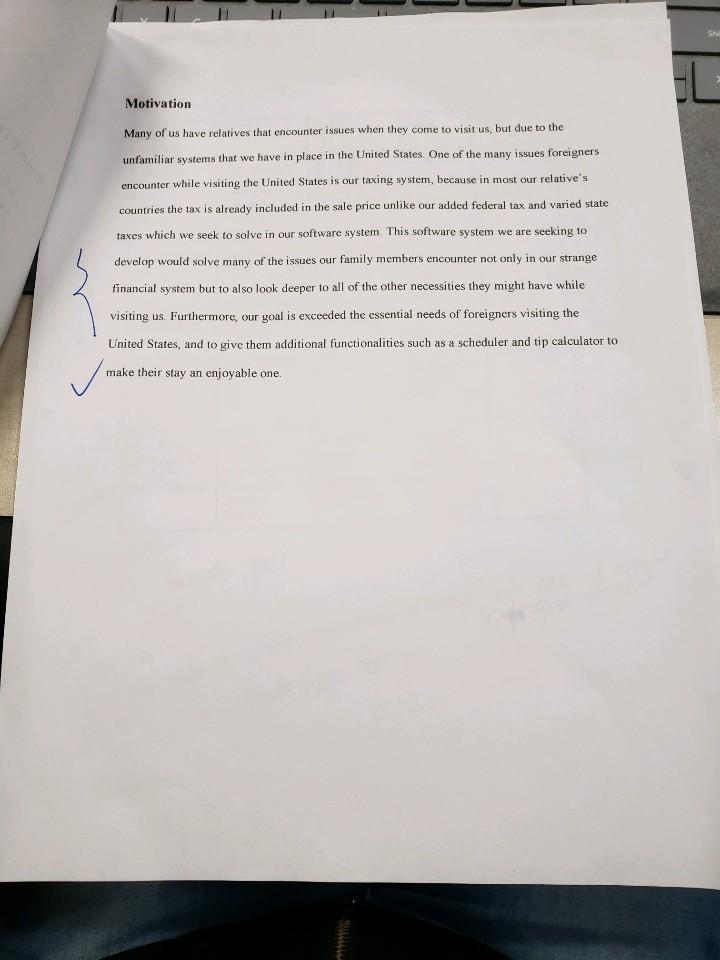
### **Motivation**

Many of us have relatives that encounter issues when they come to visit us, but due to the unfamiliar systems that we have in place in the United States. One of the many issues foreigners encounter while visiting the United States is our taxing system, because in most our relative’s countries the tax is already included in the sale price unlike our added federal tax and varied state taxes which we seek to solve in our software system. This software system we are seeking to develop would solve many of the issues our family members encounter not only in our strange financial system but to also look deeper to all of the other necessities they might have while visiting us. Furthermore, our goal is exceeded the essential needs of foreigners visiting the United States, and to give them additional functionalities such as a scheduler and tip calculator to make their stay an enjoyable one.

### **Final Project Draft Feedback**







Based on the feedback provided, we were to include comparison with similar software and make it different from those applications. With many people traveling all around the world, there are many travel assistant applications such as Airbnb, Expedia, Google Maps, Translator, etc. We wanted to make our application different from them by essentially combining all the features together. Applications such as Airbnb allows users to look for hotels and houses to book but does not provide ways to calculate area currencies, weather, and other extensive features. Trek application will stand out differently by combining all extensive travel assistance features into a single app.

