Project Summary

The goal of this project is to provide a differential GPS correction service to the public. Although we will be focusing on the c++ post-processing library, the project includes many parts. These parts include a RoR web-app, an Android service and application, NTRIP data streaming client, and a vast repository of retrospective GPS data made available by NGS/CORS.

The concept of differential GPS corrections is fundamentally simple. Two collocated receivers suffer from similar (nearly identical) sources of error while utilizing civilian signals. Thus, two receivers using comparable multilateration algorithms will share common inaccuracies in their computed position. This concept allows geologically stable base stations to provide a vector correction for kinematic receivers in a ~100 km radius.

[For more details, see project part 1]

Broad Overview of steps involved:

1. Generate the CSV file of the time position and satellites in view from android app (Have app working)
2. Parse CSV into data table
3. convert lat long alt to XYZ
4. check for GLONASS satellites being used (satID > 36)
5. Find nearest base station based on XYZ pos (find GLONASS station if 4 is yes)
6. Download Rinex file from base station
    a. if fail return to 5
7. Trilaterate base station coordinates using sat list
    a. if fail return to 5
8. calculate vector correction from calculated pos and true pos
9. apply vector correction to the original data
10. return data to user

Project Requirements

1. GPS Logger must run on Android
2. Android app must generate CSV file
3. System must parse CSV that user uploads
4. Must check for GLONASS satellites in view
5. Must create database of all base stations and their locations from ASCII file
6. Must be able to convert geospatial coordinates between earth centered cartesian and lat long alt formats
7. Must calculate nearest base station based off of current XYZ position
8. Must convert ms since 1/1/1970 to GPS time
9. Must download RINEX navigation and observation files for the base station/GPS time
10. Must calculate position based on list of satellites

11. Must calculate vector offset based on known position and calculated position
12. Must apply vector offset to the users position
13. Must rewrite the CSV position file
14. Must supply new CSV to client

Stretch goals:

1. make android app pretty
2. make android app submit data directly to system.
3. Rails app
4. Implement NMEA input format
5. Implement RINEX input format
6. Implement RINEX
7. tie in Google maps API

Classes we will make:

- GpsPostProcessor
- UnitConverter (facade of GPS toolkit)
- FilesSystemMgr
- BaseStationMgr
- BaseStation
- DownloadMgr
- RinexMgr(facade of GPS toolkit)
- ClientGpsData
- CsvHandler

Use Case ID:	UC-01.01
Use Case Name:	Submit for corrections
Description:	Users submits GPS data for corrections

