

Project group 9

Praveen Kumar Gongidi - 24

Veena Gontu - 25

Simon Moeller - 47

Ganesh Taduri - 59

Introduction

The project for Group 9 is a driving route web application, called Best Route Map Aggregator, or BuRMA for short. There are hundreds of map applications that provide driving directions, each with its own set of specific features. What sets BuRMA apart from the existing driving directions apps is the unique combination of features included. BuRMA utilizes multiple APIs to find the best driving route, will identify the most ideal waypoints along the route based on the user's needs, and includes media and additional features to enhance the user experience.

The BuRMA application utilizes four of the most popular driving route APIs to provide the single most efficient driving route for the user. The user can select whether the route should be optimized for shortest time or shortest distance. While some general testing has shown Google Maps to provide the most efficient route the majority of the time, there are some routes that were found to be more efficient from another map provider.

In addition to finding the most efficient route, BuRMA supports finding waypoints along the route that meet user supplied criteria. The waypoints can be searched for using a free form search, and can be restricted to keep the entire route under a maximum distance or time. If multiple waypoints meet the route requirements, user ratings of the waypoints can be used to pick the best option.

Along with the destination and waypoint optimization, BuRMA will inform the user of the weather at all stops along the route, and provides other media enhanced features. These include street views of destinations, driving videos, and pictures of locations along the route.

A backend database saves key information for each user, from the user's default starting address, to login information, to route optimization preferences. All of this information is also cached at the client end to improve performance, but initial user authentication is always performed against the backend database to avoid authorization tampering.

Project Goal and Objectives

The idea for BuRMA came from a summer trip to Osage Beach at the Lake of the Ozarks. The route to the lake was determined using the vehicle's built in GPS system. This route involved a number of very slow, curvy roads. On the drive home from the lake, Google Maps was used, and provided a route that utilized faster, straighter roads. Analysis after the fact showed that the vehicle's built in GPS route was the shortest possible route by distance, but was not the fastest route. The route provided by Google Maps showed to be the fastest route, but was about 5% greater distance than the vehicle's GPS route. In addition, it was noticed that both routes had long stretches of road that did not have any gas stations or other stops. Many of the gas stations that were found on the route looked very run down, with decent gas stations even further apart.

This difference in routes from different driving direction sources, and the desire to be able to find the best gas station along a route led to the concept for the BuRMA application. BuRMA is able to ensure that the route chosen is the best route from all routes available from the supported mapping APIs. BuRMA is also able to identify the ideal waypoint for the user's needs, whether that is a gas station, a sandwich shop, or a place to buy a gift for the wedding the user is enroute to. BuRMA continues to enhance the experience with real time weather updates for the destinations, and additional rich media.

The BuRMA app should be able to run equally well on a desktop computer for pre-driving planning, to a tablet or smartphone for real time directions. In accordance with some of the API requirements, much of the interfacing between the GUI and the remote APIs will be handled by a backend component.

Project Background

There are numerous mapping and driving directions apps currently on the market. However, a search through the Apple App Store did not find any that currently have the feature set in BuRMA. Here are four driving route apps that are very popular for specific sets of features, and how those compare to BuRMA.

inRoute

inRoute provides an optimized route to go from a starting point to a destination point, with included waypoints. inRoute's algorithms require that all waypoints must be included in the final route, even if this significantly alters the route. BuRMA includes restrictions to prevent potential waypoints from altering the proposed route by more than the user would be comfortable with. In addition, inRoute does not include weather or a rich media experience.

Google Maps

Google Maps is the de facto king of mapping apps, however it lacks a few key features that are included in BuRMA. These include real time weather information, and the ability to choose a better route provided by one of Google Map's competitors.

Mapquest

Mapquest is one of the oldest mapping tools still in active use. Mapquest also provides generally good driving directions, but it is not always the fastest or shortest route available. Mapquest also lacks real time weather information, and has minimal media options.

Waze

Waze uses social networking concepts to provide the most accurate traffic and construction information of any driving directions app. Aside from this social aspect, Waze is a fairly barebones mapping app, and lacks many of the other more advanced and media rich features found in some other apps. Although BuRMA won't allow for direct participation in social networking driving, it will be able to take advantage of the information provided through Waze. This will be accomplished through the Google Maps traffic information, which is fed in part by the social information from Waze.