### **Team GitHub Repository**

<https://github.com/CZhang1997/3354-Trek>

### **Task Delegation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Assignee** | **Description** | **Status** |
|  |  |  |  |
| **Create GitHub Account** | All members |  | Complete |
| **Create GitHub repository** | Churong Zhang | Create repository with 3354-Trek | Complete |
| **Add team members and TA** | Churong Zhang | Add all member and TA | Complete |
| **Create a README file** | Kaitian Li | Create a README file | Complete |
| **Include GitHub url** | Jisoo Kim | Ensure all information in Github | Complete |
| **Make “project\_scope” commit** | Xizhen Yang | 1.5 project scrope | Complete |
|  |  |  |  |
| **Research software process model** | Kent G | Find which software model should be implemented | Complete |
| **List software requirements** | Churong Zhang | Include functional and non-functional requirements | Complete |
| **Create use case diagram** | Alejandro Giron | More than one use cases | Complete |
| **Create sequence diagram** | Kaitian Li | Individual diagram for each use case | Complete |
| **Create class diagram** | Frank Cazarez | Includes all classes of your project | Complete |
| **Architectural Design** | Xizhen Yang | Choose one and apply to project | Complete |
|  |  |  |  |
| **Write Report** | Jisoo Kim | Organize all contents and write deliverable | Complete |
| **Evaluate Report** | All members |  |  |

### 

### **Software Process Model**

So, in our project we chose to go with the spiral software project model because of both its iterative and controlled systematic qualities. We believe that given the nature of our product it will suit us perfectly. We want to be able to deliver a base platform for trial purposes that can give a feel for what we are trying to accomplish all while being able to go back and update by adding more features that will benefit the user as well as our company. We also know that our target industry is very dynamic and as such we want to be able to constantly adapt all while remaining reliability. The spiral model is perfect for just that as it will allow for later updating to current platforms without failing to guarantee that each deliverable is reliable and ready to use. Travel is a dynamic industry by nature, not only because technology is always adapting and changing but also because technology is adapting and changing at different rates in different countries, and we have to be able to accommodate anyone from anywhere. Spiral model, we believe, offers the best platform to accommodate everyone everywhere. Not only will this model help ensure that we can develop the software that we are seeking to create but it will also help us learn what the consumers liked about what we’ve created and what they feel is lacking. It will allow us to seek and respond to feedback. Considering our app will be all about helping end users achieve their goals and facilitate their travels, we are excited to be able to help them and get even better at helping them as we go. We also go into this with the knowledge that where we start is not where we will end up. We would be disappointed if our end product is not vastly different and evolved from where it first started, which is precisely why we love the evolutionary aspect of the spiral model.

### **Software Requirements**

#### **Functional Requirements:**

* Currency converter
* Calculate the sale tax based on the city
* Tips suggestions on various services based on user’s preference
* Local weather information
* Distance and time calculator if you want to visit certain locations
* Scheduler that will pull up recommendations within certain parameters of the area you are located and plan the path of travel for user
* Locate and dial closest emergency services based on GPS location

#### **Non-functional Requirements:**

* Currency converter should support up to 166 different currencies, and must be real time exchange rate
* Provide sales tax information for at least more than 100 cities in United States
* Tip suggestion should be simple to use
* The distance and arrival time calculation should take the traffic and other automotive incidents into consideration
* Local weather should have current weather, and up to 5 days advance
* Scheduler should save user activity and travel log permanently
* Emergency services should be easy to access and should be located easily within the app

