Transportation for Visual Impaired  
Cycle 1 Report

By

Aymeric Zuurhout

Tyler Espy

Taylor Lucy

Submitted in partial fulfillment of the requirements for  
COMP 4710 Senior Design  
to the Department of Computer Science and Software Engineering,  
Samuel Ginn College of Engineering, Auburn University

Auburn, Alabama

October 4, 2015

# Table of Contents

Table of Contents 2

1 Executive Summary (System Metaphor) 3

2 Project Introduction 4-3

2.1 Previous Development 4

2.2 Intent This Cycle 4

2.3 Future Work 4

3 Requirements / User Stories 5

3.1 User Stories 5

3.2 User Stories 6

4 Design Documentation 7-8

5 Management Plan 13

5.1 Task Assignments 13

5.2 Development Schedule 14

5.3 Planned Code / Feature Freeze 14

6 Risk Mitigation 15

7 Test Plan and Test Procedures 12

8.1 Test Plan 12

8.2 Test Procedures 12

8.2.1 Procedure 1: Sample Test Procedure 13

9 Lessons Learned 14

References 15

Appendix A 16

A.1 Status Reports CD

A.2 Meeting Minutes CD

A.3 Size Estimation Documentation CD

A.4 Problem Reports / Change Requests CD

A.5 Correspondence CD

A.6 Source Code CD

# System Metaphor

Alabama Institute for the Deaf and Blind has a transportation service located in the greater Talladega area. They currently do all their reservations manually by hand. Our goal is to modernize, optimize, and automate their reservation process with an online database system and a webpage and IOS application frontend.

# Project Introduction

In this project we are building a transportation system for the Alabama Institute for the Deaf and Blind. It is located in Talladega, and there are two functioning schools in the area. Their current transportation system is directed towards aiding those in the community who are unable to commute on their own due to their disabilities. The Transportation Department consists of three individuals, Zoe Rose, Jennifer Reeve and Donte Little. Both Ms. Rose and Ms. Reeves (later referred hereafter as “schedulers”) handle scheduling responsibilities with Mr. Little as their manager. There current system is mostly done by hand. They field calls from clients needing transportation to and from a specific location anywhere in Alabama on a specific date. The date, location and time varies and is handled on a first come first serve basis. During the call schedulers, follow a simple process described below: (Steps followed in sequential order)

1. Check for availability on an Outlook calendar
   1. If opening is available they ask for client information
   2. If opening is unavailable they decline give the client available openings
   3. When opening is chosen schedulers move on to step 2
2. Ask client for information and handwrite the information on Appointment sheet
3. Create and event in outlook on the date chosen
4. Insert scheduled information in event
   1. Client Name
   2. Start Time
   3. End Time
   4. Scheduled driver
   5. Vehicle
5. Store hand written Appointment sheet in binder for future use
6. Field further calls repeating step 1-6

At the time of an appointment

1. Schedulers find Appointment sheet in binder
2. Transfer information from Appointment sheet to Driver Information Sheet
3. Driver picks up the sheet from schedulers and conducts the transport
4. Upon successful transport Driver initials form and returns it to schedulers
5. Schedulers file form record keeping
6. Steps 7 – 9 continue until all transports have been conducted

Our project is devoted to making this process easier by digitalizing it, replacing hardcopies with digital forms. Our system will feature connectivity between three primary forms:

1. Calendar Form
2. Scheduling Form
3. Driver Information Form

All of which retain an ability for easy printing to make record keeping easier, an auto-fill feature will be implemented to reduce the redundancy of filling in the same information across forms. Our users (schedulers) themselves are visually impaired, to facilitate their disabilities all forms are made compatible with programs such as JAWS and Zoomtext designed to help visually impaired read and use information.

Compatibility In Depth:

1. Compatible means: able to exist or occur together without conflict
2. A compatible form in this case comes in various shapes, sizes and colors
   1. One form is viewable in negative colors
   2. One Expands the screen, to include a single text field

The calendar also contains the aesthetic qualities to help our customers see events, by choice of month, week, or day. Each event is placed close the one prior, removing white space that previously inhibited the schedulers. With the calendar the schedulers will be able to select a date, view currently scheduled appointments, and if an opening exists create a new appointment. Each day contains a link to the Scheduling Form to make scheduling a new appointment easy. After an appointment is saved it is viewable from the calendar. On the day of an appointment the scheduler, can simply click the event upon which a link transfers them to an auto-filled Driver Information Form. From there they can view appointment information and simply print out the form for Drivers Use. A new step by step process is shown below:

