*Kiran Patel*

*travelMe*

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

Table of Contents

[Summary 3](#_Toc427727675)

[User Stories 3](#_Toc427727676)

[Business Objects 3](#_Toc427727677)

[User 3](#_Toc427727678)

[Trip 4](#_Toc427727679)

[Post 4](#_Toc427727680)

[Media 4](#_Toc427727681)

[Object Relationships 4](#_Toc427727682)

[Tools & Technologies 4](#_Toc427727683)

[Architecture 4](#_Toc427727684)

[Methodology 5](#_Toc427727685)

[Documentation 5](#_Toc427727686)

# 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

## User

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Notes** |
| ID | Uniqueidentifier | PRIMARY KEY NOT NULL |
| FirstName | Nvarchar(100) | NOT NULL |
| LastName | Nvarchar(100) |  |
| DateOfBirth | DATETIME |  |
| ProfilePicture | Varbinary |  |

## Trip

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Notes** |
| ID | Uniqueidentifier | PRIMARY KEY NOT NULL |
| TripName | Nvarchar(20) | NOT NULL |
| TripDescription | Nvarchar(50) |  |
| UserID | Uniqueidentifier | FOREIGN KEY CONSTRAINT TO USER |

## Post

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Notes** |
| ID | Uniqueidentifier | PRIMARY KEY NOT NULL |
| Post | Nvarchar(255) |  |
| TripID | Uniqueidentifier | FOREIGN KEY CONSTRAINT TO TRIP |

## Media

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Notes** |
| ID | Uniqueidentifier | PRIMARY KEY NOT NULL |
| Media | Nvarchar(255) |  |
| PostID | Uniqueidentifier | FOREIGN KEY CONSTRAINT TO POST |

Uniqueidentifier is used as a primary key data type instead of INT, this is because the entities will use GUIDs. This decision was made to ensure that each record was unique.

The Decision was made to use nvarchar instead of varchar because nvarchar can store unicode data whilst varchar is restricted.

# Object Relationships

* Each User (1:1) has no or many Trips (0:\*)
* Each Trip (1:1) has no or many Posts (0:\*)
* Each Post (1:1) has no or many 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.

Layers:

* SQL Server
  + Stores data with a table for each entity
* Data Access
  + Interacts with the database to execute CRUD functions. This is done through Object-Relational Mapping (nHibernate)
* Repository
  + Interacts with the data access layer where there is a Generic Repository and all entities will use this repository to access the database.
* Service Layer
  + Contains all the business logic in order to make the controller thin. Service layer will interact with the repository.
* Controller <-> Model
  + Controller handles user input and output, Controller will have minimal logic as most will be in service.
  + Models are the models of the application to represent entities.
* View
  + Display cshtml to the user

# Methodology

The project will be developed using a Test Driven Development(TDD) methodology. The use of frameworks such as nUnit and Moq will be used to test the functional requirements of the project. The project will aim to have >90% code coverage.

After each unit of the application is written, unit tests will be used to see if it works as functioned.

# Documentation

The project will be documented through documents such as these like a specification. In code the plugin Stylecop will be used to thoroughly document the project.