Regis University

ePublications at Regis University

Regis University Student Publications (comprehensive collection)

Regis University Student Publications

Spring 2006

Velocal: A Web-Based Cycling Event Scheduler: A Web-Based Cycling Event Scheduler

Karl Olson
Regis University

Follow this and additional works at: https://epublications.regis.edu/theses

Part of the Computer Sciences Commons

Recommended Citation

Olson, Karl, "Velocal: A Web-Based Cycling Event Scheduler: A Web-Based Cycling Event Scheduler" (2006). *Regis University Student Publications (comprehensive collection)*. 754.
https://epublications.regis.edu/theses/754

This Thesis - Open Access is brought to you for free and open access by the Regis University Student Publications at ePublications at Regis University. It has been accepted for inclusion in Regis University Student Publications (comprehensive collection) by an authorized administrator of ePublications at Regis University. For more information, please contact epublications@regis.edu.

Regis University
School for Professional Studies Graduate Programs
Final Project/Thesis

Disclaimer

Use of the materials available in the Regis University Thesis Collection ("Collection") is limited and restricted to those users who agree to comply with the following terms of use. Regis University reserves the right to deny access to the Collection to any person who violates these terms of use or who seeks to or does alter, avoid or supersede the functional conditions, restrictions and limitations of the Collection.

The site may be used only for lawful purposes. The user is solely responsible for knowing and adhering to any and all applicable laws, rules, and regulations relating or pertaining to use of the Collection.

All content in this Collection is owned by and subject to the exclusive control of Regis University and the authors of the materials. It is available only for research purposes and may not be used in violation of copyright laws or for unlawful purposes. The materials may not be downloaded in whole or in part without permission of the copyright holder or as otherwise authorized in the "fair use" standards of the U.S. copyright laws and regulations.

REGIS UNIVERSITY

SCHOOL FOR PROFESSIONAL STUDIES

MASTER OF SCIENCE

IN

COMPUTER INFORMATION TECHNOLOGY

VeloCal

A Web-Based Cycling Event Scheduler

PROFESSIONAL PROJECT

Karl Olson

February 2006

Regis University
School for Professional Studies
MSCIT Program

Acknowledgements

Sincere appreciation goes to the following individuals and groups for their incredibly valuable contributions that made this project a success:

- My wife, Amy, for support, motivation, and for sacrificing so much to allow me to finish this enormous endeavor, despite the incredible challenges and joys of becoming the parents of two children during the life of this project

- Fellow students, Ann Brecht and Terry Clark, for moral support, application testing, and documentation review

- Friend and co-worker, Steve Douglas, for his ideas, vision, encouragement, testing efforts, and being my bridge to many members of the local cycling community

- Friend, Jay Kost, for sharing his fantastic graphic arts skills with his creation of the VeloCal logo

- The executive council of Cycling Friends of Fort Collins for sponsoring this project, and remaining flexible with an ever-changing schedule

Abstract

This project focused on the creation of a web-based ride scheduling application for Cycling Friends of Fort Collins (CFFC). The CFFC is a recently organized non-competitive bicycling club founded with the purpose of facilitating rides of a variety of levels and encouraging cyclists to ride together. The primary goal of this application is to allow any user to schedule a group ride and send email notification and reminders of the event to all interested members. Individuals can then optionally respond with their attendance intentions. Providing an easy interface to schedule events will encourage more group rides and help foster a friendly and active cycling community.

Keywords: database, cycling, web, application, email, event scheduling

Table of Contents

Abstract **5**

1.0 Introduction **8**
1.1 Problem Statement 8
1.2 Review of Existing Situation 8
1.3 Goals of Project 9
1.4 Barriers and/or Issues 11
1.5 Scope of Project 12

2.0 Review of Literature and Research **14**
2.1 Review of Existing Solutions 14
2.2 Research Methods Used 17
2.3 Observations 17
2.4 Contributions 18
2.5 Open Source 19
2.6 Summary 21

3.0 Project Methodology **22**
3.1 Research Methods Used 22
3.2 Software Development Life-Cycle Model Followed 23
3.2.1 Requirements Phase 24
3.2.2 Design Phase 24
3.2.3 Construction Phase 26
3.2.4 Verification Phase 27
3.2.5 Implementation Phase 28
3.2.6 Maintenance Phase 28
3.3 Short Review of Deliverables from Each Phase 29
3.4 Short Review of Milestones from Between the Phases 32
3.5 Outcomes 32
3.6 Project Methodology Summary 32

4.0 Project History **34**
4.1 How the Project Began 34
4.2 How the Project Was Managed 34
4.3 Was the Project Considered a Success? 34
4.4 What Changes Occurred to the Plan? 35
4.5 How Did the Project End? 36
4.6 What Went Right and What Went Wrong? 36
4.7 Project Variables and their Impact 37
4.8 Findings and Analysis Results 38
4.9 Project Summary 39

5.0 Lessons Learned **40**
5.1 What Was Learned from the Project Experience? 40

5.2 What Could Have Been Done Differently? 40
5.3 Did the Project Meet Initial Expectations? 41
5.4 Next Stage of Evolution for the Project if Continued 41
5.5 Conclusions / Recommendations 42
5.6 Summary 43

References 44

Appendix A: Requirements 46
Registration 46
User Profile 46
Security / Privacy 47
Navigation 47
Calendar 47
Groups 48
Support / Help 49
Events 49

Appendix B: Entity Relationship Diagram (ERD) 51

Appendix C: System Architecture Diagram 52

Appendix D: Page Navigation Map 53

Appendix E: Screenshots 54
Ride Calendar 54
New Event 55
Event Details 56
Member Profile 57
User Profile 58
Edit Profile 59
Join a Group 60
Create a Group 61
Group Profile 62
Group Preferences 63
Manage a Group 64
Manage Group Administrators 65
Inviting Users 66
Managing the Join Queue 67
Support / Feedback 68

Appendix F: The VeloCal Logo 69

1.0 Introduction

1.1 Problem Statement

Cycling Friends of Fort Collins (CFFC) is a recently organized non-competitive bicycling club founded with the purpose of facilitating rides of a variety of levels and encouraging cyclists to ride together. The club organizes one-time and recurring rides around the Fort Collins, Colorado area. To assist in the organization and announcement of these events, the club was in need of a web-based, database-driven application.

The primary goal of this application is to allow any user to schedule a group ride and send email notification and reminders to all interested members who can optionally respond with their attendance intentions. Providing an easy interface to schedule rides will encourage more group rides and help foster a friendly and active bicycling community.

1.2 Review of Existing Situation

A number of months ago, a popular web-based ride scheduler (clubroadie.com) was inexplicably shut down and attempts to contact the owners failed. The site was regularly used by a number of local riders, including members of the CFFC, so the absence of this site had a major impact on coordinating rides. Even when the site was available, however, the owners were unresponsive to suggestions for enhancements. The site has

recently been re-activated, but as a pay-for-use site where annual fees are paid by organizations, not individuals.

The CFFC was in need of a free, flexible application that could be customized to its unique requirements. For these reasons, the CFFC decided that creating its own application, and sharing it with other local cycling organizations, would address the inadequacies of existing solutions and allow for useful customizations. It was the hope of the CFFC that this scheduling application would become a welcome and useful contribution to the local cycling community.

1.3 Goals of Project

Early in the life of this project, there was intensive communication with the CFFC council regarding the general goals of the final application. Brainstorming sessions identified features not available with existing solutions that were considered critical; and ongoing discussions defined a specific, prioritized list of project requirements and deliverables. Following is a list of overall project goals:

Goal: To keep CFFC costs minimal by leveraging existing infrastructure to build and deploy the application. The CFFC had provided funding for domain name registration and had committed to pay these fees in the future. It also had already established a web presence at http://cyclingfriends.org. This infrastructure was selected as the best choice for the development and deployment of the application.

Goal: To define and implement the application requirements by working closely with the CFFC. General requirements were initially communicated informally. A specific list of features and behavior were later identified formally and ultimately approved.

Goal: To develop a web-based, database-driven application that offers the following general features (see Appendix A for a more complete list of application requirements):

- an easy, but secure, registration process
- the ability to view a calendar of upcoming rides
- the ability to click on a specific ride in the calendar for details
- ride details to include a start time, a start location, a ride pace, and a list of riders who have already registered their intention of attending
- the ability for registered users to enroll in a ride and receive ride announcements via email
- the ability for users to set their preferences regarding email notifications
- the ability for registered users to schedule a group ride
- the ability for ride leaders to send out ride announcements to other users who have specified their preference to receive announcements

Goal: To release the application to both the CFFC and the general public. The best way to foster a friendly cycling community and encourage more group rides is to make tools like this application free and easy to use.

Goal: To write well-documented and maintainable code. There is the possibility that future development and maintenance of the application will shift to other developers, so efforts were made in the development phase to ease such a transition by writing understandable code and providing plenty of helpful comments.

1.4 Barriers and/or Issues

As this project focused on the initial creation of the application, there were no requirements regarding current users, nor were there any data migration issues. The development environment simply became the production environment when the application was deemed complete. At that time, test data was removed from the database and users were able to begin inputting data.

This application was developed specifically for the CFFC; therefore, requirements were defined and final approval for implementation was given by the CFFC. The club meets on an *ad hoc* basis, so acquiring approval required that a special meeting be called. However, the long-term intention was to make VeloCal available to other cycling organizations.

In the interest of maintainability, supportability, security, interoperability, and zero-cost, all tools used to build the application and all software employed was open source. For example, the database selected was MySQL; the Mail Transport Agent (MTA), sendmail; the web server, Apache; and scripts were written in Perl. Open source tools and software were key elements in

facilitating the development of a standards-compliant interface and allowing users to access the application with any standards-compliant browser.

1.5 Scope of Project

To facilitate ride scheduling via a web-based interface and email notification about upcoming rides, a set of web-based scripts were created to interface with a database containing ride and user data. A UNIX-based system hosts a MySQL database, serves web pages with an Apache web server, and sends email messages via a sendmail MTA. The only requirement to interact with the application is a standards-compliant web browser. Perl scripts are executed via mod_perl and are used to generate dynamic pages, process user requests, and communicate with the underlying database through the DBI Perl module.

This project focuses on the release of the first version of this application. It offers a minimal set of features required for the initial release as defined by the CFFC. After this project's completion, application development will continue by addressing change requests and enhancement requests made by the CFFC and users.

The major steps of the project were to:

- define the application requirements
- develop an Entity Relationship Diagram (ERD) that models the data defined in the requirements

- develop a Page Navigation Map to identify user navigation and underlying application functionality
- create the database based on the above ERD
- develop the application in Perl based on the application requirements
- test the application and acquire approval from the CFFC
- resolve any issues discovered during the testing phase
- release the application for general use

2.0 Review of Literature and Research

2.1 Review of Existing Solutions

There already exists a number of generic, web-based event schedulers, the most popular and notable being Yahoo Groups (groups.yahoo.com) and Evite (evite.com). There is one specific to cycling, Club Roadie (clubroadie.com). These solutions have a number of distinct advantages for users. They are applications that are mature in their development, and they have been well tested and used by a large number of users. Cycling organizations might prefer these existing solutions because they are mature, supported, and well-tested. Users may appreciate using them because they may already have existing accounts and will not need to register for yet another account at another site, and they will already be familiar with the interface and the application behavior, so there is nothing new to learn.

However, these existing applications have a number of drawbacks in terms of the goals of this project. None of these event schedulers simplify the process of announcing cycling-specific events or foster a healthy cycling community. They are intended for much wider audiences and a wide array of events such as birthdays, holidays, or concerts. As such, they do not have cycling-specific event attributes such as ride mileage, terrain, or pace (Yahoo, Evite, ClubRoadie).

VeloCal has some unique features that set it apart from these existing solutions:

- ride leaders can anticipate the turn-out for a particular event because riders can indicate, through a link in their email notices, their intentions of attending the event as ‘Yes,’ ‘No,’ or ‘Maybe’
- users can optionally add comments to their attendance intentions
- each event has attributes that are specific to cycling such as the expected terrain or pace
- users can see a consolidated calendar view, indicating all events for all the user’s groups
- groups can be open to the public, allowing any user to join at any time; or restricted, requiring administrator approval for access; or private, requiring administrator approval and completely hiding the group from any non-members

Figure 1 provides a comparison of the features in VeloCal and the leading event scheduling applications indicating the distinct advantage VeloCal offers for cycling organizations and users (Yahoo, Evite, ClubRoadie):

Feature	Yahoo Groups	Evite	Club Roadie	VeloCal
Users can indicate their attendance intentions	Yes	Yes	Yes	Yes
Free for individuals and groups	Yes	Yes	No	Yes
Cycling-specific ride attributes	No	No	Yes	Yes
Consolidated calendar view	No	No	No	Yes
Multiple group administrators	No	No	No	Yes
Public, Restricted, and Private Groups	No	No	No	Yes

Figure 1: Comparison of Existing Solutions

Based on feedback gathered during the project requirements phase, the most important of the above features was the consolidated calendar view, something unavailable in all other existing solutions. A major goal of VeloCal was to allow an already logged-in user, when returning to the VeloCal home

page, to be immediately presented with a calendar view of all scheduled rides for all the user's subscribed groups, thereby minimizing interaction with the application and simplifying the entire event process.