1. Check for availability on calendar
   1. If opening is available they ask for client information
   2. If opening is unavailable they decline give the client available openings
   3. When opening is chosen schedulers move on to step 2
2. Ask client for information and enter information in Appointment sheet
3. Print out Appointment Sheet and save appointment to calendar
4. Store Appointment sheet in binder for record keeping
5. Field further calls repeating steps 1-4

At the time of an appointment

1. Schedulers click on appointment in calendar
2. Print out Driver Information Form
3. Driver picks up the sheet from schedulers and conducts the transport
4. Upon successful transport Driver initials form and returns it to schedulers
5. Schedulers file form record keeping
6. Steps 7 – 10 continue until all transports have been conducted

## Previous Development

**Architectural Spike**

Starting from scratch during our Architectural Spike we set a goal to have an initial working form, rough calendar layout, and set up development tools. We completed the initial form, a date selector and set up development tools listed below.

1. Git repository for team sharing
2. AptanaStudio3 workspace for html, css, php, JavaScript development
3. Development Server

**Cycle 1**

In Cycle 1 we set a goal to clean up our initial forms, and set up connectivity between all forms. We added a print out form, cleaned up the initial reservation form and implemented a new calendar design.

## Cycle Intent

Cycle 2 revolved around 4 primary tasks.

1. AutoFill Print Out Form

Detailed Description:

The print out form is designed as simple reference sheet for the driver. Each is linked to a particular reservation stored in the reservation table, in our MySQL Database. The form is accessed by clicking an event shown on the calendar. This event is connected to the same Reservation Table entry as the print out form. Using the information stored on the event (id) the print out form finds the desired Reservation table entry, pulls the information, and displays the information in the desired fields.

1. Populate Calendar

Detailed Description:

The Calendar API we are using has a simple event parser that takes in a string of information labeled by identifiers. Using this information the calendar can populate itself with the desired events. Our events are reservations. Reservations are stored in a reservation table in our MySQL database. When the calendar is initialized (i.e. the link to it is clicked, this link could be anywhere, for instance I recently emailed one to our client) a string of events is created using the information stored in the Reservation Table. This string is parsed by our calendar and the resulting calendar displays the events on the required date.

1. Show Real Car Availability

Detailed Description:

An emphasis for our customer was an ability to see Real Car Availability. Our design and implementation of this feature is on the Reservation Form and the Calendar. Each car is labeled as a color. The calendar colors each event depending on the car assigned for the reservation. This allows the schedulers to easily see the availability of each driver. Unless specified, the Reservation form automatically assigns a car depending on its availability. Availability is determined by information stored in the database, further described below.

Each Reservation has a:

* expected duration of time
* the start time
* the car being used.

If the schedulers try to over book a vehicle, the reservation form will not allow it. Saving is prohibited under this circumstance, and an error message is displayed describing the error below the car.

1. Create new reservations

Detailed Description:

A new reservation is created through the Reservation form. When the submit button is clicked the information in the text fields of the Reservation form are saved to an newly created slot I the Reservation Table. Every field has a corresponding field in the Reservation Table. Each slot in the Reservation Table is defined by an ID. The id auto increments, to insure each Reservation is unique.

## Future Work

Our remaining work will revolve around two concepts: Refactoring and Additional Features. Specifically, we would like to refactor the database to better organize our information. Additional Features include but are not limited to:

