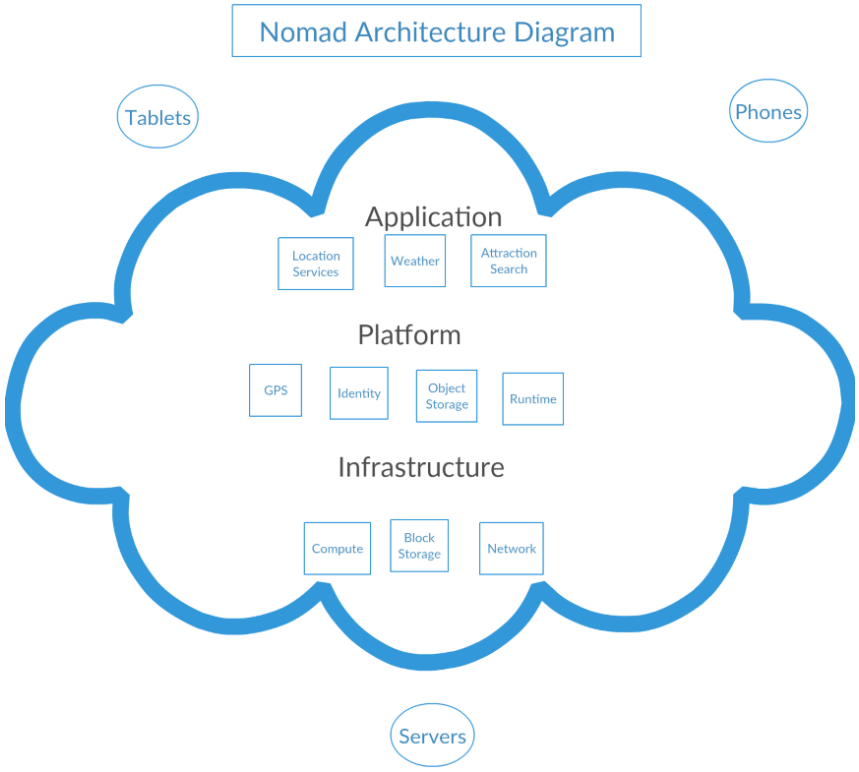


We chose to base our architecture off the Cloud Computing Architecture. Some of the advantages to this architecture in which our application can benefit from are its reliability, manageability, scalability, storage for data, device independence and location independence.

Our architecture is shown above and includes the following components:
Servers: hold data needed for application, servers will include our own application server used for user data and will bring data from the other servers into it for access in the application. Other servers needed will include google geo location, google places, and weather. These servers will be used for accessing google maps/locations, attractions near the destination, and weather conditions at the destination, respectively.
Tablets and Phones: represent the devices the application will be used on
Location Services: will use Google Geo Location API, will hold the libraries that are able to access the GPS to determine the devices current location
Attraction Search: will use Google Places API, will have information regarding what attractions are near the destination location
Weather: will use Weather API, will have information regarding weather conditions at the destination location
GPS: hardware of the operating system for GPS application, location services will access
Identity: used for user authentication and identity to track user information and storage
Object Storage: server side storage, stores user data, tokens, history of user searching, will hold all user data in the database

Runtime: runtime will include application launches, service calls, such as authentication, location, weather, attractions to find results, accessing storage, display results to user
Compute: not all hardware and software application will use will be on device or in device's network, will be provided as a service by application and accessed over the internet in a seamless way
Block Storage: local caching, keeps track of session tokens, will cache server data in case user closes application, the application will not have to recall server when user opens application again
Network: calls to servers on other networks, used to transfer data to and from application

Using APIs and the cloud:

-google geo location: would interact with cloud to provide location of the device for the application
-google places: would interact with cloud to provide places nearby the location of the device for the application
-weather: would interact with cloud to provide weather of current location of the device for the application