2.2 Research Methods Used

During the investigation phase of the project, meetings were held with the council for Cycling Friends of Fort Collins (CFFC). Discussions focused on the existing options for communicating and managing rides. Consensus was that choices were either non-existent or too limiting in features to be useful. Brainstorming sessions provided details on a minimum feature set that would be required for initial usage. Additional ideas were added to a list of features that could be added at a later time and would be useful enhancements (see Appendix A).

In addition to meeting formally with the CFFC, there were a number of informal discussions with various ride leaders of other *ad hoc* groups. Their comments were consistent with the feedback and ideas captured during discussions with the CFFC council. This affirmed the soundness of the proposal and strengthened the decision to continue moving forward on the project.

2.3 Observations

To gain a better understanding of the application's requirements and typical use model, the email traffic of a few local cycling groups was monitored for a

number of months. Typically, the contents of the emails included the expected information for any event announcement: date, start time, start location, type of ride, and a solicitation for other people's interest. Based upon this observation, the data contained in the emails was considered a requirement for the new application to be able to handle.

To better understand the advantages and disadvantages of existing solutions, accounts were created for experimentation at Yahoo Groups, Evite, and Club Roadie. Events were created using each application to observe any challenges or frustrating interfaces to avoid in the new design. Similarly, attractive features or interfaces of these existing solutions were noted as best practices to adopt (Yahoo, Evite, ClubRoadie).

2.4 Contributions

The number of available cycling-specific event scheduling applications is extremely limited. VeloCal makes a significant contribution to this area by providing a free alternative and offers a number of features not available in other known applications. The features provided are based on extensive interviews, discussions, and observations with members of the local cycling community who are not satisfied with the solutions currently available. The application simplifies the process of scheduling or finding a ride, thereby providing a helpful and useful service that will contribute to the healthy growth of new and existing cycling communities.

2.5 Open Source

At the highest-level view, VeloCal is simply a Perl script that interfaces with a MySQL database. Perl and MySQL are both open source software tools that, like any open source software, offer a number of critical advantages to this project (Perl, MySQL):

- free
- open standards based for interoperability
- well-supported through a highly active open source community
- maintainable
- void of any licensing issues
- flexible (source code is available and free to use and/or modify)
- platform-independent

There is a wealth of information and help available on the Internet for open source tools. Web-based forums, newsgroups, help documents, tutorials, and articles provide the support needed for everything from databases to HTML, CSS, or JavaScript. Often, this support is faster and/or more complete than support from a commercial software vendor. Plus, commercial software often requires recurring support contracts that can be quite expensive.

Although other scripting languages such as Ruby or PHP would have been just as appropriate, Perl was chosen primarily because of familiarity and because it commands the largest development community, meaning better support and documentation and, most importantly, a wide number of available

modules (Ruby, PHP). Perl modules are well documented, easy to install, and widely adopted and tested (CPAN). By drawing upon the large pool of available modules and re-using code for common tasks, significant savings in terms of code development time and effort were achieved. For example, the CGI module provided functions to print HTML tags and always generated HTML-compliant code that would be correctly interpreted by all standards-compliant web browsers. Another example used in VeloCal was the Date::Calc module that performs date calculations, a complex process prone to human error. By using the module, the developer does not need to be concerned about the number of days in a given month, leap years, or the process of determining the day of the week for a particular date. By using the provided functions of Date::Calc, accurate values for complex date calculations are returned (Date::Calc).

Like Perl, the MySQL database was chosen for familiarity reasons. It is the most popular open source database and is the preferred database for applications built with Perl and Apache (MySQL Popularity). It scales well should application usage dramatically increase. Performance benchmarks have shown it to perform as well as any other available database offering, if not better (MySQL Performance). Thorough documentation is widely available as are technical forums (MySQL Documentation). It is simple to compile, install, configure and maintain. Backup solutions are flexible, easy, and scriptable (MySQL Backups).

Special mention should be made of the issue of platform independence with the open source software selected for this project's implementation. Because of hardware failures and replacement selection, the development platform dramatically shifted in the middle of development from a Linux x86 solution to a Macintosh OS/X solution. After a brief effort to compile all the required software on OS/X, development continued without a single progress blocker – a true testament to the efficiencies and power of open source software.

2.6 Summary

Based on extensive conversations with leaders in the local cycling community, requirements for a cycling-specific ride scheduler were identified and agreed upon. A survey of existing solutions was undertaken, and the strengths and weaknesses of each were identified. Using this data, an application was defined that would be unique and would benefit the general cycling community.

Before development began, a development environment was decided upon. An open source solution for scripting the dynamic generation of HTML-compliant web pages and an open source database for storing and retrieving user data were deemed the best solution. Using the most popular open source scripting language (Perl) and database (MySQL) offered superior support, maintainability, interoperability, and flexibility. Unexpected hardware changes late in the development cycle further validated that these choices offered ease of code and data portability across different hardware platforms.

3.0 Project Methodology

3.1 Research Methods Used

Distinct research was conducted for two main phases of this project: requirements gathering and code development.

While defining the parameters of the project and identifying the application requirements, research was focused on three methods: interviews with future users of the application, observing current practices of the future users, and analyzing the strengths and weaknesses of existing applications.

Interviews were conducted both formally and informally. Formal interviews were conducted at scheduled CFFC council meetings. Questions were asked about current methods of organizing rides, challenges with these current methods, and desires for future solutions. Discussions centered on desires to create an application that was simple to use, but was more powerful in certain areas than existing solutions. Council members, in the interest of making a contribution to the general cycling community, identified requirements that would likely be universal to all organizations, not just the CFFC.

Informal interviews consisted of unplanned discussions via email or in person with CFFC council members. Questions or potential ideas that were raised while formalizing the requirements were posed and answered via these informal discussions.

Observing the current practices of CFFC members involved monitoring the email traffic from group members and observing the behaviors of members using existing applications. Watching users employ these existing applications quickly revealed the advantages and disadvantages of each. Many of the core features of VeloCal were modeled after the successful elements of the existing applications; and many of the frustrations users experienced were eliminated and creative workarounds became new features of VeloCal.

Research conducted during the code development phase of the project was primarily technical research surrounding specific implementation details. An example was implementing the storage of images in the database as binary objects. The specifics of implementing this feature were unfamiliar, so further research was required. Answers were found in official MySQL documentation and online user forums where other developers had shared their success and had documented commonly accepted practices (MySQL BLOBs).

3.2 Software Development Life-Cycle Model Followed

The Software Development Life-Cycle (SDLC) model used for this project was the familiar and common “Waterfall Method.” The method was introduced in 1970 by the software engineering researcher W. W. Royce (Waterfall). In this model, development is modeled as flowing steadily downwards, like a waterfall, through the phases of requirements analysis, design, construction, verification, implementation, and maintenance (See Figure 1).

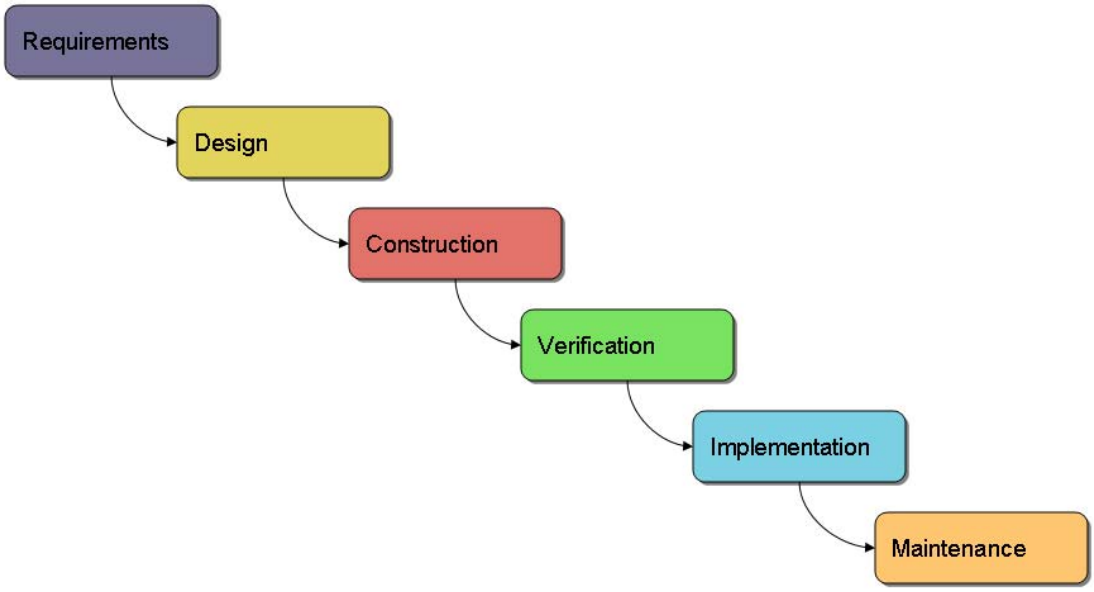


Figure 2: Waterfall Method

3.2.1 Requirements Phase

As described earlier, interviews, both formal and informal, were performed with CFFC members, and an analysis of existing applications was conducted. The outcome of these efforts was a list of requirements agreed upon by the CFFC council (see Appendix A).

3.2.2 Design Phase

The first step in the design phase was to model the data and create an Entity Relationship Diagram (ERD). The database template in Microsoft Visio was used to create the ER Diagram (see Appendix B). Once the data were accurately modeled, it was a simple matter to create the database tables using MySQL's Data Definition Language (DDL).

Based on prior experience with similar web-based tools, the decision was made to use a single Perl script for all dynamic HTML and web-based interaction with the database. The structure of the script is a large IF … ELSIF condition. Based on the action, or link, selected by the user, only the specific IF clause for that action is executed.

Cascading Style Sheet (CSS) data is stored in an external file, and referenced by the dynamic HTML. This allows clients to use cached versions of the file to reduce the size of each page download, effectively increasing application speed and reducing server bandwidth consumption (CSS).

For performance reasons, mod_perl was used. This optional module for the Apache web server embeds a Perl interpreter into the web server so that dynamic content produced by Perl scripts can be served in response to incoming requests. This eliminates the significant overhead of re-launching the Perl interpreter for each request (Young).

Much of the design of VeloCal centers on sending email notifications and reminders to users and group administrators; therefore, some design consideration was made for a Mail Transport Agent (MTA). Since one of the goals of this project was to minimize any infrastructure investment by using already available systems, the decision was made to use the sendmail MTA already configured on the development environment.

For sending out reminder email notices, a separate Perl daemon was designed. The script is small in scope and uses a single SQL query to identify

users that need to receive email reminders. Email notices are sent, the database is updated, and the daemon sleeps for a customizable period of time.

3.2.3 Construction Phase

The first step in the construction phase was to create the tables in MySQL that were modeled in the ERD. This was a simple process that only required executing the valid MySQL DDL identified earlier.

Development of the Perl script took place using the vi editor. All significant versions were checked into RCS, a source control management tool commonly found on all UNIX-based operating systems (RCS).

Unit testing was frequently performed during development through web browser interaction with the script. Errors and debug messages were monitored in the Apache error log file. Since many major functional pieces were missing at different stages of development, test data was frequently entered into the database manually to verify correct behavior of the code being developed.

After development of the main script was completed, the reminder daemon was constructed. This was also written in Perl and took minimal effort compared to the size and scope of the main script. The most significant challenge in this step was composing the SQL query needed to identify all

users waiting for a reminder notice at a user-defined period of time prior to the event.

3.2.4 Verification Phase

Testing of the application was performed in three different ways. There was modular testing performed during code development, partner testing at major milestones, and general beta testing at the end of development.

Whenever a significant number of features were implemented, a milestone was declared and communicated to test partners. Testers were given two weeks to review the new version. Email communication to the testers included a list of all new features, recommendations for their testing focus, and a date when testing should be completed.

After code development was complete, a beta testing phase was undertaken with a small, local group of cyclists that used the application to organize their weekly rides. The beta test phase lasted through the month of January 2006.

Throughout testing, efforts were made to verify compatibility with a wide variety of web browsers, including Firefox, Safari, Opera, and Internet Explorer. Compatibility concerns were minimal because of the limited complexity and amount of JavaScript used and because of the HTML-compliant code automatically generated by the Perl CGI module. The biggest challenge was modifying CSS code to render correctly in different versions of Internet Explorer (CSS support in this browser is notoriously troublesome).

VeloCal is a multi-user, email-focused application, so prior to beta release, use with multiple email accounts was tested to assure success by beta test users. For example, events created under one email account were verified to send notification to the other email accounts. The use of multiple email accounts was also critical to testing the process of joining restricted and private groups.

