Specifications Document

Redha Elminyawi <relminya>
CS190: Software System Design
2/11/2005

Description

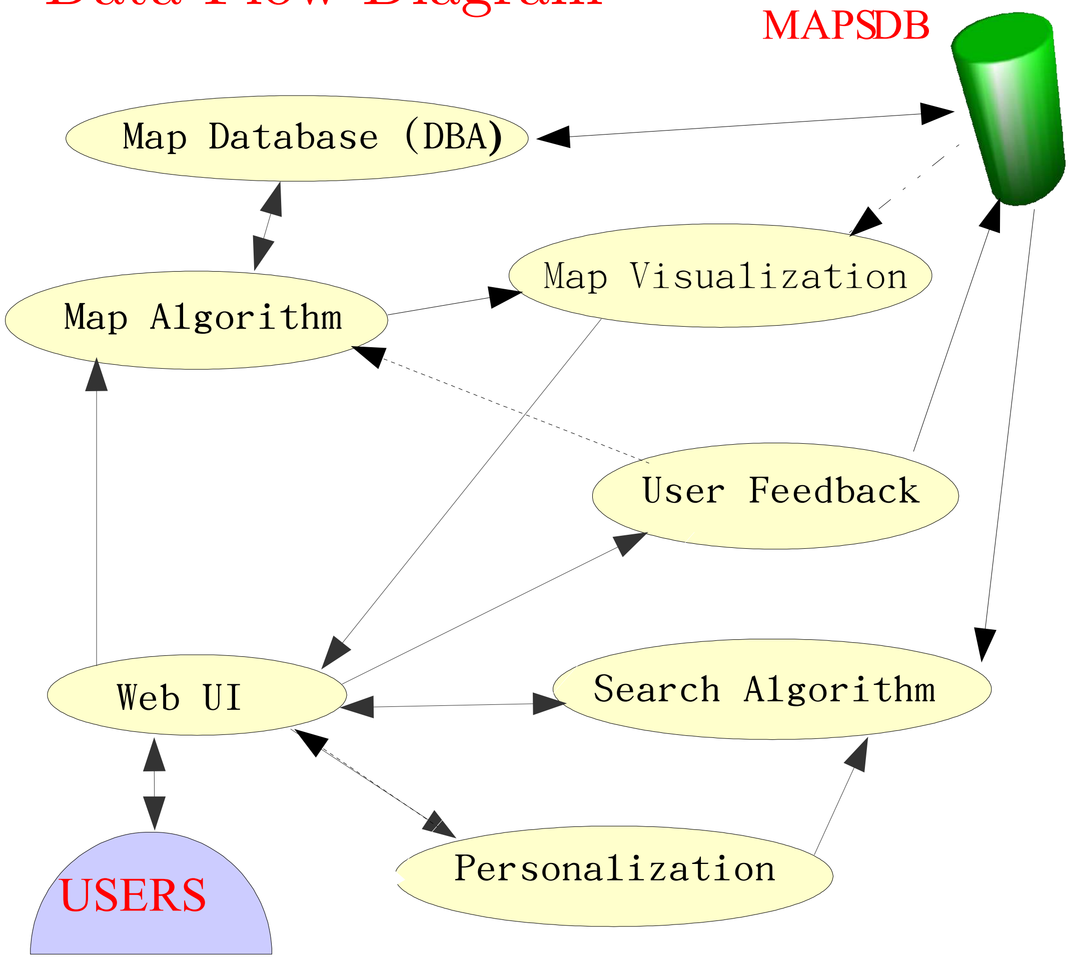
Existing solutions to aid cyclists in route creation include paper maps, highlighters and an even lower tech approach to distance estimation - string and a map scale. The following specifications document for BikeQuest will enumerate the features necessary to solve the route discovery problems of local cycling enthusiasts. BikeQuest will be a cyclist friendly mapping program. In addition to the instant creation of both point to point and loop routes, we intend to give the user relevant information on road conditions, traffic, and scenery among other route qualities. Since this data is constantly changing, BikeQuest will accept user input detailing the routes' property changes, and react accordingly with altered route information upon search.