### **Use case diagram**

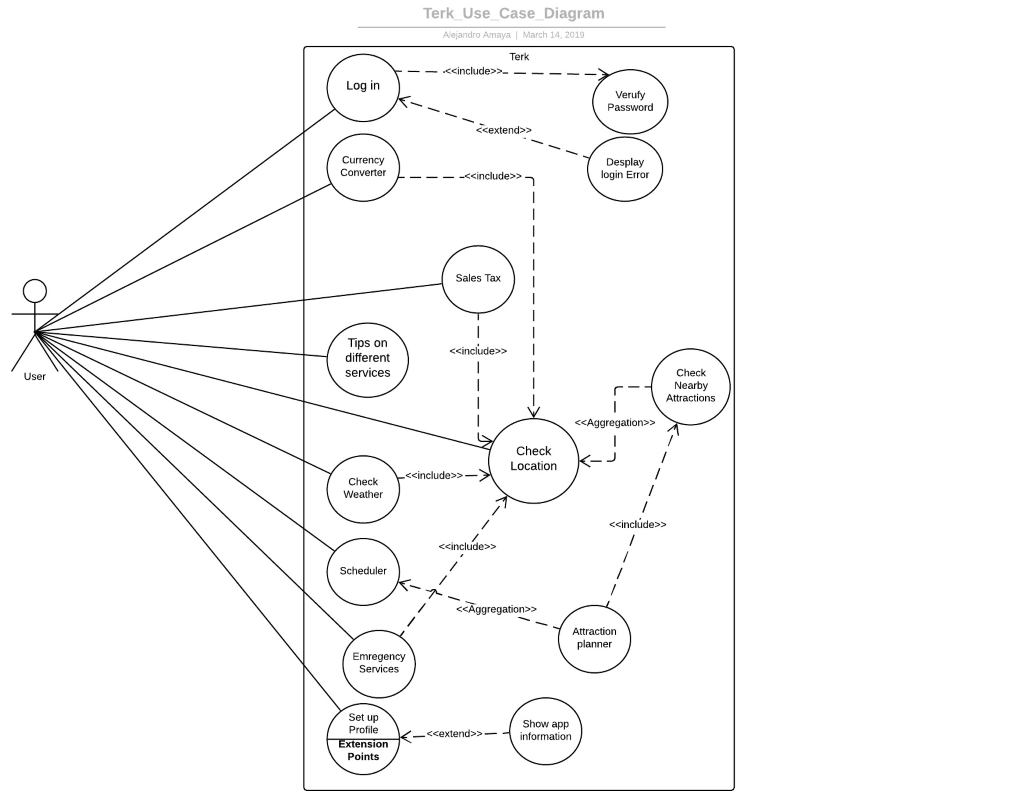
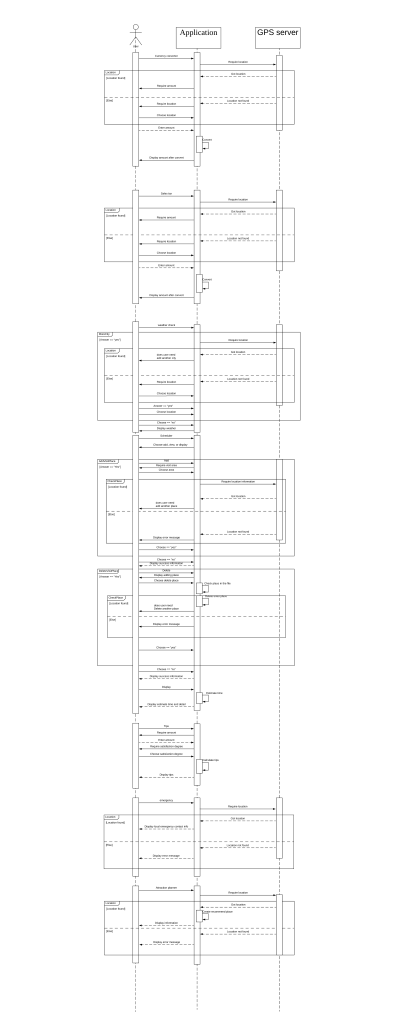


Fig 1. Use Case Diagram

**Sequence Diagram**:

### 

### **Class diagram**

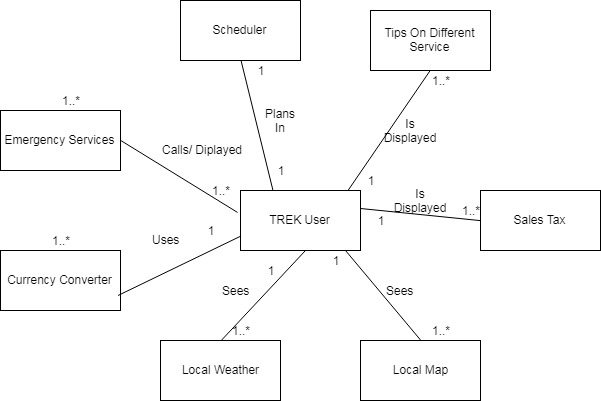


Fig 3. Class Diagram

|  |
| --- |
| **TREK User** |
| - Time : Date  - Name: String  - Email : String  - Phone : int  - Username: String  - Password: String |
| +getTime()  +getName()  -getEmail()  -getPhone()  -getUser()  -getPassword() |

