REPORT

on

***“*Event mate*”***

**An application to connect event goers with each other.**

Submitted in partial fulfilment of the requirement of University of Mumbai

For the Degree of

**Bachelor of Engineering in Computer Engineering**

Submitted by

**Mr. Amit K. Solanki**

**Ms. Sneha Naik**

**Ms. Vinita L. Poptani**

Under the guidance of

**Prof. Prashant Misal**

**Department of Computer Engineering**

**Watumull Institute of Electronics Engineering and Computer Technology** Worli, Mumbai – 400 013

(University of Mumbai)

2013-2014

WATUMULL COLLEGE OF ELECTRONICS ENGINEERING AND COMPUTER TECHNOLOGY, WORLI, MUMBAI

CERTIFICATE

*This is to certify that*

**Mr. Amit K. Solanki**

**Ms. Sneha Naik**

**Ms. Vinita L. Poptani**

*Have satisfactorily completed the requirements of the* **PROJECT**

***“*Event mate*”***

*As prescribed by the* **University of Mumbai.**

**Prof. Prashant Misal**

**(Project Guide)**

**Mr. Prashant Misal Mr. Nilesh Mehta Ms. Sandhya Desai**

Project Coordinator Head of the Dept. Principal

**ACKNOWLEDGEMENT**

We thank our guide Prof. Prashant Misal, for immense support and helpful insight in the project. He has motivated us and guided us in the proper direction. We thank him for spending his valuable time in correcting n modelling our project.

We are thankful to HOD, principal and management for providing us with proper infrastructure and facilities to aid our project.

We also thank all the lab attends for being helpful and co-operating with us throughout the making of this project.

Last but not the least; we would like to thank our friends and family for giving us proper feedback on our project.

Without the valuable help of all the above mentioned people, our project would not have been possible.

A big thank you to all of them.

Presented by-

Mr. Amit Solanki

Ms. Sneha Naik

Ms. Vinita Poptani

**TABLE OF CONTENTS**

**CHAPTER TITLE**

**1. INTRODUCTION**

**2. MAJOR FUNCTIONALITY**

**3. SYSTEM DEPLOYEMENT DIAGRAM**

**4. KNOWLEDGE BASE REQUIREMENT**

**5. LITERATURE REVIEW**

**6. REFERENCE**

**LIST OF FIGURES**

1. **SYSTEM DEPLOYMENT DIAGRAM**
2. **MVC ARCHITECTURE**
3. **WINDOWS PHONE ARECHITECTURE**
4. **USE CASE DIAGRAM**
5. **GENERAL INTRODUCTION**

As the online community is increasing, the ways of socializing has changed a lot. With this, there has been a change in the way people meet and socialize. The world has become smaller than it ever was.

But the real way of socializing and networking is to meet in person. There aren’t many solutions to bring the online community to meet offline with ease, or bringing like-minded people to meet in real life. Although there are some web applications but they are jack of all trades and not concentrating on bringing the real change. With our application we hope to bring people with similar minds and liking to connect easily.

The most efficient method of getting people to meet each other, is by bringing them together at the places they would go. One wouldn’t want to socialize with people they don’t share common interests with. It has always been hard to find a group of people with common interest through online medium, except for online forums and groups. We try to solve this problem and want to create a new way of finding like-minded people, so that they can communicate in a much better way.

**2. MAJOR FUNCTIONALITY**

**1. Searching for events:** People would list any new event that is going to happen at a place nearby, these events will be searchable. They can also invite their friends to visit the events.

**2. Finding people going to the event**: once someone would like to visit a place the can go through the people who would like to go to the same place.

**3. In app messaging system:** when two people would like to meet, they would definitely like to have a chat on some topic, messaging system would fulfill this need.

**4. Managing the profile:** Each person will have a profile with their interest and topics listed. They should be able to manage their profile within the application.

**5. Ease of signups:** People would be able to login through their facebook accounts.

**6. Ease of finding events:** people should be able to find events easily using the map, with pinups at places where events are occurring.

# 3. SYSTEM DEPLOYEMENT DIAGRAM

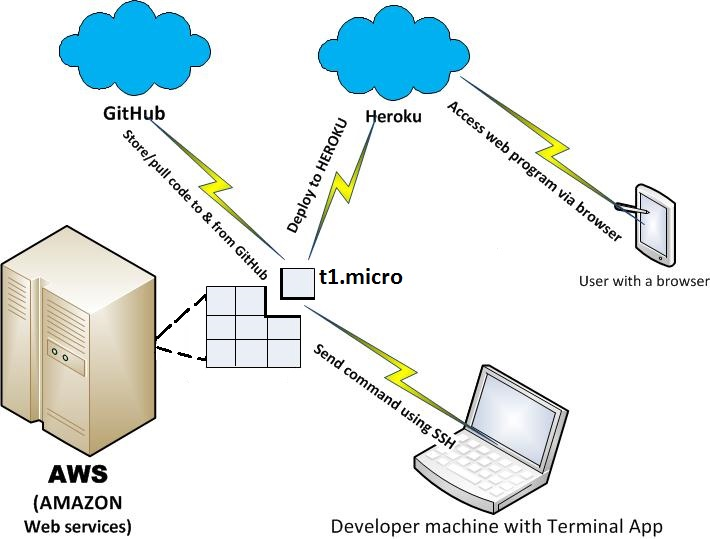


Fig.1 System Deployment Diagram

The architecture of the project is simple:

1. The user with the help of any given system, can access the application.

The system can be -mobile and PC based.

