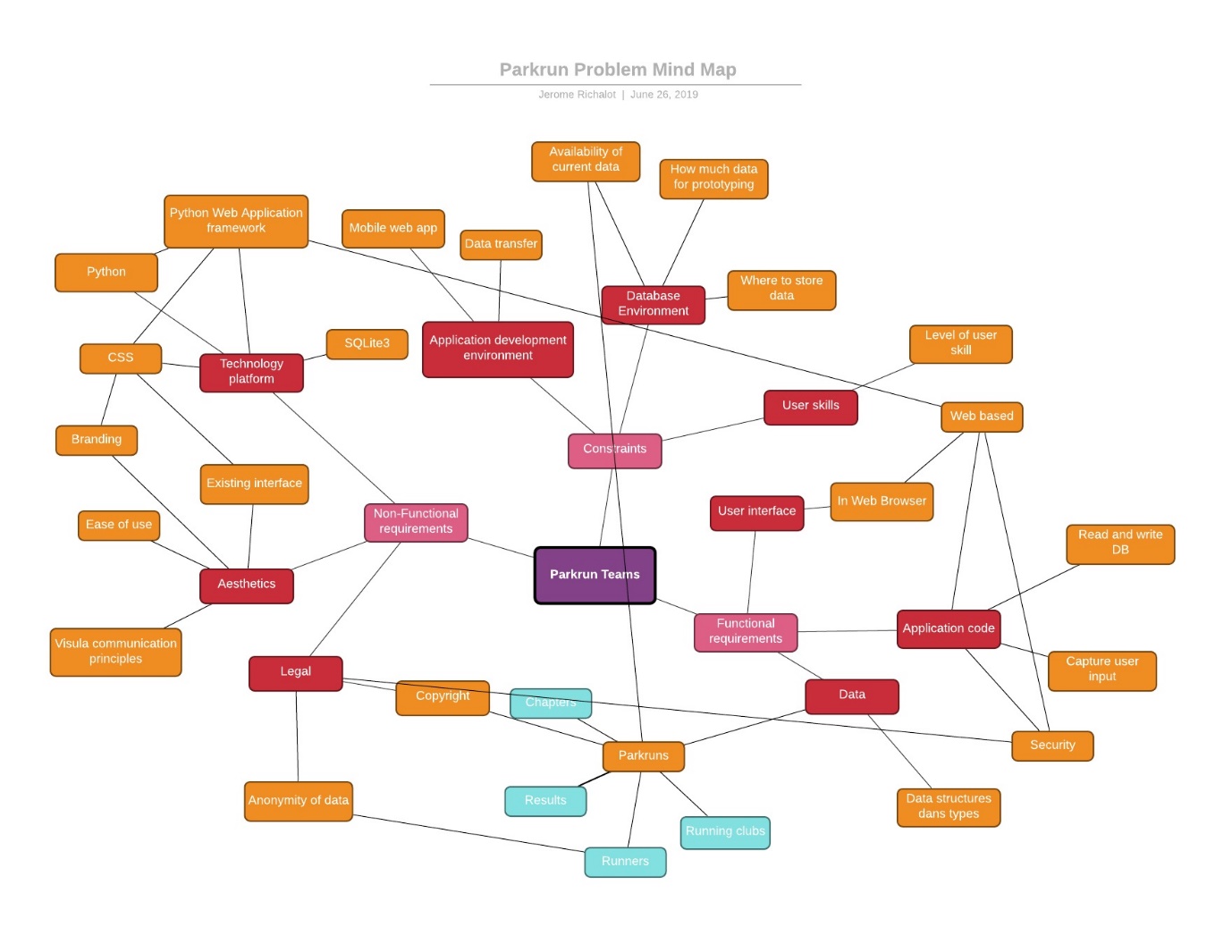
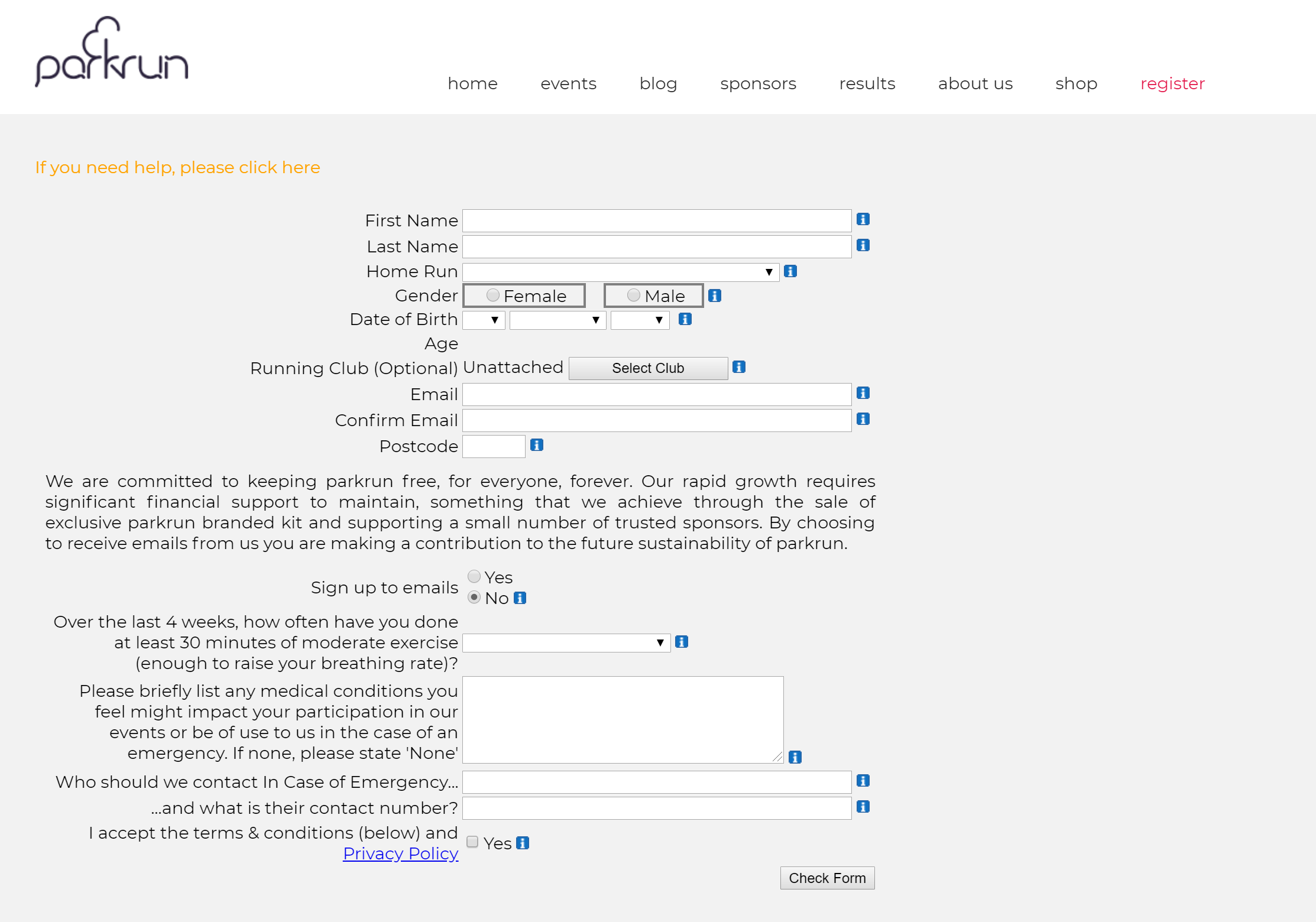
Exploring parkrun



