

# Technical Design Document

## 1 Purpose

The purpose of this document is to outline the technical design of the Cycling Route Planner app and provide an overview of its implementation.

- Detail the functionality of that each component will provide.
- Provide a guideline for the app's design and development.

## 2 Design Overview

The overall goal is to create an app to assist cyclists in planning their optimal route, and to collect data about the choices that they make.

### 2.1 Design Principles

- Robust

The application should not stop working if the user does something unexpected. This includes but is not limited to uploading data when no network connection exists.

- Scalable

The app should be able to scale to encompass cities other than Dublin.

- Flexible

It should also easy integration of future requirements without affecting the existing operations. This will require a modular approach to development.

### 2.2 Design Pattern

“Model - View - Controller” Design.

- Model

The Model manages the behaviour of the application. It responds to information requests from the View and Controller. When a change is made to the Model it notifies its *Observers*.

- View

The View displays the Model as an interface. It provides a the user with a way of interacting with the Model.

- Controller

The Controller will deal with any inputs and generates a response by interacting with the Model. Sources of input include GUI, Network, System.

### **3 Model**

The Model manages the behaviour of the application. It responds to information requests from the View and Controller. When a change is made to the Model it notifies its *Observers*.

#### **3.1 Application States:**

- Idle
- Recording
- Paused
- Uploading

#### **3.2 Classes:**

- Route Class - the co-ordinates of the route
- Trip Class - the actual trip taken/underway
- PersonalisedRoute Class - the suggested route to take
- PersonalInfo Class - age, weight, gender, etc.
- Preferences - used to query the startup settings
- History Class - a list of all trips
- Statistics Class - used to store and calculate stats

### 3.3 Services:

- Recording Service
- Uploading Service

### 3.4 Model Class Design

#### 3.4.1 Route Class

A Route object contains the geometric co-ordinates of the route.

##### Constructors

```
Route()  
Route(File)
```

##### Methods

```
public addMarker(int, int, int)  
public removeMarker()  
private parseKML(File)  
private parseGPX(File)
```

##### Accessors/Modifiers

##### Members

```
private File      gpx  
private File      kml
```

#### 3.4.2 Trip Class

A Trip object contains all information associated with a single journey.

##### Constructors

```
Trip()
```

##### Methods

```
public upload()           // Sends the Trip to the server  
public finish()           // Ends the recording trip  
public update()           // Updates the Trip's variables  
private updateRoute()     // Adds the latest co-ordinates to the Route object  
private updateTime()      // Updates the elapsedTime  
private updateDistance()  // Updates the distance  
private updatePace()      // Updates the averagePace and maxPace
```

```
private updateSpeed()      // Updates the averageSpeed and maxSpeed
private compileKey()      // Creates a new Trip key
```

Accessors/Modifiers

## Members

```
private String      tripID
private String      userID
private boolean     finished
private boolean     uploaded
private Route       route

private double      distance
private float       elapsedTime
private double      averageSpeed
private double      maxSpeed
private double      averagePace
private double      maxPace

private int         iReason      // Reason for the trip, as an integer ID
private String      sReason      // Reason for the trip, as a descriptive String
private int         iType        // The perceived Type of trip, as an integerID
private String      sType        // The perceived Type of trip, as a descriptive String

private String      key
```

### 3.4.3 PersonalisedRoute Class

A PersonalisedRoute object contains the information about the suggested personalised route.

## Constructor

```
PersonalisedRoute()
PersonalisedRoute(File)
```

## Methods

Accessors/Modifiers

## Members

```
private Route      route
```

### 3.4.4 PersonalInformation Class

A PersonalInformation object contains information input by the user. It is used to calculate

things like “Calories Burned”.

### Constructor

PersonallInformation()

### Methods

private File      convertToKML(File gpx)  
private File      convertToGPX(File kml)

Accessors/Modifiers

### Members

private int	age
private double	weight
private double	height
private bit	gender
private int	proficiency

## 3.4.5 Preferences Class

A Preferences object contains the User’s preferences and the configuration when the program was last shut down.

### Constructor

Preferences()

### Methods

Accessors/Modifiers

### Members

private int	iReason	// Reason for the trip, as an integer ID
private String	sReason	// Reason for the trip, as a descriptive String
private int	iType	// The perceived Type of trip, as an integerID
private String	sType	// The perceived Type of trip, as a descriptive String

## 3.4.6 History Class

A History object is a list of recorded Trips stored in a Hash Table.

### Constructor

History()

### Methods

```
public addTrip(Trip)
public removeTrip(int)
public clearAll()
public getTrip(int)
```

Accessors/Modifiers

## Members

```
private HashTable    history
```

### 3.4.7 Statistics Class

A Statistics object contains persistent information about the user's aggregated stats.

## Constructor

```
Statistics()
```

## Methods

Accessors/Modifiers

## Members

```
private float        totalTime
private float        totalDistance
private double       averageSpeed
private double       maxSpeed
private double       averagePace
private double       maxPace
```

### 3.4.8 Recording Service

The Recording Service controls the recording of the current Trip.

## Constructor

```
Recording()
```

## Methods

```
public void    resume()
public void    pause()
public void    stop()
private void    update()
```

Accessors/Modifiers

## Members

```
private Trip    current    // The Trip that is being created
private float   timer      // Value used to determine how often data is recorded
```

### 3.4.9 Uploading Service

The Uploading Service is used to send Trip(s) to the database.