3.2.5 Implementation Phase

The VeloCal development environment became the production environment, so no special effort was required to roll out the application at the implementation phase. A domain name (velocal.org) was registered and the Apache configuration file was updated to support this new domain via the virtual host feature of Apache (Apache). After a meeting with CFFC council members, the decision was made to announce the availability of the tool to all interested parties.

3.2.6 Maintenance Phase

The maintenance phase has been focused on responding to minor issues discovered by users and responding to questions and enhancement requests from users submitted via the ‘Support’ page. A backup strategy was developed, implemented, and monitored. There have also been efforts to update documentation and the list of Frequently Asked Questions on the ‘Support’ page.

3.3 Short Review of Deliverables from Each Phase

Following is a list of each major project phase including a brief description of the deliverables generated for each phase.

Requirements Phase: This phase generated a list of application requirements. Refer to Appendix A for the complete list of requirements as approved by the CFFC.

Design Phase: The Entity Relationship Diagram (ERD), the System Architectural Diagram, and the Page Navigation Map were the three deliverables from this phase of the project. Refer to Appendix B for the ERD, Appendix C for the System Architectural Diagram, and Appendix D for the Page Navigation Map.

Construction Phase: The first deliverable from this phase was the creation of the tables in the MySQL database. Next, the main Perl script was created. Finally, the event reminder daemon was written.

Figure 3 is a version history for the main Perl script, indicating the version number, the date the version was released and announced, and the major features delivered in the version:

Version	Release	Major features Implemented
Version 0.1	01/17/05	User registration. Login and logout.
Version 0.2	01/31/05	Profile edit. Photo edit. Change password. Change email address.
Version 0.3	02/05/05	Create, edit, and delete groups. Subscribe and unsubscribe from groups.
Version 0.4	11/15/05	Calendar view with scheduled events. Create events.
Version 0.5	12/02/05	Email notifications about events. Edit and delete existing events. Attendance intention tracking.
Version 0.6	12/06/05	CSS layout and graphics.
Version 0.7	12/09/05	General content cleanup. Reminder daemon.
Version 0.8	01/04/06	Minor bug fixes based on testing results.

Figure 3: Construction Phase Releases

Verification Phase: Although the only deliverables from this phase were minor script updates based on errors discovered during testing, this phase was one of the most critical phases of the project. Various user scenarios were simulated using a variety of web browsers and a number of email accounts. Test groups were added, modified, and deleted; and correct behavior was verified with each scenario. The overall design proved valid, and only minor fixes were required.

Implementation Phase: The development environment became the production environment at the implementation phase, so no major efforts were required. A new domain name (velocal.org) was registered, and changes to the Apache web server configuration file were made to support this new virtual host. Announcements were sent to two local cycling organizations that now have begun to use the application as their primary method of scheduling events.

Maintenance Phase: The maintenance phase was primarily focused on responding to user feedback and fixing minor bugs. A backup solution was already in place on the shared hardware, so no extra work was required to execute and verify a successful backup strategy. Backups are surprisingly simple and scriptable with MySQL (MySQL Backups). A single command, executed via cron, regularly backs up the entire database to an external disk enclosure (Cron).

3.4 Short Review of Milestones from Between the Phases

The employment of the waterfall method for this project meant that each phase was completed 100% before moving on to the next phase. Each completed phase built the foundation for the next phase in the project and required that each phase was finished, accurate, and sound. Pausing between phases, reflecting on the work completed, and verifying completeness ensured that moving forward was appropriate.

3.5 Outcomes

The waterfall model was an excellent choice for this project. Although it involves a significant upfront investment, the model requires that the design and requirements are of high quality. Because no requirements were changed during the short lifetime of the project, constructing the application was a straightforward process of implementing each static requirement. Getting the design right at the beginning before writing code saved significant time and effort in the later phases of the project. Although the model is often criticized for its rather simple approach to development, it was highly appropriate for a project of this scope; and the milestones were clear and easy to recognize.

3.6 Project Methodology Summary

Development of this application followed the traditional waterfall method. Research and analysis was first conducted via interviews with potential users about existing applications. Based on the list of requirements gathered from

the research and analysis phase, a design was created. The solution was decided to be a Perl script used to generate dynamic web pages served through an Apache web server. Data is stored and retrieved in a MySQL database. A small daemon application was written to send out reminder notices to riders.

Development began with unit testing occurring throughout development. Major milestones were declared and communicated to testers who then focused testing efforts on the new features. After the completion of the code development, final testing was performed including a major beta testing effort with two local cycling groups.

After successful completion of the testing phase, the application was made available to the public. During a short maintenance phase, questions or support issues from users were addressed with communication or minor code fixes. Future enhancement requests are being logged for future version releases.

4.0 Project History

4.1 How the Project Began

The idea for this project began with informal discussions with members of the then newly formed cycling club, Cycling Friends of Fort Collins. Many members had been using an online ride scheduler, Club Roadie, but were frustrated by some of its limitations. When the site went offline without explanation and then returned as a fee-based service, many members were determined to create their own solution. Coincidently, the desire to create this new application came at the same time an appropriate final project for Regis needed to be selected. The decision to work on this project was quick and easy.

4.2 How the Project Was Managed

Although there was frequent consultation with the CFFC council for project requirements, there were no other individuals responsible for development efforts. Frequent status updates were given to council members. At various project milestones, windows of time for testing were announced. A few select volunteers assisted with testing the application.

4.3 Was the Project Considered a Success?

The list of desired features for the application was large and continued to grow, as with any software project. All attempts were made early in the project to identify requirements that were “musts” versus “wants.” During the

project's development, some requirements on the "musts" list were re-prioritized based on available time. The CFFC council was extremely willing to move some items to future releases, and the code was developed with these future enhancements in mind. Adding them later should not require any significant design changes. As the application has begun to be used, response has been extremely positive. Without question, the project has fulfilled its goals, and excitement surrounds future plans for enhancements.

4.4 What Changes Occurred to the Plan?

There were a few major features that were moved to future releases during the project's development. Being aware of these future changes while developing the code allowed for "code stubs" to be put in place, allowing these features to be added easily later.

For example, originally there was a requirement to allow each group to maintain a "ride library" of their favorite rides. Each library item would have a name, description, typical mileage, notes, and multiple images (such as pictures from the route or maps). This is a significant feature, but it was not critical to the first release of VeloCal. All of the technical challenges surrounding implementing these features have already been solved in other areas of the application. For example, the problem of storing and retrieving images in the database was solved in the code for allowing users to add photos to their user profiles. This code was also modularized, so adding images to a route will be as simple as calling an existing function.

There was a significant delay in the overall schedule, with the first release occurring almost a year behind schedule. An increased employment workload and dramatic changes at home (adjusting from 0 to 2 kids in 18 months' time) resulted in much less available time to make progress on the project. Thankfully, the CFFC council was very forgiving of the delays.

4.5 How Did the Project End?

Final changes to the code were made in mid-December 2005. At that time, an announcement was made to testing volunteers, and testing efforts commenced. A few minor changes were required based on feedback from testing. On January 2nd, 2006, the final testing phase was completed and final code changes were completed. A formal announcement was sent to the CFFC, informing them that the application was available for general use. In early January, the first group began using VeloCal for their weekly rides.

4.6 What Went Right and What Went Wrong?

There were many things that went right with this project. Frequent communication with the CFFC council was critical in maintaining a friendly, constructive relationship. Setting expectations about available time and realistic deliverables was also important. Flexibility from the CFFC was demonstrated with their understanding of schedule delays and willingness to move larger features into future releases.

Frequent testing at various milestones was helpful by allowing users to see early prototypes of application interfaces and to make decisions about the direction of development.

As with virtually all software projects, this application was aggressive in its list of features to be delivered in the first release. Efforts were made to try to balance a list of features that would make the application useful with estimates of the time required to develop them. Ultimately, however, this feature list had to be scaled back. While not necessarily a failure, it meant not delivering 100% of what was originally expected by the CFFC.

4.7 Project Variables and their Impact

By far, the most critical variable throughout this project was available time. There were no other developers, which meant no challenges in coordinating development efforts; but as time became limited during the development phase, it meant that little progress was being made.

A variable with positive impact was the interaction with the CFFC council. Their demands were minimal, and their flexibility with the schedule and changing requirements was appreciated.

Volunteer testers were an important element in validating the project's direction and code correctness. As their time was limited too, attempts were made to allow a reasonable amount of time for their testing efforts, but waiting

for their feedback delayed further development progress. Typically, two weeks were allowed for each review.

An unexpected variable that had a significant negative effect on the schedule was two major hardware failures. First, the Linux x86 system that was being used for development experienced a motherboard failure. Since long-term plans were to replace this system with a Macintosh system, a Mac Mini was put in its place and development was migrated over to this new system. This required compiling all the necessary infrastructure while simultaneously learning some of the subtleties of OS/X.

The second failure was a catastrophic hard drive failure on the new Mac system. All data on the old drive was lost and a new drive was installed. Although the VeloCal code had been backed up, the supporting applications had to be re-compiled; but executing these steps the second time was much quicker.

4.8 Findings and Analysis Results

The goal of this project was to create a web-based ride scheduler, under the sponsorship of the CFFC, for use by the general cycling community. Using existing infrastructure and building upon existing experience with web-based technologies and databases, an application was developed using Perl and MySQL to fulfill these objectives. This unique application was developed in a series of phases, using standard project management processes, while following the waterfall method of the System Development Lifecycle.

4.9 Project Summary

There were a number of challenges to this project. There were major delays in the project schedule, there were unexpected hardware challenges, and a number of expected features of the application were moved out to future releases. Despite these challenges, development progressed, and the CFFC continued their enthusiastic support and showed great flexibility. The application was released to the public in its first version in January 2006 and is already in use. Feedback has been extremely positive, and there is general excitement about future releases. Development continues and another version release is being planned for Summer 2006.

5.0 Lessons Learned

5.1 What Was Learned from the Project Experience?

Frequent communication with the CFFC was key to this project's success. Also, being careful to capture all requirements was important. The users of other solutions showed themselves to be easily frustrated, so being sure to incorporate the successful features of other applications and also successfully integrate the new features would be key to making this project a success.

It is important to have long-term project plans, and have a good understanding of what future application versions will look like, but make the early versions small in scope and able to easily accommodate future enhancements.

5.2 What Could Have Been Done Differently?

Considering the amount of coding effort involved with an application of this size and complexity (the application contains over 6000 lines of Perl code), it would have been reasonable to consider adding other developers to the project. Unfortunately, doing so would have added project overhead in terms of communication and coordination. However, additional developers may have sped development efforts because of the many restrictions on time available.

Also, requirements were defined by only one cycling organization, yet a major goal is to make the application available to any group. Therefore, it may have

been a good idea to solicit the feedback and review of other organizations. Feedback and use over time will hopefully confirm that what the CFFC considered a generic set of requirements will be applicable and useful to other organizations.

5.3 Did the Project Meet Initial Expectations?

According to the feedback received by users of the first release, this project has met and even exceeded expectations. The application has a number of small, unplanned features that were simple to implement, and were unexpected and welcomed by the users. Feedback has been that the application is surprisingly simple and easy to use.

5.4 Next Stage of Evolution for the Project if Continued

The next stage of this project is another major version release which adds features that were removed from the initial list of requirements. Also considered will be any enhancement requests from current users.

More mature cycling organizations may be interested in enhancing reports about past rides. For example, reports on the number of rides, number of riders per event, total miles ridden, number of cancelled rides, etc. To facilitate this type of reporting, some additional data will need to be collected for each event.

There are plans already in place for the next version to allow each group to maintain a ride library. Each ride in the library will have a name, description,

notes, approximate distance, and multiple images such as photos from the event or maps of the route.

To address privacy concerns, users will be provided an option in their profile to hide their email address from other users.

A current limitation with the process of creating events is that they can only be one-time events. Future versions will allow users to create recurring events.

Although users will typically create events according to their local time zone, the event reminder daemon will need to be modified in future versions to be aware of a user's default time zone and send out notices at the correct time.

To encourage wider adoption of the application, communication needs to be spread to potential users. A number of local club members have volunteered to post announcements and encourage others to use the application.

5.5 Conclusions / Recommendations

This project was rewarding in a number of ways. Many of the techniques and technologies employed were new and will be valuable in other similar, but unrelated, projects. Users have expressed their excitement and have begun actively using the application, so seeing users adopt it and provide positive feedback has been encouraging. Finally, single-handedly managing a project of this scope and complexity, despite its many challenges, was educational and incredibly rewarding.

5.6 Summary

VeloCal is a relational database-driven, web-based ride scheduler for fans of cycling. Its purpose is to offer a solution currently not available in any other form, encourage group rides, and foster a healthy cycling community. This application was developed over approximately one year and is already being used by two local cycling organizations. Feedback has been positive, and enhancements are already planned for the next version. This project was completed as a partial fulfillment of the requirements for Master of Science in Computer Information Technology, School for Professional Studies for Regis University in Denver, Colorado.

References

Apache Virtual Hosts. 28 February 2006. <http://httpd.apache.org/docs/1.3/vhosts/>.

ClubRoadie. 28 February 2006. <http://www.clubroadie.com>.