# The parkrun registration page

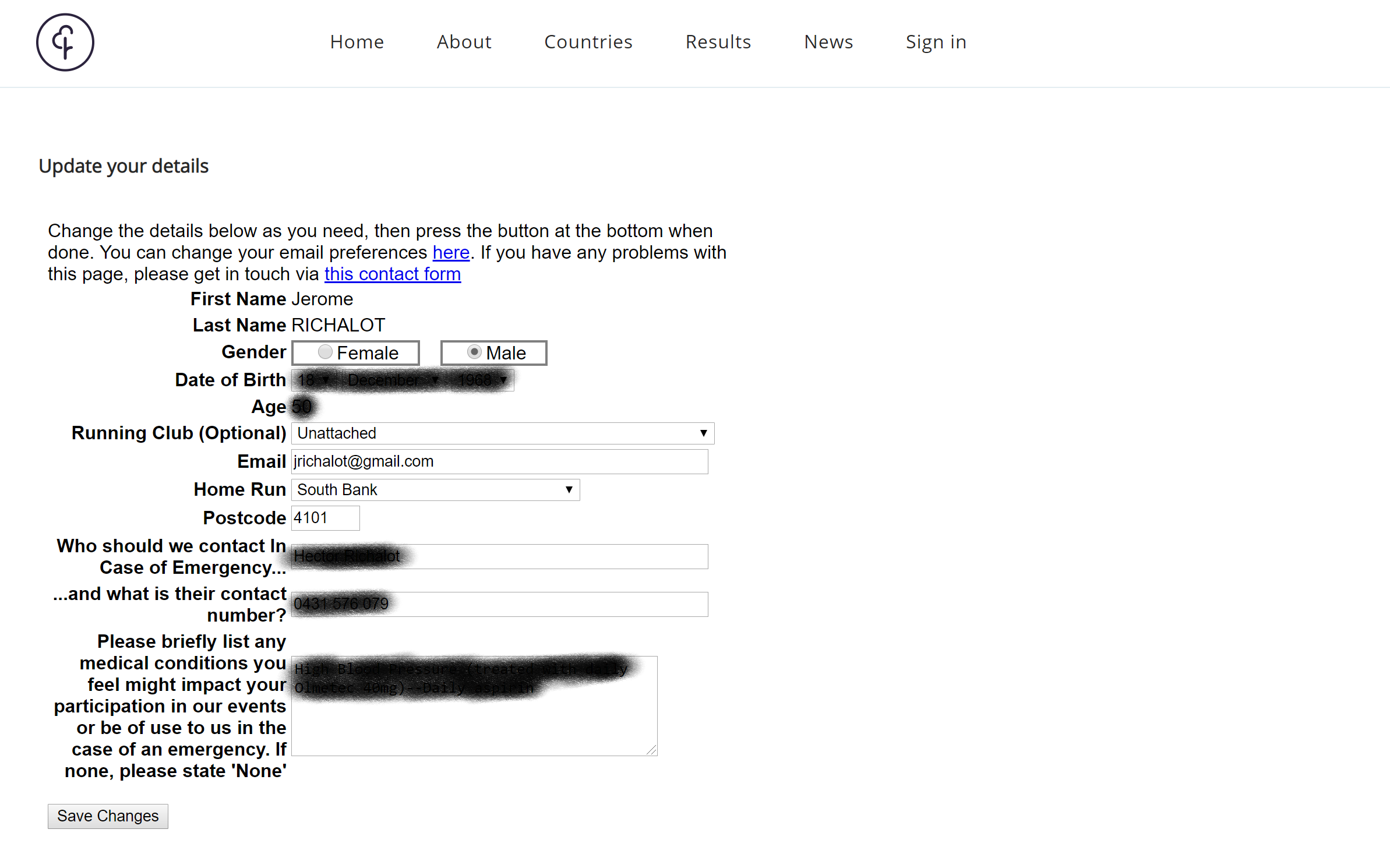


## Observations on the registration page:

* First name and last name, email and confirm email but no postal address
* Home run to be selected from an existing list of parkrun chapters (compulsory)
* Running Club to be selected from an existing list of running clubs (optional), default is “unattached”
* Recent exercise to be selected from an existing list of recent exercise frequency

