Contents

Overview

Tribe is an application for efficient families.

Each member of the family uses Tribe. Family members complete daily tasks to earn points which unlock rewards.

The Problem Tribe solves

- Kids need motivation to do homework and do tasks like cleaning their bedrooms
- Families are busy and find it hard to keep track of what other family members are doing and when
- Family members spend their days in different locations

Aims of Tribe

Motivation Create motivation for family members to work together on completing day-to-day tasks.

Schoolwork Remind and encourage kids to complete homework.

Accountability Allow parents to check that kids' homework has been done. Make sure everybody knows the dog has already been walked and fed.

Family Interaction Encourage the family to spend time together and co-operate.

Fun Give a sense of satisfaction and joint achievement and add a fun element to day-to-day tasks.

Requirements spec

Overview of the application

Each member of the family who is signed up to Tribe becomes a "tribe member".

A parent, (or both parents, or even an older child or child minder) becomes the "tribe leader".

Tasks

Tribe leaders create tasks.

Tasks can be once-off or recurring, eg. daily tasks ("do homework") or weekly tasks ("take out bins").

Tribe members complete tasks, and mark them as "done" using either the web interface or mobile app. Optionally a photo can be uploaded as proof that the task has been completed.

Points

A Tribe Leader confirms that the task has been done and the points are awarded to the Tribe as a whole.

The accumulated points are visible to all tribe members.

Rewards

If the tribe scores enough points in a given week, rewards are unlocked. Rewards are in "levels" eg., the Level 1 rewards might be unlocked after 100 points, Level 2 after 200, and so on.

Rewards are set by a tribe leader, and are specific to tribe members. For example, for parents the Level 1 reward might be a bottle of wine or cinema trip. For the older kids, the equivalent reward might be pocket money or phone top-up. For the youngest kid, it might be sweets. It is up to the tribe leader to ensure the rewards are roughly equal in value. All tribe members can view what rewards (for both themselves and other tribe members) are available for each level on a given week.

Tribe has a selection of default rewards to choose from, however the Tribe leaders have the option to customise the rewards.

Sample rewards for kids are

- amounts of pocket money (a multiplier system is available to award appropriate levels of pocket money based on the kids' ages)
- phone top-up credit
- presents (eg. a toy, new phone)

Categories

Tasks can fit into a category.

Information about how many points have been scored in a certain category are available, giving the Tribe an idea of where they are doing well and where improvements are needed.

Tribe provides several categories by default:

- Household
- Diet
- Fitness
- Pets
- Learning and School

User Stories

Roles

Tribe Leader: This is usually a parent but can be any tribe member (eg. babysitter)

Tribe Member: Any member of the family/tribe who has signed up

---

As a new user I want to sign up to Tribe because I want to get my family to do some work.

The user enters

- email
- family name
- password

As a tribe leader I want to add my family members to the tribe because I want to have my whole family set up using Tribe

Data entered

- name
- email
- password

As a tribe member I want to sign in so I can use features of Tribe

User needs to enter

- email
- password

As a tribe member I want to edit my profile because I want to add details about myself and choose a picture.

- name
- picture (upload or choose an icon)

As a tribe leader I want to add a task because I have thought of a new task that needs doing.

The tribe leader enters

- task name
- description
- task category
- location
- recurring strategy (including once-off)
- number of points rewarded for completing the task

As a tribe leader I want to edit a task which has already been created because an element of the task has changed or was entered incorrectly.

All elements of a task are editable via a form.

As a tribe leader I want to delete a task because I've decided this task should no longer be in the system or it doesn't need doing anymore.

The user should be asked to confirm they want to delete the task completely.

As a tribe member I want to see the number of points my tribe has accumulated this week because I want to see how close we are to winning a reward.

The points are displayed on a graph showing how close tribe is to the next reward and what the reward is for each family member.

As a tribe member I want to see a graph of how many points the tribe has accumulated on a per-week and category basis so far because I want to see if we are making an improvement or disprovement

Points are displayed on a graph showing weekly totals. Task categories can be toggled on and off.

As a tribe leader I want to create, edit or delete a reward which can be won by tribe members because I want to select this as a reward for a tribe member at a points level

Details entered are

- Reward name
- Icon image (upload or select from a selection)
- Optional description

As a tribe leader I want to set the rewards for a given week for each level because I want to decide what each tribe member will receive when the points are achieved.

The rewards for each tribe member at a specific level are entered.

As a tribe member I want to mark a task as done because I have completed a task and want to earn points for it.

This means the task has been done in real life and is ready for checking by a tribe leader.

As a tribe leader I want to create a new task category or edit existing categories because I want points to control how points are tracked.

- Category name
- Optional description
- Colour
- Icon (upload or select)

As a tribe member I want to view the current rewards levels because I want to see how many points are needed for each.

