**Group A**
Software Engineering C
Massey University
2008

# RoadMate

**19 – 9 – 2008**

Prototype 2

## Table of Contents

## Summary

**RoadMate** is a web application designed to assist commuters and avid travelers in organising car-pooling. As fuel prices continue to soar, our hope is that RoadMate will become an intricate part of the Kiwi travel planning process – Saving fuel, money and the environment. RoadMate is currently under development at Massey University as a 3rd year software engineering project.

## Project Plan

### Design Methodology

We were eager to take advantage of the benefits of the agile development techniques we have learned. Yet the design methodology we have adopted is one less agile and similar to RUP, this is for several reasons:

- Our development team is large (for the size of the project) and inexperienced. Agile development works well with small teams of experienced developers. Maintaining a repository of formal documentation, visual modeling techniques, a strong emphasis on project management all support general understanding of our model of the system, and help to keep us on track despite our inexperience.

- Skill and experience with the development platform varies significantly between team members. Agile approaches are dependent on all team members understanding the technology and being able to work relatively independently. We realize it is better for us to have one or two experts who fully understand the technology to develop the framework and teach the rest of us, than to have six equally inexperienced team members, all trying to find their way in the dark.

- There are constraints on reporting and demonstration that we as a team don't have control over. Although we can choose the development process to use, we can't avoid having to produce the project or weekly reports and the design documentation (Gantt charts, RMMM plans) that they should contain.

- We can be confident that the requirements are fully specified. We aren't dealing with a client who doesn't fully understand the system they want us to develop, so we don't need to rely on the agile techniques (like on-site client, feedback or revisions to the core requirements) that we would need if we were less certain about the requirements.

The method we have employed is a mix of aspects of agile and feature-driven development, and some Big Design Up Front practices. It is our attempt at applying the industry best practices that can meet the unique constraints of this project.

- **Iteration and prototyping** – seeking "client feedback" on delivered functionality

- **Prioritising requirements** and only implementing the top few, per iteration.

- **Use-case driven development**, each piece of functionality relates to a use-case, which in turn relates to the requirements

- **Model visually** – maintaining an ERD model of the system, so we all understand the entities and their members.

- **Maintaining documentation** – using Google Docs to collaborate on public working documents like meeting agendas, and using Subversion to store a version-tracked public repository of all design docs.

- **Code is double-checked** by a partner (not as extreme as **pair programming**; but coding as a group of two or three and rarely making changes alone).

- **Regular reports and meetings** to track the status of tasks and issues, and to report to the team manager.

## Development Process

### The Initial Approach

From the beginning, we approached this project from an agile perspective, focusing on use case driven development. We designed use cases to strictly adhere to the project specifications. Our entire effort then centered around building up the functionality defined in those use cases.

Lightweight systems were put in place to track and manage individual work. Though with such a small, tight knit team, and clear requirements, most of us were happy to sidestep these procedures for the sake of efficiency. The worst part was, because this allowed us to focus on the prototype, **we were always ahead of our initial expectations**. This only served to reinforce our opinion that **excessive documentation wasn't necessary**.

### The Consequences

Most management was done verbally; with time-consuming reports and Gantt charts being treated as an afterthought. Although this approach *seemed* to work for us, it made it difficult for managers to gauge the progress of our group. The minimal paper trail also showed signs of biting us once it came time to assess individual contributions.

### The Response

As an attempt to rectify this we set about formalising our development process. The end result being *Take Two*.

**Take Two** Improved Development Process: Better, Faster, Stronger

With Take Two, considerable effort went into designing a process that would work with developers. We focused on exploiting tools (repository, issue tracking, spreadsheets) to automate as much of the documentation and reporting as possible. The end result was a process, which offers real-time feedback to managers without hindering development efforts.

## Use-Case Driven Development – Take Two

As with our original development process, *Take Two* remains use case driven.

### Goals

The following were our main goals behind *Take Two*:

- Address the **disconnection between our documentation and the reality**
- Establish a standard way of tying project artifacts together
- Automate as much of the process as possible – developers should be focused on the site, not the reports

We wanted a living document. More importantly, **we wanted our documentation to be useful to us**: Something that integrated into our development process, as opposed to an afterthought between coding.

### What is in this Report

Below are the core artifacts of this development process. The report covers how they are generated, how they tie in with each other, and what the main goals behind each decision were. At the same time, we've tried not to bloat the report with the full specifications, as they should be available in the repository once finalised.

## Use-Case Driven Development – Take Two (Continued)

### Use Cases

Use cases underpin all our development efforts. Either directly (feature requested by the client), or indirectly (documentation, unit tests, .etc) everything ties back to a use case.