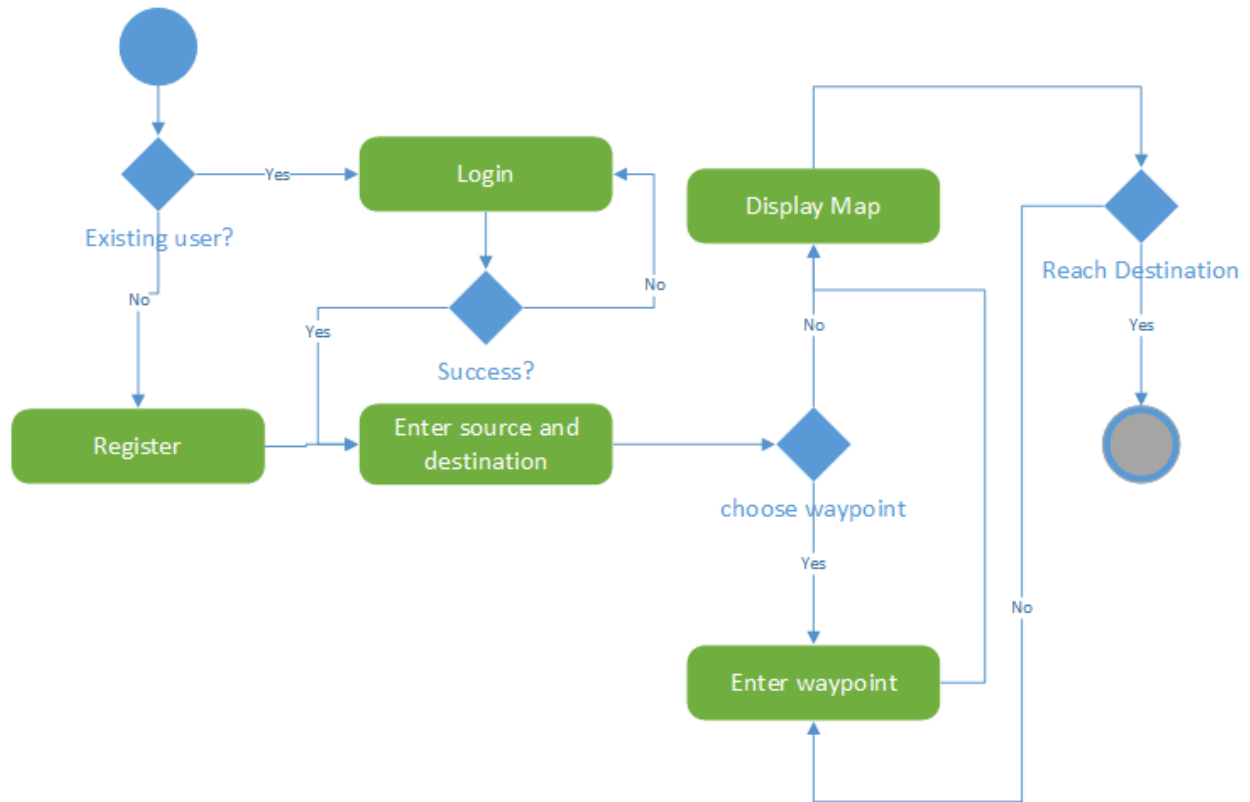
Proposed System

The individual activities are described in more detail under the Project Plan section, but there are some important notes not fully described in the Project Plan. The first is the need for a fast and efficient user experience. The feature set of the BuRMA app is such that it could conceivably all be implemented directly in the front end GUI. However, this would create a feeling of lag for the user as each individual remote API call is processed in serial and displayed. Moving the API calls to the backend allows for the API calls to run concurrently so that the user screen can load more quickly and more completely.

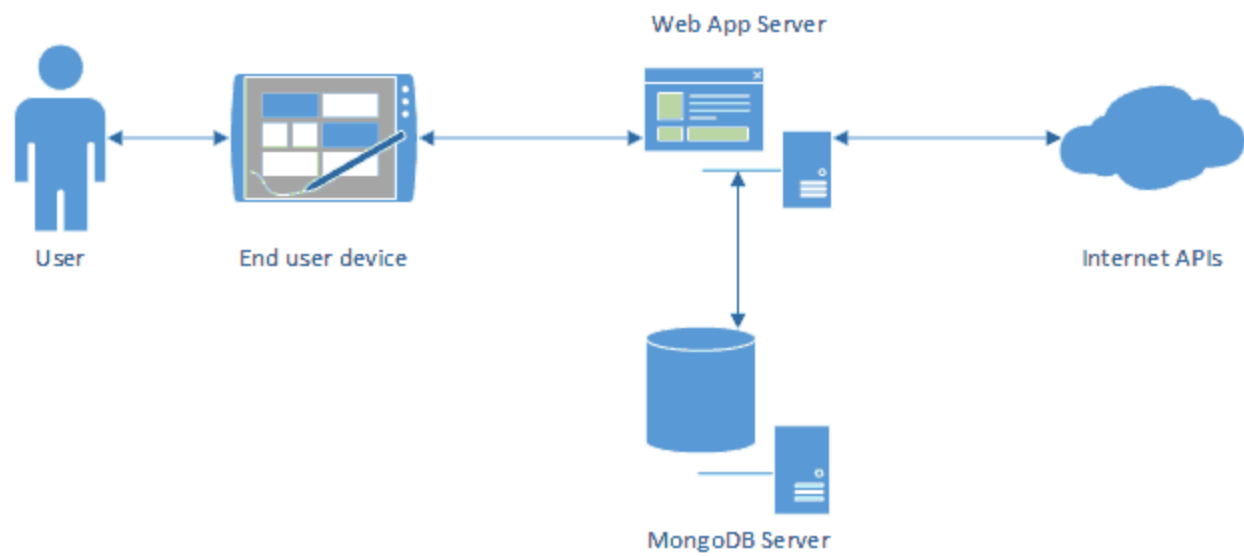
The second decision was to save user information, including a default starting address, in the backend database. Again, this information could easily be saved using only HTML5 localStorage, but doing so would cause all user saved information to be lost if the user were to clear the local caches. By saving the information in the backend database, it allows the information to be

persisted through the clearing of the local cache. However, the local cache/localstorage is still utilized as a means of improving performance, when it is available.

UML Activity Diagram



System Architecture Diagram



Services

Proprietary services developed as part of the BuRMA project are described in the Project Plan section.

External services utilized:

Google Maps API - Primary mapping API, used for routes, waypoint searches, traffic, and construction information. <https://developers.google.com/maps/?hl=en>

Mapquest API - Used for route comparisons. <https://developer.mapquest.com/>

Nokia Maps API - Used for route comparisons. <https://developer.here.com/>

Bing Maps API - Used for route comparisons.
<http://www.microsoft.com/maps/choose-your-bing-maps-API.aspx>

Google Places API - API used to get waypoint ratings. <https://developers.google.com/places/>

Weather Underground API - API for weather information. www.wunderground.com/weather/api

Google Street View API - API for displaying destination images and driving videos.
<https://developers.google.com/maps/documentation/streetview/?hl=en>

Google Translate API - API support for multiple languages.
<https://cloud.google.com/translate/docs>

Web Speech API - API to support speech to text within a web app.
<https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html>

Project Plan

The project is managed through the free KanbanTool.com . The project is broken into four sprints. The team has chosen to use color coding of the tasks to denote which team member will be working each task. Each task includes Use Cases as checklist items. Since the checklist items within each task show on the main Kanban screen as an “X/Y” of X completed from Y total, it is easy to see at a glance what the current use case status is for each item.