Edit previously recorded reservations from print out form

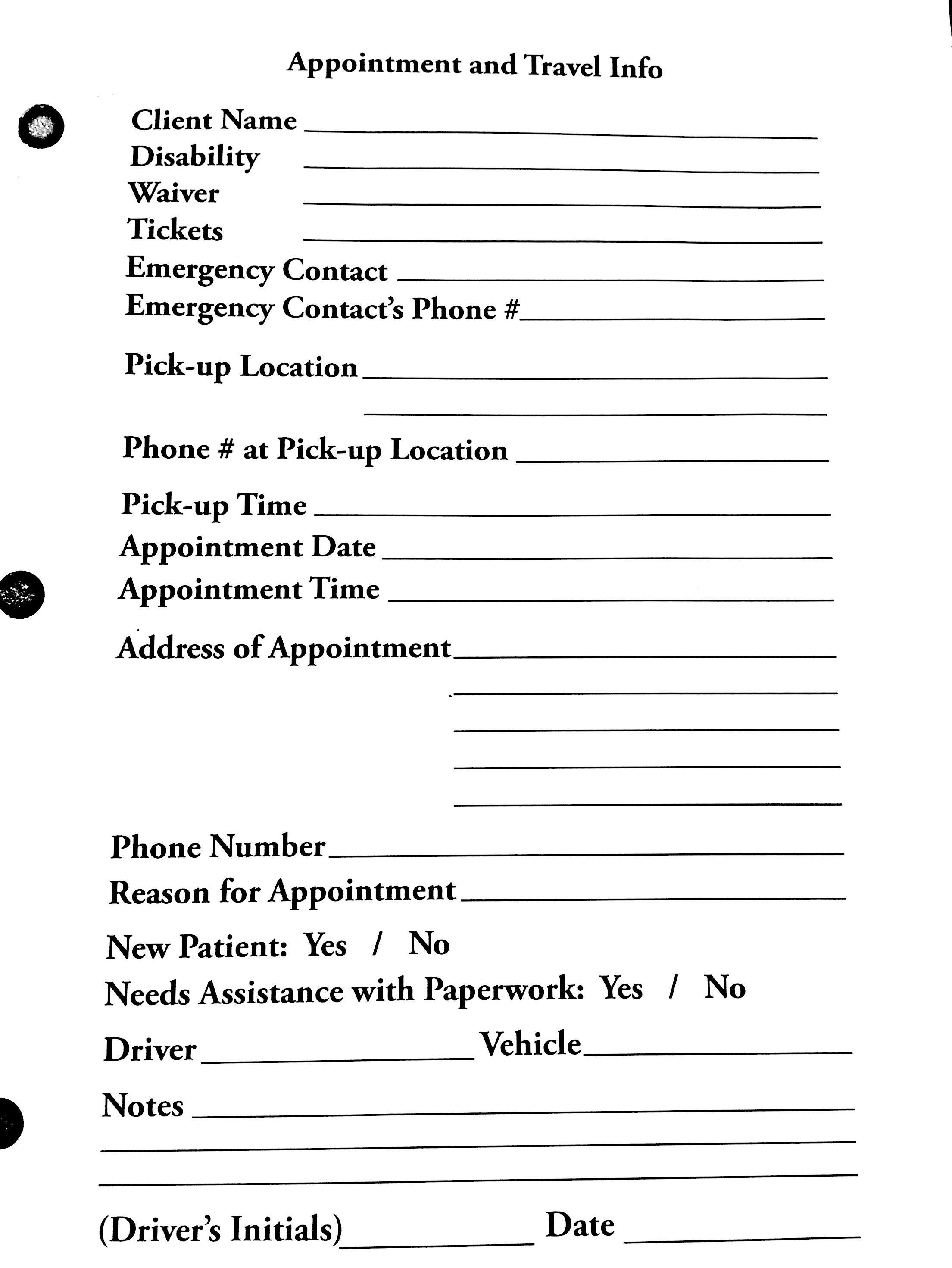
Delete Reservations

Incomplete Data error prevention (i.e. make appropriate fields on the Reservation form required)

These features are subject to change depending on customer input.

# Requirements / User Stories

Below is the form currently in use as of now to create reservations, and the template for our Driver Information Form we are to implement in our project. They want to have the ability to print it out and look similar to this.



## User Story Description

### Adding a New Reservation

Summary: User receives a call and asks questions needed to fill out Appointment Form. User completes and saves form which is then displayed on the calendar.

Description: Customer wants to be able to fill in the form when they receive a call and all information is saved and added to the calendar.

Hours: Total Planned: 30  
Planned this cycle: 20  
Total Actual: 35  
Actual this cycle: 20

Coder: Aymeric Zuurhout, Tyler Espy, Taylor Lucy

Tester: Entire team

Reviewer: Entire team

Status: Finished

Summary: User can look at the calendar and see all reservations that have been planned and the information about each one.

Description: Customer wants to be able to see on any day on the calendar what reservations are planned. From here they can make decisions on whether they can make a new reservation on that day, or what available time there is that day.

Hours: Total Planned: 30  
Planned this cycle: 25  
Total Actual: 30  
Actual this cycle: 25

Coder: Aymeric Zuurhout, Tyler Espy

Tester: Tyler Espy

Reviewer: Entire team

Status: Finished

Summary: User picks data and sees real time car availability.

Description: Customer wants to be able to see on any day on the calendar what reservations are planned. From here they can make decisions on whether they can make a new reservation on that day, or what available time there is that day.