## Notes:

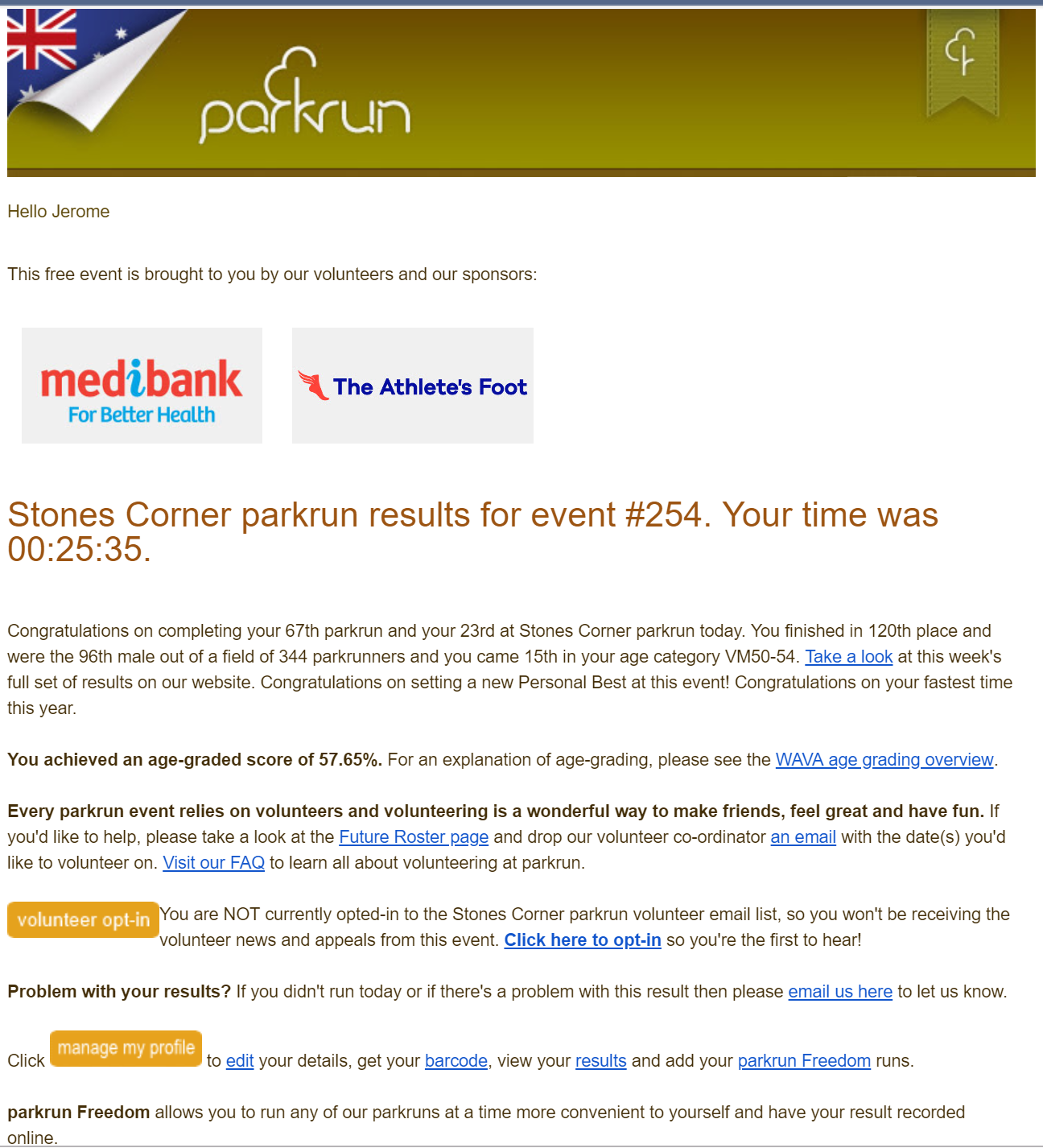
* The fields on the registration page will mostly be the ones in the Runner table
* This suggests there needs to be a Chapters table. More information will need to be gathered from exploring a Chapter page.
* This suggests there needs to be a Running Club table, “unattached” will be the default Running Club
* What is the difference between a Running Club and a Team?
* This suggests there needs to be a Recent exercise frequency table

The Parkrun runner profile page

## Observations on the registration page:

* Some fields are not displayed in the profile
  + Confirm email
  + Sign up to emails
  + Accept Terms and conditions

