**CS3354 Software Engineering**

**Final Project Deliverable I**

**Trek Travel App**

Alejandro Amaya Giron,

Churong Zhang,

Xizhen Yang,

Kaitian Li,

Jisoo Kim,

Kent G,

Frank Cazarez

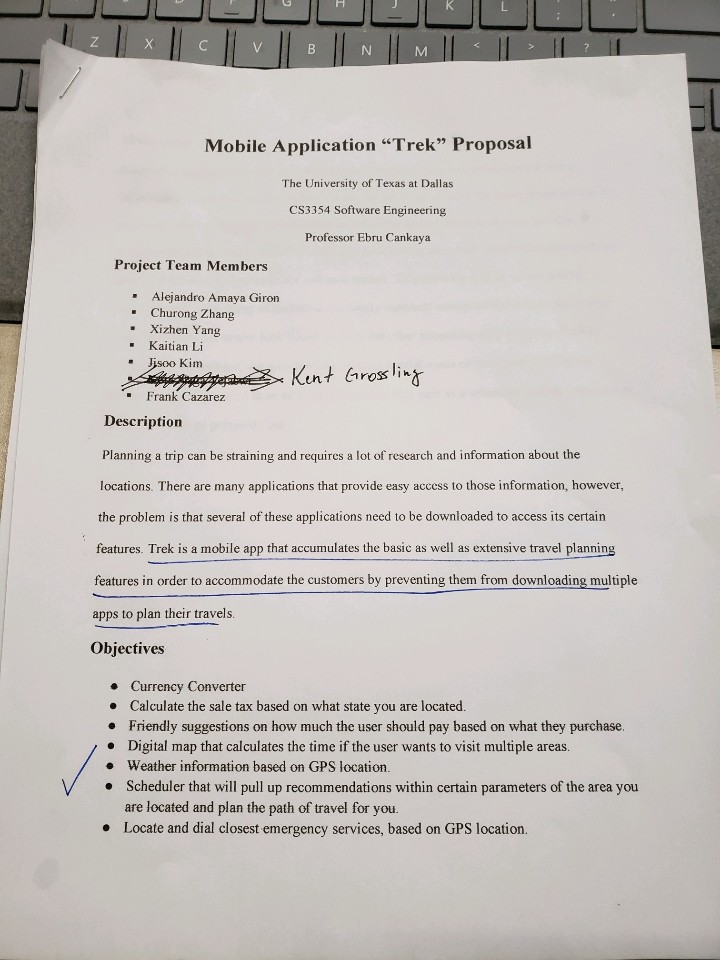
**Introduction**

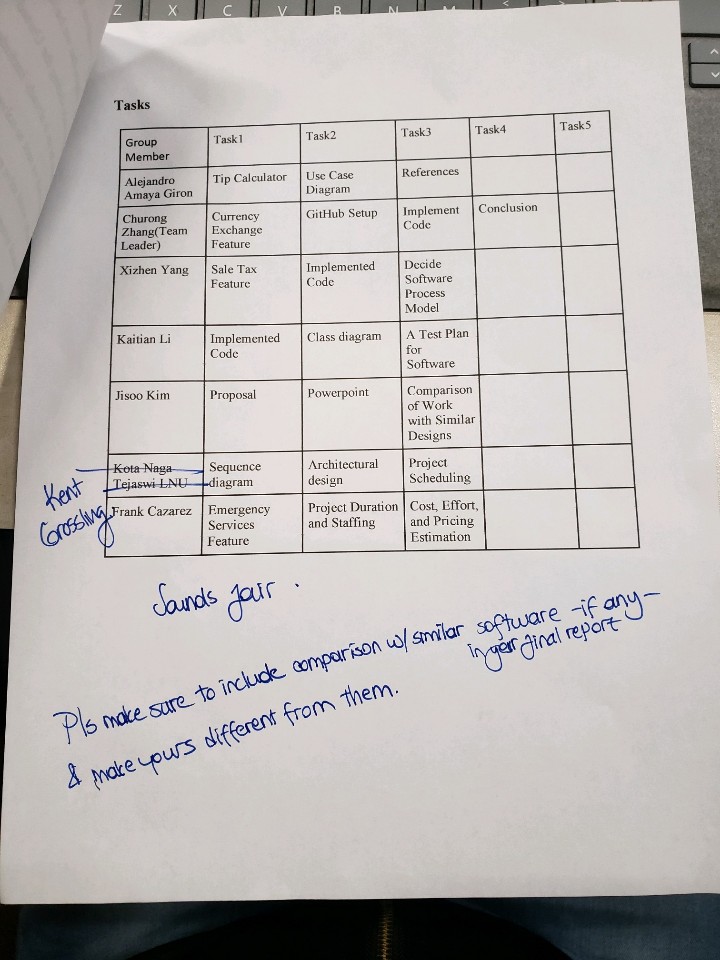
Planning a trip can be straining and requires a lot of research and information about the locations. There are many applications that provide easy access to that information. However, the problem is that several of these applications need to be downloaded to access its certain features. Trek is an app 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.