Hours: Total Planned: 25

Planned this Cycle: 25

Total Actual: 25

Actual this Cycle: 25

Coder: Taylor Lucy

Tester: Tyler Espy

Reviewer: Entire team

Status: Finished

# Design Documentation

The purpose of the application is to allow the user to input information concerning their client’s reservation. This application will allow them to move away from the old use of pen, paper, and carbon copies to the modern, and commonly used, digital logging. Doing this will allow them to easily create and find reservations for their clients, instead of searching though file cabinets.

Our process discussion was that our initial design thought was to create a calendar like function that would have access to the form need to create a reservation. However, after a bit of thought, we realized that this needed to be user friendly to people who are either blind or deaf. Since we are dealing with the deaf and blind, features that they believe will help them use our product easier is very important to them. Simple things like drop down menus, to having all reservations next to each other in order, and having reservations color-coded are just some of the simple features that will make their life much easier. We implemented most of these in order to demonstrate to our customer to make sure that is what they wanted.

At the code level, we are currently using .JS (JavaScript) files for the functionality, .HTML (Hypertext Markup Language) files to display text using the web, .CSS (Cascading Style Sheets) files to format and display the elements within the HTML files, and .PHP (Hypertext Preprocessor) which is a server-side scripting language designed for web development. On an organization level we have a form.html and a form.css for the form that the user sees and inputs text. Then we have a calendar.html along with calendar.css and calendar.js to display the calendar and have all the functionality within to work. Together the form.html and calendar.html communicate to make the interaction the user sees. One of the future files that we will create is the basic form look, so the user can see what the print out will look like. Other future files may include the displaying of car and drivers that are available at certain times and dates, but that will be a decision made when that feature comes up.

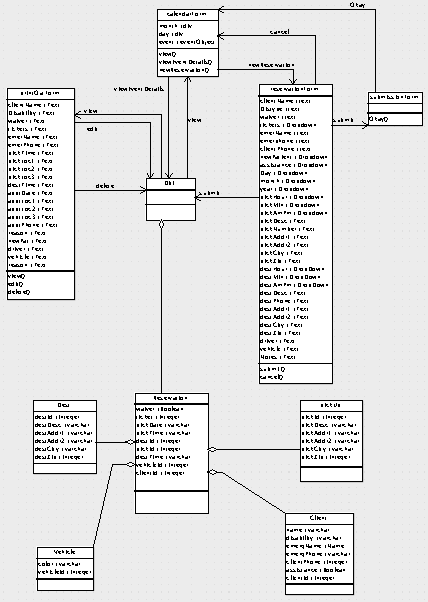
At an overview of how we modeled it, we set up to where the user would select a certain date. From there you can see all current reservations for that date, get a driver, get a vehicle, or if there isn’t current a reservation to create one. Selecting one of the current reservations will allow you to edit that form. However if select to create a new reservation, a blank form will appear and will be used to fill out all the information need for the customer.

Design Guidelines:

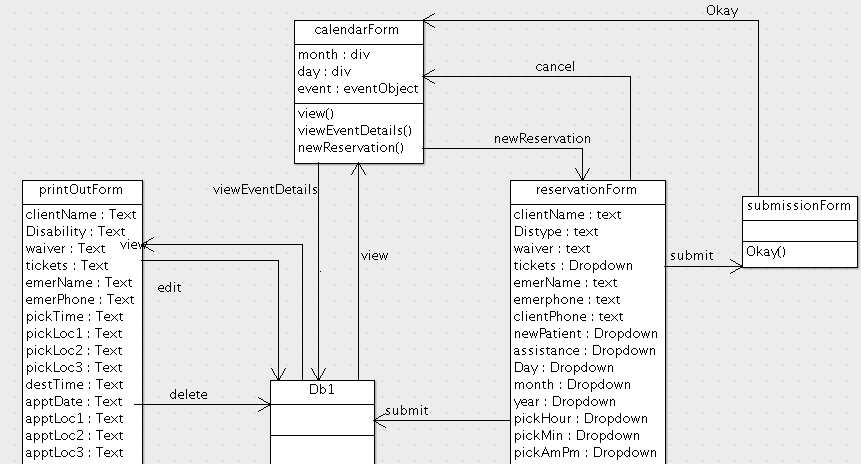
1. Calendar is viewable via link from AIBD webpage
2. Calendar is populated via information stored on AIBD servers
   1. As of now this information is stored on our development server using MySQL
   2. When ready for production our scripts will move required schemas to their database
3. Calendar is linked to two forms:
   1. Appointment Form
   2. Driver Information Form
4. Appointment form is linked to each day on the calendar
   1. Using php, Appointment dates are auto-filled
   2. Upon saving Appointment Information is stored in Reservation Table created in MySQL database
5. Driver Information Form is linked to each event on a given day
   1. Mapped to Reservation Table in MySQL database
   2. Form is auto-filled using data stored in MySQL database
   3. Form is printable
   4. It is NOT however editable

