Nomad:

Navigation for Humans

Bradley Degenaars

Winter 2013, Capstone Proposal, Neumont University

Executive Summary

Nomad and Nomad’s unique navigation utility provide real-time traffic solutions for today’s traveler. Nomad provides this by offering live traffic analytics, coupled with other, autonomous functionality that works seamlessly with the user. Through these methods of processing information, and the use of other third-party services and technologies, Nomad provides autotomized navigation planning and execution, along with messages and alerts that provide information crucial to travel. It is through Nomad’s intuitive nature and design that it provides constant value and benefits to the everyday user.

Project Description

Nomad and Nomad’s improved navigation utility are a new navigation application that takes a more practical and personal approach to getting the user where he/she needs to go. Analytical functions performed on relevant data, such as traffic patterns, accident reports, and cached data of the user’s previously navigated routes, allows Nomad’s navigation utility to guide the user on more personalized and less troublesome routes every day.

The goal of Nomad is to lessen the burden and inconvenience of everyday travel, in an effort to make traveling more of an ease and safer for everyone. Products such as Waze, Google’s Car Home, and CarHome Ultra are good examples of applications that accomplish pieces of Nomad’s goal, but not the entirety of it. Waze uses community input in order to serve its user base with the cheapest gas prices in the area and avoidance of traffic delays like rush hour delays and accidents. Google’s Car Home application and TheSpinningHead’s CarHome Ultra, provide their user’s with real time data and better accessibility to applications and utilities while driving using large tactile buttons. Although these applications fulfill parts of today’s driver needs, Nomad satisfies them all. The primary difference between Nomad and its competitors is that Nomad provides the features of its competition in one centralized place. Another way Nomad differs from its competition is that rather than users having to provide input to the application, Nomad automatically collects data from the user behind the scenes. The finished deliverable Nomad application will give the user a few HUD (Heads Up Display) items, as well as six configurable shortcut keys to other applications and utilities. The HUD items include current speed, current direction, current time, current temperature, current location, and total time elapsed in route.

The six configurable shortcuts are large touch areas that house a small number of applications and utilities available to the driver, including Nomad’s improved navigation utility. These applications will be limited for the drivers, and everyone else’s safety. For example, such applications as text messaging, Netflix, and Facebook will not be available as options. Nomad offers live alerts on traffic conditions, weather warnings, and accident reports, as well as solutions to these situations that would normally delay or increase travel times.

**The User**

Nomad’s target user is a quite broad demographic. To compensate for this fact, five distinct driver profiles define how one or many users could potentially utilize Nomad. These profiles help show how the different functions of Nomad benefit not necessarily multiple individuals, but rather how one user can benefit from Nomad using its tools in separate contexts. A user that would greatly benefit from using Nomad would fall under one or more of the following categories:

* The *Daily commuter*
  + The daily commuter is a user who usually travels the same route to and from work, or travels the same routes habitually. This user spends a considerable amount of time in traffic and other undesirable travel conditions. We also classify students as daily commuters because they follow almost identical patterns.
* The *Errand runner*
  + The errand runner is a user who travels to many different locations and takes many different routes throughout his/her day. This user is traveling during mid-day hours, possibly with companions, and knows where he/she is going because the places he/she travels to are places that he/she frequents.
* The *Driving enthusiast*
  + The driving enthusiast is a user who enjoys driving, and does not enjoy driving to be interrupted or disrupted by things such as accidents, traffic, or densely populated or frequented areas. This user may or may not have a destination or purpose for driving, other than to be out driving enjoying the road.
* The *Out-of-towner*
  + The out-of-towner is a user that may or may not fall into one of the aforementioned categories of driver types, but this is irrelevant because this user often has little to no knowledge of his/her current environment, or a well-tuned sense of direction.
* The *Local*
  + The local is a user who lives in the area he/she travels to most often. The travel patterns of this user are erratic and random this user will most likely have a well-founded knowledge of the surrounding areas, and the routes that make up said area.

Defining Nomad’s users into these categories allows for a more precise prediction of the “how” and “why” in Nomad’s use.

**Value**

Nomad’s value stems from several sources within the application. The primary feature is Nomad’s intuitive ability to route the user from point A to point B in the best, most efficient driving environment possible.

One of the greatest challenges of traveling in today’s world is planning. Planning your daily commute, your daily errands, or even a seasonal trip is far more difficult with today’s high volume of traffic, unpredictable weather conditions, and road hazards. By utilizing data from internal and external sources, Nomad can give the user a “heads-up” to bad traffic conditions, or alert the user to an alternative departure time or route that will create a better driving experience. A safer, less hectic and more aware experience for drivers, is beneficial for us all.