BikeQuest will most likely be most useful for both local recreational cyclists and those cyclists who are new in Rhode Island. Point to point routes and information presented by the software on traffic may also make it marginally useful to commuting cyclists, who we assume already have set routes to get home or go to work.

In addition to mapping tools and rating, I propose that we extend functionality specifications to let BikeQuest become an online cycling community. Support for site personalization, and a blog like open ended ratings section will increase the amount of cyclists interested in the software, as these value adders will differentiate the site from existing disparate tools, like Mapquest and Yahoo! Groups.

System Model

Fig. A.
Data Flow Diagram



A word about the arrows: arrows signify directional data flow. For example: the Web UI receives data from users, and then sends data back to users. Ellipses are temporary data stores or data processing centers, cylinder signifies persistent data storage.

Containment/Data Flow Diagram Explained

Web UI

Functionality: View for whole software, takes in all info from users and sends commands to process data to Mapping and Search algorithms. It receives data mainly from MapVis and Search. See Updated Requirements, and UI section for more detail.

Problems/Test: I feel that since the UI interacts with almost all of the packages directly, leaving more room for error in integration.

Map Visualization

Functionality: Takes raw points from mapping algorithm, in combination with our map database, returns a picture of the route desired, with text description.

Problems/Test: This could be very complex to do as I don't know the specifics of this (LineDrive was a 224 project...hmm I personally think I'll stay away from that, all in BikeQuest's theoretical group's interest)

Map Database (DBA)

Functionality: Data-store and interfaces to it. It's information effects the Mapping and Search algorithms. Receives user rating information.

Problems/Test: Biggest problem is that changing GIS data into a format useful to other team members will be tedious, and possibly very time consuming. Tests include: trying to break database by addition of too much route information, and trying to ruin the database by user entry.

Map Algorithm

Functionality: Takes in search terms from UI, and searches the database for the correct route. Then sends the relevant information to MapVis.

Problems/Test: This could also be very complex to do. To test, we will create drivers that will feed in a lot of sample points, even some that make no sense at all, and see that the correct response is given: be it rejecting the search terms, or displaying maps.

<u>User Feedback</u>

Functionality: Takes in user ratings data, and persists it to the database. Secondly, it has a loose connection to the Map Algorithm, in that it could effect route data if we change our Map data based on user feedback. Loosely coupled to Search algorithm, in that the search algorithm depends on the data persisted by UF.

Problems/Test: Not enough feedback can be a huge problem for this software, because without the feedback we basically have mapping software that aims to give bike friendly routes, but never evolves. The fact that the routes given are bike friendly is also up to debate if there is no feedback. To test, drivers will be created that pump in random ranking numbers, and check that the rankings are updating our database and view.

<u>Route Search</u>

Functionality: Will return search results to UI based on information input by the user. Also will receive information from BikeQuest database, and personalization package. Search rank algorithm will change based on personalization.

Problems/Test: This will be tricky to test since the search changes on dynamic information, and predicting the real in order results for test purposes may not be easy based on very large data sets. Testing may have to be in terms of general trends in searches.

<u>Personalization</u>

Functionality: This package receives and sends information from the front end. This information is detailed in the features section. It also sends some information to the search package, to alter search based on user. If the myRoutes feature is implemented it will persist data to the map DB.

Problems/Test: I am not familiar with security and login things, but I bet someone in the class is. To test we can create a multitude of fake users (test cases) and attempt to use and break our functionality, such as route saving. Functionality such as suggested routes should be tested to ensure either sane results or no results at all (Amazon doesnt always come up with suggestions).