### Goals

The following were the main goals behind the new use case structure:

- **Standardised documentation**
  Use cases are all derived from a standard template and follow set naming conventions.
  By standardising the naming of use cases, we were able to make groupings clear, and enable easy referencing from other artifacts. This is especially helpful if the referencing artifact doesn't support direct links (i.e: Issue Tickets in Google Code).
- **Compartmentalised files**
  Use cases are now stored in separated files, as opposed to the original – monolithic file.
  The benefits from this were three fold: 1) Better support for concurrent modification of the use cases, which is crucial to agile development. 2) Allowed easy referencing back to individual use cases. 3) Allowed us to support use case versioning (see below).
- **Versioning** to manage changes in requirements
  Firstly **Version! = Revision**. Versioning tackles the issue of managing changes in requirements, without bloating the set of use cases in the process. Versions are tracked as separate *sheets* within each use case's Excel file.
  An example might help clarify this: Suppose our initial *Offer Ride* use case (version 1) only requires that users be able to offer rides *between* towns. Now consider that as our project progresses we decide to ride offers *within* towns or maybe allow users to plot rides offers using Google Maps. This functionality still falls under the *Offer Ride* use case, but just become part of version 2.

### Naming Convention

Use cases adhere to the following naming convetion:

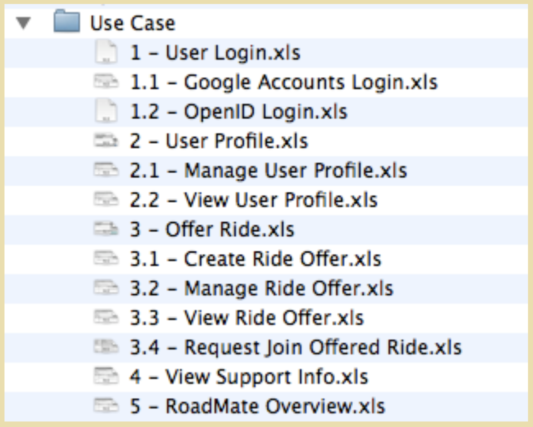
`<code>[.<subcode>] - <short name>` for example: **1 – User Login**

General use cases such as **User Login** can be broken into components when sensible. These sub-components are denoted using the dot notation, for example: **1.2 – OpenID Login** where **OpenID Login** is a sub-component of **1 – User Login**.

**Current Use Cases**

As an example the current use cases from RoadMate have been listed to the right.

These are available from the repository under /trunk/docs/Use Case/


Use Case
1 – User Login.xls
1.1 – Google Accounts Login.xls
1.2 – OpenID Login.xls
2 – User Profile.xls
2.1 – Manage User Profile.xls
2.2 – View User Profile.xls
3 – Offer Ride.xls
3.1 – Create Ride Offer.xls
3.2 – Manage Ride Offer.xls
3.3 – View Ride Offer.xls
3.4 – Request Join Offered Ride.xls
4 – View Support Info.xls
5 – RoadMate Overview.xls

## Use-Case Driven Development – Take Two (Continued)

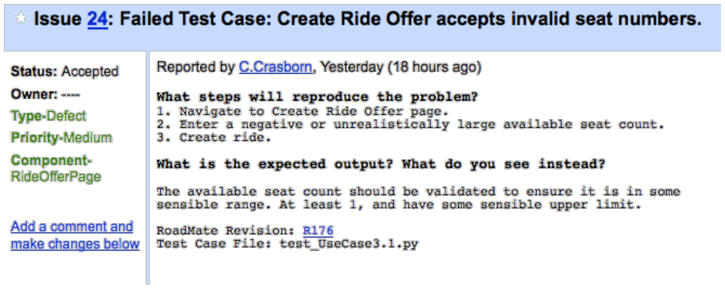
### Issue Tickets

As part of our improved development process, we aimed to take advantage of Google Code's issue tracking system to act as an online project management tool. We now use it not only to track issues, but also manage feature sets and development schedules.

### Issue Tracking

Issues are tracked just as in any other issue tracking system. We place strong emphasis on providing clear summaries, appropriate tags, and tying issues to the revision they were identified in.

When a test case fails we immediately record it in the issue tracking system with a back reference to the test case. (See example on left).



☆ Issue 24: Failed Test Case: Create Ride Offer accepts invalid seat numbers.