|  |
| --- |
| **CurrencyConverter** |
| +currentCurrency: double  +convertedCurrency: double |
| +setCurrency(currentCurrency: double)  +setconvertedCurrency(convertedCurrency: double)  +convert(currentCurrency: double, convertedCurrency: double)  +getConverted() |

|  |
| --- |
| **SalesTax** |
| +Currency: double  +Zip: int  +City: String  +Tax: double |
| +setCurrency(Currency: double)  +setZip(Zip: int)  +setCity(City:String)  +getTax1(City:String)  +getTax2(Zip:int)  +AddTax(Currency : double, Tax: double)  +getFinal() |

|  |
| --- |
| **Tip** |
| +Currency: double  +Service: String  +Tip: double  +Satisfaction: Boolean  +Recommend: int |
| +setCurrency(Currency: double)  +setService(Service:String)  +setSatisfaction(Satisfaction:boolean)  +setTip(Service:String, Satisfaction:boolean, Recommend: int )  +AddTip(Currency : double, Tip: double)  +getFinal() |

|  |
| --- |
| **LocalMap** |
| +Location: MapAPI  +Attractions: String |
| +setLocation(Location:MapAPI)  +Route()  +setAttractions(Attractions:String)  +getRecommendation(Location:MapAPI, Attraction: String) |

|  |
| --- |
| **LocalWeather** |
| +Location: WeatherAPI  +Update: boolean |
| +setLocation(Location:WeatherAPI)  +getWeather(Location:WeatherAPI)  +getWeatherUpdate(Location:WeatherAPI, Update: boolean) |

|  |
| --- |
| **Scheduler** |
| +Plans: String  +Events: String  +Reminder: String  +Important: String  +Day: Date |
| +AddPlans(Plans: String)  +AddEvent(Events:String)  +AddReminder(Reminder:String)  +AddImportant(Important:String)  +SetDate(Day:Date, Plans:String)  +SetDate(Day:Date, Events:String)  +SetDate(Day:Date, Reminder:String)  +SetDate(Day:Date, Important:String)  +EditPlans(Plans: String)  +EditEvent(Events:String)  +EditReminder(Reminder:String)  +EditImportant(Important:String)  +getPlans()  +getEvents()  +getReminders()  +getImportant() |

|  |
| --- |
| **EmergencyServices** |
| +Location: MapAPI  +Phone: int |
| +LocateNearest(Location:MapAPI)  +Route()  +setPhone(Phone:int)  +DialPhone() |

### **Architectural Design**

**Model-View-Controller (MVC)**

Reason: After the comparison of different software architectural design, MVC is the best fit for our product. Our product is a mobile application which serves its users to gain better travel experience.  Therefore, it must be able to provide optimum user interaction and user experience. The specialty of Model-View-Controller architectural design is that it divides the system into three components, the model, the view, and the controller.

* Model: Model component is in charge of the back-end support of our application. It will contain all the functions that allows multiple operations and various classes and models.
* View: View component is in charge of the user interface, specifically what the user sees visually on the app. It will present the data and other extensive travel information to the user; It contains the code and the design for the interface.
* Controller: Controller component manages the interaction of the application. It generates what to operate based on users’ action and returns appropriate result to the user. The controller bridges the interaction between the user and the software, as well as the model and the view component of the MVC model.

## **Project Scheduling**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Effort(person-days) | Duration(days) | Dependencies |
| A: Requirement gathering | 1 | 1 |  |
| B: Tasks analysis | 2 | 2 | A |
| C: Feasibility study | 2 | 2 | A |
| D: setup software | 2 | 2 | B,C |
| E: User Interface design | 8 | 4 | B,C |
| F: User Interface design implementation | 6 | 3 | D, E |
| G: Feature Implementation | 4 | 2 | D,E |
| H: Define users | 2 | 2 | D,E |
| I: Test and Debug | 25 | 5 | F,G |

## **Cost Estimation**

### **Function Point (FP) Method**

**Functions:**

A. Currency Converter

B. Sale Tax Calculator

C. Tip Estimator

D. Map

E. Weather

F. Trip Planner

G. Local Service Number

Determine function category count:

Total count 44

1. Number of user input: Counts: 11

• Function A:

o Amount needed to be converted

• Function B:

o Amount needed to be calculated

• Function C:

o Original amount

• Function D:

o Destinations (max 5)

• Function E:

o N/A

• Function F:

o Title

o Time

o Place

• Function G:

o N/A

2. Number of user output: Counts: 9

• Function A:

o Converted amount

• Function B:

o Calculated amount

• Function C:

o Suggested tip

o Total amount

• Function D:

o Suggested route

o Time

• Function E:

o Temperature

• Function F:

o Reminder

• Function G:

o Service number

3. Number of user queries: Counts: 9

• Function A:

o Select “from” currency

o Select “to” currency

• Function B:

o Select city

• Function C:

o 3 service survey questions

• Function D:

o Selection of by car/walk

• Function E:

o N/A

• Function F:

o Alert option

• Function G:

o Select Service

4. Number of data files and relational tables Counts: 6

• Function A:

o Real-time currency API

• Function B:

o Sale tax API

• Function C:

o Tip calculation API

• Function D:

o Google Map API

• Function E:

o Weather API

• Function F:

o Local data files

• Function G:

o Google Map API

5. Number of external interfaces Counts: 9

• Function A:

o General UI page

o Currency selection UI page

• Function B:

o General UI page

• Function C

o General UI page

• Function D

o Map UI page

• Function E

o Weather page

• Function F

o Table view

o Detailed view

• Function G

o Service number table view

**Determine complexity:**

Function Category Count Complexity Count \* Complexity

1 Number of user input 11 4 44

2 Number of user output 9 4 36

3 Number of user queries 9 6 54

4 Number of data files and relational table 6 15 90

5 Number of external interfaces 9 10 90

Compute gross function point:

GPF = 44 + 36 + 54 + 90 + 90 = 317

Determine processing complexity (PC):

1. Does the system require reliable backup and recovery?

2 - moderate

2. Are data communications required?

5 - essential

3. Are there distributed processing functions?

2 - moderate

4. Is performance critical?

4 - significant

5. Will the system run in an existing, heavily utilized operational environment?

5 - essential

6. Does the system require online data entry?

5 - essential

7. Does the online data entry require the input transaction to be built over multiple screens or operations?

3 - average

8. Are the master files updated online?

2 - moderate

9. Are the inputs, outputs, files, or inquiries complex?

2 - moderate

10. Is the internal processing complex?

1 - incidental

11. Is the code designed to be reusable?

4 - significant

12. Are conversion and installation included in the design?

1 - incidental

13. Is the system designed for multiple installations in different organizations?

4 - significant

14. Is the application designed to facilitate change and ease of use by the user?

5 - essential

**Compute processing complexity adjustment (CPA):**

PCA = 0.65 + 0.01 \* (PC1 + … + PC14) = 1.1

**Compute function point (FP):**

FP = GFP \* PCA = 317 \* 1.1 = 348.7 FP

**Estimated effort:**

E = FP / productivity = 348.7 / 40 = 8.7175 ~= 9 person-weeks

**Project duration:**

D = E / team size = 9 / 8 ~= 2 weeks

### Hardware Cost

Computer System Requirement:

Windows (XP or later), Linux, or Mac OS

3 GB RAM minimum, 8GB RAM recommended

2 GB disk space (500 MB for IDE, 1.5 GB for Android SDK)

IDE

Eclipse or Android Studio

Android SDK, Java

Mac or Android mobile

Assume programmers have computer for the whole project. Thus, the hardware cost is $0.

