# BikeQuest

**- Top Level Design Document -**

**Hubert Tse (htse)**

**2/23/2005**

## Project Overview

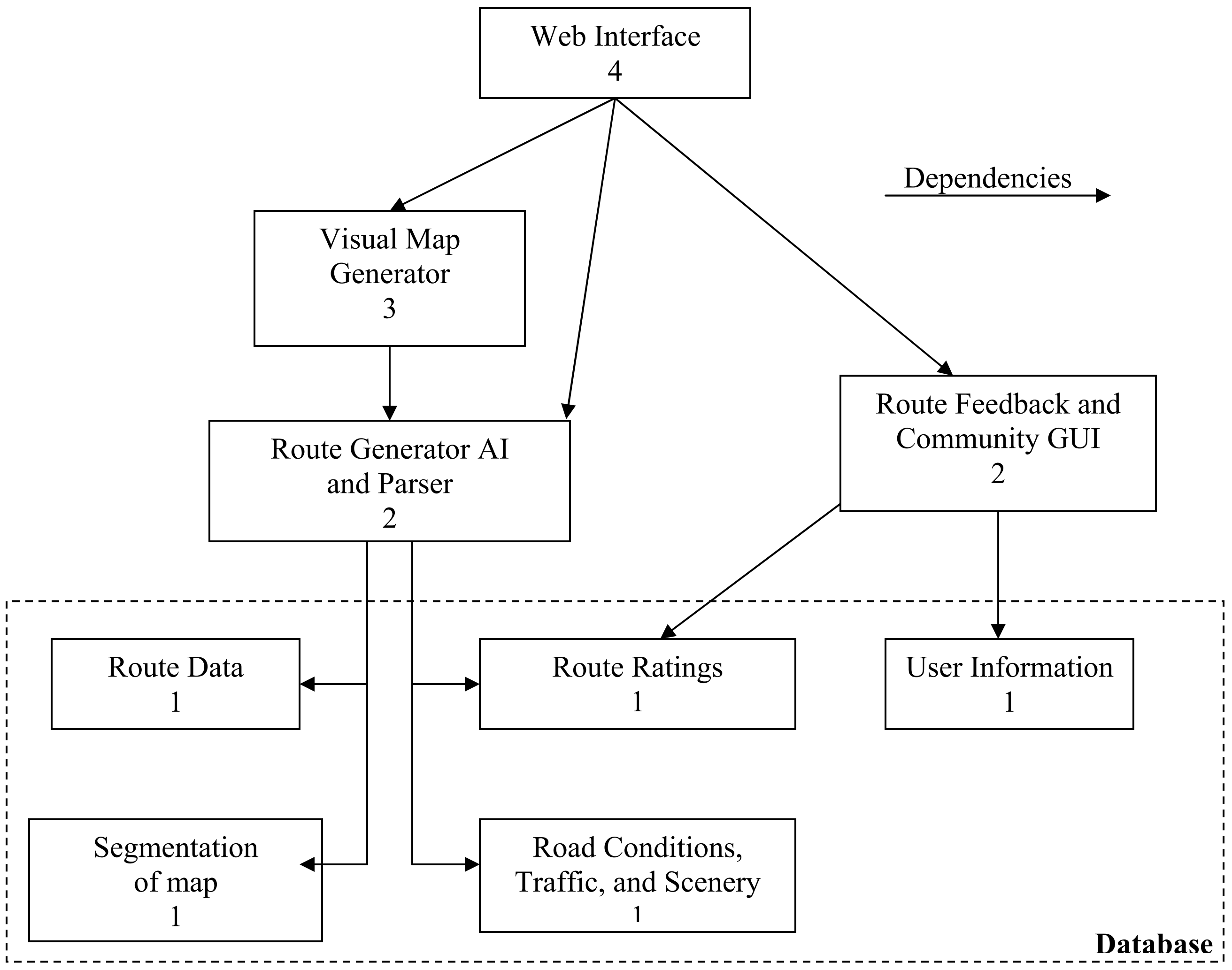
**(Excerpt from Redha Elminyawi's specifications document):**

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. 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.

## Levelized High-Level Components Diagram



## Component Descriptions

### Database

The database will store all the information which includes routes, conditions, feedback and user information. The database structure will be in the 3rd Normal Form. Every major component within the database can be built separately without depending on each other.

#### Route Data

This table will contain all the routes generated by the route generator and user input ones. Each route will be identified by a unique id (key). The stored route should be in an easy to store and parse format. (Coordinate system with delimiters?)

**Route Ratings**
This table will contain all the feedback given to the routes in the Route Data table. This includes numerical rating (rating scale 1-5) and user feedback in terms of words.

**Road Conditions, Traffic and Scenery**
This table will contain current data for all the roads' characteristics. This information includes the road's condition, traffic and scenery ratings.

**Segmentation of map**
This table will contain how the map of Providence, RI will be divided up into sections. This division is important to limit the boundaries of the routes. This will also speed up the time it takes to create routes with specific boundary limits imposed by the user.

**User Information**
This table will contain all users' information. This includes username/passwords, contact information, favorite route bookmarks and other information about the user preferences.