Sprint 1

The first sprint is the initial exploratory sprint, and runs from September 7 through September 25. This sprint was focused on better defining the feature set, technologies that would be used, and developing the GUI. The tasks included in sprint 1 are the creation of a Homepage screen, User Registration screen, User Login screen, Navigation screen, and Map screen, and the design and creation of a backend database. In sprint 1 there is no direct interaction between the GUI and the backend database; this interaction will be added in a later sprint.

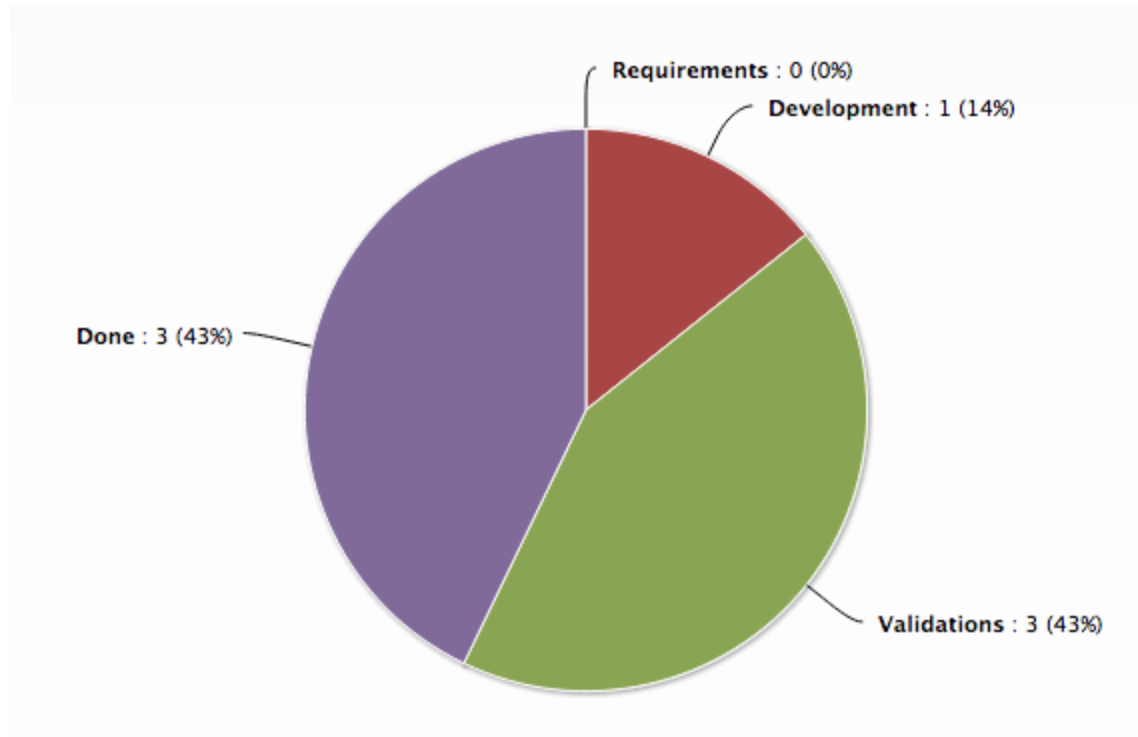
Since sprint 1 is focused primarily on GUI design, the testing for this sprint is comprised of some AngularJS unit tests, and manually implemented use case tests for the basic GUI functionality.

Sprint 1 is estimated at a total of 8 story points. This is anticipated to be a much smaller volume of work than will be accomplished in future sprints. This is due to project startup overhead, with the initial planning of the project. The team did not completely move all stories to a Done state, but the team is working to get these stories completed over the weekend, so that we can get back on track.

Current task screenshot from Kanbantool.com:

	Requirements		Development 4 / 2		Validations		Done
	Gathering <small>+ add task</small>	Analysis <small>+ add task</small>	Queued <small>+ add task</small>	In Progress <small>+ add task</small>	Queued <small>+ add task</small>	In Progress <small>+ add task</small>	
Sprint 1: 07-09 - 25-09 ●				0/2 User database		1/2 Map screen Iteration 1 report 0/4 Navigation screen	0/4 Home Page screen 0/3 User registration screen 0/3 User Login screen

Current status of the sprint 1 tasks:



- Homepage screen use cases (1 point):
 - If a user is detected as being an existing, logged in user via saved login information in the localstorage, the user should be automatically logged in using this information, and redirected to the navigation screen
 - If the user does not have saved login information in the localstorage, the user should be presented with both a Login and a Register button
 - If the user selects the Login button, the user should be redirected to the Login screen
 - If the user selects the Register button, the user should be redirected to the Register screen
 -
- Registration screen use cases (1 point):
 - If either the name, login name, email, or password are left blank, the app should alert the user and stay on the Registration screen
 - If the login name the user selects already exists in the database, the user should be alerted to select a different login name
 - If all mandatory fields are filled in, and the login name does not already exist, the user's registration information should be saved in localstorage and the user should be redirected to the navigation screen
- Login screen use cases (1 point):
 - If the form is submitted without filling in both the username and the password, the user should be alerted and the login screen should be presented again