### Software Cost

1. Currency API [1]

a. Professional Plan

b. $39.99 per month

c. 100,000 Requests per month

d. 10-minute Updates

e. Historical Rates

f. Unlimited Support

g. HTTPS EncryptionC

h. Source Currency Switching

i. Currency Conversion

j. Time-Frame Queries

2. Sale Tax API [2]

a. Premium Plan

b. $29 per month

c. 100,000 requests per month

3. Tips Data Table

a. Free

4. Google Map API [3]

a. $0.50 per request

b. estimate $500 per month

5. Weather API [4]

a. Startup Plan

b. $40 per month

c. 5 days/3 hour forecast API

d. 16 days/daily forecast API

e. Weather maps 1.0

f. Weather alerts

6. Trip Planner

a. Free local data from user’s device

7. Local Services API

a. Use the same data from google Map API

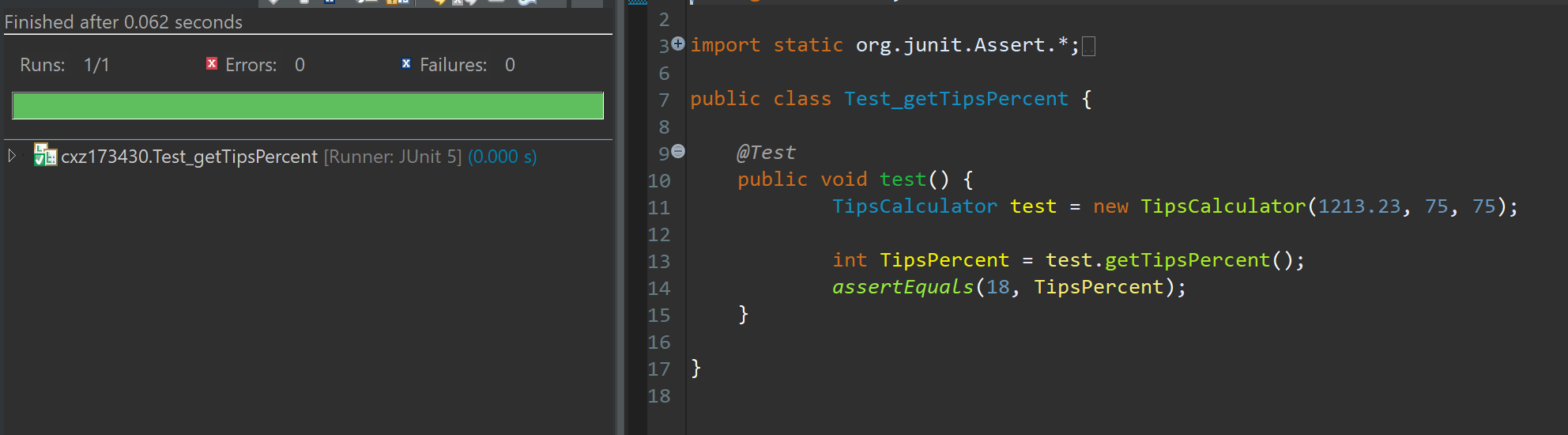
**Estimated Total: $610 per month to keep all services online.**