Status: Accepted
Owner: ----
Type-Defect
Priority-Medium
Component-RideOfferPage

Add a comment and make changes below

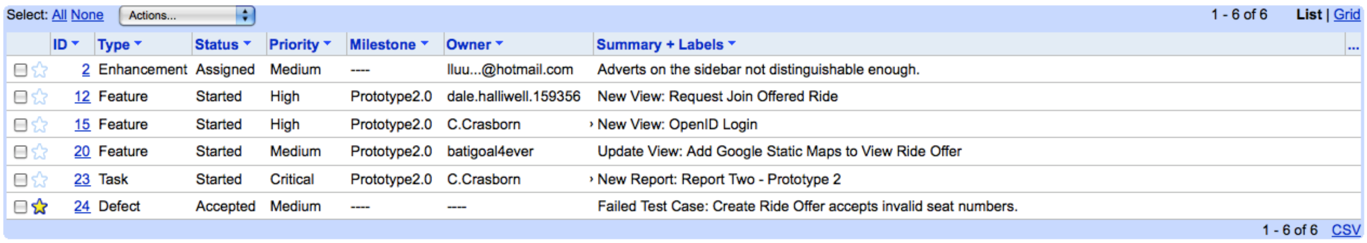
Reported by C.Crasborn, Yesterday (18 hours ago)

```
What steps will reproduce the problem?
1. Navigate to Create Ride Offer page.
2. Enter a negative or unrealistically large available seat count.
3. Create ride.

What is the expected output? What do you see instead?

The available seat count should be validated to ensure it is in some
sensible range. At least 1, and have some sensible upper limit.

RoadMate Revision: R176
Test Case File: test_UseCase3.1.py
```



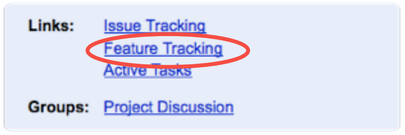
Select: All None   Actions...   1 - 6 of 6   List | Grid

| ID | Type | Status | Priority | Milestone | Owner | Summary + Labels |
|---|---|---|---|---|---|---|
| 2 | Enhancement | Assigned | Medium | ---- | lluu...@hotmail.com | Adverts on the sidebar not distinguishable enough. |
| 12 | Feature | Started | High | Prototype2.0 | dale.halliwell.159356 | New View: Request Join Offered Ride |
| 15 | Feature | Started | High | Prototype2.0 | C.Crasborn | New View: OpenID Login |
| 20 | Feature | Started | Medium | Prototype2.0 | batigoal4ever | Update View: Add Google Static Maps to View Ride Offer |
| 23 | Task | Started | Critical | Prototype2.0 | C.Crasborn | New Report: Report Two - Prototype 2 |
| 24 | Defect | Accepted | Medium | ---- | ---- | Failed Test Case: Create Ride Offer accepts invalid seat numbers. |

1 - 6 of 6   CSV

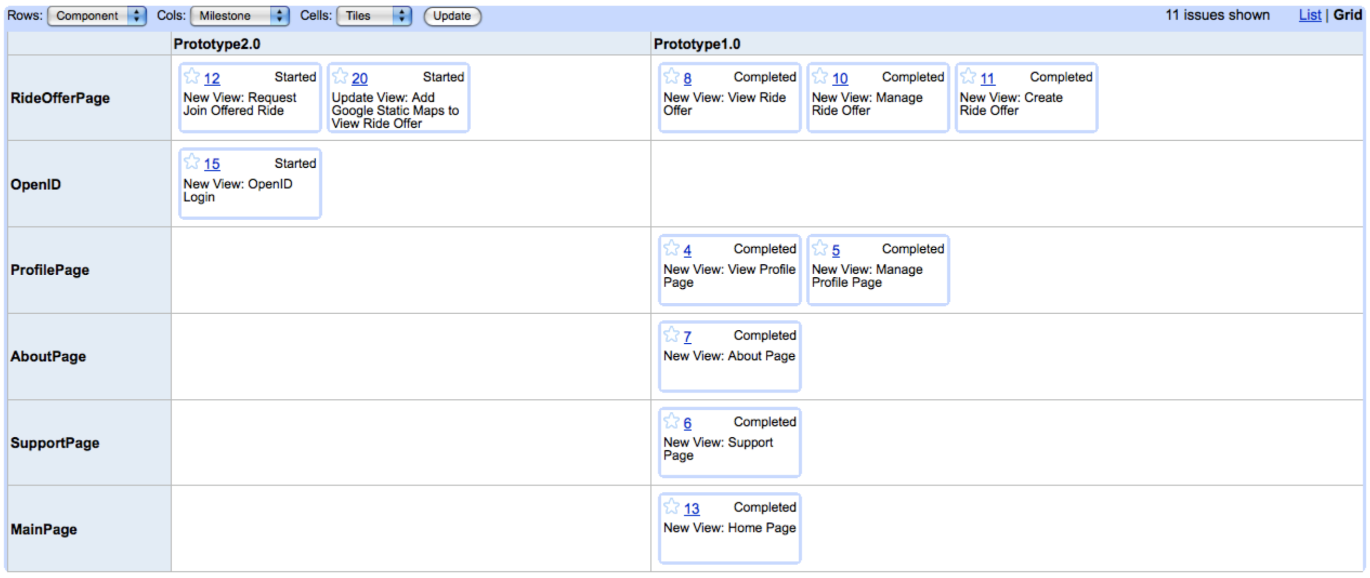
### Feature Management

Features are tagged with the site component they belong to and milestone we expect to have them completed by. With a little creative abuse of the issue tracking system, we are able to automatically generate current and expected feature sets. (See below).