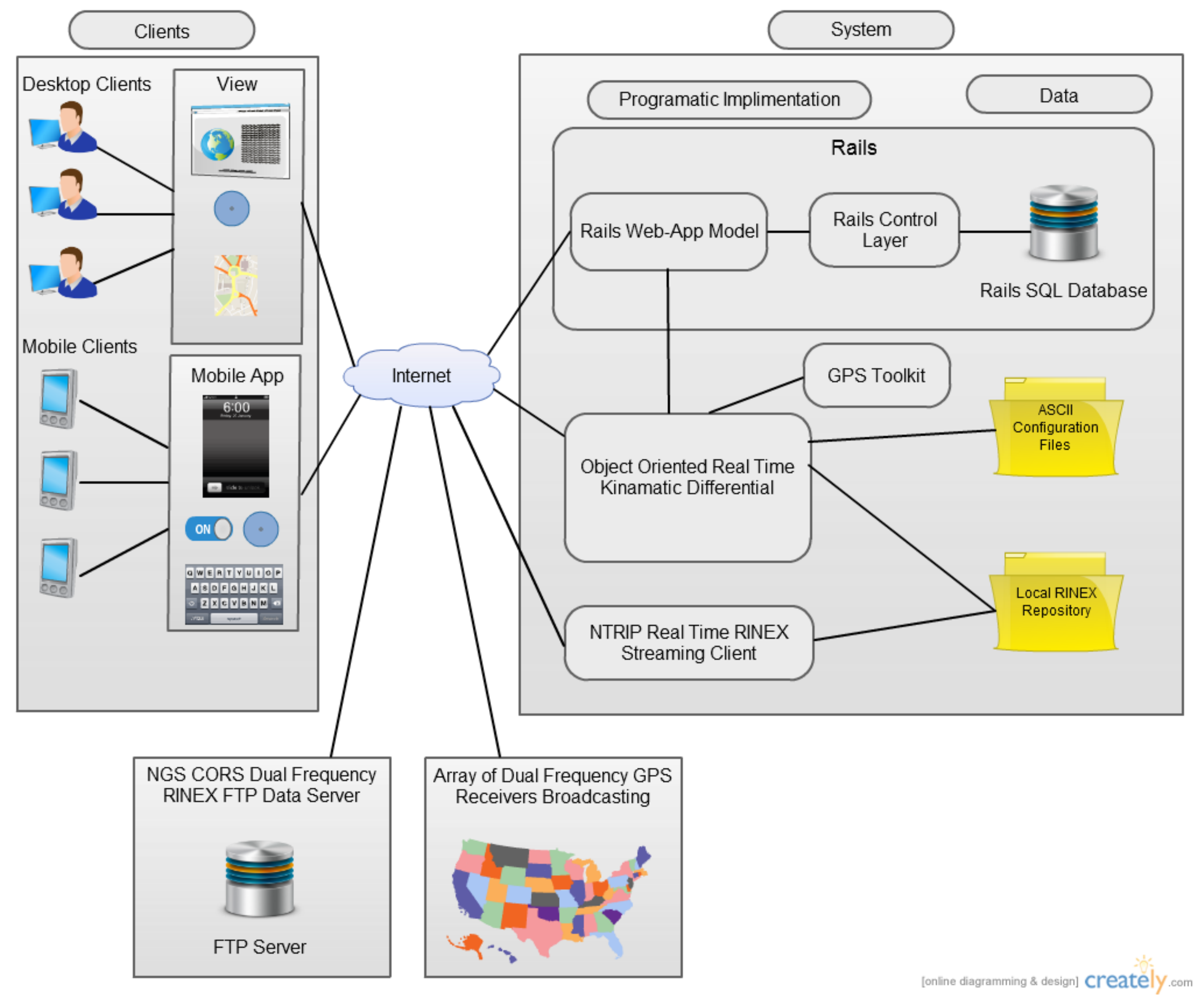
Actors:	Users, System
Pre-Conditions:	User has set of GPS data to submit
Post-Conditions:	**On Success**: The user receives a new set of data points with the correction applied **On Failure**: The user receives the same data points that were submitted with a notice that the operation failed.
Frequency of Use:	Potentially thousands of times a day
Flow of Events:	see below
Variations:	If there is an error and corrections are unable to be made then the same data is returned with a notice that the corrections have failed
Notes and Issues	This is the main use case because most of the work we are doing is back end work for the system.

	Actor Action	System Response
1.	Open GPS corrector app	
2.	Actor selects submit data and submits it.	Data is gathered. corrections are applied
3.		corrections returned to the user

Activity Diagram

See *activity - user submits csv of GPS locations.png*

Architecture Diagram


Clients
Desktop Clients
View
Mobile Clients
Mobile App
Internet
System
Programatic Implimentation
Data
Rails
Rails Web-App Model
Rails Control Layer
Rails SQL Database
GPS Toolkit
ASCII Configuration Files
Object Oriented Real Time Kinamatic Differential
Local RINEX Repository
NTRIP Real Time RINEX Streaming Client
NGS CORS Dual Frequency RINEX FTP Data Server
FTP Server
Array of Dual Frequency GPS Receivers Broadcasting
[online diagramming & design] creately.com

Data Storage

- CSV Configuration files
    - base station IDs and coordinates
- CSV coordinate data files (submitted by and returned to the user)
    - (UT, lat, long, alt, Satlist[])
- SQL database
    - sites, queried to find nearest site
- RINEX Observation files
    - supplied via FTP by the National Geodetic Survey
- RINEX Navigation files
    - supplied via FTP by the National Geodetic Survey

UI Mockups

- *ui mockup - android screenshot.jpg*
- *ui mockup - main menu.jpg*
- *ui mockup - real time submit (stretch goal).jpg*

User Interactions (sequence diagrams with each interaction)

