UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

2017 Class Project NavUP

Team Broadsword Team Navigation Module

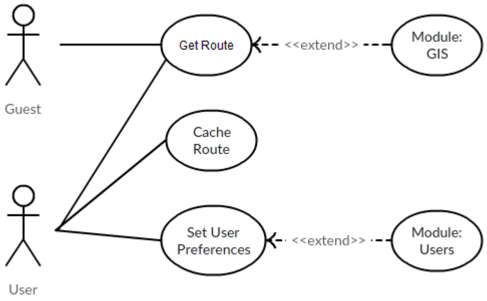
Bondjobo, Jocelyn 13232852
du Plooy, Andries 15226183
Brijlal, Yashvir 14387744
Jones, Keanan 13036892
Nxumalo, Banele 12201911
van Schalkwyk, John 14307317

Contents

1 Navigation Module

1.1 Scope

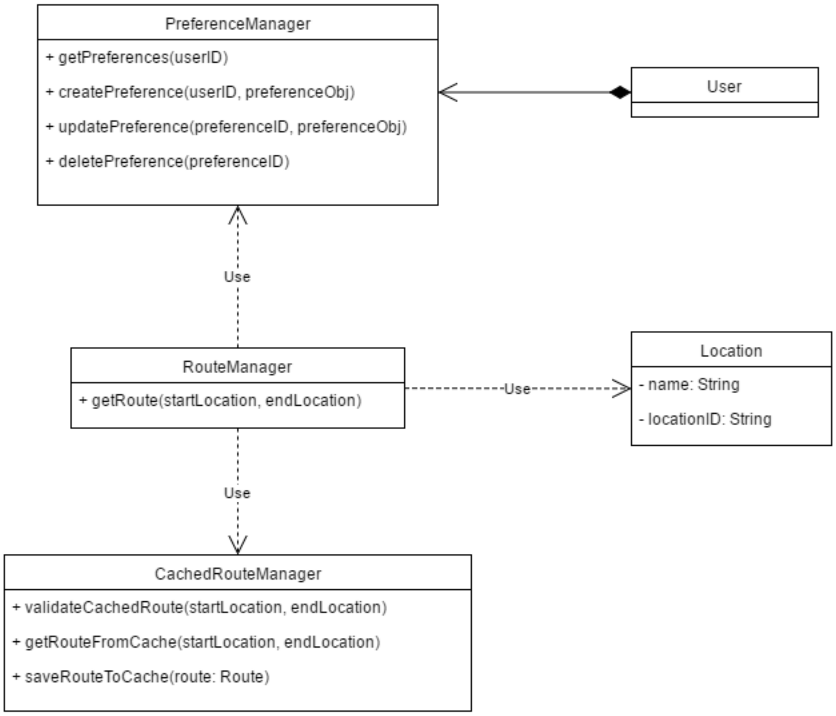
Navigation employs a caching mechanism whereby commonly used routes are stored within a database in order to mitigate the cost of calculating popular routes multiple times. Additionally, it acts as the Controller in the Model-View-Controller architecture with Access acting as the View and GIS being the model. The precise nature of the scope of the Navigation module is shown in the diagram below.



- Get Route: Routes need to be calculated, thus the GIS module is queried in order to get the route.
- Cache Route: Routes are persisted to avoid recalculation, thus popular routes with similar user preferences will be readily available.
- Set User Preferences: The user's prefernces are stored locally, in order for the correct path for each user to be calculated.