Divisibility

Each of the sections above need at least one person. Candidates for two coders include Map Visualization, Map Algorithm, and Route Search. These all need two people because of the complexity of the algorithms, and the endless possibilities for errors in implementation. This leaves us with a group of 7-10 people, not including documentation, dedicated testers, or project manager. Since the class only has 17 people, the anticipated personnel overhead is a reason this project may not be a good fit for CS190. A solution could be to rotate programmers between the functional groups to assist a section that needs it. Yet another solution could be to limit scope after further probing a user base with surveys to determine what is most desirable.

Flow of Control

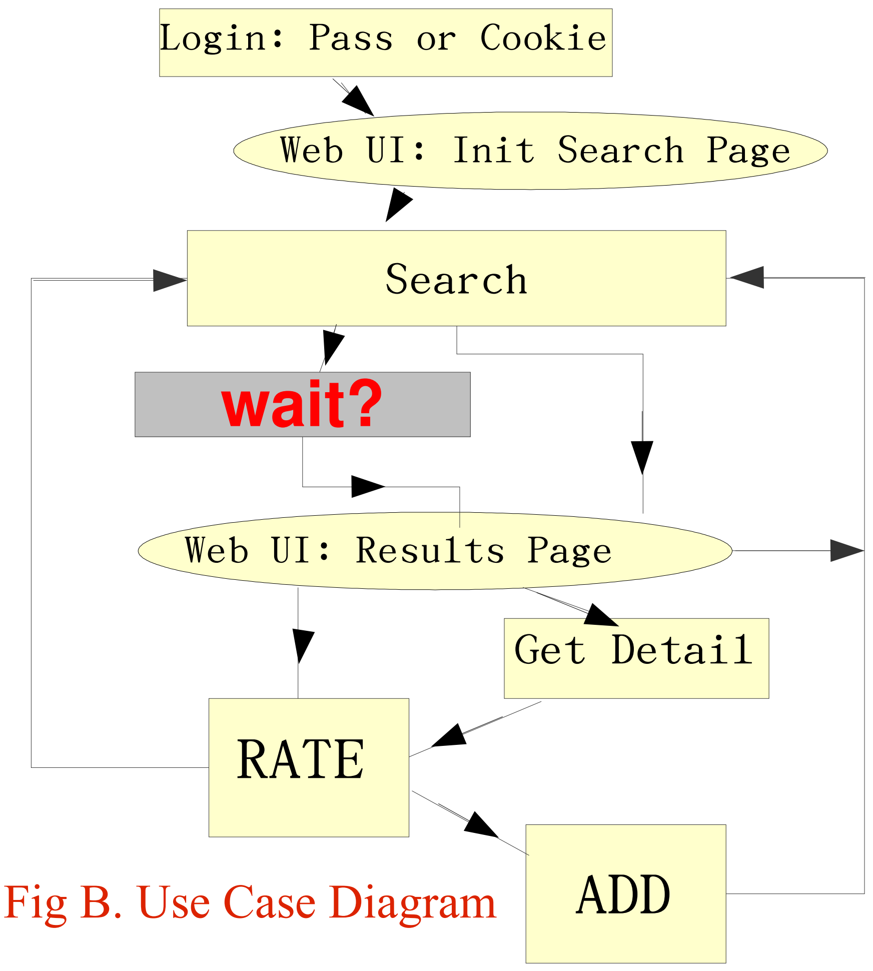
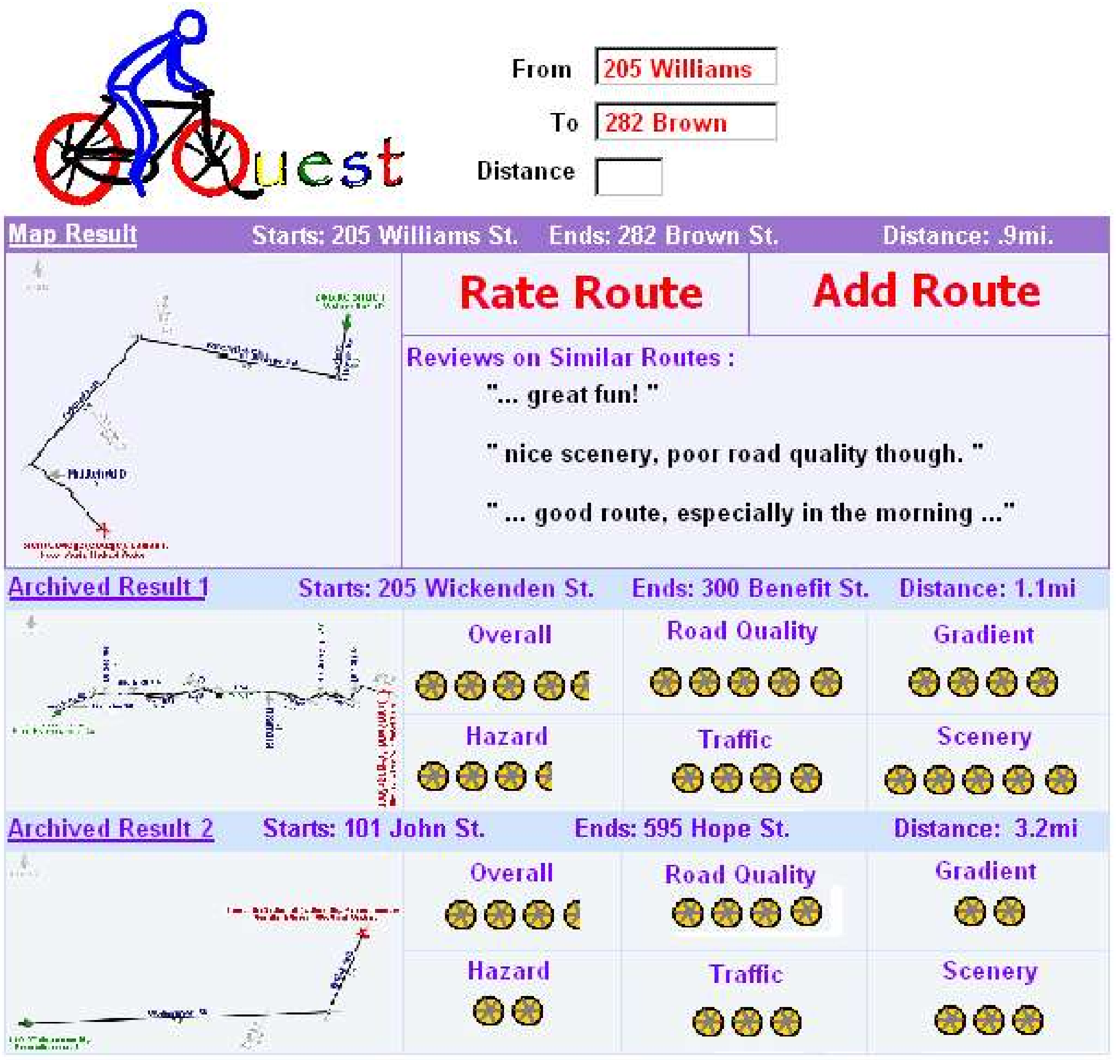