Nomad’s services are significant to its users because at Nomad’s core, tailoring the application to the user’s preferences, both on a conscience and subconscious level. Nomad’s ability to learn from the users travel patterns and habits allows Nomad’s services to produce routes, and beneficial alerts and messages, that the user can appreciate and use constructively. The following is a breakdown of the individual benefits that users will reap from the use of Nomad and its uniquely designed utilities.

* The *Daily Commuter*
  + This user would heavily rely on Nomad’s unique navigation routes, and its sophisticated messaging and alert system. For this user, knowing that he/she could leave for his/her destination fifteen or twenty minutes later than usual and still make it there the same time due to traffic would be wonderful knowledge to have.   
      
    An accident on the highway, and any road for that matter, is a dangerous event in today’s world and it happens all the time. Knowing that there is a delay in the user’s daily commute home due to an accident, allows the user to avoid a troublesome drive home, as well as creating a less busy road way for EMS and other safety or emergency vehicles.   
      
    Nomad takes all of the previously mentioned problems of the daily commuter, and adapts to them, providing solutions before they become problems in a seamless fashion.
* The *Errand Runner*
  + For the errand runner, most days consist of tasks that require completion in a succinct, organized fashion within a specific period. Throughout this user’s day, he/she does not have a lot of control over how long their daily process takes. One thing that this user does have control over, is the order in which he/she runs his/her errands, and the routes that he/she takes from each errand to the next.   
      
    Nomad accomplishes this for the user by finding several suitable routes according to the information Nomad knows about the user, then navigating the user to his/her desired locations in one of the most efficient ways. This gives the user more control over the amount of time they spend traveling to and from their daily activities.   
      
    This user is traveling to places that he/she frequents on a weekly, if not daily basis, and because these places are all but second nature to the user, they should be second nature to Nomad as well. Through Nomad’s ability to learn from the user’s actions, Nomad can then make suggestions to the user about things such as his/her departure time or alternative routes, all in an effort to relieve the user of having to make such decisions (which can be costly without the proper information). Nomad accomplishes this task via its messaging and alert system.
* The *Driving Enthusiast*
  + This user is usually not concerned with the time it takes to get from A to B, because more often than not, there is no B; this user is just driving for the sake of driving. Someone might wonder then, why this user would have any interest in Nomad in the first place. When this user goes to leave on his/her relaxing drive, maybe this user has a direction in mind, or a repeatedly used route, and upon leaving the user finds that the local highway used for beginning the leisurely drive is backed up for two miles. Nomad would have already notified this user, and provided him/her with an alternative route, all without the user executing a single action.
* The *Out-of-Towner*
  + When a user is in an area or region he/she is unfamiliar with, the troubles of navigating start to compound quickly. Not only will this user deal with the usual unpredictable road conditions, but he/she must now also tackle the task of finding how to get from A to B with no knowledge of the local road system, and possibly no sense of direction.   
      
    Fortunately, Nomad is already familiar with the user’s habits and preferences so even in places the user is not accustomed to traveling in, Nomad knows the way that user would travel if the user was accustomed to the area. Having these options makes driving in new places more enjoyable and more efficient for Nomad users.
* The *Local*
  + Becoming acquainted with an area over time (rural or urban) users are aware of most roadways and which routes they prefer to travel. So again the question presented is, why would this user benefit from using Nomad? This user knows how to get from A to B, and how to do it in, more or less, in the most efficient way. What this user still lacks is the knowledge of unknown delay-causing events, such as unexpected traffic, accidents, and weather conditions. However, the real benefit to the user lies within Nomad’s analytics.   
      
    This user often times has multiple options in the routes traveled each day, but the user may be unaware of these alternatives, or may think that the current route he/she travels is superior to the rest, all without backing that assumption with any conclusive or reliable data. With Nomad, this user will never wonder again or be unsure about the routes he/she travels because Nomad will automatically make suggestions about better possible routes, and why the new suggested route is better than the old one. Nomad also logs data about the user’s navigated routes of travel, and that data is available for review and exportation to the user for any additional analytics that the user would like to perform.

Project Requirements

The following section outlines the requirements for Nomad, an application developed for the mobile Android platform. This section of the document will outline the user’s capabilities inside the application, and the level of importance for each requirement.

1. Home screen [Threshold]
   1. The user is able to access information about the current environment provided by Nomad’s HUD. The user can also access other utilities and application functions.   
      1. HUD [Threshold]
         1. The HUD should relay information to the user about current environment and travel data. The user can view data in both the top and bottom margins of the home screen.  
            1. Top margin [Threshold]

The user can view data such as current outside temperature, current address or latitude and longitude, and the local time.