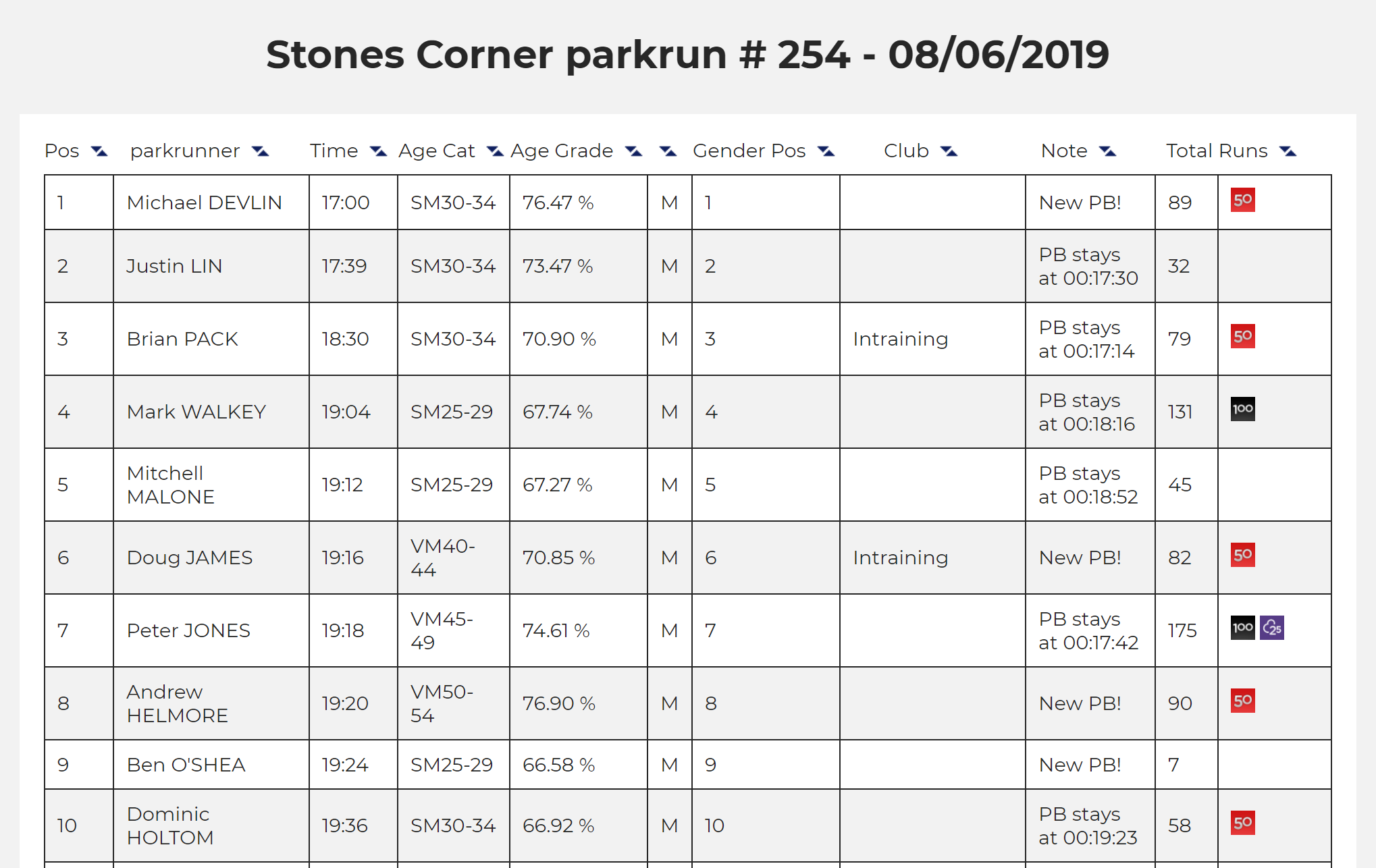
# The result email



## Observations on the result email

* Individual results are given in minutes and seconds per Chapter per event
* The overall number of events completed by a runner is tallied for all events and for events at a particular chapter
* A result for a particular event is a runner, an event, a time, an overall position, an age category position, an age-grade score (percentage)
* The email reveals the existence of a Roster for volunteers
* The email provides a link to the runners past results and statistics

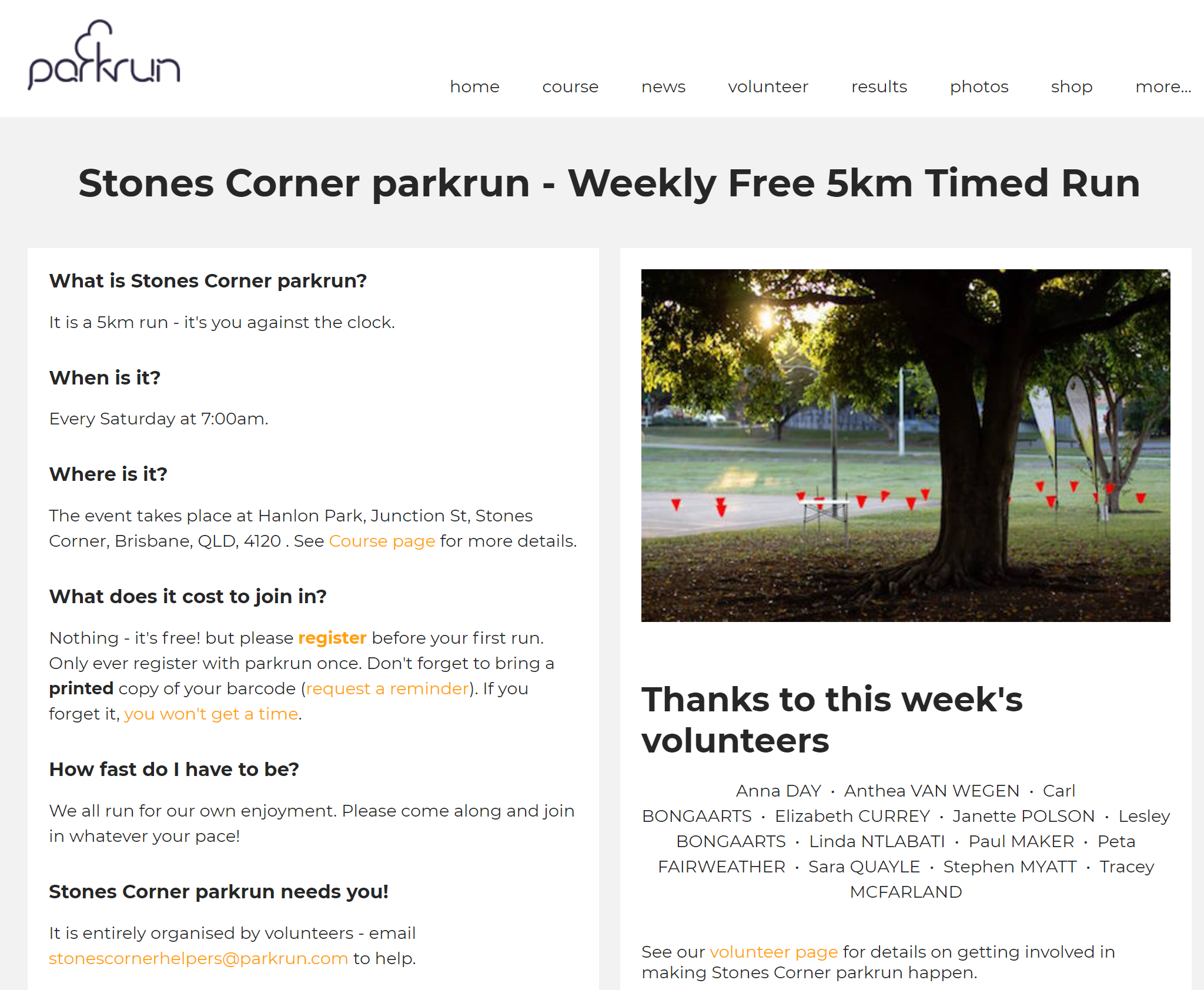
# The event results (sample)