Links: Issue Tracking
Feature Tracking
Active Tasks

Groups: Project Discussion



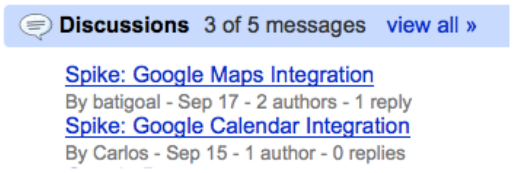
Rows: Component   Cols: Milestone   Cells: Tiles   Update   11 issues shown   List | Grid

| | Prototype2.0 | Prototype1.0 |
|---|---|---|
| RideOfferPage | 12 Started New View: Request Join Offered Ride; 20 Started Update View: Add Google Static Maps to View Ride Offer | 8 Completed New View: View Ride Offer; 10 Completed New View: Manage Ride Offer; 11 Completed New View: Create Ride Offer |
| OpenID | 15 Started New View: OpenID Login | |
| ProfilePage | | 4 Completed New View: View Profile Page; 5 Completed New View: Manage Profile Page |
| AboutPage | | 7 Completed New View: About Page |
| SupportPage | | 6 Completed New View: Support Page |
| MainPage | | 13 Completed New View: Home Page |

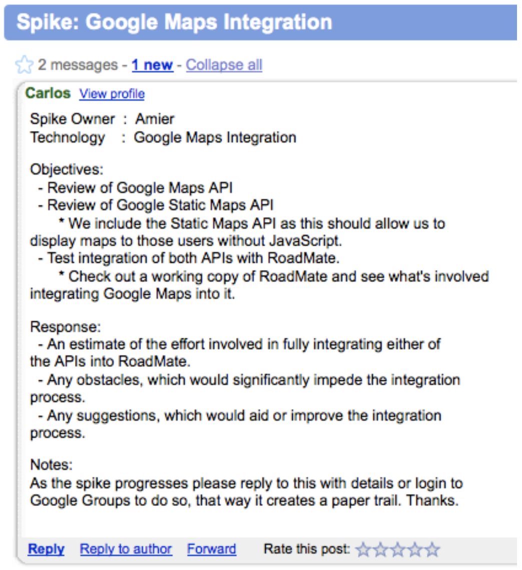
## Use-Case Driven Development – Take Two (Continued)

### Spike Management

Spikes are an important part of our development process. While active, spikes are tracked and managed as discussions on our Google Groups page. Because this acts as a hub for team communication, developers carrying out spikes can use this service to gain timely feedback and assistance from other team members. It also ensures that all discussions are documented in a single location.


**Discussions** 3 of 5 messages view all »

Spike: Google Maps Integration
By batigoal - Sep 17 - 2 authors - 1 reply
Spike: Google Calendar Integration
By Carlos - Sep 15 - 1 author - 0 replies


**Spike: Google Maps Integration**

2 messages - 1 new - Collapse all

**Carlos** View profile

Spike Owner : Amier
Technology : Google Maps Integration

Objectives:
- Review of Google Maps API
- Review of Google Static Maps API
    * We include the Static Maps API as this should allow us to display maps to those users without JavaScript.
- Test integration of both APIs with RoadMate.
    * Check out a working copy of RoadMate and see what's involved integrating Google Maps into it.

Response:
- An estimate of the effort involved in fully integrating either of the APIs into RoadMate.
- Any obstacles, which would significantly impede the integration process.
- Any suggestions, which would aid or improve the integration process.

Notes:
As the spike progresses please reply to this with details or login to Google Groups to do so, that way it creates a paper trail. Thanks.

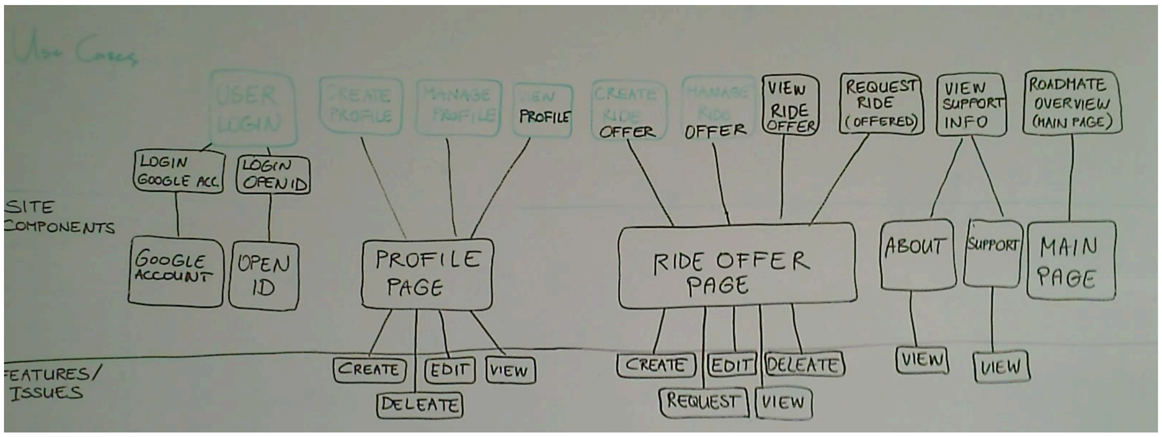
Reply Reply to author Forward Rate this post: ☆☆☆☆☆

Our initial plan was to compile all the documentation collected during a spike into wiki page, once the spike ended. This would have provided a more cohesive record of the spike for developers, but due to time constrains was left as an optional requirement.