* + - * 1. Bottom margin [Threshold]

The user can view information including, current speed, time spent driving, and direction of travel.

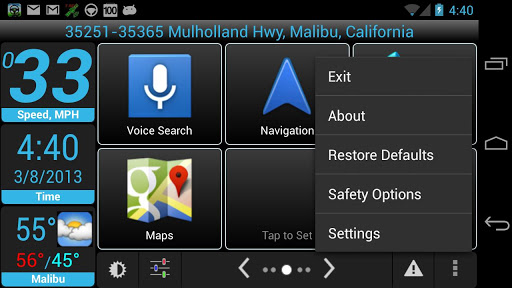
* + 1. Hot pads [Target]
       1. Three of these pads come preconfigured with Nomad’s navigation utility, Android’s voice search function, and Android’s voice dialer. Other limited applications and utilities are assignable to the other unbound hot pads. To do this, the user will simply hold down on one of the open hot pads, and select an application function or utility from the pop-up dialog window.   
          1. Default hot pads [Target]

Selecting these hot pads executes the respective utility allowing the user to access the full capabilities of that utility.

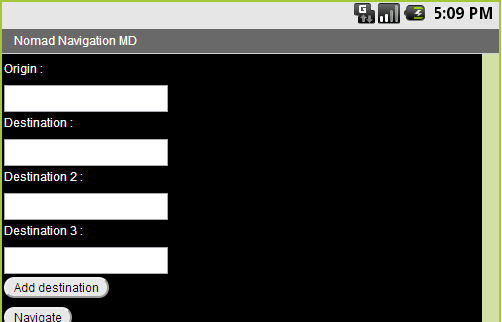
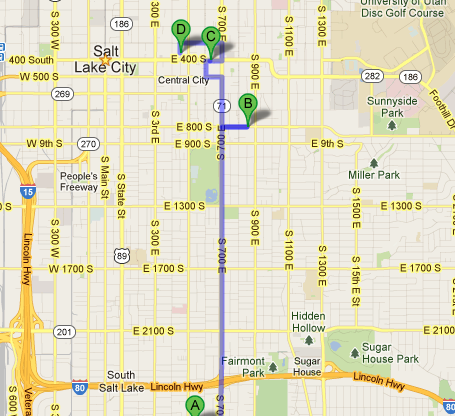
* + 1. Hands-Free Capabilities [Target]
       1. Via the application menu, users can enable a hands-free environment. Enabling this feature forces Nomad to accept only voice commands and disables all touch input. Disabling this feature should require a voice command.

1. Nomad navigation utility [Threshold]
   1. The user can enter a destination(s). In time, the user should not have to select the paths that the user frequently travels; Nomad should be able to make those decisions for the user autonomously.   
      1. Traffic diversion [Threshold]
         1. Divert the user to a different route that has less traffic, or less of an obstacle, than the original route. Nomad executes this operation autonomously if Nomad has enough data about the user’s daily routine to do so.
      2. Nomadic Salesmen [Target]
         1. Allows the user to enter multiple destinations and navigate to each of the provided destinations using the route Nomad believes to be the most desired and efficient route.  
            1. Allows the user to choose between multiple alternative sets of routes [Stretch]
         2. Gathers information about each destination such as hours of operation and the business type in order to validate destination choices [Target]
      3. Timer [Target]
         1. A timer starts when the user starts traveling, and stops when the user reaches his/her destination.
      4. Google navigation [Stretch]
         1. Integrate other native Google navigation options so the user can also utilize all other Google map and navigation functions.
2. Messaging & Alerts [Threshold]
   1. The user can view messages and alerts in the user’s device notification window.   
      1. Traffic & Accident alerts [Threshold]
         1. Make the user aware of bad traffic conditions on the user’s next route. Provide the user with alternative routes to remedy the situation.
      2. Bad weather conditions [Target]
         1. Notifies the user of bad weather that could potentially pose a challenge for upcoming travel plans
         2. Estimates additional drive time required by user to arrive at his/her destination as if there were no conditions causing the delay. [Stretch]
      3. Better routes [Target]
         1. When a user frequents the same route, double-check that the current route is the best route available for that user. If a more efficient route exists, notify the user.
         2. Notify the user why the new route is superior. [Stretch]
3. Caching [Target]
   1. Stores data about the user such as places he/she frequently travels to and routes taken. The application uses this cached data for a specific period to make the process of finding and selecting routes faster and more efficient for the user. Stores this data and reevaluate on a monthly basis to ensure the data remains current and accurate.