## Observation on the event results

* Confirms observation of individual results
* Shows parkrun 50/100 club badges and 25V badge
* An event does not exist without results as there is no pre-registration. Taking part in an event is effectively recording a time for that event. If no-one turned up for an event, the event would not exist. So the event page of the results’ page are effectively the same thing

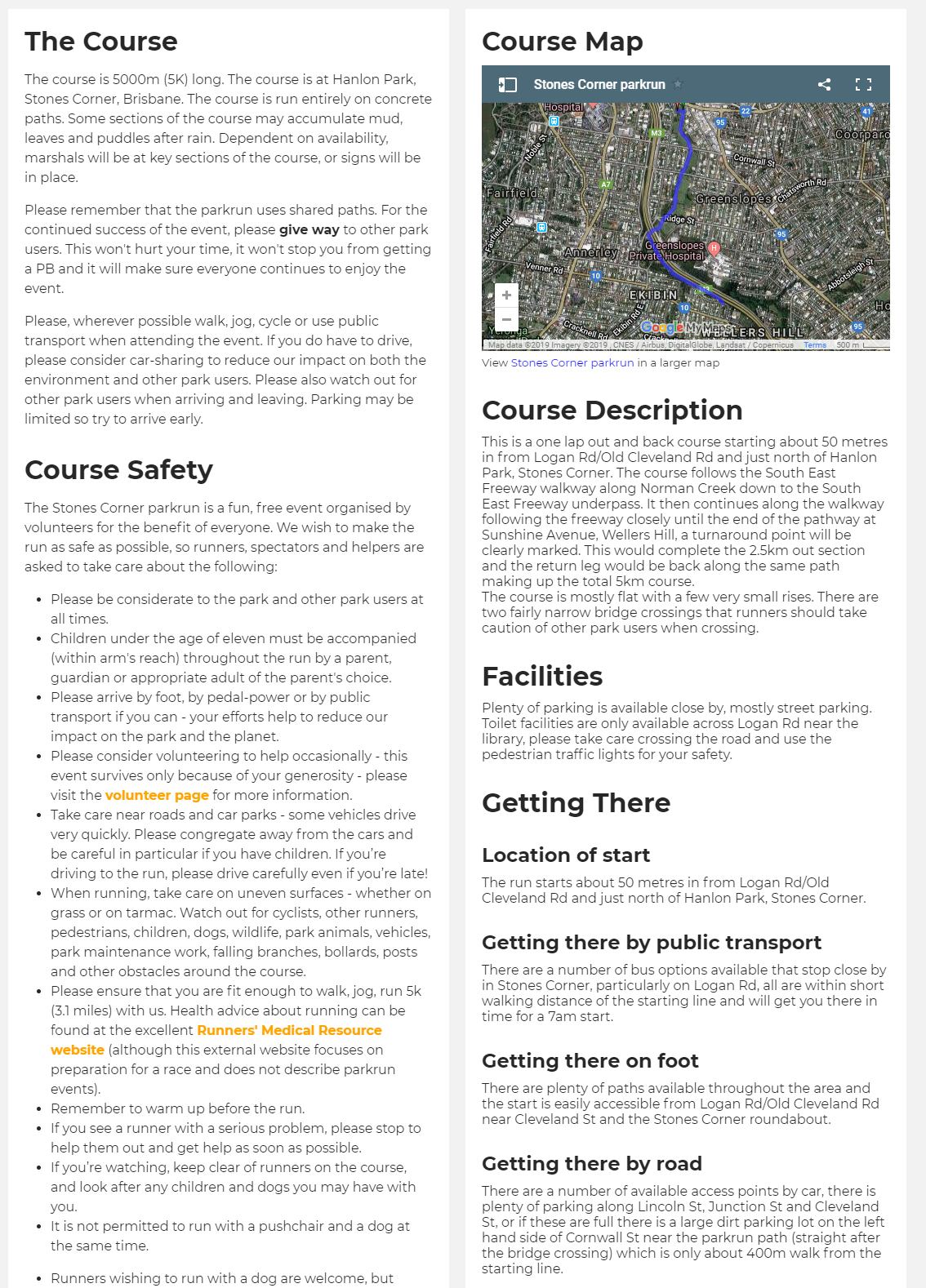
# The Chapter page



## Notes on the Chapter page:

* A Chapter has a name, an address, a photo, an email address and a course

# The Course page



## Notes on the course page:

* Some information on the course page is generic and does not need to be stored in the database. e.g. Course safety
* A course has a map, a description, facilities, a location of start, a getting there by public transport, getting there on foot and getting there by road
* Since a chapter has one course and one course can only be run by a chapter, the course information can be in the Chapter table.