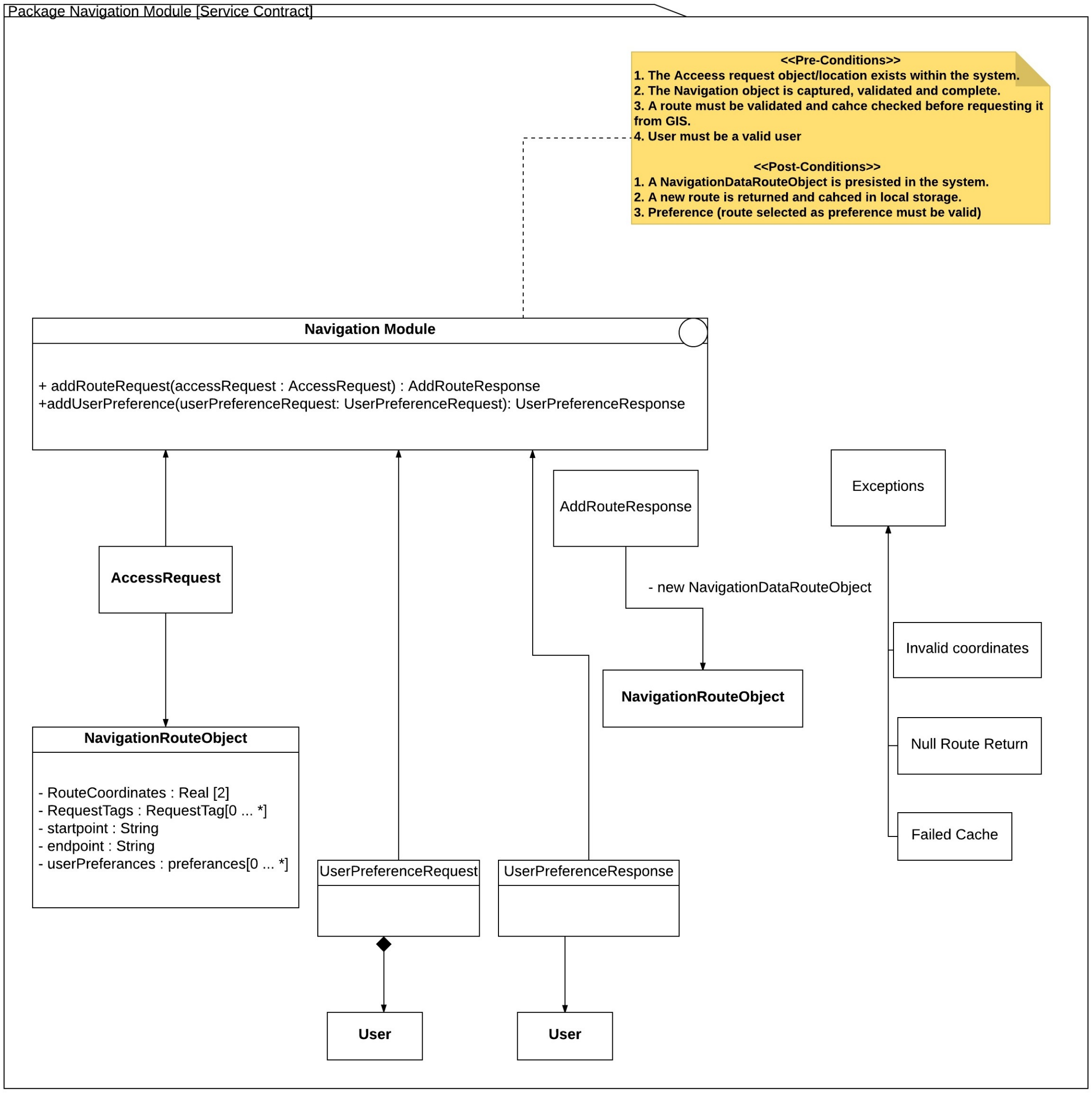
1.2 Domain Model

The domain model encapsulates a simplified model of the core functionality provided within the navigation module. The route manager class handles route requests by receiving a start and end point and retrieving the route either from cache or, if not present in cache, retrieving it from the GIS module. The cached routes manager handles creating, saving and updating of cached routes through its various functions. User preferences are required to retrieve filtered routes. The preference manager encapsulates creating


PreferenceManager
+ getPreferences(userID)
+ createPreference(userID, preferenceObj)
+ updatePreference(preferenceID, preferenceObj)
+ deletePreference(preferenceID)
User
Use
RouteManager
+ getRoute(startLocation, endLocation)
Use
Location
- name: String
- locationID: String
Use
CachedRouteManager
+ validateCachedRoute(startLocation, endLocation)
+ getRouteFromCache(startLocation, endLocation)
+ saveRouteToCache(route: Route)

retrieving updating and deleting of users preferences to be used by the route manager in retrieving filtered routes based on a users preferences.

1.3 Service Contracts



Package Navigation Module [Service Contract]
<<Pre-Conditions>>
1. The Acceess request object/location exists within the system.
2. The Navigation object is captured, validated and complete.
3. A route must be validated and cahce checked before requesting it from GIS.
4. User must be a valid user
<<Post-Conditions>>
1. A NavigationDataRouteObject is presisted in the system.
2. A new route is returned and cahced in local storage.
3. Preference (route selected as preference must be valid)
Navigation Module
+ addRouteRequest(accessRequest : AccessRequest) : AddRouteResponse
+addUserPreference(userPreferenceRequest: UserPreferenceRequest): UserPreferenceResponse
AccessRequest
NavigationRouteObject
- RouteCoordinates : Real [2]
- RequestTags : RequestTag[0 ... *]
- startpoint : String
- endpoint : String
- userPreferances : preferances[0 ... *]
UserPreferenceRequest
UserPreferenceResponse
User
User
AddRouteResponse
- new NavigationDataRouteObject
NavigationRouteObject
Exceptions
Invalid coordinates
Null Route Return
Failed Cache

The reasons for which the service contract will fail can be due to a few actions occuring.

- Firstly, When incorrect start and end locations are sent to the Navigation system, the system will not recognise it and will not return a route. Thus the correct desired start and endpoints must be received.
- A second occurance can be when the GIS module does not find a route, resulting in a null being returned. The null retuns must be handles as they will invalidate cached routes.

The only time GIS will be queries is the first time every route is requested. After that first request local cache will first be queried, thus reducing networking as well as request speed.

1.4 Technology

1.4.1 MEAN STACK

MEAN stack is the best solution for building application using javascript and since we are implementing the MVC architectural pattern for the navUP system , MEAN stack is the best way to deploy the current architectural pattern . It includes mongo , express , angular and nodejs which allows us to built fast and maintable applications.

1.4.2 MongoDB

MongoDB is a document-oriented database that provides high perfomace,availability and scalability. It works on the concepts of collection and documents. For the NavUP application , we used mongoDB to store data such as start-point,endpoint or user preferences.

1.4.3 Express

Express is fast and flexible framework for nodejs .

1.4.4 Nodejs

Nodejs is JavaScript run-time environment for executing JavaScript code server-side. It is used for easily building fast and scalable applications. All the APIs of nodejs library are asynchronous means a Node.js based server never waits for an API to return data.

1.4.5 Mongoose

This will be our Object Relational Mapping tool which will allow us to intuitively persist objects straight into the database. Mongoose allows us to have access to the MongoDB commands for CRUD easily

1.4.6 NSQ

NSQ is a library used for communications between different modules like access ,gis and navigation modules.