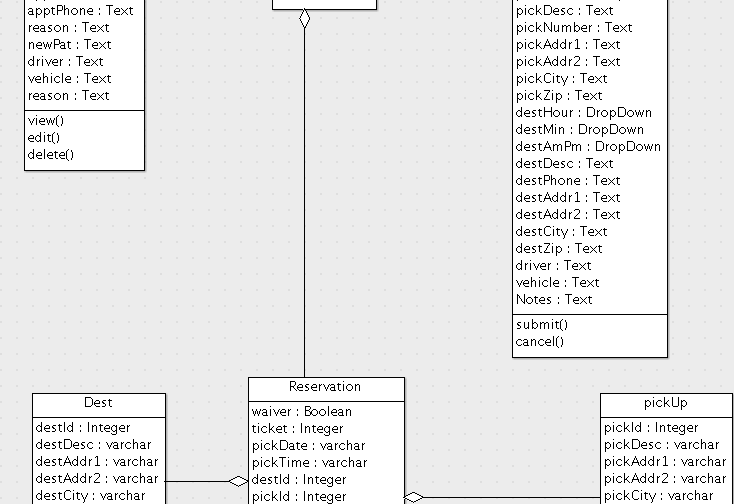
## UML Diagram

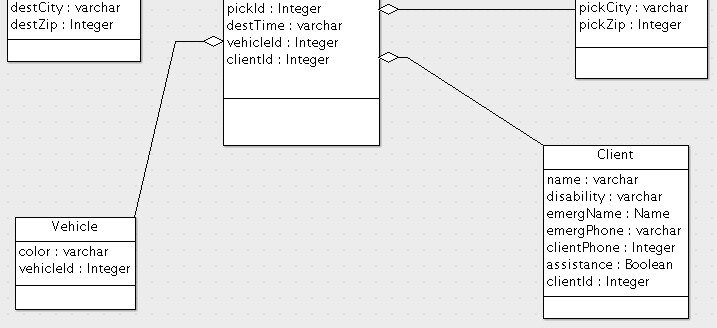
### Structure



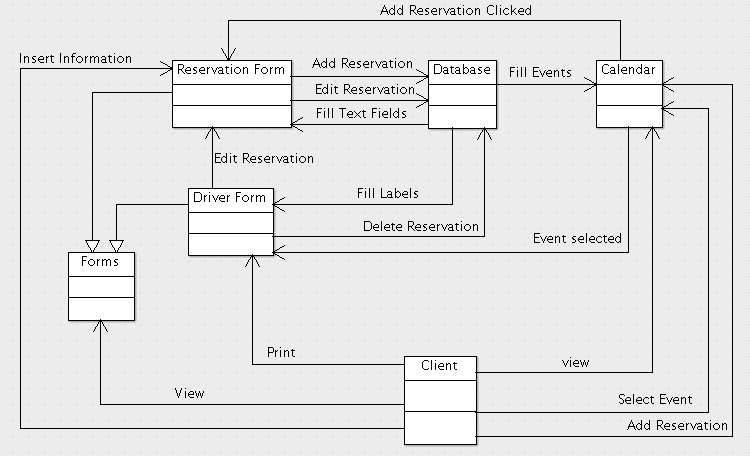
### Close Up Shot







## State Diagram



# Management Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | User Story | Cycle planned for completion | Total planned hours | Status | Actual hours this cycle |
| 1 | User wants to be able to when they receive a call add a new reservation. | 1 | 10 | In Review | 10 |
| 2 | User picks data and sees real time car availability. | 3 | 25 | In Review | 25 |
| 3 | User wants to be able to see all current/available time to make reservation. | 2 | 15 | In Review | 20 |
| 4 | User's information is stored within a database | 2 | 15 | In Review | 15 |
|  |  | **Planned Total** | 65 | **Actual Total** | 70 |