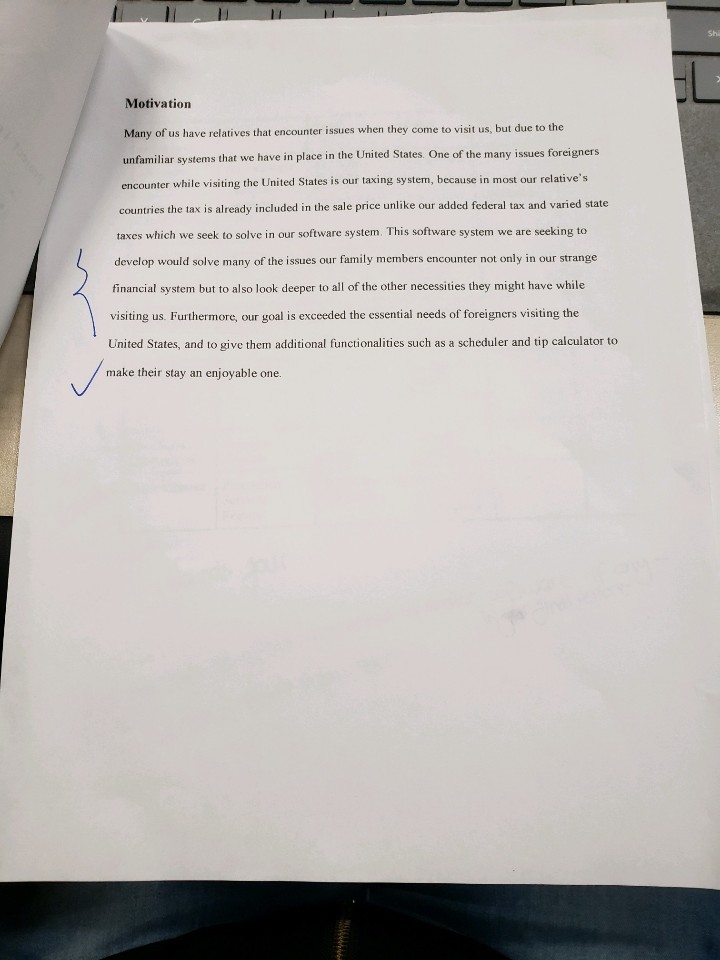
**Motivation**

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 suite 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**