Project Specification

1. Home Screen
   1. The user will have executed the application and the user’s device will show Nomad’s home screen. The user will observe a full screen application, landscape orientation, with margins located at the top and bottom of the user’s screen. These margins have within them information pertaining to the user’s current state. The top margin consumes about one-tenth of the screen, and the bottom margin consumes double the space of the top margin, about one-fifth of the screen size. The center of the application’s current screen houses six square application shortcuts, termed “hot pads”. These hot pads are uniform in size and shape, and they display the bound applications icon in the center of the designated hot pad’s area.  
      1. HUD
         1. The heads up display should encapsulate every display item on the home screen except the hot pads. Display the items that make up the HUD in a tabular format. The top margin should mirror the position of the top margin in the image provided below. The bottom margin should resemble the left margin of the image provided, but its position should be underneath Nomad’s hot pads.
      2. Hot pad’s
         1. Upon selecting an application bound hot pad, the application that is bound will launch and carry out its default launch sequence.
      3. Menu key
         1. By selecting the menu hardware button, Nomad will display its menu, which contains things like Nomad’s settings, user preferences, and the log.   
            

(NOTE: The image above is an example of the layout of Nomad’s home screen)

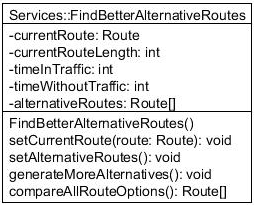
* 1. Upon selecting Nomad’s navigation utility, Nomad redirects the user to another screen containing two options: Nomad single destination and Nomad multiple destinations.   
     1. Single destination
        1. The user observes a different screen containing a start-point input field, populated with the user’s current location, and a second input field, the user’s desired destination. The user then selects the button at the bottom of the current screen, and afterwards a map displays the correct route and navigation begins to the user’s defined destination.   
           
     2. Multiple destinations
        1. This selection will prompt the user with a field for his/her starting location, populated with the user’s current location, and a single destination field. Below the second input is a button to add another destination. Pressing it causes another destination input to appear. At the bottom of the screen is a button whose action will initiate navigation to the first destination. After reaching the first destination, Nomad will begin a navigation route to the second, continuing this process until reaching the user’s final destination.   
             
             
           (NOTE: The image above is an example of multiple waypoint directions from Google maps. Using multiple destinations will result in a map looking something like this image. Nomad would be avoiding the highlighted routes if those routes had undesirable road conditions and selecting better alternatives, and would create a round trip adding a final route from the last destination D back to the origin A.)