## Task Assignments

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task** | **User Story** | **Planned Date** | **Description** | **Member** | **Hours** | **Status** |
| 1 | 3,4 | 10/10/15 | Calendar Event Population | Aymeric Zuurhout | 7 | Completed On Date |
|  |  |  |  | Tyler Espy | 10 |  |
|  |  |  |  | Taylor Lucy | 7 |  |
| 2 | 1,4 | 11/17/15 | Adding Reservation to DB from Reservation Form | Aymeric Zuurhout | 3 | Completed On Date |
|  |  |  |  | Tyler Espy | 10 |  |
|  |  |  |  | Taylor Lucy | 8 |  |
| 3 | 4 | 10/10/15 | Auto Filling Print Out Form/Reformatting | Aymeric Zuurhout | 6 | Completed On Date |
|  |  |  |  | Tyler Espy | 8 |  |
|  |  |  |  | Taylor Lucy | 6 |  |
| 4 | 2,4 | 10/24/15 | Showing Car Availability | Aymeric Zuurhout | 5 | Completed On Date |
|  |  |  |  | Tyler Espy | 8 |  |
|  |  |  |  | Taylor Lucy | 12 |  |
| 5 | N/A |  | Documentation | Aymeric Zuurhout | 6 | Completed On Date |
|  |  |  |  | Tyler Espy | 3 |  |
|  |  |  |  | Taylor Lucy | 1 |  |
| 6 | N/A |  | Customer Communication | Aymeric Zuurhout | 0 | Completed On Date |
|  |  |  |  | Tyler Espy | 1 |  |
|  |  |  |  | Taylor Lucy | 0 |  |

## Development Schedule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Week** | **Goal #** | **Description** | **Functionality** | **Status** |
| **1**  10/24/15  10/10/15 | 1 | Add Auto Filling Features the Driver Print Out Form | Client clicks on event from calendar, resulting driver form displays event information | Completed |
|  | 2 | Calendar Event Population | Client views calendar, calendar shows all reservations stored in reservation table, on the date of the reservation | Completed |
|  | 3 | Rework Forms to better suit customer | Client is provided needed fields, such as pick up address previously not included | Completed |
| **2**  10/11/15-  10/17/15 | 1 | Implement Reservation Saving abilities | Client inserts information in the Reservation form, after clicking submit the a new reservation with the correct information is added to the table | Completed |
|  | 2 | Rework Forms to better suit customer | Client is provided needed fields, previously not included | Completed |
| **3**  10/18/15-  10/24/15 | 1 | Update Client | Latest material sent to customer for review | Completed |
|  | 2 | Implement Car Availability Features | Client can easily tell when a particular car is in use and when it is available | Completed |
| **4**  10/25/15-  10/31/15 | 1 | Code Freeze | Freeze code changes to focus on review | Completed |
|  | 2 | Documentation | Document Progress and Plan for next Cycle | Completed |

## Planned Code/Freeze

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Goal** | **Functionality** | **Status** |
| calendarDemo.php | implement php event filling queries | Calendar populates with events stored in reservation table of DB1 | Completed |
| insertForm.php | Implement php submit reservation queries | Reservation Form creates Reservation in reservation table of DB1 on submit | Completed |
| printOutForm.php | Implement php autofill queries | Driver Form autofills with desired reservation data | Completed |

# Risk Mitigation

Risks:

1. Overwriting working code
   1. With github’s version control we are able revert back to any previous version
2. Releasing defective code
   1. We believe our acceptance tests (seen below) account for the most pertinent defects
      1. We test our product against these test every week to ensure our product is running efficiently
   2. Although we believe in our acceptance tests we don’t solely rely on these
      1. Our product is available to our customers
      2. We are constantly showing our product to friends to test
3. Releasing unfinished Product
   1. Every product increment from this cycle forward is a fully functioning product, ready for release
4. Implementing unwanted Features:
   1. We attempt to maintain constant contact with our clients
      1. To help our clients, with our next cycle we will increase email frequency

# Test Plan and Test Procedures

Since we are constructing a web-based application, testing our product mostly happens when we functionally do something. So testing would require our product to first off be in place with the able to read and write what we specify it to. Connectivity between all our files must be in place too. All functionalities right now are connected, but we are trouble shooting our database right now. Information is not being saved; therefore, almost all our user stories are put on halt until we get our problem with our database resolved.

# Lessons Learned

Lessons:

1. Do better on documentation
2. Avoid changes in cycle focus, but if necessary keep record of it