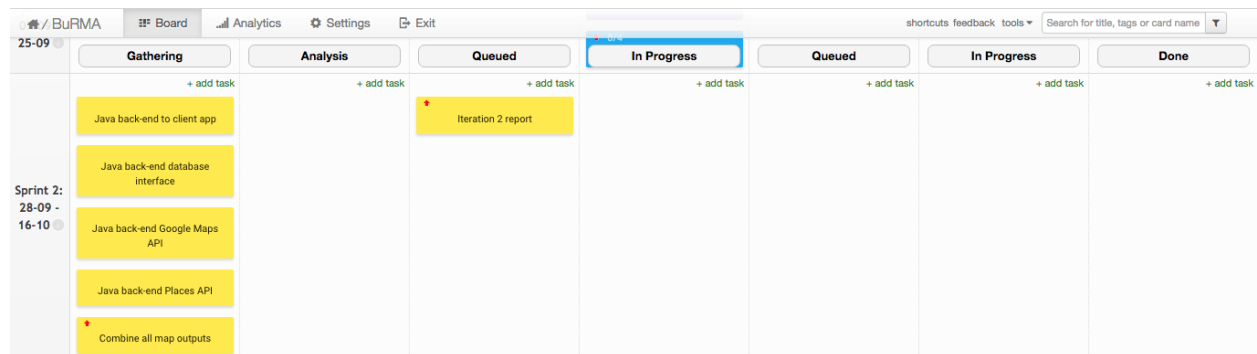
- If the username and/or password do not match a registered user in localStorage, a login failure alert should be displayed, and the user should be prompted to login again
- If the username and password match a user in localStorage, the user should be logged on and automatically redirected to the Navigation screen
- Navigation screen use cases (1 point):
 - If the user does not enter both a start point and a destination point, an alert should be displayed and the user should be re-prompted for the information
 - If the user enters a start and a destination point and clicks Navigate, the Sort By preference and the start and end points should be saved in localStorage, and the user should be redirected to the Map screen
 - If the user includes a waypoint search term, the term should also be saved in the localStorage when redirected to the Map screen
- Map screen use cases (1 point):
 - The map should be Google Maps and should support the standard Google Maps features like zoom, drag, select items, street view, etc.
 - The start point and end point should be loaded from localStorage, and the route should be displayed on the map
 - If waypoint information is also saved in the localStorage, all matching waypoints should be overlaid on the route map
- Backend database use cases (1 point):
 - The database should provide a storage location for all user objects (login name, name, password, email, phone #, and Location for default start location)
 - The database should provide a storage location for location objects (address1, address2, city, region, country, postal code, GPS lat, GPS long)

Sprint 2

Sprint 2 is focused on backend functionality, and runs from September 28 through October 16. In this sprint the team is planning on developing the Java backend interfaces, including API interfaces, interfaces for communicating with the GUI front end, and database interfaces. The tasks planned for sprint 2 are a Java backend piece to provide communications with the GUI front end, a piece to provide an interface into the MongoDB, and API support for Google Maps and Google Places. In addition, the Java backend will be written to select best path information from multiple mapping APIs, even though only one mapping API will be implemented at this point.

Unit testing of the code changes in sprint 2 will be accomplished using JUnit. In following best practices, the JUnit unit tests will be developed prior to the development of each unit of code. This will help ensure continuous stability and correct functionality of the code. Use case testing will be done manually, and is driven by the use cases saved in the Kanban Tool. These use cases will be developed as part of the initial sprint planning for sprint 2.

Sprint 2 is planned to be entirely backend related work. There are currently 15 points of work planned for sprint 2. The first user story is to provide an interface between the Java backend and the AngularJS frontend. The second user story provides an interface between the Java backend and the MongoDB database. The third user story integrates the Google Maps API into the Java backend, to allow for Google Maps queries from the backend. The fourth user story integrates the Google Places API into the Java backend, to allow for queries against the Places API for business ratings information. The fifth and final user story for sprint 2 is to develop the backend Java code that can take as input an arbitrary number of route information inputs, and can pick the best route from all inputs, based on the user's preference.



Sprint 3

Sprint 3 is focused on waypoint functionality and weather information support, and runs from October 19 through November 6. The waypoint implementation is broken into four separate user stories. Weather information support has been broken into three separate user stories. All seven of these user stories are implemented in the backend Java code.

Unit testing of the code changes in sprint 3 will be accomplished using JUnit. In following best practices, the JUnit unit tests will be developed prior to the development of each unit of code. This will help ensure continuous stability and correct functionality of the code. Use case testing will be done manually, and is driven by the use cases saved in the Kanban Tool. These use cases will be developed as part of the initial sprint planning for sprint 3.

Sprint 3 is planned to be entirely backend related work. There are currently 15 points of work planned for sprint 3. The first waypoint user story is to add basic search capability to the backend. The second waypoint user story is to include an algorithm for selecting the single best waypoint based on deviation from the existing route. The third waypoint user story is to include business rating information in the determination of best way point. The fourth waypoint user story is to include maximum route deviation restrictions to the selection criteria. The weather information related user stories are broken out into a user story for providing weather information for each of the three potential stops on the route, the start, waypoint, and destination.

	Gathering	Analysis	Queued	In Progress	Queued	In Progress	Done
	+ add task	+ add task	+ add task	+ add task	+ add task	+ add task	+ add task
Sprint 3: 19-10 - 06-11	Destination weather		Iteration 3 report				
	Starting point weather						
	Search for waypoints						
	Find best route way point						
	Way point weather						
	Highest rated way point						
	Max way point deviation from route						

Sprint 4

Sprint 4 is the final sprint in this project. As such, this sprint will be used to add in additional features, as much as project time allows. Some higher priority features for inclusion in sprint 4 include street view of the destination(s), support for additional map route APIs, and inclusion of road hazard information. Some lower priority features that will be included if time allows are full driving video, speech control, multi-language support, and the inclusion of scenic stops along the route.

Unit testing of the code changes in sprint 4 will be accomplished using JUnit for backend Java code, and Karma and Jasmine for front end AngularJS code. In following best practices, the unit tests will be developed prior to the development of each unit of code. This will help ensure continuous stability and correct functionality of the code. Use case testing will be done manually, and is driven by the use cases saved in the Kanban Tool. These use cases will be developed as part of the initial sprint planning for sprint 4.