Fig B. Use Case Diagram

Flow of Control – User Clickthrough Explained

User will go to website address and either be met with a customized start page based on cookies on client computer or a login and password screen. After this there is an initial search page with fields for start point, end point, and distance desired. An advanced search page will also be available, and will include terms for the road quantifiers we include. When a user searches they may have to wait some time (order of Orbitz search) to get their results dependent on how fast our algorithm is. With search results in hand a user can then select the map generated route, or a databased route. After this the user can rate, and depending on the source of the map, add the route to BikeQuest's database. At any point during a user's trip to BikeQuest, they will be able to search again.

User Interface


From 205 Williams

To 282 Brown

Distance

Map Result	Starts: 205 Williams St.	Ends: 282 Brown St.	Distance: .9mi.

Rate Route | Add Route

Reviews on Similar Routes :

"... great fun! "

" nice scenery, poor road quality though. "

" ... good route, especially in the morning ..."

Archived Result 1	Starts: 205 Wickenden St.	Ends: 300 Benefit St.	Distance: 1.1mi

Overall	Road Quality	Gradient
●●●●◐	●●●●●	●●●●
Hazard	**Traffic**	**Scenery**
●●●◐	●●●●	●●●●●

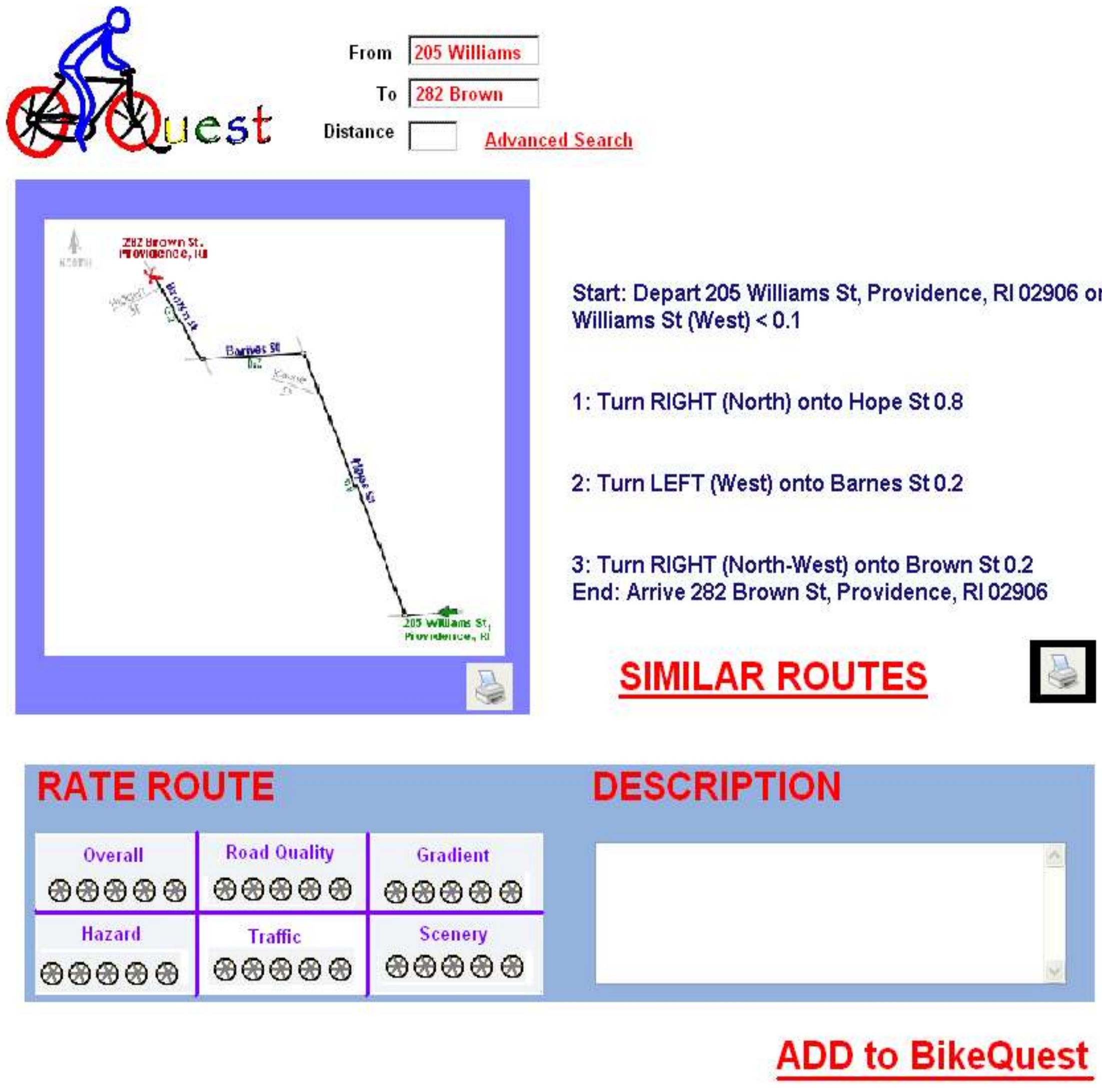
Archived Result 2	Starts: 101 John St.	Ends: 595 Hope St.	Distance: 3.2mi