CPAN. 28 February 2006. <http://www.cpan.org>.

Cron. 28 February 2006. <http://en.wikipedia.org/wiki/Cron>.

CSS. 28 February 2006 <http://www.w3.org/TR/CSS21/>.

Date::Calc. 28 February 2006. <http://search.cpan.org/~stbey/Date-Calc-5.4/>.

Evite. 28 February 2006. <http://www.evite.com>.

MySQL. 28 February 2006. <http://www.mysql.com>.

MySQL Backups. 28 February 2006. <http://dev.mysql.com/doc/refman/5.0/en/backup.html>.

MySQL BLOBs. 28 February 2006. <http://dev.mysql.com/doc/refman/5.0/en/blob.html>.

MySQL Documentation. 28 February 2006. <http://dev.mysql.com/doc/>.

MySQL Performance. 28 February 2006. <http://www.mysql.com/why-mysql/benchmarks/>.

MySQL Popularity. 28 February 2006. <http://en.wikipedia.org/wiki/Mysql>.

PHP. 28 February 2006. <http://www.php.net/>.

RCS. 28 February 2006 <http://www.gnu.org/software/rcs/rcs.html>.

Ruby. 28 February 2006. <http://en.wikipedia.org/wiki/Ruby_programming_language>.

Waterfall model. 28 February 2006. <http://en.wikipedia.org/wiki/Waterfall_model>.

Yahoo Groups. 28 February 2006. <http://groups.yahoo.com>.

Young, Geoffrey, et.al. mod_perl Developer's Cookbook. Sams Publishing, 2002.

Appendices

Appendix A: Requirements

Registration

- Registration requires a valid email address.
- Registration should follow a two-step process to verify the user receives email at the address specified.
- Users should be able to request that their password be emailed to them.
- Cookies will be used to keep users logged in and authenticate access.
- Users will be able to un-register from the application, removing all traces of their name and email address.
- Users will be able to logout via a link displayed on every page.
- Users must be logged in to view their calendar and perform group functions.

User Profile

- Users will be able to modify their profile.
- Users will be able to add an optional photograph to their profile.
- Users will be able to change their password.
- Users will be able to change their email address.
- Users will be able to specify a biography, preferred pace, and preferred terrain.

Security / Privacy

- Users should not be able to view events for groups to which they are not registered.
- Users should not be able to gain access to private groups without the proper authorization.
- Users will not be able to view a list of group members without being a member of the group.
- All users will have their email address displayed for other group members.

Navigation

- Every page will display a navigation bar that includes links to the major areas of the application.
- Every page will display a user's custom navigation area, listing all of the user's groups and any appropriate links for each group.

Calendar

- A logged-in user should be presented with the user's custom ride calendar on the main page.
- The ride calendar should display all events for all the user's groups.
- Each event in the calendar will display the group name, event name, time, and, if already specified, attendance intention.

- Users will be able to add an event by clicking on an ‘Add’ link displayed in each day.
- Navigation will be provided to navigate to the next or previous month based on the current month displayed.
- Clicking on the “Ride Calendar” navigation link will take the user to the ride calendar for the current month.
- The current day of the month will be displayed uniquely from other days.

Groups

- Any user can create any number of groups.
- Users can join any number of groups.
- Groups must be designated private, restricted, or public.
- Public groups will allow any user to join at any time without any other user’s intervention.
- Restricted groups require a user to request authorization to join.
- Administrators can approve or deny restricted group access requests.
- Private groups can only be joined by invitation.
- Private group invitations will contain a unique key required for each invitee to join the group.
- Private groups are completely hidden from non-members.
- Users will be able to browse lists of restricted and public groups by state and city.

- Administrators will be able to change group types at any time.
- Groups will have a default pace, default terrain, default start time, and default start location.
- Group members will be able to view a list of all members of the group.
- Groups can be deleted.
- Only group administrators can delete groups.
- When groups are deleted, all group events are also deleted; and all group members are notified.
- Group administrators can individually choose to receive notification when a user joins or leaves a group.

Support / Help

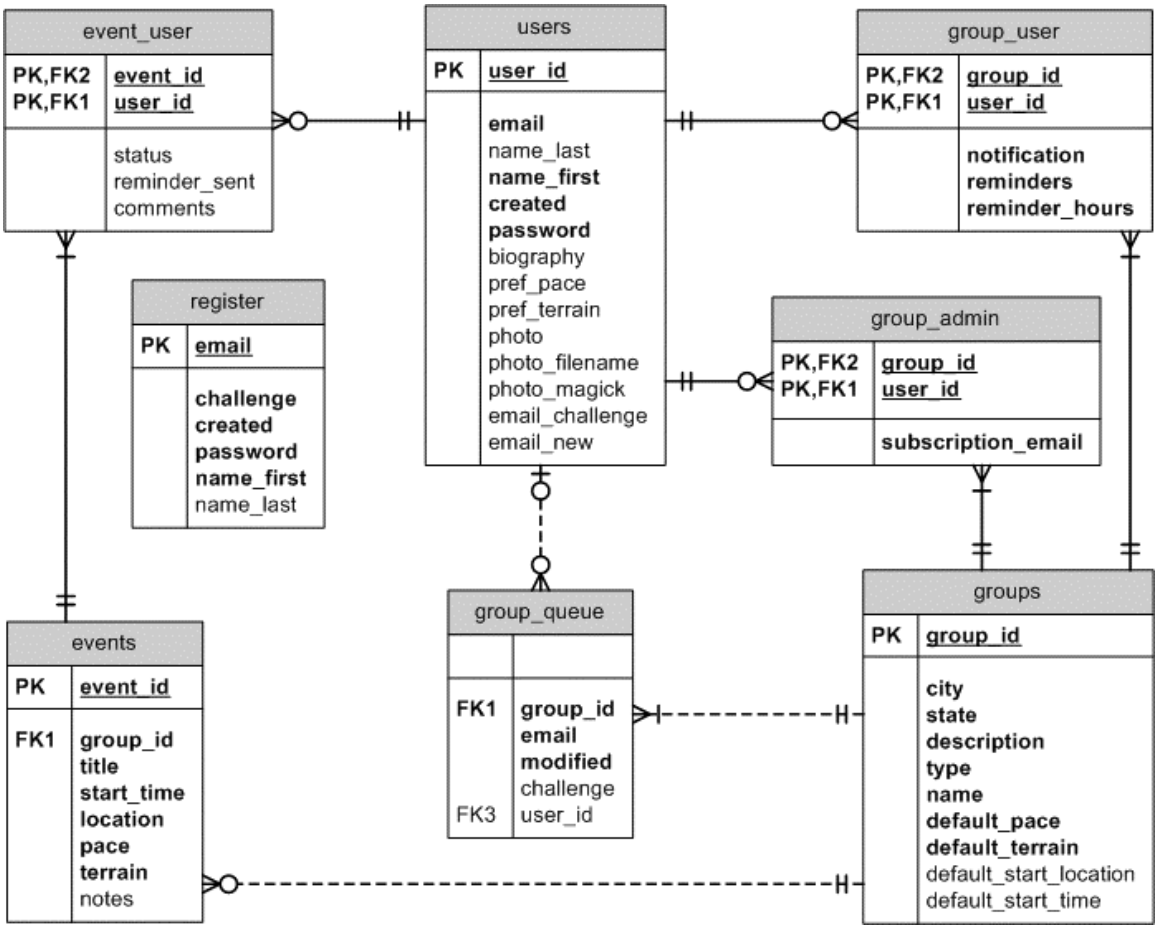
- A support link will be provided on all pages.
- Users will be able to submit feedback or questions through a form.
- A list of Frequently Asked Questions (FAQ) will be available on the support page.

Events

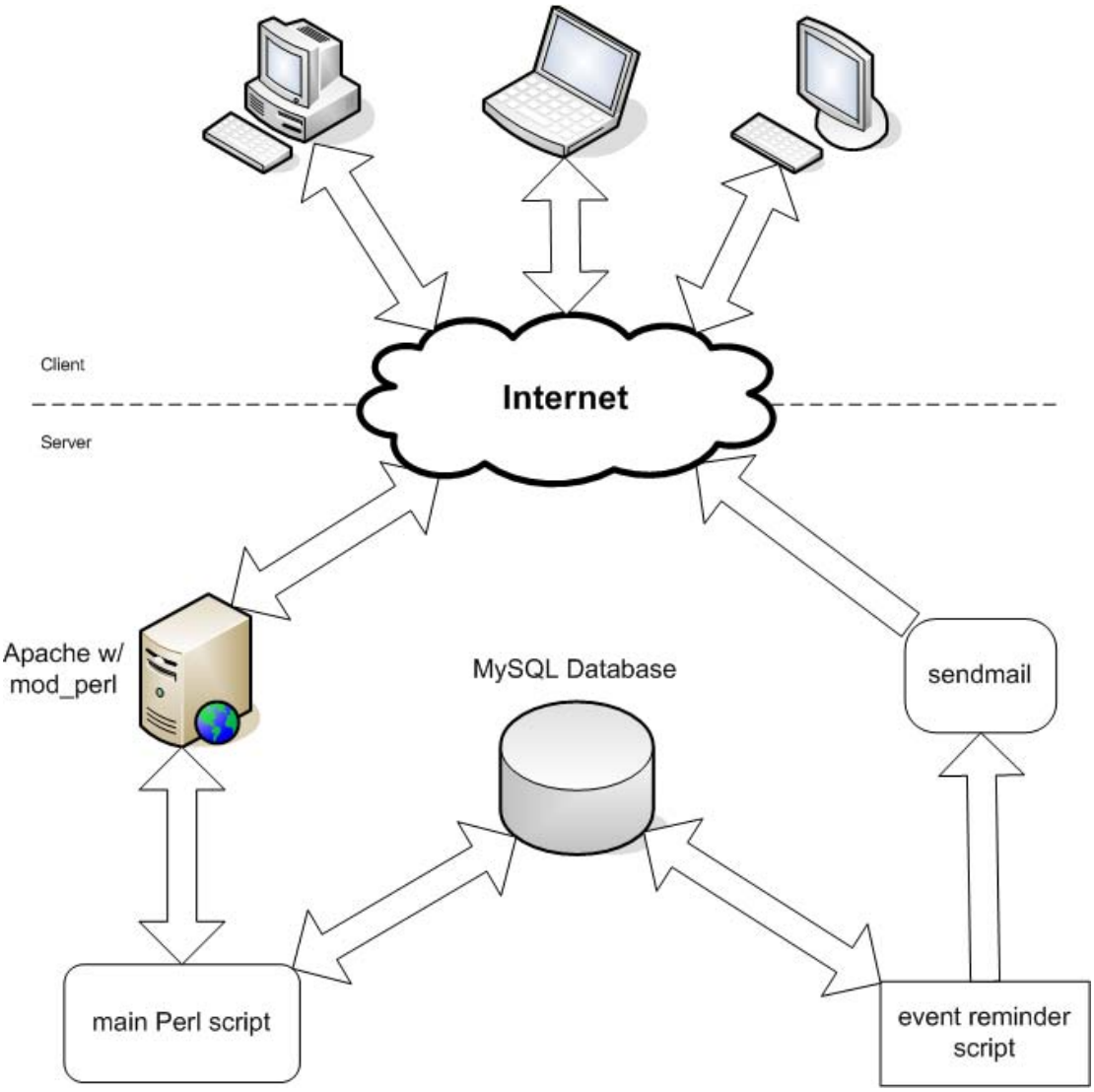
- Any member of a group will be able to create an event for that group.
- When creating a new event for a group, the group's default values will be automatically entered into the event creation form.
- Attendance intentions of group members will be recorded and displayed.

- Email reminders will be sent to users requesting reminders, and at a time in advance as specified by each user.
- Event creation or modification notices will be sent to all interested members.
- Events can be cancelled and notices will be sent to all interested members.