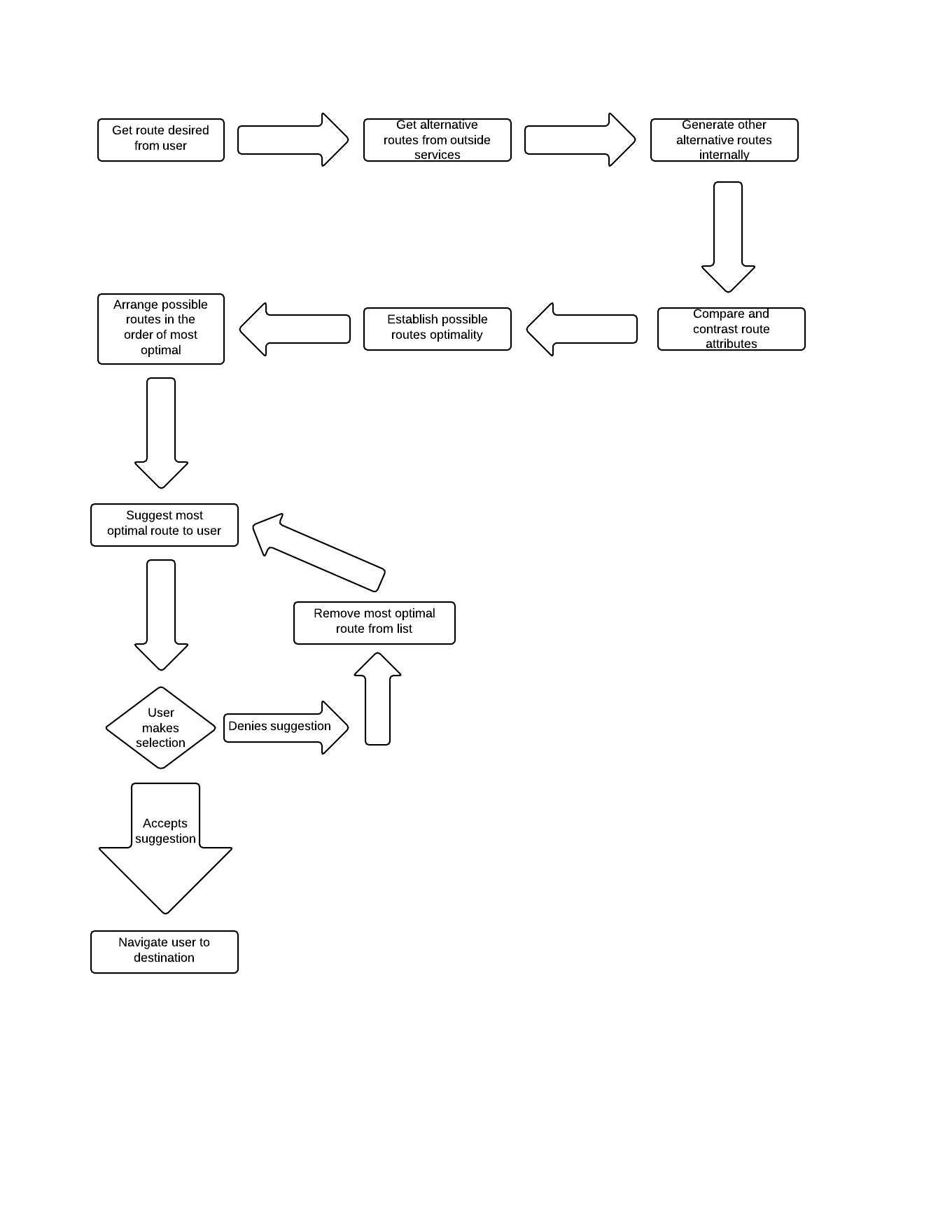
1. Messaging & Alerts
   1. When a user receives a notification from Nomad that notification displays in the user’s device notification window (an example of the notifications location can be seen in the image provided for the home screen specification), then by selecting the notification the user triggers Nomad to launch and display the full contents of the message or alert; as well as the solution Nomad has created to resolve the issue. Message and alerts are loaded in the user’s notification drawer with the Nomad logo and a short description of the message or alert. Some notifications my not warrant any action from the user and may just display information that could be useful to the user, in this case, there would be no option to execute a solution. In the event the notification is a suggestion for a better route, if the user selects this notification, Nomad will immediately begin navigating the suggested route.   
      1. Bad weather
         1. Notify the user via their device’s notification window of poor weather conditions, and if possible provide information in the description about possible time delays the aforementioned conditions may cause.
      2. Better routes
         1. Notify the user about better alternative routes when they are available. Displayed in the user’s notification window accompanied with a brief description of the messages details.
      3. Pre-problem notifications
         1. Based on user data, Nomad determines the user’s travel patterns and makes suggestions based on other available data. Roughly 15-30 minutes before the user heads home from work, Nomad would notify the user “there’s bad traffic on your usual route”
2. Caching
   1. Caching is not something the user generally “sees” happen, although the user does have access to view logged data inside Nomad. If the user should so choose, he/she can view cached data such as routes and frequented locations. Users can retrieve this information by using the menu button from the application’s home screen and selecting the “Log” option. The log option then redirects the user to another screen, where he/she can view the aforementioned data in a tabular format.

Project Architecture

**Finding optimal routes**

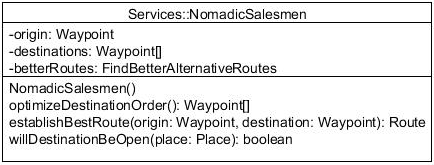
The ability to compare and contrast possible alternative routes is the most significant process of Nomad. This process involves both the creation of new routes, as well as the task of processing routes, and route information, provided by Google’s Direction, Places, and Distance Matrix services. This service in Nomad also utilizes non-Google services such as Bing’s traffic and incident reporting and MapQuest’s road data. This class, named “FindBetterAlternativeRoutes”, has a variable, “currentRoute”, that is set once the user selects both an origin and destination. Nomad then sends a request, with the route attached, to Google’s Direction service; Google sends back a response with all route data, which Nomad then parses and places into a model object. Nomad begins analyzing the new current route against other possible routes, generated by both Nomad and Google. Bing and MapQuest’s services also receive requests about the same route to fill in any gaps in information.