Overall	Road Quality	Gradient
●●●◐	●●●●	●●
Hazard	**Traffic**	**Scenery**
●●	●●●	●●●

All apologies to Google for design adaptation :-)

Search Results Page:

Once a user hits enter on their search terms a page as above should display. We will attempt to have all pages in the website have an area to search again. The first result that shows up on a user's search should be the result of a mapping algorithm with the exact specifications the user entered, maximizing overall quality of route by weighting different paths by road quality, traffic, and other qualities. The user will be able to both rate and add the route to MapQuest, if it is not already within the database. The next results will be those routes closest (to be determined by search algorithm) to the original search terms, with rating information displayed beside the map.



Start: Depart 205 Williams St, Providence, RI 02906 on Williams St (West) < 0.1

1: Turn RIGHT (North) onto Hope St 0.8

2: Turn LEFT (West) onto Barnes St 0.2

3: Turn RIGHT (North-West) onto Brown St 0.2
End: Arrive 282 Brown St, Providence, RI 02906

Map Detail & Ratings Page:

Upon clicking an unrated map (mapping algorithm generated) from BikeQuest the page displayed should look as above. The map visualization should either be in a line or full cartography map format. The directions should be brief, and possibly pictorial (e.g Mapquest). An option to print both the map and directions separately should be available. After rating the route, the user will be able to add it to the BikeQuest database. We have users rate based on the six characteristics shown above, allowing a text description which the snippets from the first page will be taken from. Adding to the database will only be allowed if the ratings are complete. Similar routes will also be accessible from this page.

Non-Functional Requirements

Performance: BikeQuest does not need to be as fast as existing route finding solutions (e.g. Mapquest, MSNmaps...) that simply use a least distance approach. Since the search I propose will use additional weighting factors that are input by users I feel that it is reasonable for users to expect the search to take the order of time that a flight search on Orbitz would take. This is further elucidated in the Updated Requirements section.

Testing: Testing will take three different phases. The first will be individual functionality testing for the code packages described in the System Model by drivers with test data. Secondly, a full integrated system test will be necessary. This will need to be a code-torture test of sorts so that we ensure our users aren't receiving routes that will send them on poor rides. Lastly, we will have to bike the routes that are input to the database to see if the algorithm is keeping honest. While this is not so traditional for testing, this could be a very fun built in activity (especially in the spring).

Reliability: As with any web service the site needs to be on 24-7 and not fall victim to server problems. A good test plan needs to be in place to ensure that the site's search results are reliable enough to ride.

Ease of use: System will have to be very easy to use. This extends from having a simple, yet powerful, front-end, to creating a bike ergonomic method of printing directions and maps.

Portability: Since the proposed system is to be on the web, it should be accessible anywhere with an Internet connection. Since our scope will be limited to Rhode Island, it will be useful only to those in the Ocean State.

Documentation: Documentation will include some on screen help tips on screens that are heavy on the user-input side.

External Dependencies: We will need to find GIS data and parse it into our own database, which needs to be set up as well (PostgreSQL, MySQL?) . We also need a loyal user base who will help beta-test before we start. Lastly, we need access to software to aid in web-scripting, which the department almost certainly has.