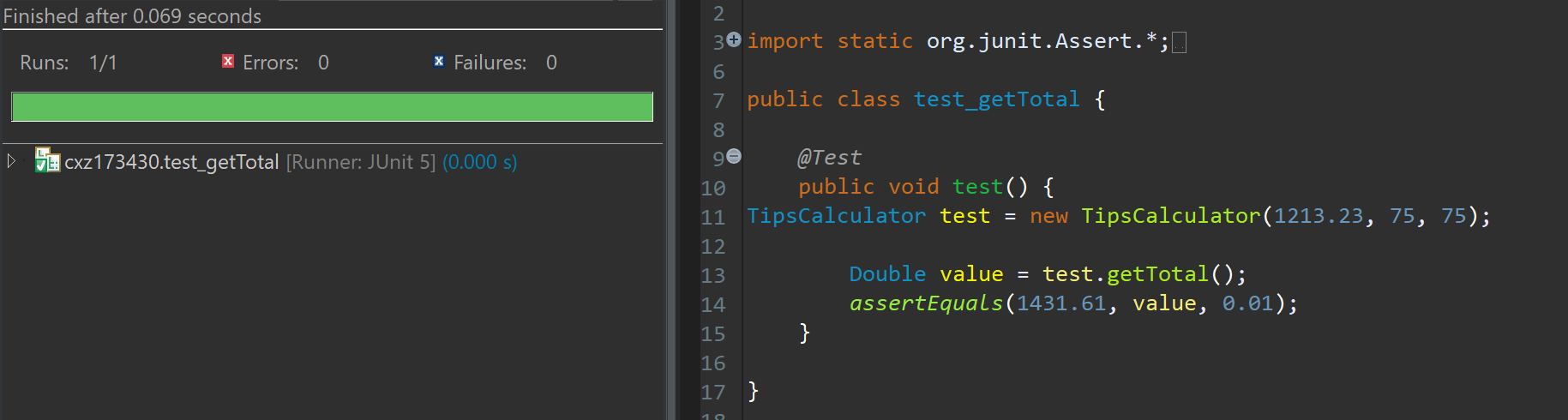
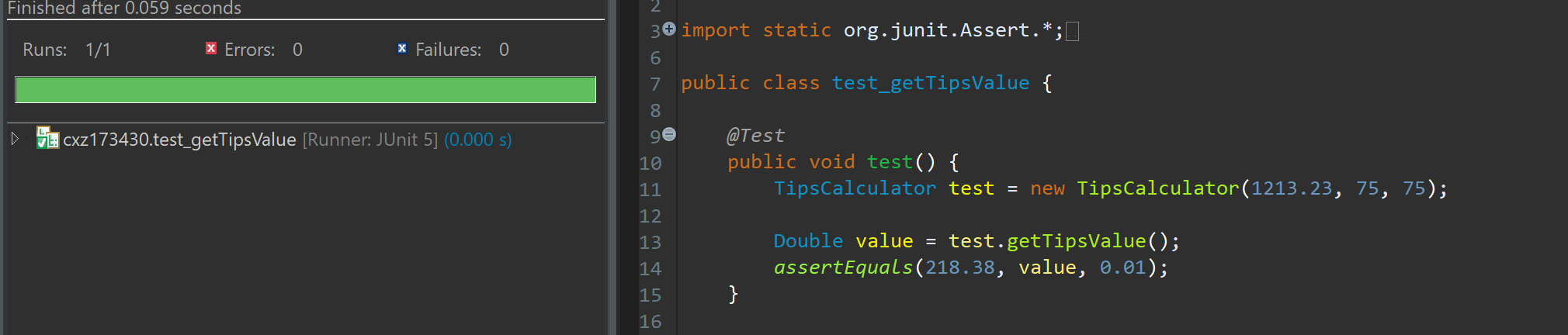
### Personnel Cost

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Resource Name | Cost | Baseline Cost | Variance | Actual Cost | Remaining |
| Project manager | $5,760.00 | $0.00 | $5,760.00 | $0.00 | $5,760.00 |
| UI designer 1 | $1,120.00 | $0.00 | $1,120.00 | $0.00 | $1,120.00 |
| UI designer 2 | $1,120.00 | $0.00 | $1,120.00 | $0.00 | $1,120.00 |
| Front-end engineer 1 | $2,520.00 | $0.00 | $2,520.00 | $0.00 | $2,520.00 |
| Front-end engineer 2 | $2,520.00 | $0.00 | $2,520.00 | $0.00 | $2,520.00 |
| Back-end engineer 1 | $2,880.00 | $0.00 | $2,880.00 | $0.00 | $2,880.00 |
| Back-end engineer 2 | $2,880.00 | $0.00 | $2,880.00 | $0.00 | $2,880.00 |
| Test engineer | $800.00 | $0.00 | $800.00 | $0.00 | $800.00 |

## **Test Plan**

Using Junit provided by Eclipse to ensure the TipsCalculator class works correctly. It mainly tested the getTipsPercent, getTipsValue, and getTotal methods. Please refer the file name Junit for more detail.





## **Comparison**

Our project consist of several functions that are already well established in the industry, the reason for this is because we are not trying to create a brand new product. The main focus in the development for this project, is to be able to provide the customer a “hub” to be able to use multiple well established and tested apps in one singular location, thus aiding the ease of use for many customers who either do not want to have multiple applications installed or are not technology fluent.