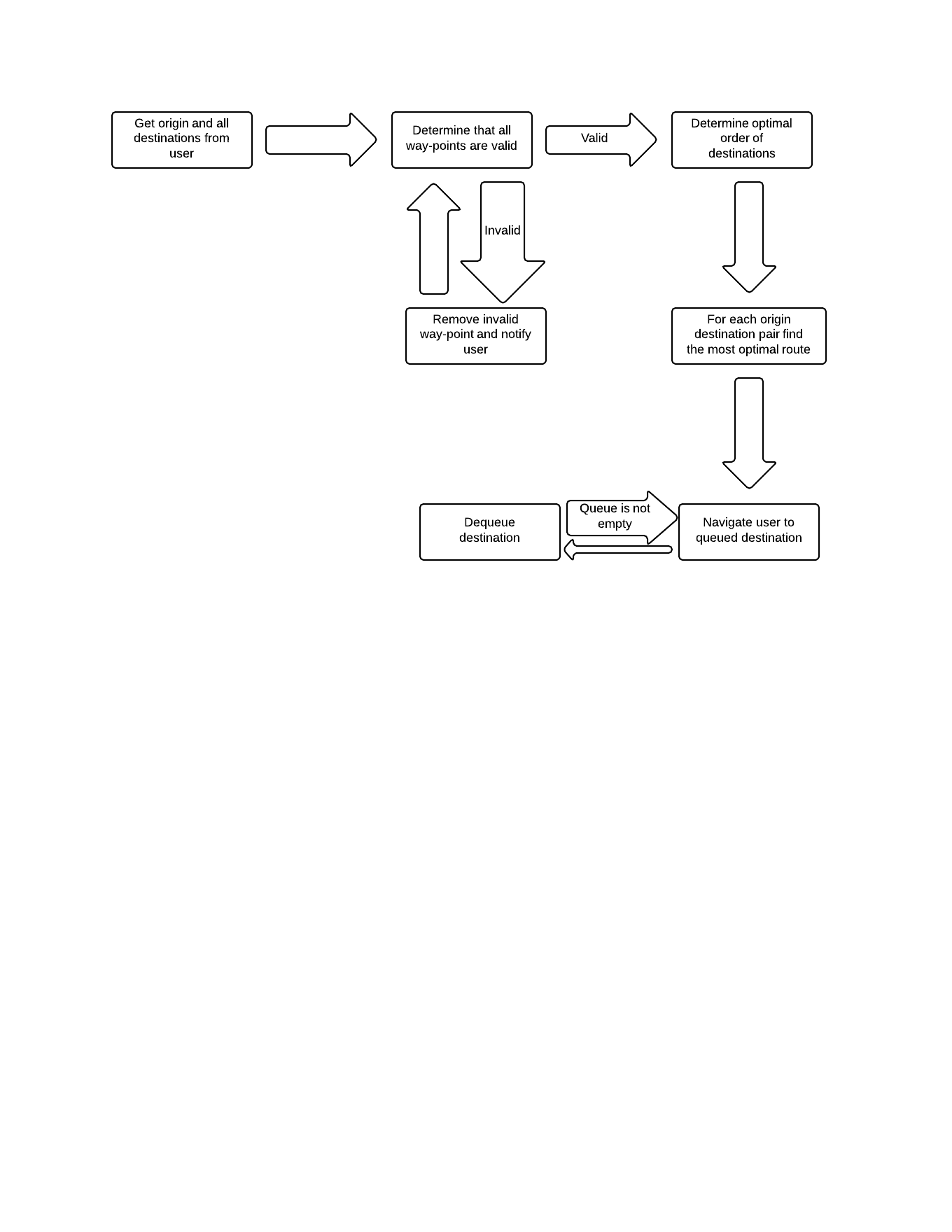


Finding optimal routes flow chart: 

**Nomadic Salesmen**

The Nomadic Salesmen function takes a set of locations or “waypoints”, and first determines the proper order in which to travel to each destination, then, takes the ordered list of waypoints and selects for each pair of waypoints the most efficient route between the two waypoints. Nomad’s service that finds better routes establishes the optimal route from each waypoint to another. Google’s Direction service provides the waypoints for Nomad to process, and Google’s Places service provides information about the waypoints if they are businesses. Using the Places service, key attributes of businesses, such as hours of operation, will help determine if waypoints are even worth traveling to at that given time. It is only after parsing all the aforementioned data and analyzing it that Nomad will produce a multi-destination route.





Technological Description

I have chosen to develop Nomad on the Android mobile platform. My reasoning behind my decision involves several factors: Android is open-source and the majority of people using smart mobile devices use Android devices. It is a platform I have little experience developing on, and the Google/Android development community is something I want to be a part of and work in.

With Android being open-source, it provides endless resources of tried and tested methods and practices for developing mobile applications, as well as endless supplies of advice and knowledge on the subject for new and upcoming developers. Although I have never developed on the Android platform, I do have heavy experience developing in Java environments and the two have similar syntax and very closely related concepts; this provides a technical challenge without diluting my development cycle with the constant need to do research. Most of the people I know have and enjoy Android devices, including myself. This was a significant influence on my decision to develop this application with Android. I wanted to build something that I could not only get a grade for and show off to friends and family, but also potentially release onto the Android Play Store and have others reap the benefits of my creation.