# Suggested tables, fields and fields parameters at this stage

### Runners[[1]](#footnote-1)

* RunnerID INTEGER PRIMARY KEY AUTOINCREMENT
* RegistrationDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP
* FirstName TEXT NOT NULL
* LastName TEXT NOT NULL
* HomeRun INTEGER NOT NULL
* Gender INTEGER[[2]](#footnote-2) NOT NULL
* RunningClubID INTEGER
* Email[[3]](#footnote-3) TEXT NOT NULL
* Postcode TEXT NOT NULL
* SignUpToEmails INTEGER DEFAULT 1
* RecentExerciseFrequency INTEGER
* MedicalCondition BLOB
* EmergencyContactName TEXT NOT NULL
* EmergencyContactNumber[[4]](#footnote-4) TEXT NOT NULL

### Chapters

* ChapterID INTEGER PRIMARY KEY AUTOINCREMENT
* ChapterName TEXT NOT NULL
* ChapterAddress TEXT NOT NULL
* ChapterPhoto TEXT
* ChapterEmail TEXT NOT NULL
* CourseMap TEXT
* CourseDescription BLOB
* Facilities BLOB
* LocationOfStart TEXT NOT NULL
* GettingTherePublicTransport BLOB
* GettingThereFoot BLOB
* GettingThereRoad BLOB

### Running Clubs

* RunningClubID INTEGER PRIMARY KEY AUTOINCREMENT
* RunningClubName TEXT NOT NULL

### Recent exercise frequency

* RecentExerciseFrequencyID INTEGER PRIMARY KEY AUTOINCREMENT
* RecentExerciseDescription TEXT NOT NULL

### Events

* EventID INTEGER PRIMARY KEY AUTOINCREMENT
* EventDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP
* RunnerID INTEGER NOT NULL
* RunnerTime INTEGER NOT NULL
* ChapterID INTEGER NOT NULL

**Events table revisited upon importing results:**

An event does not exist without results. This would mean the events table above is the normalised and correct way to go. However, it seems more readable to separate events from results and have an event table and a results table as follows:

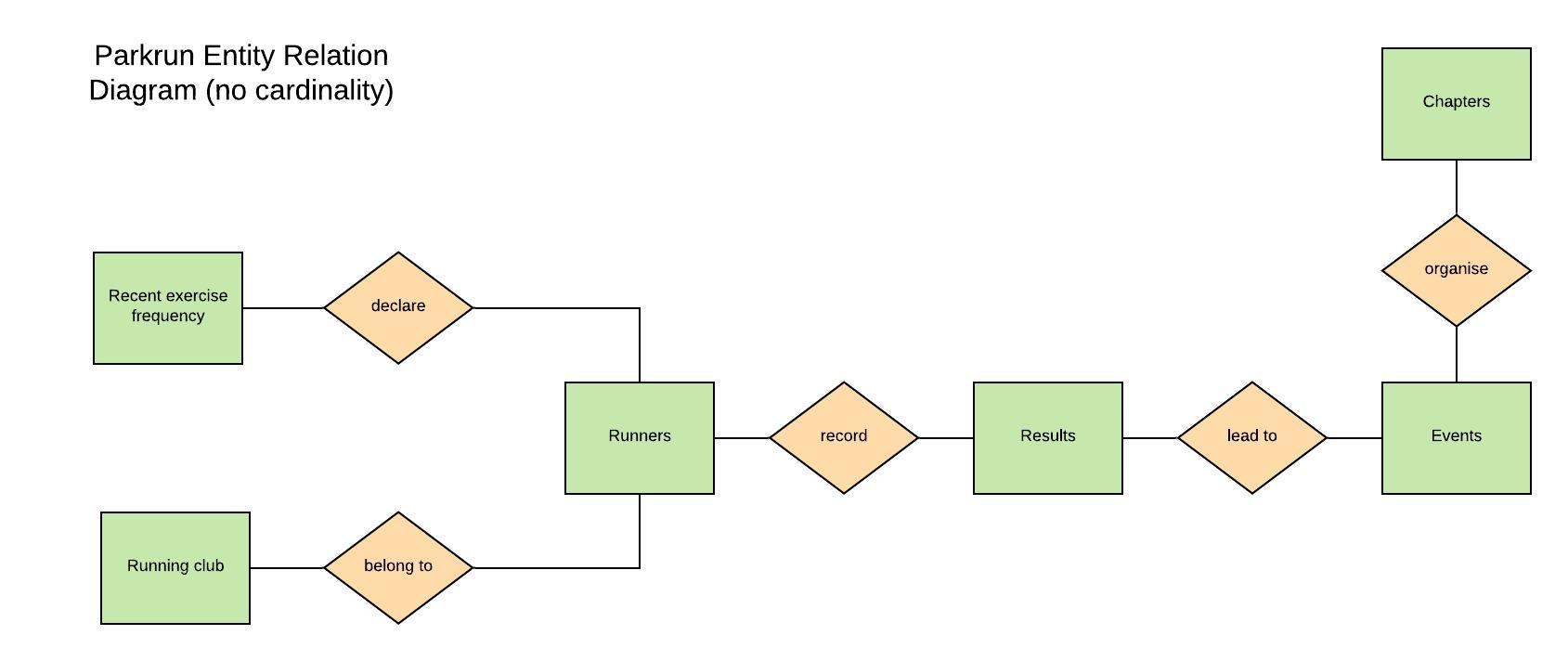
### Events

* EventID INTEGER PRIMARY KEY AUTOINCREMENT
* EventDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP
* ChapterID INTEGER NOT NULL