In the interim, the discussion pages on Google Groups (see right) provide an adequate record of spike activity.

### Tying it Together

Below is one of the early diagrams we drew it during the design phase of this development process. As of our current prototype it is slightly out of date, but serves to illustrate the connection between use cases, site components and tickets on the issue tracking system.



## Risk Mitigation, Monitoring and Management Plan

| Name | Category | Probability | Impact | Description | Monitoring | Proactive Actions | Reactive Actions |
|---|---|---|---|---|---|---|---|
| Loss of personnel | Project | Low | High | One or more team members decide to leave the team. | Monitor team members to see if they seem likely to leave. | Get a couple of weeks early notice from the team member before he/she leaves. | Reassign tasks to the available team members, and inform the manager of the loss of the team member. |
| Miscommunication with the client | Project | Low | High | Misunderstanding the client requirements can occur as project goes on. | Always check with the client whether the project is meeting their needs and if they are happy with what has been developed already. | During the prototypes demonstration with the client, check with client if everything is on track. | Find out what has been misunderstood and re-plan to meet the client needs. |
| Authentication system fails or has flaws. | Project | Low | High | Design flaws in the system allows for unauthorised access to the system. | Do a regular check, testing different components of the authentication system. | Put procedures in place to fallback to the Google authentication system, since it is a more reliable system. | Fallback from the OpenID system to the Google authentication system. |
| Project goes over time | Business | Low | Medium | A possibility is that the team doesn't manage to finish the project on time, this can happen as a result of lack of preparation. | Develop a Gantt chart that has set milestones for critical stages, this will help us identify our progress. | Keep modifying and updating the Gantt chart so that it shows "milestones". | Attempt to renegotiate the deadline, explaining why that is the case. |
| Problems with technology (Google App Engine) | Technical | Medium | High | We will be using some new technologies and there is the possibility that we struggle and don't mange to learn the new tech. | Team leader to continuously check with team members and see how comfortable they are with the technology, and if they aren't, find out what's bothering them and help them to manage it. | Before choosing the technology make sure that team member are confident in using the technology and they have no problem learning it. Also have another technology that can be used with the same PIM | Find out the problems that are faced and try and solve them, otherwise change the technology to a Java development environment, since this is widely supported and understood at the university. |
|  |  |  |  |  |  |  |  |

Continued

| Name | Category | Probability | Impact | Description | Monitoring | Proactive Actions | Reactive Actions |
|---|---|---|---|---|---|---|---|
| End product has usability issues | Technical | Low | Medium | User finds it hard to understand the GUI, and find it hard to navigate the website. | Adopt good HCI practices. Test usability. | Test the GUI before the final deployment. Tests must include a wide range of users (novice and expert). | Understand the difficulties the user is facing and modify the GUI in accordance to that. |
| Project requirements change | Project | Medium | Medium | Client changes the requirements sometime during development. | Regularly check progress against Requirements. | Try and maintain regular communication with the client. Use agile development techniques. | Check if the new requirements require re planning and reassigning tasks to the team. |
| Deviation from software engineering standards | Technical | Low | Medium | With the progress of the development it is possible to lose track and start deviating from the common and known software engineering standards. | All relevant documents must be complete and as accurate as possible to ensure they conform to common software engineering standards. | Technical reviews to be done during major milestones to ensure that the project is in line with established software engineering standards. | Steps must be taken back as soon as the deviations are spotted, Try and guide the project back to meet the standards. |
| Loss of data | Project | Low | High | While team members are working on their own copy of the application, they might encounter loss of data. | Keep tabs on all working copies. | Save regularly and commit often to the repository. | Get the team member who was working on that lost data to start re-developing what's lost immediately, assign another team member if help is needed. |
| People don't work well together | Staff | Medium | Medium | Team members fail to communicate with each other. | Team leader to find out early how does the team work together and decided whether a change is needed. | Get regular feedback from members to see if there are problems. | Find out what's wrong, try and solve the problem. Don't let it affect the rest of the team. |
| Lack of team Knowledge | Staff | High | Low | The team might lack someone who specializes in a particular field. | Up to the team leader to know each team member's strengths and weaknesses. | Before the implementation phase check whether the members have the knowledge to accomplish all the given tasks. | Seek knowledge; assign task to a member and the member needs to learn what's needed to accomplish the task. |
| | | | | | | | |

# Design Documentation

## Use Case Diagram

Below are the use case diagram and actor descriptions for our second prototype.