	Gathering	Analysis	Queued	In Progress	Queued	In Progress	Done
	+ add task	+ add task	+ add task	+ add task	+ add task	+ add task	+ add task
Sprint 4: 09-11 - 04-12	Bing maps support		Iteration 4 report				
	Road hazards						
	View destination						
	Nokia maps support						
	Mapquest maps support						
	Full driving video						
	Multi-language support						
	Scenic stops on route						
	Speech control						

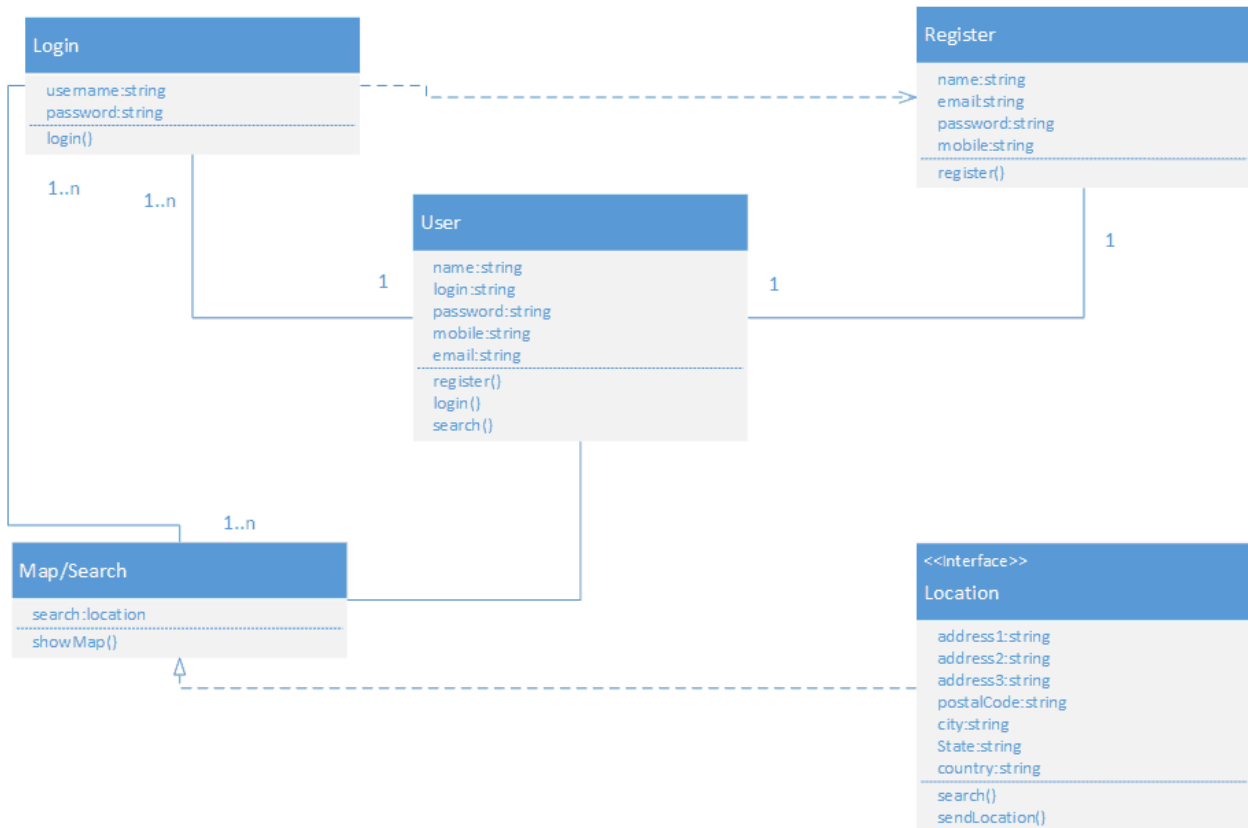
Project Information

The project timeline is specified to run from September 7, 2015, through December 4, 2015. This time is broken up into four sprints, the first three of which are three week sprints, and the final sprint is a four week sprint.