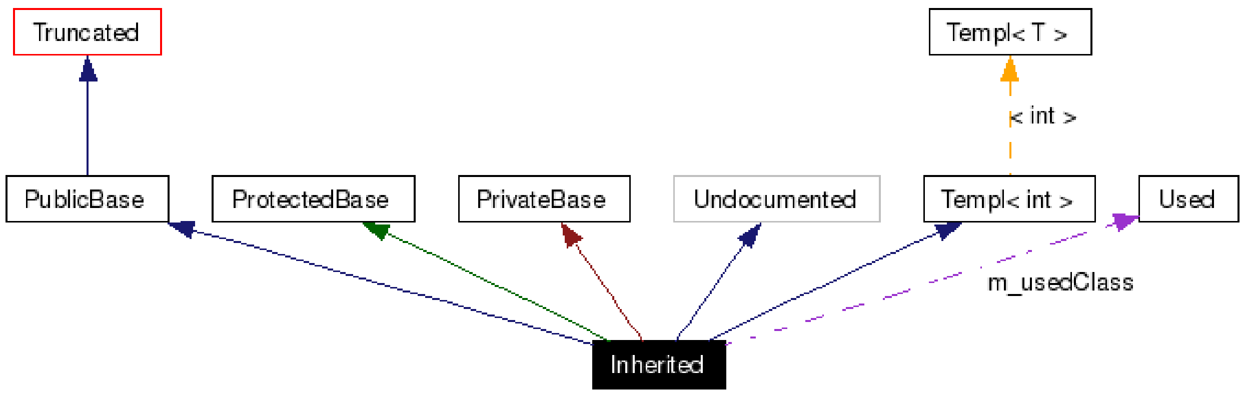
- *sequence - user submission.jpg*
- *sequence - start up.jpg*

Class Diagram

- *uml - class diagram.png*
    - *https://www.draw.io/#G0B1qTpXqWjW9sUVZBUms0VTJMQ1k*

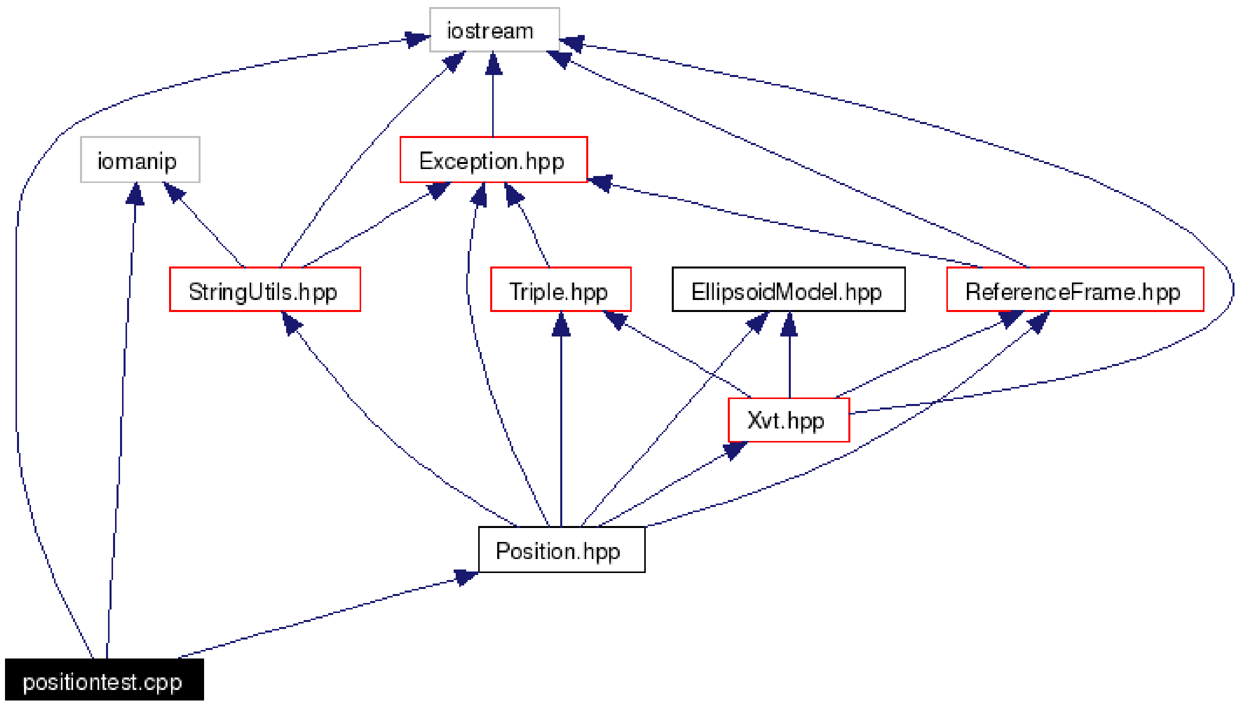
GPS Toolkit Context

Legend for reading the following diagrams:



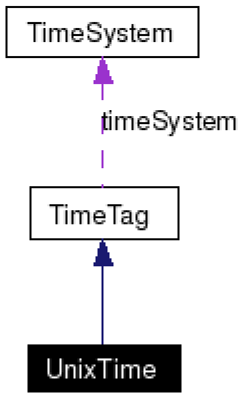
http://www.gpstk.org/doxygen/graph_legend.html