### Actors

**Visitor** – Any visitor to the RoadMate website.
**Driver** – A registered user, who is offering a ride.
**Passenger** – A registered user, who is accepting a ride.

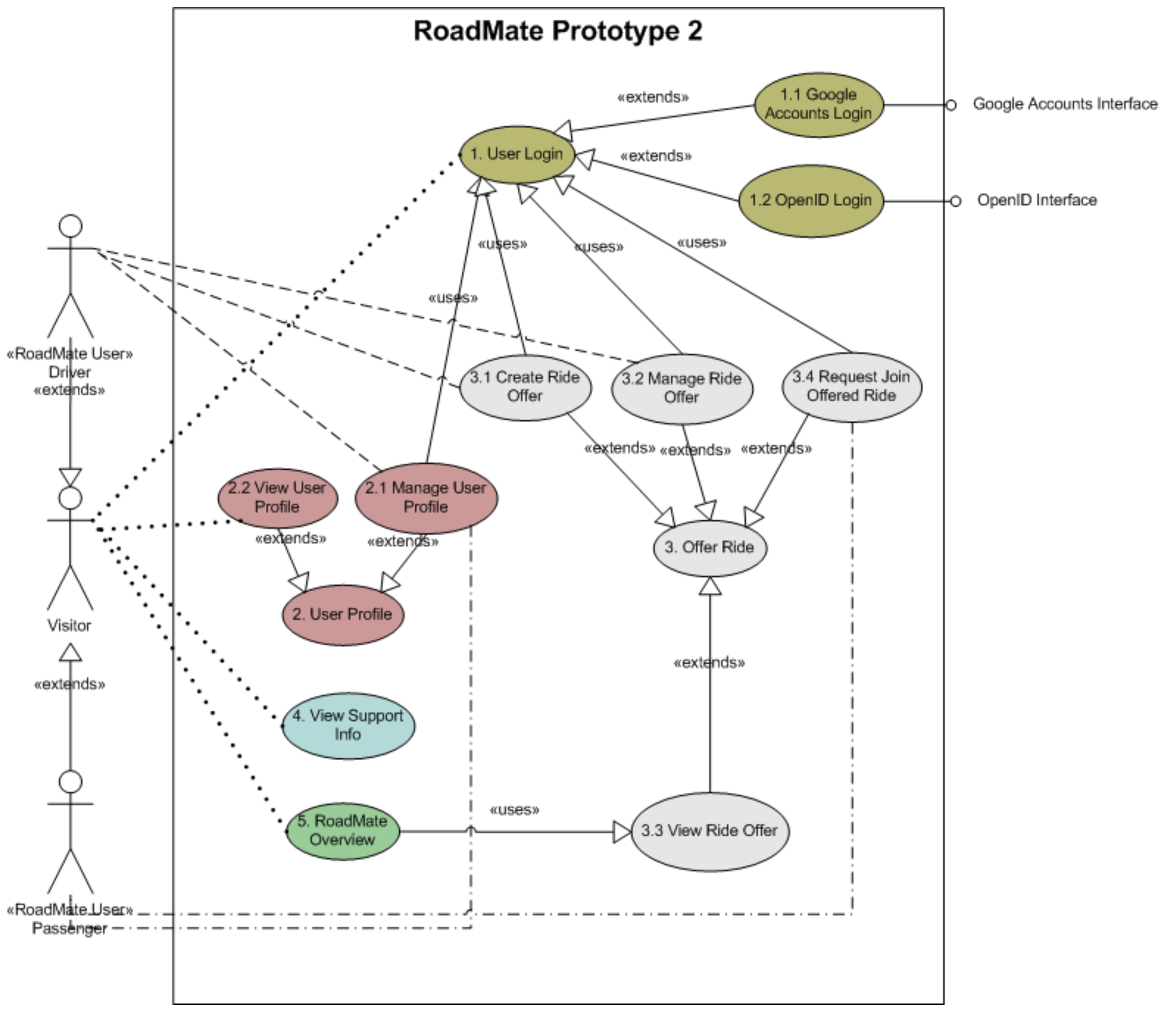
**Note** Use cases are colored

## Use Case Descriptions

The individual use case descriptions in standard template form are **available from the repository**.

### Repository Location

/trunk/docs/Use Case/



## Entity – Relationship Model

From the relatively tight scope of our first prototype, we needed to refactor our model to take into account the additional Use Cases that we included in the scope of our second prototype. From our first prototype, the number of entities in our ER model has doubled, we had only three entities: RoadMateUser, RideOffer, and Town.

From the feedback we got on the first prototype, we knew we needed the ability for users to define Locations as any street address with a name tag attached, instead of having a Town entity; this meant the rides could happen within cities, not just between them.