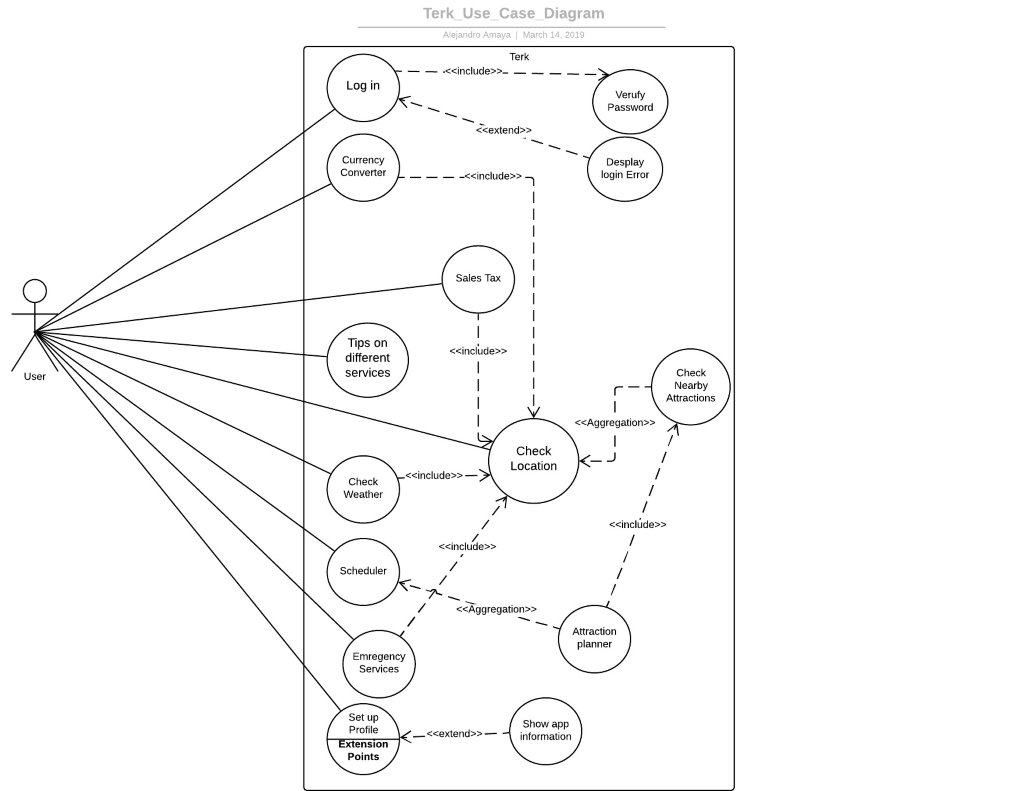
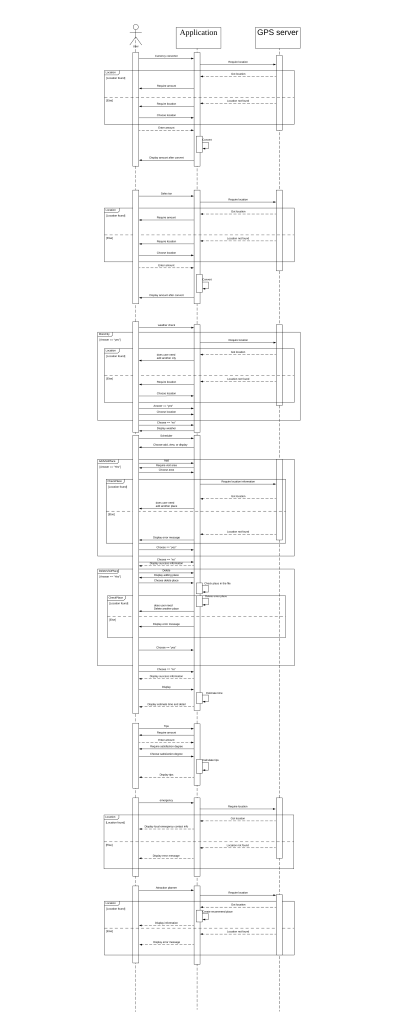


Fig 1. Use Case Diagram

**Sequence Diagram**

 Fig 2. Sequence Diagram

**Class diagram**

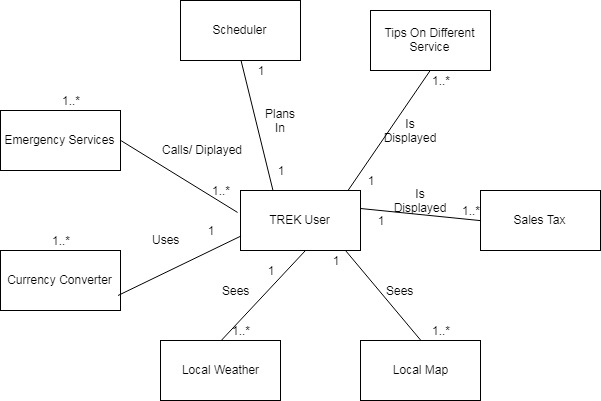


Fig 3. Class Diagram

|  |
| --- |
| **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.