As a tribe leader I want to edit the current reward levels so that they are appropriate to the level of points the family can earn.

Each level required points are editable.

As a tribe leader I want to create or edit an achievement to encourage the family to win more points.

Data entered:

- Achievement name
- Task(s) related to the achievement
- Pattern required to achieve (eg. "10 in a row")
- Points awarded when this achievement is earned

As a tribe member I want to view my or another tribe member's profile because I want to see how they are doing and see what achievements they have gained

Profile page shows

- Tribe member's name
- Stats on points and tasks
- Achievements gained

As a tribe leader I want to see what rewards are owed to a tribe member in real life and mark them as complete once they have been awarded in real life

The rewards which are owed to which tribe member are visible and marked as pending or complete (meaning the reward has been physically given to the tribe member).

REST API

...

User Manual

Web interface

Mobile App

REST API

Technical Report

Source Control

Agile

Taiga.io

Testing

My tests

...

Continuous Integration

Travis CI was used.

https://travis-ci.com/

Travis CI is a web service which links to a Github account to provide continuous integration. Travis CI is notified when your Github repository is updated and then clones the repository and runs the instructions in the .travis.yml file to download the required dependencies and run the tests for your application.

I followed the following steps to set up Travis CI and activate continuous integration with my project's repository.

I created a .travis.yml file in my repository's base directory. The travis.yml had the following content:

I set the language to Python so Travis knows which interpreter to use.

```
language: python
```

Tell Travis the Python version. This is very important as between Python versions especially version 2 to 3 code may become invalid.

```
python:
  - "3.4"
```

The following tells Travis to install the required libraries for my project. The requirements.txt file is in the base folder of the repository and is simply a list of my required libraries and their versions.

```
install:
  - pip install -r tribe/requirements.txt
```

The next section tells Travis what it has to do to run my tests. Each line of this represents a command to be run.

```
script:
  - cd tribe
  - python manage.py test
```

Signing up for a Travis account was quite straightforward. It is important to note that for my purposes travis-ci.com was used and not travis-ci.org, because travis-ci.org only allows for public accounts and for academic purposes I needed to keep my builds private.

I signed up for the free trial using my Github account. This was a one-click operation. I was then able to navigate to the billing settings and notify Travis that I had a Github student account. Travis are part of this plan so I was then allowed free access to a private account.

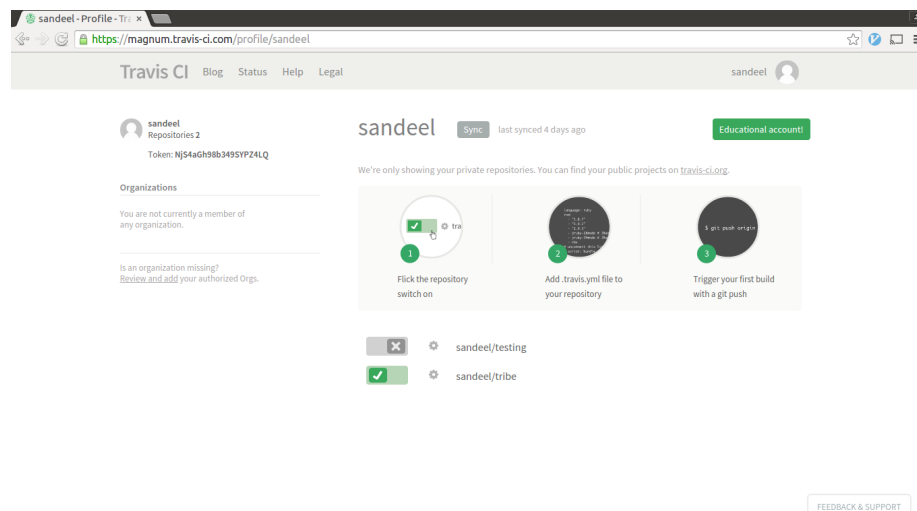I was then able to switch on or off repositories from my Github account.

Figure 1: Activating repositories in Travis-CI

Travis-CI helpfully then sent me emails stating when my build had failed and when it was fixed.