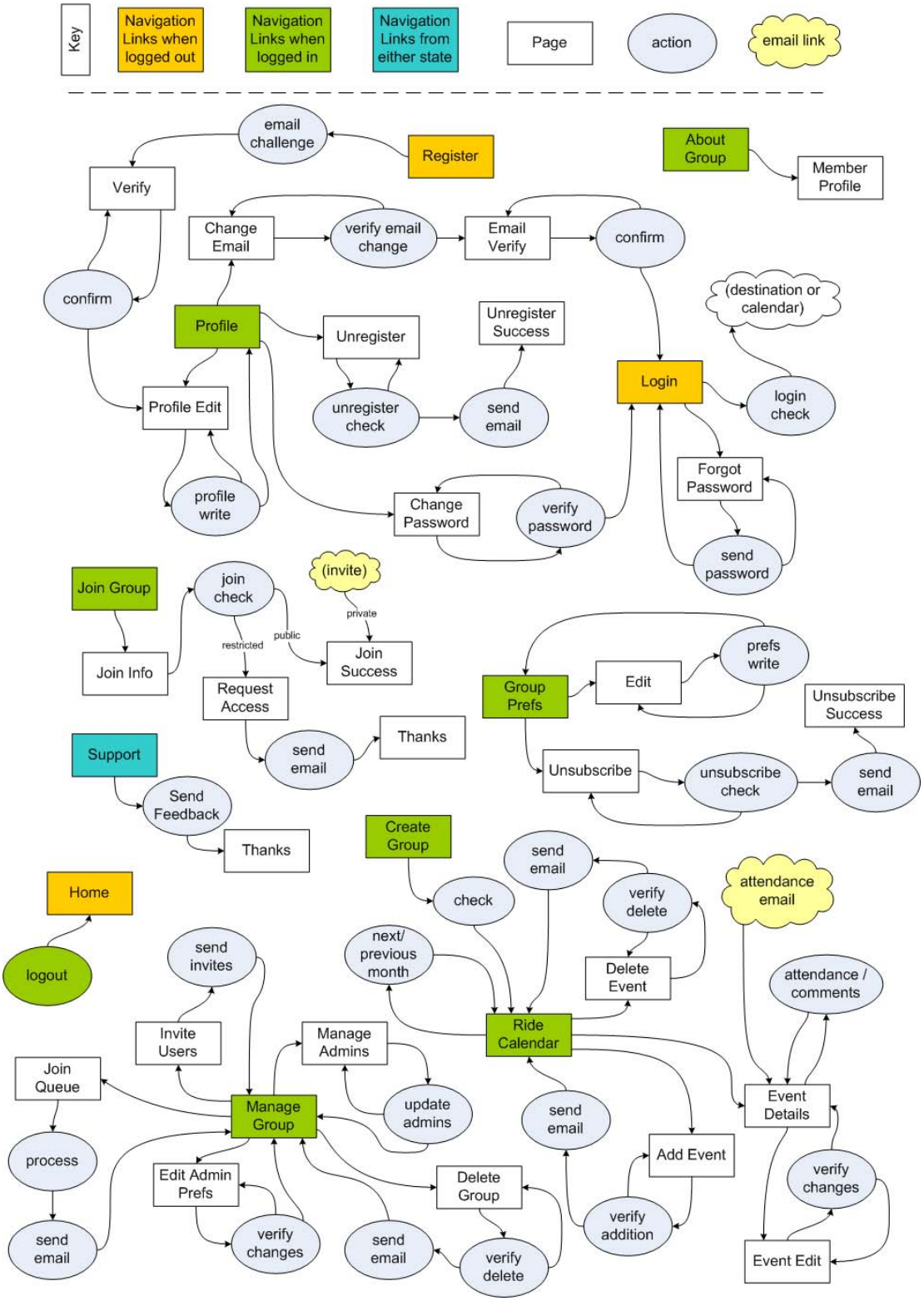
Appendix B: Entity Relationship Diagram (ERD)



Appendix C: System Architecture Diagram

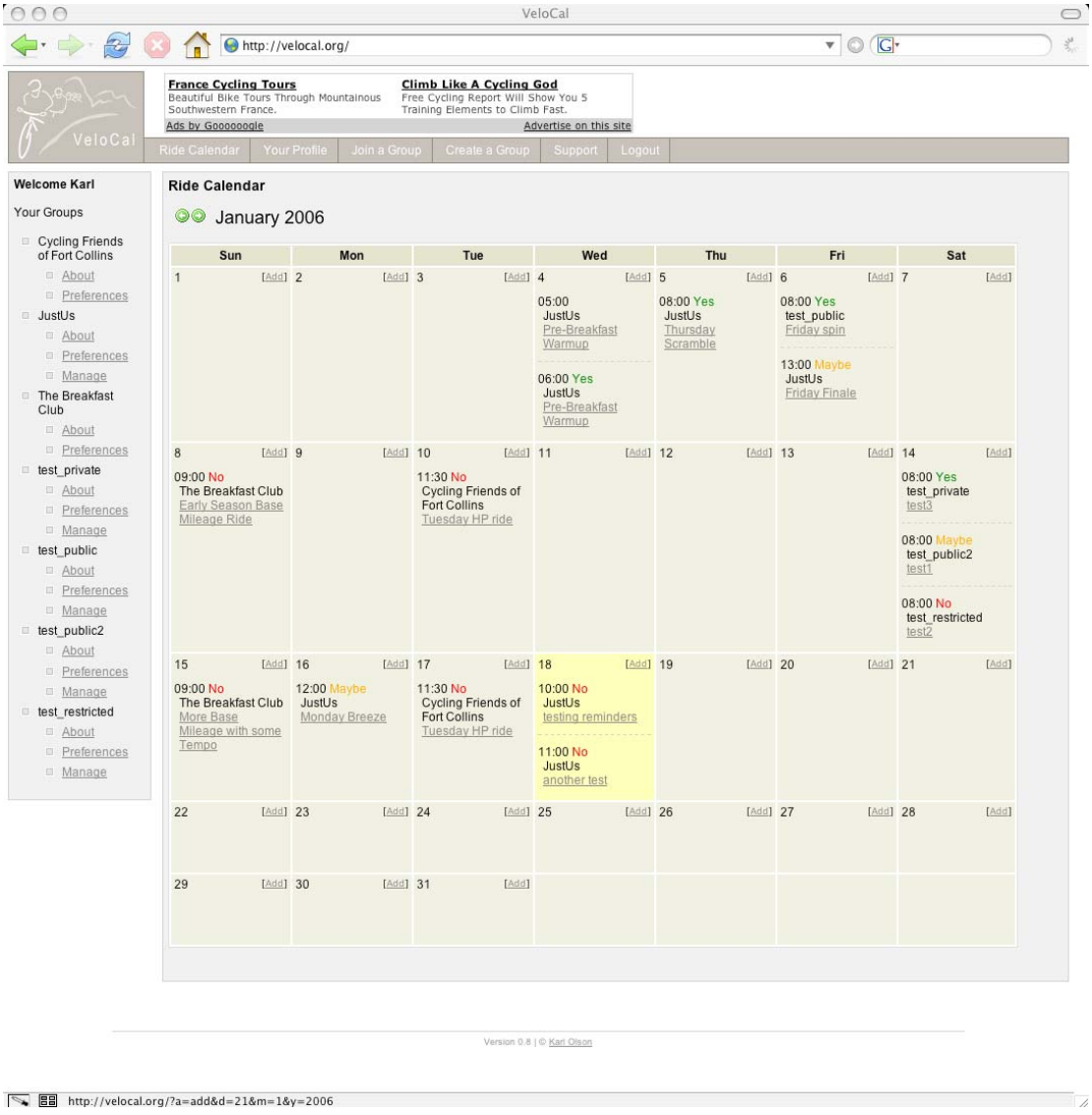


Appendix D: Page Navigation Map



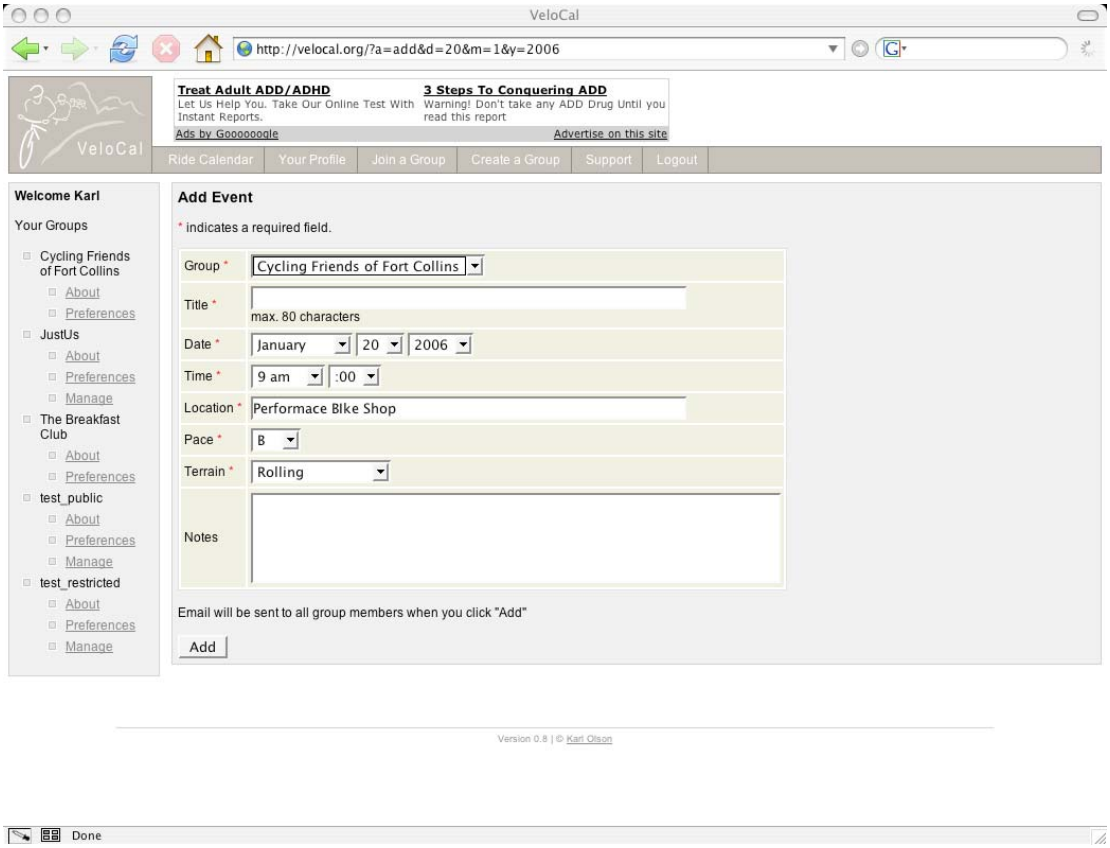
Appendix E: Screenshots

Ride Calendar



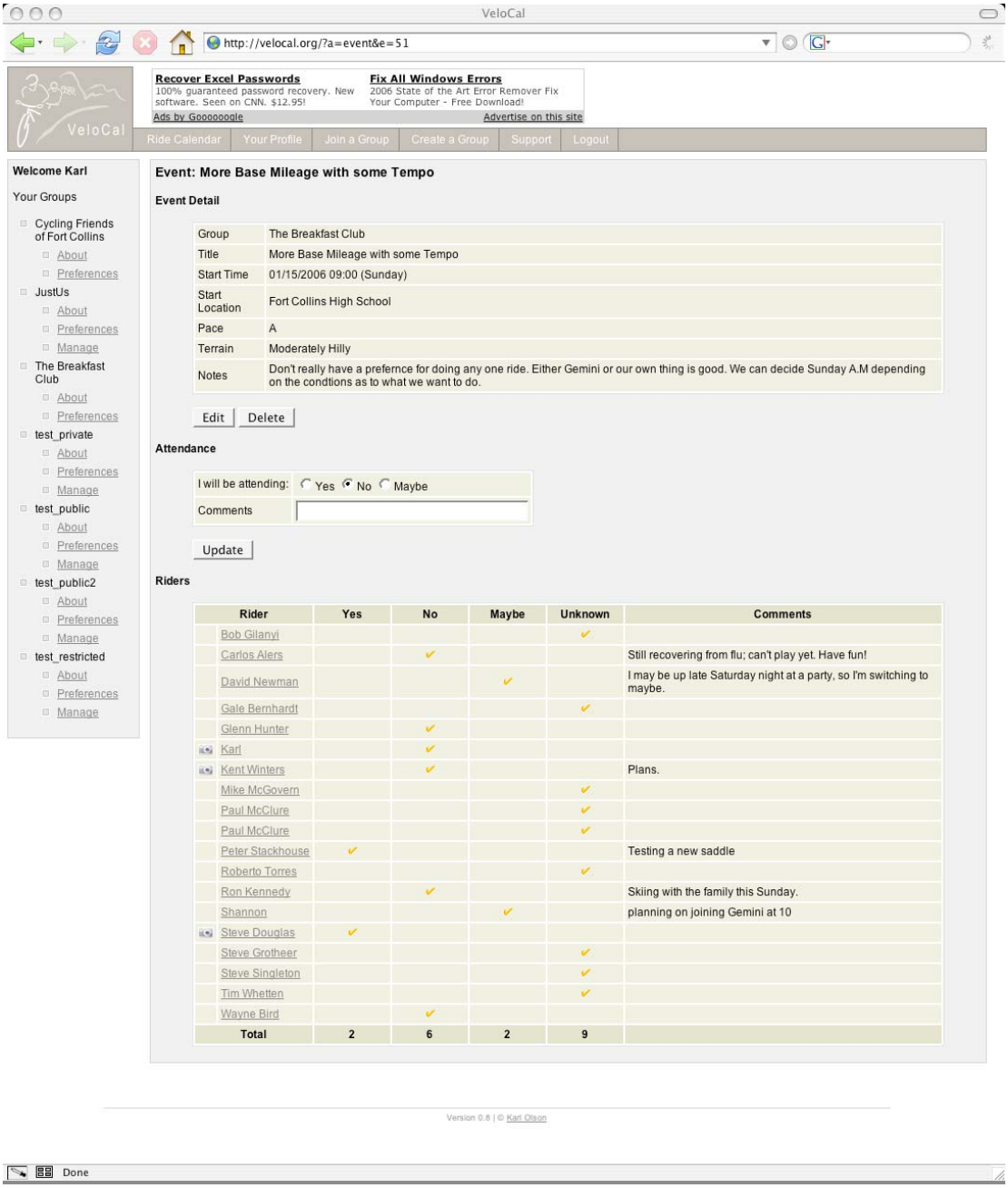
A user's personalized ride calendar showing events from multiple groups, and any optional attendance intentions