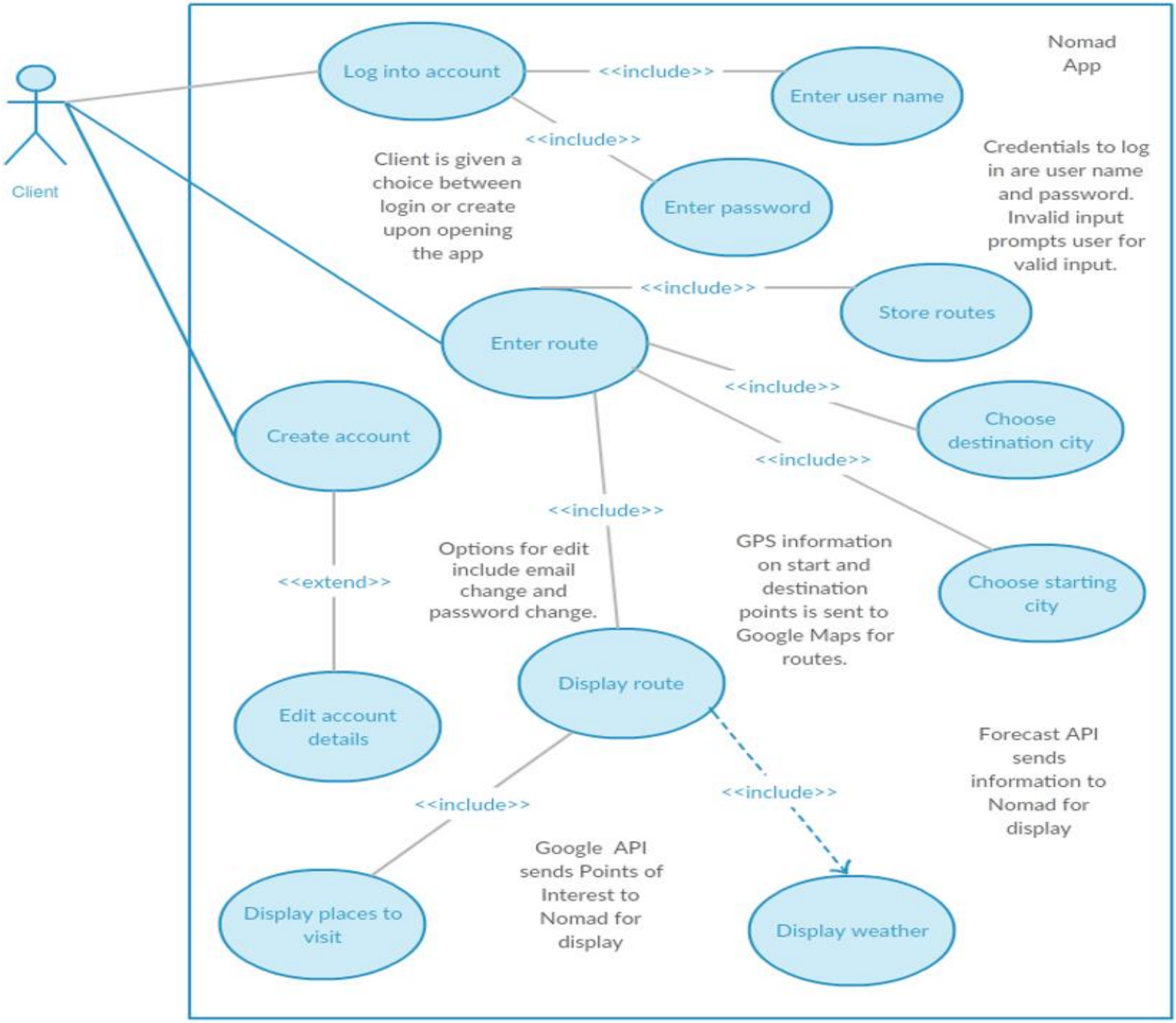
Architecture Style:

We have chosen the Pipe and Filter style for our 'Nomad' use case. We felt like this style would work well for 'Nomad' due to the style's incremental design. For our application, independent components will process data in this fashion, so Pipe and Filter is a compatible style choice. Some of the advantages of Pipe and Filter style that also make it appealing are its simplicity, concurrent execution abilities and its ease of maintenance, enhancement and evolution. Pipe ad Filter allows designers to understand the input and output behavior of the system by its filters. Also each filter can be implemented as a separate task and be executed in parallel with other filters if that is desired.

Intended operations of application:

-user will create an account and then can log in to account
    -system will have authentication capabilities
-user will enter source and destination
    -system will calculate shortest path to destination, using gps
-user will proceed to route
    -system will ensure user stays on route, using location services
    -system will display weather during journey, using weather api
-user will reach destination
    -system will store route in user history
    -system will display places to visit, using google places api
-user will be able to access its route history and reselect old routes

Project 457
Group 3
Nomad Use Case



Project 457
Group 3
Nomad Use Case

Activity diagram represents the standard events of the system:

-User will open the application, in which the system will require user to log in

-User will login, system will check for validity

-User will request route, system will determine route and return result to user

-Using gps and location services

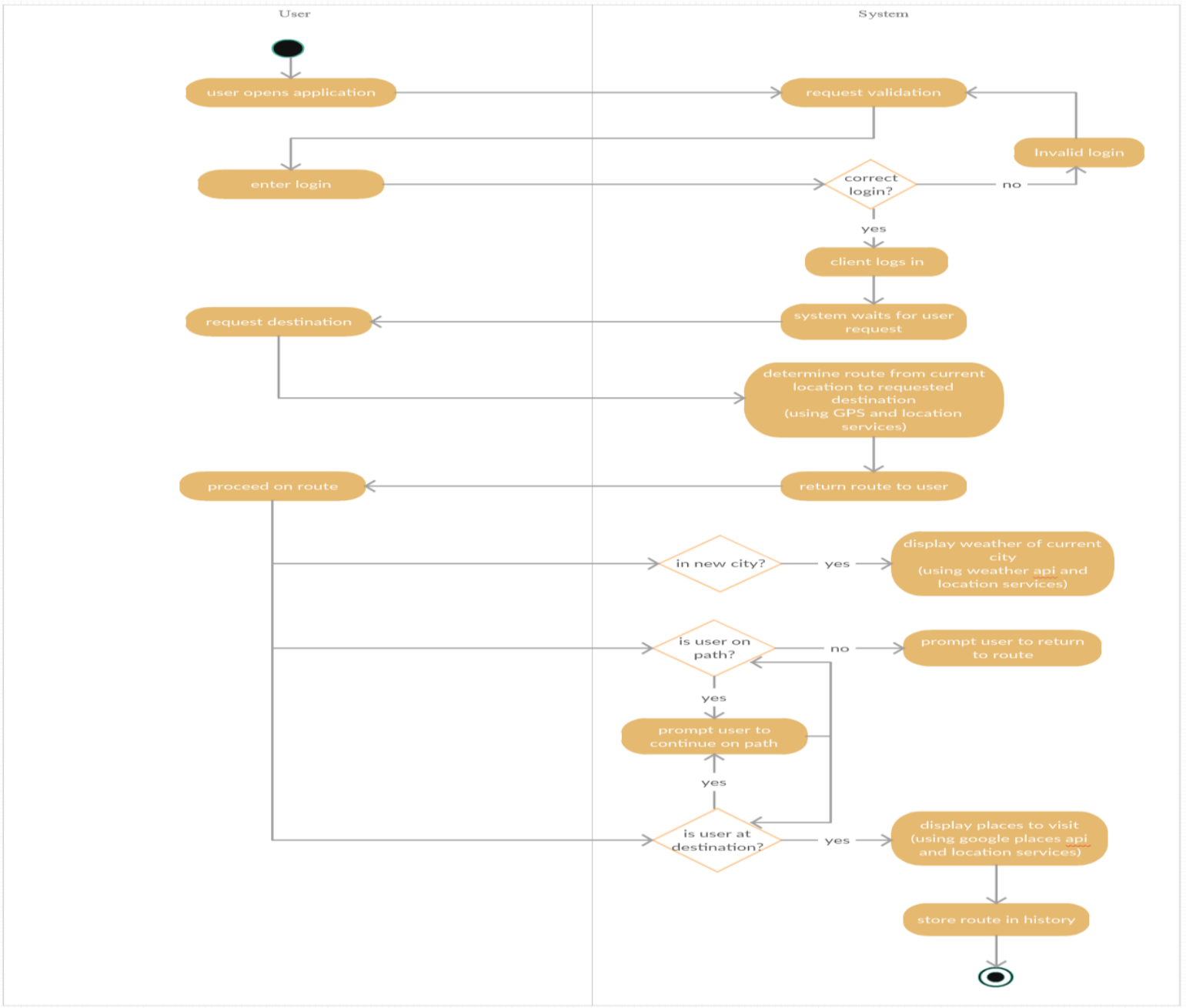
-User will proceed to route, system will display weather along route and make sure user stays on path