The build can be watched in real-time if you log into your Travis account and then push to your Github repository.
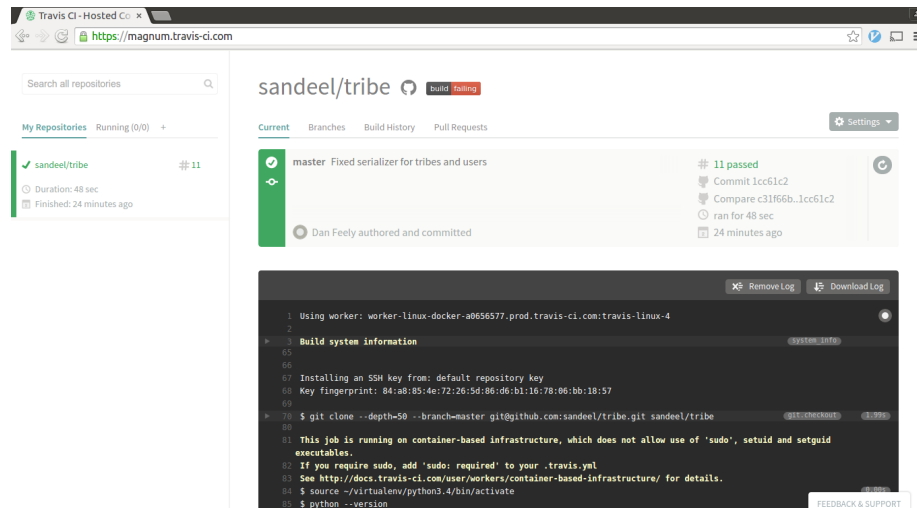

Figure 2: Real-time build on Travis server

There are also various tools available on Travis's web interface. One of the more useful views I found was the "Build History" view.

I was also able to put a "light" on my repository's Github page which turned green when build was passing and red when build failing.

Overall I think that while using continuous integration didn't gain me a huge amount because I was only a one-person team, it still had some benefits and didn't take too much time to set up. It was also of benefit to learn this skill as I can see how continuous integration would be extremely useful in large teams who are working together on a single codebase.

Deployment

—– why was do chosen

I needed to install git to clone my repository. This was done using Ubuntu's apt-get package manager:

```
apt-get update
apt-get install git
```
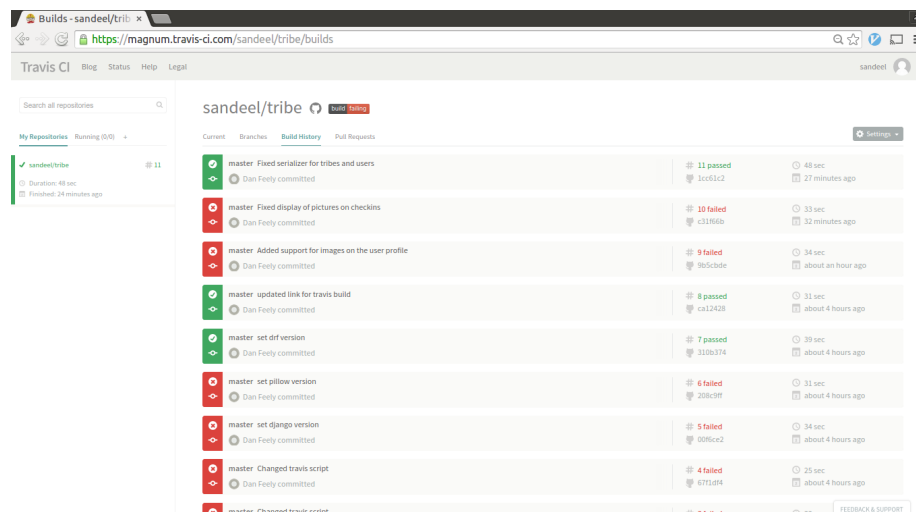
I then cloned my repository:

Figure 3: Travis-CI build history for my project

```
git clone https://github.com/sandeel/tribe.git
```

And changed into the directory for the Django project

```
cd tribe/tribe
```

To install the python libraries needed, I first had to install Python's package manager, PIP.

```
apt-get install [[python-pip]]
```

And some Python development libraries:

```
apt-get install python-dev
```

The requirements are stored in a text file called requirements.txt. PIP can parse through this file and install all the dependencies:

```
pip install -r requirements.txt
```

Then to run the development server for testing:

```
python manage.py runserver 0.0.0.0:8000
```

11

Timeline

2014-12-08
Came up with a name for the application: tribe * Created the git repo locally and on Github *
Signed up to Taiga.io * Created a project for tribe on Taiga.io * Created some user stories on
Taiga

2014-12-12 Created a powerpoint for the mid point presentation Put together a very rough demo
of the application using Django

2014-12-13 Presented the idea and received feedback from Eamonn

2014-12-14 Worked on the requirement spec Submitted the requirement spec on Moodle Emailed
Mikhail requirement spec requesting feedback Started putting all user stories into Taiga started
creating a text file with all my entities worked more on the entities

2014-12-16 coding

2014-12-17 Coding researching Django REST Framework

2014-12-18 * started tutorial for Django Rest Framework. Completed this section * discovered
httpie

Sprint 1

2014-12-28 * mapped out 10 sprints on taiga.io leading up to project completion date

2014-12-29 * continued tutorial on Django Rest Framework * updated some project documenta-
tion

2014-12-30 * continued tutorial on Django Rest Framework * Discovered MkDocs, which I may
use for the documentation of the project * This page of DRF tutorial will be useful for user API
auth * completed tutorial on DRF

2014-12-31 * Several hours working on permissions in the Django Rest Framework * Created
permission for allowing unregistered users to create a new user (eg. register via mobile app)

2015-01-01 * Almost completed sign up by API

2015-01-02 * Completed sign up via API and also implemented creating a tribe via the API

2015-01-04 * Started to implement inviting new users.