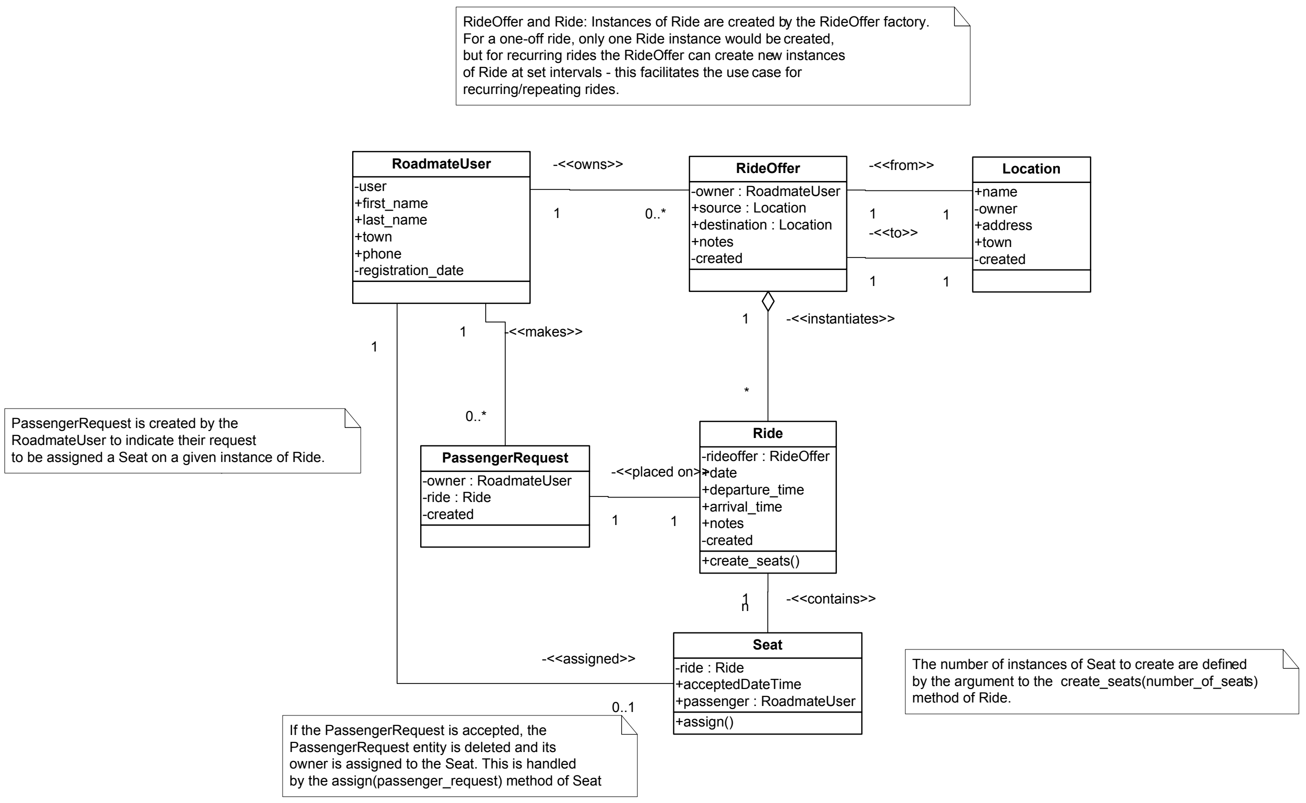
We needed a way of implementing recurring rides, the solution was to use the Abstraction-Occurrence design pattern and to split off the recurring attributes of RideOffer. We defined a RideOffer as the offer of one of more rides between Locations - this represented the <<Abstraction>> stereotype and would hold the attributes general to all instances of the occurrence (the owner, the start point, end point, etc), and created the Ride entity of an <<Occurrence>> stereotype, which would represent the instances of those rides and hold the instance-specific attributes (such as date, departure time, and the available seats).

The Seat objects themselves are child objects of the Ride, since the number of seats on any given ride can vary, and we need a way of bounding the relation between passengers and rides, so that there can never be more passengers than there are available seats. The number of available seats is easy to calculate at any time, just take a count() aggregation of the number of Seats that do not have assigned passengers.

Also, we needed some way of allowing users to place a request to become a passenger on a given ride (UC 3.4 - Request Join Offered Ride). We decided to use an entity representing this request to become a passenger, using a note as a metaphor. Thus the PassengerRequest object is placed against the Ride, and if it is accepted, its owner is assigned to a Seat on that Ride, and the PassengerRequest itself is destroyed.

Up-to-date models are available from the repository at /trunk/roadmate/roadmate/models.

### Prototype 2


RideOffer and Ride: Instances of Ride are created by the RideOffer factory. For a one-off ride, only one Ride instance would be created, but for recurring rides the RideOffer can create new instances of Ride at set intervals - this facilitates the use case for recurring/repeating rides.