For third-party services, I have chosen primarily to use Google’s Map, Directions, Places, Distance Matrix, and Geocoding services and API to facilitate fetching the data Nomad requires to operate and perform its essential functions. I have also chosen to use Bing and MapQuest to pull additional data necessary to Nomad, which Google does not provide. I chose to use Google services primarily because Nomad is an Android application, so the two go together nicely. In addition, Google’s services send the data they return via XML and JSON, of which both Android handles and parses without a problem. I chose Bing and MapQuest because their traffic and accident data is far more accurate and accessible than Google. Google in fact does not offer out its traffic data at all, and does not release accident data.

My IDE for the project will be Jet Brains IntelliJ 12. IntelliJ offers a very accessible and intuitive design, and it supports Android application development. I am very familiar with IntelliJ and have done much of my Java development in its environment, which is the primary reason that I chose it as my development environment for this project.

Development Process

For this project, I have chosen the agile development process. An agile development process reduces the risk of projects failing or loosing value because this methodology has the ability to adapt to changing circumstances and environments quickly, before potentially overlooked factors kill said project. Having the constant cycle of development and testing, reworking, and then testing again, will help insure that I identify roadblocks and overcome them quickly before they cause more issues. Using the agile structure will open up more personal freedom in making slight design or functionality changes without much reworking of whole, larger pieces of functionality within the application. Using an agile methodology, I will be developing pieces of functionality on a daily basis in four-day intervals and testing those individual pieces independently, using the fifth day of the week, and weekends, to test the functionality created earlier in the week as a whole unit. Choosing this kind of process will allow adequate time to develop, debug, and test all functionality properly within a finite period.

Development Schedule

* Week 1
  + Monday
    - Home screen
      * Page wire frame
        + Template for placement of items on home screen
    - Data
      * Place calls to specified services and start designing how to model data and domain models
  + Tuesday
    - Home screen
      * HUD
        + Both top and bottom margins display dummy data
    - Data
      * Place calls to specified services and start designing how to model data and domain models
  + Wednesday
    - Home screen
      * Hot pads
        + All six hot pads appear on screen with no default application shortcuts
        + Holding a pad down displays applications the user may bind to it
    - Data
      * Place calls to specified services and start designing how to model data and domain models
  + Thursday
    - Home screen
      * Hot pads
        + Three defaulted hot pads successfully display their default application shortcut
        + Selecting a hot pad launches the application bound to it
    - Data
      * Place calls to specified services and start designing how to model data and domain models
  + Friday
    - Home screen
      * Test all implemented functionality as a whole
      * All functionality should pass
    - Data
      * Create data and domain models for service data
* Week 2
  + Monday
    - Nomad navigation utility
      * Request routes and other data from third-party services
      * Analyze/Map response data
  + Tuesday
    - Nomad navigation utility
      * Create model classes based on mapped response data
      * Parse data from response
  + Wednesday
    - Nomad navigation utility
      * Continue parse function for service response data
      * Create model objects to store parsed data
      * Display data saved to model objects
  + Thursday
    - Nomad navigation utility
      * Single/Multiple destination selection screen
        + Design and build wire frame
        + Work with placing buttons on screen to select either option
  + Friday
    - Nomad navigation utility
      * Test week’s functionality as a whole working in unison
        + Test service response
        + Test parse function for response
        + Test single/multiple destination buttons
* Week 3
  + Monday
    - Nomad navigation utility
      * Single/Multiple destination redirection to appropriate next screen(s)
      * Place input fields for single/multiple destination navigation
  + Tuesday
    - Nomad navigation utility
      * Single destination navigation submits origin and destination to service
      * Service returns route and application parses data correctly
      * Map is generated with provided route
  + Wednesday
    - Nomad navigation utility
      * Enable multiple destination inputs
  + Thursday
    - Nomad navigation utility
      * Send multiple destinations to services
      * Receive response from services and parse data
      * Save data in model object
      * Display generated map
  + Friday
    - Nomad navigation utility
      * Test single/multiple destination functions working together
* Week 4
  + Monday
    - Nomad navigation utility
      * Capture alternative routes
      * Compare alternative routes
        + Find shortest
        + Quickest
  + Tuesday
    - Nomad navigation utility
      * Compare alternative routes
        + Find route with least amount of traffic
      * Return most optimal route
  + Wednesday
    - Nomad navigation utility
      * Order optimal routes in order of most optimal to least
      * Give user option of selecting which alternative to navigate
  + Thursday
    - Nomad navigation utility
      * Implement weeks functionality for multiple destinations
  + Friday
    - Nomad navigation utility
      * Test single and multiple best alternative routes
* Week 5
  + Monday
    - Nomad navigation utility
      * Implement timer function for each type of navigation function
        + Single and multiple destination
      * Pull accident data
      * Find a path in the alternative paths for a route that avoids traffic
  + Tuesday
    - Nomad navigation utility
      * Nomadic salesmen algorithm
        + Check that all way points are viable travel options
  + Wednesday
    - Nomad navigation utility
      * Nomadic salesmen algorithm
        + Optimal order for destinations
  + Thursday
    - Nomad navigation utility
      * Nomadic salesmen algorithm
        + Optimal routes for all destinations
  + Friday
    - Nomad navigation utility
      * Test/ continue work on optimal routes for all destinations
        + Test through weekend if necessary
* Week 6
  + Monday
    - Messaging and alerts
      * Create message and alert models
      * Get message and alert to show in notification window
  + Tuesday
    - Messaging and alerts
      * Create bad traffic/accident alert
      * Give suggestion for alternative route
  + Wednesday
    - Messaging and alerts
      * Create bad weather alert
      * Calculate additional drive time required
  + Thursday
    - Messaging and alerts
      * Create better route situation and identify the better route
      * Notify user of a better route
  + Friday
    - Messaging and alerts
      * Test messaging functionality in different situations
* Week 7
  + Monday
    - Caching
      * Save traveled routes on device
      * Parse information back out
  + Tuesday
    - Caching
      * Save locations traveled to
      * Parse information back out
  + Wednesday
    - Caching
      * Enable updating once a month
  + Thursday
    - Caching
      * Create analytical function for autonomous actions
  + Friday
    - Caching
      * Testing saved data and parsing back out
      * Test once a month updating
* Week 8
  + Monday
    - Caching
      * Continue developing analytical functions for autonomous actions
  + Tuesday
    - Caching
      * Continue developing analytical functions for autonomous actions
  + Wednesday
    - Nomad navigation utility
      * Create autonomous bad traffic redirection
  + Thursday
    - Nomad navigation utility
      * Create autonomous bad traffic redirection
  + Friday
    - Nomad navigation utility and Caching
      * Test autonomous functions working in unison
* Week 9
  + Monday
    - Automation
      * Anything left over from week 8
      * Compose data about places traveled on certain days or during certain times, or the combination.
  + Tuesday
    - Automation
      * Create possible places the user could be traveling to
  + Wednesday
    - Automation
      * Determine which place from possible places is mostly likely to be the user’s destination
  + Thursday
    - Automation
      * Integrate all bettering of routes to automated processes
  + Friday
    - Automation
      * Test available functionality
* Week 10
  + Monday
    - Automation
      * Integrate all bettering of routes to automated processes
  + Tuesday
    - Testing / Fill gaps
  + Wednesday
    - Testing / Fill gaps
  + Thursday
    - Testing / Fill gaps
  + Friday
    - Testing / Fill gaps