New Event



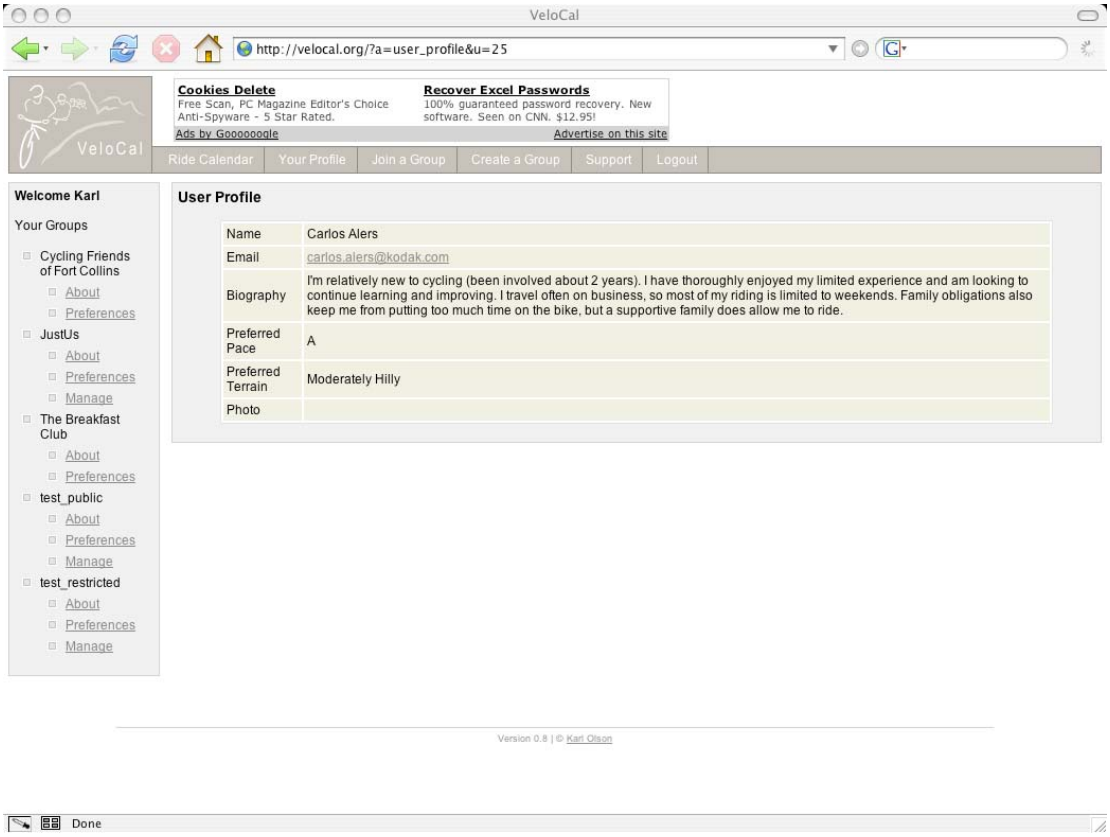
Adding a new event

Event Details



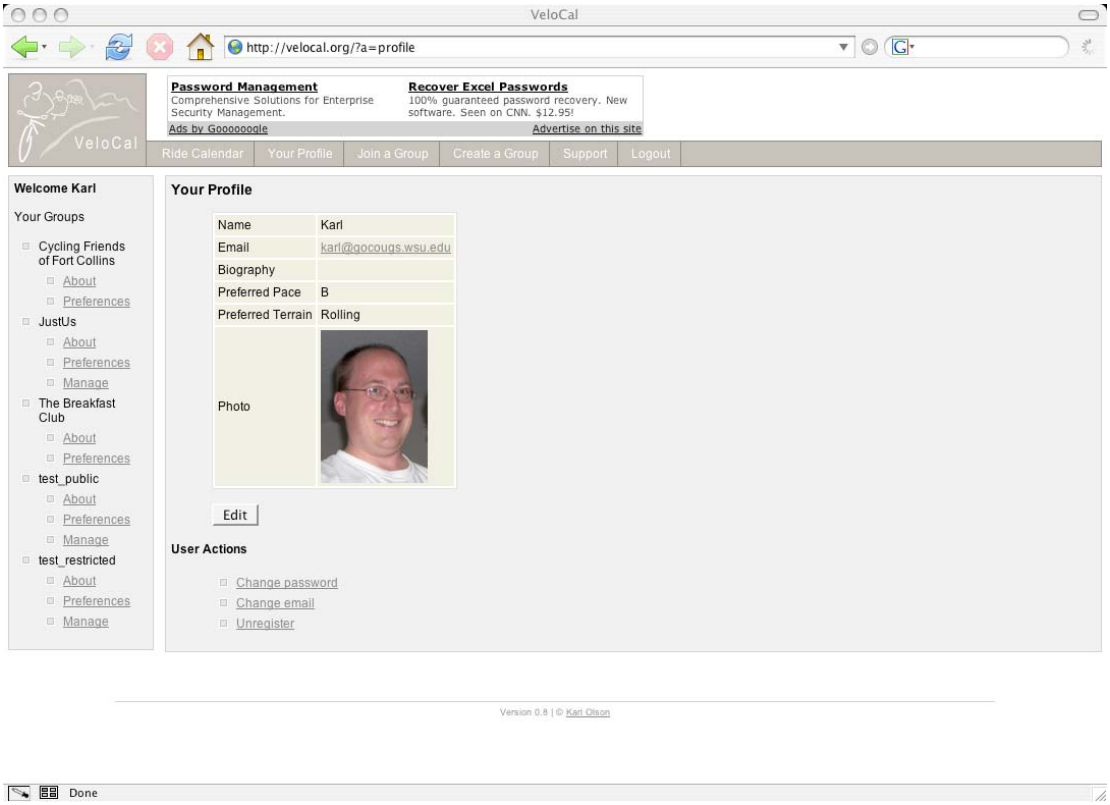
Viewing the details of an existing event, including attendance intentions of group members

Member Profile



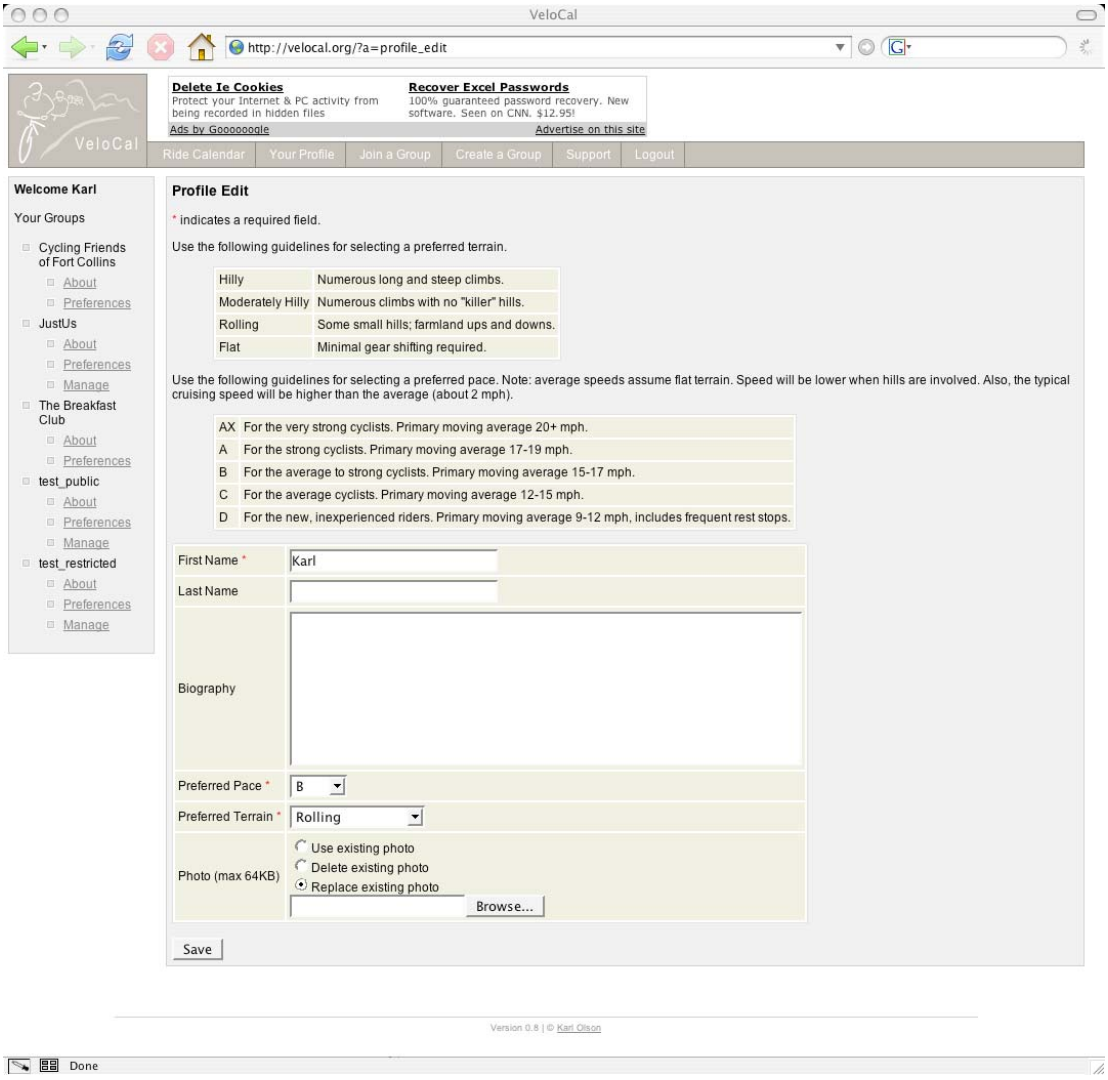
Viewing another member's profile

User Profile



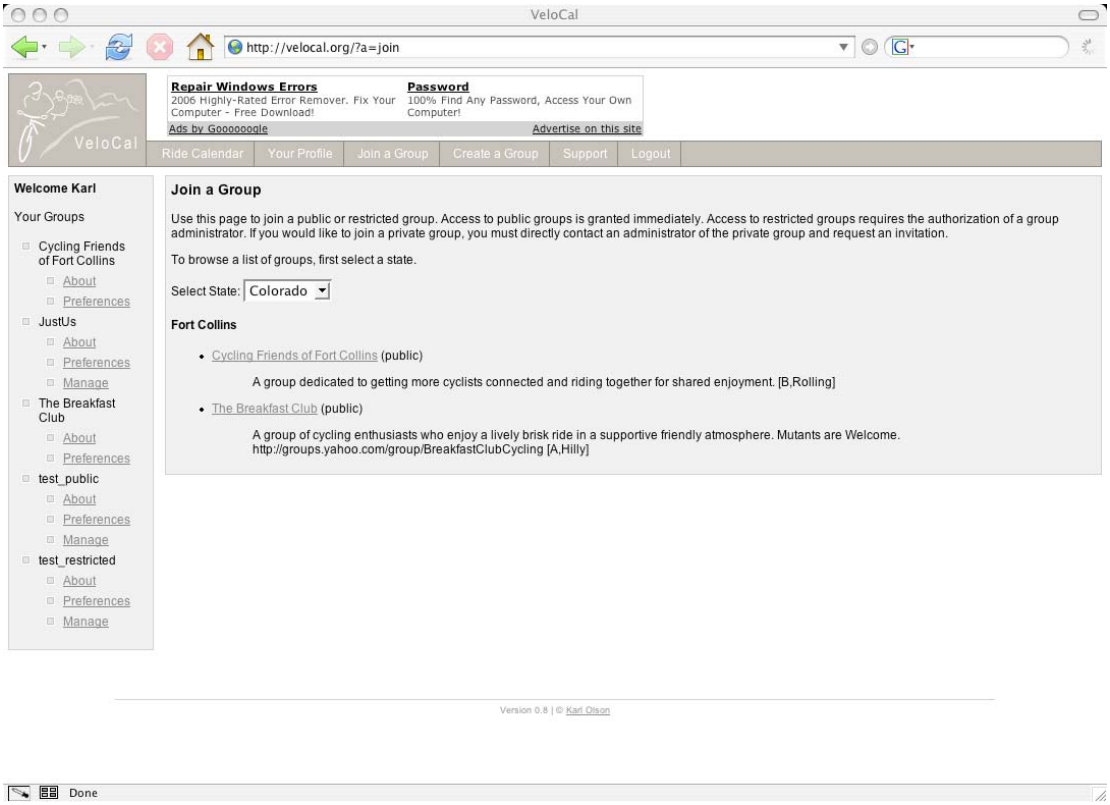
User's profile, including an optional photo