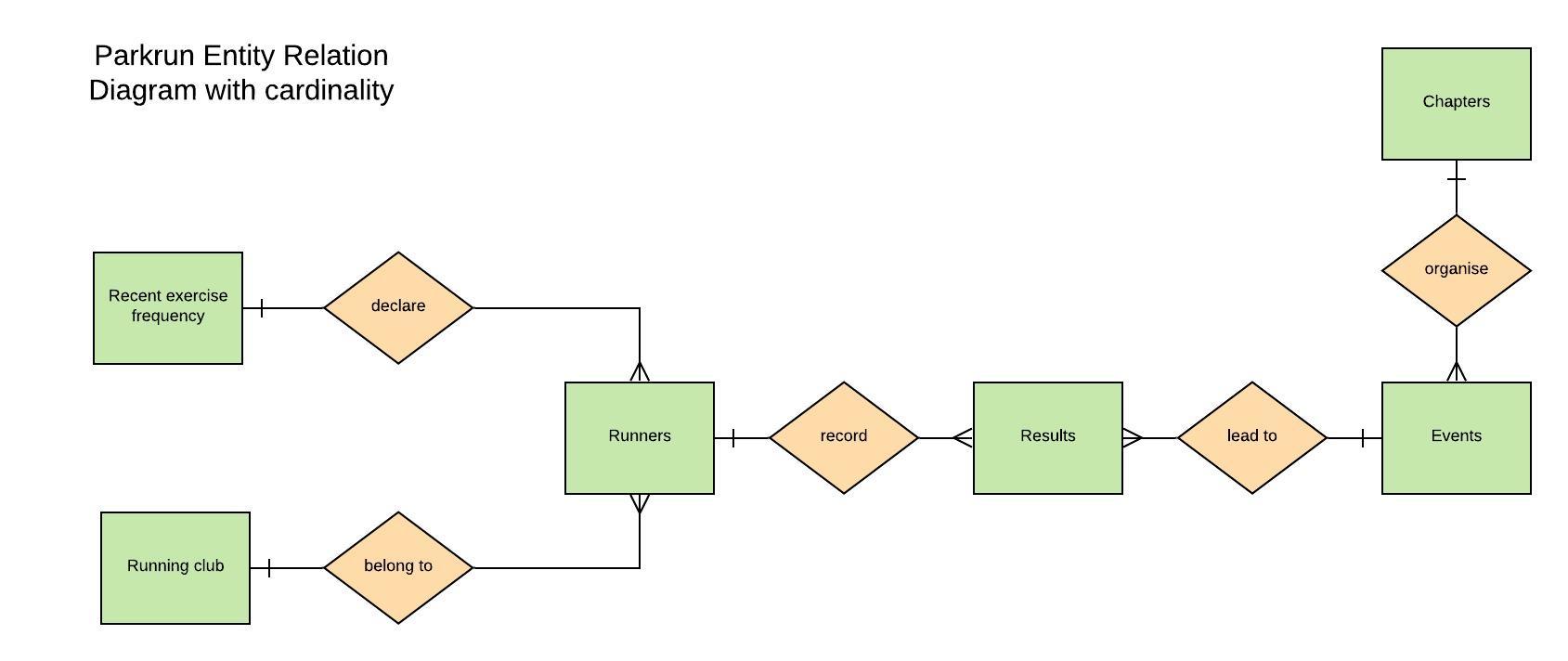
### Results

* ResultID INTEGER PRIMARY KEY AUTOINCREMENT
* RunnerID INTEGER NOT NULL
* RunnerTime INTEGER NOT NULL
* EventIDate INTEGER