-Weather will be updated each time user enters new city, using location services

-Using location services to make sure user is on route

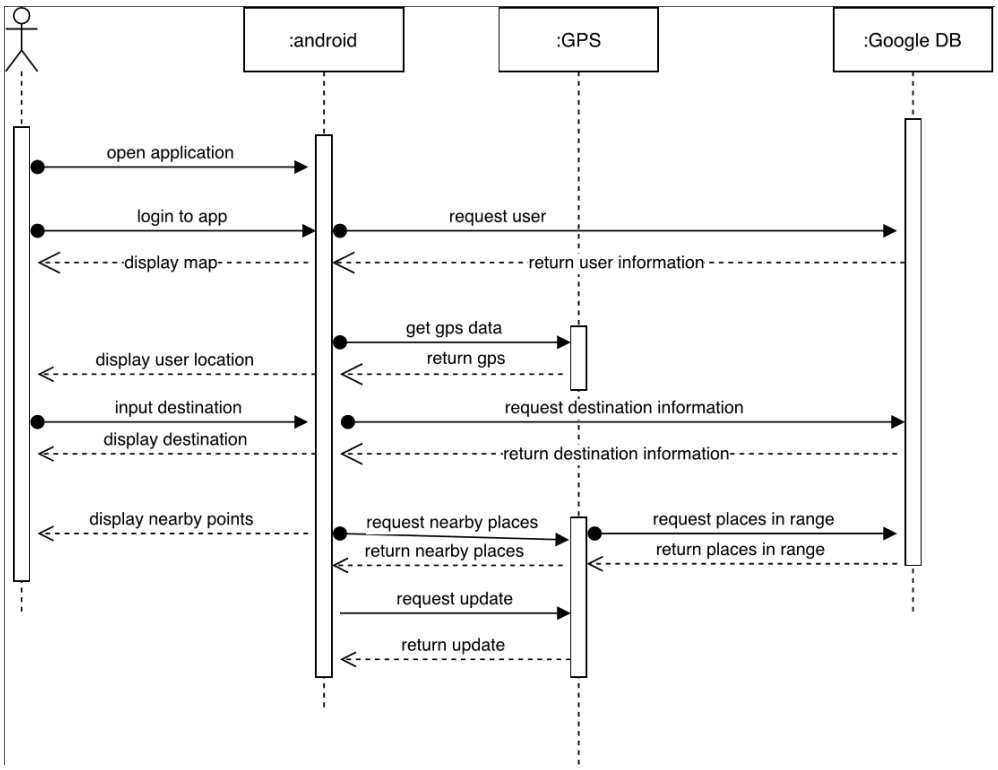
-User will reach destination, system will store route in history and display places to visit at destination

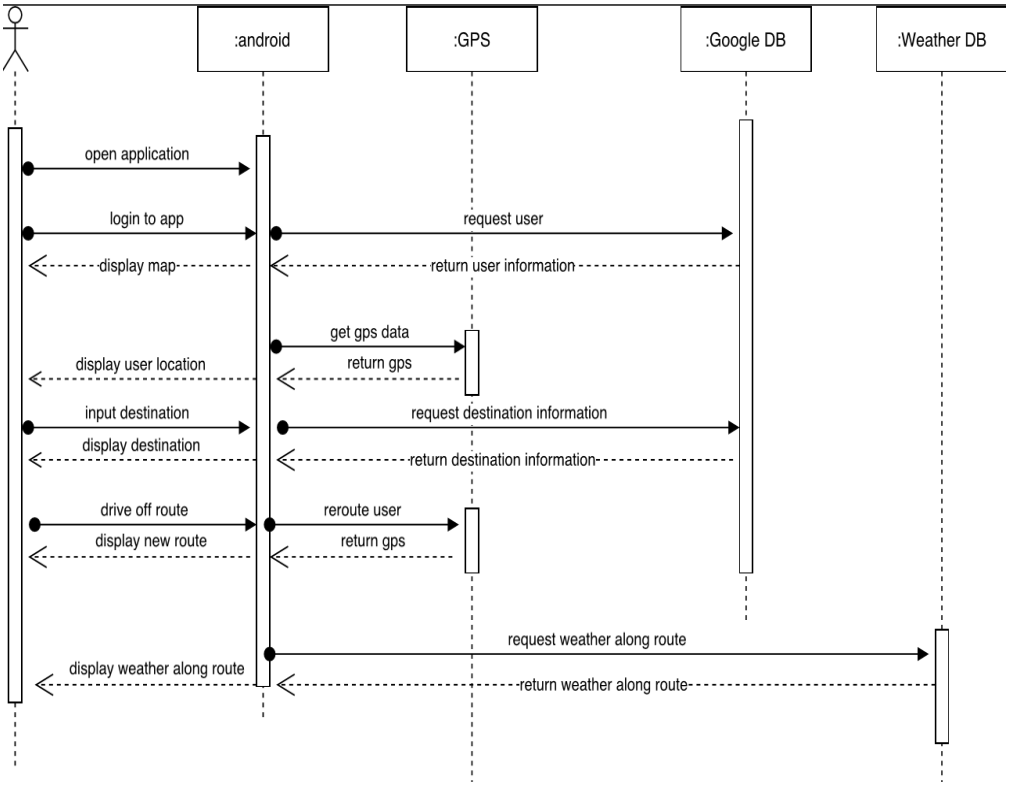
-Using google places and location services