Edit Profile



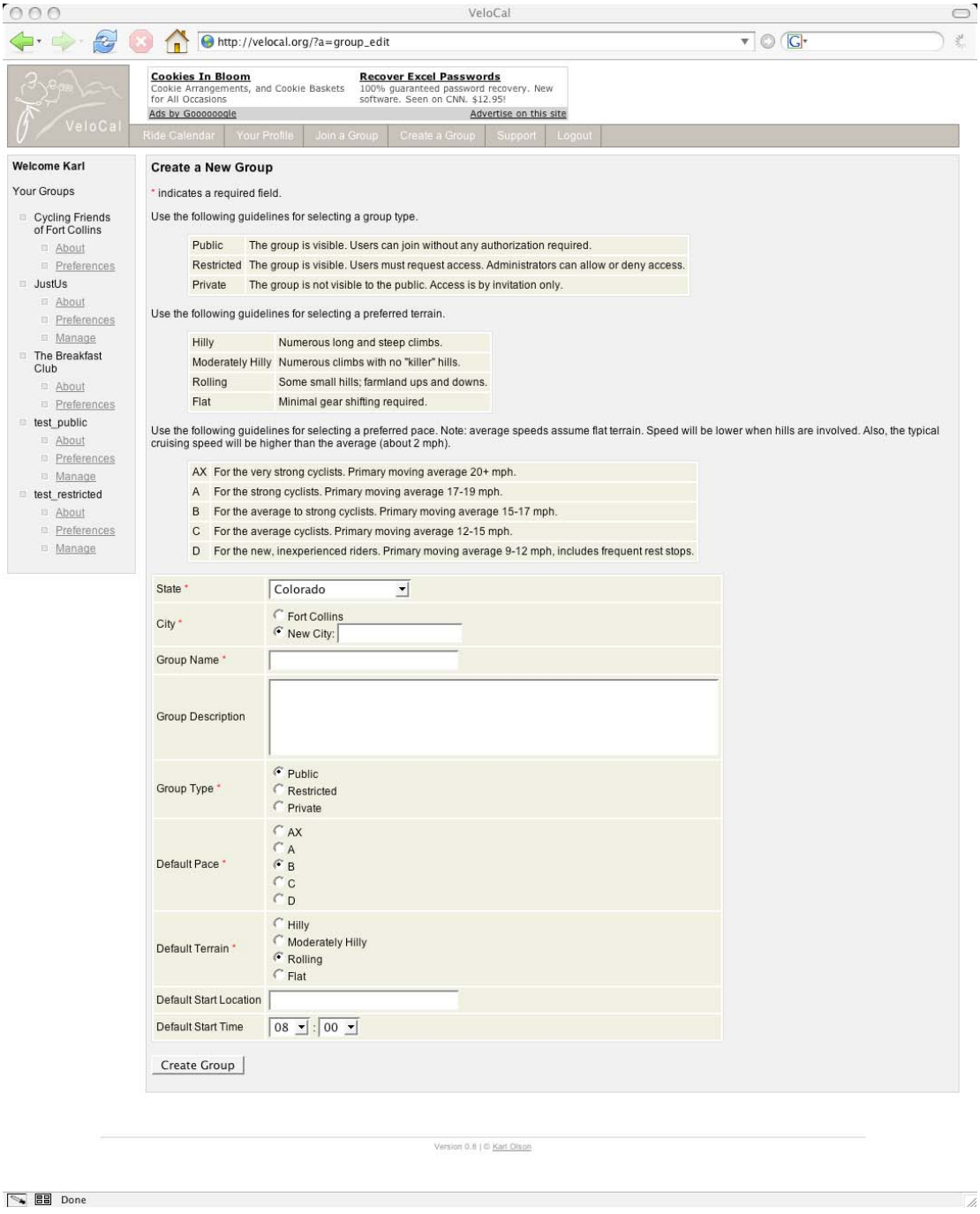
A user's profile edit page

Join a Group



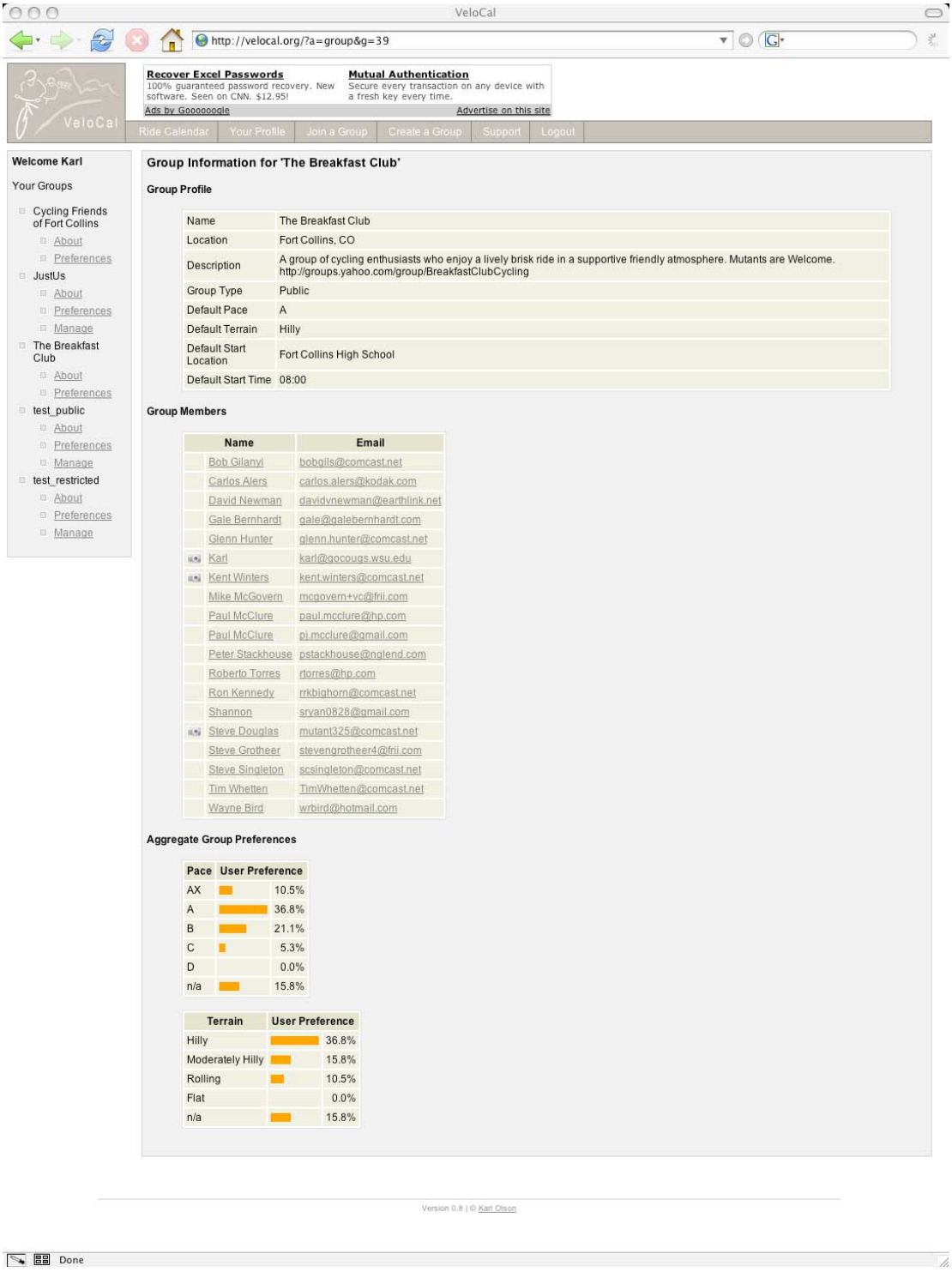
A list of groups to join, after first selecting a state

Create a Group



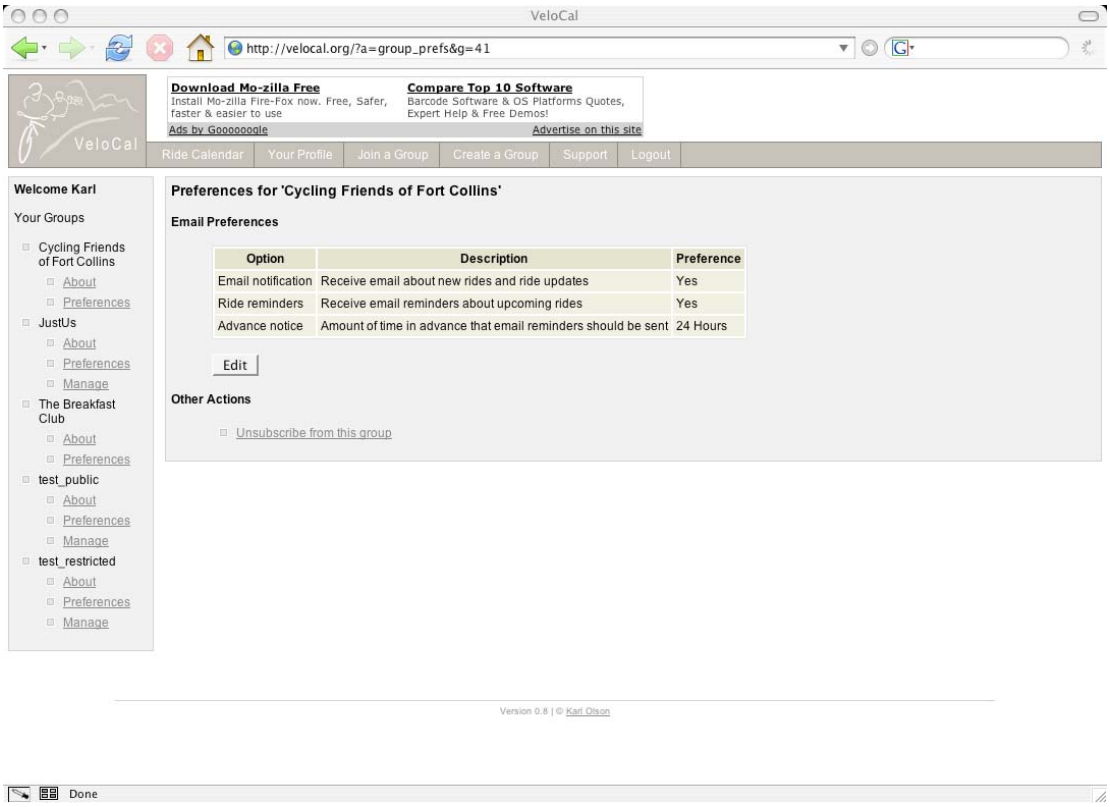
Creating a new group

Group Profile



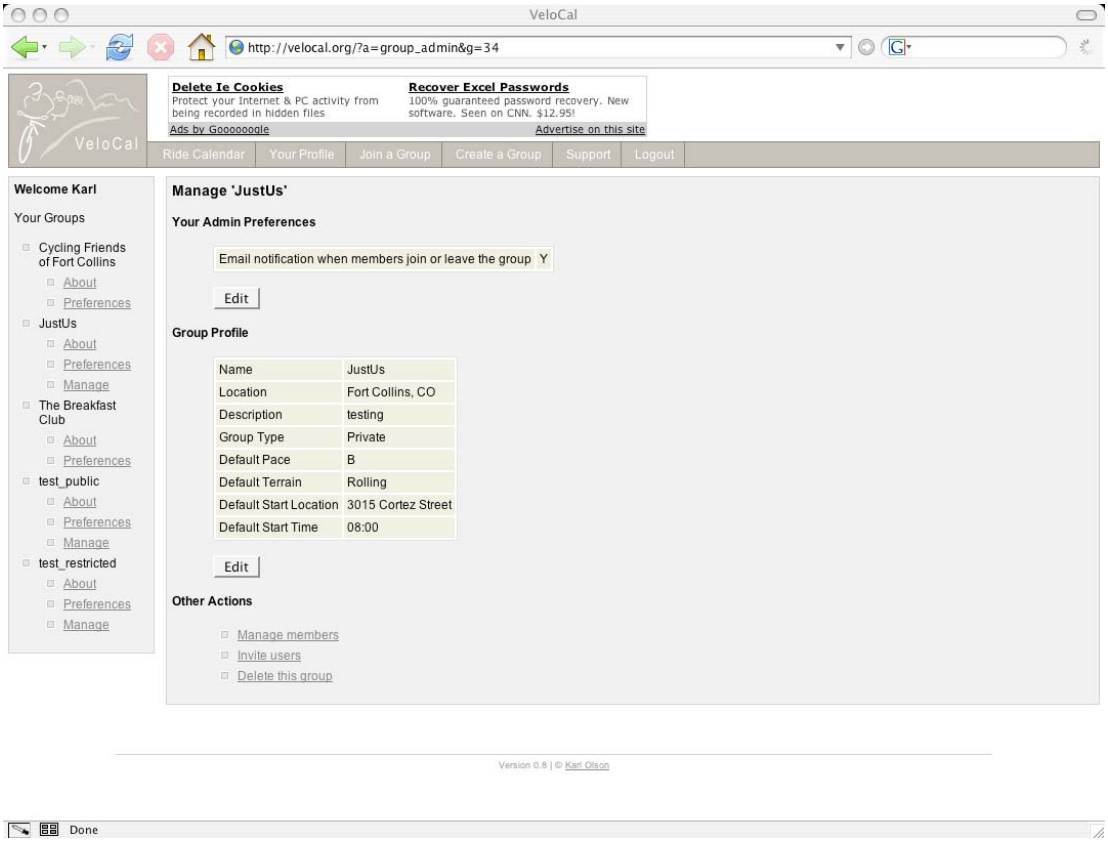
A group profile, indicating group defaults, a list of group members, and aggregate preferences