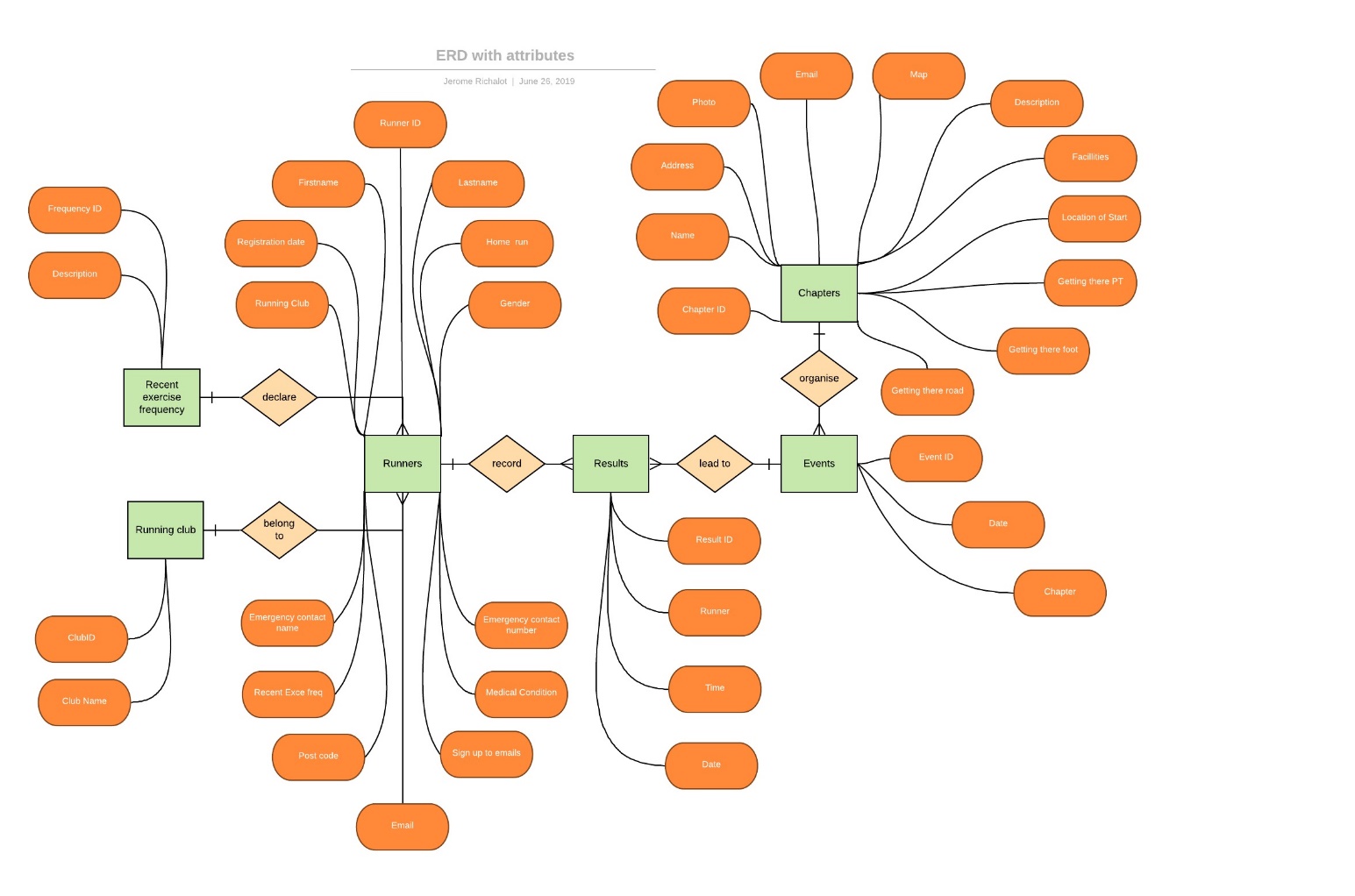
Proposed Entity relationships diagram



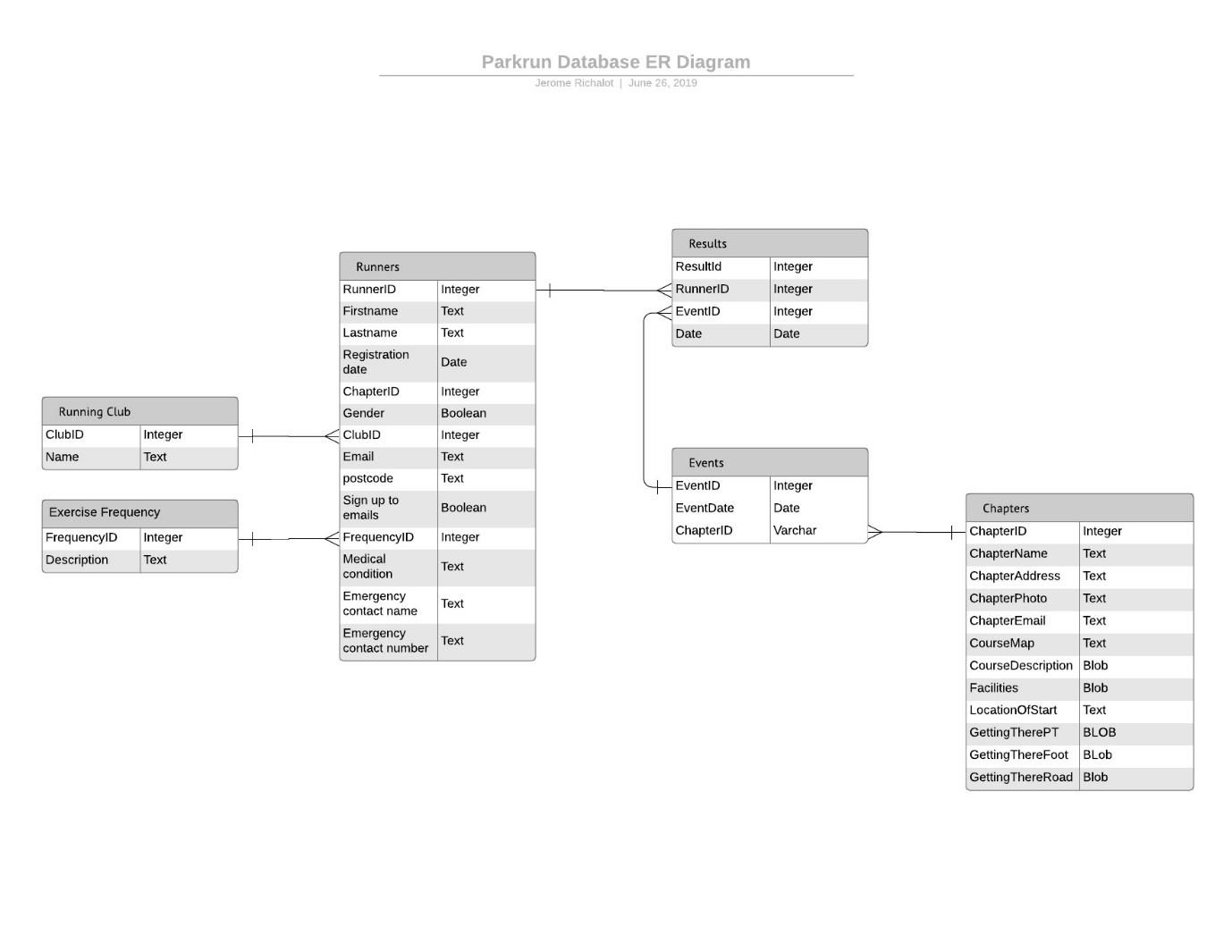
TASK: Add cardinality to the ERD above



TASK: Add attributes to the ERD



Based on the ERD above develop a Database ERD:



1. Start thinking a bout the syntax of the SQL CREATE Table when you format your list to make your work easier later in the process [↑](#footnote-ref-1)
2. As per <https://www.sqlite.org/datatype3.html> 2.1 “SQLite does not have a separate Boolean storage class. Instead, Boolean values are stored as integers 0 (false) and 1 (true).” [↑](#footnote-ref-2)
3. The confirm email “field” does not need to be stored in the database. It will be tested against the email before committing data to the database [↑](#footnote-ref-3)
4. Note the phone number is stored as text. Numbers in a phone number have no numerical value as such. Storing them as text allows for easier formatting and potentially checking. [↑](#footnote-ref-4)