Updated Requirements

Web UI

- [High] Very simple interface for Basic Search, including ST., END, DIST.
- [Med] Interface for Advanced Search, including terms for Traffic, etc.
- [High] Help and Explanation pages
- [High] Search results are presented in an orderly fashion. Ability to easily tell which routes are from database, which arent.
- [Med] Search Results show review snippets, and review ratings.
- [High] Search loading graphic if search takes over 1 minute.
- [High] Ability to add route to BikeQuest database.
- [Low] Create mobile interface so information can be reached by cell phone.
- Map detail pages include:
  - [High] Map, and text route information
  - [Med] Print buttons on screen which format map/text in bike-friendly format.
  - [High] Ability to rate route on screen
    - Overall[1-5]Low-High
    - Traffic[1-5]High-Low
    - Scenery[1-5]Low-High
    - Road Condition[1-5]Low-High
    - Hazard Conditions[1-5]High-Low
    - Gradation[1-5]Low Variation – High
    - Free Description area

Map Visualization

- [High] Accepts geographical data points and returns a .gif or .jpg image of the route and a text description of the route.
- [High] Map in either hand drawn line or cartesian format.
- [Med] Map in both hand drawn line and cartesian formats.
- [High] Ability to turn map and text into a bike-friendly printout.
- [Low] Dynamic maps that tell ratings of area currently in focus.

Map Database (DBA)

- [High] Parsing of an external format DB (RIGIS, TigerMaps) into our own database that only includes BikeQuest relevant information.
- [High] User database that contains the user's favorite routes.
- [High] Pool of saved routes.
- [High] Ratings database, one set for each saved route.
- [Med] Ability to turn DB tables to XML, for easier processing on UI side.

Map Algorithm

- [High] Dynamic route finding – both point to point routes, and loop routing done by accepting a distance and start point. Loop routing will return one result (not inf.) that is the best based on available ratings information.
- [High] Routes will match search term, or return with "SORRY!" page.
- [Med] Ability to compare routes and tell search components the degree of variance.
- [Low] Algorithm is smart enough to route users away from trouble spots during only the time they are actually trouble spots. Since we get this information from users' input, we will have to get confirmation of end time of danger (e.g. high traffic, construction)

User Feedback

- [High] Archive user statements, and rankings to database. Do rankings calculations with this information.
- [Med] Text parser to find good snippets from user statements.
- [High] Smart system for handling cases of no user feedback on streets.

Route Search

- [High] Databased route finding – return top ranked results, determined by an algorithm which considers ratings, and physical distance from original search terms
- [Med] Determine which mapping algorithm routes are too close to an already archived route, and disallow 'ADD' feature.
- [Low] Search results take past searches into effect to customize routes based on per user preference.

Personalization

- [High] Either a login/password screen, or cookie login to differentiate users.
- [High] Personal myRoutes library to save prior routes that appealed to user.
- [Med] Route suggestions to user on splash screen based upon routes similar to those in myRoutes library.
- [Low] Route suggestions to user based on memory of what user searches for.
- [Med] Front page danger alerts for reported poor/dangerous conditions on routes that appeal to user, using myRoutes as a reference.

Risks ... and Rewards

- **Risk One: Death by convergence**.
    - Since BikeQuest is essentially a convergence tool - a biker's blog with a biker oriented map search, we run the risk of our user base simply using a blog plus MapQuest to do the tasks BikeQuest does for them.
- **Risk Two: Driving the site with user input.**
    - Will the users take the time to input the ratings information that our search will depend on to learn.
    - Any algorithm can be 'gamed', like Google's. For example, will advertisers attempt to input routes that lead by their stores?
- **Risk Three: System administration for the long term.**
    - Physical road changes will necessitate changing GPS database information.
    - Our search algorithm will need to be changed if it is gamed by users.
- **Risk Four: Development risks.**
    - Students will have to learn GPS format RIGIS and convert it into our own database, which may be tedious and time consuming.
    - The wide scope necessary to make BikeQuest useful could put the project overboard in terms of the time allowed to develop.
    - As mentioned in the divisibility section, there may be too much for 10 people to split up.
- **Ultimate Reward**: If done well, with value adders like personalization, this can be a revenue generating site. We can attract new users by duplicating the site across the US, and doing mutual advertising. Advertisements in bike shops in exchange for BikeQuest advertisements will bring new users, and hence more money.