Context for the position conversion class (Position):



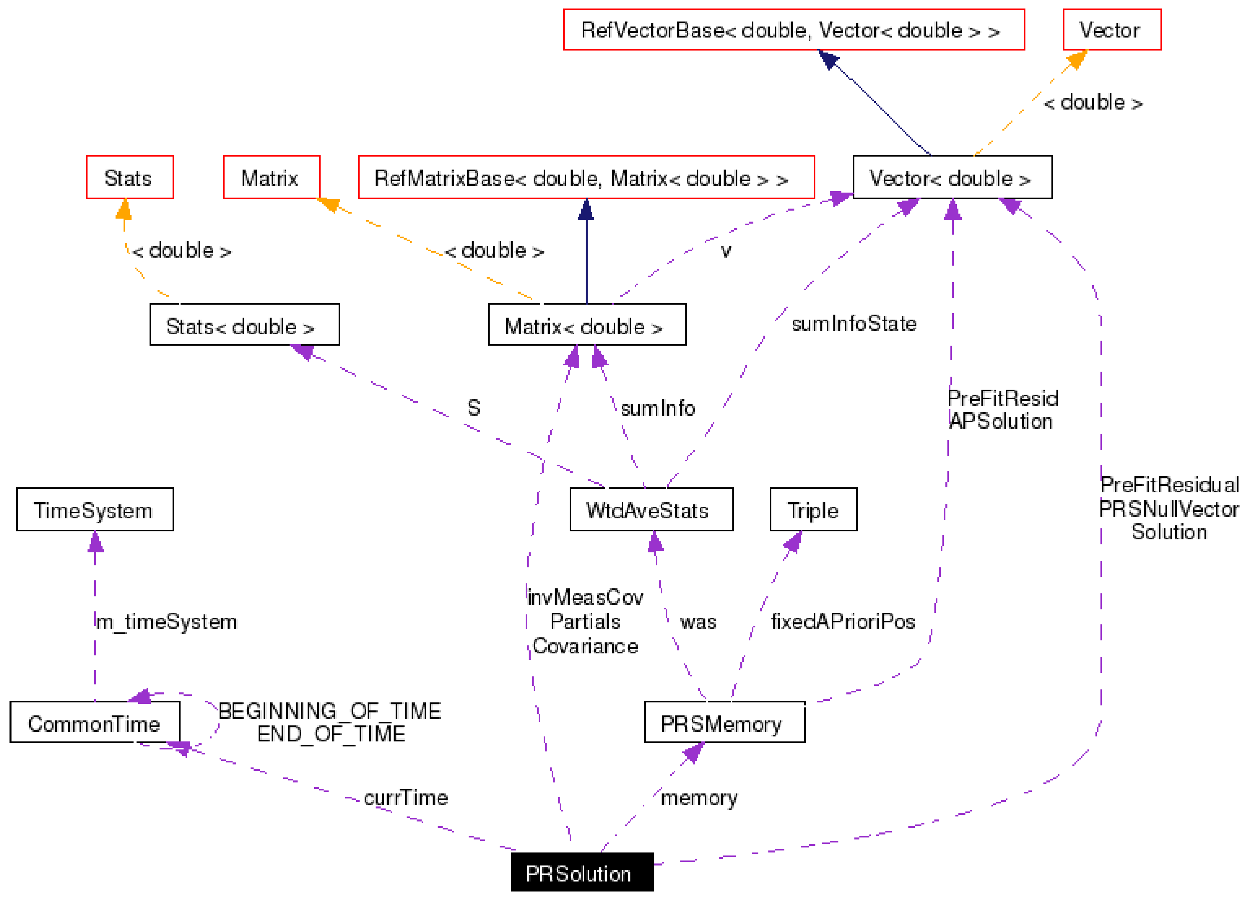
http://www.gpstk.org/doxygen/positiontest_8cpp.html

Time conversion class and dependencies (UnixTime):



http://www.gpstk.org/doxygen/classgpstk_1_1UnixTime.html

Context for RINEX position solver (PRSolution)



http://www.gpstk.org/doxygen/classgpstk_1_1PRSolution.html

For more information about GPSTk, please see the following links.

Home page: http://www.gpstk.org/bin/view/Documentation/WebHome
Doxygen documentation: http://www.gpstk.org/doxygen/index.html