## Route Generator AI and Parser

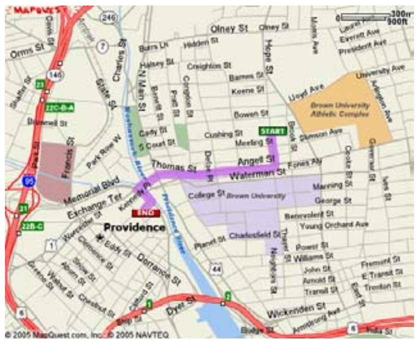
The Route Generator AI and Parser is essentially the largest component. The component's function is to generate routes and parse it to/from the database.

The Route Generator takes in the user's preferences and generates a route. The route can be loops or direct point A to B paths, depending on user input.

The Parser will take the route and parse it into a format that is easily storable in the database. The Parser must also be able to take the formatted route from the database and transform it back into a route that the Generator can understand. This data will essentially be passed onto the Visual Map Generator for the user to see.

## Visual Map Generator

The Visual Map Generator will take the route from the Route Generator and display it to the user through the web interface. The Visual Map Generator should be able to zoom in and out, and pan. Below is an example of how the map may look like with the route.


START
END
Providence
Brown University
Brown University Athletic Complex
Angell St
Waterman St
Olney St
Hope St
© 2005 MapQuest.com, Inc.; © 2005 NAVTEQ

### Route Feedback and Community GUI
This component will handle all the community interactions. This includes a front for users to give feedback about certain routes and for users to interact with one another. This system could expand into a blog like feedback model and even a trip organizer for community bikers to get together and go on biking trips.

### Web Interface
The Web Interface is the shell of the whole system. The user will interact with the route system through solely the web interface. The Web Interface will take in all user input and pass it onto the components it is dependant on. The data returned will be displayed to the user.

## External Dependencies

**Biker Community**
The success of this project will depend largely on community feedback. The feedback users give to the system makes the Route Generator AI "smarter" when generating new routes or simply displaying one that has good feedback. The biker community also helps keep the information up to date. This is very important to keep people coming back and using the service.

**Administration/Webmaster**
With a community there must be administration, and someone to look after the system. The system must be updated with information like new area developments and data integrity checks. There are bound to be those rascals around that input false information like fake road conditions, etc.

**Server**
BikeQuest is web based system and thus needs a server online. BikeQuest.com and bike-quest.com are already taken; we may need to come up with a new name.

**Map Data**
RIGIS was brought up as a possibility for the source of map data. But depending on how the map data is given to us, we will parse this information differently. The whole coordinates structure will be built around the map data we're given. This is a fairly large dependency.

## Task Breakdown

**Web Interface Team [1 person(s)]**
- Create all web pages in .php and .html format
- Install forum software
- Create all graphics for site

- Create feedback forms and ratings for routes generated
- Create all community interaction scripts

**Database team [1 person(s)]**

- Create all the databases listed in the diagram above and add more if needed
- Make all SQL queries that are needed for all components that depend on the database
- Tweak the database system to be as efficient and robust as possible
- Must be proficient and experienced with database design
- Create sufficient documentation to the other teams about database structure

**Route Generator AI and Parser team [2 persons(s)]**

- Write this program in C++
- Create a route generator that outputs a final route
- Able to handle all user input preferences
- Able to pick route depending on route feedback
- Take road conditions into account
- Parse data into the database
- Parse data to the Visual Map Generator

**Visual Map Generator Team [2 persons(s)]**

- Create a visual map generator that outputs a .jpg file
- Take in the route from the Route Generator
- Able to zoom and pan in map

**Administration Team [2 person(s)]**

- Create all documentations on how to use the system
- Check that everyone is on schedule
- Check in with each team and understand what is going on and write necessary documentations to other groups
- Team leader
- Create test cases
- Assigning duties to the team members

## Schedule

**Week 1:**

- Database design and setup
- Server setup
- Route search algorithm design
- Feedback system design
- Familiarize with map data format

**Week 2:**

- Start to work on web design
- Create Route generator system route to database parser
- Create route generator system route to visual map generator parser
- Fine tune databases
- Create community and feedback forms and web design
- Install community forums script
- Start to work on implementing route generator search algorithm

**Week 3:**

- Start to furnish web design
- Integrate web design with visual map generator image output
- Visual map generator to generate generic route on one zoom map image
- Further work on route generator to be affected by road conditions and user feedback

**Week 4:**

- Test web forms and correct insertion of users into user tables
- Test users to be able to find bookmarked routes
- Database structure and data insertion should work
- Data integrity check with test cases
- Route generator to generate all sorts of routes from developed test cases
- Map generator to be able to pan and zoom

**Week 5:**

- Fit everything together on Web Interface
- Final run-throughs to check everything is working out
- Start working on documentation and content for web interface
- Help files for users
- Special test cases
- Performance test cases
- Test if route generator routes are “intuitive”
- Test if route generator becomes smarter with more positive/negative feedbacks and sudden road condition changes

**Week 6:**

- Final testing of all functionality
- Work check on all documentations

## Assumptions

The largest assumption we have is map data. Our only current source id RIGIS, which we do not have any knowledge about that at all. This could be a big problem if we cannot parse and interact with that map data correctly.