2015-01-10 * Working for few hours on adding new users to your tribe via the edit tribe page.
Almost ready for test. * uploaded December diary to Moodle

2015-01-11 * Finished form for adding a new tribe member to your tribe * Wrote unit test for
above functionality

Sprint 2 (12th Jan 2015 - 25th Jan 2015)

2015-01-12 * Added models for TaskTemplates and Categories. * Added a view for viewing Task templates

2015-01-14 * Working on the models/database tables for TaskTemplates * Form for new Task-Template

2015-01-14 * Forms for editing TaskTemplates and viewing details.

2015-01-20 * Added models, forms, views etc. for Categories * Added some more user stories to Taiga.io project * Got most of the user stories laid out in sprints in Taiga.io

2015-01-21 * Created a serializer for Categories * Created API endpoint for creating and viewing categories * Wrote tests for creating Categories via API

2015-01-22 * Created a serializer for Task * Created API endpoints for task. * Need to write tests for this next

2015-01-25 * Today I started refactoring the code so that the forms use the app's own API

Sprint 3 (26th Jan 2015 - 08th Feb 2015)

2015-01-26 * Created CheckIn class. There will now be no TaskTemplate, instead the Task will be a monolithic object which has available times and a CheckIn will refer to an instance of marking a task as done. This is to give more flexibility to different types of tasks eg. family dinners.

2015-01-28 * Added a "Check In" button to tasks which creates an instance of a check in for a user on that task and awards points. This uses the API in the background.

2015-01-31 * Created an approvals system for tribe leaders to approve tasks * Started working on some test data

2015-02-01 * Wrote lots of tests today

2015-02-02 * wrote more tests * Investigating and reading about Apache Cordova * Started working on the mobile app. * Using ngrok to publicly host the site for the app to communicate with.

2015-02-03 * Working more on the mobile app and working on styling to ensure site looks ok on both mobile and desktop * Added basic points calculation for users (done by counting up the points awarded for each checkin which has been approved)

2015-02-04 * Redesigned home page and user interface (prototype design for finished product). Started CSS style sheet for site. * Changed permissions for some views eg. Task list.

2015-02-05 * Investigated continuous integration service Travis CI. * Created account with travis and config file to tell it how to test my code * Builds passing. Added small button to my project's github page to indicated if builds are passing or not. * Added a collapsible navigation sidebar to the site. Ensured it also works on mobile

Sprint 4 "Points"

10th February - 22nd February 2015

2015-02-17 * Created calculation methods for total points for a Tribe. (still need to write tests for these) * Created a display for these points on a per-day basis on the points page

Sprint 5 "Rewards"

2015-02-24 * Created a model for a reward * Created serializer for a reward * Created views for adding rewards

2015-02-26 * Created views for editing rewards * Researching drawing graphs in HTML5 * Researching the native Progress element in HTML5 * Added a basic progress bar for weekly points using bootstrap for the styles

2015-03-10 * Revamped the UI

2015-03-11 * Working on rewards and points required for rewards * rewards can now be assigned to users

2015-03-14 * Changes to the user model. New users now need a Name * Added tests to ensure new users have names and get a default tribe created * Researched testing JSON responses * Started to create a test family which gets created by a script (for test data) * Worked a lot on created test data. To do this I had to make tweaks to some models * Redesigning the tasks page

2015-03-18 * Researching Django Crispy Forms * Designed the login page

2015-03-19 * Working on the points page. Now shows the rewards and a users progress towards rewards. * Points page now shows points today, this week, and total points. * Still working on the script for making fake data. Much more useful now as makes use of a Python fake data generator which creates random users, tasks and rewards.

2015-03-20 * Working on the user's profile page. * Changes to form for editing user.

2015-03-22 * Working on progress bars for rewards on the points page.

2015-03-23 * More work on the models for Rewards and Achieved rewards. * Points page now shows complete when reward achieved. * Achieved rewards now show on the user's homepage * Changes to the test data generator

2015-04-16 * uploaded showcase information

2015-04-18 * Worked on getting tests working with Travis-CI * Added user photo support * Redesigned profile page to allow for photos

2015-04-25 * Styled the sign-up page * Added to the script for creating test data * Changes to Tribe management

2015-05-02 * Changed some styling * Changes to the points page

2015-05-03 *

2015-05-05 * Redesigned the tasks page * Split tasks out into categories * Implemented bootstrap accordion

2015-05-06 * Redesigning a lot of the interface * Modifying the stylesheets to add colour