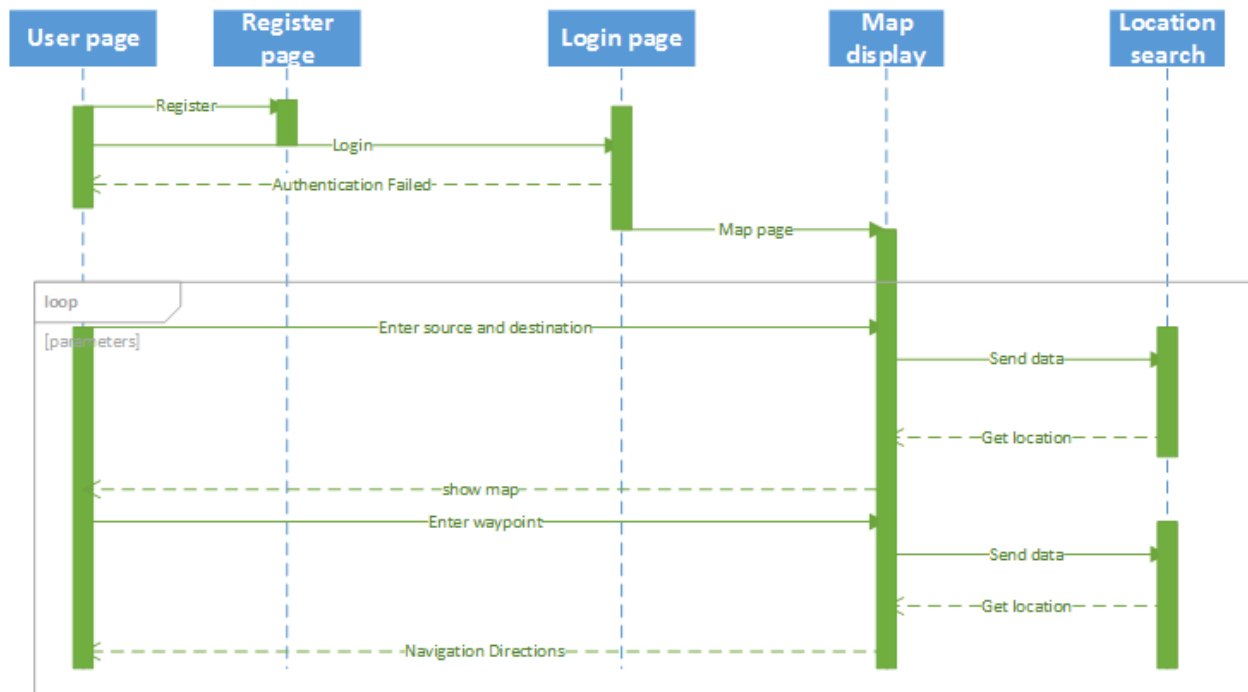
The team is comprised of Praveen Kumar Gongidi, class id 24, Veena Gontu, class id 25, Simon Moeller, class id 47, and Ganesh Taduri, class id 59. The team has chosen to optimize our efforts by allowing each team member to work on the tasks that they would be most efficient at. Due to the required tasks not aligning perfectly with all team member's existing strengths, there is some need for team members to work on some tasks that they are not as efficient at. This helps the team better achieve the overall needed velocity over the course of the project. In general, Simon and Veena are working backend tasks, and in general Ganesh and Praveen are working front end tasks.

First Increment Report

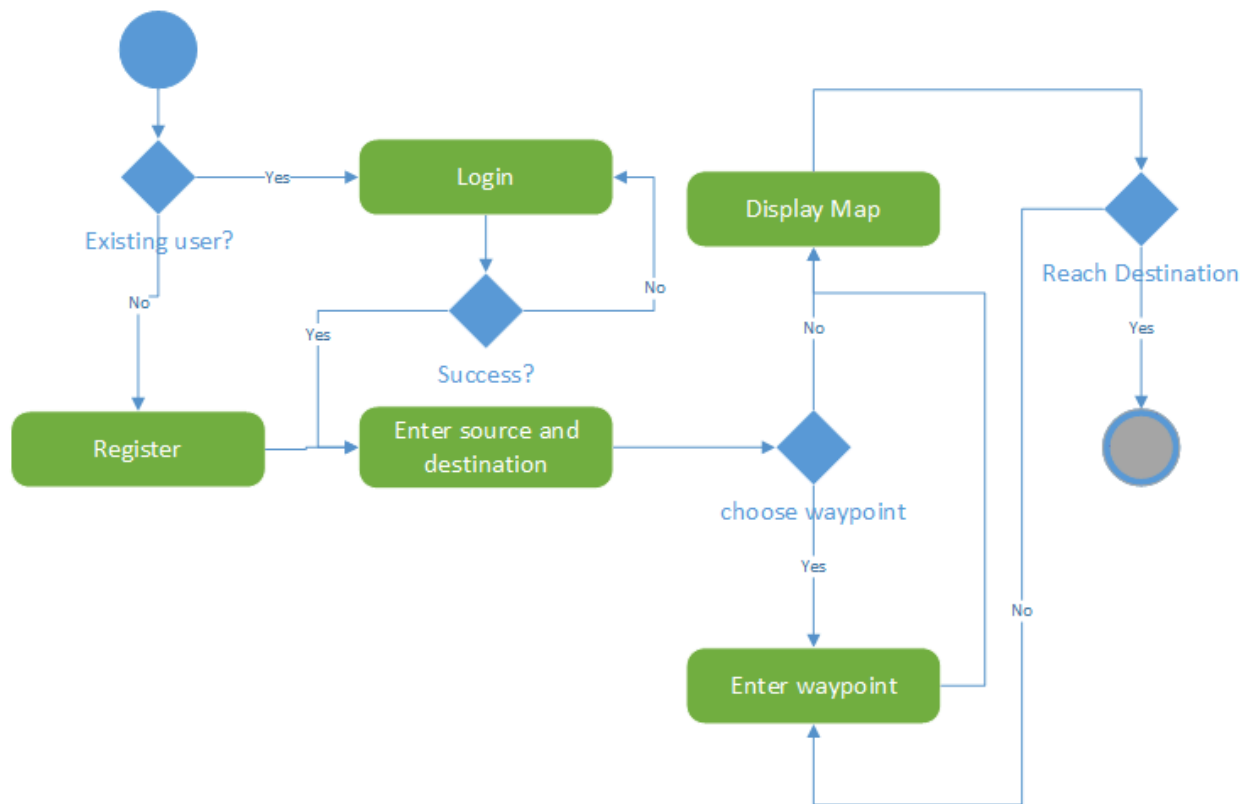
Class diagram



Sequence diagram



State diagram



Wireframes

Intro screen

WELCOME TO BURMA
BURMA FULLFORM




Signup screen



BURMA

[Sign in](#)

Sign Up

First Name*	<input type="text"/>
Last Name*	<input type="text"/>
DOB*	<input type="text" value="MM/DD/YYYY"/> 
Mobile Number*	<input type="text"/> <input type="text"/>
User Name*	<input type="text"/>
Password*	<input type="password"/>
Re-enter Password*	<input type="password"/>
Email	<input type="text" value="For Password recovery"/>

[Sign up](#)

Login screen

BURMA

Log In

User Name:

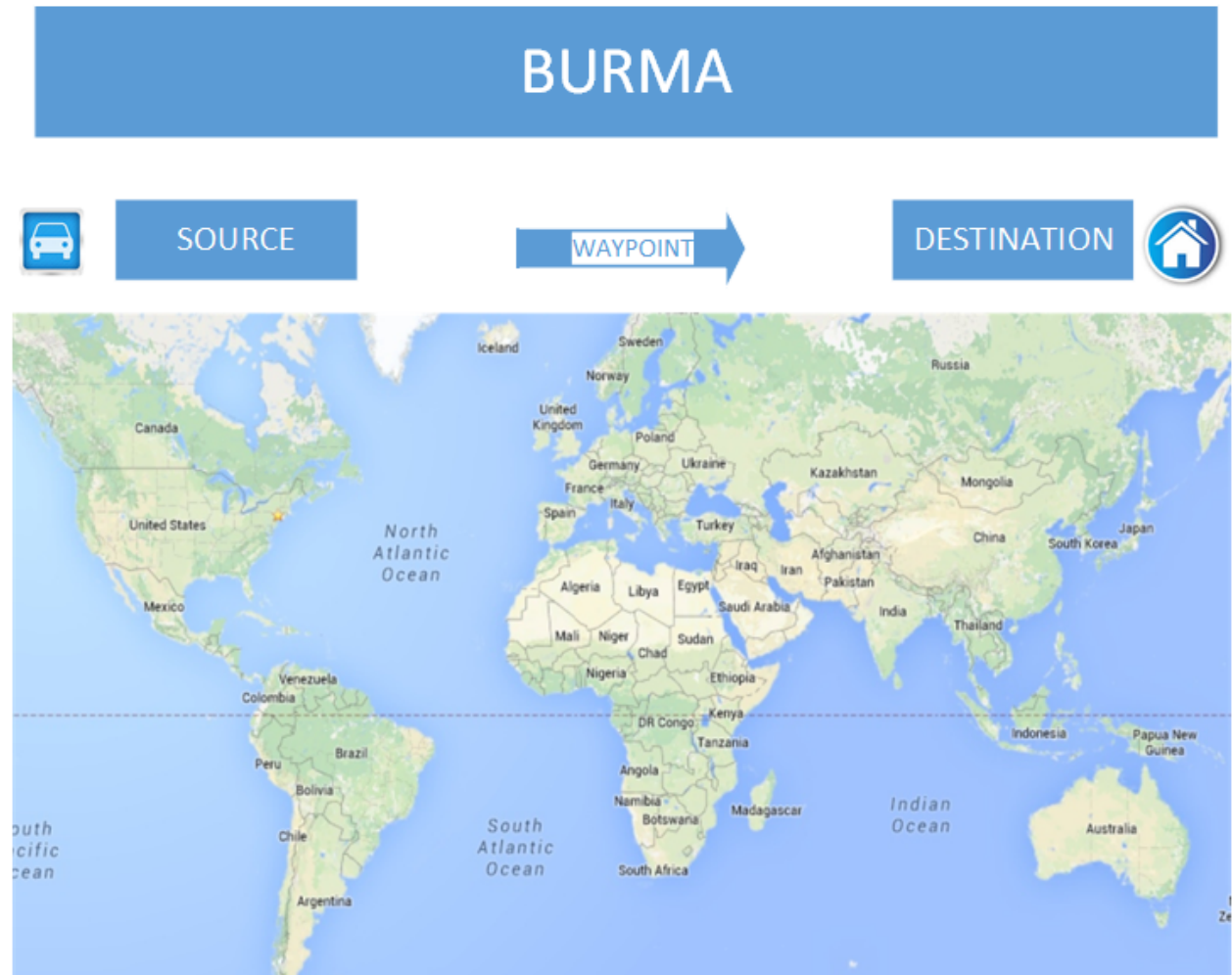
Password:

Log In

[Forgot Password?](#)