Resources

Since the Android language and development environment are new to me, resources will be especially essential to the success of my project. The resources that will aid me in becoming proficient enough in the Android language to complete my project will be a combination of on-line and literature sources. The online sources will consist of the Google developer’s resource, which includes the Android API, support forums and social support circles, examples, tutorials, and instructions on Android’s use, as well as Android’s specific developer site. I will also be utilizing online tutorials, as well as peers who have previous experience working with the Android language and environment. My literature resources will consist of *Android Application Development for Java Programmers* by James Sheusi, and *Professional Android 4 Application Development* by Reto Meier.

Version Log

1. Draft D2
   1. Reviewed by Matt Warner – March 13, 2013
      1. Suggestions made
         1. Shorten executive summary
         2. Move current executive summary to the product description
         3. Rewrite executive summary
         4. Concentrate on the first five parts of the proposal, they are the most important
         5. Match requirements with specification and the rest of the proposal, be consistent
      2. Changes made
         1. Made all changes Matt suggested
         2. Finished parts six through eleven of my proposal
         3. Removed a lot of fluff from first few sections
2. Draft D3
   1. Reviewed & approved by Kristen Parker – March 14, 2013
      1. Suggestions made
         1. Follow the revisions marked on draft D2
         2. Email Matt for potential final pass off
      2. Changes made
         1. Fixed grammatical and usage mistakes
3. Draft D4
   1. Reviewed by Matt Warner – March 20, 2013
      1. Suggestions made
         1. Complete project description, requirements, specification, and development schedule and resubmit by March 25, 2013 at noon.
      2. Changes made
         1. Adjusted necessary sections based off of Matt’s feedback
            1. Revised project description to include competitors and how I have taken their models and design into consideration
            2. Elaborated on specifications and included screen shots
            3. Added necessary requirements
            4. Adjusted schedule to be more agile-like
            5. Added version entry to log