PassengerRequest is created by the RoadmateUser to indicate their request to be assigned a Seat on a given instance of Ride.

The number of instances of Seat to create are defined by the argument to the create_seats(number_of_seats) method of Ride.

If the PassengerRequest is accepted, the PassengerRequest entity is deleted and its owner is assigned to the Seat. This is handled by the assign(passenger_request) method of Seat

## Project Scope

### Terminology

The following is brief outline of the terminology used within the project scope section.

**Visitor** – A person viewing the site, who may or may not be logged in.
**User** – A person viewing the site, who **is** logged in.

### Prototype One

The following is an outline of the site structure and functionally as provided in the current iteration of prototype one.

- **Home**
  The home page acts as the main gateway to the rest of RoadMate.
    - Welcomes visitors and provides a brief description of RoadMate.
    - Provides a list of the latest rides submitted to RoadMate. (Limit 10)
    - Provides any important news relating to RoadMate or its users.

- **Profile**
  The profile page serves two purposes:
    - Serves a read-only version of a users profile to visitors.
    - Serves an editable version of the profile page to the profile owner.

- **Offer Ride**
  The offer ride page provides all functionally pertaining to ride offer creation, modification and viewing.

- **Browse Rides**
  The browse rides page provide basic search functionally over the currently offered rides.
  The current search criteria are:
    - Source – Where the ride it leaving from.
    - Destination – Where the ride is going.

- **Support**
  Provides basic help information to visitors
    - Explains basic functionality of RoadMate.
    - Provides technical contact information.

- **About**
  Explains what RoadMate is, and what we're trying to achieve with it.

### Prototype 2

Most of the effort during our second iteration cycle went into the formalisation of our development process. As such this was feature-light release.

- **Added Support for Google Maps**
- **Transitioned from Town-based system to Location-based system.**
  This allowed us to support rides within towns
- **Significant restructuring of the underlying code base to support both Google Accounts and OpenID**

## Software Quality Assurance

### Unit Testing

As part of our software quality assurance process, we aim to make extensive use of unit testing to ensure our code meets both functional and non-functional requirements.

**Selenium** A suite of tools to automate web application testing

Selenium IDE is a Firefox add-on that records clicks, typing and other actions to make a test, which you can play back in the browser. Selenium also supports exporting your actions as unit tests, which can be integrated into your project.

With the help of the Selenium IDE we can generate unit tests, which emulate common actions users would make while using RoadMate. These site tests in combination with our server side tests give us complete coverage over the functionally provided by RoadMate.

Because these unit tests are platform independent, they easily integrate into our continuous integration process:

- **Developers**
  All unit tests must pass before code can be checked into the repository.
- **Local Server**
  Every 20 minutes our local server downloads the latest version of RoadMate from the repository. We hope to have it use the unit tests to determine whether it should update is live copy of the site. The feasibility of this is currently being explored.

### Usability

As part of our software quality assurance process, we aim for the highest standards in usability.

As a testament to this, **our site functions without the need for JavaScript**. This allows our site to be accessed from a wide range of mobile devices, which don't have support for JavaScript.

We are aiming for government standards in accessibility and XHTML compliance to ensure the widest range of users have access our site.

### Usability Testing

Usability testing is an intricate part of our development process and something we take very seriously.

In order to ensure our site meets the highest standards in usability (our own), we make use of internal and external parties to test our site and report issues back to us.

For external parties, we have the intension of using Silverback (see below) to record user experiences help us decide at meetings

### Testing Matrix

As part of our software quality assurance process, we continually test our site against the major browsers for any inconsistencies.

Testing is predominantly carried out on **Firefox** and **Internet Explorer 7,** as these are the browsers installed on our development machines. Testing on the reaming browsers is carried out when approaching a point release.

**Silverback** Spontaneous, unobtrusive usability testing software for designers and developers

Silverback makes it easy, quick and cheap for everyone to perform guerilla usability tests with no setup and no expense, using hardware already in your Mac. By capturing screen activity along with videoing the participant's reactions and voice, Silverback provides a complete usability-testing suite.

## Individual Member Skill Sets

As requested, below is a collection of brief snippets covering the skills developed and applied by each team member over the course of this project. This is by no means an exhaustive list, and obviously full details of skills and technologies used can be found in each team member's individual weekly reports.

**Amir**

- HTML
- Python programming
- Database management (SQL)
- UML (understanding class diagrams and entity relationship diagrams)

**Carlos**

- HTML & CSS
- Python programming (Django + Google APIs)
- Bash scripting
- Database modeling
- Repository management
- Issue tracking
- Team organisation

**Dale**

- HTML & CSS
- Python programming
- Database modeling
- UML modeling (ER-Model)

**Lucia**

- HTML
- UML modeling (Use Case Diagrams)
- Excel (Use Case Descriptions)
- MS Project (Gantt Chart)
- Issue tracking

**Yizhang**

- HTML & CSS
- Python programming
- Selenium Unit Tests