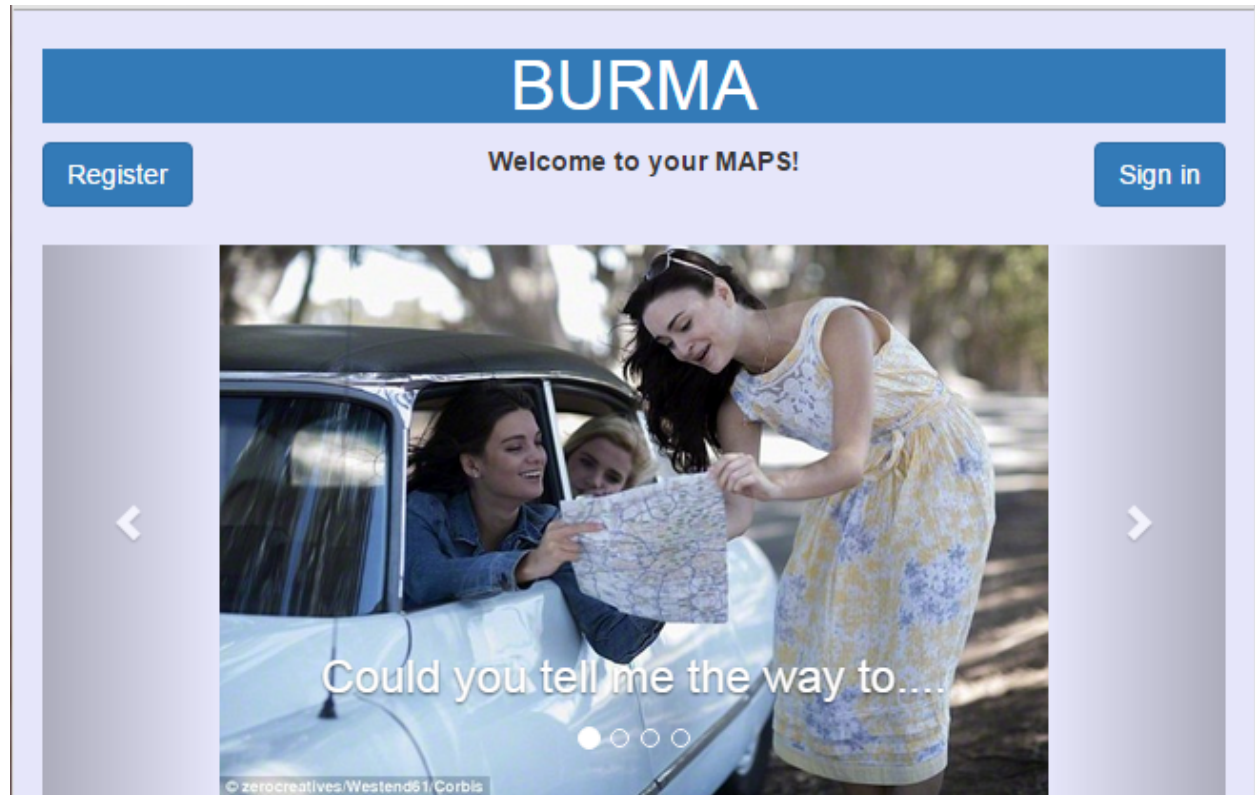


Map screen



Design of Client using Brackets

First page:



Registration page:

BURMA

Welcome to Your Maps!

[Sign in](#)

Sign Up

First Name:

Last Name:

DOB:

☐ Male ☐ Female

Mobile Number:

User Name:

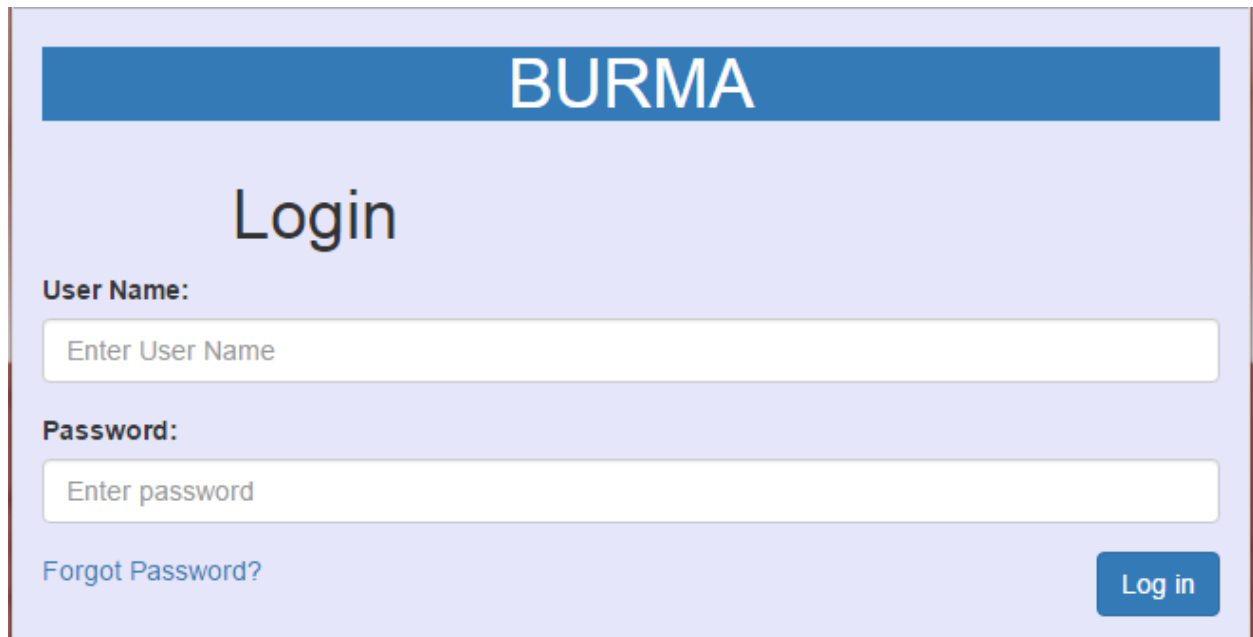
Password:

Re- enter Password:

Email:

[Sign Up](#)

Login page:



The image shows a login page for a system named 'BURMA'. At the top, there is a dark blue header bar with the word 'BURMA' in white, bold, uppercase letters. Below the header, the word 'Login' is centered in a large, black, sans-serif font. Underneath 'Login', there are two labels: 'User Name:' and 'Password:', both in a bold, black, sans-serif font. Each label is followed by a white text input field with a thin grey border. The first input field contains the placeholder text 'Enter User Name' in a light grey font. The second input field contains the placeholder text 'Enter password' in a light grey font. At the bottom left of the form area, there is a link that says 'Forgot Password?' in a blue, sans-serif font. At the bottom right, there is a dark blue rectangular button with the text 'Log in' in white, sans-serif font.

BURMA

Login

User Name:

Password:

[Forgot Password?](#)

Log in

Maps page:

BURMA

Hello Simon!

From

To

Waypoint

Map data ©2015 Google, INEGI Terms of Use

Bibliography

Moscaritolo, Angela. "Google Maps Gets Waze Real-Time Traffic Reports." *PC Magazine*. August 20, 2013: <http://www.pcmag.com/article2/0,2817,2423328,00.asp>

Google Maps API. <https://developers.google.com/maps/?hl=en>

Mapquest API. <https://developer.mapquest.com/>

Nokia Maps API. <https://developer.here.com/>

Bing Maps API. <http://www.microsoft.com/maps/choose-your-bing-maps-API.aspx>

Google Places API. <https://developers.google.com/places/>

Weather Underground API. www.wunderground.com/weather/api

Google Street View API. <https://developers.google.com/maps/documentation/streetview/?hl=en>

Google Translate API. <https://cloud.google.com/translate/docs>

Web Speech API. <https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html>