Group Preferences



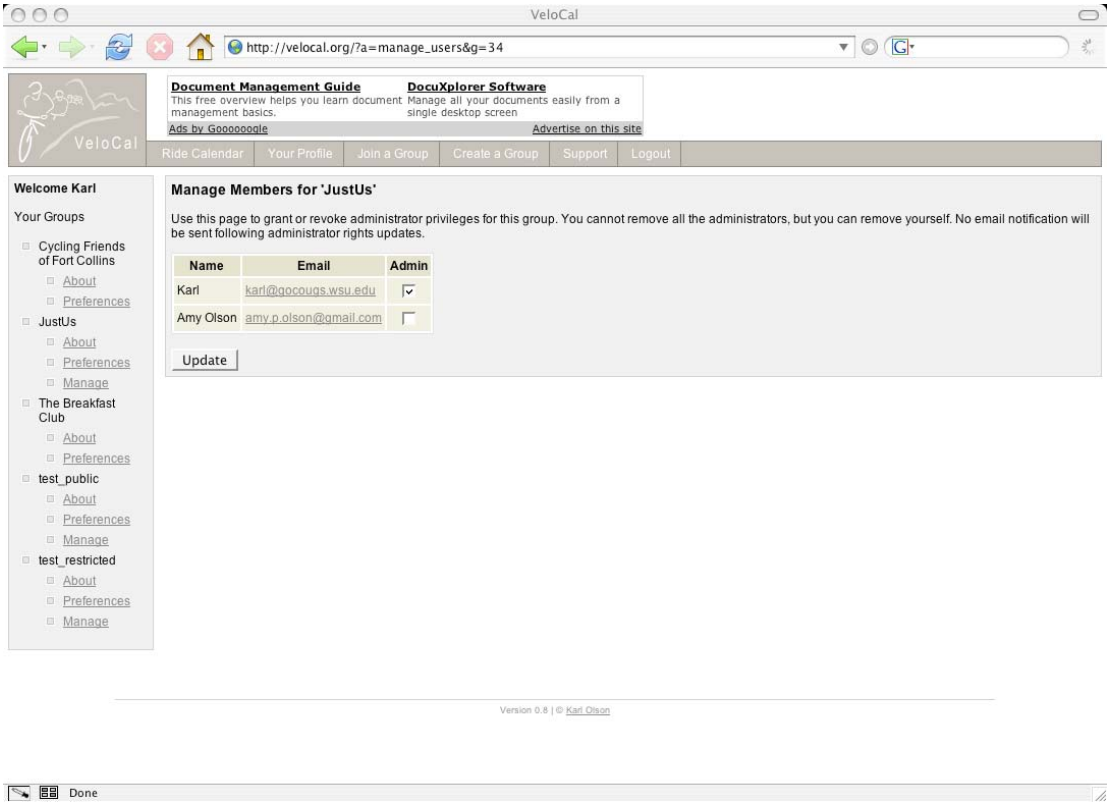
User preferences for an individual group

Manage a Group



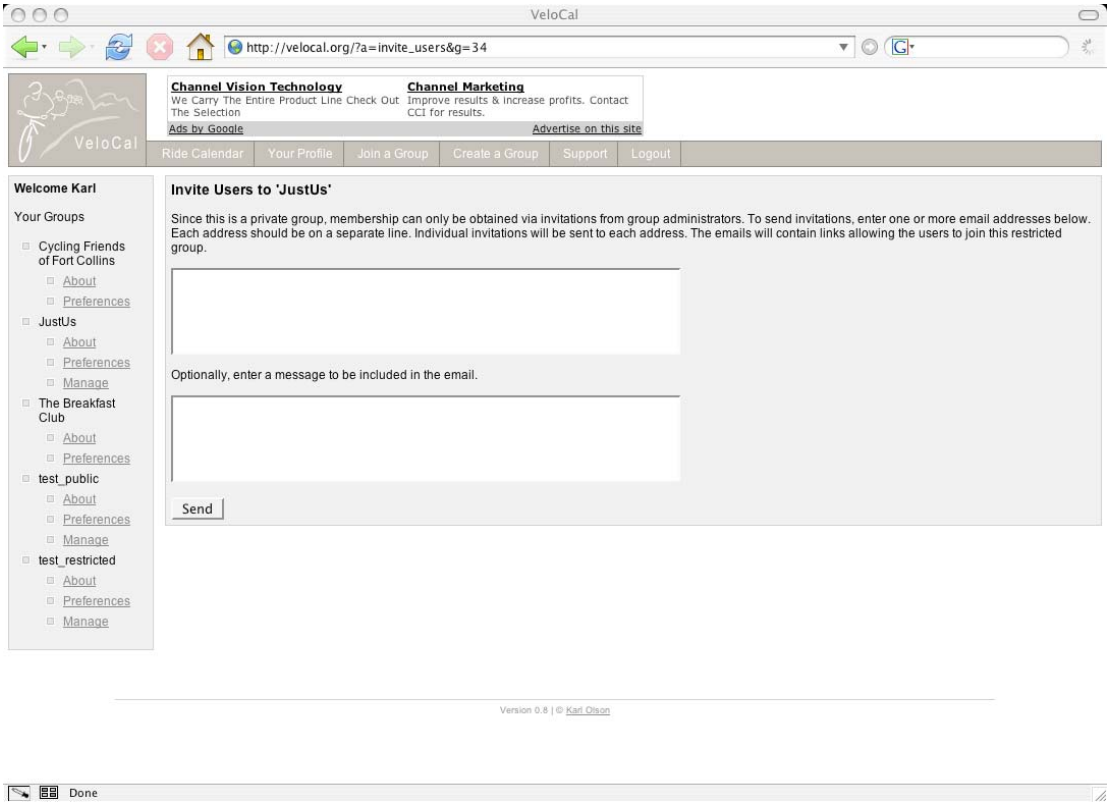
Administrative actions for a group

Manage Group Administrators



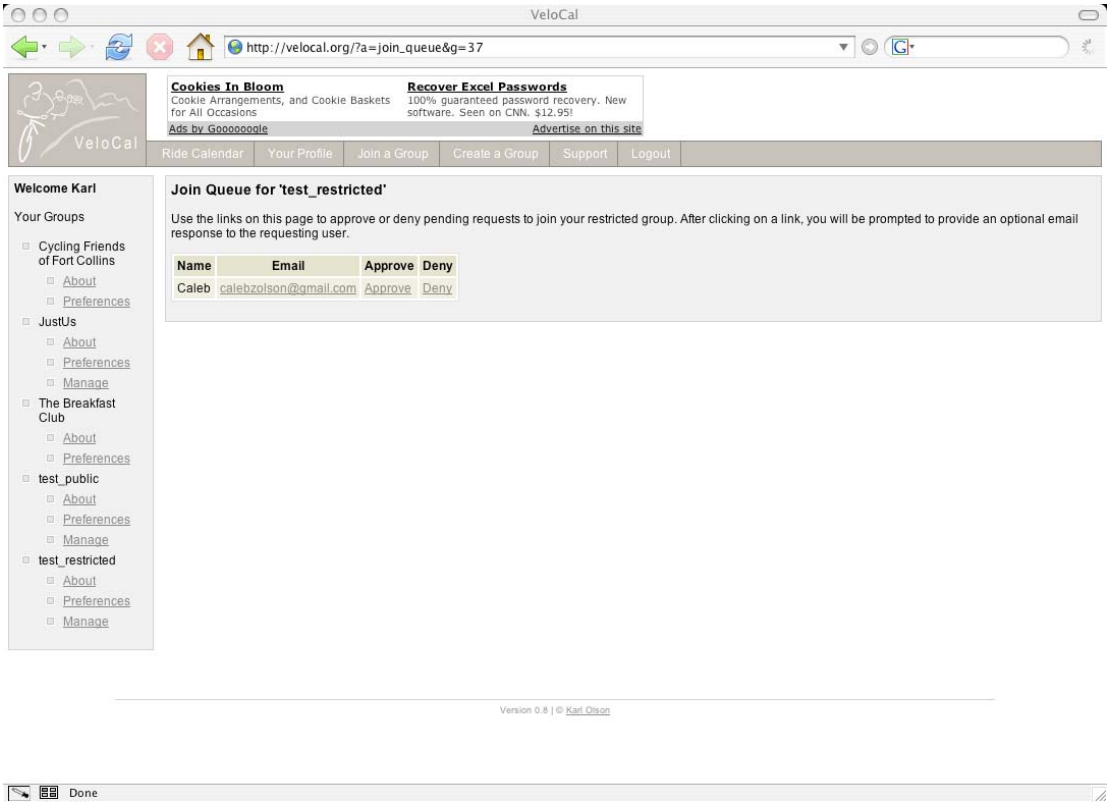
Granting/revoking administrator privileges to other group members

Inviting Users



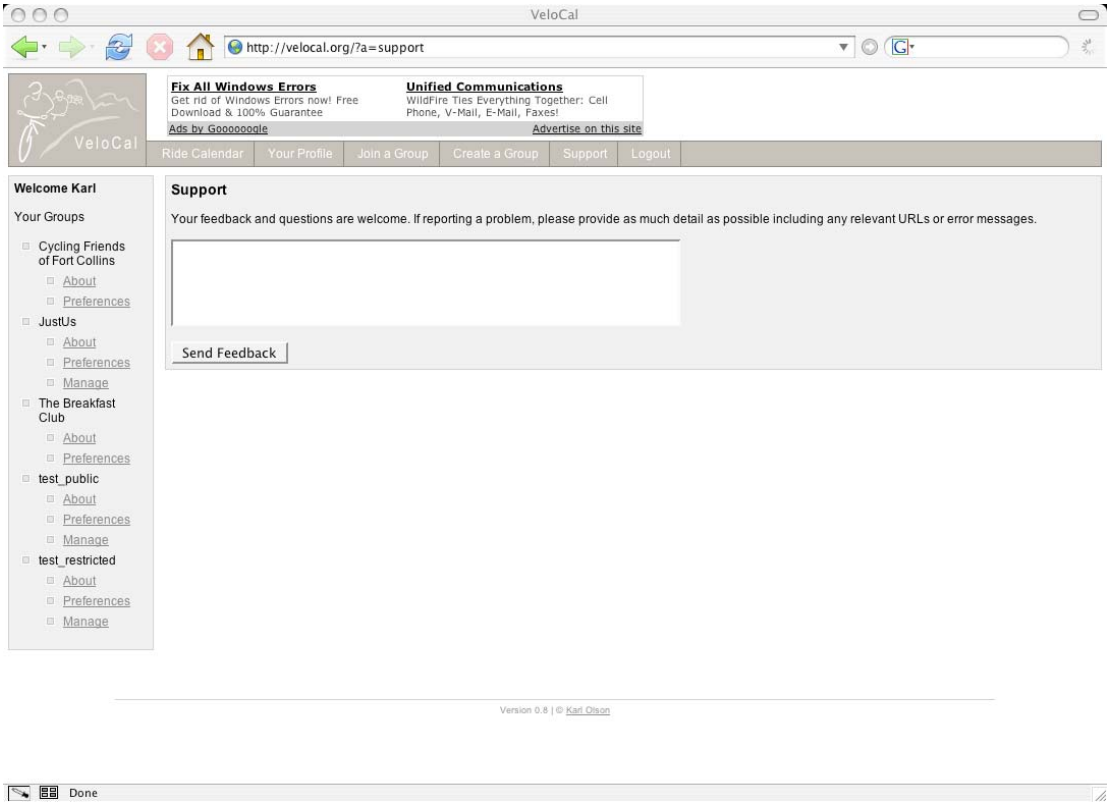
Inviting users to join a private group

Managing the Join Queue



Managing the list of users that have requested access to a restricted group

Support / Feedback



Support page, featuring an area for guests or users to send feedback or questions.

Appendix F: The VeloCal Logo

The VeloCal logo was created by Jay Kost. Loosely based on photos of cyclists enjoying annual Colorado cycling events, the logo abstractly depicts cyclists riding together as a group. Because of the author's connection to Colorado, the logo also incorporates images of dramatic mountain peaks.