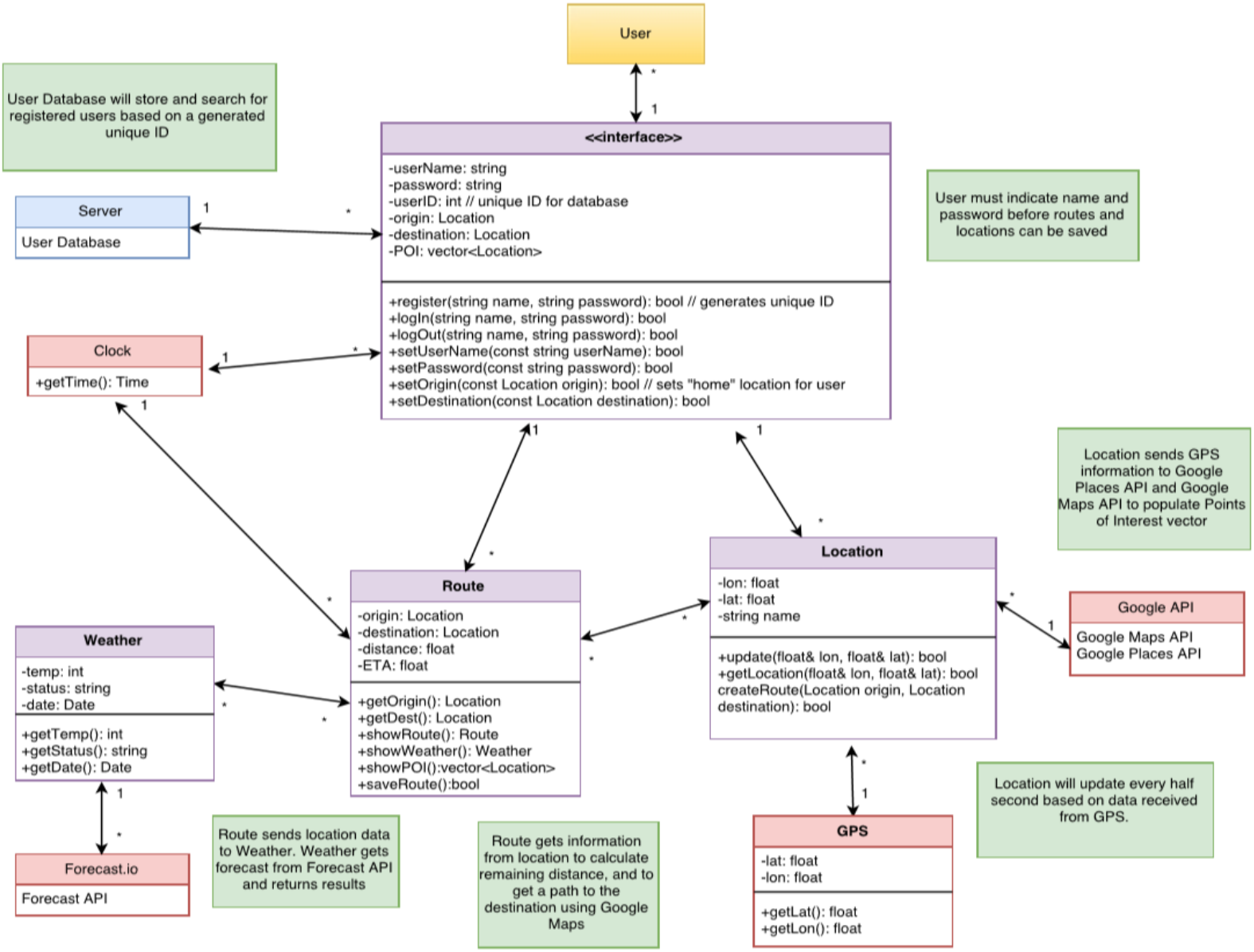
Use cases for sequence diagrams:

1. Route to destination
2. Display weather along route
3. Display places to visit upon arrival of destination
4. Prompt user to return to route if off route





Project 457
Group 3
Nomad Use Case



Project 457
Group 3
Nomad Use Case

Design Pattern:

Observer Pattern

-Chose this pattern because system and user will need to notified when certain event occur during application execution

-Events such as:

-user veers off route and needs to return to correct path

-user enters new city and current weather needs to be updated

-user reaches destination and places to visit need to be displayed

-When looking at the class diagram, the Route class will be the ‘observer’ when implementing this pattern