## Constructor

```
Uploading()
```

## Methods

```
public upload(Trip)           // Uploads a specific Trip
public uploadAll()            // Uploads any Trips in the History that are not already
                               uploaded
```

## Accessors/Modifiers

## Members

```
private Trip                current
private PersonalisedRoute   suggested
```

## 4 View

The View displays the Model as an interface. It provides a the user with a way of interacting with the Model.

### 4.1 Classes:

- SplashActivity - The screen displayed on startup, author & logo
- MainActivity - The main activity, has map embedded and menus
- MapView - The map API
- OverlayPlanRouteView - A custom menu on top of the MapView
- PlanRouteActivity - Setup screen for a Trip
- OverlayRecordingView - A custom menu for the starting/stopping
- EndTripActivity - Tabulated gui showing stats and finishing options
- OverlayStatsView - Shows current Trip stats at top of the screen
- PersonalisedRouteView - The overlay of the suggested route

- **StatisticsActivity** - Screen showing the aggregated user stats
- **UploadNowActivity** - Uploads all un-uploaded trips.
- **HistoryActivity** - Shows previous Trips and allows uploading
- **HelpActivity** - Help information
- **FeedbackActivity** - A form to submit feedback to the developers
- **Settings Submenu** - A submenu for configuring settings
- **About Dialogue** - Version information, etc.

## 4.2 View Class Design

### 4.2.1 Splash Activity Class

This screen is showed when the application starts. Make it brief, less than 2 seconds.

Show Title, Logo, Version, Author

### 4.2.2 Main Activity Class

The main activity is the application main screen.

#### Members

```
private MapView
private OverlayPlanRouteView
private OverlayRecordingView
private OverlayStatsView
```

### 4.2.3 MapView Class

The MapView object is the visual element of the map. It is embedded in the MainActivity and fills most of the screen.

Makes use of the Google Maps API.

### 4.2.4 OverlayPlanRouteView Class

This is a custom interface in the bottom part of the screen which enables the user to begin the route planning process.

It may be a single button saying "Plan Route".

### 4.2.5 PlanRouteActivity Class

Is a screen where the user inputs route information:

- Reason for Trip
- Desired Trip Type

Once ready the "Calculate" the desired trip.



#### 4.2.6 OverlayRecordingView Class

Once the trip is underway the user uses the OverlayRecordingView to manage it. Here they can Stop, Pause and Resume their trip.

#### 4.2.7 EndTripActivity Class

When the user signals the end of their journey they are brought to the EndTripActivity screen. This screen has 2 tabs:

- Stats: A summary of the trip's stats.
- Finish: Which facilitates uploading, saving and exiting.

#### 4.2.8 OverlayStatsView Class

This will be a toggleable interface in the top of the screen showing the current trip statistics:

- Time Elapsed
- Distance travelled
- Speed
- Pace

#### 4.2.9 PersonalisedRouteView Class

A coloured line which is drawn on the map which shows your recommended route.

#### 4.2.10 StatisticsActivity Class

The statistics screen will show the final stats for the trip that was just finished. It will display maximum values, averages, and sums.

#### 4.2.11 UploadNowActivity Class

This option will immediately upload all un-uploaded trips that are in the users history. Requires a connection to the internet. Can probably be achieved through a context menu.

#### 4.2.12 HistoryActivity Class

This screen will provide a list of all completed trips. The user can navigate through the individual trips and browse the stats of the trip. They may also choose to upload any trips that weren't previously uploaded.

#### 4.2.13 HelpActivity Class

This screen will show the user help documentation. It will explain how to use the app, its features and its academic purpose. It is accessible through the menu button.

#### 4.2.14 FeedbackActivity Class

This screen shows the user help documentation and will be accessible through the menu.

#### 4.2.15 Settings Submenu

This menu will give access to important features of the app:

- UploadNowActivity
- StatisticsActivity
- HistoryActivity
- Settings Submenu
- FeedbackActivity
- HelpActivity
- About Dialogue

#### 4.2.16 About Dialogue

A Dialogue window which simply shows app name, app logo, version information, author name, company name, website.

## 5 Controller

The Controller will deal with any inputs or events and generates a response by interacting with the Model. Sources of input include GUI, Network, System.

### 5.1 Interfaces with:

- Model
- View
- System - clock, accelerometer, GPS
- Network - Internet database

### 5.2 Classes:

- MapController
- InterfaceController
- NetworkController
- GPSController

### 5.3 Controller Class Design

#### 5.3.1 MapController Class

The MapController deals with events generated by the map.

##### Constructors

MapController()

##### Methods

// Probably methods for touch() and pinch(), etc.

##### Accessors/Modifiers

##### Members

// To be decided

#### 5.3.2 InterfaceController Class

The InterfaceController will react to listener events generated by the Interface classes. Its primary function will probably be transitioning screens.

##### Constructors

InterfaceController()

### Methods

// Not yet defined

### Members

// Not yet defined

## 5.3.3 NetworkController Class

The NetworkController will handle incoming data from the internet - database and Elevation API.

### Constructors

NetworkController()

### Methods

// Not yet defined

### Members

// Not yet defined

## 5.3.2 GPSController Class

All incoming GPS updates enter the program through the GPSController.

### Constructors

GPSController()

### Methods

// Not yet defined

### Members

// Not yet defined

## 6 References

Google Code Site: <http://code.google.com/p/routeplanner/>

Code Repository: <http://code.google.com/p/routeplanner/source/browse/>

Design Document: <http://routeplanner.googlecode.com/files/Design%20Document.pdf>