Some of the most prominent functions used in this project that are used by many other companies are Google Maps, and OpenWeatherMap.Google Maps API is used by companies that are based on findinding the shortest location from point A to point B. Some Companies that have already implemented the use of the Google Maps API include : Uber, and Lyft[5][6]. Due to the OpenWeatherMap API function being one of the top API’s used/downloaded for determining the weather in a location. As well for its pay model that allows for different tiers of payment, starting from free for 60 calls in a minute, all the way to 2,000 dollars for 200,000 calls per minute.[4]

For the tip calculator/currency converter/sales tax/scheduler/local services, it can be seen that there are many independent apps that focus on one of the functions that are implemented into the application, but there is currently no application that focuses on being a hub for the use functions that we are implementing. For the tip calculator/currency converter/sales tax/scheduler/local services functions, the respective applications that they can be compared to are: QuickTip(Tip calculator)[7], XE Currency (Currency converter)[8], Sales Tax Calculator (Sales tax)[9], Schedule Planner (Scheduler)[10], and anyService - local services (Local services)[11].

## **Conclusion**

#### Challenges

Brainstorming a software idea that we as a team believed was achievable and something that we could relate to. Developing a multi purpose software that would include essential functionalities while filtering out the functions that did not seem needed to minimize the software complexity. Also, maintaining the correct components in several of our diagrams ranging from our abstract use-case diagram to the more specific class diagram.

#### Achievements

Were able to efficiently communicate with each other that enabled task distribution and cross examination to check for flaws and cohesiveness during software development. Our group gained valuable experience in various stages in software development and tools that helps us visualize the functions of our software. We were able to make educated cost and developmental predictions that sets a timeline and resources needed to be allocated in able to develop TREK.

## 

## **References**

**General Internet Sites**

1. “Pricing Plans | currencylayer API,” *Real-time Currency Converter API, JSON format*. [Online]. Available: https://currencylayer.com/product. [Accessed: 20-Apr-2019].
2. Zip-Tax, “Tax.com Pricing,” *Zip*. [Online]. Available: http://zip-tax.com/pricing. [Accessed: 20-Apr-2019].
3. “Pricing & Plans | Google Maps Platform | Google Cloud,” *Google*. [Online]. Available: https://cloud.google.com/maps-platform/pricing/. [Accessed: 20-Apr-2019].
4. OpenWeatherMap.org, “Price,” *openweathermap*. [Online]. Available: https://openweathermap.org/price. [Accessed: 19-Apr-2019].
5. Uber, “Getting an Uber in Google Maps Just Got Easier | Uber Newsroom US,” Uber Newsroom, 12-Jan-2017. [Online]. Available: https://www.uber.com/newsroom/googlemaps/. [Accessed: 19-Apr-2019].
6. K. Hondorp, “Announcing Lyft Navigation, Built with Google Maps,” *Lyft Blog*, 12-Oct-2017. [Online]. Available: https://blog.lyft.com/posts/announcing-lyft-navigation. [Accessed: 19-Apr-2019].
7. D. Penny, “‎QuickTip™ Tip Calculator,” *App Store*, 24-Jul-2008. [Online]. Available: https://itunes.apple.com/us/app/quicktip-tip-calculator/id285924450?mt=8. [Accessed: 19-Apr-2019].
8. XE.com Inc, “‎XE Currency,” *App Store*, 20-May-2009. [Online]. Available: https://itunes.apple.com/us/app/xe-currency/id315241195?mt=8. [Accessed: 19-Apr-2019].
9. A. Samai, “Sales Tax Calculator - Apps on Google Play,” *Google*. [Online]. Available: https://play.google.com/store/apps/details?id=com.Calculator.SalesTaxCalculator&hl=en\_US. [Accessed: 19-Apr-2019].
10. D. I. G. I. LTD, “‎Schedule Planner,” *App Store*, 24-May-2011. [Online]. Available: https://itunes.apple.com/us/app/schedule-planner/id437461477?mt=8. [Accessed: 19-Apr-2019].
11. D. Haikin, “‎anyService - local services,” *App Store*, 06-Apr-2018. [Online]. Available: https://itunes.apple.com/us/app/id1225600745?mt=8. [Accessed: 19-Apr-2019].