*Kiran Patel*

*travelMe*

A blogging application allowing backpackers to blog their travels across the world

Table of Contents

[Summary 3](#_Toc427641921)

[User Stories 3](#_Toc427641922)

[Business Objects 3](#_Toc427641923)

[Object Relationships 3](#_Toc427641924)

[Tools & Technologies 3](#_Toc427641925)

[Architecture 4](#_Toc427641926)

[Methodology 4](#_Toc427641927)

# Summary

TRAVELME is blogging application designed specifically for backpackers who are travelling the world. The application will provide the functionality to post blogs along with pictures about there current travels. Journeys will be organised into ‘Trips’ where each user may add as many trips as they wish. Within each trip there will be many posts. The web application will provide a way to store these posts and also view them for later viewing. Functionality to share with friends will also be considered. TRAVELME will use location data to track a person’s travels over a period of time, this will be later displayed through a mapping api such as Google Maps.

# User Stories

Signs in

Registers

Adds Trip

Edits Trip

Deletes Trip

Adds Post

Edits Post

Deletes Post

Add Photo

Remove Photo [Whilst adding]

Remove Photo

Share Trip

Share Blog

Set Location

View Locations

# Business Objects

* User – Entity represents a user using the application
* Trip – Entity represents a trip which pertains to a user
* Post – Entity represents a post pertaining to a post will contain text and potentially images
* Media – Contains a media image where many may relate to one post

# Object Relationships

* User(1) : Trip(0:\*)
* Trip(1) : Post(0:\*)
* Post(1): Media(0:\*)

# Tools & Technologies

ASP.NET MVC

C#

Visual Studio 2013

SQL Server 2012

nHibernate

nUnit

Moq

StyleCop

# Architecture

TRAVELME will use an n-tier architecture along with the Model-View Controller design pattern. The main layers will be the presentation, business logic and data access layer. These layers will however be broken down further using the Repository Pattern and with the use of services. The application will also make use of dependency injection. The decision was made to develop an n-tier architecture so that the application could be made as maintainable as possible and also to make it testable.

# Methodology