1. The application deployed on heroku.
2. The code is version controlled and managed using git.
3. Heroku itself hosts the application on AWS(t1.micro)
4. We can manage the application by sending commands using shell
5. The database is postgresql
6. User can be anyone.
7. **KNOWLEDGE BASE REQUIREMENT**

**4.1 MVC Architecture:**

Ruby on Rails MVC framework:

The **M**odel **V**iew **C**ontroller principle divides the work of an application into three separate but closely cooperative subsystems.

**Model (ActiveRecord ):**

Maintains the relationship between Object and Database and handles validation, association, transactions, and more.

This subsystem is implemented in ActiveRecord library which provides an interface and binding between the tables in a relational database and the Ruby program code that manipulates database records. Ruby method names are automatically generated from the field names of database tables, and so on.

**View ( ActionView ):**

A presentation of data in a particular format, triggered by a controller's decision to present the data. They are script based templating systems like JSP, ASP, PHP and very easy to integrate with AJAX technology.

This subsystem is implemented in ActionView library which is an Embedded Ruby (ERb) based system for defining presentation templates for data presentation. Every Web connection to a Rails application results in the displaying of a view.

**Controller ( ActionController ):**

The facility within the application that directs traffic, on the one hand querying the models for specific data, and on the other hand organizing that data (searching, sorting, massaging it) into a form that fits the needs of a given view.

This subsystem is implemented in ActionController which is a data broker sitting between ActiveRecord (the database interface) and ActionView (the presentation engine).

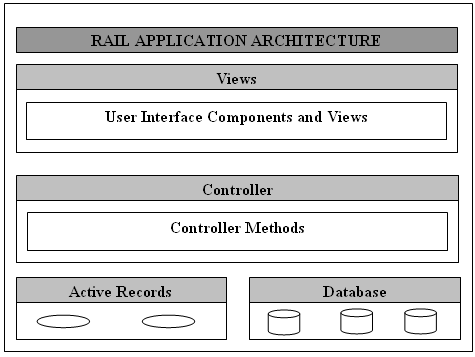


Fig.2 Model-View-Controller Architecture

### Advantages of an MVC-Based Web Application:

The Rails MVC framework offers the following advantages:

* It makes it easier to manage complexity by dividing an application into the model, the view, and the controller.
* It does not use view state or server-based forms. This makes the MVC framework ideal for developers who want full control over the behavior of an application.
* It uses a Front Controller pattern that processes Web application requests through a single controller. This enables you to design an application that supports a rich routing infrastructure.
* It provides better support for test-driven development (TDD) or behavior driven development.
* It works well for Web applications that are supported by large teams of developers and Web designers who need a high degree of control over the application behavior.

**Infrastructure Requirements:**

1. Ubuntu operating system

2. Desktop

3. Heroku for deployment

4. amazon s3 for images hosting

5. facebook app for fb login

**Skillset Requirements:**

1. Ruby, ruby on rails

2. Backbone.js, Javascript

3. AJAX

4. Rails Web Application Development

4.3 Backbone.js

When working on a web application that involves a lot of JavaScript, one of the first things you learn is to stop tying your data to the DOM. It's all too easy to create JavaScript applications that end up as tangled piles of jQuery selectors and callbacks, all trying frantically to keep data in sync between the HTML UI, your JavaScript logic, and the database on your server. For rich client-side applications, a more structured approach is often helpful.

With Backbone, you represent your data as [Models](http://backbonejs.org/#Model), which can be created, validated, destroyed, and saved to the server. Whenever a UI action causes an attribute of a model to change, the model triggers a *"change"* event; all the [Views](http://backbonejs.org/#View) that display the model's state can be notified of the change, so that they are able to respond accordingly, re-rendering themselves with the new information. In a finished Backbone app, you don't have to write the glue code that looks into the DOM to find an element with a specific *id*, and update the HTML manually — when the model changes, the views simply update themselves.

Philosophically, Backbone is an attempt to discover the minimal set of data-structuring (models and collections) and user interface (views and URLs) primitives that are generally useful when building web applications with JavaScript. In an ecosystem where overarching, decides-everything-for-you frameworks are commonplace, and many libraries require your site to be reorganized to suit their look, feel, and default behavior — Backbone should continue to be a tool that gives you the *freedom* to design the full experience of your web application.

4.4 Facebook integration

## Secure Registration

Facebook Login has several security features to protect people's information, to let people control what they share and to let developers safely request access to this information.

#### Secure Authorization

We use the [OAuth2.0](http://oauth.net/2/) open protocol for confirming a person's identity (authentication) and giving them control over right of access to their information (authorization). [Read more about the protocol and its specification](http://oauth.net/2/).

## The Login Process

Facebook offers several login flows for different devices and projects. Some are easily implemented using our [official SDKs](https://developers.facebook.com/docs/sdks) while others will require some additional code. Each flow follows the same general steps:

* Determine whether someone is already logged in.
* If they aren't logged in, prompt them to do so (with the Login dialog).
* Exchange secure codes to confirm identity.
* Generate an access token.

## Access Tokens

An access token is a random string that gives an app temporary and secure access to Facebook APIs. An access token can be created on behalf of a person, a Facebook Page or an app. The token is generated in the last step of the login flow. Facebook SDKs handle the generation and storage of tokens automatically. Apps using other methods will need to follow the login flow to create tokens.

The token stores information about permissions that have been granted as well as information about when the token will expire and which app generated it. To maintain information security, almost all API calls at Facebook need to have an access token passed in the parameters of the request.

We will be using Oauth 2.0 for facebook login.

4.5 Google maps integration

Google maps provide and api access key to use thier maps and thier databse of places. we will be using it to show map of where the event is occuring and will use their geo coding api for encoding the places, so as to search efficiently in the database of places.

1. **REFERENCE**

api.rubyonrails.org

backbonejs.org

developer.facebook.com

<https://developers.google.com/maps/documentation/javascript/>